

this document downloaded from

vulcanhammer.net

Since 1997, your complete
online resource for
information geotechnical
engineering and deep
foundations:

The Wave Equation Page for
Piling

*Online books on all aspects of
soil mechanics, foundations and
marine construction*

Free general engineering and
geotechnical software

And much more...

Terms and Conditions of Use:

All of the information, data and computer software ("information") presented on this web site is for general information only. While every effort will be made to insure its accuracy, this information should not be used or relied on for any specific application without independent, competent professional examination and verification of its accuracy, suitability and applicability by a licensed professional. Anyone making use of this information does so at his or her own risk and assumes any and all liability resulting from such use. The entire risk as to quality or usability of the information contained within is with the reader. In no event will this web page or webmaster be held liable, nor does this web page or its webmaster provide insurance against liability, for any damages including lost profits, lost savings or any other incidental or consequential damages arising from the use or inability to use the information contained within.

This site is not an official site of Prentice-Hall, Pile Buck, the University of Tennessee at Chattanooga, or Vulcan Foundation Equipment. All references to sources of software, equipment, parts, service or repairs do not constitute an endorsement.

**Visit our
companion site**

<http://www.vulcanhammer.org>



INTRODUCTION to FINITE ELEMENT METHODS

CARLOS A. FELIPPA

Department of Aerospace Engineering Sciences
and Center for Aerospace Structures
University of Colorado
Boulder, Colorado 80309-0429, USA

Last updated Fall 2004

Material assembled from Lecture Notes for the course
Introduction to Finite Elements Methods (ASEN 5007)
offered from 1986 to date at the Aerospace Engineering
Sciences Department of the University of Colorado at Boulder.

Preface

This textbook presents an Introduction to the computer-based simulation of linear structures by the Finite Element Method (FEM). It assembles the “converged” lecture notes of **Introduction to Finite Element Methods** or IFEM. This is a core graduate course offered in the Department of Aerospace Engineering Sciences of the University of Colorado at Boulder.

IFEM was first taught on the Fall Semester 1986 and has been repeated every year since. It is taken by both first-year graduate students as part of their M.S. or M.E. requirements, and by senior undergraduates as technical elective. Selected material in Chapters 1 through 3 is used to teach a two-week introduction of Matrix Structural Analysis and Finite Element concepts to junior undergraduate students who are taking their first Mechanics of Materials course.

Prerequisites for the graduate-level course are multivariate calculus, linear algebra, a basic knowledge of structural mechanics at the Mechanics of Materials level, and some familiarity with programming concepts learnt in undergraduate courses.

The course originally used Fortran 77 as computer implementation language. This has been gradually changed to *Mathematica* since 1995. The changeover is now complete. No prior knowledge of *Mathematica* is required because that language, unlike Fortran or similar low-level programming languages, can be picked up while “going along.” Inasmuch as *Mathematica* supports both symbolic and numeric computation, as well as direct use of visualization tools, the use of the language is interspersed throughout the book.

Book Objectives

“In science there is only physics; all the rest is stamp collecting” (Lord Kelvin). The quote reflects the values of the mid-XIX century. Even now, at the dawn of the XXIth, progress and prestige in the natural sciences favors fundamental knowledge. By contrast, engineering knowledge consists of three components:¹

1. Conceptual knowledge: understanding the framework of the physical world.
2. Operational knowledge: methods and strategies for formulating, analyzing and solving problems, or “which buttons to push.”
3. Integral knowledge: the synthesis of conceptual and operational knowledge for technology development.

The language that connects conceptual and operational knowledge is mathematics, and in particular the use of mathematical models. Most engineering programs in the USA correctly emphasize both conceptual and operational components. They differ, however, in how well the two are integrated. The most successful curricula are those that address the tendency to “horizontal disconnection” that bedevils engineering students suddenly exposed to a vast array of subjects.

Integral knowledge is unique to the engineering profession. Synthesis ability is a personal attribute that cannot be coerced, only encouraged and cultivated, the same as the best music programs do not

¹ Extracted from: B. M. Argrow, Pro-active teaching and learning in the Aerospace Engineering Sciences Curriculum 2000, internal report, University of Colorado, February 2001.

automatically produce Mozarts. Studies indicate no correlation between good engineers and good students.² The best that can be done is to provide an adequate (and integrated) base of conceptual and operational knowledge to potentially good engineers.

Where does the Finite Element Method (FEM) fit in this framework?

FEM was developed initially, and prospered, as a computer-based simulation method for the analysis of aerospace structures. Then it found its way into both design and analysis of complex structural systems, not only in Aerospace but in Civil and Mechanical Engineering. In the late 1960s it expanded to the simulation of non-structural problems in fluids, thermomechanics and electromagnetics. This “Physical FEM” is an *operational tool*, which fits primarily the operational knowledge component of engineering, and draws from the mathematical models of the real world. It is the form emphasized in the first part of this book.

The success of FEM as a general-purpose simulation method attracted attention in the 1970s from two quarters beyond engineering: mathematicians and software entrepreneurs. The world of FEM eventually split into applications, mathematics, and commercial software products. The former two are largely housed in the comfortable obscurity of academia. There is little cross-talk between these communities. They have different perspectives. They have separate constituencies, conferences and publication media, which slows down technology transfer. As of this writing, the three-way split seems likely to continue, as long as there is no incentive to do otherwise.

This book aims to keep a presentation balance: the physical and mathematical interpretations of FEM are used eclectically, with none overshadowing the other. Key steps of the computer implementation are presented in sufficient detail so that a student can understand what goes on behind the scenes of a “black box” commercial product. The goal is that students navigating this material can eventually feel comfortable with any of the three “FEM communities” they come in contact during their professional life, whether as engineers, managers, researchers or teachers.

Book Organization

The book is divided into four Parts. The first three are of roughly similar length.

Part I: The Direct Stiffness Method. This part comprises Chapters 1 through 11. It covers major aspects of the Direct Stiffness Method (DSM). This is the most important realization of FEM, and the one implemented in general-purpose commercial finite element codes used by practicing engineers. Following an introductory first chapter, Chapters 2-4 present the fundamental steps of the DSM as a matrix method of structural analysis. A plane truss structure is used as motivating example. This is followed by Chapters 5-10 on programming, element formulation, modeling issues, and techniques for application of boundary conditions. Chapter 11 deals with relatively advanced topics including condensation and global-local analysis. Throughout these chapters the physical interpretation is emphasized for pedagogical convenience, as unifying vision of this “horizontal” framework.

Part II: Formulation of Finite Elements. This part extends from Chapters 12 through 19. It is more focused than Part I. It covers the development of elements from the more general viewpoint of the variational (energy) formulation. The presentation is inductive, always focusing on specific elements and progressing from the simplest to more complex cases. Thus Chapter 12 rederives the

² As evaluated by conventional academic metrics, which primarily test operational knowledge. One difficulty with teaching synthesis is that good engineers and designers are highly valued in industry but rarely comfortable in academia.

plane truss (bar) element from a variational formulation, while Chapter 13 presents the plane beam element. Chapter 14 introduces the plane stress problem, which serves as a testbed for the derivation of two-dimensional isoparametric elements in Chapter 15 through 18. This part concludes with an overview of requirements for convergence.

Part III: Computer Implementation. Chapters 20 through 29 deal with the computer implementation of the finite element method. Experience has indicated that students profit from doing computer homework early. This begins with Chapter 5, which contains an Introduction to *Mathematica*, and continues with homework assignments in Parts I and II. The emphasis changes in Part III to a systematic description of components of FEM programs, and the integration of those components to do problem solving.

Part IV: Structural Dynamics. This part, which starts at Chapter 30, is under preparation. It is intended as a brief introduction to the use of FEM in structural dynamics and vibration analysis, and is by nature more advanced than the other Parts.

Exercises

Most Chapters are followed by a list of homework exercises that pose problems of varying difficulty. Each exercise is labeled by a tag of the form

[type:rating]

The type is indicated by letters A, C, D or N for exercises to be answered primarily by analytical work, computer programming, descriptive narration, and numerical calculations, respectively. Some exercises involve a combination of these traits, in which case a combination of letters separated by + is used; for example A+N indicates analytical derivation followed by numerical work. For some problems heavy analytical work may be helped by the use of a computer-algebra system, in which case the type is identified as A/C.

The rating is a number between 5 and 50 that estimates the degree of difficulty of an Exercise, in the following “logarithmic” scale:

- 5 A simple question that can be answered in seconds, or is already answered in the text if the student has read and understood the material.
- 10 A straightforward question that can be answered in minutes.
- 15 A relatively simple question that requires some thinking, and may take on the order of half to one hour to answer.
- 20 Either a problem of moderate difficulty, or a straightforward one requiring lengthy computations or some programming, normally taking one to six hours of work.
- 25 A scaled up version of the above, estimated to require six hours to one day of work.
- 30 A problem of moderate difficulty that normally requires on the order of one or two days of work. Arriving at the answer may involve a combination of techniques, some background or reference material, or lengthy but straightforward programming.
- 40 A difficult problem that may be solvable only by gifted and well prepared individual students, or a team. Difficulties may be due to the need of correct formulation, advanced mathematics, or high level programming. With the proper preparation, background and tools these problems may be solved in days or weeks, while remaining inaccessible to unprepared or average students.
- 50 A research problem, worthy of publication if solved.

Most Exercises have a rating of 15 or 20. Assigning three or four per week puts a load of roughly 5-10 hours of solution work, plus the time needed to prepare the answer material. Assignments of difficulty 25 or 30 are better handled by groups, or given in take-home exams. Assignments of difficulty beyond 30 are never assigned in the course, but listed as a challenge for an elite group.

Occasionally an Exercise has two or more distinct but related parts identified as items. In that case a rating may be given for each item. For example: [A/C:15+20]. This does not mean that the exercise as a whole has a difficulty of 35, because the scale is roughly logarithmic; the numbers simply rate the expected effort per item.

Selecting Course Material

The number of chapters has been coordinated with the 28 lectures and two midterm exams of a typical 15-week semester course offered with two 75-minute lectures per week. The expectation is to cover one chapter per lecture. Midterm exams cover selective material in Parts I and II, whereas a final exam covers the entire course. It is recommended to make this final exam a one-week take-home to facilitate computer programming assignments. Alternatively a final term project may be considered. The experience of the writer, however, is that term projects are not useful at this level, since most first-year graduate students lack the synthesis ability that develops in subsequent years.

The writer has assigned weekly homeworks by selecting exercises from the two Chapters covered in the week. Choices are often given. The rating may be used by graders to weight scores. Unlike exams, group homeworks with teams of two to four students are recommended. Teams are encouraged to consult other students, as well as the instructor and teaching assistants, to get over gaps and hurdles. This group activity also lessens schedule conflicts common to working graduate students.

Feedback from course offerings as well as advances in topics such as programming languages resulted in new material being incorporated at various intervals. To keep within course coverage constraints, three courses of action were followed in revising the book.

Deleted Topics. All advanced analysis material dealing with variational calculus and direct approximation methods such as Rayleigh-Ritz, Galerkin, least squares and collocation, was eliminated by 1990. The few results needed for Part II are stated therein as recipes. That material was found to be largely a waste of time for engineering students, who typically lack the mathematical background required to appreciate the meaning and use of these methods in an application-independent context.³ Furthermore, there is abundant literature that interested students may consult should they decide to further their knowledge in those topics for self-study or thesis work.

Appendices. “Refresher” material on vector and matrix algebra has been placed on Appendices A through D. This is material that students are supposed to know as a prerequisite. Although most of it is covered by a vast literature, it was felt advisable to help students in collecting key results for quick reference in one place, and establishing a consistent notational system.

Starred Material. Chapter-specific material that is not normally covered in class is presented in fine print sections marked with an asterisk. This material belongs to two categories. One is extension to the basic topics, which suggest the way it would be covered in a more advanced FEM course. The other includes general exposition or proofs of techniques presented as recipes in class for expedience or time constraints. Starred material may be used as source for term projects or take-home exams.

³ This is a manifestation of the disconnection difficulty noted at the start of this Preface.

The book organization presents flexibility to instructors in organizing the coverage for shorter courses, for example in a quarter system, as well as fitting a three-lectures-per-week format. For the latter case it is recommended to cover two Chapters per week, while maintaining weekly homework assignments. In a quarter system a more drastic condensation would be necessary; for example much of Part I may be left out if the curriculum includes a separate course in Matrix Structural Analysis, as is common in Civil and Architectural Engineering.

Acknowledgements

Thanks are due to students and colleagues who have provided valuable feedback on the original course Notes, and helped its gradual metamorphosis into a textbook. Two invigorating sabbaticals in 1993 and 2001 provided blocks of time to develop, reformat and integrate material. The hospitality of Dr. Pål G. Bergan of Det Norske Veritas at Oslo, Norway and Professor Eugenio Oñate of CIMNE/UPC at Barcelona, Spain, during those sabbaticals is gratefully acknowledged.

Chapter Contents

Section

1	Overview	1-1
2	The Direct Stiffness Method: Breakdown	2-1
3	The Direct Stiffness Method: Assembly and Solution	3-1
4	The Direct Stiffness Method: Miscellaneous Topics	4-1
5	Analysis of Example Truss by a CAS	5-1
6	Constructing MOM Members	6-1
7	Finite Element Modeling: Introduction	7-1
8	Finite Element Modeling: Mesh, Loads, BCs	8-1
9	Multifreedom Constraints I	9-1
10	Multifreedom Constraints II	10-1
11	Superelements and Global-Local Analysis	11-1
12	The Bar Element	12-1
13	The Beam Element	13-1
14	The Plane Stress Problem	14-1
15	The Linear Triangle	15-1
16	The Isoparametric Representation	16-1
17	Isoparametric Quadrilaterals	17-1
18	Shape Function Magic	18-1
19	FEM Convergence Requirements	19-1
20	(Moved to AFEM)	20-1
21	Implementation of One-Dimensional Elements	21-1
22	FEM Programs for Plane Trusses and Frames	22-1
23	Implementation of iso-P Quadrilateral Elements	23-1
24	Implementation of iso-P Triangular Elements	24-1
23	The Assembly Procedure	23-1
24	FE Model Definition	24-1
25	Solving FEM Equations	25-1
26	(under revision)	26-1
27	(under revision)	27-1
28	Stress Recovery	28-1
29	(placeholder)	29-1
30	(under preparation)	30-1
31	(under preparation)	31-1

Appendices

A	Matrix Algebra: Vectors	A-1
B	Matrix Algebra: Matrices	B-1
C	Matrix Algebra: Determinants, Inverses, Eigenvalues	C-1
D	Matrix Calculus	D-1
H	History of Matrix Structural Analysis	H-1

R References R-1

1

Overview

TABLE OF CONTENTS

	Page
§1.1. Book Scope	1–3
§1.2. Where the Material Fits	1–3
§1.2.1. Top Level Classification	1–3
§1.2.2. Computational Mechanics	1–3
§1.2.3. Statics versus Dynamics	1–5
§1.2.4. Linear versus Nonlinear	1–5
§1.2.5. Discretization Methods	1–5
§1.2.6. FEM Formulation Levels	1–6
§1.2.7. FEM Choices	1–7
§1.2.8. Finally: What The Book Is About	1–7
§1.3. What Does a Finite Element Look Like?	1–7
§1.4. The FEM Analysis Process	1–9
§1.4.1. The Physical FEM	1–9
§1.4.2. The Mathematical FEM	1–11
§1.4.3. Synergy of Physical and Mathematical FEM	1–11
§1.4.4. Streamlined Idealization and Discretization	1–13
§1.5. Method Interpretations	1–13
§1.5.1. Physical Interpretation	1–13
§1.5.2. Mathematical Interpretation	1–14
§1.6. Keeping the Course	1–15
§1.7. *What is Not Covered	1–15
§1.8. The Origins of the Finite Element Method	1–16
§1.9. Recommended Books for Linear FEM	1–16
§1.9.1. Hasta la Vista, Fortran	1–16
§1. Notes and Bibliography.	1–17
§1. References	1–18
§1. Exercises	1–19

§1.1. Book Scope

This is a textbook about *linear structural analysis* using the Finite Element Method (FEM) as a discretization tool. It is intended to support an introductory course at the first-year level of graduate studies in Aerospace, Mechanical, or Civil Engineering.

Basic prerequisites to understanding the material covered here are: (1) a working knowledge of matrix algebra, and (2) an undergraduate structures course at the Materials of Mechanics level. Helpful but not required are previous courses in continuum mechanics and advanced structures.

This Chapter presents an overview of what the book covers, and what finite elements are.

§1.2. Where the Material Fits

This Section outlines where the book material fits within the vast scope of Mechanics. In the ensuing multilevel classification, topics addressed in some depth in this book are emphasized in **bold** typeface.

§1.2.1. Top Level Classification

Definitions of *Mechanics* in dictionaries usually state two flavors:

- The branch of Physics that studies the effect of forces and energy on physical bodies.¹
- The practical application of that science to the design, construction or operation of material systems or devices, such as machines, vehicles or structures.

These flavors are science and engineering oriented, respectively. But dictionaries are notoriously archaic. For our objectives it will be convenient to distinguish *four* flavors:

$$\text{Mechanics} \left\{ \begin{array}{l} \textit{Theoretical} \\ \textit{Applied} \\ \textbf{Computational} \\ \textit{Experimental} \end{array} \right. \quad (1.1)$$

Theoretical mechanics deals with fundamental laws and principles studied for their intrinsic scientific value. *Applied mechanics* transfers this theoretical knowledge to scientific and engineering applications, especially as regards the construction of mathematical models of physical phenomena. *Computational mechanics* solves specific problems by model-based simulation through numerical methods implemented on digital computers. *Experimental mechanics* subjects the knowledge derived from theory, application and simulation to the ultimate test of observation.

Remark 1.1. Paraphrasing an old joke about mathematicians, one may define a computational mechanician as a person who searches for solutions to given problems, an applied mechanician as a person who searches for problems that fit given solutions, and a theoretical mechanician as a person who can prove the existence of problems and solutions. As regards experimentalists, make up your own joke.

¹ Here the term “bodies” includes all forms of matter, whether solid, liquid or gaseous; as well as all physical scales, from subatomic through cosmic.

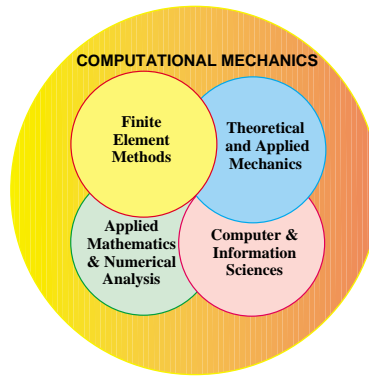


FIGURE 1.1. The “pizza slide:” Computational Mechanics integrates aspects of four disciplines.

§1.2.2. Computational Mechanics

Computational Mechanics represents the integration of several disciplines, as depicted in the “pizza slice” Figure 1.1. Several branches of computational mechanics can be distinguished according to the *physical scale* of the focus of attention:

$$\text{Computational Mechanics} \left\{ \begin{array}{l} \text{Nanomechanics} \\ \text{Micromechanics} \\ \text{Continuum mechanics} \left\{ \begin{array}{l} \text{Solids and Structures} \\ \text{Fluids} \\ \text{Multiphysics} \end{array} \right. \\ \text{Systems} \end{array} \right. \quad (1.2)$$

Nanomechanics deals with phenomena at the molecular and atomic levels. As such, it is closely related to particle physics and chemistry. At the atomic scale it transitions to quantum mechanics.

Micromechanics looks primarily at the crystallographic and granular levels of matter. Its main technological application is the design and fabrication of materials and microdevices.

Continuum mechanics studies bodies at the macroscopic level, using continuum models in which the microstructure is homogenized by phenomenological averaging. The two traditional areas of application are *solid* and *fluid mechanics*. *Structural mechanics* is a conjoint branch of solid mechanics, since structures, for obvious reasons, are fabricated with solids. Computational solid mechanics favors an applied-sciences approach, whereas computational structural mechanics emphasizes technological applications to the analysis and design of structures.

Computational fluid mechanics deals with problems that involve the equilibrium and motion of liquid and gases. Well developed related subareas are hydrodynamics, aerodynamics, atmospheric physics, propulsion, and combustion.

Multiphysics is a more recent newcomer.² This area is meant to include mechanical systems that transcend the classical boundaries of solid and fluid mechanics. A key example is interaction

² This unifying term is in fact missing from most dictionaries, as it was introduced by computational mechanicians in the 1970s. Several multiphysics problems, however, are older. For example, aircraft aeroelasticity emerged in the 1920s.

between fluids and structures, which has important application subareas such as aeroelasticity and hydroelasticity. Phase change problems such as ice melting and metal solidification fit into this category, as do the interaction of control, mechanical and electromagnetic systems.

Finally, *system* identifies mechanical objects, whether natural or artificial, that perform a distinguishable function. Examples of man-made systems are airplanes, building, bridges, engines, cars, microchips, radio telescopes, robots, roller skates and garden sprinklers. Biological systems, such as a whale, amoeba, virus or pine tree are included if studied from the viewpoint of biomechanics. Ecological, astronomical and cosmological entities also form systems.³

In the progression of (1.2), *system* is the most general concept. Systems are studied by *decomposition*: its behavior is that of its components plus the interaction between the components. Components are broken down into subcomponents and so on. As this hierarchical process continues the individual components become simple enough to be treated by individual disciplines, but their interactions may get more complex. Thus there are tradeoff skills in deciding where to stop.⁴

§1.2.3. Statics versus Dynamics

Continuum mechanics problems may be subdivided according to whether inertial effects are taken into account or not:

$$\text{Continuum mechanics} \left\{ \begin{array}{l} \text{Statics} \left\{ \begin{array}{l} \text{Time Invariant} \\ \text{Quasi-static} \end{array} \right. \\ \text{Dynamics} \end{array} \right. \quad (1.3)$$

In *statics* inertial forces are ignored or neglected. These problems may be subclassified into *time invariant* and *quasi-static*. For the former time need not be considered explicitly; any time-like response-ordering parameter (should one be needed) will do. In quasi-static problems such as foundation settlements, creep flow, rate-dependent plasticity or fatigue cycling, a more realistic estimation of time is required but inertial forces are ignored as long as motions remain slow.

In *dynamics* the time dependence is explicitly considered because the calculation of inertial (and/or damping) forces requires derivatives respect to actual time to be taken.

§1.2.4. Linear versus Nonlinear

A classification of static problems that is particularly relevant to this book is

$$\text{Statics} \left\{ \begin{array}{l} \text{Linear} \\ \text{Nonlinear} \end{array} \right. \quad (1.4)$$

Linear static analysis deals with static problems in which the *response* is linear in the cause-and-effect sense. For example: if the applied forces are doubled, the displacements and internal stresses also double. Problems outside this domain are classified as *nonlinear*.

³ Except that their function may not be clear to us. “What is it that breathes fire into the equations and makes a universe for them to describe? The usual approach of science of constructing a mathematical model cannot answer the questions of why there should be a universe for the model to describe. Why does the universe go to all the bother of existing?” (Stephen Hawking).

⁴ Thus in breaking down a car engine, say, the decomposition does not usually proceed beyond the components that may be bought at a automotive shop.

§1.2.5. Discretization Methods

A final classification of computational solid and structural mechanics (CSSM) for static analysis is based on the discretization method by which the continuum mathematical model is *discretized* in space, *i.e.*, converted to a discrete model of finite number of degrees of freedom:

$$\text{CSSM spatial discretization} \left\{ \begin{array}{l} \text{Finite Element Method (FEM)} \\ \text{Boundary Element Method (BEM)} \\ \text{Finite Difference Method (FDM)} \\ \text{Finite Volume Method (FVM)} \\ \text{Spectral Method} \\ \text{Mesh-Free Method} \end{array} \right. \quad (1.5)$$

For *linear* problems finite element methods currently dominate the scene, with boundary element methods posting a strong second choice in selected application areas. For *nonlinear* problems the dominance of finite element methods is overwhelming.

Classical *finite difference* methods in solid and structural mechanics have virtually disappeared from practical use. This statement is not true, however, for fluid mechanics, where finite difference discretization methods are still important although their dominance has diminished over time. *Finite-volume methods*, which focus on the direct discretization of conservation laws, are favored in highly nonlinear problems of fluid mechanics. *Spectral methods* are based on global transformations, based on eigendecomposition of the governing equations, that map the physical computational domain to transform spaces where the problem can be efficiently solved.

A recent newcomer to the scene are the *mesh-free methods*. These are finite different methods on arbitrary grids constructed using a subset of finite element techniques

§1.2.6. FEM Formulation Levels

The term *Finite Element Method* actually identifies a broad spectrum of techniques that share common features. Since its emergence in the framework of the Direct Stiffness Method (DSM) over 1956–1964, [747,750] FEM has expanded like a tsunami, surging from its origins in aerospace structures to cover a wide range of nonstructural applications, notably thermomechanics, fluid dynamics, and electromagnetics. The continuously expanding range makes taxonomy difficult. Restricting ourselves to applications in computational solid and structural mechanics (CSSM), one classification of particular relevance to this book is

$$\text{FEM-CSSM Formulation Level} \left\{ \begin{array}{l} \text{Mechanics of Materials (MoM) Formulation} \\ \text{Conventional Variational Formulation} \\ \text{Advanced Variational Formulation} \\ \text{Template Formulation} \end{array} \right. \quad (1.6)$$

The MoM formulation is applicable to simple structural elements such as bars and beams, and does not require any knowledge of variational methods. This level is accessible to undergraduate students, as only require some elementary knowledge of linear algebra and makes no use of variational calculus. The second level is characterized by two features: the use of standard work and energy

methods (such as the Total Potential Energy principle), and focus on full compliance with the requirements of the classical Ritz-Galerkin direct variational methods (for example, interelement continuity). It is appropriate for first year (master level) graduate students with basic exposure to variational methods. The two lower levels were well established by 1970, with no major changes since, and are those used in the present book.

The next two levels are covered in the Advanced Finite Element Methods book [251]. The third one requires a deeper exposure to variational methods in mechanics, notably multifield and hybrid principles. The last level (templates) is the pinnacle “where the rivers of our wisdom flow into one another.” Reaching it requires both mastery of advanced variational principles, as well as the confidence and fortitude to discard them along the way to the top.

§1.2.7. FEM Choices

A more down to earth classification considers two key selection attributes: Primary Unknown Variable(s), or PUV, and solution method:⁵

$$\text{PUV Choice} \left\{ \begin{array}{l} \text{Displacement (a.k.a. Primal)} \\ \text{Force (a.k.a. Dual or Equilibrium)} \\ \text{Mixed (a.k.a. Primal-Dual)} \\ \text{Hybrid} \end{array} \right. \quad \text{Solution Choice} \left\{ \begin{array}{l} \text{Stiffness} \\ \text{Flexibility} \\ \text{Combined} \end{array} \right. \quad (1.7)$$

Here **PUV Choice** governs the variational framework chosen to develop the discrete equations; if one works at the two middle levels of (1.6). It is possible, however, to develop those completely *outside* a variational framework, as noted there. The solution choice is normally dictated by the PUV, but exceptions are possible.

§1.2.8. Finally: What The Book Is About

Using the classification of (1.1) through (1.5) we can now state the book topic more precisely:

The model-based simulation of linear static structures discretized by FEM, formulated at the two lowest levels of (1.6).

(1.8)

Of the FEM variants listed in (1.7) emphasis will be placed on the *displacement* PUV choice and *stiffness* solution, just like in [253]. This particular combination is called the *Direct Stiffness Method* or DSM.

⁵ The alternative PUV terms: primal, dual or primal-dual, are those used in FEM non-structural applications, as well as in more general computational methods.

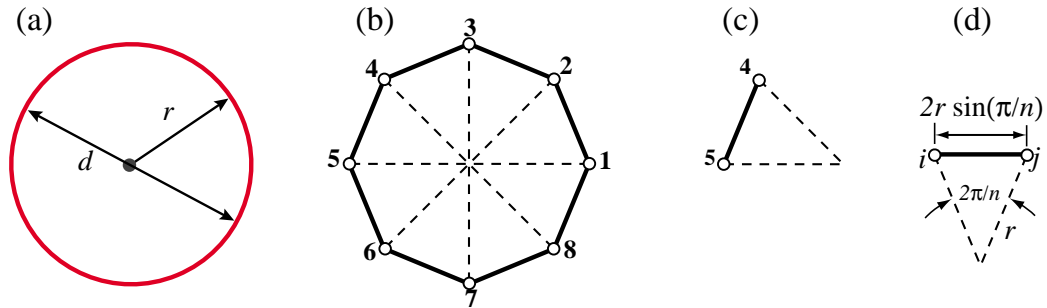


FIGURE 1.2. The “find π ” problem treated with FEM concepts: (a) continuum object, (b) a discrete approximation by inscribed regular polygons, (c) disconnected element, (d) generic element.

§1.3. What Does a Finite Element Look Like?

The subject of this book is FEM. But what *is* a finite element? As discussed later, the term admits of two interpretations: physical and mathematical. For now the underlying concept will be partly illustrated through a truly ancient problem: find the perimeter L of a circle of diameter d . Since $L = \pi d$, this is equivalent to obtaining a numerical value for π .

Draw a circle of radius r and diameter $d = 2r$ as in Figure 1.2(a). Inscribe a regular polygon of n sides, where $n = 8$ in Figure 1.2(b). Rename polygon sides as *elements* and vertices as *nodes*. Label nodes with integers $1, \dots, 8$. Extract a typical element, say that joining nodes 4–5, as shown in Figure 1.2(c). This is an instance of the *generic element* i – j pictured in Figure 1.2(d). The element length is $L_{ij} = 2r \sin(\pi/n)$. Since all elements have the same length, the polygon perimeter is $L_n = nL_{ij}$, whence the approximation to π is $\pi_n = L_n/d = n \sin(\pi/n)$.

Table 1.1. Rectification of Circle by Inscribed Polygons (“Archimedes FEM”)

n	$\pi_n = n \sin(\pi/n)$	Extrapolated by Wynn- ϵ	Exact π to 16 places
1	0.0000000000000000		
2	2.0000000000000000		
4	2.828427124746190	3.414213562373096	
8	3.061467458920718		
16	3.121445152258052	3.141418327933211	
32	3.136548490545939		
64	3.140331156954753	3.141592658918053	
128	3.141277250932773		
256	3.141513801144301	3.141592653589786	3.141592653589793

Values of π_n obtained for $n = 1, 2, 4, \dots, 256$ and $r = 1$ are listed in the second column of Table 1.1. As can be seen the convergence to π is fairly slow. However, the sequence can be transformed by Wynn’s ϵ algorithm⁶ into that shown in the third column. The last value displays 15-place accuracy.

⁶ A widely used lozenge extrapolation algorithm that speeds up the convergence of many sequences. See, e.g. [794].

Some key ideas behind the FEM can be identified in this example. The circle, viewed as a *source mathematical object*, is replaced by polygons. These are *discrete approximations* to the circle. The sides, renamed as *elements*, are specified by their end *nodes*. Elements can be separated by disconnecting nodes, a process called *disassembly* in the FEM. Upon disassembly a *generic element* can be defined, *independently of the original circle*, by the segment that connects two nodes i and j . The relevant element property: side length L_{ij} , can be computed in the generic element independently of the others, a property called *local support* in the FEM. The target property: polygon perimeter, is obtained by reconnecting n elements and adding up their length; the corresponding steps in the FEM being *assembly* and *solution*, respectively. There is of course nothing magic about the circle; the same technique can be used to rectify any smooth plane curve.⁷

This example has been offered in the FEM literature, e.g. in [464], to aduce that finite element ideas can be traced to Egyptian mathematicians from *circa* 1800 B.C., as well as Archimedes' famous studies on circle rectification by 250 B.C. But comparison with the modern FEM, as covered in following Chapters, shows this to be a stretch. The example does not illustrate the concept of degrees of freedom, conjugate quantities and local-global coordinates. It is guilty of circular reasoning: the compact formula $\pi = \lim_{n \rightarrow \infty} n \sin(\pi/n)$ uses the unknown π in the right hand side.⁸ Reasonable people would argue that a circle is a simpler object than, say, a 128-sided polygon. Despite these flaws the example is useful in one respect: showing a fielder's choice in the replacement of one mathematical object by another. This is at the root of the simulation process described next.

§1.4. The FEM Analysis Process

Processes that use FEM involve carrying out a sequence of steps in some way. Those sequences take two canonical configurations, depending on (i) the environment in which FEM is used and (ii) the main objective: model-based simulation of physical systems, or numerical approximation to mathematical problems. Both are reviewed below to introduce terminology used in the sequel.

§1.4.1. The Physical FEM

A canonical use of FEM is simulation of physical systems. This requires models of such systems. Consequently the methodology is often called *model-based simulation*.

The process is illustrated in Figure 1.3. The centerpiece is the *physical system* to be modeled. Accordingly, this configuration is called the *Physical FEM*. The processes of idealization and discretization are carried out *concurrently* to produce the discrete model. The solution step is handled by an equation solver often customized to FEM, which delivers a discrete solution (or solutions).

Figure 1.3 also shows an *ideal mathematical model*. This may be presented as a *continuum limit* or “continuification” of the discrete model. For some physical systems, notably those well modeled by continuum fields, this step is useful. For others, such as complex engineering systems (say, a flying aircraft) it makes no sense. Indeed Physical FEM discretizations may be constructed and adjusted *without reference to mathematical models*, simply from experimental measurements.

⁷ A similar limit process, however, may fail in three dimensions for evaluation of surface areas.

⁸ The circularity objection is bypassed if n is advanced as a power of two, as in Table 1.1, by using the half-angle recursion

$$\sqrt{2} \sin \alpha = \sqrt{1 - \sqrt{1 - \sin^2 2\alpha}}, \text{ started from } 2\alpha = \pi \text{ for which } \sin \pi = -1.$$

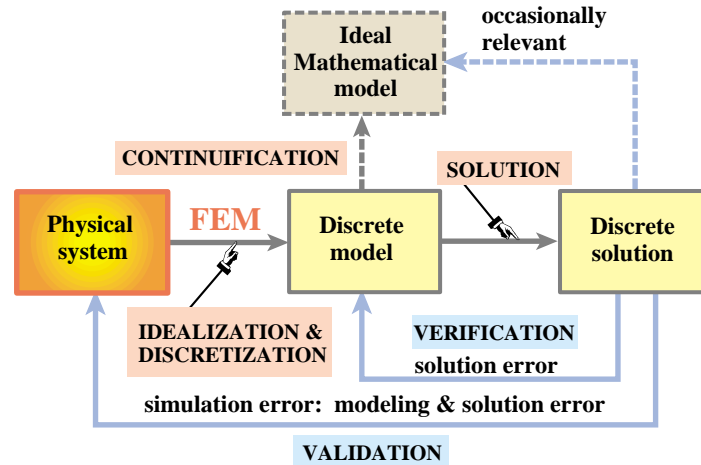


FIGURE 1.3. The Physical FEM. The physical system (left box) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

The concept of *error* arises in the Physical FEM in two ways. These are known as *verification* and *validation*, respectively. Verification is done by replacing the discrete solution into the discrete model to get the solution error. This error is not generally important. Substitution in the ideal mathematical model in principle provides the *discretization error*. This step is rarely useful in complex engineering systems, however, because there is no reason to expect that the continuum model exists, and even if it does, that it is more physically relevant than the discrete model.

Validation tries to compare the discrete solution against observation by computing the *simulation error*, which combines modeling and solution errors. As the latter is typically unimportant, the simulation error in practice can be identified with the modeling error. In real-life applications this error overwhelms the others.⁹

One way to adjust the discrete model so that it represents the physics better is called *model updating*. The discrete model is given free parameters. These are determined by comparing the discrete solution against experiments, as illustrated in Figure 1.4. Inasmuch as the minimization conditions are generally nonlinear (even if the model is linear) the updating process is inherently iterative.

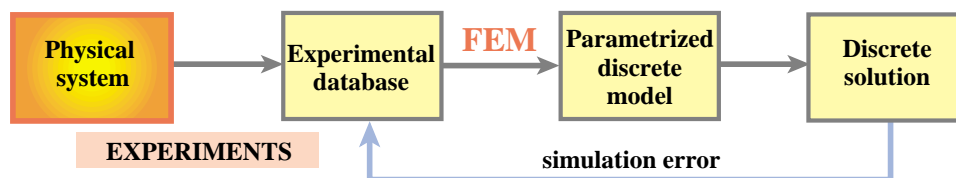


FIGURE 1.4. Model updating process in the Physical FEM.

⁹ “All models are wrong; some are useful” (George Box)

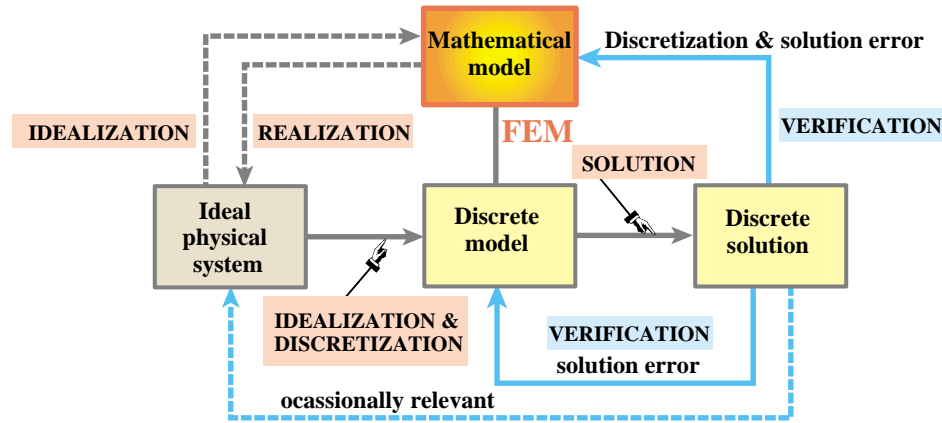


FIGURE 1.5. The Physical FEM. The physical system (left box) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

§1.4.2. The Mathematical FEM

The other canonical way of using FEM focuses on the mathematics. The process steps are illustrated in Figure 1.5. The spotlight now falls on the *mathematical model*. This is often an ordinary differential equation (ODE), or a partial differential equation (PDE) in space and time. A discrete finite element model is generated from a variational or weak form of the mathematical model.¹⁰ This is the *discretization* step. The FEM equations are solved as described for the Physical FEM.

On the left, Figure 1.5 shows an *ideal physical system*. This may be presented as a *realization* of the mathematical model. Conversely, the mathematical model is said to be an *idealization* of this system. E.g., if the mathematical model is the Poisson's PDE, realizations may be heat conduction or an electrostatic charge-distribution problem. This step is inessential and may be left out. Indeed Mathematical FEM discretizations *may be constructed without any reference to physics*.

The concept of *error* arises when the discrete solution is substituted in the “model” boxes. This replacement is generically called *verification*. As in the Physical FEM, the *solution error* is the amount by which the discrete solution fails to satisfy the discrete equations. This error is relatively unimportant when using computers, and in particular direct linear equation solvers, for the solution step. More relevant is the *discretization error*, which is the amount by which the discrete solution fails to satisfy the mathematical model.¹¹ Replacing into the ideal physical system would in principle quantify modeling errors. In the Mathematical FEM this is largely irrelevant, however, because the ideal physical system is merely that: a figment of the imagination.

§1.4.3. Synergy of Physical and Mathematical FEM

The foregoing canonical sequences are not exclusive but complementary. This synergy¹² is one of the reasons behind the power and acceptance of the method. Historically the Physical FEM was the

¹⁰ The distinction between strong, weak and variational forms is discussed in advanced FEM courses. In the present book such forms will be largely stated (and used) as recipes.

¹¹ This error can be computed in several ways, the details of which are of no importance here.

¹² Such interplay is not exactly a new idea: “The men of experiment are like the ant, they only collect and use; the reasoners resemble spiders, who make cobwebs out of their own substance. But the bee takes the middle course: it gathers its material from the flowers of the garden and field, but transforms and digests it by a power of its own.” (Francis Bacon).

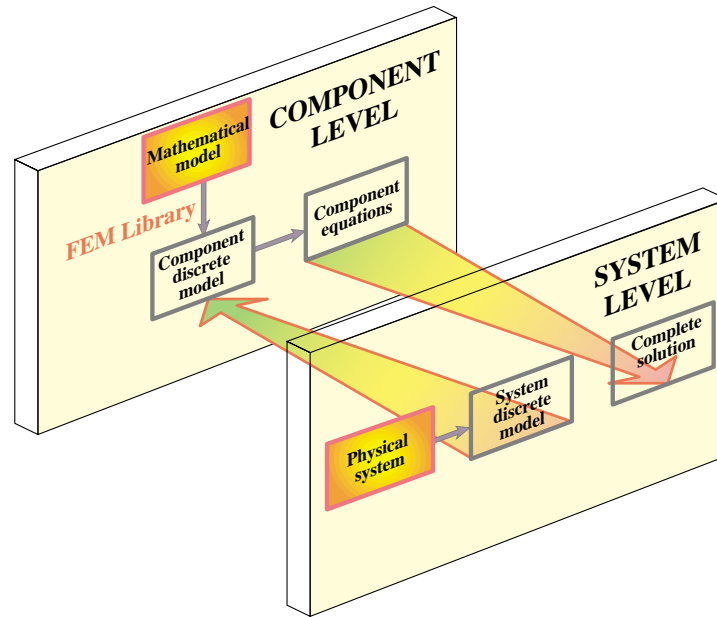


FIGURE 1.6. Combining physical and mathematical modeling through multilevel FEM. Only two levels (system and component) are shown for simplicity.

first one to be developed to model complex physical systems such as aircraft, as narrated in §1.7. The Mathematical FEM came later and, among other things, provided the necessary theoretical underpinnings to extend FEM beyond structural analysis.

A glance at the schematics of a commercial jet aircraft makes obvious the reasons behind the Physical FEM. There is no simple differential equation that captures, at a continuum mechanics level,¹³ the structure, avionics, fuel, propulsion, cargo, and passengers eating dinner. There is no reason for despair, however. The time honored *divide and conquer* strategy, coupled with *abstraction*, comes to the rescue.

First, separate the structure out and view the rest as masses and forces. Second, consider the aircraft structure as built up of *substructures* (a part of a structure devoted to a specific function): wings, fuselage, stabilizers, engines, landing gears, and so on.

Take each substructure, and continue to break it down into *components*: rings, ribs, spars, cover plates, actuators, etc. Continue through as many levels as necessary. Eventually those components become sufficiently simple in geometry and connectivity that they can be reasonably well described by the mathematical models provided, for instance, by Mechanics of Materials or the Theory of Elasticity. At that point, *stop*. The component level discrete equations are obtained from a FEM library based on the mathematical model.

The system model is obtained by going through the reverse process: from component equations to substructure equations, and from those to the equations of the complete aircraft. This *system*

¹³ Of course at the (sub)atomic level quantum mechanics works for everything, from landing gears to passengers. But it would be slightly impractical to represent the aircraft by, say, 10^{36} interacting particles modeled by the Schrödinger equations. More seriously, Truesdell and Toupin correctly note that “*Newtonian mechanics, while not appropriate to the corpuscles making up a body, agrees with experience when applied to the body as a whole, except for certain phenomena of astronomical scale*” [741, p. 228].

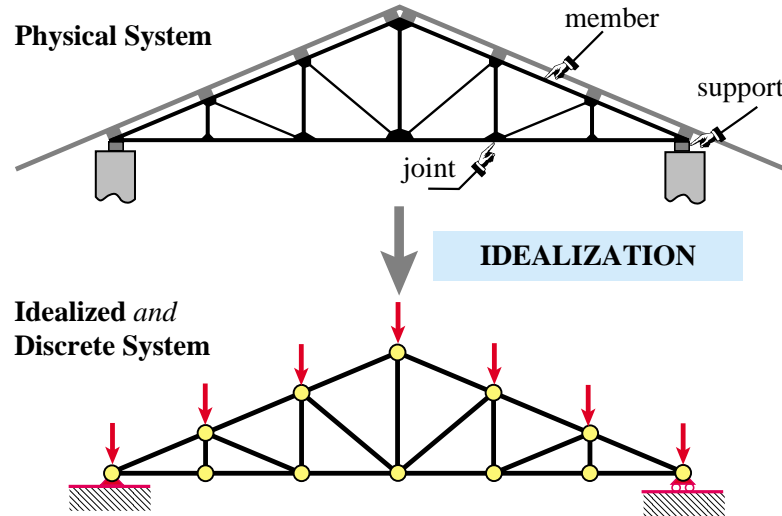


FIGURE 1.7. The idealization process for a simple structure. The physical system — here a conventional roof truss — is directly idealized by the mathematical model: a pin-jointed bar assembly. For this particular structure idealized and discrete models coalesce.

assembly process is governed by the classical principles of Newtonian mechanics, which provide the necessary inter-component “glue.” The multilevel decomposition process is diagramed in Figure 1.6, in which intermediate levels are omitted for simplicity

Remark 1.2. More intermediate decomposition levels are used in systems such as offshore and ship structures, which are characterized by a modular fabrication process. In that case multilevel decomposition mimics the way the system is actually fabricated. The general technique, called *superelements*, is discussed in Chapter 10.

Remark 1.3. There is no point in practice in going beyond a certain component level while considering the complete system. The reason is that the level of detail can become overwhelming without adding relevant information. Usually that point is reached when uncertainty impedes further progress. Further refinement of specific components is done by the so-called global-local analysis technique outlined in Chapter 10. This technique is an instance of *multiscale analysis*.

§1.4.4. Streamlined Idealization and Discretization

For sufficiently simple structures, passing to a discrete model is carried out in a single *idealization and discretization* step, as illustrated for the truss roof structure shown in Figure 1.7. Other levels are unnecessary in such cases. Of course the truss may be viewed as a substructure of the roof, and the roof as a substructure of a building. If so the multilevel process would be more appropriate.

§1.5. Method Interpretations

Just like there are two complementary ways of using the FEM, there are two complementary interpretations for explaining it, a choice that obviously impacts teaching. One interpretation stresses the *physical* significance and is aligned with the Physical FEM. The other focuses on the *mathematical* context, and is aligned with the Mathematical FEM. They are outlined next.

§1.5.1. Physical Interpretation

The physical interpretation focuses on the flowchart of Figure 1.3. This interpretation has been shaped by the discovery and extensive use of the method in the field of structural mechanics. The historical connection is reflected in the use of structural terms such as “stiffness matrix”, “force vector” and “degrees of freedom,” a terminology that carries over to non-structural applications.

The basic concept in the physical interpretation is the *breakdown* (\equiv disassembly, tearing, partition, separation, decomposition) of a complex mechanical system into simpler, disjoint components called finite elements, or simply *elements*. The mechanical response of an element is characterized in terms of a finite number of degrees of freedom. These degrees of freedoms are represented as the values of the unknown functions as a set of node points. The element response is defined by algebraic equations constructed from mathematical or experimental arguments. The response of the original system is considered to be approximated by that of the *discrete model* constructed by *connecting* or *assembling* the collection of all elements.

The breakdown-assembly concept occurs naturally when an engineer considers many artificial and natural systems. For example, it is easy and natural to visualize an engine, bridge, aircraft or skeleton as being fabricated from simpler parts.

As discussed in §1.4.3, the underlying theme is *divide and conquer*. If the behavior of a system is too complex, the recipe is to divide it into more manageable subsystems. If these subsystems are still too complex the subdivision process is continued until the behavior of each subsystem is simple enough to fit a mathematical model that represents well the knowledge level the analyst is interested in. In the finite element method such “primitive pieces” are called *elements*. The behavior of the total system is that of the individual elements plus their interaction. A key factor in the initial acceptance of the FEM was that the element interaction could be physically interpreted and understood in terms that were eminently familiar to structural engineers.

§1.5.2. Mathematical Interpretation

This interpretation is closely aligned with the flowchart of Figure 1.5. The FEM is viewed as a procedure for obtaining numerical approximations to the solution of boundary value problems (BVPs) posed over a domain Ω . This domain is replaced by the union \cup of disjoint subdomains $\Omega^{(e)}$ called finite elements. In general the geometry of Ω is only approximated by that of $\cup \Omega^{(e)}$.

The unknown function (or functions) is locally approximated over each element by an interpolation formula expressed in terms of values taken by the function(s), and possibly their derivatives, at a set of *node points* generally located on the element boundaries. The states of the assumed unknown function(s) determined by unit node values are called *shape functions*. The union of shape functions “patched” over adjacent elements form a *trial function basis* for which the node values represent the generalized coordinates. The trial function space may be inserted into the governing equations and the unknown node values determined by the Ritz method (if the solution extremizes a variational principle) or by the Galerkin, least-squares or other weighted-residual minimization methods if the problem cannot be expressed in a standard variational form.

Remark 1.4. In the mathematical interpretation the emphasis is on the concept of *local (piecewise) approximation*. The concept of element-by-element breakdown and assembly, while convenient in the computer implementation, is not theoretically necessary. The mathematical interpretation permits a general approach

to the questions of convergence, error bounds, trial and shape function requirements, etc., which the physical approach leaves unanswered. It also facilitates the application of FEM to classes of problems that are not so readily amenable to physical visualization as structures; for example electromagnetics and heat conduction.

Remark 1.5. It is interesting to note some similarities in the development of Heaviside's operational methods, Dirac's delta-function calculus, and the FEM. These three methods appeared as ad-hoc computational devices created by engineers and physicists to deal with problems posed by new science and technology (electricity, quantum mechanics, and delta-wing aircraft, respectively) with little help from the mathematical establishment.¹⁴ Only some time after the success of the new techniques became apparent were new branches of mathematics (operational calculus, distribution theory and piecewise-approximation theory, respectively) constructed to justify that success. In the case of the finite element method, the development of a formal mathematical theory started in the late 1960s, and much of it is still in the making.

§1.6. Keeping the Course

The first Part of this book, covered in Chapters 2 through 10, stresses the physical interpretation of FEM within the framework of the Direct Stiffness Method (DSM). This is done on account of its instructional advantages. Furthermore the computer implementation becomes more transparent because the sequence of operations can be placed in close correspondence with the DSM steps.

Chapters 11 through 19 deal specifically with element formulations. Ingredients of the mathematical interpretation are called upon whenever it is felt proper and convenient to do so. Nonetheless excessive entanglement with the mathematical theory is avoided if it may obfuscate the physics.

In Chapters 2 and 3 the time is frozen at about 1965, and the DSM presented as an aerospace engineer of that time would have understood it. This is not done for sentimental reasons, although that happens to be the year in which the writer began thesis work on FEM under Ray Clough. Virtually all FEM commercial codes are now based on the DSM and the computer implementation has not essentially changed since the late 1960s.¹⁵ What has greatly improved since is “marketing sugar”: user interaction and visualization.

§1.7. *What is Not Covered

The following topics are not covered in this book:

1. Elements based on equilibrium, mixed and hybrid variational formulations.
2. Flexibility and mixed solution methods.
3. Plate and shell elements.
4. Variational methods in mechanics.
5. General mathematical theory of finite elements.
6. Buckling and stability analysis.
7. General nonlinear response analysis.
8. Structural optimization.

¹⁴ Oliver Heaviside took heavy criticism from the lotus eaters, which he returned with gusto. His legacy is a living proof that “England is the paradise of individuality, eccentricity, heresy, anomalies, hobbies and humors” (George Santayana). Paul Dirac was luckier: he was shielded as member of the physics establishment and eventually received a Nobel Prize. Gilbert Strang, the first mathematician to dwell in the real FEM (the one created by engineers) was kind to the founders.

¹⁵ With the gradual disappearance of Fortran as a “live” programming language, noted in §1.7.7, changes at the implementation level have recently accelerated. E.g., C++, Python, Java and Matlab “wrappers” are becoming more common.

9. Error estimates and problem-adaptive discretizations.
10. Non-structural and multiphysics applications of FEM.
11. Designing and building production-level FEM software and use of special hardware (*e.g.* vector and parallel computers)

Topics 1–5 belong to what may be called “Advanced Linear FEM”, which is covered in the book [251]. Topics 6–7 pertain to “Nonlinear FEM”, which is covered in the book [254]. Topics 8–10 fall into advanced applications, covered in other books in preparation, whereas 11 is an interdisciplinary topic that interweaves with computer science.

§1.8. The Origins of the Finite Element Method

This section moved to Appendix O to facilitate further expansion.

§1.9. Recommended Books for Linear FEM

The literature is voluminous: over 200 textbooks and monographs have appeared since 1967. Some recommendations for readers interested in further studies within *linear* FEM are offered below.

Basic level (reference): Zienkiewicz and Taylor [818]. This two-volume set is a comprehensive upgrade of the previous edition [816]. Primarily an encyclopædic reference work that gives a panoramic coverage of FEM applications, as well as a comprehensive list of references. Not a textbook or monograph. Prior editions suffered from loose mathematics, largely fixed in this one. A three-volume fifth edition has appeared recently.

Basic level (textbook): Cook, Malkus and Plesha [147]. The third edition is comprehensive in scope although the coverage is more superficial than Zienkiewicz and Taylor. A fourth edition has appeared recently.

Intermediate level: Hughes [380]. It requires substantial mathematical expertise on the part of the reader. Recently (2000) reprinted as Dover edition.

Mathematically oriented: Strang and Fix [690]. Still the most readable mathematical treatment for engineers, although outdated in several subjects. Out of print.

Best value for the \$\$\$: Przemieniecki’s Dover edition [590], list price \$15.95 (2003). A reprint of a 1966 McGraw-Hill book. Although woefully outdated in many respects (the word “finite element” does not appear except in post-1960 references), it is a valuable reference for programming simple elements. Contains a fairly detailed coverage of substructuring, a practical topic missing from the other books. Comprehensive bibliography in Matrix Structural Analysis up to 1966.

Most fun (if you appreciate British “humor”): Irons and Ahmad [392]. Out of print.

For buying out-of-print books through web services, check the metasearch engine in www3.addall.com (most comprehensive; not a bookseller) as well as that of www.amazon.com. A newcomer is www.campusi.com

§1.9.1. Hasta la Vista, Fortran

Most FEM books that include programming samples or even complete programs use Fortran. Those face an uncertain future. Since the mid-1990s, Fortran is gradually disappearing as a programming language taught in USA engineering undergraduate programs. (It still survives in some Physics and

Chemistry departments because of large amounts of legacy code.) So one end of the pipeline is drying up. Low-level scientific programming¹⁶ is moving to C and C++, mid-level to Java, Perl and Python, high-level to Matlab, Mathematica and their free-source Linux equivalents. How attractive can a book teaching in a dead language be?

To support this argument with some numbers, here is a September-2003 snapshot of ongoing open source software projects listed in <http://freshmeat.net>. This conveys the relative importance of various languages (a mixed bag of newcomers, going-strongs, have-beens and never-was) in the present environment.

Lang	Projects	Perc	Lang	Projects	Perc	Lang	Projects	Perc
Ada	38	0.20%	APL	3	0.02%	ASP	25	0.13%
Assembly	170	0.89%	Awk	40	0.21%	Basic	15	0.08%
C	5447	28.55%	C#	41	0.21%	C++	2443	12.80%
Cold Fusion	10	0.05%	Common Lisp	27	0.14%	Delphi	49	0.26%
Dylan	2	0.01%	Eiffel	20	0.10%	Emacs-Lisp	33	0.17%
Erlang	11	0.06%	Euler	1	0.01%	Euphoria	2	0.01%
Forth	15	0.08%	Fortran	45	0.24%	Haskell	28	0.15%
Java	2332	12.22%	JavaScript	236	1.24%	Lisp	64	0.34%
Logo	2	0.01%	ML	26	0.14%	Modula	7	0.04%
Object Pascal	9	0.05%	Objective C	131	0.69%	Ocaml	20	0.10%
Other	160	0.84%	Other Scripting Engines	82	0.43%	PHP	2020	10.59%
Pascal	38	0.20%	Perl	2752	14.42%	Pliant	1	0.01%
Pike	3	0.02%	PL/SQL	58	0.30%	Python	1171	6.14%
PROGRESS	2	0.01%	Prolog	8	0.04%	Scheme	76	0.40%
Rexx	7	0.04%	Ruby	127	0.67%	SQL	294	1.54%
Simula	1	0.01%	Smalltalk	20	0.10%	Vis Basic	15	0.08%
Tcl	356	1.87%	Unix Shell	550	2.88%	Zope	34	0.18%
Xbasic	1	0.01%	YACC	11	0.06%			
Total Projects: 19079								

Notes and Bibliography

Here is Ray Clough's personal account of how FEM and DSM emerged at Boeing in the early 1950s. (For further historical details, the interested reader may consult Appendices H and O.)

“ My involvement with the FEM began when I was employed by the Boeing Airplane Company in Seattle during summer 1952 as a member of their summer faculty program. When I had joined the civil engineering faculty at Berkeley in 1949, I decided to take advantage of my MIT structural dynamics background by taking up the field of Earthquake Engineering. So because the Boeing summer faculty program offered positions with their structural dynamics unit, I seized on that as the best means of advancing my preparation for the earthquake engineering field. I was particularly fortunate in this choice of summer work at Boeing because the head of their structural dynamics unit was Mr. M. J. Turner — a very capable man in dealing with problems of structural vibrations and flutter.

When I arrived for the summer of 1952, Jon Turner asked me to work on the vibration analysis of a delta wing structure. Because of its triangular plan form, this problem could not be solved by procedures based on standard beam theory; so I spent the summer of 1952 trying to formulate a delta wing model built up as an assemblage of one-dimensional beams and struts. However, the results of deflection analyses based on this type of mathematical model were in very poor agreement with data obtained from laboratory tests of a scale model of a delta wing. My final conclusion was that my summer's work was a total failure—however, at least I learned what did not work.

¹⁶ “A programming language is low level when its programs require attention to the irrelevant” (Alan Perlis).

Chapter 1: OVERVIEW

Spurred by this disappointment, I decided to return to Boeing for the summer faculty program in 1953. During the winter, I stayed in touch with Jon Turner so I was able to rejoin the structural dynamics unit in June. The most important development during the winter was that Jon suggested we try to formulate the stiffness property of the wing by assembling plane stress plates of either triangular or rectangular shapes. So I developed stiffness matrices for plates of both shapes, but I decided the triangular form was much more useful because such plates could be assembled to approximate structures of any configuration. Moreover, the stiffness properties of the individual triangular plates could be calculated easily based on assumptions of uniform states of normal stress in the X and the Y directions combined with an uniform state of shear stress. Then the stiffness of the complete structure was obtained by appropriate addition of the contributions from the individual pieces. The Boeing group called this procedure the direct stiffness method.

The remainder of the summer of 1953 was spent in demonstrating that deflections calculated for structures formed as assemblages of triangular elements agreed well with laboratory measurements on the actual physical models. Also, it became apparent that the precision of the calculated results could be improved asymptotically by continued refinement of the finite element mesh. The conclusions drawn from that summer's work were presented in a paper given by Jon Turner at the annual meeting of the Institute of Aeronautical Sciences in January 1954. However, for reasons I never understood Jon did not submit the paper for publication until many months later. So this paper, which often is considered to be the first published description of the FEM, was not published until September 1956 — more than two years after the verbal presentation.

It is important to note that the basic purpose of the work done by Jon Turner's structural dynamics unit was vibration and flutter analysis. They were not concerned with stress analysis because that was the responsibility of the stress analysis unit. However, it was apparent that the model formed by the direct stiffness method could be used for stress analysis as well as for vibration analysis, and I made plans to investigate this stress analysis application as soon as possible. However, because of my other research responsibilities, I was not able to spend any significant time on the stress analysis question until I went on my sabbatical leave to Trondheim, Norway in September 1956. Then, when I arrived in Norway all I could do was to outline the procedures for carrying out the analysis, and to do calculations for very small systems using a desk calculator because the Norwegian Institute of Technology did not yet have an automatic digital computer.

The presentation of the paper to the Institute of Aeronautical Sciences was the first introduction of the principles of the FEM to a technical audience; although some of the basic concepts of the method were stated a short time later in a series of articles published in *Aircraft Engineering* by Dr. John H. Argyris during October 1954 to May 1955. However, the rectangular element presented in those articles is only a minor part of that contribution. The Argyris work came to my attention during my sabbatical leave in Norway, and I considered it then (as I still do now) to be the most important series of papers ever published in the field of Structural Mechanics. I credit that work for extending the scope of my understanding of structural theory to the level it eventually attained.

From my personal point of view, the next important event in finite element history was the coining of the name FEM. My purpose in choosing that name was to distinguish clearly the relatively large size pieces of the structure that make up a finite element assemblage as contrasted with the infinitesimal contributions that go into evaluation of the displacements of a structure in a typical virtual work analysis. The name first appeared in a publication that was written to demonstrate the finite element procedure for the civil engineering profession. A much more significant application of the method was presented at the Symposium on the use of Computers in Civil Engineering, held in Lisbon, Portugal in 1962, where it was used to evaluate the stress concentrations developed in a gravity dam that had cracked at its mid-section."

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 1**Overview**

EXERCISE 1.1 [A:15] Work out Archimedes' problem using a circumscribed regular polygon, with $n = 1, 2, 4, \dots 256$. Does the sequence converge any faster?

EXERCISE 1.2 [D:20] Select one of the following vehicles: truck, car, motorcycle, or bicycle. Draw a two level decomposition of the structure into substructures, and of selected components of some substructures.

EXERCISE 1.3 [D:30] In one of the earliest articles on the FEM, Clough [137] writes:

“When idealized as an assemblage of appropriately shaped two- and three-dimensional elements in this manner, an elastic continuum can be analyzed by standard methods of structural analysis. It should be noted that the approximation which is employed in this case is of physical nature; a modified structural system is substituted for the actual continuum. There need be no approximation in the mathematical analysis of this structural system. This feature distinguishes the finite element technique from finite difference methods, in which the exact equations of the actual physical system are solved by approximate mathematical procedures.”

Discuss critically the contents of this paragraph while placing it in the context of time of writing (early 1960s). Is the last sentence accurate?

2

The Direct Stiffness Method I

TABLE OF CONTENTS

	Page
§2.1. Foreword	2–3
§2.2. Why A Plane Truss?	2–3
§2.3. Truss Structures	2–3
§2.4. Idealization	2–4
§2.5. The Example Truss	2–5
§2.6. Members, Joints, Forces and Displacements	2–6
§2.7. The Master Stiffness Equations	2–7
§2.8. The DSM Steps	2–8
§2.9. Breakdown Stage	2–10
§2.9.1. Disconnection	2–10
§2.9.2. Localization	2–11
§2.9.3. Member Stiffness Equations	2–11
§2.10. Assembly and Solution Stage: Globalization	2–12
§2.10.1. Displacement and Force Transformations	2–12
§2.10.2. Global Member Stiffness Equations	2–14
§2. Notes and Bibliography.	2–15
§2. References	2–15
§2. Exercises	2–16

§2.1. Foreword

This Chapter begins the exposition of the Direct Stiffness Method (DSM) of structural analysis. The DSM is by far the most common implementation of the Finite Element Method (FEM). In particular, all major commercial FEM codes are based on the DSM.

The exposition is done by following the DSM steps applied to a simple plane truss structure. The method has two major stages: breakdown, and assembly+solution. This Chapter covers primarily the breakdown stage.

§2.2. Why A Plane Truss?

The simplest structural finite element is the two-node bar (also called linear spring) element, which is illustrated in Figure 2.1(a). A six-node triangle that models thin plates, shown in Figure 2.1(b) displays intermediate complexity. Perhaps the most geometrically complex finite element (at least as regards number of degrees of freedom) is the curved, three-dimensional, 64-node “brick” element depicted in Figure 2.1(c).

Yet the remarkable fact is that, in the DSM, all elements, regardless of complexity, are treated alike! To illustrate the basic steps of this democratic method, it makes educational sense to keep it simple and use a structure composed of bar elements.

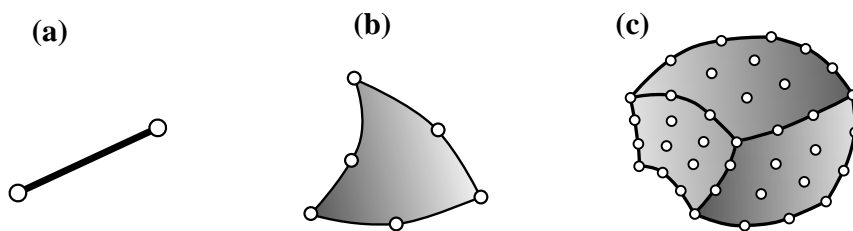


FIGURE 2.1. From the simplest through progressively more complex structural finite elements: (a) two-node bar element for trusses, (b) six-node triangle for thin plates, (c) 64-node tricubic, “brick” element for three-dimensional solid analysis.

A simple yet nontrivial structure is the *pin-jointed plane truss*, whose members may be modeled as two-node bars.¹ Using a plane truss to teach the stiffness method offers two additional advantages:

- (a) Computations can be entirely done by hand as long as the structure contains just a few elements. This allows various steps of the solution procedure to be carefully examined and understood (learning by doing) before passing to the computer implementation. Doing hand computations on more complex finite element systems rapidly becomes impossible.
- (b) The computer implementation on any programming language is relatively simple and can be assigned as preparatory computer homework before reaching Part III.

¹ A one dimensional bar assembly would be even simpler. That kind of structure would not adequately illustrate some of the DSM steps, however, notably the back-and-forth transformations from global to local coordinates.

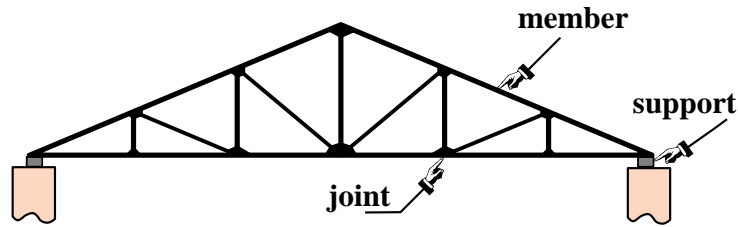


FIGURE 2.2. An actual plane truss structure. That shown is typical of a roof truss used in building construction for rather wide spans, say, over 10 meters. For shorter spans, as in residential buildings, trusses are simpler, with fewer bays.

§2.3. Truss Structures

Plane trusses, such as the one depicted in Figure 2.2, are often used in construction, particularly for roofing of residential and commercial buildings, and in short-span bridges. Trusses, whether two or three dimensional, belong to the class of *skeletal structures*. These structures consist of elongated structural components called *members*, connected at *joints*. Another important subclass of skeletal structures are frame structures or *frameworks*, which are common in reinforced concrete construction of buildings and bridges.

Skeletal structures can be analyzed by a variety of hand-oriented methods of structural analysis taught in beginning Mechanics of Materials courses: the Displacement and Force methods. They can also be analyzed by the computer-oriented FEM. That versatility makes those structures a good choice to illustrate the transition from the hand-calculation methods taught in undergraduate courses, to the fully automated finite element analysis procedures available in commercial programs.

§2.4. Idealization

The first analysis step carried out by a structural engineer is to replace the actual physical structure by a *mathematical model*. This model represents an *idealization* of the actual structure. For truss structures, by far the most common idealization is the *pin-jointed truss*, which directly maps to a FEM model. See Figure 2.3.

The replacement of true by idealized is at the core of the *physical interpretation* of the finite element method discussed in Chapter 1. The axially-carrying-load members and frictionless pins of the pin-jointed truss are only an approximation of the physical one. For example, building and bridge trusses usually have members joined to each other through the use of gusset plates, which are attached by nails, bolts, rivets or welds. Consequently members will carry some bending as well as direct axial loading.

Experience has shown, however, that stresses and deformations calculated using the simple idealized model will often be satisfactory for preliminary design purposes; for example to select the cross section of the members. Hence the engineer turns to the pin-jointed assemblage of axial-force-carrying elements and uses it to perform the structural analysis.

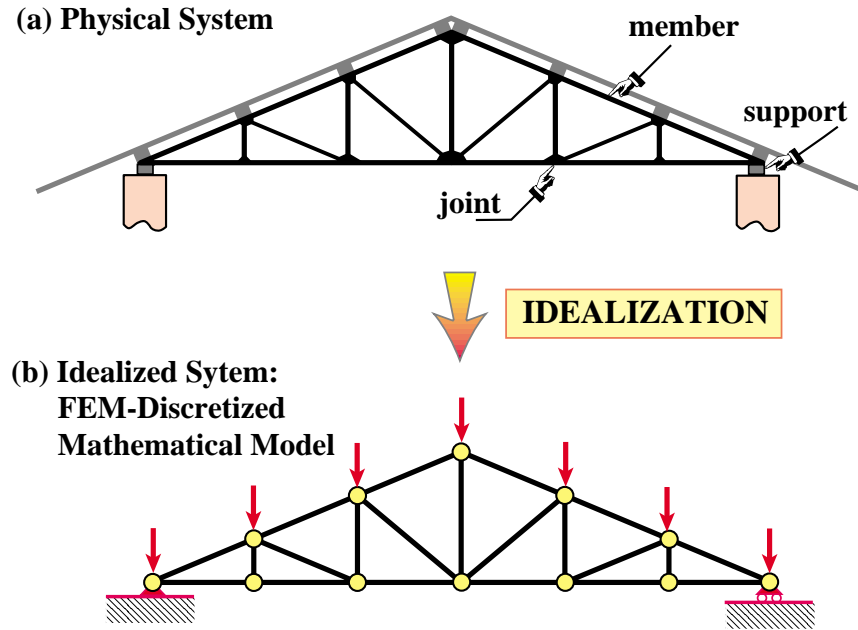


FIGURE 2.3. Idealization of roof truss: (a) physical system, (b) idealization as FEM discretized mathematical model.

In this and following Chapter we will go over the basic steps of the DSM in a “hand-computer” calculation mode. This means that although the steps are done by hand, whenever there is a procedural choice we shall either adopt the way that is better suited towards the computer implementation, or explain the difference between hand and computer computations. The actual computer implementation using a high-level programming language is presented in Chapter 4.

§2.5. The Example Truss

To keep hand computations manageable we will use just about the simplest structure that can be called a plane truss, namely the three-member truss illustrated in Figure 2.4(a). The *idealized* model of this physical truss as a pin-jointed assemblage of bars as shown in Figure 2.4(b). In this idealization truss members carry only axial loads, have no bending resistance, and are connected by frictionless pins.

Geometric, material and fabrication properties of the idealized truss are given in Figure 2.4(c),² while idealized loads and support conditions are provided in Figure 2.4(d).

It should be noted that as a practical structure the example truss is not particularly useful — that shown in Figure 2.2 is more common in construction. But with the example truss we can go over the basic DSM steps without getting mired into too many members, joints and degrees of freedom.

² Member mass densities are given in Figure 2.4(c) so that this truss can be used later for examples in vibration and dynamics analysis.

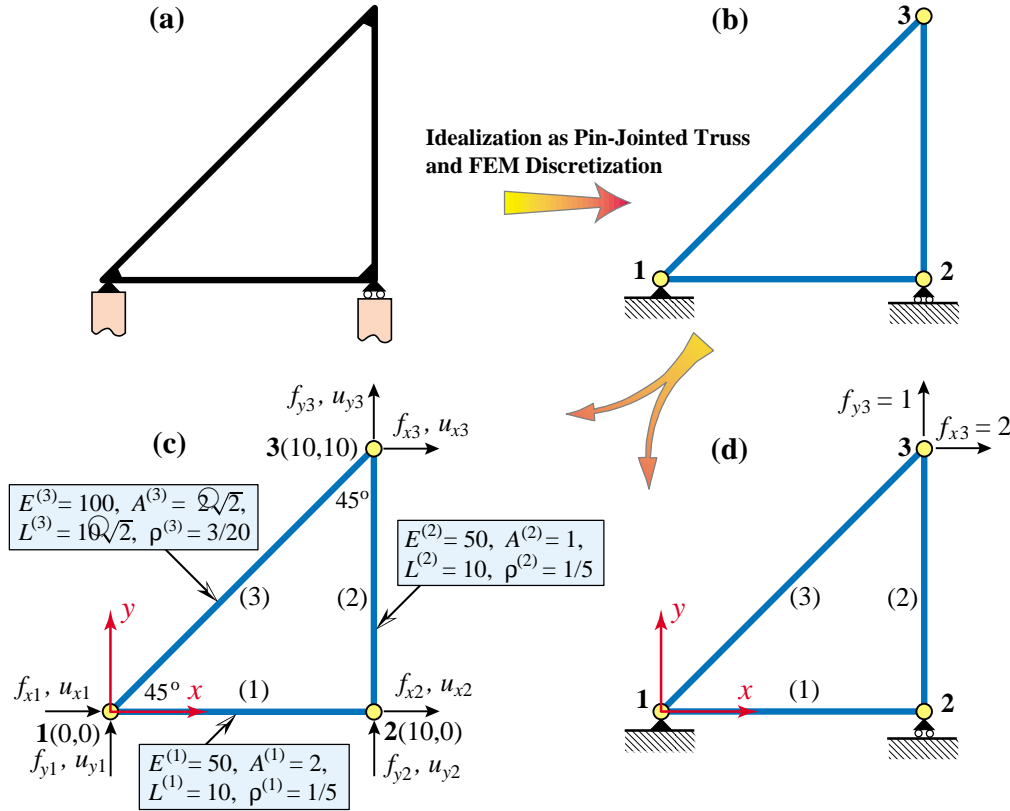


FIGURE 2.4. The three-member example truss: (a) physical structure; (b) idealization as a pin-jointed bar assemblage; (c) geometric, material and fabrication properties; (d) support conditions and applied loads.

§2.6. Members, Joints, Forces and Displacements

The pin-jointed idealization of the example truss, pictured in Figure 2.4(b,c,d), has three *joints*, which are labeled 1, 2 and 3, and three *members*, which are labeled (1), (2) and (3). Those members connect joints 1–2, 2–3, and 1–3, respectively. The member lengths are denoted by $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$, their elastic moduli by $E^{(1)}$, $E^{(2)}$ and $E^{(3)}$, and their cross-sectional areas by $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$. Note that an element number superscript is enclosed in parenthesis to avoid confusion with exponents. Both E and A are assumed to be constant along each member.

Members are generically identified by index e (because of their close relation to finite elements, as explained below). This index is placed as superscript of member properties. For example, the cross-section area of a generic member is A^e . The member superscript is *not* enclosed in parentheses in this case because no confusion with exponents can arise. But the area of member 3 is written $A^{(3)}$ and not A^3 .

Joints are generically identified by indices such as i , j or n . In the general FEM, the names “joint” and “member” are replaced by *node* and *element*, respectively. This dual nomenclature is used in the initial Chapters to stress the physical interpretation of the FEM.

The geometry of the structure is referred to a common Cartesian coordinate system $\{x, y\}$, which will be called the *global coordinate system*. Other names for it in the literature are *structure coordinate system* and *overall coordinate system*. For the example truss its origin is at joint 1.

The key ingredients of the stiffness method of analysis are the *forces* and *displacements* at the joints. In a idealized pin-jointed truss, externally applied forces as well as reactions *can act only at the joints*. All member axial forces can be characterized by the x and y components of these forces, denoted by f_x and f_y , respectively. The components at joint i will be identified as f_{xi} and f_{yi} , respectively. The set of all joint forces can be arranged as a 6-component column vector called \mathbf{f} .

The other key ingredient is the displacement field. Classical structural mechanics tells us that the displacements of the truss *are completely defined by the displacements of the joints*. This statement is a particular case of the more general finite element theory. The x and y displacement components will be denoted by u_x and u_y , respectively. The *values* of u_x and u_y at joint i will be called u_{xi} and u_{yi} . Like joint forces, they are arranged into a 6-component vector called \mathbf{u} . Here are the two vectors of nodal forces and nodal displacements, shown side by side:

$$\mathbf{f} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \quad (2.1)$$

In the DSM these six displacements are the primary unknowns. They are also called the *degrees of freedom* or *state variables* of the system.³

How about the displacement boundary conditions, popularly called support conditions? This data will tell us which components of \mathbf{f} and \mathbf{u} are actual unknowns and which ones are known *a priori*. In pre-computer structural analysis such information was used *immediately* by the analyst to discard unnecessary variables and thus reduce the amount of hand-carried bookkeeping.

The computer oriented philosophy is radically different: *boundary conditions can wait until the last moment*. This may seem strange, but on the computer the sheer volume of data may not be so important as the efficiency with which the data is organized, accessed and processed. The strategy “save the boundary conditions for last” will be followed here also for the hand computations.

Remark 2.1. Often column vectors such as (2.1) will be displayed in row form to save space, with a transpose symbol at the end. For example, $\mathbf{f} = [f_{x1} \ f_{y1} \ f_{x2} \ f_{y2} \ f_{x3} \ f_{y3}]^T$ and $\mathbf{u} = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2} \ u_{x3} \ u_{y3}]^T$.

§2.7. The Master Stiffness Equations

The *master stiffness equations* relate the joint forces \mathbf{f} of the complete structure to the joint displacements \mathbf{u} of the complete structure *before* specification of support conditions.

Because the assumed behavior of the truss is linear, these equations must be linear relations that connect the components of the two vectors. Furthermore it will be assumed that if all displacements vanish, so do the forces.⁴ If both assumptions hold the relation must be homogeneous and

³ *Primary unknowns* is the correct mathematical term whereas *degrees of freedom* has a mechanics flavor: “any of a limited number of ways in which a body may move or in which a dynamic system may change” (Merriam-Webster). The term *state variables* is used more often in nonlinear analysis, material sciences and statistics.

⁴ This assumption implies that the so-called *initial strain* effects, also known as *prestress* or *initial stress* effects, are neglected. Such effects are produced by actions such as temperature changes or lack-of-fit fabrication, and are studied in Chapter 29.

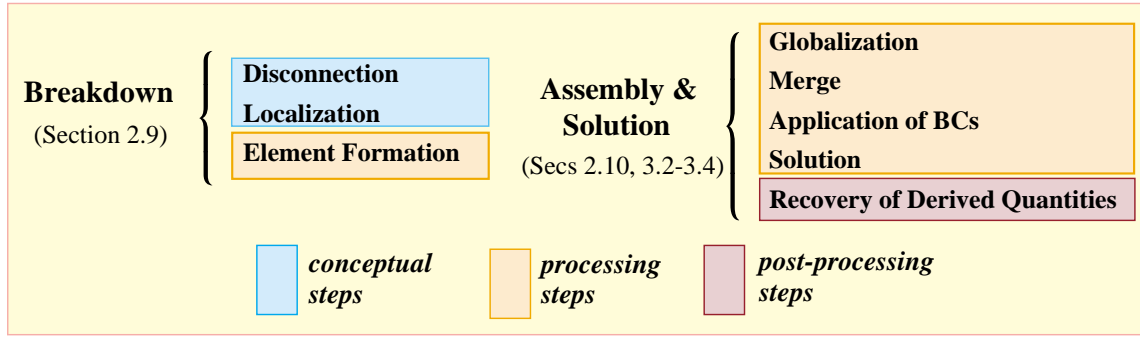


FIGURE 2.5. The Direct Stiffness Method steps.

expressable in component form as

$$\begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} K_{x1x1} & K_{x1y1} & K_{x1x2} & K_{x1y2} & K_{x1x3} & K_{x1y3} \\ K_{y1x1} & K_{y1y1} & K_{y1x2} & K_{y1y2} & K_{y1x3} & K_{y1y3} \\ K_{x2x1} & K_{x2y1} & K_{x2x2} & K_{x2y2} & K_{x2x3} & K_{x2y3} \\ K_{y2x1} & K_{y2y1} & K_{y2x2} & K_{y2y2} & K_{y2x3} & K_{y2y3} \\ K_{x3x1} & K_{x3y1} & K_{x3x2} & K_{x3y2} & K_{x3x3} & K_{x3y3} \\ K_{y3x1} & K_{y3y1} & K_{y3x2} & K_{y3y2} & K_{y3x3} & K_{y3y3} \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \quad (2.2)$$

In matrix notation:

$$\mathbf{f} = \mathbf{K}\mathbf{u}. \quad (2.3)$$

Here \mathbf{K} is the *master stiffness matrix*, also called *global stiffness matrix*, *assembled stiffness matrix*, or *overall stiffness matrix*. It is a 6×6 square matrix that happens to be symmetric, although this attribute has not been emphasized in the written-out form (2.2). The entries of the stiffness matrix are often called *stiffness coefficients* and have a physical interpretation discussed below.

The qualifiers (“master”, “global”, “assembled” and “overall”) convey the impression that there is another level of stiffness equations lurking underneath. And indeed there is a *member level* or *element level*, into which we plunge in the **Breakdown** section.

Remark 2.2. Interpretation of Stiffness Coefficients. The following interpretation of the entries of \mathbf{K} is valuable for visualization and checking. Choose a displacement vector \mathbf{u} such that all components are zero except the i^{th} one, which is one. Then \mathbf{f} is simply the i^{th} column of \mathbf{K} . For instance if in (2.3) we choose u_{x2} as unit displacement,

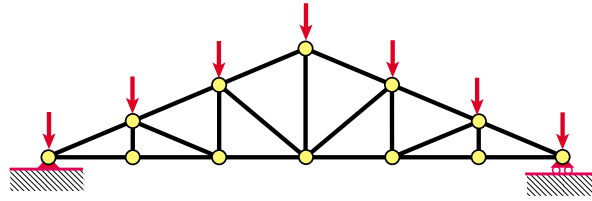
$$\mathbf{u} = [0 \ 0 \ 1 \ 0 \ 0 \ 0]^T, \quad \mathbf{f} = [K_{x1x2} \ K_{y1x2} \ K_{x2x2} \ K_{y2x2} \ K_{x3x2} \ K_{y3x2}]^T. \quad (2.4)$$

Thus K_{y1x2} , say, represents the y -force at joint 1 that would arise on prescribing a unit x -displacement at joint 2, while all other displacements vanish. In structural mechanics this property is called *interpretation of stiffness coefficients* as *displacement influence coefficients*. It extends unchanged to the general finite element method.

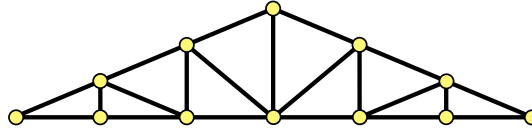
§2.8. The DSM Steps

The DSM steps, major and minor, are summarized in Figure 2.5 for the convenience of the reader. The two major stages are **Breakdown**, followed by **Assembly & Solution**. A postprocessing step may follow, although this is not part of the DSM proper.

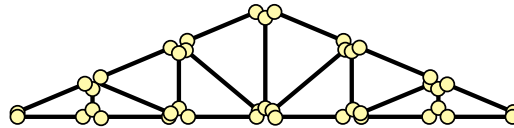
FEM model:



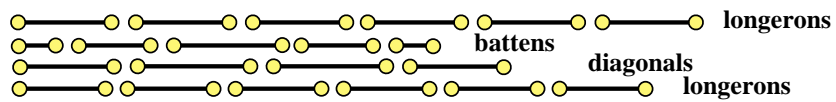
**Remove loads
& supports:**



Disassemble:



Localize:

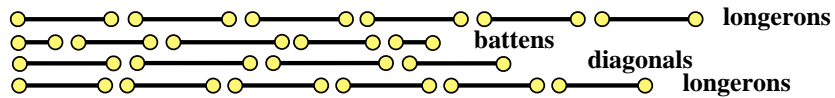


Generic element:

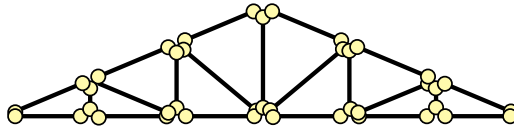


FIGURE 2.6. Visualization of the DSM breakdown stage.

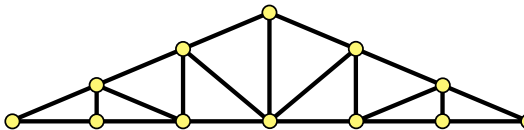
**Form
elements:**



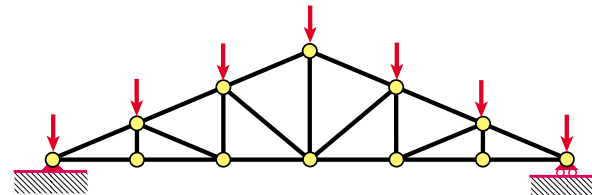
Globalize:



Merge:



**Apply loads
and supports:**



**Solve for joint
displacements:**

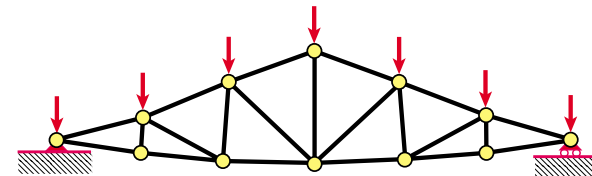


FIGURE 2.7. Visualization of the DSM assembly-and-solution stage.

The first three DSM steps are: (1) disconnection, (2) localization, and (3) computation of member stiffness equations. Collectively these form the *breakdown* stage. The first two are flagged as *conceptual* in Figure 2.5 because they are not actually programmed as such: they are implicitly carried out either through the user-provided problem definition, or produced by separate preprocessing programs such as CAD front ends. DSM processing actually begins at the element-stiffness-equation forming step.

Before starting with the detailed, step by step description of the DSM, it is convenient to exhibit the whole process through a graphic sequence. This is done in Figures 2.6 and 2.7, which are shown to students as animated slides. The sequence starts with the FEM model of the typical roof truss displayed in Figure 2.3(b). Thus the idealization step pictured there is assumed to have been carried out.⁵

§2.9. Breakdown Stage

The three breakdown steps: disconnection, localization and formation of the element stiffness equations, are covered next.

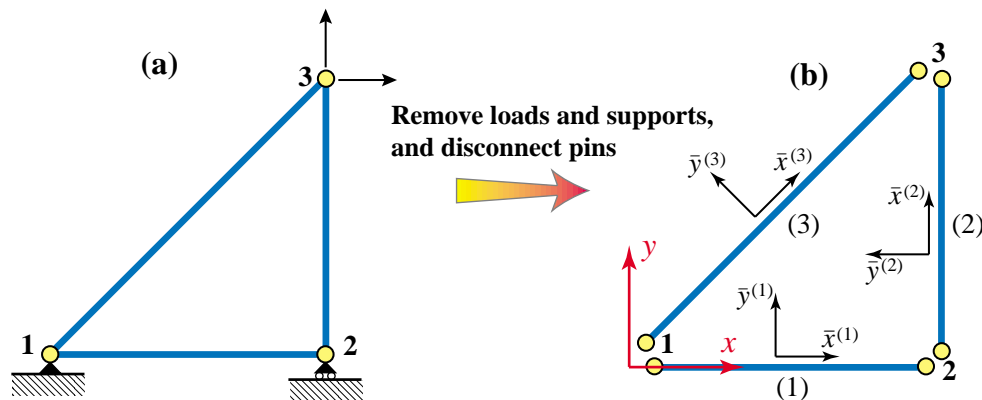


FIGURE 2.8. Disconnection step: (a) idealized example truss; (b) removal of loads and support, disconnection into members (1), (2) and (3), and selection of local coordinate systems. The latter are drawn offset from member axes for visualization convenience.

§2.9.1. Disconnection

To carry out the first breakdown step we begin by discarding all loads and supports (the so-called boundary conditions). Next we *disconnect* or *disassemble* the structure into its components. For a pin-jointed truss disconnection can be visualized as removing the pin connectors. This is illustrated for the example truss in Figure 2.8. To each member $e = 1, 2, 3$ assign a Cartesian system $\{\bar{x}^e, \bar{y}^e\}$. Axis \bar{x}^e is aligned along the axis of the e^{th} member. Actually \bar{x}^e runs along the member longitudinal axis; it is drawn offset in Figure 2.8 (b) for clarity.

By convention the positive direction of \bar{x}^e runs from joint i to joint j , where $i < j$. The angle formed by \bar{x}^e and x is the *orientation angle* φ^e , positive CCW. The axes origin is arbitrary and may be placed at the member midpoint or at one of the end joints for convenience.

⁵ The sequence depicted in Figures 2.6 and 2.7 uses the typical roof truss of Figures 2.2 and 2.3, rather than the 3-member example truss. Reason: those pictures are more representative of actual truss structures, as seen often by students.

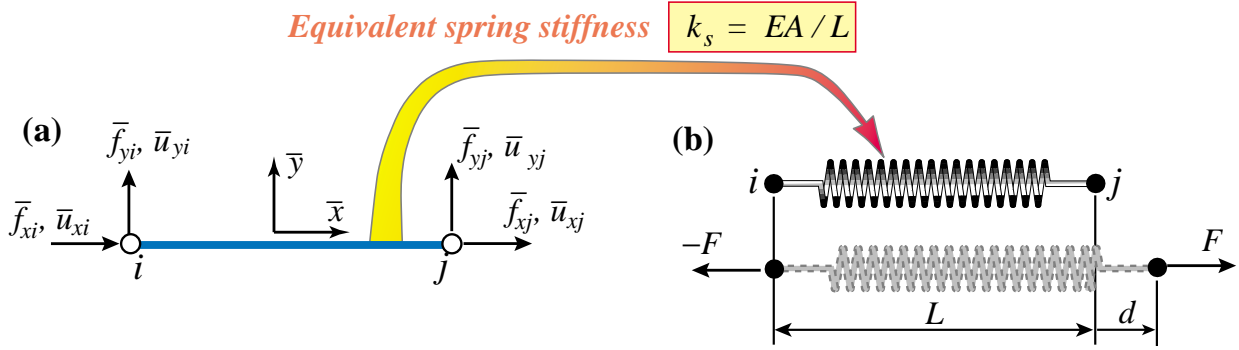


FIGURE 2.9. Generic truss member referred to its local coordinate system $\{\bar{x}, \bar{y}\}$: (a) idealization as 2-node bar element, (b) interpretation as equivalent spring. Element identification number e dropped to reduce clutter.

Systems $\{\bar{x}^e, \bar{y}^e\}$ are called *local coordinate systems* or *member-attached coordinate systems*. In the general finite element method they also receive the name *element coordinate systems*.

§2.9.2. Localization

To reduce clutter we drop the member identifier e so we are effectively dealing with a *generic* truss member, as illustrated in Figure 2.9(a). The local coordinate system is $\{\bar{x}, \bar{y}\}$. The two end joints are labelled i and j . As shown in that figure, a generic plane truss member has four joint force components and four joint displacement components (the member degrees of freedom). The member properties are length L , elastic modulus E and cross-section area A .

§2.9.3. Member Stiffness Equations

The force and displacement components of the generic truss member shown in Figure 2.9(a) are linked by the *member stiffness relations*

$$\bar{\mathbf{f}} = \bar{\mathbf{K}} \bar{\mathbf{u}}, \quad (2.5)$$

which written out in full become

$$\begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} \bar{K}_{xixi} & \bar{K}_{xiyi} & \bar{K}_{xixj} & \bar{K}_{xiyj} \\ \bar{K}_{yixi} & \bar{K}_{yiyi} & \bar{K}_{yixj} & \bar{K}_{yiyj} \\ \bar{K}_{xjxi} & \bar{K}_{xjyi} & \bar{K}_{xjxj} & \bar{K}_{xjyj} \\ \bar{K}_{yjxi} & \bar{K}_{yjyi} & \bar{K}_{yjxj} & \bar{K}_{yjyj} \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix}. \quad (2.6)$$

Vectors $\bar{\mathbf{f}}$ and $\bar{\mathbf{u}}$ are called the *member joint forces* and *member joint displacements*, respectively, whereas $\bar{\mathbf{K}}$ is the *member stiffness matrix* or *local stiffness matrix*. When these relations are interpreted from the standpoint of the general FEM, “member” is replaced by “element” and “joint” by “node.”

There are several ways to construct the stiffness matrix $\bar{\mathbf{K}}$ in terms of L , E and A . The most straightforward technique relies on the Mechanics of Materials approach covered in undergraduate courses. Think of the truss member in Figure 2.9(a) as a linear spring of equivalent stiffness k_s , an

interpretation illustrated in Figure 2.9(b). If the member properties are *uniform* along its length, Mechanics of Materials bar theory tells us that⁶

$$k_s = \frac{EA}{L}, \quad (2.7)$$

Consequently the force-displacement equation is

$$F = k_s d = \frac{EA}{L} d, \quad (2.8)$$

where F is the internal axial force and d the relative axial displacement, which physically is the bar elongation. The axial force and elongation can be immediately expressed in terms of the joint forces and displacements as

$$F = \bar{f}_{xj} = -\bar{f}_{xi}, \quad d = \bar{u}_{xj} - \bar{u}_{xi}, \quad (2.9)$$

which express force equilibrium⁷ and kinematic compatibility, respectively. Combining (2.8) and (2.9) we obtain the matrix relation⁸

$$\bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix} = \bar{\mathbf{K}} \bar{\mathbf{u}}, \quad (2.10)$$

Hence

$$\bar{\mathbf{K}} = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \quad (2.11)$$

This is the truss stiffness matrix in local coordinates.

Two other methods for deriving the local force-displacement relation (2.8) are covered in Exercises 2.6 and 2.7.

§2.10. Assembly and Solution Stage: Globalization

The first step in the assembly & solution stage, as shown in Figure 2.5, is *globalization*. This operation is done member by member. It refers the member stiffness equations to the global system $\{x, y\}$ so it can be merged into the master stiffness. Before entering into details we must establish relations that connect joint displacements and forces in the global and local coordinate systems. These are given in terms of *transformation matrices*.

⁶ See for example, Chapter 2 of [62].

⁷ Equations $F = \bar{f}_{xj} = -\bar{f}_{xi}$ follow by considering the free body diagram (FBD) of each joint. For example, take joint i as a FBD. Equilibrium along x requires $-F - \bar{f}_{xi} = 0$ whence $F = -\bar{f}_{xi}$. Doing the same on joint j yields $F = \bar{f}_{xj}$.

⁸ The matrix derivation of (2.10) is the subject of Exercise 2.3.

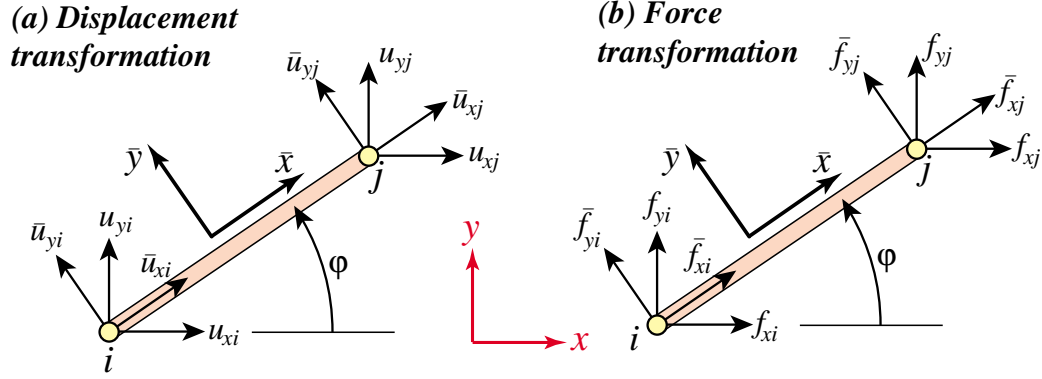


FIGURE 2.10. The transformation of node displacement and force components from the local system $\{\bar{x}, \bar{y}\}$ to the global system $\{x, y\}$.

§2.10.1. Displacement and Force Transformations

The necessary transformations are easily obtained by inspection of Figure 2.10. For the displacements

$$\begin{aligned}\bar{u}_{xi} &= u_{xi}c + u_{yi}s, & \bar{u}_{yi} &= -u_{xi}s + u_{yi}c, \\ \bar{u}_{xj} &= u_{xj}c + u_{yj}s, & \bar{u}_{yj} &= -u_{xj}s + u_{yj}c,\end{aligned}\tag{2.12}$$

in which $c = \cos \varphi$, $s = \sin \varphi$ and φ is the angle formed by \bar{x} and x , measured positive CCW from x . The matrix form that collects these relations is

$$\begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 \\ -s & c & 0 & 0 \\ 0 & 0 & c & s \\ 0 & 0 & -s & c \end{bmatrix} \begin{bmatrix} u_{xi} \\ u_{yi} \\ u_{xj} \\ u_{yj} \end{bmatrix}.\tag{2.13}$$

The 4×4 matrix that appears above is called a *displacement transformation matrix* and is denoted⁹ by \mathbf{T} . The node forces transform as $f_{xi} = \bar{f}_{xi}c - \bar{f}_{yi}s$, etc., which in matrix form become

$$\begin{bmatrix} f_{xi} \\ f_{yi} \\ f_{xj} \\ f_{yj} \end{bmatrix} = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & c & -s \\ 0 & 0 & s & c \end{bmatrix} \begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix}.\tag{2.14}$$

The 4×4 matrix that appears above is called a *force transformation matrix*. A comparison of (2.13) and (2.14) reveals that the force transformation matrix is the *transpose* \mathbf{T}^T of the displacement transformation matrix \mathbf{T} . This relation is not accidental and can be proved to hold generally.¹⁰

⁹ This matrix will be called \mathbf{T}_d when its association with displacements is to be emphasized, as in Exercise 2.5.

¹⁰ A simple proof that relies on the invariance of external work is given in Exercise 2.5. However this invariance was only checked by explicit computation for a truss member in Exercise 2.4. The general proof relies on the Principle of Virtual Work, which is discussed later.

Remark 2.3. Note that in (2.13) the local system (barred) quantities appear on the left-hand side, whereas in (2.14) they show up on the right-hand side. The expressions (2.13) and (2.14) are discrete counterparts of what are called covariant and contravariant transformations, respectively, in continuum mechanics. The continuum counterpart of the transposition relation is called *adjointness*. Collectively these relations, whether discrete or continuous, pertain to the subject of *duality*.

Remark 2.4. For this particular structural element \mathbf{T} is square and orthogonal, that is, $\mathbf{T}^T = \mathbf{T}^{-1}$. But this property does not extend to more general elements. Furthermore in the general case \mathbf{T} is not even a square matrix, and consequently does not possess an ordinary inverse. However the congruent transformation relations (2.15)–(2.17) given below do hold generally.

§2.10.2. Global Member Stiffness Equations

From now on we reintroduce the member (element) index, e . The member stiffness equations in global coordinates will be written

$$\mathbf{f}^e = \mathbf{K}^e \mathbf{u}^e. \quad (2.15)$$

The compact form of (2.13) and (2.14) for the e^{th} member is

$$\bar{\mathbf{u}}^e = \mathbf{T}^e \mathbf{u}^e, \quad \mathbf{f}^e = (\mathbf{T}^e)^T \bar{\mathbf{f}}^e. \quad (2.16)$$

Inserting these matrix expressions into $\bar{\mathbf{f}}^e = \bar{\mathbf{K}}^e \bar{\mathbf{u}}^e$ and comparing with (2.15) we find that the member stiffness in the global system $\{x, y\}$ can be computed from the member stiffness $\bar{\mathbf{K}}^e$ in the local system $\{\bar{x}, \bar{y}\}$ through the congruent transformation¹¹

$$\mathbf{K}^e = (\mathbf{T}^e)^T \bar{\mathbf{K}}^e \mathbf{T}^e. \quad (2.17)$$

Carrying out the matrix multiplications in closed form (Exercise 2.8) we get

$$\mathbf{K}^e = \frac{E^e A^e}{L^e} \begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix}, \quad (2.18)$$

in which $c = \cos \varphi^e$, $s = \sin \varphi^e$, with e superscripts of c and s suppressed to reduce clutter. If the orientation angle φ^e is zero we recover (2.10), as may be expected. \mathbf{K}^e is called a *member stiffness matrix in global coordinates*. The proof of (2.17) and verification of (2.18) is left as Exercise 2.8.

The globalized member stiffness equations for the example truss can now be easily obtained by inserting appropriate values provided in Figure 2.4(c). into (2.18).

For member (1), with end joints 1–2, $E^{(1)} A^{(1)} = 100$, $L^{(1)} = 10$, and $\varphi^{(1)} = 0^\circ$:

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \end{bmatrix} = 10 \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1}^{(1)} \\ u_{y1}^{(1)} \\ u_{x2}^{(1)} \\ u_{y2}^{(1)} \end{bmatrix}. \quad (2.19)$$

¹¹ Also known as *congruential transformation* and *congruence transformation* in linear algebra books.

For member (2), with end joints 2–3, $E^{(2)}A^{(2)} = 50$, $L^{(2)} = 10$, and $\varphi^{(2)} = 90^\circ$:

$$\begin{bmatrix} f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = 5 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x2}^{(2)} \\ u_{y2}^{(2)} \\ u_{x3}^{(2)} \\ u_{y3}^{(2)} \end{bmatrix}. \quad (2.20)$$

For member (3), with end joints 1–3, $E^{(3)}A^{(3)} = 200\sqrt{2}$, $L^{(3)} = 10\sqrt{2}$, and $\varphi^{(3)} = 45^\circ$:

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = 20 \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} u_{x1}^{(3)} \\ u_{y1}^{(3)} \\ u_{x3}^{(3)} \\ u_{y3}^{(3)} \end{bmatrix}. \quad (2.21)$$

In the following Chapter we complete the DSM steps by putting the truss back together through the merge step, and solving for the unknown forces and displacements.

Notes and Bibliography

The Direct Stiffness Method has been the dominant FEM version since the mid-1960s, and is the procedure followed by all major commercial codes in current use. The general DSM was developed at Boeing in the mid and late 1950s, through the leadership of Jon Turner [681,683], and had defeated its main competitor, the Force Method, by 1970 [218].

All applications-oriented FEM books cover the DSM, although the procedural steps are sometimes not clearly delineated. In particular, the textbooks recommended in §1.8.7 offer adequate expositions.

Trusses, also called bar assemblies, are usually the first structures treated in Mechanics of Materials books written for undergraduate courses in Aerospace, Civil and Mechanical Engineering. Two widely used textbooks at this level are [62] and [533].

Steps in the derivation of stiffness matrices for truss elements are well covered in a number of early treatment of finite element books, of which Chapter 5 of Przemieniecki [540] is a good example.

Force and displacement transformation matrices for structural analysis were introduced by G. Kron [384].

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 2 The Direct Stiffness Method I

EXERCISE 2.1 [D:10] Explain why *arbitrarily oriented* mechanical loads on an *idealized* pin-jointed truss structure must be applied at the joints. [Hint: idealized truss members have no bending resistance.] How about actual trusses: can they take loads applied between joints?

EXERCISE 2.2 [A:15] Show that the sum of the entries of each row of the master stiffness matrix \mathbf{K} of any plane truss, before application of any support conditions, must be zero. [Hint: apply translational rigid body motions at nodes.] Does the property hold also for the columns of that matrix?

EXERCISE 2.3 [A:15] Using matrix algebra derive (2.10) from (2.8) and (2.9). Note: Place *all equations in matrix form first* and eliminate d and F by matrix multiplication. Deriving the final form with scalar algebra and rewriting it in matrix form gets no credit.

EXERCISE 2.4 [A:15] By direct multiplication verify that for the truss member of Figure 2.9(a), $\bar{\mathbf{f}}^T \bar{\mathbf{u}} = F d$. Interpret this result physically. (Hint: what is a force times displacement in the direction of the force?)

EXERCISE 2.5 [A:20] The transformation equations between the 1-DOF spring and the 4-DOF generic truss member may be written in compact matrix form as

$$d = \mathbf{T}_d \bar{\mathbf{u}}, \quad \bar{\mathbf{f}} = F \mathbf{T}_f, \quad (\text{E2.1})$$

where \mathbf{T}_d is 1×4 and \mathbf{T}_f is 4×1 . Starting from the identity $\bar{\mathbf{f}}^T \bar{\mathbf{u}} = F d$ proven in the previous exercise, and using *compact matrix notation*, show that $\mathbf{T}_f = \mathbf{T}_d^T$. Or in words: *the displacement transformation matrix and the force transformation matrix are the transpose of each other.* (This can be extended to general systems)

EXERCISE 2.6 [A:20] Derive the equivalent spring formula $F = (EA/L) d$ of (2.8) by the Theory of Elasticity relations $e = d\bar{u}(\bar{x})/d\bar{x}$ (strain-displacement equation), $\sigma = Ee$ (Hooke's law) and $F = A\sigma$ (axial force definition). Here e is the axial strain (independent of \bar{x}) and σ the axial stress (also independent of \bar{x}). Finally, $\bar{u}(\bar{x})$ denotes the axial displacement of the cross section at a distance \bar{x} from node i , which is linearly interpolated as

$$\bar{u}(\bar{x}) = \bar{u}_{xi} \left(1 - \frac{\bar{x}}{L}\right) + \bar{u}_{xj} \frac{\bar{x}}{L} \quad (\text{E2.2})$$

Justify that (E2.2) is correct since the bar differential equilibrium equation: $d[A(d\sigma/d\bar{x})]/d\bar{x} = 0$, is verified for all \bar{x} if A is constant along the bar.

EXERCISE 2.7 [A:20] Derive the equivalent spring formula $F = (EA/L) d$ of (2.8) by the principle of Minimum Potential Energy (MPE). In Mechanics of Materials it is shown that the total potential energy of the axially loaded bar is

$$\Pi = \frac{1}{2} \int_0^L A \sigma e d\bar{x} - Fd, \quad (\text{E2.3})$$

where symbols have the same meaning as the previous Exercise. Use the displacement interpolation (E2.2), the strain-displacement equation $e = d\bar{u}/d\bar{x}$ and Hooke's law $\sigma = Ee$ to express Π as a function $\Pi(d)$ of the relative displacement d only. Then apply MPE by requiring that $\partial \Pi / \partial d = 0$.

EXERCISE 2.8 [A:20] Derive (2.17) from $\bar{\mathbf{K}}^e \bar{\mathbf{u}}^e = \bar{\mathbf{f}}^e$, (2.15) and (2.17). (*Hint: premultiply both sides of $\bar{\mathbf{K}}^e \bar{\mathbf{u}}^e = \bar{\mathbf{f}}^e$ by an appropriate matrix*). Then check by hand that using that formula you get (2.18). Falk's scheme is recommended for the multiplications.¹²

¹² This scheme is useful to do matrix multiplication by hand. It is explained in §B.3.2 of Appendix B.

EXERCISE 2.9 [D:5] Why are disconnection and localization labeled as “conceptual steps” in Figure 2.5?

EXERCISE 2.10 [C:20] (Requires thinking) Notice that the expression (2.18) of the globalized bar stiffness matrix may be factored as

$$\mathbf{K}^e = \frac{E^e A^e}{L^e} \begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix} = \begin{bmatrix} -c \\ -s \\ c \\ s \end{bmatrix} \frac{E^e A^e}{L^e} \begin{bmatrix} -c & -s & c & s \end{bmatrix} \quad (\text{E2.4})$$

Interpret this relation physically as a chain of global-to-local-to-global matrix operations: global displacements \rightarrow axial strain, axial strain \rightarrow axial force, and axial force \rightarrow global node forces.

3

The Direct Stiffness Method II

TABLE OF CONTENTS

	Page
§3.1. The Remaining DSM Steps	3–3
§3.2. Assembly: Merge	3–3
§3.2.1. Governing Rules	3–3
§3.2.2. Assembly by Hand Using Augment-and-Add	3–4
§3.3. Solution	3–6
§3.3.1. Applying Boundary Conditions by Reduction	3–6
§3.3.2. Solving for Displacements	3–7
§3.4. PostProcessing	3–7
§3.4.1. Recovery of Reaction Forces	3–8
§3.4.2. Recovery of Internal Forces and Stresses	3–8
§3.4.3. *Reaction Recovery: General Case	3–9
§3.5. *Computer Oriented Assembly and Solution	3–10
§3.5.1. *Assembly by Freedom Pointers	3–10
§3.5.2. *Applying DBC by Modification	3–11
§3.6. Prescribed Nonzero Displacements	3–11
§3.6.1. Application of Nonzero-DBC's by Reduction	3–11
§3.6.2. *Application of Nonzero-DBC's by Modification	3–13
§3.6.3. *Matrix Forms of Nonzero-DBC Application Methods	3–14
§3. Notes and Bibliography	3–14
§3. References	3–15
§3. Exercises	3–16

§3.1. The Remaining DSM Steps

Chapter 2 covered the initial steps of the DSM. The three breakdown steps: *disconnection*, *localization* and *formation of member stiffness* take us down all the way to the generic truss element: the highest level of fragmentation. This is followed by the *assembly and solution* stage.

Assembly involves *merging* the stiffness equations of each member into the global stiffness equations. For this to make sense, the member equations must be referred to a common coordinate system, which for a plane truss is the global Cartesian system $\{x, y\}$. This is done through the globalization process covered in §2.10. On the computer the formation, globalization and merge steps are done concurrently, member by member. After all members are processed we have the *free-free master stiffness equations*.

Next comes the *solution*. This process embodies two steps: applying boundary conditions (BC), and solving for the unknown joint displacements. Application of BC is done by modifying the free-free master stiffness equations to take into account which components of the joint displacements and forces are given and which are unknown.

The modified equations are submitted to a linear equation solver, which returns the unknown joint (node) displacements. As discussed under **Notes and Bibliography**, on some FEM implementations — especially programs written in the 1960s and 1970s — one or more of the foregoing operations are done concurrently.

The solution step completes the DSM proper. *Postprocessing* steps may follow, in which derived quantities such as internal forces and stresses are recovered from the displacement solution.

§3.2. Assembly: Merge

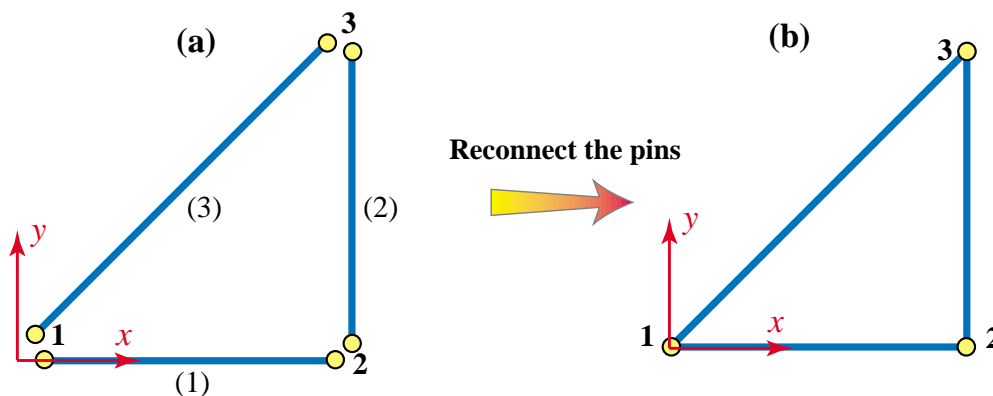


FIGURE 3.1. Physical meaning of merge operation: (a) disconnected example truss after globalization; (b) reconnected truss with the pins put back into the joints.

§3.2.1. Governing Rules

The key operation of the assembly process is the “placement” of the contribution of each member to the master stiffness equations. The process is technically called *merge* of individual members. The merge operation can be physically interpreted as *reconnecting* that member in the process of

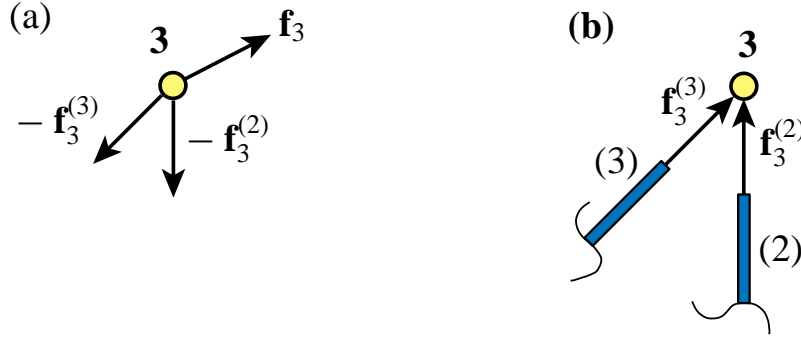


FIGURE 3.2. The force equilibrium of joint 3 of the example truss, depicted as an FBD in (a). Here \mathbf{f}_3 is the known external joint force applied on the joint. Internal forces $\mathbf{f}_3^{(2)}$ and $\mathbf{f}_3^{(3)}$ are applied by the joint on the members, as illustrated in (b). Thus the forces applied by the members on the joint are $-\mathbf{f}_3^{(2)}$ and $-\mathbf{f}_3^{(3)}$. These forces would act in the directions shown in (a) if members (2) and (3) were in tension. The free-body equilibrium statement is $\mathbf{f}_3 - \mathbf{f}_3^{(2)} - \mathbf{f}_3^{(3)} = \mathbf{0}$ or $\mathbf{f}_3 = \mathbf{f}_3^{(2)} + \mathbf{f}_3^{(3)}$. This translates into the two component equations: $f_{x3} = f_{x3}^{(2)} + f_{x3}^{(3)}$ and $f_{y3} = f_{y3}^{(2)} + f_{y3}^{(3)}$, of (3.2).

fabricating the complete structure. For a pin-jointed (model of a) truss structure, reconnection means inserting the pins back into the joints. See Figure 3.1.

Merge logic is mathematically governed by two rules of structural mechanics:

1. *Compatibility of displacements*: The displacements of all members that meet at a joint are the same.
2. *Force equilibrium*: The sum of internal forces exerted by all members that meet at a joint balances the external force applied to that joint.

(3.1)

The first rule is physically obvious: reconnected joints must move as one entity. The second one can be visualized by doing the free body diagram (FBD) of the joint, although some care is required in the separation of external and internal forces, and their signs. Notational conventions to this effect are explained in Figure 3.2 for joint 3 of the example truss, at which members (2) and (3) meet. Application of the foregoing rules at that particular joint gives

$$\text{Rule 1: } u_{x3}^{(2)} = u_{x3}^{(3)}, \quad u_{y3}^{(2)} = u_{y3}^{(3)}.$$

$$\text{Rule 2: } f_{x3} = f_{x3}^{(2)} + f_{x3}^{(3)} = f_{x3}^{(1)} + f_{x3}^{(2)} + f_{x3}^{(3)}, \quad f_{y3} = f_{y3}^{(2)} + f_{y3}^{(3)} = f_{y3}^{(1)} + f_{y3}^{(2)} + f_{y3}^{(3)}. \quad (3.2)$$

The addition of $f_{x3}^{(1)}$ to $f_{x3}^{(2)} + f_{x3}^{(3)}$ and of $f_{y3}^{(1)}$ to $f_{y3}^{(2)} + f_{y3}^{(3)}$, respectively, changes nothing because member (1) is not connected to joint 3. We are simply adding zeros. But appending those terms enables us to write the compact matrix relation for the *complete* structure:

$$\mathbf{f} = \mathbf{f}^{(1)} + \mathbf{f}^{(2)} + \mathbf{f}^{(3)}. \quad (3.3)$$

§3.2.2. Assembly by Hand Using Augment-and-Add

To directly visualize how the two rules (3.1) translate to merging logic, we first *augment* the global stiffness equations listed in §2.10.2 by adding zero rows and columns as appropriate to *complete* the force and displacement vectors.

For member (1):

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \\ f_{x3}^{(1)} \\ f_{y3}^{(1)} \end{bmatrix} = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1}^{(1)} \\ u_{y1}^{(1)} \\ u_{x2}^{(1)} \\ u_{y2}^{(1)} \\ u_{x3}^{(1)} \\ u_{y3}^{(1)} \end{bmatrix}. \quad (3.4)$$

For member (2):

$$\begin{bmatrix} f_{x1}^{(2)} \\ f_{y1}^{(2)} \\ f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 0 & 5 \end{bmatrix} \begin{bmatrix} u_{x1}^{(2)} \\ u_{y1}^{(2)} \\ u_{x2}^{(2)} \\ u_{y2}^{(2)} \\ u_{x3}^{(2)} \\ u_{y3}^{(2)} \end{bmatrix}. \quad (3.5)$$

For member (3):

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x2}^{(3)} \\ f_{y2}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 0 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & 0 & 10 & 10 \end{bmatrix} \begin{bmatrix} u_{x1}^{(3)} \\ u_{y1}^{(3)} \\ u_{x2}^{(3)} \\ u_{y2}^{(3)} \\ u_{x3}^{(3)} \\ u_{y3}^{(3)} \end{bmatrix}. \quad (3.6)$$

According to the first rule, we can *drop the member identifier* in the displacement vectors that appear in the foregoing matrix equations. Hence the reconnected member equations are

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \\ f_{x3}^{(1)} \\ f_{y3}^{(1)} \end{bmatrix} = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}, \quad (3.7)$$

$$\begin{bmatrix} f_{x1}^{(2)} \\ f_{y1}^{(2)} \\ f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 0 & 5 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}, \quad (3.8)$$

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x2}^{(3)} \\ f_{y2}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 0 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & 0 & 10 & 10 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \quad (3.9)$$

These three equations can be represented in compact matrix notation as

$$\mathbf{f}^{(1)} = \mathbf{K}^{(1)} \mathbf{u}, \quad \mathbf{f}^{(2)} = \mathbf{K}^{(2)} \mathbf{u}, \quad \mathbf{f}^{(3)} = \mathbf{K}^{(3)} \mathbf{u}. \quad (3.10)$$

According to the second rule, expressed in matrix form as (3.3), we have

$$\mathbf{f} = \mathbf{f}^{(1)} + \mathbf{f}^{(2)} + \mathbf{f}^{(3)} = (\mathbf{K}^{(1)} + \mathbf{K}^{(2)} + \mathbf{K}^{(3)}) \mathbf{u} = \mathbf{K} \mathbf{u}, \quad (3.11)$$

so all we have to do is add the three stiffness matrices that appear in (3.7) through (3.9), and we arrive at the master stiffness equations:

$$\begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \quad (3.12)$$

Using this technique *member merging* becomes simply *matrix addition*.

This explanation of the assembly process is conceptually the easiest to follow and understand. It is virtually foolproof for hand computations. However, this is *not* the way the process is carried out on the computer because it would be enormously wasteful of storage for large systems. A computer-oriented procedure is discussed in §3.5.

§3.3. Solution

Having formed the master stiffness equations we can proceed to the solution phase. To prepare the equations for a linear solver we need to separate known and unknown components of \mathbf{f} and \mathbf{u} . In this Section a technique suitable for hand computation is described.

§3.3.1. Applying Boundary Conditions by Reduction

If one attempts to solve the system (3.12) numerically for the displacements, surprise! The solution “blows up” because the coefficient matrix (the master stiffness matrix) is singular. The mathematical interpretation of this behavior is that rows and columns of \mathbf{K} are linear combinations of each other (see Remark 3.1 below). The physical interpretation of singularity is that there are unsuppressed *rigid body motions*: the truss still “floats” in the $\{x, y\}$ plane.

To eliminate rigid body motions and render the system nonsingular we must apply the physical *support conditions* as *displacement boundary conditions*. From Figure 2.4(d) we observe that the support conditions for the example truss are

$$u_{x1} = u_{y1} = u_{y2} = 0, \quad (3.13)$$

whereas the known applied forces are

$$f_{x2} = 0, \quad f_{x3} = 2, \quad f_{y3} = 1. \quad (3.14)$$

When solving the overall stiffness equations by hand, the simplest way to account for support conditions is to *remove* equations associated with known zero joint displacements from the master system. To apply (3.13) we have to remove equations 1, 2 and 4. This can be systematically accomplished by *deleting* or “striking out” rows and columns number 1, 2 and 4 from \mathbf{K} and the corresponding components from \mathbf{f} and \mathbf{u} . The reduced three-equation system is

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}. \quad (3.15)$$

Equation (3.15) is called the *reduced master stiffness system*. The coefficient matrix of this system is no longer singular.

Remark 3.1. In mathematical terms, the free-free master stiffness matrix \mathbf{K} in (3.12) has order $N = 6$, rank $r = 3$ and a rank deficiency of $d = N - r = 6 - 3 = 3$ (these concepts are summarized in Appendix C.) The dimension of the null space of \mathbf{K} is $d = 3$. This space is spanned by three independent rigid body motions: the two rigid translations along x and y and the rigid rotation about z .

Remark 3.2. Conditions (3.13) represent the simplest type of support conditions, namely specified zero displacements. More general constraint forms, such as prescribed nonzero displacements and multifreedom constraints, are handled as described in §3.6 and Chapters 8–9, respectively.

§3.3.2. Solving for Displacements

Solving the reduced system by hand (for example, via Gauss elimination) yields

$$\begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.4 \\ -0.2 \end{bmatrix}. \quad (3.16)$$

This is called a *partial displacement solution* (also *reduced displacement solution*) because it excludes known displacement components. This solution vector is *expanded* to six components by including the three specified values (3.13) in the appropriate slots:

$$\mathbf{u} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \\ -0.2 \end{bmatrix}. \quad (3.17)$$

This is the *complete displacement solution*, or simply the *displacement solution*.

§3.4. PostProcessing

The last processing step of the DSM is the solution for joint displacements. But often the analyst needs information on other mechanical quantities; for example the reaction forces at the supports, or the internal member forces. Such quantities are said to be *derived* because they are *recovered* from the displacement solution. The recovery of derived quantities is part of the so-called *postprocessing steps* of the DSM. Two such steps are described below.

§3.4.1. Recovery of Reaction Forces

Premultiplying the complete displacement solution (3.17) by \mathbf{K} we get

$$\mathbf{f} = \mathbf{K}\mathbf{u} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \\ -0.2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix} \quad (3.18)$$

This vector recovers the known applied forces (3.14), as can be expected. Furthermore we get three *reaction forces*: $f_{x1} = f_{y1} = -2$ and $f_{y2} = 1$ that are associated with the support conditions (3.13). It is easy to check that the complete force system is in self equilibrium for the free-free structure; this is the topic of Exercise 3.1. For a deeper look at reaction recovery, study §3.4.3.

§3.4.2. Recovery of Internal Forces and Stresses

Often the structural engineer is not so much interested in displacements as in *internal forces* and *stresses*. These are in fact the most important quantities for preliminary structural design. In pin-jointed trusses the only internal forces are the *axial member forces*. For the example truss these forces, denoted by $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$, are depicted in Figure 3.3. The average axial stress σ^e is obtained on dividing F^e by the cross-sectional area of the member.

The axial force F^e in member e can be obtained as follows. Extract the displacements of member e from the complete displacement solution \mathbf{u} to form \mathbf{u}^e . Then recover local joint displacements from $\bar{\mathbf{u}}^e = \mathbf{T}^e \mathbf{u}^e$.

Compute the member elongation d^e (relative axial displacement) and recover the axial force from the equivalent spring constitutive relation:

$$d^e = \bar{u}_{xj}^e - \bar{u}_{xi}^e, \quad F^e = \frac{E^e A^e}{L^e} d^e. \quad (3.19)$$

Note that \bar{u}_{yi}^e and \bar{u}_{yj}^e are not needed in computing d^e .

Example 3.1. Recover $F^{(2)}$ in example truss. Member (2) goes from node 2 to node 3 and $\varphi^{(2)} = 90^\circ$. Extract the global displacements of the member from (3.17): $\mathbf{u}^{(2)} = [u_{x2} \ u_{y2} \ u_{x3} \ u_{y3}]^T = [0 \ 0 \ 0.4 \ -0.2]^T$. Convert to local displacements using $\bar{\mathbf{u}}^{(2)} = \mathbf{T}^{(2)} \mathbf{u}^{(2)}$:

$$\begin{bmatrix} \bar{u}_{x2} \\ \bar{u}_{y2} \\ \bar{u}_{x3} \\ \bar{u}_{y3} \end{bmatrix} = \begin{bmatrix} \cos 90^\circ & \sin 90^\circ & 0 & 0 \\ -\sin 90^\circ & \cos 90^\circ & 0 & 0 \\ 0 & 0 & \cos 90^\circ & \sin 90^\circ \\ 0 & 0 & -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0.4 \\ -0.2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -0.2 \\ -0.4 \end{bmatrix}. \quad (3.20)$$

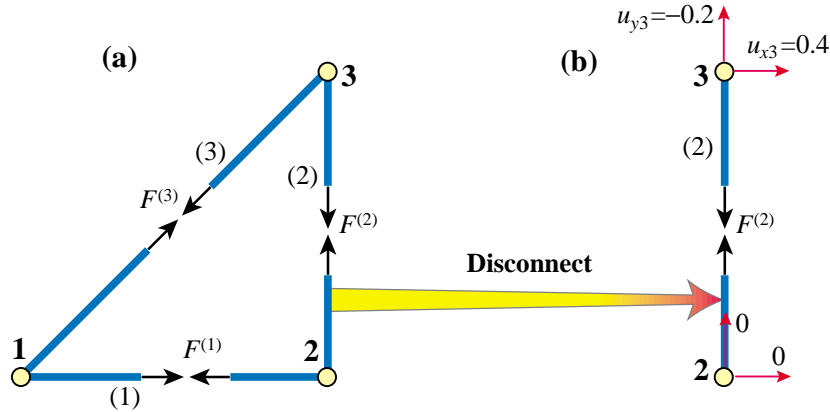


FIGURE 3.3. Internal force recovery for example truss: (a) member axial forces $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$, with arrow directions pertaining to tension; (b) details of computation for member (2).

The member elongation is $d^{(2)} = \bar{u}_{x3} - \bar{u}_{x2} = -0.2 - 0 = -0.2$, whence $F^{(2)} = (50/10) \times (-0.2) = -1$, a compressive axial force.

Following his procedure for all members provides $F^{(1)} = 0$, $F^{(2)} = -1$, and $F^{(3)} = 2\sqrt{2} = 2.82843$.

Remark 3.3. An alternative interpretation of (3.19) is to regard $e^e = d^e/L^e$ as the (average) member axial strain, $\sigma^e = E^e e^e$ as (average) axial stress, and $F^e = A^e \sigma^e$ as the axial force. This is more in tune with the Theory of Elasticity viewpoint discussed in Exercise 2.6.

§3.4.3. *Reaction Recovery: General Case

Node forces at supports recovered from $\mathbf{f} = \mathbf{K}\mathbf{u}$, where \mathbf{u} is the complete displacement solution, were called *reactions* in §3.4.1. Although the statement is correct for the example truss, it oversimplifies the general case. To cover it, consider \mathbf{f} as the superposition of applied and reaction forces:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f} = \mathbf{f}^a + \mathbf{f}^r.} \quad (3.21)$$

Here \mathbf{f}^a collects applied forces, which are known before solving, whereas \mathbf{f}^r collects unknown reaction forces to be recovered in post-processing. Entries of \mathbf{f}^r that are not constrained are set to zero. For the example truss,

$$\text{upon assembly} \quad \mathbf{f} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ 0 \\ f_{y2} \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} f_{x1} \\ f_{y1} \\ 0 \\ f_{y2} \\ 0 \\ 0 \end{bmatrix} \quad \text{upon recovery} \quad \mathbf{f} = \begin{bmatrix} -2 \\ -2 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} -2 \\ -2 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}. \quad (3.22)$$

There is a clean separation in (3.22). Every nonzero entry in \mathbf{f} comes from either \mathbf{f}^a or \mathbf{f}^r . This allows us to interpret $f_{x1} = f_{x1}^r$, $f_{y1} = f_{y1}^r$ and $f_{y2} = f_{y2}^r$ as reactions. If nonzero applied forces act directly on supported freedoms, however, a reinterpretation is in order. This often occurs when distributed loads such as pressure or own weight are *lumped* to the nodes. The adjustment can be more easily understood by following the simple example illustrated in Figure 3.4.

The fixed-free prismatic bar pictured in Figure 3.4(a) is subjected to a uniformly line load q per unit length. The bar has length L , elastic modulus E and cross-section area A . It is discretized by two equal-size elements

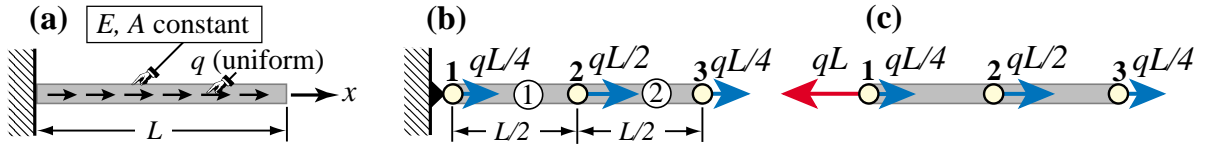


FIGURE 3.4. A simple problem to illustrate reaction recovery of support reactions when nonzero applied loads act on supports. (a) bar under distributed load; (b) two-element FEM idealization; (c) free body diagram showing applied node forces in blue and support reaction in red.

as shown in Figure 3.4(b). The three x node displacements $u_1 = u_{x1}$, $u_2 = u_{x2}$ and $u_3 = u_{x3}$ are taken as degrees of freedom. The line load is converted to node forces $f_1^a = \frac{1}{4}qL$, $f_2^a = \frac{1}{2}qL$ and $f_3^a = \frac{1}{4}qL$ at nodes 1, 2 and 3, respectively, using the EbE method discussed in Chapter 7. The master stiffness equations configured as per (3.21) are

$$\frac{2EA}{L} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \mathbf{f} = \mathbf{f}^a + \mathbf{f}^r = \frac{qL}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} + \begin{bmatrix} f_1^r \\ 0 \\ 0 \end{bmatrix}. \quad (3.23)$$

Applying the displacement BC $u_1 = 0$ and solving gives $u_2 = 3qL^2/(8EA)$ and $u_3 = qL^2/(2EA)$. Force recovery yields

$$\mathbf{f} = \mathbf{K} \mathbf{u} = \frac{qL}{4} \begin{bmatrix} -3 \\ 2 \\ 1 \end{bmatrix}, \quad \mathbf{f}^r = \mathbf{f} - \mathbf{f}^a = \frac{qL}{4} \begin{bmatrix} -3 \\ 2 \\ 1 \end{bmatrix} - \frac{qL}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} -qL \\ 0 \\ 0 \end{bmatrix}. \quad (3.24)$$

The fixed-end reaction emerges as $f_1^r = -qL$, the correctness of which may be verified on examining the FBD of Figure 3.4(c). Note that taking $f_1 = -3qL/4$ as reaction would be in error by 25%. This general recovery procedure should always be followed when reaction values are used in the design of structural supports.

§3.5. *Computer Oriented Assembly and Solution

§3.5.1. *Assembly by Freedom Pointers

The practical computer implementation of the DSM assembly process departs significantly from the “augment and add” technique described in §3.2.2. There are two major differences:

- (I) Member stiffness matrices are *not* expanded. Their entries are directly merged into those of \mathbf{K} through the use of a “freedom pointer array” called the *Element Freedom Table* or EFT.
- (II) The master stiffness matrix \mathbf{K} is stored using a special format that takes advantage of symmetry and sparseness.

Difference (II) is a more advanced topic that is deferred to the last part of the book. For simplicity we shall assume here that \mathbf{K} is stored as a *full square matrix*, and study only (I). For the example truss the freedom-pointer technique expresses the entries of \mathbf{K} as the sum

$$K_{pq} = \sum_{e=1}^3 K_{ij}^e \quad \text{for } i = 1, \dots, 4, \quad j = 1, \dots, 4, \quad p = \text{EFT}^e(i), \quad q = \text{EFT}^e(j). \quad (3.25)$$

Here K_{ij}^e denote the entries of the 4×4 globalized member stiffness matrices in (3.7) through (3.9). Entries K_{pq} that do not get any contributions from the right hand side remain zero. EFT^e denotes the Element Freedom Table for member e . For the example truss these tables are

$$\text{EFT}^{(1)} = \{1, 2, 3, 4\}, \quad \text{EFT}^{(2)} = \{3, 4, 5, 6\}, \quad \text{EFT}^{(3)} = \{1, 2, 5, 6\}. \quad (3.26)$$

Physically these tables map local freedom indices to global ones. For example, freedom number 3 of member (2) is u_{x3} , which is number 5 in the master equations; consequently $\text{EFT}^{(2)}(3) = 5$. Note that (3.25) involves three nested loops: over e (outermost), over i , and over j . The ordering of the last two is irrelevant. Advantage may be taken of the symmetry of \mathbf{K}^e and \mathbf{K} to roughly halve the number of additions. Exercise 3.5 follows the scheme (3.25) by hand.

The assembly process for general structures using this technique is studied in Chapter 25.

§3.5.2. *Applying DBC by Modification

In §3.3.1 the support conditions (3.13) were applied by reducing (3.12) to (3.15). Reduction is convenient for hand computations because it cuts down on the number of equations to solve. But it has a serious flaw for computer implementation: the equations must be rearranged. It was previously noted that on the computer the number of equations is not the only important consideration. Rearrangement can be as or more expensive than solving the equations, particularly if the coefficient matrix is stored in sparse form or on secondary storage.¹

To apply support conditions without rearranging the equations we clear (set to zero) rows and columns corresponding to prescribed zero displacements as well as the corresponding force components, and place ones on the diagonal to maintain non-singularity. The resulting system is called the *modified* set of master stiffness equations. For the example truss this approach yields

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 1 \end{bmatrix}, \quad (3.27)$$

in which rows and columns for equations 1, 2 and 4 have been cleared. Solving this modified system produces the complete displacement solution (3.17) directly.

Remark 3.4. In a “smart” stiffness equation solver the modified system need not be explicitly constructed by storing zeros and ones. It is sufficient to *mark* the equations that correspond to displacement BCs. The solver is then programmed to skip those equations. However, if one is using a standard solver from, say, a library of scientific routines or a commercial program such as *Matlab* or *Mathematica*, such intelligence cannot be expected, and the modified system must be set up explicitly.

§3.6. Prescribed Nonzero Displacements

The support conditions considered in the example truss resulted in the specification of zero displacement components; for example $u_{y2} = 0$. There are cases, however, where the known value is nonzero. This happens, for example, in the study of settlement of foundations of ground structures such as buildings and bridges, and in the analysis of motion-driven machinery components.

Mathematically these are called non-homogenous boundary conditions. The treatment of this generalization of the FEM equations is studied in the following subsections.

¹ On most modern computers, reading a floating-point number from memory at a random address takes 100 to 1000 times as long as performing a floating-point arithmetic operation on numbers that are already in registers.

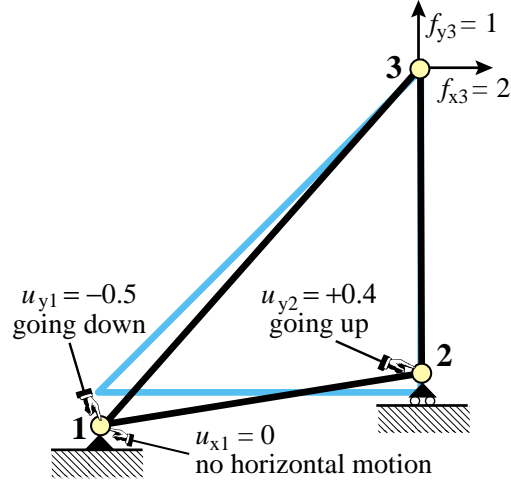


FIGURE 3.5. The example truss with prescribed nonzero vertical displacements at joints 1 and 2.

§3.6.1. Application of Nonzero-DBCs by Reduction

We describe first a matrix reduction technique, analogous to that used in §3.3.1, which is suitable for hand computations. Recall the master stiffness equations (3.12) for the example truss:

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} \quad (3.28)$$

Suppose that the applied forces are again (3.14) but the prescribed displacements change to

$$u_{x1} = 0, \quad u_{y1} = -0.5, \quad u_{y2} = 0.4 \quad (3.29)$$

This means that joint 1 goes down vertically whereas joint 2 goes up vertically, as depicted in Figure 3.5. Inserting the known data into (3.28) we get

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ -0.5 \\ u_{x2} \\ 0.4 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ 0 \\ f_{y2} \\ 2 \\ 1 \end{bmatrix} \quad (3.30)$$

The first, second and fourth rows of (3.30) are removed, leaving only

$$\begin{bmatrix} -10 & 0 & 10 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ -0.5 \\ u_{x2} \\ 0.4 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \quad (3.31)$$

Columns 1, 2 and 4 are removed by transferring all known terms from the left to the right hand side:

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} (-10) \times 0 + 0 \times (-0.5) + 0 \times 0.4 \\ (-10) \times 0 + (-10) \times (-0.5) + 0 \times 0.4 \\ (-10) \times 0 + (-10) \times (-0.5) + (-5) \times 0.4 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ -2 \end{bmatrix}. \quad (3.32)$$

These are the *reduced stiffness equations*. Note that its coefficient matrix of (3.32) is exactly the same as in the reduced system (3.15) for prescribed zero displacements. The right hand side, however, is different. It consists of the applied joint forces *modified by the effect of known nonzero displacements*. These are called the *modified node forces* or *effective node forces*. Solving the reduced system yields

$$\begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0.2 \end{bmatrix}. \quad (3.33)$$

Filling the missing entries with the known values (3.29) yields the complete displacement solution (listed as row vector to save space):

$$\mathbf{u} = [0 \quad -0.5 \quad 0 \quad 0.4 \quad -0.5 \quad 0.2]^T. \quad (3.34)$$

Taking the solution (3.34) and going through the postprocessing steps discussed in §3.4, we can find that *reaction forces and internal member forces do not change*. This is a consequence of the fact that the example truss is *statically determinate*. The force systems (internal and external) in such structures are insensitive to movements such as foundation settlements.

§3.6.2. *Application of Nonzero-DBC's by Modification

The computer-oriented modification approach follows the same idea outlined in §3.5.2. As there, the main objective is to avoid rearranging the master stiffness equations. To understand the process it is useful to think of being done in two stages. First equations 1, 2 and 4 are modified so that they become trivial equations, as illustrated for the example truss and the displacement boundary conditions (3.29):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0.4 \\ 2 \\ 1 \end{bmatrix} \quad (3.35)$$

The solution of this system recovers (3.30) by construction (for example, the fourth equation is simply $1 \times u_{y2} = 0.4$). In the next stage, columns 1, 2 and 4 of the coefficient matrix are cleared by transferring all known terms to the right hand side, following the same procedure explained in (3.33). We thus arrive at

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0.4 \\ -3 \\ -2 \end{bmatrix} \quad (3.36)$$

As before, these are called the *modified master stiffness equations*. Note that (3.36) retains the original number and order as well as matrix symmetry. Solving this system yields the complete displacement solution (3.34).

If all prescribed displacements are zero, forces on the right hand side are not modified, and one would get (3.27) as may be expected.

Remark 3.5. The modification is not actually programmed as discussed above. First the applied forces in the right-hand side are modified for the effect of nonzero prescribed displacements, and the prescribed displacements stored in the reaction-force slots. This is called the *force modification* step. Second, rows and columns of the stiffness matrix are cleared as appropriate and ones stored in the diagonal positions. This is called the *stiffness modification* step. It is essential that the procedural steps be executed in the indicated order, because stiffness terms must be used to modify forces before they are zeroed out.

§3.6.3. *Matrix Forms of Nonzero-DBC Application Methods

The reduction and modification techniques for applying DBCs can be presented in compact matrix form. First, the free-free master stiffness equations $\mathbf{Ku} = \mathbf{f}$ are partitioned as follows:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \quad (3.37)$$

In this matrix equation, subvectors \mathbf{u}_2 and \mathbf{f}_1 collect displacement and force components, respectively, that are *known, given or prescribed*. Subvectors \mathbf{u}_1 and \mathbf{f}_2 collect force and displacement components, respectively, that are *unknown*. Forces in \mathbf{f}_2 represent reactions on supports; consequently \mathbf{f}_2 is called the *reaction vector*. On transferring the known terms to the right hand side the first matrix equation becomes

$$\mathbf{K}_{11}\mathbf{u}_1 = \mathbf{f}_1 - \mathbf{K}_{12}\mathbf{u}_2. \quad (3.38)$$

This is the *reduced master equation system*. If the support B.C.s are homogeneous (that is, all prescribed displacements are zero), $\mathbf{u}_2 = \mathbf{0}$, and we do not need to change the right-hand side:

$$\mathbf{K}_{11}\mathbf{u}_1 = \mathbf{f}_1. \quad (3.39)$$

Examples that illustrate (3.38) and (3.39) are (3.32) and (3.27), respectively.

The computer-oriented modification technique retains the same joint displacement vector as in (3.38) through the following rearrangement:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 - \mathbf{K}_{12}\mathbf{u}_2 \\ \mathbf{u}_2 \end{bmatrix}. \quad (3.40)$$

This *modified system* is simply the reduced equation (3.38) augmented by the trivial equation $\mathbf{I}\mathbf{u}_2 = \mathbf{u}_2$. This system is often denoted as

$$\hat{\mathbf{K}}\mathbf{u} = \hat{\mathbf{f}}. \quad (3.41)$$

Solving (3.41) yields the complete displacement solution, including the specified displacements \mathbf{u}_2 .

For the computer implementation it is important to note that the partitioned form (3.37) is only used to allow use of compact matrix notation. In actual programming the equations are *not* explicitly rearranged: they retain their original numbers. For instance, in the example truss

$$\mathbf{u}_1 = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{y2} \end{bmatrix} \equiv \begin{bmatrix} \text{DOF \#1} \\ \text{DOF \#2} \\ \text{DOF \#4} \end{bmatrix}, \quad \mathbf{u}_2 = \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} \equiv \begin{bmatrix} \text{DOF \#3} \\ \text{DOF \#5} \\ \text{DOF \#6} \end{bmatrix}. \quad (3.42)$$

The example shows that \mathbf{u}_1 and \mathbf{u}_2 are generally interspersed throughout \mathbf{u} . Thus, matrix operations such as $\mathbf{K}_{12}\mathbf{u}_2$ involve indirect (pointer) addressing so as to avoid explicit array rearrangement.

Notes and Bibliography

The coverage of the assembly and solution steps of the DSM, along with globalization and application of BCs, is not uniform across the wide spectrum of FEM books. Authors have introduced “quirks” although the overall concepts are not affected. The most common variations arise in two contexts:

- (1) Some treatments apply support conditions *during* merge, explicitly eliminating known displacement freedoms as the elements are processed and merged into \mathbf{K} . The output of the assembly process is what is called here a reduced stiffness matrix.²
- (2) In the *frontal solution method* of Irons [361,362], assembly and solution are done concurrently. More precisely, as elements are formed and merged, displacement boundary conditions are applied, and Gauss elimination and reduction of the right hand side starts once the assembler senses (by tracking an “element wavefront”) that no more elements contribute to a certain node.

Both variants appeared in FEM programs written during the 1960s and 1970s. They were motivated by computer resource limitations of the time: memory was scarce and computing time expensive.³ On the negative side, interweaving leads to unmodular programming (which easily becomes “spaghetti code” in low-level languages such as Fortran). Since a frontal solver has to access the element library, which is typically the largest component of a general-purpose FEM program, it has to know how to pass and receive information about each element. A minor change deep down the element library can propagate and break the solver.

Squeezing storage and CPU savings on present computers is of less significance. Modularity, which simplifies scripting in higher order languages such as *Matlab* is desirable because it increases “plug-in” operational flexibility, allows the use of built-in solvers, and reduces the chance for errors. These priority changes reflect economic reality: human time is nowadays far more expensive than computer time.

A side benefit of modular assembly-solution separation is that often the master stiffness must be used in a different way than just solving $\mathbf{Ku} = \mathbf{f}$; for example in dynamics, vibration or stability analysis. Or as input to a model reduction process. In those cases the solution stage can wait.

Both the hand-oriented and computer-oriented application of boundary conditions have been presented here, although the latter is still considered an advanced topic. While hand computations become unfeasible beyond fairly trivial models, they are important from an instructional standpoint.

The augment-and-add procedure for hand assembly of the master stiffness matrix is due to H. Martin [420].

The general-case recovery of reactions, as described in §3.4.3, is not covered in any FEM textbook.

References

Referenced items have been moved to Appendix R.

² For the example truss, the coefficient matrix in (3.15) is a reduced stiffness whereas that in (3.27) is a modified one.

³ As an illustration, the first computer used by the writer, the “classical mainframe” IBM 7094, had a magnetic-core memory of 32,768 36-bit words (≈ 0.2 MB), and was as fast as an IBM PC of the mid 1980s. One mainframe, with the processing power of an iPhone, served the whole Berkeley campus. Ph.D. students were allocated 2 CPU hours per semester. Getting a moderately complex FE model through involved heavy use of slower secondary storage such as disk (or magnetic tape) in batch jobs.

Homework Exercises for Chapter 3

The Direct Stiffness Method II

EXERCISE 3.1 [A:15] Draw a free body diagram of the nodal forces (3.18) acting on the free-free truss structure, and verify that this force system satisfies translational and rotational (moment) equilibrium.

EXERCISE 3.2 [A:15] Using the method presented in §3.4.2 compute the axial forces in the three members of the example truss. Partial answer: $F^{(3)} = 2\sqrt{2}$.

EXERCISE 3.3 [A:20] Describe an alternative method that recovers the axial member forces of the example truss from consideration of joint equilibrium, without going through the computation of member deformations. Can this method be extended to arbitrary trusses?

EXERCISE 3.4 [A:20] Suppose that the third support condition in (3.13) is $u_{x2} = 0$ instead of $u_{y2} = 0$. Rederive the reduced system (3.15) for this case. Verify that this system cannot be solved for the joint displacements u_{y2} , u_{x3} and u_{y3} because the reduced stiffness matrix is singular.⁴ Offer a physical interpretation of this failure.

EXERCISE 3.5 [N:20] Construct by hand the free-free master stiffness matrix of (3.12) using the freedom-pointer technique (3.25). Note: start from \mathbf{K} initialized to the null matrix, then cycle over $e = 1, 2, 3$.

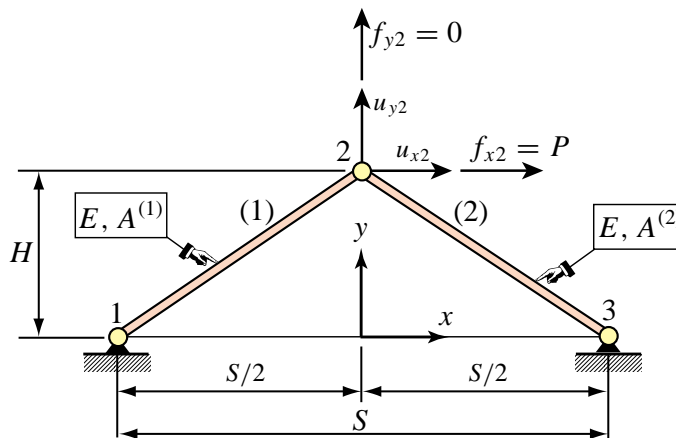


FIGURE E3.1. Truss structure for Exercises 3.6 and 3.7.

EXERCISE 3.6 [N:25] Consider the two-member arch-truss structure shown in Figure E3.1. Take span $S = 8$, height $H = 3$, elastic modulus $E = 1000$, cross section areas $A^{(1)} = 2$ and $A^{(2)} = 4$, and horizontal crown force $P = f_{x2} = 12$. Using the DSM carry out the following steps:

- Assemble the master stiffness equations. Any method: augment-and-add, or the more advanced “freedom pointer” technique explained in §3.5.1, is acceptable.
- Apply the displacement BCs and solve the reduced system for the crown displacements u_{x2} and u_{y2} . Partial result: $u_{x2} = 9/512 = 0.01758$.
- Recover the node forces at all joints including reactions. Verify that overall force equilibrium (x forces, y forces, and moments about any point) is satisfied.
- Recover the axial forces in the two members. Result should be $F^{(1)} = -F^{(2)} = 15/2$.

⁴ A matrix is singular if its determinant is zero; cf. §C.2 of Appendix C for a “refresher” in that topic.

EXERCISE 3.7 [N:20] Resolve items (a) through (c) — omitting (d) — of the problem of Exercise 3.6 if the vertical right support “sinks” so that the displacement u_{y3} is now prescribed to be -0.5 . Everything else is the same. Use the matrix reduction scheme of §3.6.1 to apply the displacement BCs.

EXERCISE 3.8 [A/C:25] Consider the truss problem defined in Figure E3.2. All geometric and material properties: L , α , E and A , as well as the applied forces P and H , are to be kept as variables. This truss has 8 degrees of freedom, with six of them removable by the fixed-displacement conditions at nodes 2, 3 and 4. Unlike previous examples, this structure is statically indeterminate as long as $\alpha \neq 0$.

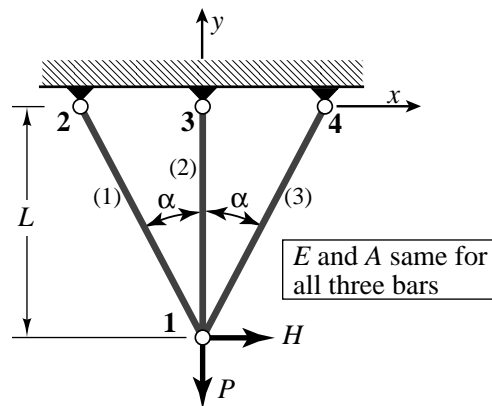


FIGURE E3.2. Truss structure for Exercise 3.8.

(a) Show that the master stiffness equations are

$$\frac{EA}{L} \begin{bmatrix} 2cs^2 & 0 & -cs^2 & c^2s & 0 & 0 & -cs^2 & -c^2s \\ & 1+2c^3 & c^2s & -c^3 & 0 & -1 & -c^2s & -c^3 \\ & & cs^2 & -c^2s & 0 & 0 & 0 & 0 \\ & & & c^3 & 0 & 0 & 0 & 0 \\ & & & & 0 & 0 & 0 & 0 \\ & & & & & 1 & 0 & 0 \\ & & & & & & cs^2 & c^2s \\ \text{symm} & & & & & & & c^3 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \\ u_{x4} \\ u_{y4} \end{bmatrix} = \begin{bmatrix} H \\ -P \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad (\text{E3.1})$$

in which $c = \cos \alpha$ and $s = \sin \alpha$. Explain from physics why the 5th row and column contain only zeros.

- (b) Apply the BCs and show the 2-equation modified stiffness system.
- (c) Solve for the displacements u_{x1} and u_{y1} . Check that the solution makes physical sense for the limit cases $\alpha \rightarrow 0$ and $\alpha \rightarrow \pi/2$. Why does u_{x1} “blow up” if $H \neq 0$ and $\alpha \rightarrow 0$?
- (d) Recover the axial forces in the three members. Partial answer: $F^{(3)} = -H/(2s) + Pc^2/(1+2c^3)$. Why do $F^{(1)}$ and $F^{(3)}$ “blow up” if $H \neq 0$ and $\alpha \rightarrow 0$?

4

Analysis of Example Truss by a CAS

TABLE OF CONTENTS

	Page
§4.1. Computer Algebra Systems	4-3
§4.1.1. Why Mathematica?	4-3
§4.1.2. How to Get It	4-3
§4.1.3. Programming Style and Prerequisites	4-4
§4.1.4. Class Demo Scripts	4-5
§4.2. Program Organization	4-6
§4.3. The Element Stiffness Module	4-7
§4.3.1. Module Description	4-7
§4.3.2. Programming Remarks	4-8
§4.3.3. Case Sensitivity	4-9
§4.3.4. Testing the Member Stiffness Module	4-9
§4.4. Merging a Member into the Master Stiffness	4-9
§4.5. Assembling the Master Stiffness	4-11
§4.6. Modifying the Master System	4-11
§4.7. Recovering Internal Forces	4-13
§4.8. Putting the Pieces Together	4-14
§4.8.1. The Driver Script	4-15
§4.8.2. Is All of This Worthwhile?	4-15
§4. Notes and Bibliography	4-17
§4. References	4-17
§4. Exercises	4-18

§4.1. Computer Algebra Systems

Computer algebra systems, known by the acronym CAS, are programs designed to perform symbolic and numeric manipulations following the rules of mathematics.¹ The development of such programs began in the mid 1960s. The first comprehensive system — the “granddaddy” of them all, called *Macsyma* (an acronym for Project **Mac** Symbolic **Manipulator**) — was developed using the programming language Lisp at MIT’s famous Artificial Intelligence Laboratory over the period 1967 to 1980.

The number and quality of symbolic-manipulation programs has expanded dramatically since the availability of graphical workstations and personal computers has encouraged interactive and experimental programming. As of this writing the leading general-purpose contenders are *Maple* and *Mathematica*.² In addition there are a dozen or so more specialized programs, some of which are available free or at very reasonable cost. See **Notes and Bibliography** at the end of the Chapter.

§4.1.1. Why Mathematica?

In the present book *Mathematica* will be used for Chapters and Exercises that develop symbolic and numerical computation for matrix structural analysis and FEM implementations. *Mathematica* is a commercial product developed by Wolfram Research, web site: <http://www.wolfram.com>. The version used to construct the code fragments presented in this Chapter is 4.1, which was commercially released in 2001. (Update: The latest version is 8, released in 2011.) The main advantages of *Mathematica* for technical computing are:

1. Availability on a wide range of platforms that range from PCs and Macs through Unix workstations. At CU Boulder, it is available through a free campus license (see §4.1).
2. Up-to-date user interface. On all machines *Mathematica* offers a graphics user interface called the Notebook front-end. This is mandatory for serious work. It provides professional typesetting for results.
3. A powerful programming language.
4. Good documentation and abundance of application books at all levels.

One common disadvantage of CAS, and *Mathematica* is no exception, is computational inefficiency in numerical calculations compared with a low-level implementation in, for instance, C or Fortran. The relative penalty can reach several orders of magnitude. For instructional use, however, the penalty is acceptable when compared to *human efficiency*. This means the ability to get FEM programs up and running in very short time, with capabilities for symbolic manipulation and graphics as a bonus.

¹ Some CAS vendors call that kind of activity “doing mathematics by computer.” It is more appropriate to regard such programs as enabling tools that help humans with complicated and error-prone manipulations. Mettle *and* metal. As of now, only humans can do mathematics.

² Another commonly used program for engineering computations: *Matlab*, does only numerical computations although a [poorly done] interface to *Maple* can be purchased as a toolbox. Historical notes: *Macsyma* died as a commercial product in 1999, although the Lisp source code of some versions is freely available. The *Maple* developer company *Waterloo Maple Inc.*, also known as *Maplesoft*, was purchased in 2009 by the Japanese software retailer Cybernet Systems. As a result, the *Matlab* symbolic toolbox is likely to be replaced.

Eligibility: students, faculty, staff and departments of all CU campuses
Platforms: Mac OSX, Windows, Linux, Solaris, AIX, HP-UX

How to Get It

1) Download and install software as per instructions at
<http://oit.colorado.edu/software-hardware/site-licenses/mathematica>
or request an installation CD from the Site Licensing office:
sitelic@colorado.edu, 303-492-8995

2) Register your copy online at <http://register.wolfram.com/> using
campus license number L2437-5121 plus your computer MathID number.
Be sure to use your CU mail address.

3) A password (unique to your computer) will be forwarded to you via
email from Site Licensing

ITS Support

Question & tech support problems: send email to sitelic@colorado.edu
Periodic hands-on workshops are available for those new to the software,
click on [Workshops](#) on the Web page given above.
Details (for example: what is a MathID?) about licensing & support are
posted at the above web site.

FIGURE 4.1. Instructions to get *Mathematica 8* for free from CU-OIT (Office of Information Technology).

§4.1.2. How to Get It

Starting 1 August 2007, a free one-year license from CU’s Information Technology Services (ITS) is available. Students may renew this license as long as they are registered. See Figure 4.1 for the “How to Get It” instructions.

If you plan to keep *Mathematica* for a longer time, an academic version is available. Registered students may also purchase the student version for about \$150 at the UMC bookstore.³ You will need to show proof you are a bona-fide student at the register.⁴ If you are not on campus (e.g. a CAETE student) you may purchase it directly at the vendor’s web site <http://www.wolfram.com>. Again proof of registration must be provided.

The student version is cheap, since the standard personal license costs over \$1K and company licenses go for over \$3K per seat. Unlike other commercial software products, you get the full thing; no capabilities are emasculated. But terms are strict: once installed on your laptop or desktop, it cannot be transferred to another computer since the license is forever keyed to the disk identification. (For example, it won’t launch if the disk is erased or replaced.)

§4.1.3. Programming Style and Prerequisites

The following material assumes that you are a moderately experienced user of *Mathematica*, or

³ This was written in 2001; probably this option does not exist anymore.

⁴ Check for discounts when new versions come out; unsold previous-version copies may go for as little as \$50.

are willing to learn to be one. See **Notes and Bibliography** for a brief discussion of tutorial and reference materials in case you are interested.

The best way to learn it from scratch is trying it as a calculator and using online help as needed. Practice with the program until you reach the level of writing functions, modules and scripts with relative ease. With the Notebook interface and a good primer it takes only a few hours. When approaching that level you may notice that functions in *Mathematica* display many aspects similar to C.⁵ You can exploit this similarity if you are proficient in that language. But *Mathematica* functions do have some unique aspects, such as matching arguments by pattern, and the fact that internal variables are global unless otherwise made local.⁶

Modification of function arguments should be avoided because it may be difficult to trace side effects. The programming style enforced here outlaws output arguments and a function can only return its name. But since the name can be a list of arbitrary objects the restriction is not serious.⁷

Our objective is to develop a symbolic program written in *Mathematica* that solves the example plane truss as well as some symbolic versions thereof. The program will rely heavily on the development and use of *functions* implemented using the `Module` construct of *Mathematica*. Thus the style will be one of procedural programming.⁸ The program will not be particularly modular (in the computer science sense) because *Mathematica* is not suitable for that programming style.⁹

The code presented in §4.2 through §4.8 uses a few language constructs that may be deemed as advanced, and these are briefly noted in the text so that appropriate reference to the *Mathematica* reference manual can be made.

§4.1.4. Class Demo Scripts

The cell scripts shown in Figures 4.2 and 4.3 will be used to illustrate the organization of a Notebook file and the “look and feel” of some basic *Mathematica* commands. These scripts will be demonstrated in class from a laptop.

⁵ Simple functions can be implemented in *Mathematica* directly, for instance `DotProduct[x_, y_] := x.y`; more complicated functions are handled by the `Module` construct. These constructs are called *rules* by computer scientists.

⁶ In *Mathematica* everything is a function, including programming constructs. Example: in C `for` is a loop-opening keyword, whereas in *Mathematica* `For` is a function that runs a loop according to its arguments.

⁷ Such restrictions on arguments and function returns are closer in spirit to C than *Fortran* although you can of course modify C-function arguments using pointers — exceedingly dangerous but often unavoidable.

⁸ The name `Module` should not be taken too seriously: it is far away from the concept of modules in Ada, Modula, Oberon or Fortran 90. But such precise levels of interface control are rarely needed in symbolic languages.

⁹ Indeed none of the CAS packages in popular use is designed for strong modularity because of historical and interactivity constraints.

Integration example

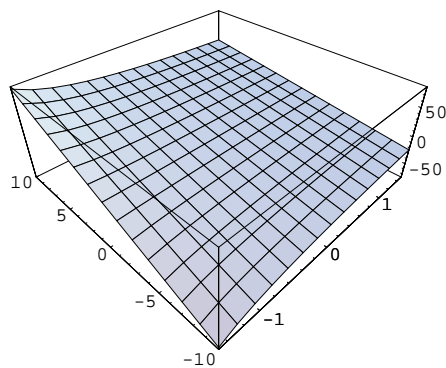
```
f[x_,α_,β_] := (1+β*x^2)/(1+α*x+x^2);
F=Integrate[f[x,-1,2],{x,0,5}];
F=Simplify[F];
Print[F]; Print[N[F]];
F=NIntegrate[f[x,-1,2],{x,0,5}];
Print["F=",F//InputForm];
```

```
10 + Log[21]
13.0445
F=13.044522437723455
```

FIGURE 4.2. Example cell for class demo.

```
Fa=Integrate[f[z,a,b],{z,0,5}]; Fa=Simplify[Fa]; Print["Fa=",Fa];
Plot3D[Fa,{a,-1.5,1.5},{b,-10,10},ViewPoint->{-1,-1,1}];
Fa=FullSimplify[Fa]; (* very slow but you get *) Print["Fa=",Fa];
```

$$\begin{aligned} Fa = & \frac{1}{2\sqrt{4-a^2}} \left(i b \log\left(\frac{ia}{\sqrt{4-a^2}} + 1\right) a^2 + i b \log\left(\frac{-ia + \sqrt{4-a^2} - 10i}{\sqrt{4-a^2}}\right) a^2 - i b \log\left(\frac{ia + \sqrt{4-a^2} + 10i}{\sqrt{4-a^2}}\right) a^2 - \right. \\ & \sqrt{4-a^2} b \log(5a+26)a + 10\sqrt{4-a^2} b - i((a^2-2)b+2) \log\left(1 - \frac{ia}{\sqrt{4-a^2}}\right) - 2ib \log\left(\frac{ia}{\sqrt{4-a^2}} + 1\right) + 2i \log\left(\frac{ia}{\sqrt{4-a^2}} + 1\right) - \\ & \left. 2ib \log\left(\frac{-ia + \sqrt{4-a^2} - 10i}{\sqrt{4-a^2}}\right) + 2i \log\left(\frac{-ia + \sqrt{4-a^2} - 10i}{\sqrt{4-a^2}}\right) + 2ib \log\left(\frac{ia + \sqrt{4-a^2} + 10i}{\sqrt{4-a^2}}\right) - 2i \log\left(\frac{ia + \sqrt{4-a^2} + 10i}{\sqrt{4-a^2}}\right) \right) \end{aligned}$$



Result after Simplify[..]



Result after FullSimplify[..]

$$Fa = -\frac{1}{2} a \log(5a+26)b + 5b - \frac{i((a^2-2)b+2) \left(\log\left(1 - \frac{ia}{\sqrt{4-a^2}}\right) - \log\left(\frac{ia}{\sqrt{4-a^2}} + 1\right) - \log\left(1 - \frac{i(a+10)}{\sqrt{4-a^2}}\right) + \log\left(\frac{i(a+10)}{\sqrt{4-a^2}} + 1\right) \right)}{2\sqrt{4-a^2}}$$

FIGURE 4.3. Another example cell for class demo. (Note: displayed results were obtained with *Mathematica* version 4.2. Integration answers from versions 5 and up are quite different.)

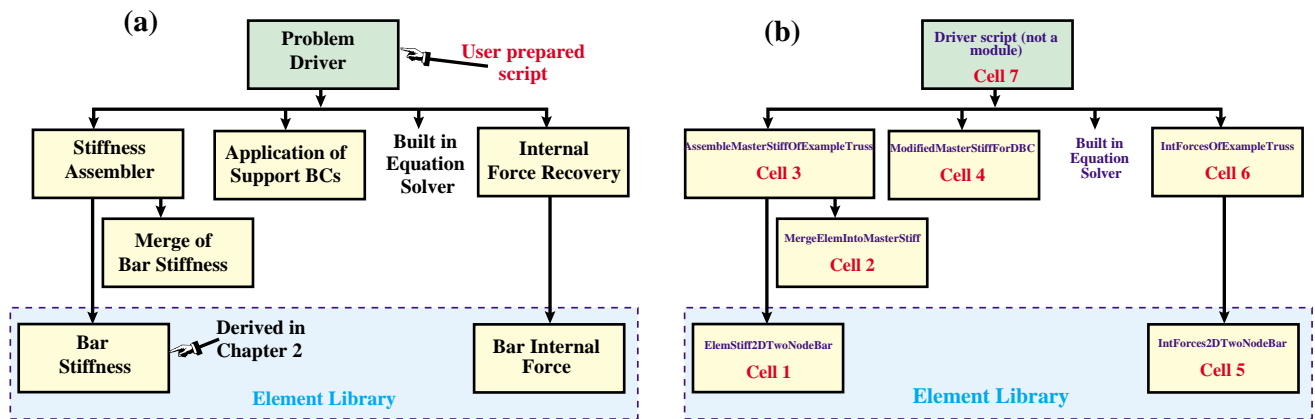


FIGURE 4.4. Example truss program: (a) organization by function; (b) organization by cell and module names.

§4.2. Program Organization

The overall organization of the example truss program is flowcharted in Figure 4.4. Figure 4.4(a) illustrates the program as divided into functional tasks. For example the driver program calls the stiffness assembler, which in turn calls two functions: form bar stiffness and merge into master stiffness. This kind of functional division would be provided by any programming language.

On the other hand, Figure 4.4(b) is *Mathematica* specific. It displays the names of the module that implement the functional tasks and the cells where they reside. (Those program objects: modules and cells, are described in the sections below.)

The following sections describe the code segments of Figure 4.4(b), one by one. For tutorial purposes this is done in a “bottom up” fashion, that is, going cell by cell, from left to right and bottom to top.

§4.3. The Element Stiffness Module

As our first FEM code segment, the top box of Figure 4.5 shows a module that evaluates and returns the 4×4 stiffness matrix of a plane truss member (two-node bar) in global coordinates. The text in that box of that figure is supposed to be placed on a Notebook cell. Executing the cell, by clicking on it and hitting an appropriate key (<Enter> on a Mac), gives the output shown in the bottom box. The contents of the figure is described in further detail below.

§4.3.1. Module Description

The stiffness module is called `ElemStiff2DTwoNodeBar`. Such descriptive names are permitted by the language. This reduces the need for detailed comments.

The module takes two arguments:

`{{x1,y1},{x2,y2}}` A two-level list¹⁰ containing the $\{x, y\}$ coordinates of the bar end

¹⁰ A level-one list is a sequence of items enclosed in curly braces. For example: $\{x1,y1\}$ is a list of two items. A level-two list is a list of level-one lists. An important example of a level-two list is a matrix.

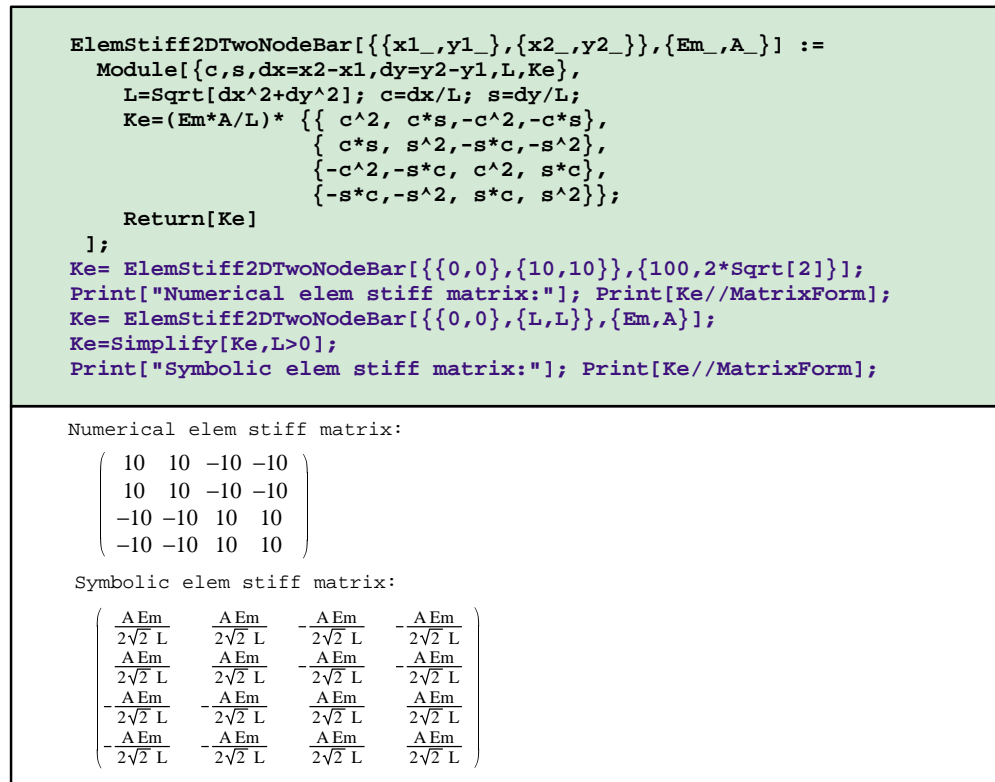


FIGURE 4.5. Module `ElemStiff2DTwoNodeBar` to form the element stiffness of a 2D 2-node truss element in global coordinates. Test statements (in blue) and test output.

nodes labelled as 1 and 2.¹¹

`{ Em, A }`

A one-level list containing the bar elastic modulus, E and the member cross section area, A . See §4.3.3 as to why name E cannot be used.

The use of the underscore after argument item names in the declaration of the `Module` is a requirement for pattern-matching in *Mathematica*. If, as recommended, you have learned functions and modules this language-specific feature should not come as a surprise.

The module name returns the 4×4 member stiffness matrix internally called `Ke`. The logic that leads to the formation of that matrix is straightforward and need not be explained in detail. Note, however, the elegant direct declaration of the matrix `Ke` as a level-two list, which eliminates the fiddling around with array indices typical of low-level programming languages. The specification format in fact closely matches the mathematical expression given as (2.18) in Chapter 2.

§4.3.2. Programming Remarks

The function in Figure 4.5 uses several intermediate variables with short names: `dx`, `dy`, `s`, `c` and `L`. It is strongly advisable to make these symbols *local* to avoid potential names clashes somewhere else.¹² In the `Module[...]` construct this is done by listing those names in a list immediately

¹¹ These are called the *local node numbers*, and replace the i, j of previous Chapters. This is a common FEM programming practice.

¹² The “global by default” choice is the worst one, but we must live with the rules of the language.

after the opening bracket. Local variables may be *initialized* when they are constants or simple functions of the argument items; for example on entry to the module `dx=x2-x1` initializes variable `dx` to be the difference of x node coordinates, namely $\Delta x = x_2 - x_1$.

The `Return` statement fulfills the same purpose as in C or Fortran 90. *Mathematica* guides and textbooks advise against the use of that and other C-like constructs. The writer strongly disagrees: the `Return` statement makes clear what the `Module` gives back to its invoker and is self-documenting.

§4.3.3. Case Sensitivity

Mathematica, like most recent computer languages, is case sensitive so that for instance `E` is not the same as `e`. This is fine. But the language designer decided that names of system-defined objects such as built-in functions and constants must begin with a capital letter. Consequently the liberal use of names beginning with a capital letter may run into clashes. For example you cannot use `E` because of its built-in meaning as the base of natural logarithms.¹³

In the code fragments presented throughout this book, identifiers beginning with upper case are used for objects such as stiffness matrices, modulus of elasticity, and cross section area. This follows established usage in Mechanics. When there is danger of clashing with a protected system symbol, additional lower case letters are used. For example, `Em` is used for the elastic modulus instead of `E` because (as noted above) the latter is a reserved symbol.

§4.3.4. Testing the Member Stiffness Module

Following the definition of `ElemStiff2DTwoNodeBar` in Figure 4.5 there are several statements that constitute the *module test script*. This script calls the module and prints returned results. Two cases are tested. First, the stiffness of member (3) of the example truss, using all-numerical values. Next, some of the input arguments for the same member are given symbolic names so they stand for variables. For example, the elastic module is given as `Em` instead of 100 as in the foregoing test. The print output of the test is shown in the lower portion of Figure 4.5.

The first test returns the member stiffness matrix (2.21), as may be expected. The second test returns a symbolic form in which three symbols appear: the coordinates of end node 2, which is taken to be located at $\{L, L\}$ instead of $\{10, 10\}$, `A`, which is the cross-section area and `Em`, which is the elastic modulus. Note that the returning matrix `Ke` is subject to a `Simplify` step before printing it, the reason for which is the subject of an Exercise. The ability to carry along variables is of course a fundamental capability of any CAS, and the main reason for which such programs are used.

§4.4. Merging a Member into the Master Stiffness

The next code fragment, listed in Figure 4.6, is used in the assembly step of the DSM. Module `MergeElemIntoMasterStiff` receives the 4×4 element stiffness matrix formed by `FormElemStiff2DNodeBar` and “merges” it into the master stiffness. It takes three arguments:

`Ke` The 4×4 member stiffness matrix to be merged. This is a level-two list.

¹³ In retrospect this appears to have been a highly questionable decision. System defined names should have been identified by a reserved prefix or postfix to avoid surprises, as done in *Macsyma* or *Maple*. *Mathematica* issues a warning message, however, if an attempt to redefine a “protected symbol” is made.

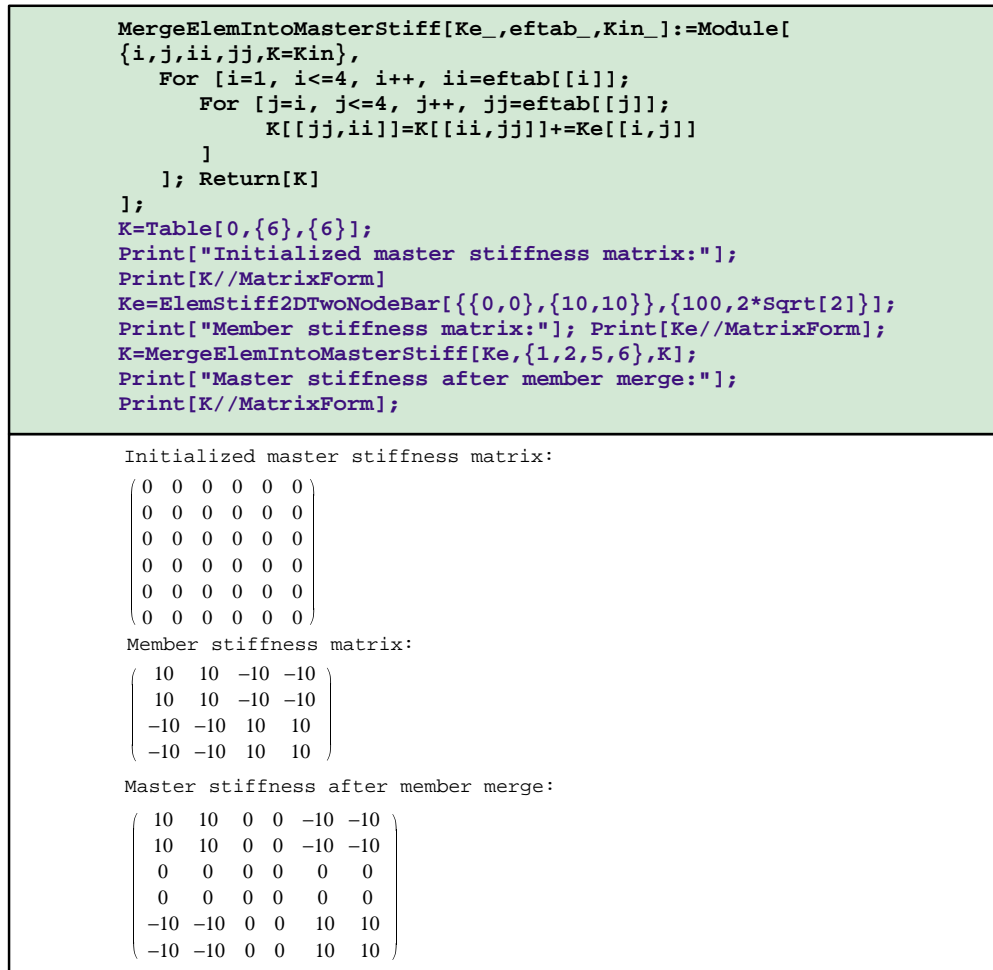


FIGURE 4.6. Module `MergeElemIntoMasterStiff` to merge a 4×4 bar element stiffness into the master stiffness matrix. Test statements (in blue) and test output.

eftab The column of the Element Freedom Table or EFT, defined in §3.5.1, appropriate to the member being merged. Recall that the EFT lists the global equation numbers for the four member degrees of freedom, cf. (3.26) in Chapter 3. This is a level-one list consisting of 4 integers.

Kinp The incoming 6×6 master stiffness matrix. This is a level-two list.

`MergeElemIntoMasterStiff` returns, as module name, the updated master stiffness matrix internally called `K` with the member stiffness merged in. We encounter here a novelty: an input-output argument. Because a formal argument cannot be modified, the situation is handled by copying the incoming `Kinp` into `K` (a local variable declared in the list that follows the `Module` name) on entry. That is the copy updated and returned via the `Return` statement. The implementation has a strong C flavor with two nested `For` loops. Because the iterators are very simple, nested `Do` loops could have been used as well.

The statements after the module provide a simple test. Before the first call to this function, the master stiffness matrix must be initialized to a zero 6×6 array. This is done in the first test statement using the `Table` function. The test member stiffness matrix is that of member (3) of the example

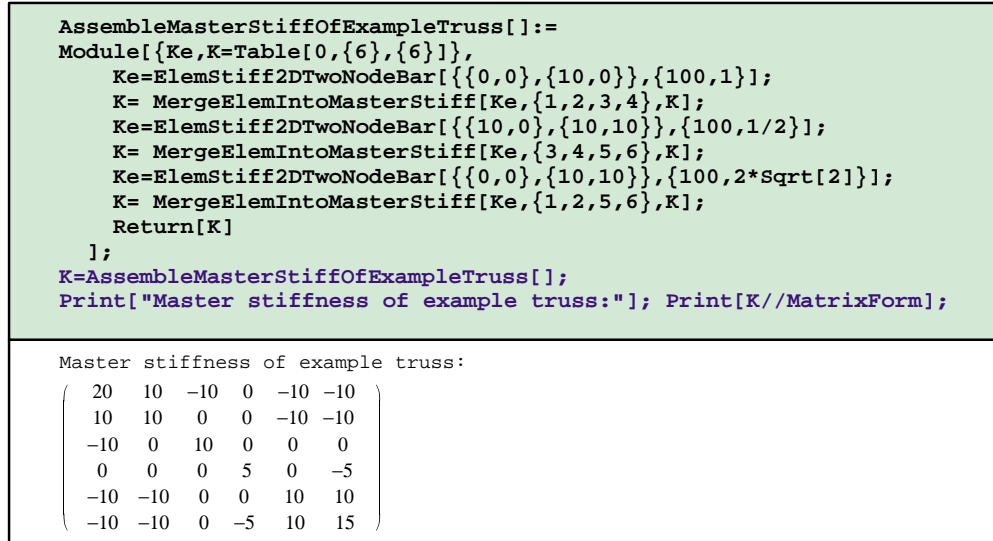


FIGURE 4.7. Module `AssembleMasterStiffOfExampleTruss` that forms the 6×6 master stiffness matrix of the example truss. Test statements (in blue) and test output.

truss, and is obtained by calling `ElemStiff2DTwoNodeBar`. The EFT is $\{1, 2, 5, 6\}$ since element freedoms 1,2,3,4 map into global freedoms 1,2,5,6. Running the test statements yields the listing given in Figure 4.6. The output is as expected.

§4.5. Assembling the Master Stiffness

The module `AssembleMasterStiffOfExampleTruss`, listed in the top box of Figure 4.7, makes use of the foregoing two modules: `ElemStiff2DTwoNodeBar` and `MergeElemIntoMasterStiff`, to form the master stiffness matrix of the example truss. The initialization of the stiffness matrix array in `K` to zero is done by the `Table` function of Mathematica, which is handy for initializing lists. The remaining statements are self explanatory. The module is similar in style to C functions with no arguments. All the example truss data is “wired in.”

The output from the test script is shown in the lower box of Figure 4.7. The output stiffness matches that in Equation (3.12), as can be expected if all code fragments used so far work correctly.

§4.6. Modifying the Master System

Following the assembly process the master stiffness equations $\mathbf{Ku} = \mathbf{f}$ must be modified to account for single-freedom displacement boundary conditions. This is done through the computer-oriented equation modification process outlined in §3.5.2.

Module `ModifiedMasterStiffForDBC` and `ModifiedMasterForcesForDBC` carry out this modification for \mathbf{K} and \mathbf{f} , respectively. These two modules are listed in the top box of Figure 4.8, along with test statements. The logic of both functions is considerably simplified by assuming that *all prescribed displacements are zero*. That is, the BCs are homogeneous. (The more general case of nonzero prescribed values is treated in Part III of the book.) The module has two arguments:

- pdof A list of the prescribed degrees of freedom identified by their global number. For the example truss this list contains three entries: $\{1, 2, 4\}$.

```

ModifiedMasterStiffForDBC[pdof_,K_] := Module[
  {i,j,k,nk=Length[K],np=Length[pdof],Kmod=K},
  For [k=1,k<=np,k++, i=pdof[[k]]];
    For [j=1,j<=nk,j++, Kmod[[i,j]]=Kmod[[j,i]]=0];
    Kmod[[i,i]]=1];
  Return[Kmod]
];
ModifiedMasterForcesForDBC[pdof_,f_] := Module[
  {i,k,np=Length[pdof],fmod=f},
  For [k=1,k<=np,k++, i=pdof[[k]]]; fmod[[i]]=0];
  Return[fmod]
];
K=Array[Kij,{6,6}]; Print["Assembled master stiffness:"];
Print[K//MatrixForm];
K=ModifiedMasterStiffForDBC[{1,2,4},K];
Print["Master stiffness modified for displacement B.C.:"];
Print[K//MatrixForm];
f=Array[fi,{6}]; Print["Force vector:"]; Print[f];
f=ModifiedMasterForcesForDBC[{1,2,4},f];
Print["Force vector modified for displacement B.C.:"]; Print[f];

```

```

Assembled master stiffness:
( Kij[1, 1] Kij[1, 2] Kij[1, 3] Kij[1, 4] Kij[1, 5] Kij[1, 6] )
( Kij[2, 1] Kij[2, 2] Kij[2, 3] Kij[2, 4] Kij[2, 5] Kij[2, 6] )
( Kij[3, 1] Kij[3, 2] Kij[3, 3] Kij[3, 4] Kij[3, 5] Kij[3, 6] )
( Kij[4, 1] Kij[4, 2] Kij[4, 3] Kij[4, 4] Kij[4, 5] Kij[4, 6] )
( Kij[5, 1] Kij[5, 2] Kij[5, 3] Kij[5, 4] Kij[5, 5] Kij[5, 6] )
( Kij[6, 1] Kij[6, 2] Kij[6, 3] Kij[6, 4] Kij[6, 5] Kij[6, 6] )
Master stiffness modified for displacement B.C.:
( 1 0 0 0 0 0 )
( 0 1 0 0 0 0 )
( 0 0 Kij[3, 3] 0 Kij[3, 5] Kij[3, 6] )
( 0 0 0 1 0 0 )
( 0 0 Kij[5, 3] 0 Kij[5, 5] Kij[5, 6] )
( 0 0 Kij[6, 3] 0 Kij[6, 5] Kij[6, 6] )
Force vector:
{fi[1], fi[2], fi[3], fi[4], fi[5], fi[6]}
Force vector modified for displacement B.C.:
{0, 0, fi[3], 0, fi[5], fi[6]}

```

FIGURE 4.8. Modules `ModifiedMasterStiffForDBC` and `ModifiedMasterForceForDBC` that modify the master stiffness matrix and force vector of a truss to impose displacement BCs. Test statements (in blue) and test output.

K The master stiffness matrix **K** produced by the assembly process.

The function clears appropriate rows and columns of **K**, places ones on the diagonal, and returns the modified **K** as function value. The only slightly fancy thing in this module is the use of the *Mathematica* function `Length` to extract the number of prescribed displacement components: `Length[pdof]` here will return the value 3, which is the length of the list `pdof`. Similarly `nk=Length[K]` assigns 6 to `nk`, which is the order of matrix **K**. Although for the example truss these values are known *a priori*, the use of `Length` serves to illustrate a technique that is heavily used in more general code.

Module `ModifiedMasterForcesForDBC` has similar structure and logic and need not be described in detail. It is important to note that for homogeneous BCs the modules are independent of each other and may be called in any order. On the other hand, if there were nonzero prescribed displacements present the force modification must be done *before* the stiffness modification. This is because

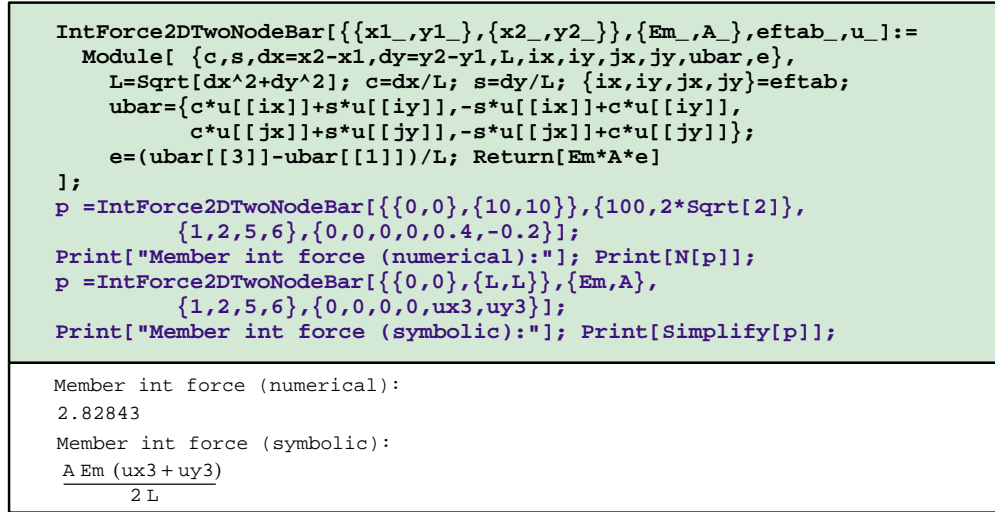


FIGURE 4.9. Module `IntForce2DTwoNodeBar` for computing the internal force in a bar element. Test statements (in blue) and test output.

stiffness coefficients that are cleared in the latter are needed for modifying the force vector.

The test statements are purposely chosen to illustrate another feature of *Mathematica*: the use of the `Array` function to generate subscripted symbolic arrays of one and two dimensions. The test output is shown in the bottom box of Figure 4.8, which should be self explanatory. The force vector and its modified form are printed as row vectors to save space.

§4.7. Recovering Internal Forces

Since *Mathematica* provides built-in matrix operations for solving a linear system of equations and multiplying matrices by vectors, we do not need to write application functions for the solution of the modified stiffness equations, and for the recovery of nodal forces as $\mathbf{f} = \mathbf{K}\mathbf{u}$. Consequently, the last application functions we need are those for internal force recovery.

Function `IntForce2DTwoNodeBar` listed in the top box of Figure 4.9 computes the internal force in an individual bar element. It is somewhat similar in argument sequence and logic to `ElemStiff2DTwoNodeBar` of Figure 4.5. The first two arguments are identical. Argument `eftab` provides the Element Freedom Table array for the element. The last argument, `u`, is the vector of computed node displacements.

The logic of `IntForce2DTwoNodeBar` is straightforward and follows the method outlined in §3.2.1. Member joint displacements $\bar{\mathbf{u}}^{(e)}$ in local coordinates $\{\bar{x}, \bar{y}\}$ are recovered in array `ubar`, then the longitudinal strain $e = (\bar{u}_{xj} - \bar{u}_{xi})/L$ and the internal (axial) force $p = EAe$ is returned as function value. As coded the function contains redundant operations because entries 2 and 4 of `ubar` (that is, components \bar{u}_{yi} and \bar{u}_{yj}) are not actually needed to get p , but were kept to illustrate the general backtransformation of global to local displacements.

Running this function with the test statements shown after the module produces the output shown in the bottom box of Figure 4.9. The first test is for member (3) of the example truss using the actual nodal displacements (3.17). It also illustrates the use of the *Mathematica* built in function

```

IntForcesOfExampleTruss[u_]:= Module[{f=Table[0,{3}]},
  f[[1]]=IntForce2DTwoNodeBar[{{0,0},{10,0}},{100,1},{1,2,3,4},u];
  f[[2]]=IntForce2DTwoNodeBar[{{10,0},{10,10}},{100,1/2},{3,4,5,6},u];
  f[[3]]=IntForce2DTwoNodeBar[{{0,0},{10,10}},{100,2*Sqrt[2]},
    {1,2,5,6},u];
  Return[f]
];
f=IntForcesOfExampleTruss[{0,0,0,0,0.4,-0.2}];
Print["Internal member forces in example truss:"];Print[N[f]];

Internal member forces in example truss:
{0., -1., 2.82843}

```

FIGURE 4.10. Module `IntForceOfExampleTruss` that computes internal forces in the 3 members of the example truss. Test statements (in blue) and test output.

`N` to produce output in floating-point form. The second test does a symbolic calculation in which several argument values are fed in variable form.

The top box of Figure 4.10 lists a higher-level function, `IntForceOfExampleTruss`, which has a single argument: `u`. This is the complete 6-vector of joint displacements \mathbf{u} . This function calls `IntForce2DTwoNodeBar` three times, once for each member of the example truss, and returns the three member internal forces thus computed as a 3-component list.

The test statements listed after `IntForcesOfExampleTruss` feed the displacement solution (3.24) to the module. Running the test produces the output shown in the lower box of Figure 4.10. The internal forces are $F^{(1)} = 0$, $F^{(2)} = -1$ and $F^{(3)} = 2\sqrt{2} = 2.82843$, in agreement with the values found in §3.4.2.

§4.8. Putting the Pieces Together

After all this development and testing effort, documented in Figures 4.5 through 4.10, we are ready to make use of all these bits and pieces of code to analyze the example plane truss. This is actually done with the logic shown in Figure 4.11. This particular piece of code is called the *driver script*. Note that it is *not* a Module. It uses the seven previously described modules

```

ElemStiff2DTwoNodeBar
MergeElemIntoMasterStiff
AssembleMasterStiffOfExampleTruss
ModifiedMasterStiffForDBC
ModifiedMasterForcesForDBC
IntForce2DTwoNodeTruss
IntForcesOfExampleTruss

```

(4.1)

These functions must have been defined (“compiled”) at the time the driver scripts described below are run. A simple way to making sure that all of them are defined is to put all these functions in the same Notebook file and to mark them as *initialization cells*. These cells may be executed by picking up Kernel → Initialize → Execute Initialization.¹⁴ (An even simpler scheme would to group them all in one cell, but that would make placing separate test statements messy.)

¹⁴ In *Mathematica* versions 6 and higher, there is an initialization shortcut.

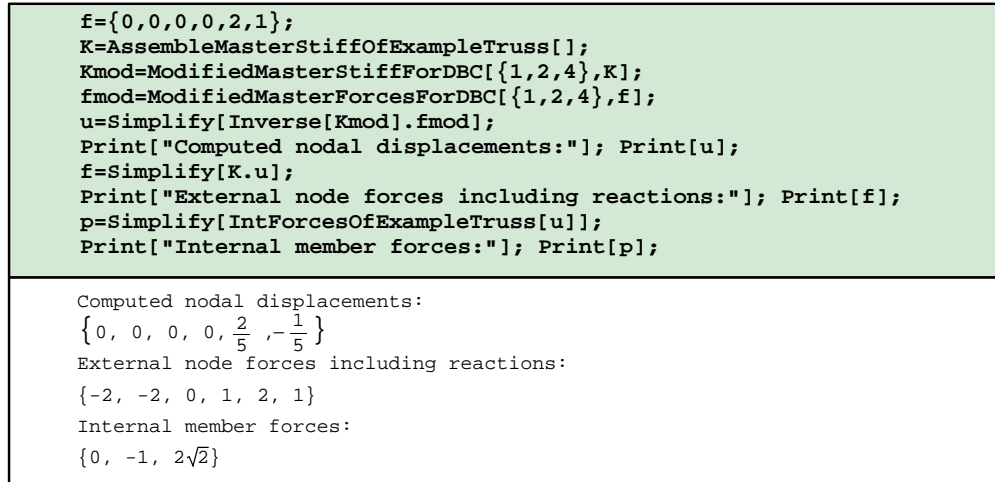


FIGURE 4.11. Driver script for numerical analysis of example truss and its output.

For a hierarchical version of (4.1), see the last CAETE slide.

§4.8.1. The Driver Script

The code listed in the top box of Figure 4.11 first assembles the master stiffness matrix through `AssembleMasterStiffOfExampleTruss`. Next, it applies the displacement boundary conditions through `ModifiedMasterStiffForDBC` and `ModifiedMasterForcesForDBC`. Note that the modified stiffness matrix is placed into `Kmod` rather than `K` to save the original form of the master stiffness for the reaction force recovery later. The complete displacement vector is obtained by the matrix calculation

$$\mathbf{u} = \text{Inverse}[\mathbf{Kmod}] \cdot \mathbf{fmod}; \quad (4.2)$$

This statement takes advantage of two built-in *Mathematica* functions. `Inverse` returns the inverse of its matrix argument.¹⁵ The dot operator denotes matrix multiplication (here, matrix-vector multiply.) The enclosing `Simplify` function in Figure 4.11 simplifies the expression of vector \mathbf{u} in case of symbolic calculations; it is actually redundant if all computations are numerical.

The remaining statements recover the node vector including reactions via the matrix-vector multiply $\mathbf{f} = \mathbf{K} \cdot \mathbf{u}$ (recall that \mathbf{K} contains the unmodified master stiffness matrix) and the member internal forces \mathbf{p} through `IntForcesOfExampleTruss`. The code prints \mathbf{u} , \mathbf{f} and \mathbf{p} as row vectors to save space.

Running the script of the top box of Figure 4.11 produces the output shown in the bottom box of that figure. The results confirm the hand calculations of Chapter 3.

§4.8.2. Is All of This Worthwhile?

At this point you may wonder whether all of this work is worth the trouble. After all, a hand calculation (typically helped by a programable calculator) would be quicker in terms of flow time. Typing and debugging the *Mathematica* fragments displayed here took the writer about six hours (although about two thirds of this was spent in editing and getting the fragment listings into the

¹⁵ This is a highly inefficient way to solve $\mathbf{K}\mathbf{u} = \mathbf{f}$ if the linear system becomes large. It is done here for simplicity.

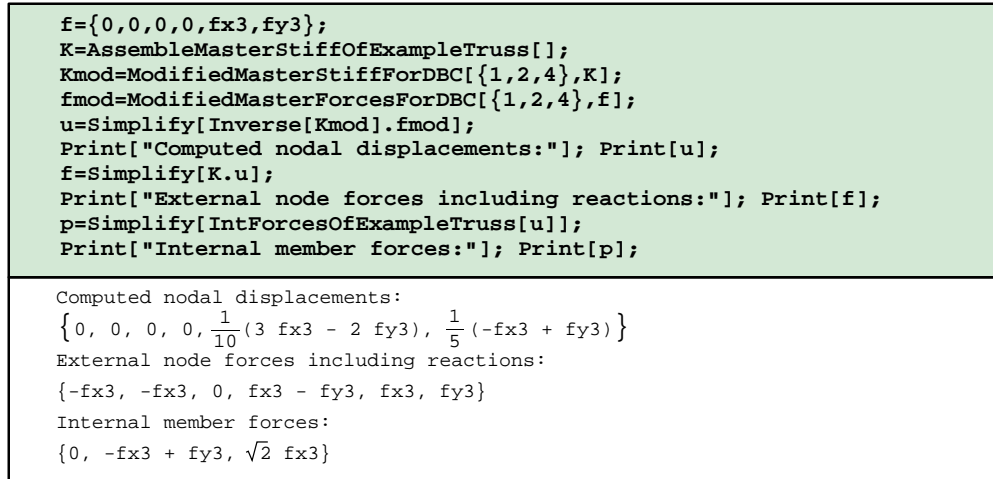


FIGURE 4.12. Driver script for symbolic analysis of example truss and its output.

Chapter.) For larger problems, however, *Mathematica* would certainly beat hand-plus-calculator computations, the cross-over typically appearing for 10 to 20 equations. For up to about 500 equations and using floating-point arithmetic, *Mathematica* gives answers within minutes on a fast PC or Mac with sufficient memory but eventually runs out of steam at about 1000 equations. For a range of 1000 to about 50000 equations, *Matlab*, using built-in sparse solvers, would be the best compromise between human and computer flow time. Beyond 50000 equations a program in a low-level language, such as C or Fortran, would be most efficient in terms of computer time.¹⁶

One distinct advantage of computer algebra systems emerges when you need to *parametrize* a small problem by leaving one or more problem quantities as variables. For example suppose that the applied forces on node 3 are to be left as f_{x3} and f_{y3} . You replace the last two components of array p as shown in the top box of Figure 4.12, execute the cell and shortly get the symbolic answer shown in the bottom box of that figure. This is the answer to an infinite number of numerical problems. Although one may try to undertake such studies by hand, the likelihood of errors grows rapidly with the complexity of the system. Symbolic manipulation systems can amplify human abilities in this regard, as long as the algebra “does not explode” because of combinatorial complexity. Examples of such nontrivial calculations will appear throughout the following Chapters.

Remark 4.1. The “combinatorial explosion” danger of symbolic computations should be always kept in mind. For example, the numerical inversion of a $N \times N$ matrix is a $O(N^3)$ process, whereas symbolic inversion goes as $O(N!)$. For $N = 48$ the floating-point numerical inverse will be typically done in a fraction of a second. But the symbolic adjoint will have $48! = 12413915592536072670862289047373375038521486354677760000000000$ terms, or $O(10^{61})$. There may be enough electrons in our Universe to store that, but barely ...

¹⁶ The current record for FEM structural applications is about 100 million equations, done on a massively parallel super-computer (ASCI Red at SNL). Fluid mechanics problems with over 500 million equations have been solved.

Notes and Bibliography

The hefty *Mathematica Book* [721] is a reference manual. Since the contents are available online (click on **Help** in topbar), buying the printed book makes little sense.¹⁷

There is a nice tutorial available by Glynn and Gray [282], list: \$35, dated 1999. (Theodore Gray invented the Notebook front-end that appeared in version 2.2.) It can be also purchased on CDROM from MathWare, Ltd, P. O. Box 3025, Urbana, IL 61208, e-mail: info@mathware.com. The CDROM is a hyperlinked version of the book that can be installed on the same directory as *Mathematica*. More up to date and comprehensive is the recent appeared *Mathematica Navigator* in two volumes [586,587]; list: \$69.95 but used copies are discounted, down to about \$20. The recently appeared *Mathematica Cookbook* by S. Mangano, published by O'Reilly, is being reviewed as of this writing; if useful, it will be placed in the reference list.

Beyond these, there are many books at all levels that expound on the use of *Mathematica* for various applications ranging from pure mathematics to physics and engineering. A web search on September 2003 done on www3.addall.com hit 150 book titles containing *Mathematica*, compared to 111 for *Maple* and 148 for *Matlab*. A google search (August 2005) hits 3,820,000 pages containing *Mathematica*, but here *Matlab* wins with 4,130,000.

Wolfram Research hosts the MathWorld web site at <http://mathworld.wolfram.com>, maintained by Eric W. Weisstein. It is essentially a hyperlinked, on-line version of his *Encyclopædia of Mathematics* [705].

To close the topic of symbolic versus numerical computation, here is a nice summary by A. Grozin, posted on the Usenet:

“Computer Algebra Systems (CASs) are programs [that] operate with formulas. Mathematica is a powerful CAS (though quite expensive). Other CASs are, e.g., Maple, REDUCE, MuPAD <...>. There are also quite powerful free CASs: Maxima and Axiom. In all of these systems, it is possible to do some numerical calculations (e.g., to evaluate the formula you have derived at some numerical values of all parameters). But it is a very bad idea to do large-scale numerical work in such systems: performance will suffer. In some special cases (e.g., numerical calculations with very high precision, impossible at the double-precision level), you can use Mathematica to do what you need, but there are other, faster ways to do such things.

There are a number of programs to do numerical calculations with usual double-precision numbers. One example is Matlab; there are similar free programs, e.g., Octave, Scilab, R, ... Matlab is very good and fast in doing numerical linear algebra: if you want to solve a system of 100 linear equations whose coefficients are all numbers, use Matlab; if coefficients contain letters (symbolic quantities) and you want the solution as formulas, use Mathematica or some other CAS. Matlab can do a limited amount of formula manipulations using its “symbolic toolbox,” which is an interface to a cut-down Maple. It’s a pain to use this interface: if you want Maple, just use Maple.”

References

Referenced items have been moved to Appendix R.

¹⁷ The fifth edition, covering version 5, lists for \$49.95 but older editions are heavily discounted on the web, some under \$2. There are no printed manual versions for version 6 and higher.

Homework Exercises for Chapter 4

Analysis of Example Truss by a CAS

Before doing any of these Exercises, download the *Mathematica* Notebook file `ExampleTruss.nb` from the course web site. (Go to Chapter 4 Index and click on the link). Open this Notebook file using version 4.1 or later one. The first six cells contain the modules and test statements listed in the top boxes of Figures 4.5 through 4.10. These six are marked as *initialization cells*. Before running driver scripts, they should be executed by picking up Kernel → Evaluation → Execute Initialization (or equivalent in newer versions).

Verify that the output of those six cells agrees with that shown in the bottom boxes of those figures. Then execute the driver script in Cell 7 by clicking on the cell and pressing the appropriate key: <Enter> on a Mac, <Shift-Enter> on a Windows PC. Compare the output with that shown in 4.11. If the output checks out, you may proceed to the Exercises.

EXERCISE 4.1 [C:10] Explain why the `Simplify` command in the test statements of Figure 4.5 says $L > 0$. (One way to figure this out is to just say `Ke=Simplify[Ke]` and look at the output. Related question: why does *Mathematica* refuse to simplify `Sqrt[Lexp2]` to L unless one specifies the sign of L in the `Simplify` command?

EXERCISE 4.2 [C:10] Explain the logic of the `For` loops in the merge function `MergeElemIntoMasterStiff` of Figure 4.6. What does the operator `+=` do? (If you are a C programmer, all of this should be easy.)

EXERCISE 4.3 [C:10] Explain the reason behind the use of `Length` in the modules of Figure 4.6. Why not simply set `nk` and `np` to 6 and 3, respectively?

EXERCISE 4.4 [C:15] Of the seven modules listed in Figures 4.5 through 4.10, with names collected in (4.1), two can be used only for the example truss, three can be used for any plane truss, and two can be used for other structures analyzed by the DSM. Identify which ones, and briefly state the reasons for your classification.

EXERCISE 4.5 [C:20] Modify the modules `AssembleMasterStiffOfExampleTruss`, `IntForcesOfExampleTruss`, and the driver script of Figure 4.11, to solve numerically the three-node, two-member truss of Exercise 3.6. Verify that the output reproduces the solution given for that problem. Procedural recommendation: modify cells but keep a copy of the original Notebook handy in case things go wrong.

EXERCISE 4.6 [C:25] Expand the logic of `ModifiedMasterForcesForDBC` to permit specified nonzero displacements. Specify these in a second argument called `pval`, which contains a list of prescribed values paired with `pdof`.

```
xynode={{0,0},{10,0},{10,10}}; elenod={{1,2},{2,3},{3,1}};
unode={{0,0},{0,0},{2/5,-1/5}}; amp=5; p={};
For [t=0,t<=1,t=t+1/5,
  For [e=1,e<=Length[elenod],e++, {i,j}=elenod[[e]];
    xyi=xynode[[i]];ui=unode[[i]];xyj=xynode[[j]];uj=unode[[j]];
    p=AppendTo[p,Graphics[Line[{xyi+amp*t*ui,xyj+amp*t*uj}]]];
  ];
];
Show[p,Axes->False,AspectRatio->Automatic];
```

FIGURE E4.1. Mystery script for Exercise 4.7.

EXERCISE 4.7 [C:20] Explain what the program of Figure E4.1 does, and the logic behind what it does. (You may want to put it in a cell and execute it.) What modifications would be needed so it can be used for any plane struss?

5

Constructing MoM Members

TABLE OF CONTENTS

	Page
§5.1. Introduction	5–3
§5.2. Formulation of MoM Members	5–3
§5.2.1. What They Look Like	5–3
§5.2.2. End Quantities, Degrees of Freedom, Joint Forces	5–4
§5.2.3. Internal Quantities	5–4
§5.2.4. Discrete Field Equations, Tonti Diagram	5–5
§5.3. Simplex MoM Members	5–6
§5.3.1. The Bar Element Revisited	5–7
§5.3.2. The Spar Element	5–8
§5.3.3. The Shaft Element	5–9
§5.4. *Non-Simplex MoM Members	5–11
§5.4.1. *Formulation Rules	5–11
§5.4.2. *Examples	5–12
§5. Notes and Bibliography.	5–13
§5. References	5–13
§5. Exercises	5–14

§5.1. Introduction

The truss member used as example in Chapters 2–4 is an instance of a *structural element*. Such elements may be formulated directly using concepts and modeling techniques developed in Mechanics of Materials (MoM).¹ The construction does not involve the more advanced tools that are required for the continuum finite elements that appear in Part II.

This Chapter presents an overview of the technique to construct the element stiffness equations of “MoM members” using simple matrix operations. These simplified equations come in handy for a surprisingly large number of applications, particularly in skeletal structures. Focus is on *simplex elements*, which may be formed directly as a sequence of matrix operations. Non-simplex elements are presented as a recipe because their proper formulation requires work theorems not yet studied.

The physical interpretation of the FEM is still emphasized. Consequently we continue to speak of structures built up of *members* (elements) connected at *joints* (nodes).

§5.2. Formulation of MoM Members

§5.2.1. What They Look Like

MoM-based formulations are largely restricted to *intrinsically one-dimensional members*. These are structural components one of whose dimensions, called *longitudinal*, is significantly larger than the other two, called the *transverse dimensions*. Such members are amenable to the simplified structural theories developed in MoM textbooks. This Chapter covers only *straight* members with geometry defined by the two end joints.² The member *cross sections* are defined by the intersection of planes normal to the longitudinal dimension with the member. See Figure 5.1. Note that although the individual member will be idealized as being one-dimensional in its intrinsic or local coordinate system, it often functions as component of a two- or three-dimensional structure.

This class of structural components embodies bars, beams, beam-columns, shafts and spars. Although geometrically similar, the names distinguish the main kind of internal forces the member resists and transmits: axial forces for bars, bending and shear forces for beams, axial compression and bending for beam-columns, torsion forces for shafts, and shear forces for spars.

Members are connected at their end joints by displacement degrees of freedom. For truss (bar) and spar members those freedoms are translational components of the joint displacements. For other types, notably beams and shafts, nodal rotations are chosen as additional degrees of freedom.

Structures fabricated with MoM members are generally three-dimensional. Their geometry is defined with respect to a global Cartesian coordinate system $\{x, y, z\}$. Two-dimensional idealizations are useful simplifications should the nature of the geometry and loading allow the reduction of the structural model to one plane of symmetry, which is chosen to be the $\{x, y\}$ plane. Plane trusses and plane frameworks are examples of such simplifications.

¹ Mechanics of Materials was called Strength of Materials in older texts. It covers bars, beams, shafts, arches, thin plates and shells, but only one-dimensional models are considered in introductory undergraduate courses. MoM involves *ab initio* phenomenological assumptions such as “plane sections remain plane” or “shear effects can be neglected in thin beams.” These came about as the byproduct of two centuries of structural engineering practice, justified by success. A similar acronym (MOM) is used in Electrical Engineering for something completely different: the Method of Moments.

² Advanced Mechanics of Materials includes curved members. Plane arch elements are studied in Chapter 13.

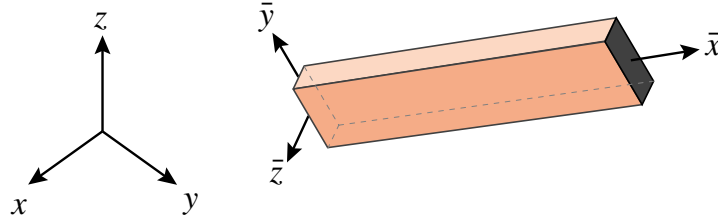


FIGURE 5.1. A Mechanics of Materials (MoM) member is a structural element one of whose dimensions (the longitudinal dimension) is significantly larger than the other two. Local axes $\{\bar{x}, \bar{y}, \bar{z}\}$ are chosen as indicated. Although the depicted member is prismatic, some applications utilize tapered or stepped members, the cross section of which varies as a function of \bar{x} .

In this Chapter we study generic structural members that fit the preceding class. An individual member is identified by e but this superscript will be usually suppressed in the equations below to reduce clutter. The local axes are denoted by $\{\bar{x}, \bar{y}, \bar{z}\}$, with \bar{x} along the longitudinal direction. See Figure 5.1.

The mathematical model of a MoM member is obtained by an idealization process. The model represents the member as a line segment that connects the two end joints, as depicted in Figure 5.2.

§5.2.2. End Quantities, Degrees of Freedom, Joint Forces

The set of mathematical variables used to link members are called *end quantities* or *connectors*. In the Direct Stiffness Method (DSM) these are joint displacements (the degrees of freedom) and the joint forces. These quantities are related by the member stiffness equations.

The degrees of freedom at the end joints i and j are collected in the joint displacement vector $\bar{\mathbf{u}}$. This may include translations only, rotations only, or a combination of translations and rotations.

The vector of joint forces $\bar{\mathbf{f}}$ groups components in one to one correspondence with $\bar{\mathbf{u}}$. Component pairs must be *conjugate* in the sense of the Principle of Virtual Work. For example if the \bar{x} -translation at joint i : \bar{u}_{xi} appears in $\bar{\mathbf{u}}$, the corresponding entry in $\bar{\mathbf{f}}$ is the \bar{x} -force \bar{f}_{xi} at i . If the rotation about \bar{z} at joint j : $\bar{\theta}_{zj}$ appears in $\bar{\mathbf{u}}$, the corresponding entry in $\bar{\mathbf{f}}$ is the \bar{z} -moment \bar{m}_{zj} .

§5.2.3. Internal Quantities

Internal quantities are mechanical actions that take place within the member. Those actions involve stresses and deformations. Accordingly two types of internal quantities appear:

Internal member forces form a finite set of stress-resultant quantities collected in an array \mathbf{p} . They are obtained by integrating stresses over each cross section, and thus are also called generalized stresses in structural mechanics. This set characterizes the forces resisted by the material. Stresses at any point in a section may be recovered if \mathbf{p} is known.

Member deformations form a finite set of quantities, chosen in one-to one correspondence with internal member forces, and collected in an array \mathbf{v} . This set characterizes the deformations experienced by the material. They are also called generalized strains in structural theory. Strains at any point in the member can be recovered if \mathbf{v} is known.

As in the case of end quantities, internal forces and deformations are paired in one to one correspondence. For example, the axial force in a bar member must be paired either with an average

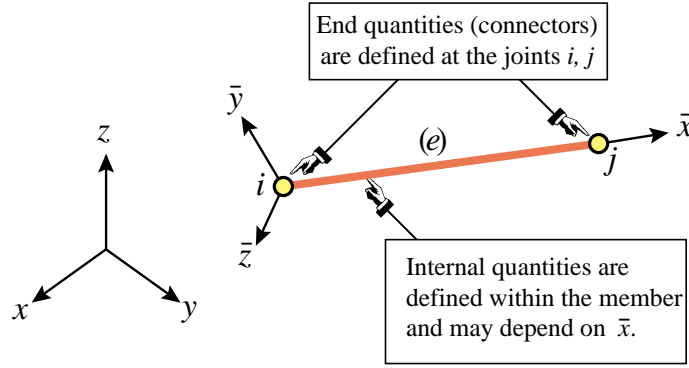


FIGURE 5.2. The FE mathematical idealization of a MoM member. The model is one-dimensional in \bar{x} . The two end joints are the site of end quantities: joint forces and displacements, that interconnect members. Internal quantities characterize internal forces, stresses and deformations in the member.

axial deformation, or with the total elongation. Pairs that mutually correspond in the sense of the Principle of Virtual Work are called *conjugate*. Unlike the case of end quantities, conjugacy of internal quantities is not a mandatory requirement although it simplifies some expressions.

§5.2.4. Discrete Field Equations, Tonti Diagram

The matrix equations that connect $\bar{\mathbf{u}}$, \mathbf{v} , \mathbf{p} and $\bar{\mathbf{f}}$ are called the *discrete field equations*. There are three of them.

The member deformations \mathbf{v} are linked to the joint displacements $\bar{\mathbf{u}}$ by the kinematic compatibility conditions, also called the deformation-displacement or strain-displacement equations:

$$\mathbf{v} = \mathbf{B} \bar{\mathbf{u}}. \quad (5.1)$$

The internal member forces are linked to the member deformations by the constitutive equations. In the absence of initial strain effects those equations are homogeneous:

$$\mathbf{p} = \mathbf{S} \mathbf{v}. \quad (5.2)$$

Finally, the internal member forces are linked to the joint forces by the equilibrium equations. If the internal forces \mathbf{p} are *constant over the member*, the relation is simply

$$\bar{\mathbf{f}} = \mathbf{A}^T \mathbf{p}. \quad (5.3)$$

In (5.3) the transpose of \mathbf{A} is used for convenience.³

The foregoing equations can be presented graphically as shown in Figure 5.3. This is a discrete version of the so-called *Tonti diagrams*, which represent governing equations as arrows linking boxes containing kinematic and static quantities. Tonti diagrams for field (continuum) equations are introduced in Chapter 11.

³ If \mathbf{p} is a function of \bar{x} the relation is of differential type. This form is studied in §5.4.

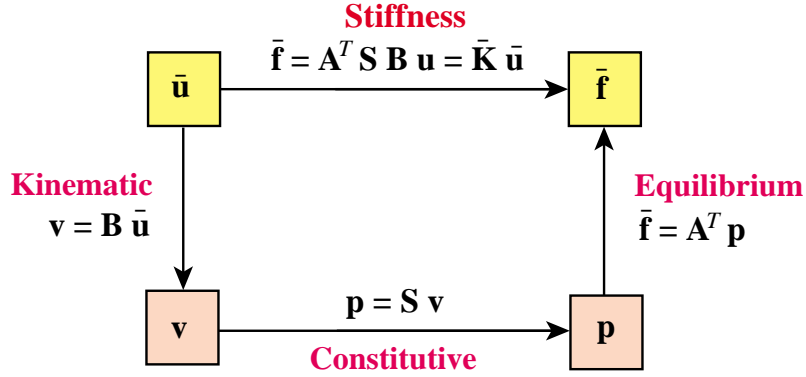


FIGURE 5.3. Tonti diagram of the three discrete field equations (5.1)–(5.3) and the stiffness equation (5.4) for a *simplex* MoM member. Internal and end quantities appear inside the orange and yellow boxes, respectively.

Matrices **B**, **S** and **A** receive the following names in the literature:

- A** Equilibrium, leverage
- S** Rigidity, material, constitutive⁴
- B** Compatibility, deformation-displacement, strain-displacement

If the element is sufficiently simple, the determination of these three matrices can be carried out through MoM techniques. If the construction requires more advanced tools, however, recourse to the general methodology of finite elements and variational principles is necessary.

§5.3. Simplex MoM Members

Throughout this section we assume that the *internal quantities are constant over the member length*. Such members are called *simplex elements*. If so the matrices **A**, **B** and **S** are *independent* of member cross section. For simplex elements the derivation of the element stiffness equations reduces to a straightforward sequence of matrix multiplications.

Under the constancy-along- \bar{x} assumption, elimination of the interior quantities **p** and **v** from (5.1) through (5.3) yields the element stiffness relation

$$\bar{\mathbf{f}} = \mathbf{A}^T \mathbf{S} \mathbf{B} \bar{\mathbf{u}} = \bar{\mathbf{K}} \bar{\mathbf{u}}, \quad (5.4)$$

whence the element stiffness matrix is

$$\bar{\mathbf{K}} = \mathbf{A}^T \mathbf{S} \mathbf{B}. \quad (5.5)$$

If both pairs: $\{\mathbf{p}, \mathbf{v}\}$ and $\{\bar{\mathbf{f}}, \bar{\mathbf{u}}\}$, are conjugate in the sense of the Principle of Virtual Work, it can be shown that **A** = **B** and that **S** is symmetric. In that case

$$\bar{\mathbf{K}} = \mathbf{B}^T \mathbf{S} \mathbf{B}. \quad (5.6)$$

is a symmetric matrix. Symmetry is computationally desirable for reasons outlined in Part III.

⁴ The name *rigidity matrix* for **S** is preferable. It is a member integrated version of the cross section constitutive equations. The latter are usually denoted by symbol **R**, as in §5.4.

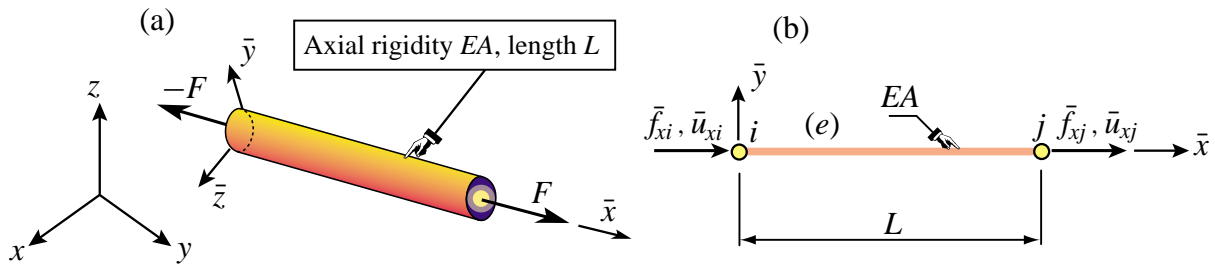


FIGURE 5.4. The prismatic bar (also called truss) member: (a) individual member shown in 3D space, (b) idealization as generic member.

Remark 5.1. If $\bar{\mathbf{f}}$ and $\bar{\mathbf{u}}$ are conjugate (as required in §5.2.2) but \mathbf{p} and \mathbf{v} are not, $\bar{\mathbf{K}}$ must come out to be symmetric even if \mathbf{S} is unsymmetric and $\mathbf{A} \neq \mathbf{B}$. However there are more opportunities to go wrong.

§5.3.1. The Bar Element Revisited

The simplest MoM element is the prismatic bar or truss member already derived in Chapter 2. See Figure 5.4. This qualifies as simplex because all internal quantities are constant. One minor difference in the derivation below is that the joint displacements and forces in the \bar{y} direction are omitted in the generic element because they contribute nothing to the stiffness equations. In the FEM terminology, freedoms associated with zero stiffness are called *inactive*. Three choices for internal deformation and force variables are considered next. The results confirm that the element stiffness equations coalesce, as expected, since the end quantities $\bar{\mathbf{f}}$ and $\bar{\mathbf{u}}$ stay the same.

Derivation Using Axial Elongation and Axial Force. The member axial elongation d is taken as deformation measure, and the member axial force F as internal force measure. Hence \mathbf{v} and \mathbf{p} reduce to the scalars $\equiv d$ and $p \equiv F$, respectively. The chain of discrete field equations is easily constructed:

$$d = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{xj} \end{bmatrix} = \mathbf{B}\bar{\mathbf{u}}, \quad F = \frac{EA}{L}d = Sd, \quad \bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{xj} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} F = \mathbf{A}^T F. \quad (5.7)$$

Consequently

$$\bar{\mathbf{K}} = \mathbf{A}^T \mathbf{S} \mathbf{B} = \mathbf{S} \mathbf{B}^T \mathbf{B} = S \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \quad (5.8)$$

Note that $\mathbf{A} = \mathbf{B}$ because F and d are conjugate: Fd is work. The foregoing equations can be represented graphically with the discrete Tonti diagram of Figure 5.5.

Derivation Using Mean Axial Strain and Axial Force. Instead of d we may use the mean axial strain $\bar{e} = d/L$ as deformation measure whereas F is kept as internal force measure. The only change is that \mathbf{B} becomes $\begin{bmatrix} -1 & 1 \end{bmatrix}/L$ whereas S becomes EA . Matrix \mathbf{A} does not change. The product $\mathbf{A}^T \mathbf{S} \mathbf{B}$ gives the same $\bar{\mathbf{K}}$ as in (5.8), as can be expected. Now \mathbf{A}^T is not equal to \mathbf{B} because F and \bar{e} are not conjugate, but they differ only by a factor $1/L$.

Derivation Using Mean Axial Strain and Axial Stress. We keep the mean axial strain $\bar{e} = d/L$ as deformation measure but take the mean axial stress $\bar{\sigma} = F/A$ (which is *not* conjugate to \bar{e}) as

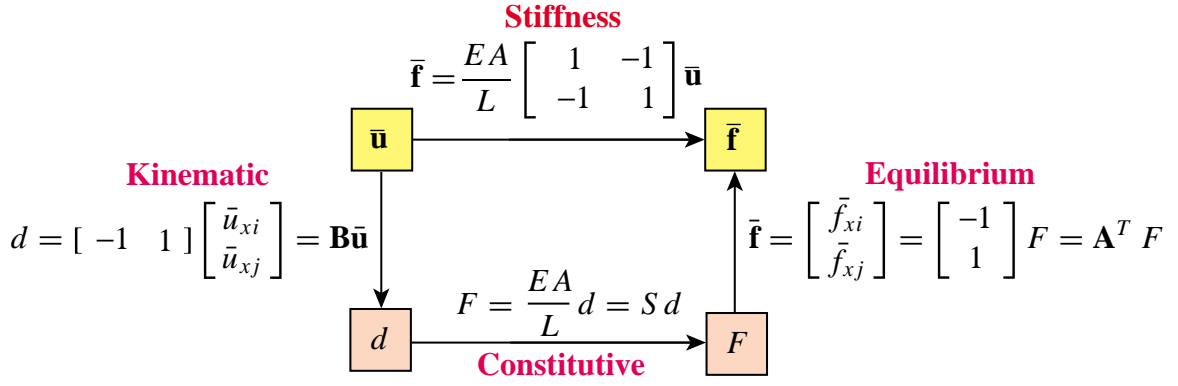


FIGURE 5.5. Tonti diagram for the bar element discrete equations (5.7)–(5.8).

internal force measure. Now $\mathbf{B} = [-1 \ 1]/L$, $\mathbf{S} = E$ and $\mathbf{A}^T = A [-1 \ 1]$. The product $\mathbf{A}^T \mathbf{S} \mathbf{B}$ gives again the same $\bar{\mathbf{K}}$, as shown in (5.8).

Transformation to Global Coordinates. Since \bar{u}_{yi} and \bar{u}_{yj} are not part of (5.7) and (5.8) the displacement transformation matrix from local to global $\{x, y\}$ coordinates is 2×4 , instead of 4×4 as in §2.8.1. On restoring the element identifier e the appropriate local-to-global transformation is

$$\bar{\mathbf{u}}^e = \begin{bmatrix} \bar{u}_{xi}^e \\ \bar{u}_{xj}^e \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{bmatrix} \begin{bmatrix} u_{xi}^e \\ u_{yi}^e \\ u_{xj}^e \\ u_{yj}^e \end{bmatrix} = \mathbf{T}^e \mathbf{u}^e, \quad (5.9)$$

in which $c = \cos \varphi^e$, $s = \sin \varphi^e$, and φ^e is the angle from x to \bar{x} , cf. Figure 2.10 of Chapter 2. The 4×4 globalized element stiffness matrix $\mathbf{K}^e = (\mathbf{T}^e)^T \bar{\mathbf{K}}^e \mathbf{T}^e$ agrees with (2.18). The extension of (5.9) to 3D is handled as an Exercise.

§5.3.2. The Spar Element

The *spar* or *shear-web* member has two joints (end nodes): i and j . This member can only resist and transmit a *constant shear force* V in the plane of the web, which is chosen to be the $\{\bar{x}, \bar{y}\}$ plane. See Figure 5.6. It is often used for modeling high-aspect aircraft wing structures, as illustrated in Figure 5.7. We consider here only *prismatic* spar members of uniform material and constant cross section, which thus qualify as simple.

The active degrees of freedom for a generic spar member of length L , as depicted in Figure 5.6(b), are \bar{u}_{yi} and \bar{u}_{yj} . Let G be the shear modulus and A_s the effective shear area. The latter is a concept developed in Mechanics of Materials; for a narrow rectangular cross section, $A_s = 5A/6$.

The shear rigidity is GA_s . As deformation measure the mean shear strain $\gamma = V/(GA_s)$ is chosen. The kinematic, constitutive, and equilibrium equations are

$$\gamma = \frac{1}{L} [-1 \ 1] \begin{bmatrix} \bar{u}_{yi} \\ \bar{u}_{yj} \end{bmatrix} = \mathbf{B} \bar{\mathbf{u}}, \quad V = GA_s \gamma = S \gamma, \quad \bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{yi} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} V = \mathbf{A}^T V, \quad (5.10)$$

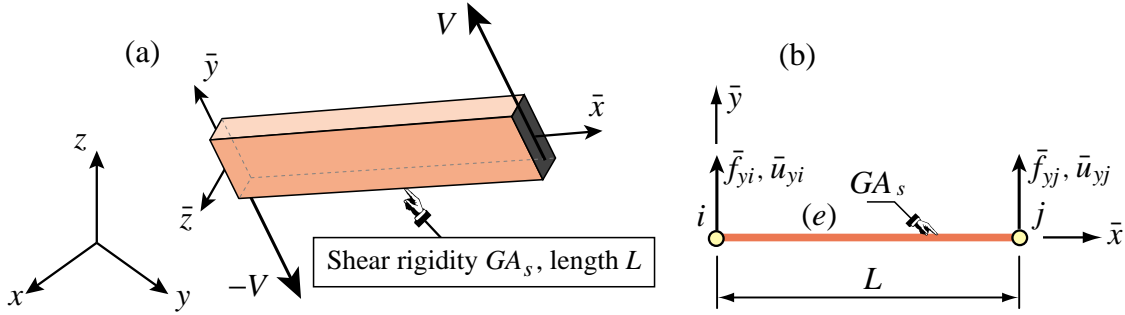


FIGURE 5.6. The prismatic spar (also called shear-web) member: (a) individual member shown in 3D space, (b) idealization as generic member in local system.

Note that $\mathbf{A} \neq \mathbf{B}$ because V and γ are not work-conjugate. (This difference is easily adjusted for, however; see Exercise 5.1.) The local stiffness equations follow as

$$\bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{yi} \\ \bar{f}_{yj} \end{bmatrix} = \mathbf{A}^T \mathbf{S} \mathbf{B} \bar{\mathbf{u}} = \frac{GA_s}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_{yi} \\ \bar{u}_{yj} \end{bmatrix} = \bar{\mathbf{K}} \bar{\mathbf{u}}. \quad (5.11)$$

If the spar member is used in a *two dimensional* context, the displacement transformation from local to global coordinates $\{x, y\}$ is

$$\bar{\mathbf{u}}^e = \begin{bmatrix} \bar{u}_{yi}^e \\ \bar{u}_{yj}^e \end{bmatrix} = \begin{bmatrix} -s & c & 0 & 0 \\ 0 & 0 & -s & c \end{bmatrix} \begin{bmatrix} u_{xi}^e \\ u_{yi}^e \\ u_{xj}^e \\ u_{yj}^e \end{bmatrix} = \mathbf{T}^e \mathbf{u}^e, \quad (5.12)$$

in which $c = \cos \varphi^e$, $s = \sin \varphi^e$, and φ^e is the angle from x to \bar{x} , cf. Figure 2.10 of Chapter 2. The 4×4 globalized spar stiffness matrix is then $\mathbf{K}^e = (\mathbf{T}^e)^T \bar{\mathbf{K}} \mathbf{T}^e$. The full expression is worked out in Exercise 5.2.

More often, however, the spar member will be a component in a *three-dimensional structural model*, e.g. the aircraft wing shown in Figure 5.7. If so the determination of the 2×6 transformation matrix is more involved, as an “orientation node” is required. This is the topic of Exercise 5.5.

§5.3.3. The Shaft Element

The *shaft*, also called *torque member*, has two joints (end nodes): i and j . A shaft can only resist and transmit a *constant torque* or *twisting moment* T along its longitudinal axis \bar{x} , as pictured in Figure 5.8(a).

We consider here only *prismatic* shaft members with uniform material and constant cross section, which thus qualify as simplex. The active degrees of freedom of a generic shaft member of length L , depicted in Figure 5.8(b), are $\bar{\theta}_{xi}$ and $\bar{\theta}_{xj}$. These are the infinitesimal end rotations about \bar{x} , positive according to the right-hand rule. The associated joint (node) forces are end moments denoted as \bar{m}_{xi} and \bar{m}_{xj} .

The only internal force is the torque T , positive if acting as pictured in Figure 5.8(a). Let G be the

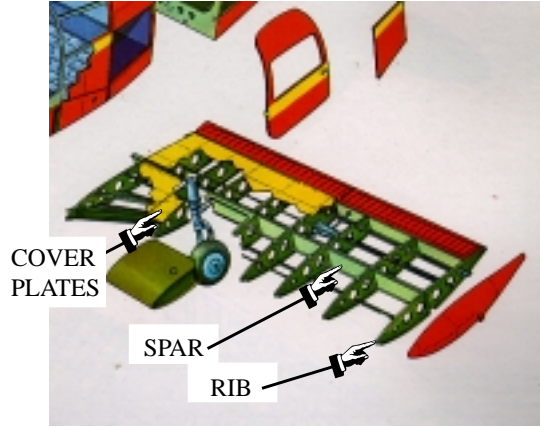


FIGURE 5.7. Spar members in aircraft wing (Piper Cherokee). For more impressive aircraft structures see CAETE slides.

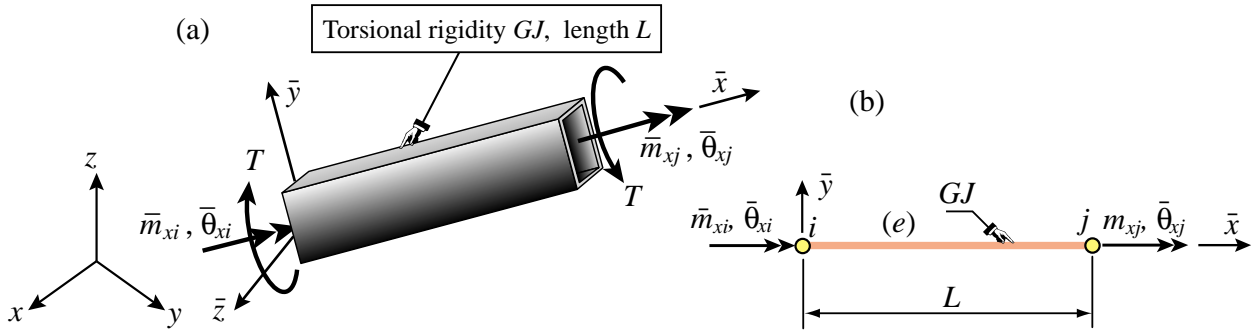


FIGURE 5.8. The prismatic shaft (also called torque member): (a) individual member shown in 3D space, (b) idealization as generic member in the local system.

shear modulus and GJ the effective torsional rigidity.⁵ As deformation measure pick the relative twist angle $\phi = \bar{\theta}_{xj} - \bar{\theta}_{xi}$. The kinematic, constitutive, and equilibrium equations provided by Mechanics of Materials are

$$\phi = \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{\theta}_{xi} \\ \bar{\theta}_{xj} \end{bmatrix} = \mathbf{B}\bar{\mathbf{u}}, \quad T = \frac{GJ}{L}\phi = S\gamma, \quad \bar{\mathbf{f}} = \begin{bmatrix} \bar{m}_{xi} \\ \bar{m}_{xj} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} T = \mathbf{B}^T T. \quad (5.13)$$

From these the local stiffness equations follow as

$$\bar{\mathbf{f}} = \begin{bmatrix} \bar{m}_{xi} \\ \bar{m}_{xj} \end{bmatrix} = \mathbf{B}^T S \mathbf{B}\bar{\mathbf{u}} = \frac{GJ}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{\theta}_{xi} \\ \bar{\theta}_{xj} \end{bmatrix} = \bar{\mathbf{K}}\bar{\mathbf{u}}. \quad (5.14)$$

If the shaft is used in a two-dimensional context, the displacement transformation to global coordi-

⁵ J has dimension of (length)⁴. For a circular or annular cross section it reduces to the polar moment of inertia about \bar{x} . The determination of J for noncircular cross sections is covered in Mechanics of Materials textbooks.

nates $\{x, y\}$ within the framework of infinitesimal rotations, is

$$\bar{\mathbf{u}}^e = \begin{bmatrix} \bar{\theta}_{xi}^e \\ \bar{\theta}_{xj}^e \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{bmatrix} \begin{bmatrix} \theta_{xi}^e \\ \theta_{yi}^e \\ \theta_{xj}^e \\ \theta_{yj}^e \end{bmatrix} = \mathbf{T}^e \boldsymbol{\theta}^e, \quad (5.15)$$

where as usual $c = \cos \varphi^e$, $s = \sin \varphi^e$, and φ^e is the angle from x to \bar{x} . Note that $\boldsymbol{\theta}^e$ collects only global node rotations components. This operation is elaborated further in Exercise 5.3.

§5.4. *Non-Simplex MoM Members

The straightforward formulation of simplex MoM elements does not immediately carry over to the case in which internal quantities \mathbf{p} and \mathbf{v} vary over the member; that is, depend on \bar{x} . The dependence may be due to element type, varying cross section, or both. As a result, one or more of the matrices \mathbf{A} , \mathbf{B} and \mathbf{S} depend on \bar{x} . Such members are called *non-simplex*.

The matrix multiplication rule (5.4) cannot be used to construct the element stiffness matrix $\bar{\mathbf{K}}^e$ of non-simplex members. This can be grasped by observing that $\mathbf{A}(\bar{x})^T \mathbf{S}(\bar{x}) \mathbf{B}(\bar{x})$ would depend on \bar{x} . On the other hand, $\bar{\mathbf{K}}^e$ must be independent of \bar{x} because it relates the end quantities $\bar{\mathbf{u}}$ and $\bar{\mathbf{f}}$.

§5.4.1. *Formulation Rules

The derivation of non-simplex MoM elements requires use of the work principles of mechanics, for example the Principle of Virtual Work or PVW. Thus, more care must be exercised in the choice of conjugate internal quantities. The following rules can be justified through the arguments presented in Part II. They are stated here as recipe, and apply only to displacement-assumed elements.

Rule 1. Select internal deformations $\mathbf{v}(\bar{x})$ and internal forces $\mathbf{p}(\bar{x})$ that are conjugate in the PVW sense. Link deformations to node displacements by $\mathbf{v}(\bar{x}) = \mathbf{B}(\bar{x})\mathbf{u}$.

Rule 2. From the PVW it may be shown (see **Remark 5.2** below) that the force equilibrium equation exists only in a differential sense:

$$\mathbf{B}^T d\mathbf{p} = d\bar{\mathbf{f}}. \quad (5.16)$$

Here d in $d\mathbf{p}$ denotes differentiation with respect to \bar{x} . The meaning of $d\mathbf{p}$ is simply $\mathbf{p}(\bar{x}) d\bar{x}$. That is, the differential of internal forces as one passes from cross-section \bar{x} to a neighboring one $\bar{x} + d\bar{x}$. The interpretation of $d\bar{\mathbf{f}}$ is less immediate because $\bar{\mathbf{f}}$ is not a function of \bar{x} . It actually means the contribution of that member slice to the building of the node force vector $\bar{\mathbf{f}}$. See (5.18) and (5.19) below.

Rule 3. The constitutive relation is

$$\mathbf{p} = \mathbf{R}\mathbf{v}, \quad (5.17)$$

in which \mathbf{R} , which may depend on \bar{x} , must be symmetric. Note that symbol \mathbf{R} in (5.17) replaces the \mathbf{S} of (5.2). Matrix \mathbf{R} pertains to a specific cross section whereas \mathbf{S} applies to the entire member. This distinction is further elaborated in Exercise 5.9.

The discrete relations supplied by the foregoing rules are displayed in the discrete Tonti diagram of Figure 5.9. Internal quantities are now eliminated starting from the differential equilibrium relation (5.16):

$$d\bar{\mathbf{f}} = \mathbf{B}^T d\mathbf{p} = \mathbf{B}^T \mathbf{p} d\bar{x} = \mathbf{B}^T \mathbf{R} \mathbf{v} d\bar{x} = \mathbf{B}^T \mathbf{R} \mathbf{B} \bar{\mathbf{u}} d\bar{x} = \mathbf{B}^T \mathbf{R} \mathbf{B} d\bar{x} \bar{\mathbf{u}}. \quad (5.18)$$

Integrating both sides over the member length L yields

$$\bar{\mathbf{f}} = \int_0^L d\bar{\mathbf{f}} = \int_0^L \mathbf{B}^T \mathbf{R} \mathbf{B} d\bar{x} \bar{\mathbf{u}} = \bar{\mathbf{K}} \bar{\mathbf{u}}, \quad (5.19)$$

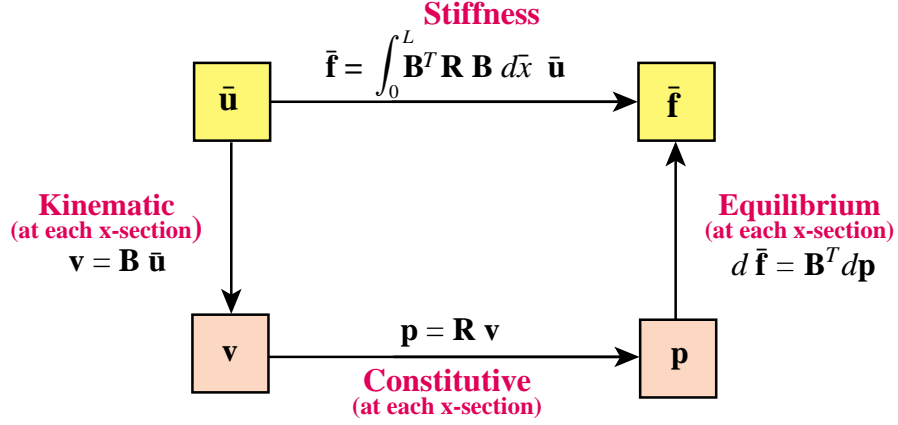


FIGURE 5.9. Discrete Tonti diagram of the equations for a non-simplex MoM member.

because $\bar{\mathbf{u}}$ does not depend on \bar{x} . Consequently the local element stiffness matrix is

$$\bar{\mathbf{K}} = \int_0^L \mathbf{B}^T \mathbf{R} \mathbf{B} d\bar{x} \quad (5.20)$$

The recipe (5.20) will be justified in Part II through energy methods. It will be seen that it generalizes to arbitrary displacement-assumed finite elements in any number of space dimensions. It is used in the derivation of the stiffness equations of the plane beam element in Chapter 12. The reduction of (5.20) to (5.6) when the dependence on \bar{x} disappears is the subject of Exercise 5.8.

Remark 5.2. The proof of (5.16) follows by equating expressions of the virtual work of a slice of length $d\bar{x}$ undergoing virtual node displacements $\delta\bar{\mathbf{u}}$ and associated deformations $\delta\mathbf{v}$: $d\bar{\mathbf{f}}^T \delta\bar{\mathbf{u}} = d\mathbf{p}^T \delta\mathbf{v} = d\mathbf{p}^T (\mathbf{B}\bar{\mathbf{u}}) = (\mathbf{B}^T d\mathbf{p})^T \delta\bar{\mathbf{u}}$. Since $\delta\bar{\mathbf{u}}$ is arbitrary, $\mathbf{B}^T d\mathbf{p} = d\bar{\mathbf{f}}$.

§5.4.2. *Examples

Example 5.1. A two-node bar element has constant elastic modulus E but a continuously varying area: A_i , A_j and A_m at i , j and m , respectively, where m is the midpoint between end joints i and j . This variation can be fitted by

$$A(\bar{x}) = A_i N_i(\bar{x}) + A_j N_j(\bar{x}) + A_m N_m(\bar{x}). \quad (5.21)$$

Here $N_i(\bar{x}) = -\frac{1}{2}\xi(1 - \xi)$, $N_j(\bar{x}) = \frac{1}{2}\xi(1 + \xi)$ and $N_m(\bar{x}) = 1 - \xi^2$, with $\xi = 2x/L - 1$, are interpolating polynomials further studied in Part II as “element shape functions.”

As internal quantities take the strain e and the axial force $p = EAe$, which are conjugate quantities. Assuming the strain e to be uniform over the element (this is characteristic of a displacement assumed element and is justified through the method of shape functions explained in Part II.) the MoM equations are

$$e = \mathbf{B}\bar{\mathbf{u}}, \quad p = EA(\bar{x})e = R(\bar{x})e, \quad d\bar{\mathbf{f}} = \mathbf{B}^T d\mathbf{p}, \quad \mathbf{B} = \frac{1}{L} \begin{bmatrix} -1 & 1 \end{bmatrix}. \quad (5.22)$$

Inserting into (5.20) and carrying out the integration yields

$$\bar{\mathbf{K}} = \frac{E\bar{A}}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad \text{with} \quad \bar{A} = \frac{1}{6}(A_i + A_j) + \frac{2}{3}A_m. \quad (5.23)$$

Example 5.2. Same as in the previous case but now the strain e is taken to be $e = p/(EA)$, whereas the axial force p is constant and defined by $p = -\bar{f}_{xi} = \bar{f}_{xj}$. The integrals become rational functions of \bar{x} and are best evaluated through *Mathematica*. The completion of this Example is the matter of an Exercise.

Notes and Bibliography

The derivation of MoM elements using straightforward matrix algebra is typical of pre-1962 Matrix Structural Analysis (MSA). The excellent book of Pestel and Leckie [519], unfortunately out of print, epitomizes that approach. Historically this idea interweaved with Generation 1 of FEM, as outlined in §1.8. By 1970 simplified derivations had fallen out of favor as yokelish. But these elements do not need improvement. They still work fine: a bar or beam stiffness today is the same as 40 years ago.⁶

The Mechanics of Materials books by Beer-Johnston [62] and Popov [533] may be cited as being widely used in US undergraduate courses. But they are not the only ones. A September 2003 in-print book search through www3.addall.com on “Mechanics of Materials” returns 99 hits whereas one on “Strength of Materials” (the older name) compiles 112. Folding multiple editions and hardback/paperback variants one gets about 60 books; by all accounts an impressive number.

Spar members are discussed only in MoM books that focus on aircraft structures, since they are primarily used in modeling shear web action. On the other hand, bars, shafts and beams are standard fare.

The framework presented here is a tiny part of MSA. A panoramic view, including linkage to continuum formulations from the MSA viewpoint, is presented in [209].

The source of Tonti diagrams is discussed in Chapter 11.

References

Referenced items have been moved to Appendix R.

⁶ The chief technical difference is the heavier use of differential equations prior to 1962, as opposed to the energy methods in vogue today. The end result for simple one-dimensional models is the same.

Homework Exercises for Chapter 5

Constructing MoM Members

EXERCISE 5.1 [A:10] Explain how to select the deformation variable v (paired to V) of the spar member formulated in §5.3.2, so that $\mathbf{A} = \mathbf{B}$. Draw the Tonti diagram with the discrete equations for that choice of v and p , using Figure 5.5 as guide (that is, with the actual matrix equations along the arrows).

EXERCISE 5.2 [A:15] Obtain the 4×4 global element stiffness matrix of a prismatic spar member in a two dimensional Cartesian system $\{x, y\}$. Start from (5.11). Indicate where the transformation (5.12) comes from (Hint: read §2.8). Evaluate $\mathbf{K}^e = (\mathbf{T}^e)^T \tilde{\mathbf{K}}^e \mathbf{T}^e$ in closed form.

EXERCISE 5.3 [A:15] Obtain the 4×4 global element stiffness matrix of a prismatic shaft element in a two dimensional Cartesian system $\{x, y\}$. Include only node rotation freedoms in the global displacement vector. Start from (?). Justify the transformation (5.15) (Hint: infinitesimal rotations transform as vectors). Evaluate $\mathbf{K}^e = (\mathbf{T}^e)^T \tilde{\mathbf{K}}^e \mathbf{T}^e$ in closed form.

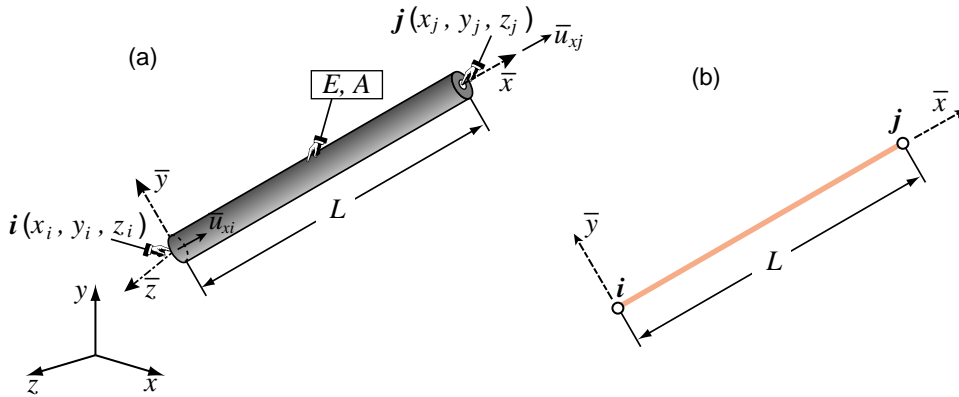


FIGURE E5.1. Bar element in 3D for Exercise 5.4.

EXERCISE 5.4 [A+N:15(10+5)] A bar element moving in three dimensional space is completely defined by the global coordinates $\{x_i, y_i, z_i\}$, $\{x_j, y_j, z_j\}$ of its end nodes i and j , as illustrated in Figure E5.1. The 2×6 displacement transformation matrix \mathbf{T} , with superscript e dropped for brevity, links $\bar{\mathbf{u}}^e = \mathbf{T}\mathbf{u}^e$. Here $\bar{\mathbf{u}}^e$ contains the two local displacements \bar{u}_{xi} and \bar{u}_{xj} whereas \mathbf{u}^e contains the six global displacements $u_{xi}, u_{yi}, u_{zi}, u_{xj}, u_{yj}, u_{zj}$.

(a) From vector mechanics show that

$$\mathbf{T} = \frac{1}{L} \begin{bmatrix} x_{ji} & y_{ji} & z_{ji} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{ji} & y_{ji} & z_{ji} \end{bmatrix} = \begin{bmatrix} c_{xji} & c_{yji} & c_{zji} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{xji} & c_{yji} & c_{zji} \end{bmatrix} \quad (\text{E5.1})$$

in which L is the element length, $x_{ji} = x_j - x_i$, etc., and $c_{xji} = x_{ji}/L$, etc., are the direction cosines of the vector going from i to j .

(b) Evaluate \mathbf{T} for a bar going from node i at $\{1, 2, 3\}$ to node j at $\{3, 8, 6\}$.

EXERCISE 5.5 [A+N:30(10+15+5)] A spar element in three dimensional space is only partially defined by the global coordinates $\{x_i, y_i, z_i\}$, $\{x_j, y_j, z_j\}$ of its end nodes i and j , as illustrated in Figure E5.2. The problem is that axis \bar{y} , which defines the direction of shear force transmission, is not uniquely defined by i and j .⁷ Most FEM programs use the *orientation node* method to complete the definition. A third node k , not

⁷ The same ambiguity arises in beam elements in 3D space. These elements are covered in Part III.

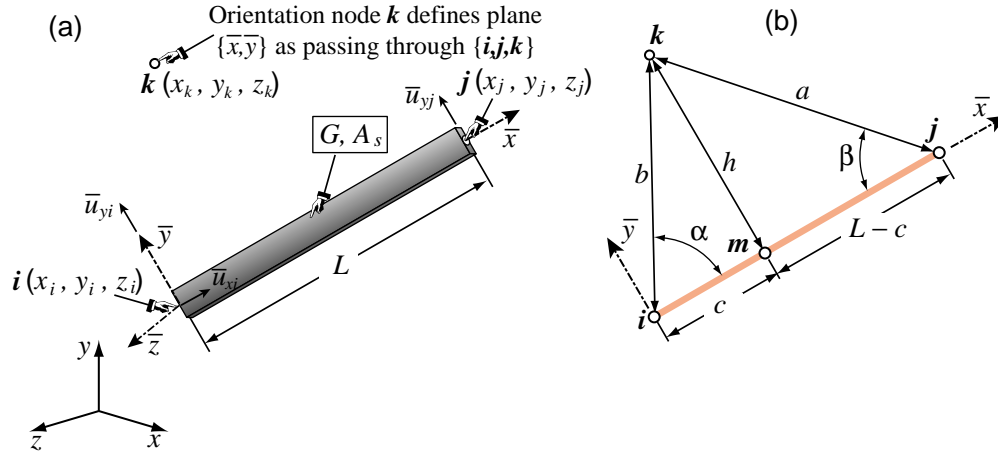


FIGURE E5.2. Spar element in 3D for Exercise 5.5.

colinear with i and j , is provided by the user. Nodes $\{i, j, k\}$ define the $\{\bar{x}, \bar{y}\}$ plane and thus \bar{z} . The projection of k on line ij is point m . The distance $h > 0$ from m to k is called h as shown in Figure E5.2(b). The 2×6 displacement transformation matrix \mathbf{T} , with superscript e omitted to reduce clutter, relates $\bar{\mathbf{u}}^e = \mathbf{T}\mathbf{u}^e$. Here $\bar{\mathbf{u}}^e$ contains the local transverse displacements \bar{u}_{yi} and \bar{u}_{yj} whereas \mathbf{u}^e contains the global displacements $u_{xi}, u_{yi}, u_{zi}, u_{xj}, u_{yj}, u_{zj}$.

(a) Show that

$$\mathbf{T} = \frac{1}{h} \begin{bmatrix} x_{km} & y_{km} & z_{km} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{km} & y_{km} & z_{km} \end{bmatrix} = \begin{bmatrix} c_{xkm} & c_{ykm} & c_{zkm} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{xkm} & c_{ykm} & c_{zkm} \end{bmatrix} \quad (\text{E5.2})$$

in which $x_{km} = x_k - x_m$, etc., and $c_{xkm} = x_{km}/h$, etc., are the direction cosines of the vector going from m to k . (Assume that the position of m is known. That computation is carried out in the next item.)

- (b) Work out the formulas to compute the coordinates of point m in terms of the coordinates of $\{i, j, k\}$. Assume a, b and L are computed immediately from the input data. Using the notation of Figure E5.2(b) and elementary trigonometry, show that $h = 2A/L$, where $A = \sqrt{p(p-a)(p-b)(p-L)}$ with $p = \frac{1}{2}(L+a+b)$ (Heron's formula), $\cos \alpha = (L^2 + b^2 - a^2)/(2bL)$, $\cos \beta = (L^2 + a^2 - b^2)/(2aL)$, $c = b \cos \alpha$, $L - c = a \cos \beta$, $x_m = x_i(L-c)/L + x_j c/L$, etc.⁸
- (c) Evaluate \mathbf{T} for a spar member going from node i at $\{1, 2, 3\}$ to node j at $\{3, 8, 6\}$. with k at $\{4, 5, 6\}$.

EXERCISE 5.6 [A:20] Explain how thermal effects can be generally incorporated in the constitutive equation (5.2) to produce an initial force vector for a simplex element.

EXERCISE 5.7 [A:15] Draw the discrete Tonti diagram for the prismatic shaft element. Use 5.5 as a guide (that is, with the actual matrix equations along the arrows).

EXERCISE 5.8 [A:15] If the matrices \mathbf{B} and \mathbf{R} are constant over the element length L , show that expression (5.20) of the element stiffness matrix for a non-simplex member reduces to (5.6), in which $\mathbf{S} = \mathbf{L}\mathbf{R}$.

EXERCISE 5.9 [A:20] Explain in detail the quickie derivation of footnote 6. (Knowledge of the Principle of Virtual Work is required to do this exercise.)

⁸ An alternative and more elegant procedure, found by a student in 1999, can be sketched as follows. From Figure E5.2(b) obviously the two subtriangles imk and jkm are right-angled at m and share side km of length h . Apply Pythagoras' theorem twice, and subtract so as to cancel out h^2 and c^2 , getting a linear equation for c that can be solved directly.

EXERCISE 5.10 [A:25(10+5+10)] Consider a non-simplex element in which \mathbf{R} varies with \bar{x} but $\mathbf{B} = \mathbf{A}$ is constant.

- (a) From (5.20) prove that

$$\bar{\mathbf{K}} = L \mathbf{B}^T \bar{\mathbf{R}} \mathbf{B}, \quad \text{with} \quad \bar{\mathbf{R}} = \frac{1}{L} \int_0^L \mathbf{R}(\bar{x}) d\bar{x} \quad (\text{E5.3})$$

- (b) Apply (E5.3) to obtain $\bar{\mathbf{K}}$ for a tapered bar with area defined by the linear law $A = A_i(1 - \bar{x}/L) + A_j\bar{x}/L$, where A_i and A_j are the end areas at i and j , respectively. Take $\mathbf{B} = [-1 \quad 1]/L$.
- (c) Apply (E5.3) to verify the result (5.23) for a bar with parabolically varying cross section.

EXERCISE 5.11 [A/C+N:30(25+5)] A prismatic bar element in 3D space is referred to a global coordinate system $\{x, y, z\}$, as in Figure E5.1. The end nodes are located at $\{x_1, y_1, z_1\}$ and $\{x_2, y_2, z_2\}$.⁹ The elastic modulus E and the cross section area A are constant along the length. Denote $x_{21} = x_2 - x_1$, $y_{21} = y_2 - y_1$, $z_{21} = z_2 - z_1$ and $L = \sqrt{x_{21}^2 + y_{21}^2 + z_{21}^2}$.

- (a) Show that the element stiffness matrix in *global* coordinates can be compactly written¹⁰ A plodding way is to start from the local stiffness (5.8) and transform to global using (?)

$$\mathbf{K}^e = \frac{EA}{L^3} \mathbf{B}^T \mathbf{B}, \quad \text{in which} \quad \mathbf{B} = [-x_{21} \quad -y_{21} \quad -z_{21} \quad x_{21} \quad y_{21} \quad z_{21}]. \quad (\text{E5.4})$$

- (b) Compute \mathbf{K}^e if the nodes are at $\{1, 2, 3\}$ and $\{3, 8, 6\}$, with elastic modulus $E = 343$ and cross section area $A = 1$. Note: the computation can be either done by hand or with the help of a program such as the following *Mathematica* module, which is used in Part III:

```
Stiffness3DBar[ncoor_,mprop_,fprop_,opt_] := Module[
  {x1,x2,y1,y2,z1,z2,x21,y21,z21,Em,Gm,rho,alpha,A,
   num,L,LL,LLL,B,Ke}, { {x1,y1,z1},{x2,y2,z2}}=ncoor;
  {x21,y21,z21}={x2-x1,y2-y1,z2-z1};
  {Em,Gm,rho,alpha}=mprop; {A}=fprop; {num}=opt;
  If [num,{x21,y21,z21,Em,A}=N[{x21,y21,z21,Em,A}]];
  LL=x21^2+y21^2+z21^2; L=PowerExpand[Sqrt[LL]];
  LLL=Simplify[LL*L]; B={{-x21,-y21,-z21,x21,y21,z21}};
  Ke=(Em*A/LLL)*Transpose[B].B;
  Return[Ke]];
```

```
ClearAll[Em,A]; Em=343; A=1;
```

⁹ End nodes are labeled 1 and 2 instead of i and j to agree with the code listed below.

¹⁰ There are several ways of arriving at this result. Some are faster and more elegant than others. Here is a sketch of one of the ways. Denote by L_0 and L the lengths of the bar in the undeformed and deformed configurations, respectively. Then

$$\frac{1}{2}(L^2 - L_0^2) = x_{21}(u_{x2} - u_{x1}) + y_{21}(u_{y2} - u_{y1}) + z_{21}(u_{z2} - u_{z1}) + Q \approx \mathbf{B}\mathbf{u},$$

in which Q is a quadratic function of node displacements which is therefore dropped in the small-displacement linear theory. Also on account of small displacements

$$\frac{1}{2}(L^2 - L_0^2) = \frac{1}{2}(L + L_0)(L - L_0) \approx L \Delta L.$$

Hence the small axial strain is $e = \Delta L/L = (1/L^2)\mathbf{B}\mathbf{u}^e$, which begins the Tonti diagram. Next is $F = EA e$. Finally you must show that force equilibrium at nodes requires $\mathbf{f}^e = (1/L)\mathbf{B}^T F$. Multiplying through gives (E5.4).

```

ncoor={{0,0,0},{2,6,3}}; mprop={Em,0,0,0}; fprop={A}; opt={False};
Ke=Stiffness3DBar[ncoor,mprop,fprop,opt];
Print["Stiffness of 3D Bar Element:"];
Print[Ke//MatrixForm];
Print["eigs of Ke: ",Eigenvalues[Ke]];

```

As a check, the six eigenvalues of this particular \mathbf{K}^e should be 98 and five zeros.

EXERCISE 5.12 [A/C:25] Complete Example 5.2.

6

FEM Modeling: Introduction

TABLE OF CONTENTS

	Page
§6.1. Introduction	6–3
§6.2. FEM Terminology	6–3
§6.3. Idealization	6–4
§6.3.1. Models	6–4
§6.3.2. Mathematical Models	6–5
§6.3.3. Implicit vs. Explicit Modeling	6–6
§6.4. Discretization	6–7
§6.4.1. Analytical or Numerical?	6–7
§6.4.2. Error Sources and Approximation	6–7
§6.4.3. Other Discretization Methods	6–8
§6.5. The Finite Element Method	6–8
§6.6. Element Attributes	6–9
§6.6.1. Element Dimensionality	6–9
§6.6.2. Element Nodes	6–10
§6.6.3. Element Geometry	6–10
§6.6.4. Element Degrees of Freedom	6–10
§6.6.5. Nodal Forces	6–10
§6.6.6. Element Constitutive Properties	6–10
§6.6.7. Element Fabrication Properties	6–10
§6.7. Classification of Mechanical Elements	6–11
§6.7.1. Primitive Structural Elements	6–11
§6.7.2. Continuum Elements	6–11
§6.7.3. Special Elements	6–12
§6.7.4. Macroelements	6–12
§6.7.5. Substructures	6–13
§6.8. Assembly	6–13
§6.9. Boundary Conditions	6–13
§6.9.1. Essential and Natural B.C.	6–13
§6.9.2. B.C. in Structural Problems	6–14
§6. Notes and Bibliography	6–14
§6. References	6–15

§6.1. Introduction

Chapters 2 through 5 cover material technically known as Matrix Structural Analysis or MSA. As chronicled in Appendix H, this is a subject that historically preceded the Finite Element Method (FEM), although it contributed substantially to it.

This Chapter begins covering the FEM proper. This is distinguished from MSA by three traits:

- The ability to go beyond structures.
- The increasing importance of continuum models in two and three space dimensions.
- The key role of variational mathematics.

This Chapter introduces terminology used in FEM modeling, and surveys attributes and types of finite elements used in structural mechanics. The next Chapter goes over more specific rules for defining meshes and boundary conditions.

§6.2. FEM Terminology

The ubiquitous term “degrees of freedom,” often abbreviated to either freedom or DOF, has figured prominently in previous Chapters. This term, as well as “stiffness matrix” and “force vector,” originated in structural mechanics, the application for which FEM was invented. These names have carried over to non-structural applications. This “terminology overspill” is discussed next.

Classical analytical mechanics is that invented by Euler and Lagrange in the XVIII century and further developed by Hamilton, Jacobi and Poincaré as a systematic formulation of Newtonian mechanics. Its objects of attention are models of mechanical systems ranging from material particles composed of sufficiently large number of molecules, through airplanes, to the Solar System.¹ The spatial configuration of any such system is described by its *degrees of freedom* or DOF. These are also called *generalized coordinates*. The terms *state variables* and *primary variables* are also used, particularly in mathematically oriented treatments.

If the number of degrees of freedom is finite, the model is called *discrete*, and *continuous* otherwise. Because FEM is a discretization method, the number of DOF of a FEM model is necessarily finite. They are collected in a column vector called \mathbf{u} . This vector is called the *DOF vector* or *state vector*. The term *nodal displacement vector* for \mathbf{u} is reserved to mechanical applications.

In analytical mechanics, each degree of freedom has a corresponding “conjugate” or “dual” term, which represents a generalized force.² In non-mechanical applications, there is a similar set of conjugate quantities, which for want of a better term are also called *forces* or *forcing terms*. They are the agents of change. These forces are collected in a column vector called \mathbf{f} . The inner product $\mathbf{f}^T \mathbf{u}$ has the meaning of external energy or work.³

Just as in the truss problem, the relation between \mathbf{u} and \mathbf{f} is assumed to be of linear and homogeneous. The last assumption means that if \mathbf{u} vanishes so does \mathbf{f} . The relation is then expressed by the master

¹ For cosmological scales, such as galaxy clusters or black holes, the general theory of relativity is necessary. For the atomic and sub-atomic world, quantum mechanics is appropriate.

² In variational mathematics this is called a duality pairing.

³ Energy is the capacity to do work. Thus energy and work potentials are the same function (or functional), but with signs reversed.

Table 6.1. Significance of \mathbf{u} and \mathbf{f} in Miscellaneous FEM Applications

<i>Application Problem</i>	<i>State (DOF) vector \mathbf{u} represents</i>	<i>Conjugate vector \mathbf{f} represents</i>
Structures and solid mechanics	Displacement	Mechanical force
Heat conduction	Temperature	Heat flux
Acoustic fluid	Displacement potential	Particle velocity
Potential flows	Pressure	Particle velocity
General flows	Velocity	Fluxes
Electrostatics	Electric potential	Charge density
Magnetostatics	Magnetic potential	Magnetic intensity

stiffness equations:

$$\mathbf{K}\mathbf{u} = \mathbf{f}. \quad (6.1)$$

\mathbf{K} is universally called the *stiffness matrix* even in non-structural applications because no consensus has emerged on different names.

The physical significance of the vectors \mathbf{u} and \mathbf{f} varies according to the application being modeled, as illustrated in Table 6.1.

If the relation between forces and displacements is linear but not homogeneous, equation (6.1) generalizes to

$$\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I. \quad (6.2)$$

Here \mathbf{f}_I is the initial node force vector introduced in Chapter 29 for effects such as temperature changes, and \mathbf{f}_M is the vector of mechanical forces.

The basic steps of FEM are discussed below in more generality. Although attention is focused on structural problems, most of the steps translate to other applications problems as noted above. The role of FEM in numerical simulation is schematized in Figure 6.1, which is a merged simplification of Figures 1.2 and 1.3. Although this diagram oversimplifies the way FEM is actually used, it serves to illustrate terminology. The three key simulation steps shown are: *idealization*, *discretization* and *solution*. Each step is a source of errors. For example, the discretization error is the discrepancy that appears when the discrete solution is substituted in the mathematical model. The reverse steps: continuification and realization, are far more difficult and (generally) ill-posed problems.

The idealization and discretization steps, briefly mentioned in Chapter 1, deserve further discussion. The solution step is dealt with in more detail in Part III of this book.

§6.3. Idealization

Idealization passes from the physical system to a mathematical model. This is the most important step in engineering practice, because it cannot be “canned.” It must be done by a human.

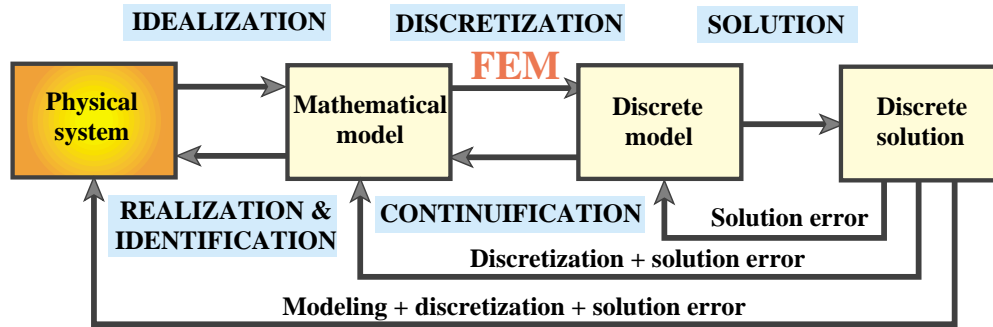


FIGURE 6.1. A simplified view of the physical simulation process, reproduced from Chapter 2 to illustrate modeling terminology.

§6.3.1. Models

The word “model” has the traditional meaning of a scaled copy or representation of an object. And that is precisely how most dictionaries define it. We use here the term in a more modern sense, which has become increasingly common since the advent of computers:

A model is a symbolic device built to simulate and predict aspects of behavior of a system.

(6.3)

Note the distinction made between *behavior* and *aspects of behavior*. To predict everything, in all physical scales, you must deal with the actual system. A model *abstracts* aspects of interest to the modeler.⁴ The qualifier *symbolic* means that a model represents a system in terms of the symbols and language of another discipline. For example, engineering systems may be (and are) modeled with the symbols of mathematics and/or computer sciences.⁵

§6.3.2. Mathematical Models

Mathematical modeling, or *idealization*, is a process by which an engineer or scientist passes from the actual physical system under study, to a *mathematical model* of the system, where the term *model* is understood in the sense of (6.3).

The process is called *idealization* because the mathematical model is necessarily an abstraction of the physical reality — note the phrase *aspects of behavior* in (6.3). The analytical or numerical results produced by the mathematical model are physically re-interpreted only for those aspects.⁶

To give an example of the choices that an engineer may face, suppose that the structure is a flat plate structure subjected to transverse loading. Here is a non-exhaustive list of four possible mathematical models:

1. A *very thin* plate model based on Von Karman’s coupled membrane-bending theory.
2. A *thin* plate model, such as the classical Kirchhoff’s plate theory.

⁴ “All models are wrong, some are useful.” (George Box)

⁵ A problem-definition input file, a digitized earthquake record, or a stress plot are examples of the latter.

⁶ Whereas idealization can be reasonably taught in advanced design courses, the converse process of “realization” or “identification” — see Figure 6.1 — generally requires considerable physical understanding and maturity that can only be gained through professional experience.

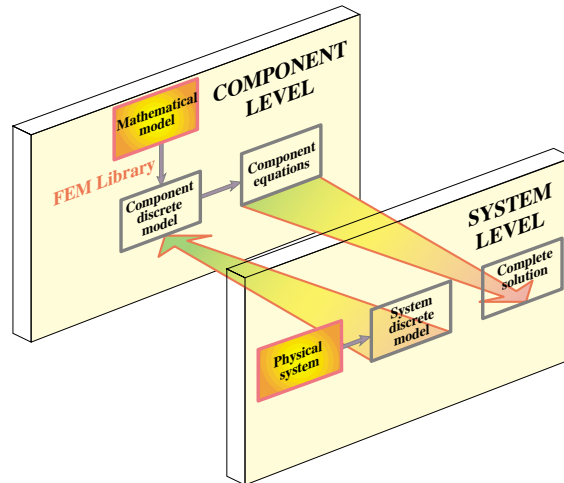


FIGURE 6.2. A reproduction of Figure 1.5 with some relabeling. Illustrates implicit modeling: picking elements from an existing FEM code consents to an idealization. This has professional as well as legal implications.

3. A *moderately thick* plate model, for example that of Mindlin-Reissner plate theory.
4. A *very thick* plate model based on three-dimensional elasticity.

The person responsible for this kind of decision is supposed to be familiar with the advantages, disadvantages, and range of applicability of each model. Furthermore the decision may be different in static analysis than in dynamics.

Why is the mathematical model an abstraction of reality? Engineering systems, particularly in Aerospace and Mechanical, tend to be highly complex. For simulation it is necessary to reduce that complexity to manageable proportions. Mathematical modeling is an abstraction tool by which complexity can be tamed.

Complexity control is achieved by “filtering out” physical details that are not relevant to the design and analysis process. For example, a continuum material model filters out the aggregate, crystal, molecular and atomic levels of matter. Engineers are typically interested in a few integrated quantities, such as the maximum deflection of a bridge or the fundamental periods of an airplane. Although to a physicist this is the result of the interaction of billions and billions of molecules, such details are weeded out by the modeling process. Consequently, picking a mathematical model is equivalent to choosing an information filter.

§6.3.3. Implicit vs. Explicit Modeling

As noted the diagram of Figure 6.1 is an oversimplification of engineering practice. The more common scenario is that pictured in Figures 1.2, 1.4 and 1.5 of Chapter 1. The latter is reproduced in Figure 6.2 for convenience.

A common scenario in industry is: you have to analyze a structure or portion(s) of one, and at your disposal is a “black box” general-purpose finite element program. Those programs offer a *catalog* of element types; for example, bars, beams, plates, shells, axisymmetric solids, general 3D solids, and so on. The moment you choose specific elements from the catalog you automatically accept the mathematical models on which the elements are based. This is *implicit modeling*. Ideally you

should be fully aware of the implications of your choice. Providing such “finite element literacy” is one of the objective of this book. Unfortunately many users of commercial programs are unaware of the implied-consent aspect of implicit modeling and their legal implications.⁷

The other extreme happens when you select a mathematical model of the physical problem with your eyes wide open and *then* either shop around for a finite element program that implements that model, or write the program yourself. This is *explicit modeling*. It requires far more technical expertise, resources, experience and maturity than implicit modeling. But for problems that fall out of the ordinary it could be the right thing to do.

In practice a combination of implicit and explicit modeling is common. The physical problem to be simulated is broken down into subproblems. Those subproblems that are conventional and fit available programs may be treated with implicit modeling, whereas those that require special handling may only submit to explicit modeling.

§6.4. Discretization

Mathematical modeling is a simplifying step. But models of physical systems are not necessarily simple to solve. They often involve coupled partial differential equations in space and time subject to boundary and/or interface conditions. Such models have an *infinite* number of degrees of freedom.

§6.4.1. Analytical or Numerical?

At this point one faces the choice of going for analytical or numerical solutions. Analytical solutions, also called “closed form solutions,” are more intellectually satisfying, particularly if they apply to a wide class of problems, so that particular instances may be obtained by substituting the values of free parameters. Unfortunately they tend to be restricted to regular geometries and simple boundary conditions. Moreover some closed-form solutions, expressed for example as inverses of integral transforms, may have to be anyway numerically evaluated to be useful.

Most problems faced by the engineer either do not yield to analytical treatment or doing so would require a disproportionate amount of effort.⁸ The practical way out is numerical simulation. Here is where finite element methods enter the scene.

To make numerical simulations practical it is necessary to reduce the number of degrees of freedom to a *finite* number. The reduction is called *discretization*. The product of the discretization process is the *discrete model*. As discussed in Chapter 1, for complex engineering systems this model is the product of a multilevel decomposition.

Discretization can proceed in space dimensions as well as in the time dimension. Because the present book deals with static problems, we need not consider the time dimension and are free to focus on *spatial discretization*.

⁷ Legal problems arise when something goes wrong. Say, a structure collapses under service (non emergency) conditions, and there is loss of life. Who is to blame? It should be noted that vendors of commercial FEM codes are generally immune to lawsuits through “accept use” clauses inserted by company lawyers.

⁸ This statement has to be tempered in two respects. First, the wider availability and growing power of computer algebra systems, outlined in Chapter 4, has widened the realm of analytical solutions than can be obtained within a practical time frame. Second, a combination of analytical and numerical techniques is often effective in reducing the dimensionality of the problem to facilitate parameter studies. Important examples are provided by Fourier analysis, perturbation and boundary-element methods.

§6.4.2. Error Sources and Approximation

Figure 6.1 conveys graphically that each simulation step introduces a source of error. In engineering practice modeling errors are by far the most important. But they are difficult and expensive to evaluate, because *model validation* requires access to and comparison with experimental results. These may be either scarce, or unavailable in the case of a new product in the design stage.

Next in order of importance is the *discretization error*. Even if solution errors are ignored — and usually they can — the computed solution of the discrete model is in general only an approximation in some sense to the exact solution of the mathematical model. A quantitative measurement of this discrepancy is called the *discretization error*. The characterization and study of this error is addressed by a branch of numerical mathematics called approximation theory.

Intuitively one might suspect that the accuracy of the discrete model solution would improve as the number of degrees of freedom is increased, and that the discretization error goes to zero as that number goes to infinity. This loosely worded statement describes the *convergence* requirement of discrete approximations. One of the key goals of approximation theory is to make the statement as precise as it can be expected from a branch of mathematics.⁹

§6.4.3. Other Discretization Methods

It was stated in Chapter 1 that the most popular discretization techniques in structural mechanics are finite element methods and boundary element methods. The finite element method (FEM) is by far the most widely used. The boundary element method (BEM) has gained in popularity for special types of problems, particularly those involving infinite domains, but remains a distant second, and seems to have reached its natural limits.

In non-structural application areas such as fluid mechanics and electromagnetics, the finite element method is gradually making up ground but faces stiff competition from both the classical and energy-based *finite difference* methods. Finite difference and finite volume methods are particularly well entrenched in computational fluid dynamics spanning moderate to high Reynolds numbers.

§6.5. The Finite Element Method

The finite element method (FEM) is the dominant discretization technique in structural mechanics. As discussed in Chapter 1, the FEM can be interpreted from either a physical or mathematical viewpoint. The treatment in Chapters 1–10 emphasizes the former.

The basic concept in the physical FEM is the subdivision of the mathematical model into disjoint (non-overlapping) components of simple geometry called *finite elements* or *elements* for short. The response of each element is expressed in terms of a finite number of degrees of freedom characterized as the value of an unknown function, or functions, at a set of nodal points. The response of the mathematical model is then considered to be approximated by that of the discrete model obtained by connecting or assembling the collection of all elements.

⁹ The discretization error is often overhyped in the FEM literature, since it provides an inexhaustible source of publishable poppycock. If the mathematical model is way off, reducing the discretization error buys nothing; just a more accurate answer to the wrong problem.

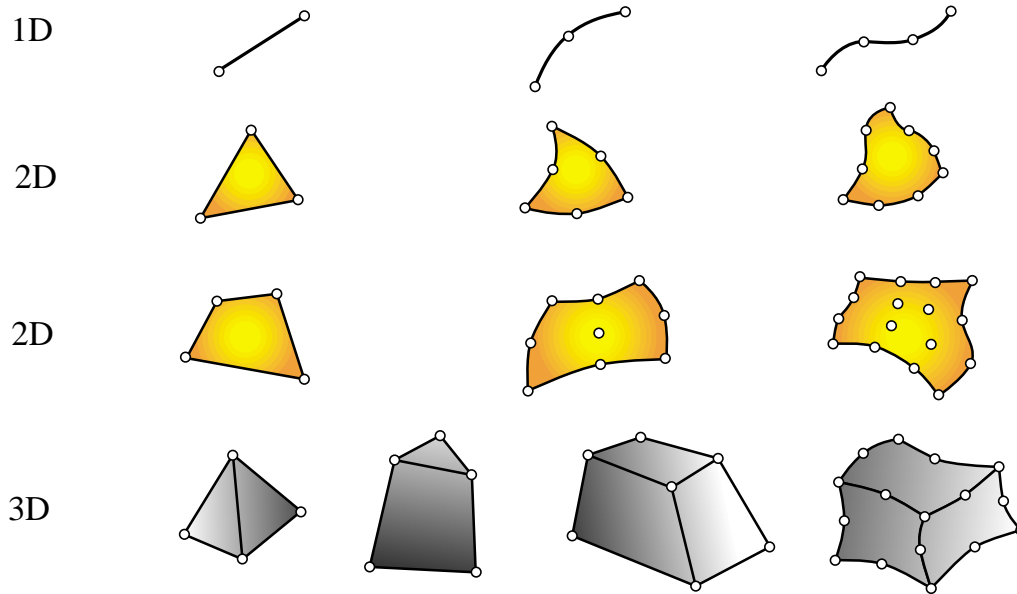


FIGURE 6.3. Typical finite element geometries in one through three dimensions.

The disconnection-assembly concept occurs naturally when examining many artificial and natural systems. For example, it is easy to visualize an engine, bridge, building, airplane, or skeleton as fabricated from simpler components.

Unlike finite difference models, finite elements *do not overlap* in space. In the mathematical interpretation of the FEM, this property goes by the name *disjoint support* or *local support*.

§6.6. Element Attributes

Just like members in the truss example, one can take finite elements of any kind one at a time. Their local properties can be developed by considering them in isolation, as individual entities. This is the key to the modular programming of element libraries.

In the Direct Stiffness Method, elements are isolated by the disconnection and localization steps, which were described for the truss example in Chapter 2. The procedure involves the separation of elements from their neighbors by disconnecting the nodes, followed by referral of the element to a convenient local coordinate system.¹⁰ After that we can consider *generic* elements: a bar element, a beam element, and so on. From the standpoint of the computer implementation, it means that you can write one subroutine or module that constructs, by suitable parametrization, all elements of one type, instead of writing a new one for each element instance.

Following is a summary of the data associated with an individual finite element. This data is used in finite element programs to carry out element level calculations.

§6.6.1. Element Dimensionality

¹⁰ Both steps are only carried out in the modeler's mind. They are placed as part of the DSM for instructional convenience. In practice, processing begins directly at the element level.

Elements can have intrinsic dimensionality of one, two or three space dimensions.¹¹ There are also special elements with zero dimensionality, such as lumped springs or point masses. The intrinsic dimensionality can be expanded as necessary by use of kinematic transformations. For example a 1D element such as a bar, spar or beam may be used to build a model in 2D or 3D space.

§6.6.2. Element Nodes

Each element possesses a set of distinguishing points called *nodal points* or *nodes* for short. Nodes serve a dual purpose: definition of element geometry, and home for degrees of freedom. When a distinction is necessary we call the former *geometric nodes* and the latter *connection nodes*. For most elements studied here, geometric and connector nodes coalesce.

Nodes are usually located at the corners or end points of elements, as illustrated in Figure 6.3. In the so-called refined or higher-order elements nodes are also placed on sides or faces, as well as possibly the interior of the element.

Remark 6.1. In some elements geometric and connection nodes may be at different locations. This is illustrated by the Veubeke equilibrium triangle described in Chapter 15. Some elements have purely geometric nodes, also called *orientation nodes* to complete the definition of certain geometric attributes. An example is the spar element in 3D shown in Figure E5.2, in which a third geometric node is used to define a local plane.

§6.6.3. Element Geometry

The geometry of the element is defined by the placement of the geometric nodal points. Most elements used in practice have fairly simple geometries. In one-dimension, elements are usually straight lines or curved segments. In two dimensions they are of triangular or quadrilateral shape. In three dimensions the most common shapes are tetrahedra, pentahedra (also called wedges or prisms), and hexahedra (also called cuboids or “bricks”). See Figure 6.3.

§6.6.4. Element Degrees of Freedom

The element degrees of freedom (DOF) specify the *state* of the element. They also function as “handles” through which adjacent elements are connected. DOFs are defined as the values (and possibly derivatives) of a primary field variable at connector node points. The actual selection depends on criteria studied at length in Part II. Here we simply note that the key factor is the way in which the primary variable appears in the mathematical model. For mechanical elements, the primary variable is the displacement field and the DOF for many (but not all) elements are the displacement components at the nodes.

§6.6.5. Nodal Forces

There is always a set of nodal forces in a one-to-one correspondence with degrees of freedom. In mechanical elements the correspondence is established through energy arguments.

§6.6.6. Element Constitutive Properties

For a mechanical element these are relations that specify the material behavior. For example, in a linear elastic bar element it is sufficient to specify the elastic modulus E and the thermal coefficient of expansion α .

¹¹ In dynamic analysis, time may appear as an additional dimension.

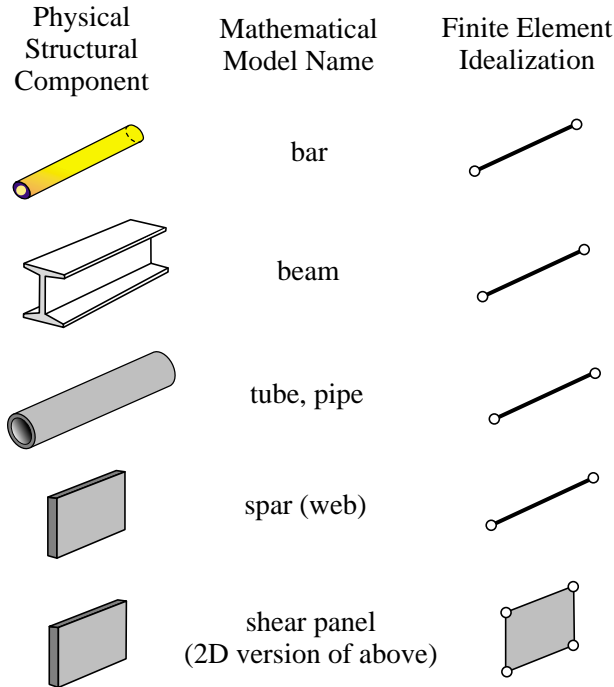


FIGURE 6.4. Examples of primitive structural elements.

§6.6.7. Element Fabrication Properties

For mechanical elements these are fabrication properties which have been integrated out from the element dimensionality. Examples are cross sectional properties of MoM elements such as bars, beams and shafts, as well as the thickness of a plate or shell element.

For computer implementation the foregoing data sets are organized into data structures. These are used by element generation modules to compute element stiffness relations in the local system.

§6.7. Classification of Mechanical Elements

The following classification of finite elements in structural mechanics is loosely based on the “closeness” of the element with respect to the original physical structure. It is given here because it clarifies points that recur in subsequent sections, as well as providing insight into advanced modeling techniques such as hierarchical breakdown and global-local analysis.

§6.7.1. Primitive Structural Elements

These resemble fabricated structural components. They are often drawn as such; see Figure 6.4. The qualifier *primitive* distinguishes them from macroelements, which is another element class described below. Primitive means that they are not decomposable into simpler elements.

These elements are usually derived from Mechanics-of-Materials simplified theories and are better understood from a physical, rather than mathematical, standpoint. Examples are the elements discussed in Chapter 5: bars, cables, beams, shafts, spars.

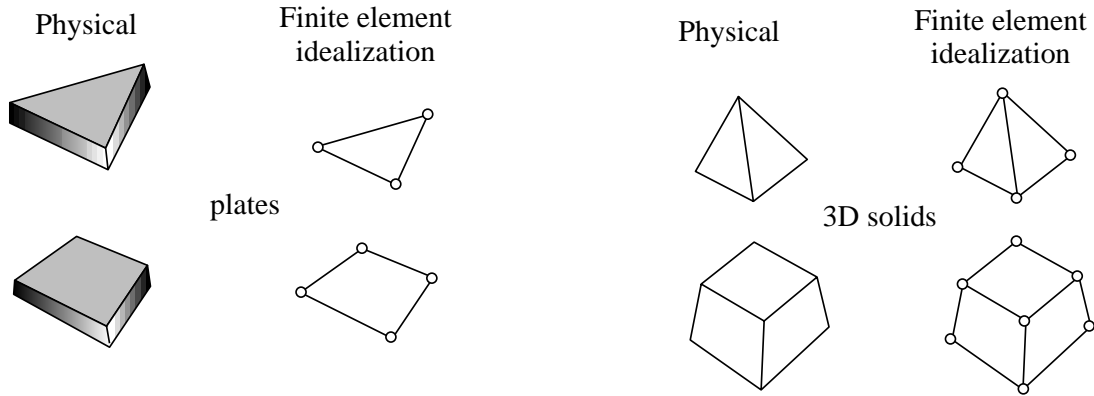


FIGURE 6.5. Continuum element examples.

§6.7.2. Continuum Elements

These do not resemble fabricated structural components at all. They result from the subdivision of “blobs” of continua, or of structural components viewed as continua.

Unlike structural elements, continuum elements are better understood in terms of their mathematical interpretation. Examples: plates, slices, shells, axisymmetric solids, general solids. See Figure 6.5.

§6.7.3. Special Elements

Special elements partake of the characteristics of structural and continuum elements. They are derived from a continuum mechanics standpoint but include features closely related to the physics of the problem. Examples: crack elements for fracture mechanics applications, shear panels, infinite and semi-infinite elements, contact and penalty elements, rigid-body elements. See Figure 6.6.

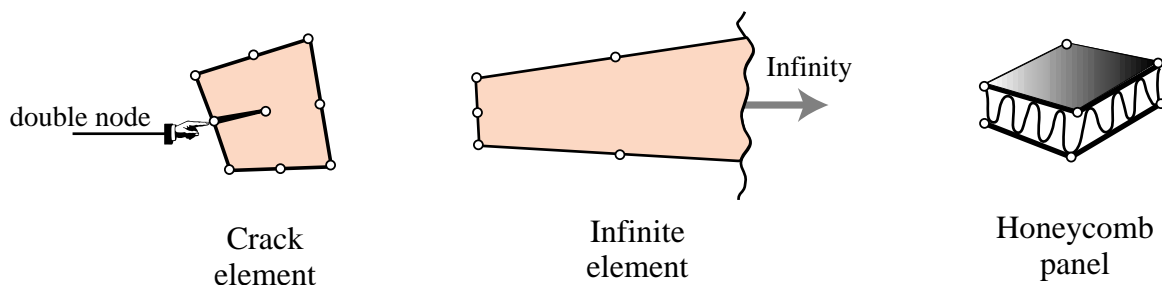


FIGURE 6.6. Special element examples.

§6.7.4. Macroelements

Macroelements are also called mesh units and superelements, although the latter term overlaps with substructures (defined below). These often resemble structural components, but are fabricated with simpler elements. See Figure 6.7.

The main reason for introducing macroelements is to simplify preprocessing tasks. For example, it may be simpler to define a regular 2D mesh using quadrilaterals rather than triangles. The fact that, behind the scene, the quadrilateral is actually a macroelement may not be important to most users.

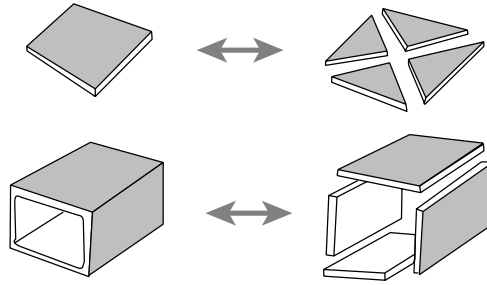


FIGURE 6.7. Macroelement examples.

Similarly a box macroelement can save modeling times for structures that are built by such components; for example box-girder bridges

§6.7.5. Substructures

Also called structural modules and superelements. These are sets of elements with a well defined structural function, typically obtained by cutting the complete structure into functional components. Examples: the wings and fuselage of an airplane; the towers, deck and cables of a suspension bridge.

The distinction between substructures and macroelements is not clear-cut. The main conceptual distinction is that substructures are defined “top down” as parts of a complete structure, whereas macroelements are built “bottom up” from primitive elements. The term *superelement* is often used in a collective sense to embrace element groupings that range from macroelements to substructures. This topic is further covered in Chapter 10.

§6.8. Assembly

The assembly procedure of the Direct Stiffness Method for a general finite element model follows rules identical in principle to those discussed for the truss example. As in that case the process involves two basic steps:

Globalization. The element equations are transformed to a common *global* coordinate system, if necessary.

Merge. The element stiffness equations are merged into the master stiffness equations by appropriate indexing and matrix-entry addition.

The hand calculations for the example truss conceal, however, the implementation complexity. The master stiffness equations in practical applications may involve thousands or even millions of freedoms, and programming can become involved. The topic is elaborated upon in Chapter 25.

§6.9. Boundary Conditions

A key strength of the FEM is the ease and elegance with which it handles arbitrary boundary and interface conditions. This power, however, has a down side. A big hurdle faced by FEM newcomers is the understanding and proper handling of boundary conditions. Below is a simple recipe for treating boundary conditions. The following Chapter provides more specific rules and examples.

§6.9.1. Essential and Natural B.C.

The key thing to remember is that boundary conditions (BCs) come in two basic flavors: essential and natural.

Essential BCs directly affect DOFs, and are imposed on the left-hand side vector \mathbf{u} .

Natural BCs do not directly affect DOFs and are imposed on the right-hand side vector \mathbf{f} .

The mathematical justification for this distinction requires use of concepts from variational calculus, and is consequently relegated to Part II. For the moment, the basic recipe is:

1. If a boundary condition involves one or more degrees of freedom in a *direct* way, it is essential. An example is a prescribed node displacement.
2. Otherwise it is natural.

The term “direct” is meant to exclude derivatives of the primary function, unless those derivatives also appear as degrees of freedom, such as rotations in beams and plates.

§6.9.2. B.C. in Structural Problems

Essential boundary conditions in mechanical problems involve *displacements* (but not strain-type displacement derivatives). Support conditions for a building or bridge problem furnish a particularly simple example. But there are more general boundary conditions that occur in practice. A structural engineer must be familiar with displacement B.C. of the following types.

Ground or support constraints. Directly restraint the structure against rigid body motions.

Symmetry conditions. To impose symmetry or antisymmetry restraints at certain points, lines or planes of structural symmetry. This allows the discretization to proceed only over part of the structure with a consequent savings in modeling effort and number of equations to be solved.

Ignorable freedoms. To suppress displacements that are irrelevant to the problem.¹² Even experienced users of finite element programs are sometimes baffled by this kind. An example are rotational degrees of freedom normal to smooth shell surfaces.

Connection constraints. To provide connectivity to adjoining structures or substructures, or to specify relations between degrees of freedom. Many conditions of this type can be subsumed under the label *multipoint constraints* or *multifreedom constraints*. These can be notoriously difficult to handle from a numerical standpoint, and are covered in Chapters 8–9.

¹² In classical dynamics these are called *ignorable coordinates*.

Notes and Bibliography

Most FEM textbooks do not provide a systematic treatment of modeling. This is no accident: few academic authors have experience with complex engineering systems. Good engineers are too busy (and in demand) to have time for writing books. This gap has been particularly acute since FEM came on the scene because of generational gaps: “real engineers” tend to mistrust the computer, and often for good reason. The notion of explicit versus implicit modeling, which has legal and professional implications, is rarely mentioned.

FEM terminology is by now standard, and so is a majority of the notation. But that is not so in early publications. E.g. \mathbf{K} is universally used¹³ for stiffness matrix in virtually all post-1960 books. There are a few exceptions: the often cited book of Przemieniecki [575] uses \mathbf{S} . There is less unanimity on \mathbf{u} and \mathbf{f} for node displacement and force vectors, respectively; some books such as Zienkiewicz and Taylor [794] still use different symbols.

The element classification given here attempts to systematize dispersed references. In particular, the distinction between macroelements, substructures and superelements is an ongoing source of confusion, particularly since massively parallel computation popularized the notion of “domain decomposition” in the computer science community. The all-encompassing term “superelement” emerged in Norway by 1968 as part of the implementation of the computer program SESAM. Additional historical details are provided in Chapter 10.

The topic of BC classification and handling is a crucial one in practice. More modeling mistakes are done in this aspect of FEM application than anywhere else.

References

Referenced items have been moved to Appendix R.

¹³ A symbol derived from the “spring constant” k that measures the stiffness of a mechanical spring.

7

FEM Modeling: Mesh, Loads and BCs

TABLE OF CONTENTS

	Page
§7.1. Introduction	7–3
§7.2. General Recommendations	7–3
§7.3. Guidelines on Element Layout	7–3
§7.3.1. Mesh Refinement	7–3
§7.3.2. Element Aspect Ratios	7–4
§7.3.3. Physical Interfaces	7–5
§7.3.4. Preferred Shapes	7–5
§7.4. Direct Lumping of Distributed Loads	7–5
§7.4.1. Node by Node Lumping	7–6
§7.4.2. Element by Element Lumping	7–6
§7.4.3. *Weighted Lumping	7–9
§7.4.4. *Energy Consistent Lumping	7–9
§7.5. Boundary Conditions	7–10
§7.6. Support Conditions	7–11
§7.6.1. Supporting Two Dimensional Bodies	7–11
§7.6.2. Supporting Three Dimensional Bodies	7–12
§7.7. Symmetry and Antisymmetry Conditions	7–12
§7.7.1. Symmetry and Antisymmetry Visualization	7–12
§7.7.2. Effect of Loading Patterns	7–13
§7. Notes and Bibliography	7–15
§7. References	7–15
§7. Exercises	7–16

§7.1. Introduction

This Chapter continues the exposition of finite element modeling principles. After some general recommendations, it gives guidelines on layout of finite element meshes, conversion of distributed loads to node forces, and handling the simplest forms of support boundary conditions. The next two Chapters deal with more complicated forms of boundary conditions called multifreedom constraints. The presentation is “recipe oriented” and illustrated by specific examples from structural mechanics. Most examples are two-dimensional. No attempt is made at rigorous justification of rules and recommendations, because that would require mathematical tools beyond the scope of this course.

§7.2. General Recommendations

The general rules that should guide you in the use of commercial or public FEM packages, are¹

- Use the *simplest* type of finite element that will do the job.
- *Never, never, never* mess around with complicated or special elements, unless you are *absolutely sure* of what you are doing.
- Use the *coarsest mesh* you think will capture the dominant physical behavior of the physical system, particularly in *design* applications.

Three word summary: *keep it simple*. Initial FE models may have to be substantially revised to accommodate design changes. There is little point in using complicated models that will not survive design iterations. The time for refinement is when the design has stabilized and you have a better picture of the underlying physics, possibly reinforced by experiments or observation.

§7.3. Guidelines on Element Layout

The following guidelines are stated for structural applications. As noted above, they will be often illustrated for two-dimensional meshes of continuum elements for ease of visualization.

§7.3.1. Mesh Refinement

Use a relatively fine (coarse) discretization in regions where you expect a high (low) *gradient* of strains and/or stresses.² Regions to watch out for high gradients are:

- Near entrant corners (see Remark below) or sharply curved edges.
- In the vicinity of concentrated (point) loads, concentrated reactions, cracks and cutouts.
- In the interior of structures with abrupt changes in thickness, material properties or cross sectional areas.

The examples in Figure 7.1 illustrate some of these “danger regions.” Away from such regions one can use a fairly coarse discretization within constraints imposed by the need of representing the structural geometry, loading and support conditions reasonably well.

¹ Paraphrasing the Bellman in The Hunting of the Snark: “what I say three times is true.”

² *Gradient* is the key word. High gradient means rapid variation. A high value by itself means nothing in this context.

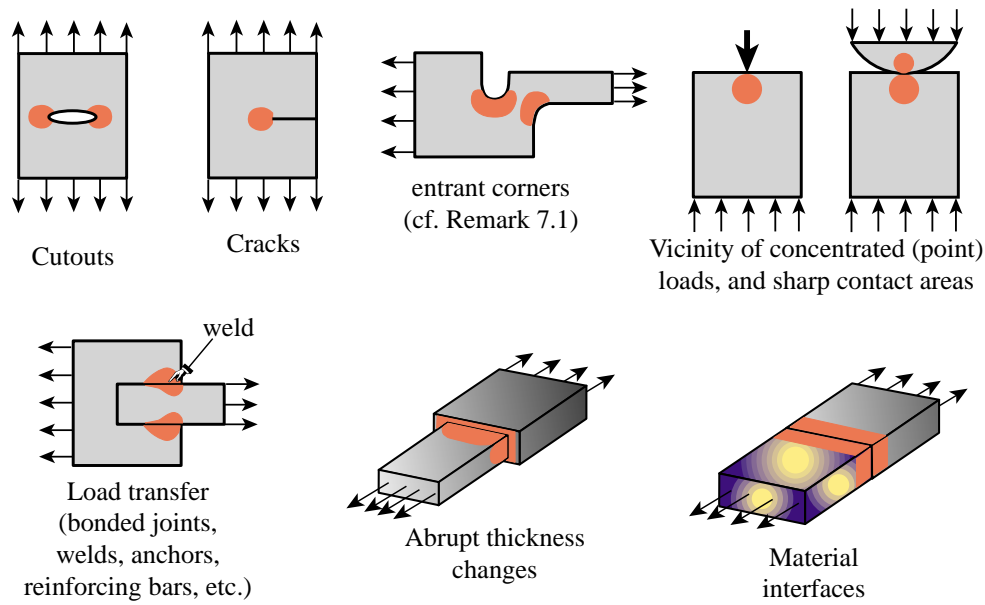


FIGURE 7.1. Some situations where a locally refined finite element discretization (in the red-shaded areas) is recommended.

Remark 7.1. The first bullet above mentions “entrant corner.” That is a region where isostatics (principal stress trajectories) “bunch up.” For a two-dimensional problem mathematically posed on a singly-connected interior domain, they can be recognized as follows. Traverse the boundary CCW so the body or structure is on your left. When hitting a sharp or rounded corner, look at the angle (positive CCW) formed by the *exterior* normals (the normal going toward your right) before and after, measured *from before to after*, and always taking the *positive* value.

If the angle exceeds 180° , it is an entrant corner. [If it is close to 360° , it is a crack tip.] For exterior problems or multiple-connected domains, the definition must be appropriately adjusted.

§7.3.2. Element Aspect Ratios

When discretizing two and three dimensional problems, try to avoid finite elements of high aspect ratios: elongated or “skinny” elements, such as the ones illustrated on the right of Figure 7.2. (The aspect ratio of a two- or three-dimensional element is the ratio between its largest and smallest dimension.)

As a rough guideline, elements with aspect ratios exceeding 3 should be viewed with caution and those exceeding 10 with alarm. Such elements will not necessarily produce bad results — that depends on the loading and boundary conditions of the problem — but do introduce the potential for trouble.

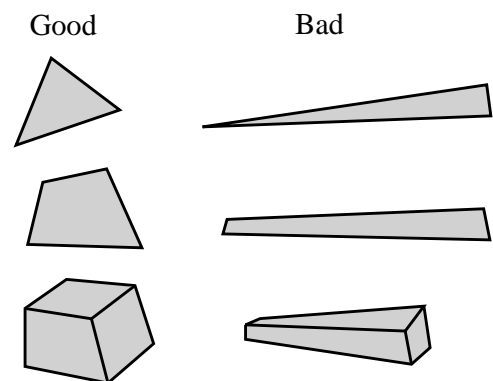


FIGURE 7.2. Elements with good and bad aspect ratios.

Remark 7.2. In many “thin” structures modeled as continuous bodies the appearance of “skinny” elements is inevitable on account of computational economy reasons. An example is provided by the three-dimensional modeling of layered composites in aerospace and mechanical engineering problems.

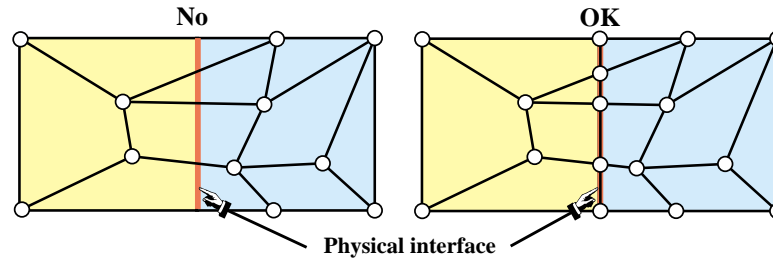


FIGURE 7.3. Illustration of the rule that elements should not cross material interfaces.

§7.3.3. Physical Interfaces

A physical interface, resulting from example from a change in material, should also be an interelement boundary. That is, *elements must not cross interfaces*. See Figure 7.3.

§7.3.4. Preferred Shapes

In 2D FE modeling, if you have a choice between triangles and quadrilaterals with similar nodal arrangement, prefer quadrilaterals. Triangles are quite convenient for mesh generation, mesh transitions, rounding up corners, and the like. But sometimes triangles can be avoided altogether with some thought. One of the homework exercises is oriented along these lines.

In 3D FE modeling, prefer strongly bricks over wedges, and wedges over tetrahedra. The latter should be used only if there is no viable alternative.³ The main problem with tetrahedra and wedges is that they can produce wrong stress results even if the displacement solution looks reasonable.

§7.4. Direct Lumping of Distributed Loads

In practical structural problems, distributed loads are more common than concentrated (point) loads.⁴ Distributed loads may be of surface or volume type.

Distributed surface loads (called surface tractions in continuum mechanics) are associated with actions such as wind or water pressure, snow weight on roofs, lift in airplanes, live loads on bridges, and the like. They are measured in force per unit area.

Volume loads (called body forces in continuum mechanics) are associated with own weight (gravity), inertial, centrifugal, thermal, prestress or electromagnetic effects. They are measured in force per unit volume.

A derived type: line loads, result from the integration of surface loads along one transverse direction, such as a beam or plate thickness, or of volume loads along two transverse directions, such as a bar or beam area. Line loads are measured in force per unit length.

Whatever their nature or source, distributed loads *must be converted to consistent nodal forces* for FEM analysis. These forces end up in the right-hand side of the master stiffness equations.

³ Unfortunately, many existing space-filling automatic mesh generators in three dimensions produce tetrahedral meshes. There are generators that try to produce bricks, but these often fail in geometrically complicated regions.

⁴ In fact, one of the objectives of a good structural design is to avoid or alleviate stress concentrations produced by concentrated forces.

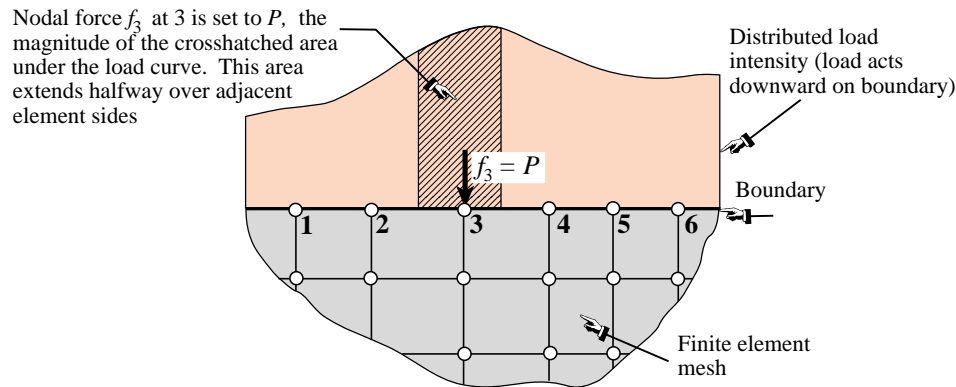


FIGURE 7.4. NbN direct lumping of distributed line load, illustrated for a 2D problem.

The meaning of “consistent” can be made precise through variational arguments, by requiring that the distributed loads and the nodal forces produce the same external work. Since this requires the introduction of external work functionals, the topic is deferred to Part II. However, a simpler approach called *direct load lumping*, or simply *load lumping*, is often used by structural engineers in lieu of the mathematically impeccable but complicated variational approach. Two variants of this technique are described below for distributed surface loads.

§7.4.1. Node by Node Lumping

The node by node (NbN) lumping method is graphically explained in Figure 7.4. This example shows a distributed surface loading acting normal to the straight boundary of a two-dimensional FE mesh. (The load is assumed to have been integrated through the thickness normal to the figure, so it is actually a *line load* measured as force per unit length.)

The procedure is also called *tributary region* or *contributing region* method. For the example of Figure 7.4, each boundary node is assigned a *tributary region* around it that extends halfway to adjacent nodes. The force contribution P of the cross-hatched area is directly assigned to node 3.

This method has the advantage of not requiring the error-prone computation of centroids, as needed in the EbE technique discussed below. For this reason it is often preferred in hand computations.⁵ It can be extended to three-dimensional meshes as well as volume loads.⁶ It should be avoided, however, when the applied forces vary rapidly (within element length scales) or act only over portions of the tributary regions.

§7.4.2. Element by Element Lumping

In this variant the distributed loads are divided over element domains. The resultant load is assigned to the centroid of the load diagram, and apportioned to the element nodes by statics. A node force is obtained by adding the contributions from all elements meeting at that node. The procedure is illustrated in Figure 7.5, which shows details of the computation over 2–3. The total force at node 3, for instance, would be that contributed by segments 2–3 and 3–4. For the frequent case in which

⁵ It has been extensively used in the aircraft industry for smooth-varying pressure loads computations.

⁶ The computation of tributary areas and volumes for general 2D and 3D regions can be done through the so-called Voronoi diagrams. This is an advanced topic in computational geometry (see, e.g., [177]) and thus not treated here.

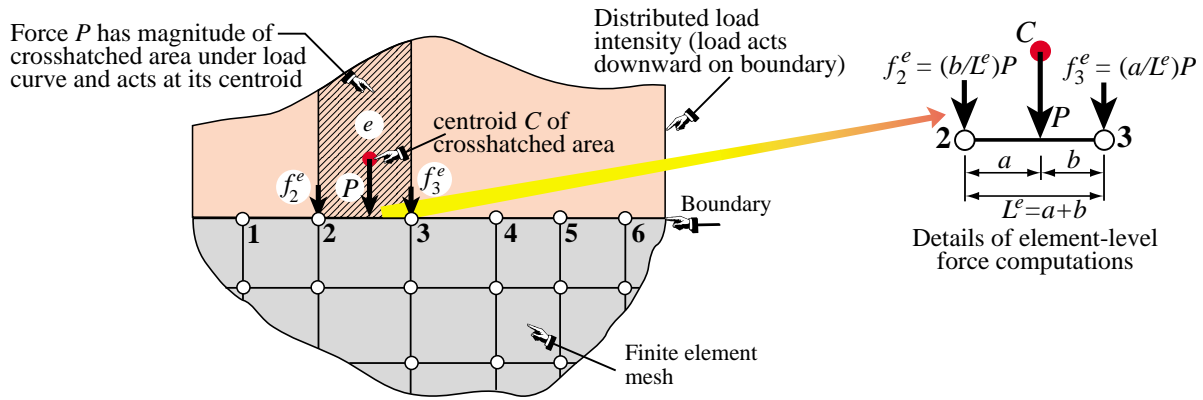


FIGURE 7.5. EbE direct lumping of distributed line load, illustrated for a 2D problem.

the variation of the load over the element is linear (so the area under the load is a trapezoid) the node forces can be computed directly by the formulas given in Figure 7.6.

If applicable, EbE is more accurate than NbN lumping. In fact it agrees with consistent node lumping for simple elements that possess only corner nodes. In those cases it is not affected by sharpness of the load variation and can be even used for point loads that are not applied at the nodes.

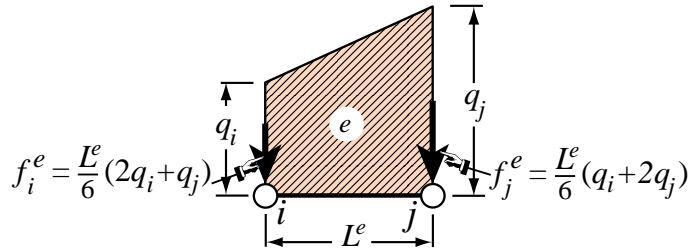


FIGURE 7.6. EbE lumping of linearly varying line load over element.

The EbE procedure is not applicable if the centroidal resultant load cannot be apportioned by statics. This happens if the element has midside faces or internal nodes in addition to corner nodes, or if it has rotational degrees of freedom. For those elements the variational-based consistent approach covered in Part II and briefly outlined in §7.3.3, is preferable.

Example 7.1. Figures 7.7(a,b) show web-downloaded pictures of the Norfork Dam, a 220-ft high, concrete-built gravity dam. Its typical cross section is shown in 7.7(c). The section is discretized by triangular elements as illustrated in Figure 7.7(d). The dam has a length of 2624 ft and was constructed over the White River in Arkansas over 1941–44.⁷ The structure is assumed to be in plane strain. Accordingly the FEM model shown in 7.7(d) is a typical 1-ft slice of the dam and near-field soil.

In the analysis of dam and marine structures, a “wet node” of a FEM discretization is one in contact with the water. The effect of hydrostatic pressure is applied to the structure through nodal forces on wet nodes. The wet nodes for a water head of 180 ft over the riverbed are shown in Figure 7.8(b). Nodes are numbered 1 through 9 for convenience. The pressure in psf (lbs per sq-ft) is $p = 62.4d$ where d is the depth in ft. Compute the horizontal hydrostatic nodal forces f_{x1} and f_{x2} using NbN and EbE, assuming that the wet face AB is vertical for simplicity.

Node by Node. For node 1 go halfway to 2, a distance of $(180 - 136)/2 = 22$ ft. The tributary load area is a triangle extending 22 ft along y with bottom pressure $62.4 \times 22 = 1372.8$ psf and unit width normal to

⁷ As discussed in **Notes and Bibliography**, this example has historical significance, as the first realistic Civil Engineering structure modeled ca. 1960 by the Finite Element Method, which until then had been largely confined to aerospace.

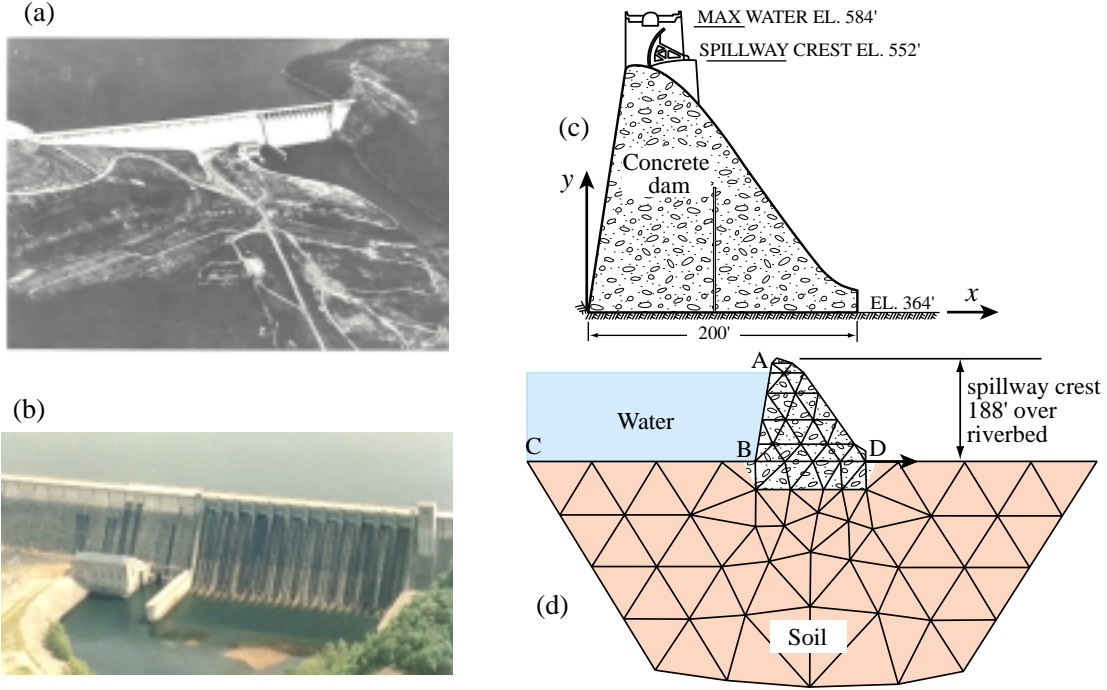


FIGURE 7.7. Norfolk Dam: (a,b) pictures; (c) cross section of dam above foundation (line inside dam is a thermally induced crack considered in the 1960 study); (d) coarse mesh including foundation and soil but not crack [127].

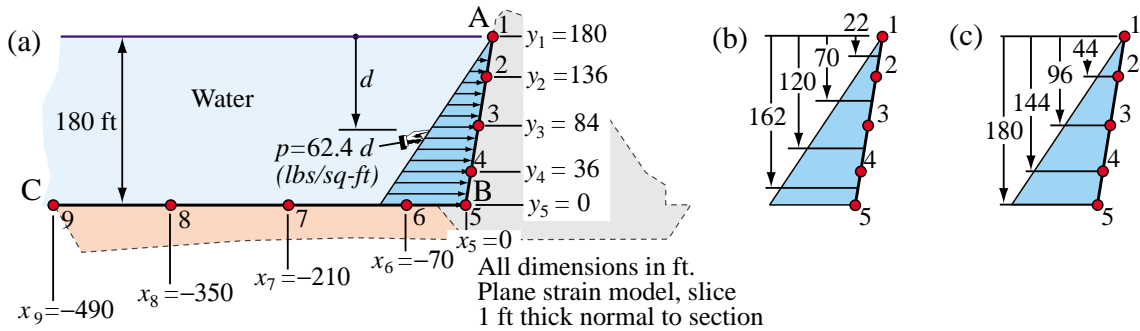


FIGURE 7.8. Norfolk Dam example: (a) computation of wet node forces due to hydrostatic pressure; (b) NbN tributary regions; (c) EbE regions.

paper, giving $f_{x1} = \frac{1}{2} 22 \times 1372.8 = 15101$ lbs. For node 2 go up 22 ft and down $(136 - 84)/2 = 26$ ft. The tributary load area is a trapezoid extending $22 + 26 = 48$ ft vertically, with pressures 1372.8 psf at the top and $62.4 \times 70 = 4368.0$ psf at the bottom. This gives $f_{x2} = 48 \times (1372.8 + 4368.0)/2 = 137779$ lbs.

Element by Element. It is convenient to pre-compute hydro pressures at wet node levels: $p_1 = 0, p_2 = 62.4 \times 44 = 2745.6$ psf, $p_3 = 62.4 \times 96 = 5990.4$. Because the variation of p is linear, the formulas of Figure 7.6 can be applied directly: $f_{x1}^{(1)} = (44/6)(2 \times 0 + 2745.6) = 20134$ lbs, $f_{x2}^{(1)} = (44/6)(0 + 2 \times 2745.6) = 40269$ lbs and $f_{x2}^{(2)} = (52/6)(2 \times 2745.6 + 5990.4) = 99507$ lbs. Adding contributions to nodes 1 and 2: $f_{x1} = f_{x1}^{(1)} = 20134$ lbs and $f_{x2} = f_{x2}^{(1)} + f_{x2}^{(2)} = 40269 + 99507 = 139776$ lbs.

The computations for wet nodes 3 through 9 are left as an exercise.

Example 7.2. Figure 7.9 shows the mesh for the $y > 0$ portion of the gravity dam of the previous example.

(The node numbering is different). The specific weight for concrete of $\gamma = 200$ pcf (pounds per cubic foot). Compute the node force f_{y11} due to own weight.

For this calculation NbN is unwieldy because computation of the nodal tributary region requires construction of a Voronoi diagram. (Furthermore for constant specific weight it gives the same answer as EbE.) To apply EbE, select the elements that contribute to 11: the six triangles (9), (10), (11), (15), (16) and (17).

The area of a triangle with corners at $\{x_1, y_1\}$, $\{x_2, y_2\}$ and $\{x_3, y_3\}$ is given by $A = (x_2y_3 - x_3y_2) + (x_3y_1 - x_1y_3) + (x_1y_2 - x_2y_1)$. Applying this to the geometry of the figure one finds that the areas are $A^{(9)} = A^{(10)} = A^{(11)} = A^{(15)} = A^{(16)} = A^{(17)} =$. The weight forces on each element are $W^{(9)} = \gamma h$, $W^{(10)} = \gamma h$, $W^{(11)} = \gamma h$, $W^{(15)} = \gamma h$, $W^{(16)} = \gamma h$, and $W^{(17)} = \gamma h$, where h is the thickness normal to the figure (1 ft here).

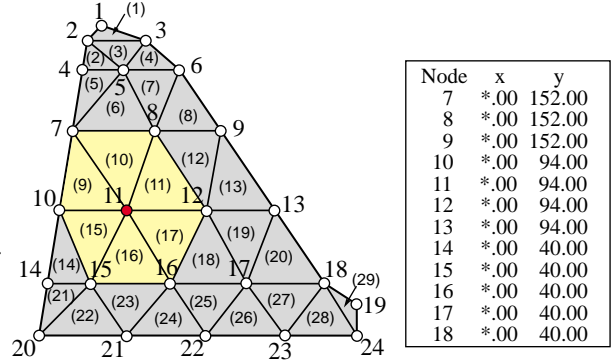


FIGURE 7.9. Computation of weight force at node 11 of gravity dam example of Figure 7.7(d).

For uniform element thickness and specific weight, one third of each force goes to each element corner. Thus node 11 receives

$$f_{y11} = \frac{1}{3}(W^{(9)} + W^{(10)} + W^{(11)} + W^{(17)} + W^{(18)}) = \quad (7.1)$$

(example incomplete, TBF)

§7.4.3. *Weighted Lumping

The NbN and EbE methods are restricted to simple elements, specifically those with corner nodes only. We outline here a general method that works for more complicated models. The mathematical justification requires energy theorems covered in Part II. Thus at this stage the technique is merely presented as a recipe.

To fix the ideas consider again the 2D situation depicted in Figures 7.4 and 7.5. Denote the distributed load by $q(x)$. We want to find the lumped force f_n at an interior node n of coordinate x_n . Let the adjacent nodes be $n - 1$ and $n + 1$, with coordinates x_{n-1} and x_{n+1} , respectively. Introduce a *weight function* $W_n(x)$ with properties to be specified below. The lumped force is given by

$$f_n = \int_{x_{n-1}}^{x_{n+1}} W_n(x) q(x) dx \quad (7.2)$$

For this formula to make sense, the weight function must satisfy several properties:

1. Unit value at node n : $W_n(x_n) = 1$.
2. Vanishes over any element not pertaining to n : $W_n = 0$ if $x \leq x_{n-1}$ or $x \geq x_{n+1}$
3. Gives the same results as NbN or EbE for constant q over elements with corner nodes only.

Both NbN and EbE are special cases of (7.2). For NbN pick

$$W_n(x) = 1 \quad \text{if} \quad \frac{1}{2}(x_{n-1} + x_n) \leq x \leq \frac{1}{2}(x_n + x_{n+1}), \quad w_n(x) = 0 \quad \text{otherwise.} \quad (7.3)$$

For EbE pick

$$W_n(x) = \begin{cases} 1 - \frac{x - x_{n-1}}{x_n - x_{n-1}} & \text{if } x_{n-1} \leq x \leq x_n, \\ 1 - \frac{x - x_n}{x_{n+1} - x_n} & \text{if } x_n \leq x \leq x_{n+1}, \\ 0 & \text{otherwise.} \end{cases} \quad (7.4)$$

These particular weight functions are depicted in Figure 7.10.

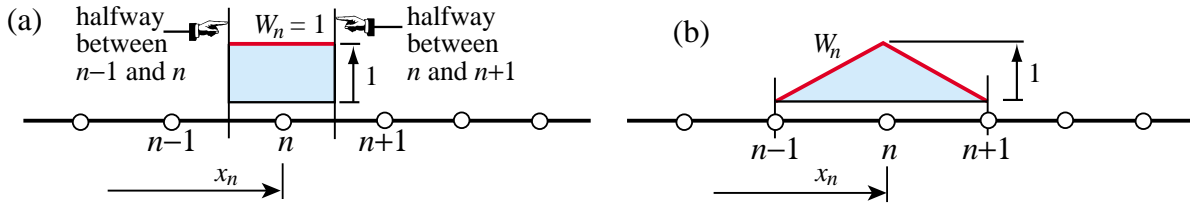


FIGURE 7.10. Weight functions corresponding to: (a) NbN lumping, and (b) EbE lumping.

§7.4.4. *Energy Consistent Lumping

The rule (7.2) can be justified from the standpoint of the Principle of Virtual Work. Let δu_n be a virtual node displacement paired to f_n . Take $\delta u_n W_n(x)$ to be the associated displacement variation. The external virtual work of $q(x)$ is $\int q(x) W_n(x) \delta u_n dx$ extended over the portion where $W_n \neq 0$. Equating this to $f_n \delta u_n$ and cancelling δu_n from both sides yields (7.2).

If W_n is the *trial displacement function* actually used for the development of element equations the lumping is called *energy consistent*, or *consistent* for short.

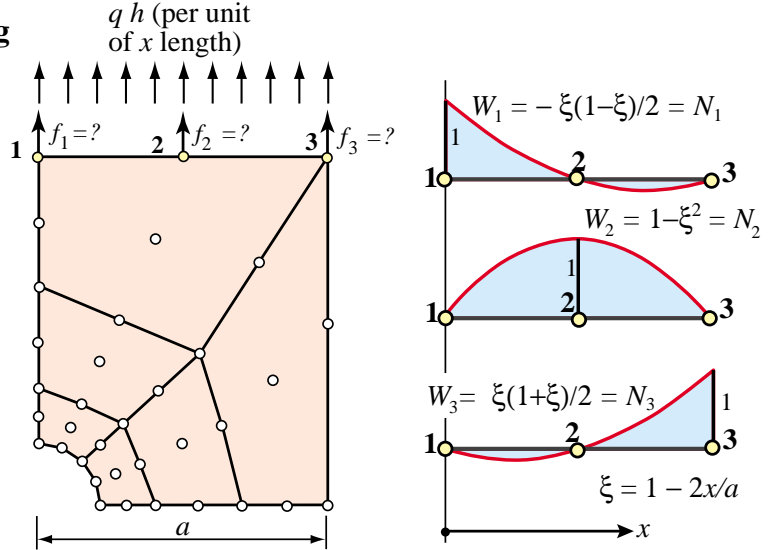


FIGURE 7.11. Example of consistent load lumping.

(As will be seen later, trial functions are the union of shape functions over the patch of all elements connected to node n .) This important technique is studied in Part II of the course after energy methods are introduced.

Example 7.3. Consider the mesh of 9-node quadrilaterals shown in Figure 7.11. (This is later used as a benchmark problem in Chapter 27.) The upper plate edge 1–3 is subject to a uniform normal load qh per unit length, where h is the plate thickness. The problem is to compute the node forces f_1 , f_2 and f_3 . If 1–2 and 2–3 were on two different elements, both NbN and EbE would give $f_1 = f_3 = \frac{1}{4}qha$ and $f_2 = \frac{1}{2}qha$. But this lumping is wrong for an element with midside nodes. Instead, pick the weight functions W_i ($i = 1, 2, 3$) shown on the right of Figure 7.11.

Since there is only one element on the loaded edge, the W_i are actually the quadratic shape functions N_i for a 3-node line element, developed in later Chapters. The dimensionless variable ξ is called an *isoparametric natural coordinate*. Applying the rule (7.2) we get

$$f_1 = \int_0^a W_1(x) q h dx = \int_{-1}^1 W_1(\xi) q h \frac{dx}{d\xi} d\xi = \int_{-1}^1 -\frac{1}{2}\xi(1-\xi) q h \left(\frac{1}{2}a\right) d\xi = \frac{1}{6}qha. \quad (7.5)$$

Similarly $f_2 = \frac{2}{3}qha$ and $f_3 = f_1$. As a check, $f_1 + f_2 + f_3 = qha$, which is the total load acting on the plate edge.

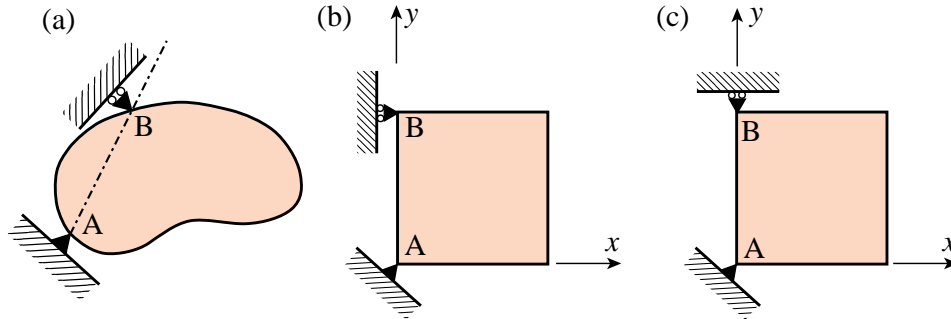


FIGURE 7.12. Suppressing two-dimensional rigid body motions.

§7.5. Boundary Conditions

The key distinction between *essential* and *natural* boundary conditions (BC) was introduced in the previous Chapter. The distinction is explained in Part II from a variational standpoint. In this section we discuss the simplest *essential* boundary conditions in structural mechanics from a physical standpoint. This makes them relevant to problems with which a structural engineer is familiar. Because of the informal setting, the ensuing discussion relies heavily on examples.

In structural problems formulated by the DSM, the recipe of §6.7.1 that distinguishes between essential and natural BC is: if it directly involves the nodal freedoms, such as displacements or rotations, it is essential. Otherwise it is natural. Conditions involving applied loads are natural. Essential BCs take precedence over natural BCs.

The simplest essential boundary conditions are support and symmetry conditions. These appear in many practical problems. More exotic types, such as multifreedom constraints, require more advanced mathematical tools and are covered in the next two Chapters.

§7.6. Support Conditions

Supports are used to restrain structures against relative rigid body motions. This is done by attaching them to Earth ground (through foundations, anchors or similar devices), or to a “ground structure” which is viewed as the external environment.⁸ The resulting boundary conditions are often called *motion constraints*. In what follows we analyze two- and three-dimensional motions separately.

§7.6.1. Supporting Two Dimensional Bodies

Figure 7.12 shows two-dimensional bodies that move in the plane of the paper. If a body is not restrained, an applied load will cause infinite displacements. Regardless of loading conditions, the body must be restrained against two translations along x and y , and one rotation about z . Thus the minimum number of constraints that has to be imposed in two dimensions is *three*.

In Figure 7.12, support A provides *translational* restraint, whereas support B, together with A, provides *rotational* restraint. In finite element terminology, we say that we *delete* (fix, remove, preclude) all translational displacements at point A, and that we delete the translational degree of

⁸ For example, the engine of a car is attached to the vehicle frame through mounts. The car frame becomes the “ground structure,” which moves with respect to Earth ground, as Earth rotates and moves through space, etc.

freedom directed along the normal to the AB direction at point B. This body is free to distort in any manner without the supports imposing any deformation constraints.

Engineers call A and B *reaction-to-ground points*. This means that if the supports are conceptually removed, applied loads are automatically balanced by reactive forces at A and B, in accordance with Newton's third law. Additional freedoms may be precluded to model greater restraint by the environment. However, Figure 7.12(a) does illustrate the *minimal* number of constraints.

Figure 7.12(b) is a simplification of Figure 7.12(a). Here the line AB is parallel to the global y axis. We simply delete the x and y translations at point A, and the x translation at point B. If the roller support at B is modified as in Figure 7.12(c), however, it becomes ineffective in constraining the infinitesimal rotational motion about point A because the rolling direction is normal to AB. The configuration of Figure 7.12(c) is called a *kinematic mechanism*, and will result in a singular modified stiffness matrix.

§7.6.2. Supporting Three Dimensional Bodies

Figure 7.13 illustrates the extension of the freedom-restraining concept to three dimensions. The minimal number of freedoms that have to be constrained is now *six* and many combinations are possible. In the example of Figure 7.13, all three degrees of freedom at point A have been fixed. This prevents all rigid body translations, and leaves three rotations to be taken care of. The x displacement component at point B is deleted to prevent rotation about z , the z component is deleted at point C to prevent rotation about y , and the y component is deleted at point D to prevent rotation about x .

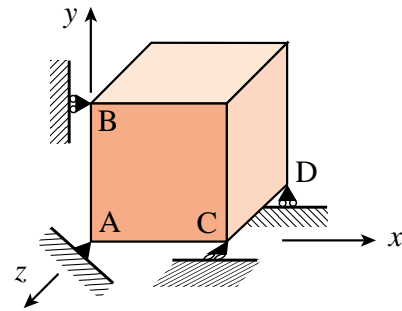


FIGURE 7.13. Suppressing rigid body motions in a three-dimensional body.

§7.7. Symmetry and Antisymmetry Conditions

Engineers doing finite element analysis should be on the lookout for conditions of *symmetry* or *antisymmetry*. Judicious use of these conditions allows only a portion of the structure to be analyzed, with a consequent saving in data preparation and computer processing time.⁹

§7.7.1. Symmetry and Antisymmetry Visualization

Recognition of symmetry and antisymmetry conditions can be done by either visualization of the displacement field, or by imagining certain rotational or reflection motions. Both techniques are illustrated for the two-dimensional case.

A *symmetry line* in two-dimensional motion can be recognized by remembering the “mirror” displacement pattern shown in Figure 7.14(a). Alternatively, a 180° rotation of the body about the symmetry line reproduces exactly the original problem.

An *antisymmetry line* can be recognized by the displacement pattern illustrated in Figure 7.14(b).

⁹ Even if symmetry or antisymmetry are not explicitly applied through boundary conditions, they provide valuable checks on the computed solution.

Alternatively, a 180° rotation of the body about the antisymmetry line reproduces exactly the original problem except that all applied loads are reversed.

Similar recognition patterns can be drawn in three dimensions to help visualization of *planes* of symmetry or antisymmetry. More complex regular patterns associated with *sectorial* symmetry (also called *harmonic* symmetry) as well as *rotational* symmetry can be treated in a similar manner, but will not be discussed here.

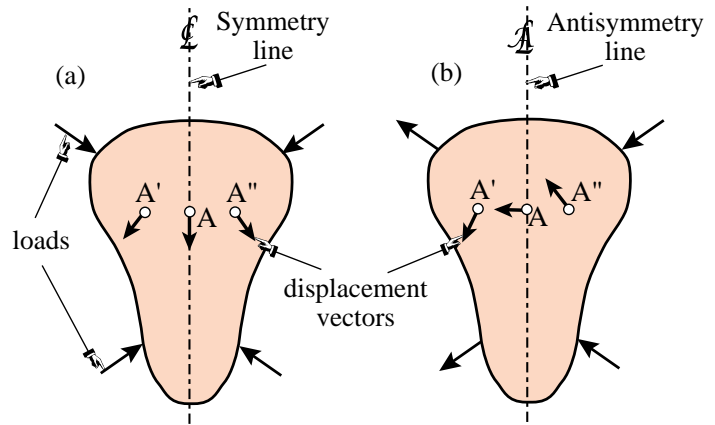


FIGURE 7.14. Visualizing symmetry and antisymmetry lines.

§7.7.2. Effect of Loading Patterns

Although the structure may look symmetric in shape, it must be kept in mind that model reduction can be used only if the loading conditions are also symmetric or antisymmetric.

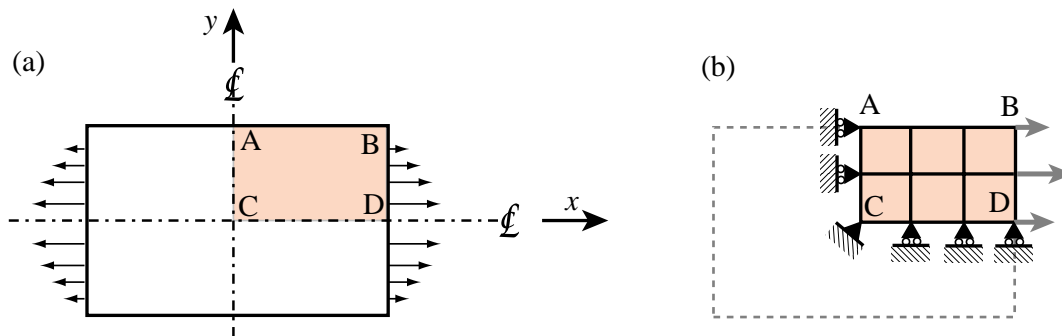


FIGURE 7.15. A doubly symmetric structure under symmetric loading.

Consider the plate structure shown in Figure 7.15(a). This structure is symmetrically loaded on the x - y plane. Applying the recognition patterns stated above one concludes that the structure is *doubly symmetric* in both geometry and loading. It is evident that no displacements in the x -direction are possible for any point on the y -axis, and that no y displacements are possible for points on the x axis. A finite element model of this structure may look like that shown in Figure 7.15(b).

On the other hand if the loading is *antisymmetric*, as illustrated in Figure 7.16(a), the x axis becomes an *antisymmetry line* as none of the $y = 0$ points can move along the x direction. The boundary conditions to be imposed on the FE model are also different, as shown in Figure 7.16(b).

Remark 7.3. For the case shown in Figure 7.16(b) note that all rollers slide in the same direction. Thus the vertical rigid body motion along y is not precluded. To do that, one node has to be constrained in the y direction. If there are no actual physical supports, the choice is arbitrary and amounts only to an adjustment on the overall (rigid-body) vertical motion. In Figure 7.16(b) the center point C has been so chosen. But any other node could be selected as well; for example A or D. The important thing is *not to overconstrain* the structure by applying more than one y constraint.

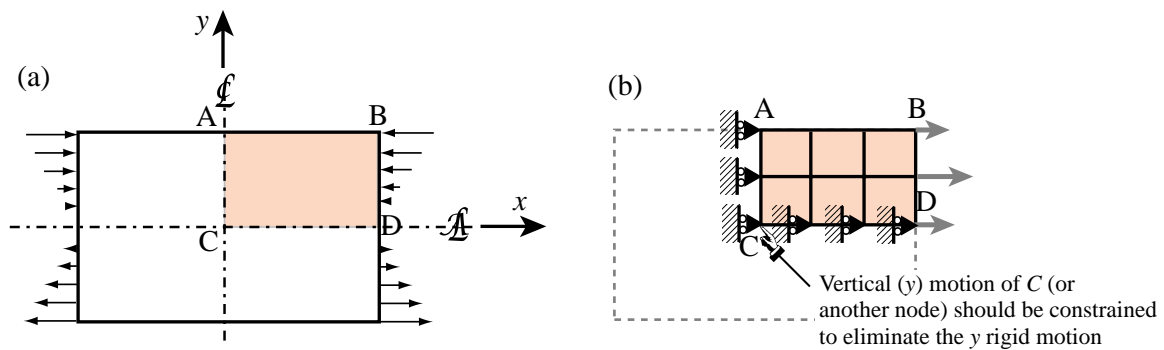


FIGURE 7.16. A doubly symmetric structure under antisymmetric loading.

Remark 7.4. Point loads acting at nodes located on symmetry or antisymmetry lines require special care. For example, consider the doubly symmetric plate structure of Figure 7.15 under the two point loads of magnitude P , as pictured in Figure 7.17(a). If the structure is broken down into 4 quadrants as in Figure 7.17(b), P must be halved as indicated in Figure 7.17(c). The same idea applies to point loads on antisymmetry lines, but there the process is trickier, as illustrated in Figure 7.18. The load must not be applied if the node is fixed against motion, since then the node force will appear as a reaction.

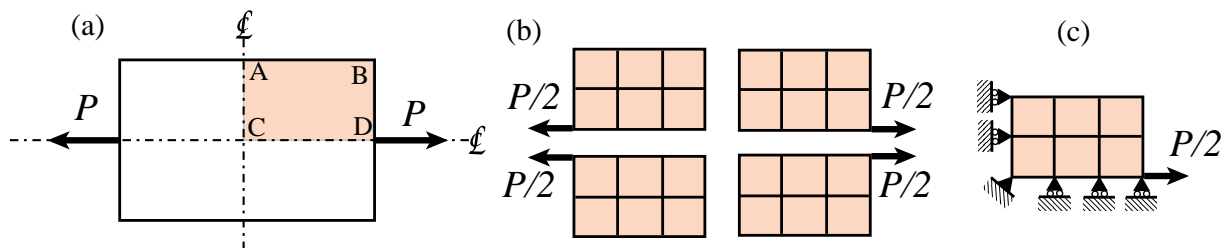


FIGURE 7.17. Breaking up a point load acting on a symmetry line node.

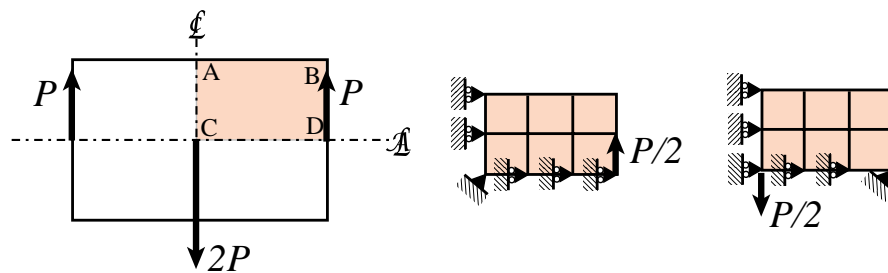


FIGURE 7.18. Breaking up a point load acting on an antisymmetry line node.

Distributed loads should not be divided when the structure is broken down into pieces, since the lumping process will take care of the necessary apportionment to nodes.

Notes and Bibliography

FEM modeling rules in most textbooks are often diffuse, if given at all. In those that focus on the mathematical interpretation of FEM they are altogether lacking; the emphasis being on academic boundary value problems. The rule collection at the start of this Chapter attempts to collect important recommendations in one place.

The treatment of boundary conditions, particularly symmetry and antisymmetry, tends to be also flaky. A notable exception is [362], which is understandable since Irons worked in industry (at Rolls-Royce Aerospace Division) before moving to academia.

The Norfolk Dam used in Examples 7.1 and 7.2 (and two Exercises) for hydrostatic load-lumping calculations was the first realistic Civil Engineering structure analyzed by FEM. It greatly contributed to the acceptance of the method beyond the aerospace industry where it had originated. How this seminal event came to pass is narrated by Wilson in [718], from which the following fragment is taken. Annotations are inserted in squared brackets.

“On the recommendation from Dr. Roy Carlson, a consultant to the Little Rock District of the Corps of Engineers, Clough [then a Professor at UC Berkeley] submitted [in 1960] a proposal to perform a finite element analysis of Norfolk Dam, a gravity dam that had a temperature induced vertical crack near the center of the section. The proposal contained a coarse mesh solution that was produced by the new program [a matrix code developed by E. L. Wilson, then a doctoral student under Clough’s supervision; the mesh is that shown in Figure 7.7(d)] and clearly indicated the ability of the new method to model structures of arbitrary geometry with different orthotropic properties within the dam and foundation. The finite element proposal was accepted by the Corps over an analog computer proposal submitted by Professor Richard MacNeal of CalTech [who later directed the development of NASTRAN under a NASA contract in the late 1960s], which at that time was considered as the state-of-the-art method for solving such problems.

The Norfolk Dam project provided an opportunity to improve the numerical methods used within the program and to extend the finite element method to the nonlinear solution of the crack closing due to hydrostatic loading. Wilson and a new graduate student, Ian King, conducted the detailed analyses that were required by the study. The significant engineering results of the project indicated that the cracked dam was safe [since the crack would be closed as the reservoir was filled].”

References

Referenced items moved to Appendix R.

Homework Exercises for Chapters 6 and 7

FEM Modeling: Mesh, Loads and BCs

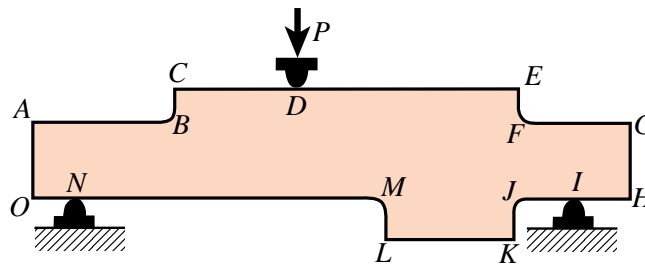


FIGURE E7.1. Inplane bent plate for Exercise 7.1.

EXERCISE 7.1 [D:10] The plate structure shown in Figure E7.1 is loaded and deforms in the plane of the paper. The applied load at D and the supports at I and N extend over a fairly narrow area. List what you think are the likely “trouble spots” that would require a locally finer finite element mesh to capture high stress gradients. Identify those spots by its letter and a reason. For example, D : vicinity of point load.

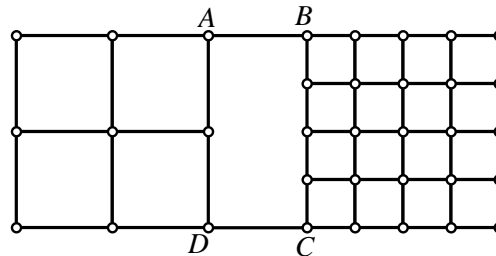


FIGURE E7.2. Transition meshing for Exercise 7.2.

EXERCISE 7.2 [D:15] Part of a two-dimensional FE mesh has been set up as indicated in Figure E7.2. Region $ABCD$ is still unmeshed. Draw a *transition mesh* within that region that correctly merges with the regular grids shown, uses 4-node quadrilateral elements (quadrilaterals with corner nodes only), and *avoids triangles*. *Note*: There are several (equally acceptable) solutions.

EXERCISE 7.3 [A:15] A rectangular plate of constant thickness h and inplane dimensions $8a$ and $6a$ is meshed with 8 rectangular elements as shown in Figure E7.3(a). The plate specific weight is γ and acts along the $-y$ axis direction.

- Compute the node forces due to plate weight at nodes 1 through 15, using the NbN method. Obtain the node-tributary regions as sketched in Figure E7.3(b), which shows each element divided by the medians drawn as dashed lines (the tributary region of node 7 is shown in yellow). Partial answer: $f_{y1} = -2a^2\gamma h$. Check that adding up all y forces at the 15 nodes one gets $W = -48a^2\gamma h$.
- Repeat the computations using the EbE method. For this, take the total weight force on each element, and assign one quarter to each corner node. (This agrees with consistent energy lumping for 4-node rectangular elements.) Do the results agree with NbN lumping?

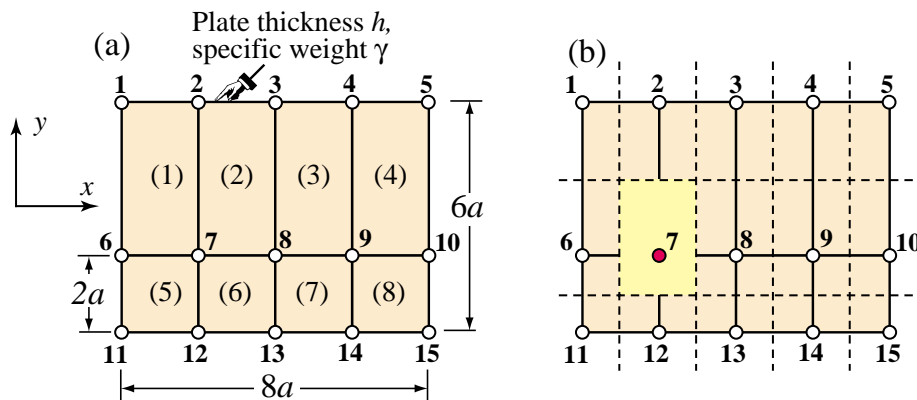


FIGURE E7.3. (a) Mesh layout for Exercise 7.3. (b) shows tributary area for node 7.

EXERCISE 7.4 [N/C:20] Complete the computation of hydrostatic node forces on the Norfolk Dam under a water head of 180 ft, initiated in Example 7.1, using the data of Figure 7.8, and either the NbN or EbE method (pick one). Assume face AB is vertical. Do two checks: sum of horizontal (x) forces on nodes 1 through 5 is $\frac{1}{2}180^2 \times 62.4$ lbs, and sum of vertical (y) forces on nodes 5 through 9 is $-180 \times 62.4 \times 490$ lbs.

EXERCISE 7.5 [N/C:25] Complete the computation of own weight nodal forces of the coarse mesh of the Norfolk dam proper, initiated in Example 7.2, reusing the data of Figure 7.9. Use the EbE method. Recover coordinates, and write a computer program to compute node forces. Compute the total weight of the dam slice in lbs.

EXERCISE 7.6 [A:10] Figure E7.4 depicts two instances of a pull test. In (a) a stiffer material (steel rod) is pulled out of a softer one (concrete block); in this case the load transfer diagram shows a rapid variation near the inner end of the rod. In (b) a softer material (plastic rod) is pulled out of a stiffer one (concrete rod), and the load transfer diagram is reversed.

If the test is to be simulated by a finite element model, indicate for (a) and (b) where a finer mesh would be desirable. Explain.

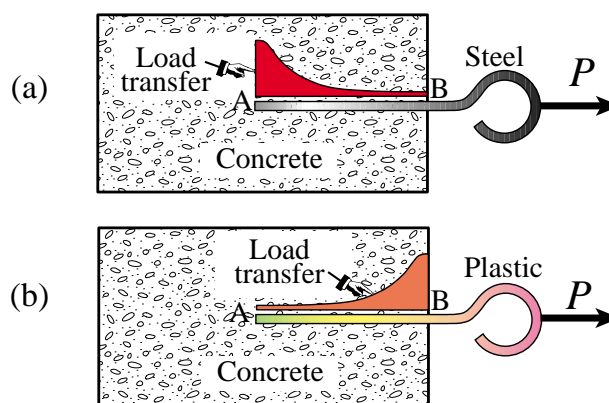


FIGURE E7.4. Pull tests for Exercise 7.6.

EXERCISE 7.7 [D:20] Identify the symmetry and antisymmetry lines in the two-dimensional problems illustrated in Figure E7.5. They are: (a) a circular disk under two diametrically opposite point forces (the famous “Brazilian test” for concrete); (b) the same disk under two diametrically opposite force pairs; (c) a clamped semiannulus under a force pair oriented as shown; (d) a stretched rectangular plate with a central circular hole. Finally (e) and (f) are half-planes under concentrated loads.¹⁰

Having identified those symmetry/antisymmetry lines, state whether it is possible to cut the complete structure to one half or one quarter before laying out a finite element mesh. Then draw a coarse FE mesh indicating, with

¹⁰ Note that (e) is the famous Flamant’s problem, which is important in the 2D design of foundations of civil structures. The analytical solution of (e) and (f) may be found, for instance, in Timoshenko-Goodier’s *Theory of Elasticity*, 2nd Edition, page 85ff.

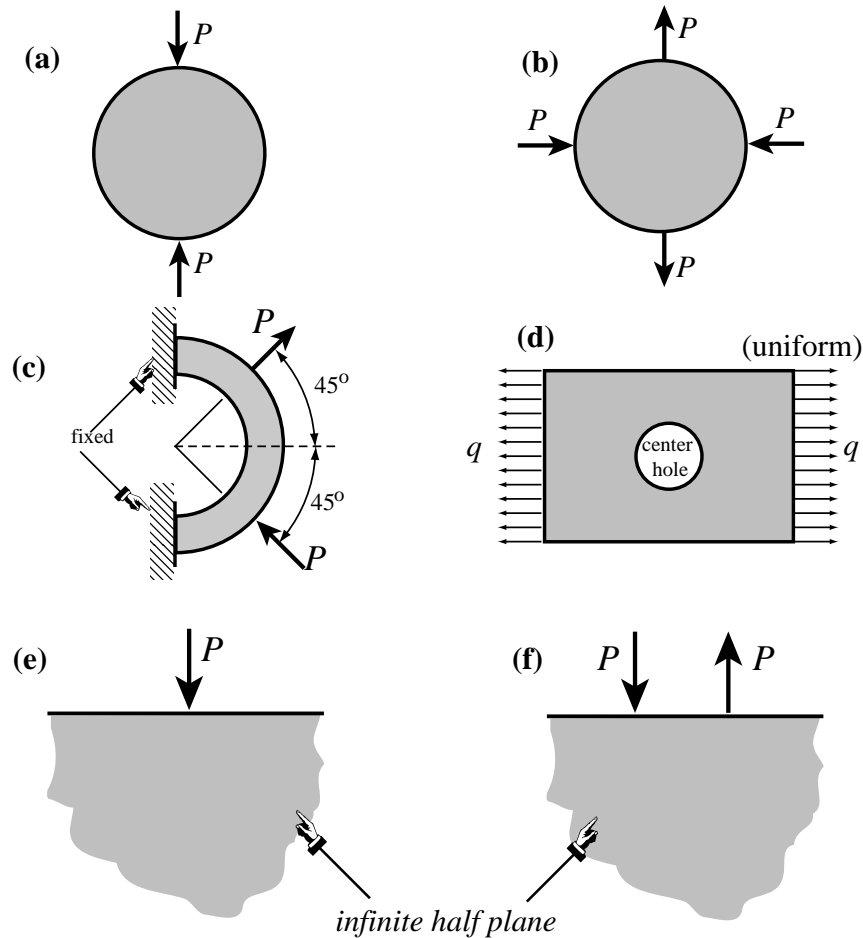


FIGURE E7.5. Problems for Exercise 7.7.

rollers or fixed supports, which kind of displacement BCs you would specify on the symmetry or antisymmetry lines. *Note: Do all sketches on your paper, not on the printed figures.*

EXERCISE 7.8 [D:20] You (a finite element guru) pass away and come back to the next life as an intelligent but hungry bird. Looking around, you notice a succulent big worm taking a peek at the weather. You grab one end and pull for dinner; see Figure E7.6.

After a long struggle, however, the worm wins. While hungrily looking for a smaller one your thoughts wonder to FEM and how the worm extraction process might be modeled so you can pull it out more efficiently. Then you wake up to face this homework question. Try your hand at the following “worm modeling” points.

- The worm is simply modeled as a string of one-dimensional (bar) elements. The “worm axial force” is of course constant from the beak B to ground level G , then decreases rapidly because of soil friction (which varies roughly as plotted in the figure above) and drops to nearly zero over DE . Sketch how a good “worm-element mesh” should look like to capture the axial force well.
- On the above model, how would you represent boundary conditions, applied forces and friction forces?
- Next you want a more refined analysis of the worm that distinguishes skin and insides. What type of finite element model would be appropriate?
- (Advanced) Finally, point out what need to be added to the model of (c) to include the soil as an elastic medium.

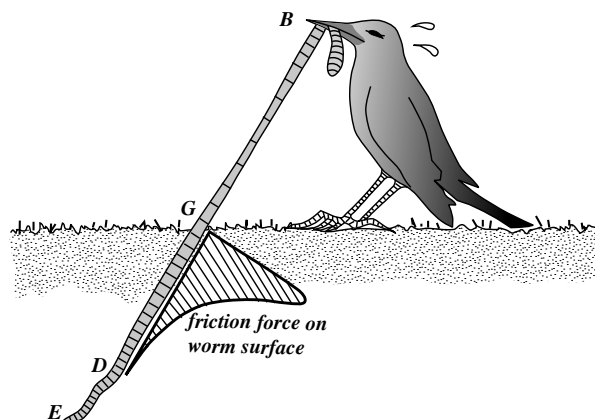


FIGURE E7.6. The hungry bird.

Briefly explain your decisions. Don't write equations.

EXERCISE 7.9 [D:15, 5 each] Prove from kinematics:

- (a) Two symmetry lines in 2D cannot cross at a finite point, unless that point is fixed.
- (b) Two antisymmetry lines in 2D cannot cross at a finite point, unless that point is fixed.
- (c) A symmetry line and an antisymmetry line must cross at right angles, unless the cross point is fixed.

Note: proofs of (a,b,c) are very similar; just draw vectors at alleged intersections.

EXERCISE 7.10 [A/D:15] A 2D body has $n > 1$ symmetry lines passing through a point C and spanning an angle π/n from each other. This is called *sectorial symmetry* if $n \geq 3$. Draw a picture for $n = 5$, say for a car wheel. Explain why point C is fixed.

EXERCISE 7.11 [A/D:25, 5 each] A body is in 3D space. The analogs of symmetry and antisymmetry lines are symmetry and antisymmetry planes, respectively. The former are also called mirror planes.

- (a) State the kinematic properties of symmetry and antisymmetric planes, and how they can be identified.
- (b) Two symmetry planes intersect. State the kinematic properties of the intersection line.
- (c) A symmetry plane and an antisymmetry plane intersect. State the kinematic properties of the intersection line. Can the angle between the planes be arbitrary?
- (d) Can two antisymmetry planes intersect?
- (e) Three symmetry planes intersect. State the kinematic properties of the intersection point.

EXERCISE 7.12 [A:25] A 2D problem is called *periodic* in the x direction if all fields, in particular displacements, repeat upon moving over a distance $a > 0$: $u_x(x + a, y) = u_x(x, y)$ and $u_y(x + a, y) = u_y(x, y)$. Can this situation be treated by symmetry and/or antisymmetry lines?

EXERCISE 7.13 [A:25] Extend the previous exercise to *antiperiodicity*, in which $u_x(x + a, y) = u_x(x, y)$ and $u_y(x + a, y) = -u_y(x, y)$.

EXERCISE 7.14 [A:20] Prove that EbE and energy consistent lumping agree if the element shape functions are piecewise linear.

EXERCISE 7.15 [A:40] If the world were spatially n -dimensional (meaning it has elliptic metric), how many independent rigid body modes would a body have? (Prove by induction)

8

MultiFreedom Constraints I

TABLE OF CONTENTS

	Page
§8.1. Classification of Constraint Conditions	8–3
§8.1.1. MultiFreedom Constraints	8–3
§8.1.2. Methods for Imposing Multifreedom Constraints	8–4
§8.1.3. *MFC Matrix Forms	8–5
§8.2. The Example Structure	8–6
§8.3. The Master-Slave Method	8–7
§8.3.1. A One-Constraint Example	8–7
§8.3.2. Multiple Homogeneous MFCs	8–8
§8.3.3. Nonhomogeneous MFCs	8–9
§8.3.4. *The General Case	8–10
§8.3.5. *Retaining the Original Freedoms	8–10
§8.4. Model Reduction by Kinematic Constraints	8–11
§8.5. Assessment of the Master-Slave Method	8–13
§8. Notes and Bibliography	8–14
§8. References	8–14
§8. Exercises	8–15

§8.1. Classification of Constraint Conditions

In previous Chapters we have considered structural support conditions that are mathematically expressible as constraints on individual degrees of freedom:

$$\boxed{\text{nodal displacement component} = \text{prescribed value.}} \quad (8.1)$$

These are called *single-freedom constraints*. Chapter 3 explains how to incorporate constraints of this form into the master stiffness equations, using hand- or computer-oriented techniques. The displacement boundary conditions studied in Chapter 7, which include modeling of symmetry and antisymmetry, lead to constraints of this form.

For example:

$$u_{x4} = 0, \quad u_{y9} = 0.6. \quad (8.2)$$

These are two single-freedom constraints. The first one is homogeneous while the second one is non-homogeneous. These attributes are defined below.

§8.1.1. MultiFreedom Constraints

The next step up in complexity involves *multifreedom equality constraints*, or *multifreedom constraints* for short, the last name being acronymed to MFC. These are functional equations that connect *two or more* displacement components:

$$\boxed{F(\text{nodal displacement components}) = \text{prescribed value,}} \quad (8.3)$$

where function F vanishes if all its nodal displacement arguments do. Equation (8.3), in which all displacement components are in the left-hand side, is called the *canonical form* of the constraint.

An MFC of this form is called *multipoint* or *multinode* if it involves displacement components at different nodes. The constraint is called *linear* if all displacement components appear linearly on the left-hand-side, and *nonlinear* otherwise.

The constraint is called *homogeneous* if, upon transferring all terms that depend on displacement components to the left-hand side, the right-hand side — the “prescribed value” in (8.3) — is zero. It is called *non-homogeneous* otherwise.

In this and next Chapter only linear constraints will be studied. Furthermore more attention is devoted to the homogeneous case, because it arises more frequently in practice.

Remark 8.1. The most general constraint class is that of inequality constraints, such as $u_{y5} - 2u_{x2} \geq 0.5$. These constraints are relatively infrequent in linear structural analysis, except in problems that involve contact conditions. They are of paramount importance, however, in other fields such as optimization and control.

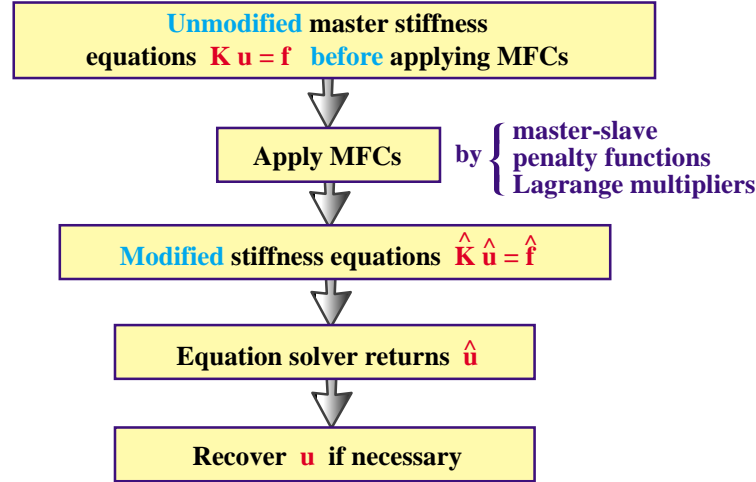


FIGURE 8.1 Schematics of MFC application.

Example 8.1. Here are three instances of MFCs:

$$u_{x2} = \frac{1}{2}u_{y2}, \quad u_{x2} - 2u_{x4} + u_{x6} = \frac{1}{4}, \quad (x_5 + u_{x5} - x_3 - u_{x3})^2 + (y_5 + u_{y5} - y_3 - u_{y3})^2 = 0. \quad (8.4)$$

The first one is linear and homogeneous. It is not a multipoint constraint because it involves the displacement components of one node: 2.

The second one is multipoint because it involves three nodes: 2, 4 and 6. It is linear and non-homogeneous.

The last one is multipoint, nonlinear and homogeneous. Geometrically it expresses that the distance between nodes 3 and 5 in two-dimensional motions on the $\{x, y\}$ plane remains constant. This kind of constraint appears in geometrically nonlinear analysis of structures, which is a topic beyond the scope of this book.

§8.1.2. Methods for Imposing Multifreedom Constraints

Accounting for multifreedom constraints is done — at least conceptually — by changing the assembled master stiffness equations to produce a *modified* system of equations:

$$\mathbf{K} \mathbf{u} = \mathbf{f} \xrightarrow{\text{MFC}} \hat{\mathbf{K}} \hat{\mathbf{u}} = \hat{\mathbf{f}}. \quad (8.5)$$

The modification process (8.5) is also called *constraint application* or *constraint imposition*. The modified system is that submitted to the equation solver, which returns $\hat{\mathbf{u}}$.

The procedure is flowcharted in Figure 8.1. The sequence of operations sketched therein applies to all methods outlined below.

Three methods for treating MFCs are discussed in this and the next Chapter:

1. *Master-Slave Elimination.* The degrees of freedom involved in each MFC are separated into master and slave freedoms. The slave freedoms are then explicitly eliminated. The modified equations do not contain the slave freedoms.
2. *Penalty Augmentation.* Also called the *penalty function method*. Each MFC is viewed as the presence of a fictitious elastic structural element called *penalty element* that enforces it approximately. This element is parametrized by a numerical *weight*. The exact constraint

is recovered if the weight goes to infinity. The MFCs are imposed by augmenting the finite element model with the penalty elements.

3. *Lagrange Multiplier Adjunction.* For each MFC an additional unknown is adjoined to the master stiffness equations. Physically this set of unknowns represent *constraint forces* that would enforce the constraints exactly should they be applied to the unconstrained system.

For each method the exposition tries to give first the basic flavor by working out the same example for each method. The general technique is subsequently presented in matrix form for completeness but is considered an advanced topic.

Conceptually, imposing MFCs is not different from the procedure discussed in Chapter 3 for single-freedom constraints. The master stiffness equations are assembled ignoring all constraints. Then the MFCs are imposed by appropriate modification of those equations. There are, however, two important practical differences:

1. The modification process is not unique because there are alternative constraint imposition methods, namely those listed above. These methods offer tradeoffs in generality, programming implementation complexity, computational effort, numerical accuracy and stability.
2. In the implementation of some of these methods — notably penalty augmentation — constraint imposition and assembly are carried out simultaneously. In that case the framework “first assemble, then modify,” is not strictly respected in the actual implementation.

Remark 8.2. The three methods are also applicable, as can be expected, to the simpler case of a single-freedom constraint such as (8.2). For most situations, however, the generality afforded by the penalty function and Lagrange multiplier methods are not warranted. The hand-oriented reduction process discussed in Chapters 3 and 4 is in fact a special case of the master-slave elimination method in which “there is no master.”

Remark 8.3. Often both multifreedom and single-freedom constraints are prescribed. The modification process then involves two stages: apply multifreedom constraints and apply single freedom constraints. The order in which these are carried out is implementation dependent. Most implementations do the MFCs first, either after the assembly is completed or during the assembly process. The reason is practical: single-freedom constraints are often automatically taken care of by the equation solver itself.

§8.1.3. *MFC Matrix Forms

Matrix forms of linear MFCs are often convenient for compact notation. An individual constraint such as the second one in (8.4) may be written

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x4} \\ u_{x6} \end{bmatrix} = 0.25. \quad (8.6)$$

In direct matrix notation:

$$\bar{\mathbf{a}}_i \bar{\mathbf{u}}_i = g_i, \quad (\text{no sum on } i) \quad (8.7)$$

in which index i ($i = 1, 2, \dots$) identifies the constraint, $\bar{\mathbf{a}}_i$ is a row vector, $\bar{\mathbf{u}}_i$ collects the set of degrees of freedom that participate in the constraint, and g_i is the right hand side scalar (0.25 in the foregoing example). The bars over \mathbf{a} and \mathbf{u} distinguishes (8.7) from the expanded form (8.9) discussed below.

For method description and general proofs it is often convenient to expand matrix forms so that they embody *all* degrees of freedom. For example, if (8.6) is part of a two-dimensional finite element model with 12 freedoms:

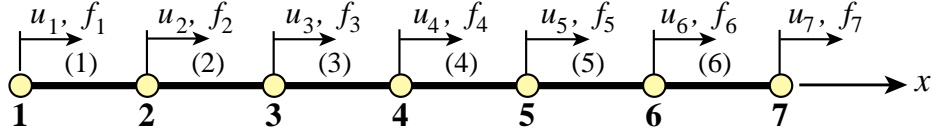


FIGURE 8.2 A one-dimensional problem discretized with six bar finite elements. The seven nodes may move only along the x direction. Subscript x is omitted from the u 's and f 's to reduce clutter.

$u_{x1}, u_{y1}, \dots, u_{y6}$, the left-hand side row vector may be expanded with nine zeros as follows

$$[0 \ 0 \ 1 \ 0 \ 0 \ 0 \ -2 \ 0 \ 0 \ 0 \ 1 \ 0] \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ \vdots \\ u_{y6} \end{bmatrix} = 0.25, \quad (8.8)$$

in which case the matrix notation

$$\mathbf{a}_i \mathbf{u} = g_i \quad (8.9)$$

is used. Finally, all multifreedom constraints expressed as (8.9) may be collected into a single matrix relation:

$$\mathbf{A} \mathbf{u} = \mathbf{g}, \quad (8.10)$$

in which rectangular matrix \mathbf{A} is formed by stacking the \mathbf{a}_i 's as rows and column vector \mathbf{g} is formed by stacking the g_i 's as entries. If there are 12 degrees of freedom in \mathbf{u} and 5 multifreedom constraints then \mathbf{A} will be 5×12 .

§8.2. The Example Structure

The one-dimensional finite element discretization shown in Figure 8.2 will be used throughout Chapters 8 and 9 to illustrate the three MFC application methods. This structure consists of six bar elements connected by seven nodes that can only displace in the x direction.

Before imposing various multifreedom constraints discussed below, the master stiffness equations for this problem are assumed to be

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \quad (8.11)$$

or

$$\mathbf{K} \mathbf{u} = \mathbf{f}. \quad (8.12)$$

The nonzero stiffness coefficients K_{ij} in (8.11) depend on the bar rigidity properties. For example, if $E^e A^e / L^e = 100$ for each element $e = 1, \dots, 6$, then $K_{11} = K_{77} = 100$, $K_{22} = \dots = K_{66} = 200$, $K_{12} = K_{23} = \dots = K_{67} = -100$. However, for the purposes of the following treatment the

coefficients may be kept arbitrary. The component index x in the nodal displacements u and nodal forces f has been omitted for brevity.

Now let us specify a multifreedom constraint that states that nodes 2 and 6 must move by the same amount:

$$u_2 = u_6. \quad (8.13)$$

Passing all node displacements to the right hand side gives the canonical form:

$$\boxed{u_2 - u_6 = 0.} \quad (8.14)$$

Constraint conditions of this type are sometimes called *rigid links* because they can be mechanically interpreted as forcing node points 2 and 6 to move together as if they were tied by a rigid member.¹

We now study the imposition of constraint (8.14) on the master equations (8.11) by the methods mentioned above. In this Chapter the master-slave method is treated. The other two methods: penalty augmentation and Lagrange multiplier adjunction, are discussed in the following Chapter.

§8.3. The Master-Slave Method

To apply this method *by hand*, the MFCs are taken one at a time. For each constraint a *slave* degree of freedom is chosen. The freedoms remaining in that constraint are labeled *master*. A new set of degrees of freedom $\hat{\mathbf{u}}$ is established by removing all slave freedoms from \mathbf{u} . This new vector contains master freedoms as well as those that do not appear in the MFCs. A matrix transformation equation that relates \mathbf{u} to $\hat{\mathbf{u}}$ is generated. This equation is used to apply a congruent transformation to the master stiffness equations. This procedure yields a set of modified stiffness equations that are expressed in terms of the new freedom set $\hat{\mathbf{u}}$. Because the modified system does not contain the slave freedoms, these have been effectively eliminated.

§8.3.1. A One-Constraint Example

The mechanics of the process is best seen by going through an example. To impose (8.14) pick u_6 as slave and u_2 as master. Relate the original unknowns u_1, \dots, u_7 to the new set in which u_6 is missing:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix}, \quad (8.15)$$

This is the required transformation relation. In compact form:

$$\boxed{\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}}. \quad (8.16)$$

¹ This physical interpretation is exploited in the penalty method described in the next Chapter. In two and three dimensions rigid link constraints are more complicated.

Replacing (8.15) into (8.12) and premultiplying by \mathbf{T}^T yields the modified system

$$\boxed{\hat{\mathbf{K}} \hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T \mathbf{K} \mathbf{T}, \quad \hat{\mathbf{f}} = \mathbf{T}^T \mathbf{f}.} \quad (8.17)$$

Carrying out the indicated matrix multiplications yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} + K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 \\ f_3 \\ f_4 \\ f_5 \\ f_7 \end{bmatrix}, \quad (8.18)$$

Equation (8.18) is a new linear system containing 6 equations in the remaining 6 unknowns: u_1 , u_2 , u_3 , u_4 , u_5 and u_7 . Upon solving it, u_6 is recovered from the constraint (8.13).

Remark 8.4. The form of modified system (8.17) can be remembered by a simple mnemonic rule: premultiply both sides of $\mathbf{T} \hat{\mathbf{u}} = \mathbf{u}$ by $\mathbf{T}^T \mathbf{K}$, and replace $\mathbf{K} \mathbf{u}$ by \mathbf{f} on the right hand side.

Remark 8.5. For a simple freedom constraint such as $u_4 = 0$ the only possible choice of slave is of course u_4 and there is no master. The congruent transformation is then nothing more than the elimination of u_4 by striking out rows and columns from the master stiffness equations.

Remark 8.6. For a simple MFC such as $u_2 = u_6$, it does not matter which degree of freedom is chosen as master or unknown. Choosing u_2 as slave produces a system of equations in which now u_2 is missing:

$$\begin{bmatrix} K_{11} & 0 & 0 & 0 & K_{12} & 0 \\ 0 & K_{33} & K_{34} & 0 & K_{23} & 0 \\ 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ K_{12} & K_{23} & 0 & K_{56} & K_{22} + K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_3 \\ f_4 \\ f_5 \\ f_2 + f_6 \\ f_7 \end{bmatrix}. \quad (8.19)$$

Although (8.18) and (8.19) are algebraically equivalent, the latter would be processed faster if a skyline solver (Part III of course) is used for the modified equations.

§8.3.2. Multiple Homogeneous MFCs

The matrix equation (8.17) in fact holds for the general case of multiple homogeneous linear constraints. Direct establishment of the transformation equation, however, is more complicated if slave freedoms in one constraint appear as masters in another. To illustrate this point, suppose that for the example system we have three homogeneous multifreedom constraints:

$$u_2 - u_6 = 0, \quad u_1 + 4u_4 = 0, \quad 2u_3 + u_4 + u_5 = 0, \quad (8.20)$$

Picking as slave freedoms u_6 , u_4 and u_3 from the first, second and third constraint, respectively, we can solve for them as

$$u_6 = u_2, \quad u_4 = -\frac{1}{4}u_1, \quad u_3 = -\frac{1}{2}(u_4 + u_5) = \frac{1}{8}u_1 - \frac{1}{2}u_5. \quad (8.21)$$

Observe that solving for u_3 from the third constraint brings u_4 to the right-hand side. But because u_4 is also a slave freedom (it was chosen as such for the second constraint) it is replaced in favor of u_1 using $u_4 = -\frac{1}{4}u_1$. The matrix form of the transformation (8.21) is

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{8} & 0 & -\frac{1}{2} & 0 \\ -\frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \\ u_7 \end{bmatrix}, \quad (8.22)$$

The modified system is now formed through the congruent transformation (8.17). Note that the slave freedoms selected from each constraint must be distinct; for example the choice u_6, u_4, u_4 would be inadmissible as long as the constraints are independent. This rule is easy to enforce when slave freedoms are chosen by hand, but can lead to implementation and numerical difficulties when it is programmed as an automated procedure, as further discussed later.

Remark 8.7. The three MFCs (8.20) with u_6, u_4 and u_2 chosen as slaves and u_1, u_2 and u_5 chosen as masters, may be presented in the partitioned matrix form:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 0 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \end{bmatrix} \quad (8.23)$$

This may be compactly written $\mathbf{A}_s \mathbf{u}_s + \mathbf{A}_m \mathbf{u}_m = \mathbf{0}$. Solving for the slave freedoms gives $\mathbf{u}_s = -\mathbf{A}_s^{-1} \mathbf{A}_m \mathbf{u}_m$. Expanding with zeros to fill out \mathbf{u} and $\hat{\mathbf{u}}$ produces (8.22). This general matrix form is considered in §8.4.4. Note that non-singularity of \mathbf{A}_s is essential for this method to work.

§8.3.3. Nonhomogeneous MFCs

Extension to non-homogeneous constraints is immediate. In this case the transformation equation becomes non-homogeneous. For example suppose that (8.14) has a nonzero prescribed value:

$$\boxed{u_2 - u_6 = 0.2} \quad (8.24)$$

Nonzero RHS values such as 0.2 in (8.24) may be often interpreted physically as “gaps” (thus the use of the symbol \mathbf{g} in the matrix form). Chose u_6 again as slave: $u_6 = u_2 - 0.2$, and build the transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ -0.2 \\ 0 \end{bmatrix}. \quad (8.25)$$

In compact matrix notation,

$$\boxed{\mathbf{u} = \mathbf{T} \hat{\mathbf{u}} + \mathbf{g}.} \quad (8.26)$$

Here the constraint gap vector \mathbf{g} is nonzero and \mathbf{T} is the same as before. To get the modified system applying the shortcut rule of Remark 8.4, premultiply both sides of (8.26) by $\mathbf{T}^T \mathbf{K}$, replace $\mathbf{K}\mathbf{u}$ by \mathbf{f} , and pass the data to the RHS:

$$\hat{\mathbf{K}} \hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T \mathbf{K} \mathbf{T}, \quad \hat{\mathbf{f}} = \mathbf{T}^T (\mathbf{f} - \mathbf{K} \mathbf{g}). \quad (8.27)$$

Upon solving (8.27) for $\hat{\mathbf{u}}$, the complete displacement vector is recovered from (8.26). For the MFC (8.24) this technique gives the system

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} + K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 - 0.2K_{66} \\ f_3 \\ f_4 \\ f_5 - 0.2K_{56} \\ f_7 - 0.2K_{67} \end{bmatrix}. \quad (8.28)$$

See Exercise 8.2 for multiple non-homogeneous MFCs.

§8.3.4. *The General Case

For implementation in general-purpose programs the master-slave method can be described as follows. The degrees of freedoms in \mathbf{u} are classified into three types: independent or unconstrained, masters and slaves. (The unconstrained freedoms are those that do not appear in any MFC.) Label these sets as \mathbf{u}_u , \mathbf{u}_m and \mathbf{u}_s , respectively, and partition the stiffness equations accordingly:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um} & \mathbf{K}_{us} \\ \mathbf{K}_{um}^T & \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{us}^T & \mathbf{K}_{ms}^T & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \mathbf{f}_m \\ \mathbf{f}_s \end{bmatrix} \quad (8.29)$$

The MFCs may be written in matrix form as

$$\mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s \mathbf{u}_s = \mathbf{g}_A, \quad (8.30)$$

where \mathbf{A}_s is assumed square and nonsingular. If so we can solve for the slave freedoms:

$$\mathbf{u}_s = -\mathbf{A}_s^{-1} \mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s^{-1} \mathbf{g}_A \stackrel{\text{def}}{=} \mathbf{T} \mathbf{u}_m + \mathbf{g}, \quad (8.31)$$

Inserting into the partitioned stiffness equations (8.30) and symmetrizing yields

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um} + \mathbf{K}_{us} \mathbf{T} \\ \text{symm} & \mathbf{K}_{mm} + \mathbf{T}^T \mathbf{K}_{ms}^T + \mathbf{K}_{ms} \mathbf{T} + \mathbf{T}^T \mathbf{K}_{ss} \mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u - \mathbf{K}_{us} \mathbf{g} \\ \mathbf{f}_m - \mathbf{K}_{ms} \mathbf{g} \end{bmatrix} \quad (8.32)$$

It is seen that the misleading simplicity of the handworked examples is gone.

§8.3.5. *Retaining the Original Freedoms

A potential disadvantage of the master-slave method in computer work is that it requires a rearrangement of the original stiffness equations because $\hat{\mathbf{u}}$ is a subset of \mathbf{u} . The disadvantage can be annoying when sparse matrix storage schemes are used for the stiffness matrix, and becomes intolerable if secondary storage is used for that purpose.

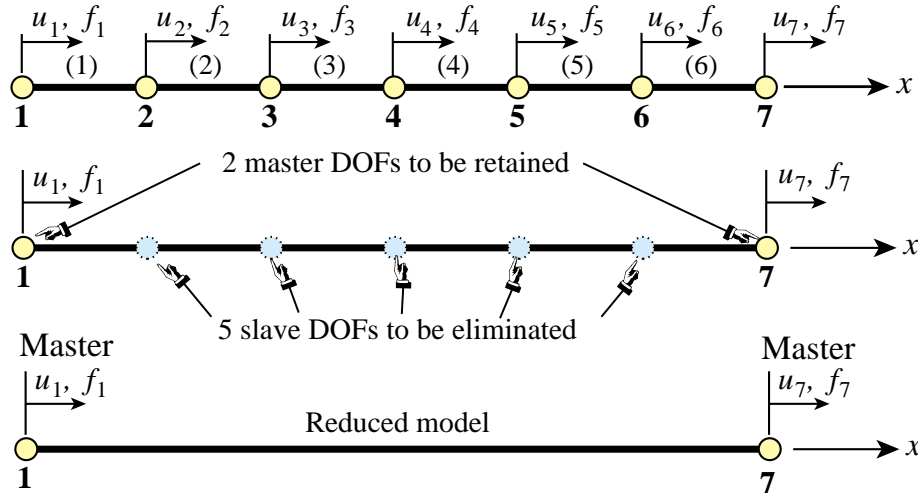


FIGURE 8.3 Model reduction of the example structure of Figure 8.2 to the end freedoms.

With a bit of trickery it is possible to maintain the original freedom ordering. Let us display it for the example problem under (8.14). Instead of (8.15), use the *square* transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix}, \quad (8.33)$$

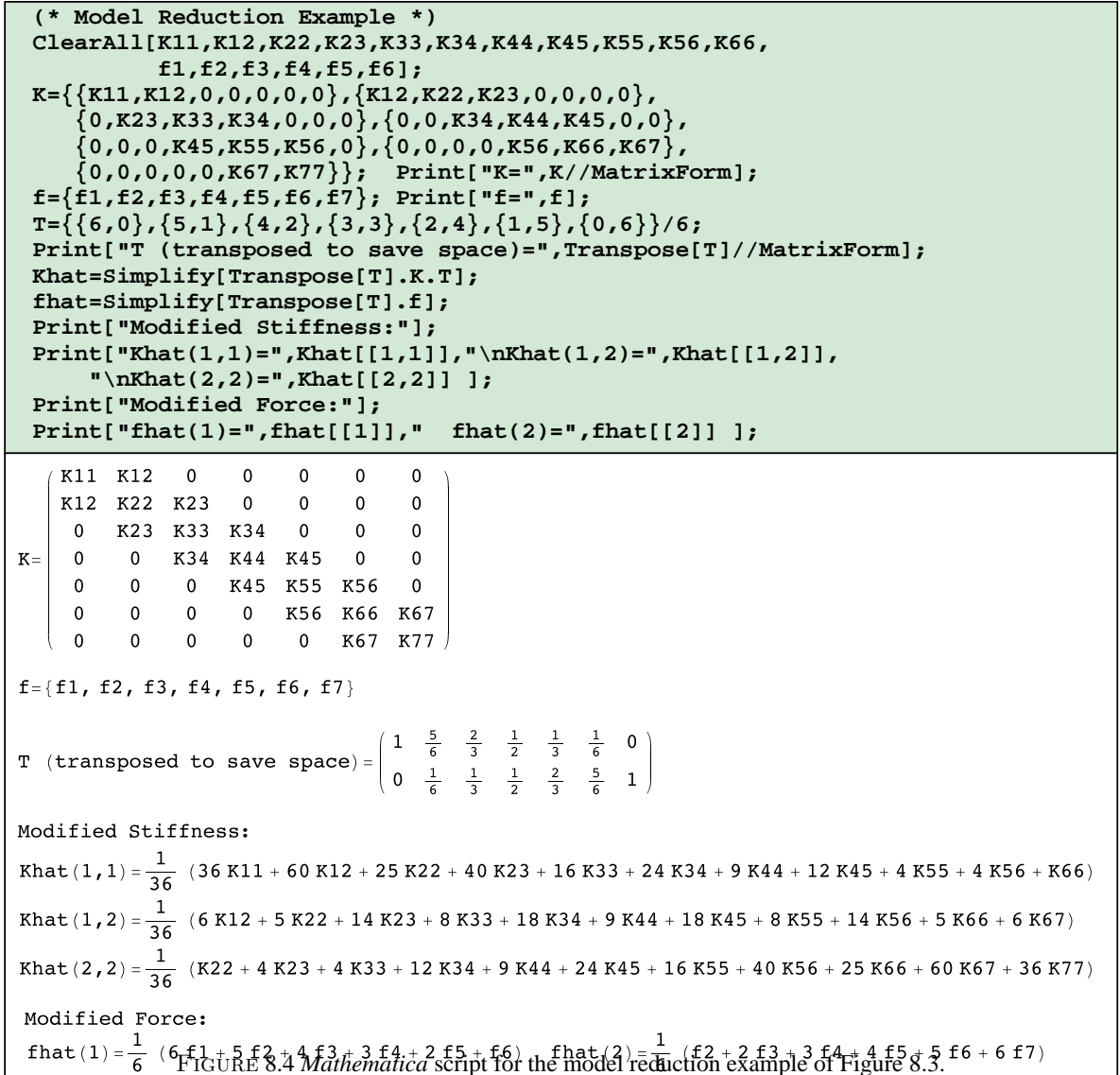
in which \tilde{u}_6 is a *placeholder* for the slave freedom u_6 . The modified equations are

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} + K_{66} & K_{23} & 0 & 0 & K_{56} & 0 & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{67} & 0 & 0 & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 \\ f_3 \\ f_4 \\ f_5 \\ 0 \\ f_7 \end{bmatrix}, \quad (8.34)$$

which are submitted to the equation solver. If the solver is not trained to skip zero rows and columns, a one should be placed in the diagonal entry for the \tilde{u}_6 (sixth) equation. The solver will return $\tilde{u}_6 = 0$, and this placeholder value is replaced by u_2 . Note several points in common with the computer-oriented placeholder technique described in §3.4 to handle single-freedom constraints.

§8.4. Model Reduction by Kinematic Constraints

The congruent transformation equations (8.17) and (8.27) have additional applications beyond the master-slave method. An important one is *model reduction by kinematic constraints*. Through this procedure the number of DOF of a static or dynamic FEM model is reduced by a significant number, typically to 1% – 10% of the original number. This is done by taking a lot of slaves and a few masters. Only the masters are left after the transformation. The reduced model is commonly

FIGURE 8.4 *Mathematica* script for the model reduction example of Figure 8.3.

used in subsequent calculations as component of a larger system, particularly during design or in parameter identification.

Example 8.2. Consider the bar assembly of Figure 8.2. Assume that the only masters are the end motions u_1 and u_7 , as illustrated in Figure 8.2, and interpolate all freedoms linearly:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5/6 & 1/6 \\ 4/6 & 2/6 \\ 3/6 & 3/6 \\ 2/6 & 4/6 \\ 1/6 & 5/6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix}, \quad \text{or} \quad \mathbf{u} = \mathbf{T} \hat{\mathbf{u}}. \quad (8.35)$$

The reduced-order-model (ROM) equations are $\hat{\mathbf{K}}\hat{\mathbf{u}} = \mathbf{T}^T \mathbf{K} \mathbf{T} \hat{\mathbf{u}} = \mathbf{T}^T \mathbf{f} = \hat{\mathbf{f}}$, or in detail

$$\begin{bmatrix} \hat{K}_{11} & \hat{K}_{17} \\ \hat{K}_{17} & \hat{K}_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix} = \begin{bmatrix} \hat{f}_1 \\ \hat{f}_7 \end{bmatrix}, \quad (8.36)$$

in which

$$\begin{aligned} \hat{K}_{11} &= \frac{1}{36}(36K_{11}+60K_{12}+25K_{22}+40K_{23}+16K_{33}+24K_{34}+9K_{44}+12K_{45}+4K_{55}+4K_{56}+K_{66}), \\ \hat{K}_{17} &= \frac{1}{36}(6K_{12}+5K_{22}+14K_{23}+8K_{33}+18K_{34}+9K_{44}+18K_{45}+8K_{55}+14K_{56}+5K_{66}+6K_{67}), \\ \hat{K}_{77} &= \frac{1}{36}(K_{22}+4K_{23}+4K_{33}+12K_{34}+9K_{44}+24K_{45}+16K_{55}+40K_{56}+25K_{66}+60K_{67}+36K_{77}), \\ \hat{f}_1 &= \frac{1}{6}(6f_1+5f_2+4f_3+3f_4+2f_5+f_6), \quad \hat{f}_7 = \frac{1}{6}(f_2+2f_3+3f_4+4f_5+5f_6+6f_7). \end{aligned} \quad (8.37)$$

This reduces the order of the FEM model from 7 to 2. A *Mathematica* script to do the reduction is shown in Figure 8.4. The key feature is that the masters are picked *a priori*, as the freedoms to be retained in the model for further use.

Remark 8.8. Model reduction can also be done by the static condensation method explained in Chapter 10. As its name indicates, condensation is restricted to static analysis. On the other hand, for such problems it is exact whereas model reduction by kinematic constraints generally introduces approximations.

§8.5. Assessment of the Master-Slave Method

What are the good and bad points of this constraint application method? It enjoys the advantage of being exact (except for inevitable solution errors) and of reducing the number of unknowns. The concept is also easy to explain and learn. The main implementation drawback is the complexity of the general case as can be seen by studying (8.29) through (8.32). The complexity is due to three factors:

1. The equations may have to be rearranged because of the disappearance of the slave freedoms. This drawback can be alleviated, however, through the placeholder trick outlined in §8.3.5.
2. An auxiliary linear system, namely (8.31), has to be assembled and solved to produce the transformation matrix \mathbf{T} and vector \mathbf{g} .
3. The transformation process may generate many additional matrix terms. If a sparse matrix storage scheme is used for \mathbf{K} , the logic for allocating memory and storing these entries can be difficult and expensive.

The level of complexity depends on the generality allowed as well as on programming decisions. If \mathbf{K} is stored as full matrix and slave freedom coupling in the MFCs is disallowed the logic is simple.² On the other hand, if arbitrary couplings are permitted and \mathbf{K} is placed in secondary (disk) storage according to some sparse scheme, the complexity can become overwhelming.

Another, more subtle, drawback of this method is that it requires decisions as to which degrees of freedom are to be treated as slaves. This can lead to implementation and numerical stability problems. Although for disjointed constraints the process can be programmed in reliable form, in more general cases of coupled constraint equations it can lead to incorrect decisions. For example, suppose that in the example problem you have the following two MFCs:

$$\frac{1}{6}u_2 + \frac{1}{2}u_4 = u_6, \quad u_3 + 6u_6 = u_7. \quad (8.38)$$

² This is the case in model reduction, since each slave freedom appears in one and only one MFC.

For numerical stability reasons it is usually better to pick as slaves the freedoms with larger coefficients. If this is done, the program would select u_6 as slave freedoms from both constraints. This leads to a contradiction because having two constraints we must eliminate two slave degrees of freedom, not just one. The resulting modified system would in fact be inconsistent. Although this defect can be easily fixed by the program logic in this case, one can imagine the complexity burden if faced with hundreds or thousands of MFCs.

Serious numerical problems can arise if the MFCs are not independent. For example:

$$\frac{1}{6}u_2 = u_6, \quad \frac{1}{5}u_3 + 6u_6 = u_7, \quad u_2 + u_3 - u_7 = 0. \quad (8.39)$$

The last constraint is an exact linear combination of the first two. If the program blindly chooses u_2 , u_3 and u_7 as slaves, the modified system is incorrect because we eliminate three equations when in fact there are only two independent constraints. Exact linear dependence, as in (8.39), can be recognized by a rank analysis of the \mathbf{A}_s matrix defined in (8.30). In floating-point arithmetic, however, such detection may fail because that kind of computation is inexact by nature.³

The complexity of slave selection is in fact equivalent to that of automatically selecting kinematic redundancies in the Force Method of structural analysis. It has led implementors of programs that use this method to require masters and slaves be prescribed in the input data, thus transferring the burden to users.

The method is not generally extendible to nonlinear constraints without case by case programming.

In conclusion, the master-slave method is useful when a few simple linear constraints are imposed by hand. As a general purpose technique for finite element analysis it suffers from complexity and lack of robustness. It is worth learning, however, because of the great importance of congruent transformations in *model reduction* for static and dynamic problems.

Notes and Bibliography

Multifreedom constraints are treated in several of the FEM books recommended in §1.7.5, notably Zienkiewicz and Taylor [741]. The master-slave method was incorporated to treat MFCs as part of the DSM developed at Boeing during the 1950s. It is first summarily described in the DSM-overview by Turner, Martin and Weikel [683, p. 212]. The implementation differs, however, from the one described here because the relation of FEM to energy methods was not clear at the time.

The master-slave method became popular through its adoption by the general-purpose NASTRAN code developed in the late 1960s [19] and early assessments of its potential [668]. The implementation unfortunately relied on user inputs to identify slave DOFs. Through this serious blunder the method gained a reputation for unreliability that persists to the present day.

The important application of master-slave to model reduction, which by itself justifies teaching the method, is rarely mentioned in FEM textbooks.

References

Referenced items have been moved to Appendix R.

³ The safest technique to identify dependencies is to do a singular-value decomposition (SVD) of \mathbf{A}_s . This can be, however, prohibitively expensive if one is dealing with hundreds or thousands of constraints.

Homework Exercises for Chapter 8

MultiFreedom Constraints I

EXERCISE 8.1 [C+N:20] The example structure of Figure 8.1 has $E^e A^e / L^e = 100$ for each element $e = 1, \dots, 6$. Consequently $K_{11} = K_{77} = 100$, $K_{22} = \dots = K_{66} = 200$, $K_{12} = K_{23} = \dots = K_{67} = -100$. The applied node forces are taken to be $f_1 = 1$, $f_2 = 2$, $f_3 = 3$, $f_4 = 4$, $f_5 = 5$, $f_6 = 6$ and $f_7 = 7$, which are easy to remember. The structure is subjected to one support condition: $u_1 = 0$ (a fixed left end), and to one MFC: $u_2 - u_6 = 1/5$.

Solve this problem using the master-slave method to process the MFC, taking u_6 as slave. Upon forming the modified system (8.27) apply the left-end support $u_1 = 0$ using the placeholder method of §3.4. Solve the equations and verify that the displacement solution and the recovered node forces including reactions are

$$\mathbf{u} = [0 \quad 0.270 \quad 0.275 \quad 0.250 \quad 0.185 \quad 0.070 \quad 0.140]^T$$

$$\mathbf{Ku} = [-27 \quad 26.5 \quad 3 \quad 4 \quad 5 \quad -18.5 \quad 7]^T \quad (\text{E8.1})$$

Use *Mathematica* or *Matlab* to do the algebra is recommended. For example, the *Mathematica* script of Figure E8.1 solves this Exercise.

```
(* Exercise 8.1 - Master-Slave Method *)
(* MFC: u2-u6 = 1/5 - slave: u6 *)
MasterStiffnessOfSixElementBar[kbar_]:=Module[
  {K=Table[0,{7},{7}], K[[1,1]]=K[[7,7]]=kbar;
  For [i=2,i<=6,i++,K[[i,i]]=2*kbar];
  For [i=1,i<=6,i++,K[[i,i+1]]=K[[i+1,i]]=-kbar];
  Return[K]];
FixLeftEndOfSixElementBar[Khat_,fhat_]:=Module[
  {Kmod=Khat,fmod=fhat}, fmod[[1]]=0; Kmod[[1,1]]=1;
  Kmod[[1,2]]=Kmod[[2,1]]=0; Return[{Kmod,fmod}]];

K=MasterStiffnessOfSixElementBar[100];
Print["Stiffness K=",K//MatrixForm];
f={1,2,3,4,5,6,7}; Print["Applied forces=",f];
T={{1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,1,0,0,0},
  {0,0,0,1,0,0},{0,0,0,0,1,0},{0,1,0,0,0,0},
  {0,0,0,0,0,1}};
Print["Transformation matrix T=",T//MatrixForm];
g={0,0,0,0,0,-1/5,0};
Print["Constraint gap vector g=",g];
Khat=Simplify[Transpose[T].K.T]; fhat=Simplify[Transpose[T].(f-K.g)];
{Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat]; (* fix left end *)
Print["Modified Stiffness upon fixing node 1:",Kmod//MatrixForm];
Print["Modified RHS upon fixing node 1:",fmod];
umod=LinearSolve[Kmod,fmod];
Print["Computed umod (lacks slave u6)=",umod];
u=T.umod+g; Print["Complete solution u=",u];
Print["Numerical u=",N[u]];
fu=K.u; Print["Recovered forces K.u with reactions=",fu];
Print["Numerical K.u=",N[fu]];
```

FIGURE E8.1 Script for solving Exercise 8.1.

EXERCISE 8.2 [C+N:25] As in the previous Exercise but applying the following three MFCs, two of which are non-homogeneous:

$$u_2 - u_6 = 1/5, \quad u_3 + 2u_4 = -2/3, \quad 2u_3 - u_4 + u_5 = 0. \quad (\text{E8.2})$$

Hint. Chose u_4 , u_5 and u_6 as slaves. Much of the script shown for Exercise 8.1 can be reused. The main changes are in the formation of \mathbf{T} and \mathbf{g} . If you are a *Mathematica* wizard (or willing to be one) those can be automatically formed by the statements listed in Figure E8.2.

```
sol=Simplify[Solve[{u2-u6==1/5, u3+2*u4== -2/3, 2*u3-u4+u5==0},{u4,u5,u6}]]];
ums={u1,u2,u3,u4,u5,u6,u7}/.sol[[1]]; um={u1,u2,u3,u7};
T=Table[Coefficient[ums[[i]],um[[j]]],{i,1,7},{j,1,4}];
g=ums/.{u1->0,u2->0,u3->0,u4->0,u5->0,u6->0,u7->0};
Print["Transformation matrix T=",T//MatrixForm];
Print["Gap vector g=",g];
```

FIGURE E8.2 Script for partially solving Exercise 8.2.

If you do this, explain what it does and why it works. Otherwise form and enter \mathbf{T} and \mathbf{g} by hand. The numerical results (shown to 5 places) should be

$$\begin{aligned} \mathbf{u} &= [0. \quad 0.043072 \quad -0.075033 \quad -0.29582 \quad -0.14575 \quad -0.15693 \quad -0.086928]^T, \\ \mathbf{Ku} &= [-4.3072 \quad 16.118 \quad 10.268 \quad -37.085 \quad 16.124 \quad -8.1176 \quad 7.]^T. \end{aligned} \quad (\text{E8.3})$$

EXERCISE 8.3 [A:25] Can the MFCs be pre-processed to make sure that no slave freedom in a MFC appears as master in another?

EXERCISE 8.4 [A:25] In the general case discussed in §8.4.4, under which condition is the matrix \mathbf{A}_s of (8.30) diagonal and thus trivially invertible?

EXERCISE 8.5 [A:25] Work out the general technique by which the unknowns need not be rearranged, that is, \mathbf{u} and $\hat{\mathbf{u}}$ are the same. Use “placeholders” for the slave freedoms. (Hint: use ideas of §3.4).

EXERCISE 8.6 [A/C:35] Is it possible to establish a slave selection strategy that makes \mathbf{A}_s diagonal or triangular? (This requires knowledge of matrix techniques such as pivoting.)

EXERCISE 8.7 [A/C:40] Work out a strategy that produces a well conditioned \mathbf{A}_s by selecting new slaves as linear combinations of finite element freedoms if necessary. (Requires background in numerical analysis and advanced programming experience in matrix algebra).

9

MultiFreedom Constraints II

TABLE OF CONTENTS

	Page
§9.1. Introduction	9–3
§9.2. The Penalty Method	9–3
§9.2.1. Physical Interpretation of Penalty Method	9–3
§9.2.2. Choosing the Penalty Weight	9–3
§9.2.3. The Square Root Rule	9–4
§9.2.4. Penalty Elements for General MFCs	9–5
§9.2.5. *The Theory Behind the Penalty Method	9–6
§9.2.6. Assessment of the Penalty Method	9–7
§9.3. Lagrange Multiplier Adjunction	9–8
§9.3.1. Physical Interpretation	9–8
§9.3.2. Lagrange Multipliers for General MFCs	9–9
§9.3.3. *The Theory Behind Lagrange Multipliers	9–10
§9.3.4. Assessment of the Lagrange Multiplier Method	9–10
§9.4. *The Augmented Lagrangian Method	9–10
§9.5. Summary	9–11
§9. Notes and Bibliography	9–11
§9. References	9–12
§9. Exercises	9–13

§9.1. Introduction

In this Chapter we continue the discussion of methods to treat multifreedom constraints (MFCs). The master-slave method described previously was easy to explain, but exhibits serious shortcomings for treating arbitrary constraints (although the method has important applications to model reduction).

We now pass to the study of two other methods: penalty augmentation and Lagrange multiplier adjunction. Both techniques are better suited to general implementations of the Finite Element Method, whether linear or nonlinear.

§9.2. The Penalty Method

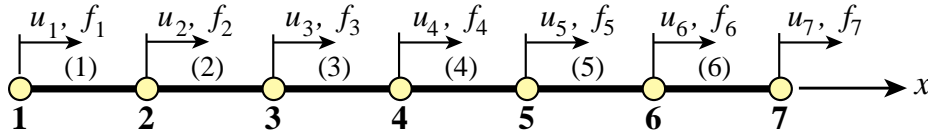


FIGURE 9.1. The example structure of Chapter 8, repeated for convenience.

§9.2.1. Physical Interpretation of Penalty Method

The penalty method will be first presented using a physical argument, leaving the mathematical formulation to a subsequent section. Consider again the example structure of Chapter 8, which is reproduced in Figure 9.1 for convenience. To impose $u_2 = u_6$ imagine that nodes 2 and 6 are connected with a “fat” bar of axial stiffness w , labeled with element number 7, as shown in Figure 9.2. This bar is called a *penalty element* and w is its *penalty weight*.

Such an element, albeit fictitious, can be treated exactly like another bar element insofar as continuing the assembly of the master stiffness equations. The penalty element stiffness equations, $\mathbf{K}^{(7)}\mathbf{u}^{(7)} = \mathbf{f}^{(7)}$, are¹

$$w \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_2 \\ u_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad (9.1)$$

Because there is one freedom per node, the two local element freedoms map into global freedoms 2 and 6, respectively. Using the assembly rules of Chapter 3 we obtain the following modified master stiffness equations: $\hat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}$, which shown in detail are

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} + w & K_{23} & 0 & 0 & -w & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & -w & 0 & 0 & K_{56} & K_{66} + w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \quad (9.2)$$

This system can now be submitted to the equation solver. Note that $\hat{\mathbf{u}} \equiv \mathbf{u}$, and only \mathbf{K} has changed.

¹ The general method to construct these equations is described in §9.1.4.

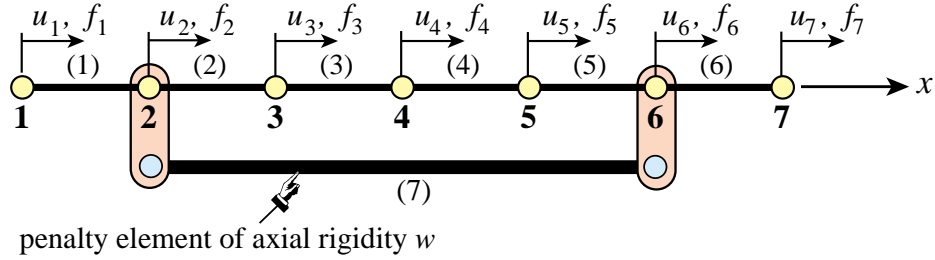


FIGURE 9.2. Adjunction of a fictitious penalty bar of axial stiffness w , identified as element 7, to enforce the MFC $u_2 = u_6$.

§9.2.2. Choosing the Penalty Weight

What happens when (9.2) is solved numerically? If a *finite* weight w is chosen the constraint $u_2 = u_6$ is approximately satisfied in the sense that one gets $u_2 - u_6 = e_g$, where $e_g \neq 0$. The “gap error” e_g is called the *constraint violation*. The magnitude $|e_g|$ of this violation depends on the weight: the larger w , the smaller the violation. More precisely, it can be shown that $|e_g|$ becomes proportional to $1/w$ as w gets to be sufficiently large (see Exercises). For example, raising w from, say, 10^6 to 10^7 can be expected to cut the constraint violation by roughly 10 if the physical stiffnesses are small compared to w .

Therefore it seems as if the proper strategy should be: try to make w as large as possible while respecting computer overflow limits. However, this is misleading. As the penalty weight w tends to ∞ the modified stiffness matrix in (9.2) becomes more and more *ill-conditioned with respect to inversion*.

To make this point clear, suppose for definiteness that the rigidities $E^e A^e / L^e$ of the actual bars $e = 1, \dots, 6$ are unity, that $w \gg 1$, and that the computer solving the stiffness equations has a floating-point precision of 16 decimal places. Numerical analysts characterize such precision by saying that $\epsilon_f = O(10^{-16})$, where $|\epsilon_f|$ is the smallest power of 10 that perceptibly adds to 1 in floating-point arithmetic.² The modified stiffness matrix of (9.2) becomes

$$\hat{\mathbf{K}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2+w & -1 & 0 & 0 & -w & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & -w & 0 & 0 & -1 & 2+w & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \quad (9.3)$$

As $w \rightarrow \infty$ rows 2 and 6, as well as columns 2 and 6, tend to become linearly dependent; in fact the negative of each other. But *linear dependency means singularity*. Therefore $\hat{\mathbf{K}}$ approaches singularity as $w \rightarrow \infty$. In fact, if w exceeds $1/\epsilon_f = 10^{16}$ the computer will not be able to distinguish $\hat{\mathbf{K}}$ from an exactly singular matrix. If $w \ll 10^{16}$ but $w \gg 1$, the effect will be seen in increasing solution errors affecting the computed displacements $\hat{\mathbf{u}}$ returned by the equation solver. These errors, however, tend to be more of a random nature than the constraint violation error.

² Such definitions are more rigorously done by working with binary numbers and base-2 arithmetic but for the present discussion the use of decimal powers is sufficient.

§9.2.3. The Square Root Rule

Plainly we have two effects at odds with each other. Making w larger reduces the constraint violation error but increases the solution error. The best w is that which makes both errors roughly equal in absolute value. This tradeoff value is difficult to find aside of systematically running numerical experiments. In practice the heuristic *square root rule* is often followed.

This rule can be stated as follows. Suppose that the largest stiffness coefficient, before adding penalty elements, is of the order of 10^k and that the working machine precision is p digits.³ Then choose penalty weights to be of order $10^{k+p/2}$ with the proviso that such a choice would not cause arithmetic overflow.⁴

For the above example in which $k \approx 0$ and $p \approx 16$, the optimal w given by this rule would be $w \approx 10^8$. This w would yield a constraint violation and a solution error of order 10^{-8} . Note that there is no simple way to do better than this accuracy aside from using extended (e.g., quad) floating-point precision. This is not easy to do when using standard low-level programming languages.

The name “square root” arises because the recommended w is in fact $10^k \sqrt{10^p}$. It is seen that picking the weight by this rule requires knowledge of both stiffness magnitudes and floating-point hardware properties of the computer used, as well as the precision selected by the program.

§9.2.4. Penalty Elements for General MFCs

For the constraint $u_2 = u_6$ the physical interpretation of the penalty element is clear. Nodal points 2 and 6 must move in lockstep long x , which can be approximately enforced by the heavy bar device shown in Figure 9.2. But how about $3u_3 + u_5 - 4u_6 = 1$? Or just $u_2 = -u_6$?

The treatment of more general constraints is linked to the theory of *Courant penalty functions*, which in turn is a topic in variational calculus. Because the necessary theory given in §9.1.5 is viewed as an advanced topic, the procedure used for constructing a penalty element is stated here as a recipe. Consider the homogeneous constraint

$$3u_3 + u_5 - 4u_6 = 0. \quad (9.4)$$

Rewrite this equation in matrix form

$$\begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 0, \quad (9.5)$$

and premultiply both sides by the transpose of the coefficient matrix:

$$\begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix} \begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \bar{\mathbf{K}}^e \mathbf{u}^e = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \quad (9.6)$$

³ Such order-of-magnitude estimates can be readily found by scanning the diagonal of \mathbf{K} because the largest stiffness coefficient of the actual structure is usually a diagonal entry.

⁴ If overflows occurs, the master stiffness should be scaled throughout or a better choice of physical units made.

Here $\bar{\mathbf{K}}^e$ is the *unscaled* stiffness matrix of the penalty element. This is now multiplied by the penalty weight w and assembled into the master stiffness matrix following the usual rules. For the example problem, augmenting (9.2) with the w -scaled penalty element (9.6) yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} + 9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55} + w & K_{56} - 4w & 0 \\ 0 & 0 & -12w & 0 & K_{56} - 4w & K_{66} + 16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \quad (9.7)$$

If the constraint is nonhomogeneous the force vector is also modified. To illustrate this effect, consider the MFC: $3u_3 + u_5 - 4u_6 = 1$. Rewrite in matrix form as

$$\begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 1. \quad (9.8)$$

Premultiply both sides by the transpose of the coefficient matrix:

$$\begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix}. \quad (9.9)$$

Scaling by w and assembling yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} + 9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55} + w & K_{56} - 4w & 0 \\ 0 & 0 & -12w & 0 & K_{56} - 4w & K_{66} + 16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 + 3w \\ f_4 \\ f_5 + w \\ f_6 - 4w \\ f_7 \end{bmatrix}. \quad (9.10)$$

§9.2.5. *The Theory Behind the Penalty Method

The rule comes from the following mathematical theory. Suppose we have a set of m linear MFCs. Using the matrix notation introduced in §8.1.3, these will be stated as

$$\mathbf{a}_p \mathbf{u} = b_p, \quad p = 1, \dots, m \quad (9.11)$$

where \mathbf{u} contains all degrees of freedom and each \mathbf{a}_p is a row vector with same length as \mathbf{u} . To incorporate the MFCs into the FEM model one selects a weight $w_p > 0$ for each constraints and constructs the so-called Courant quadratic penalty function or “penalty energy”

$$P = \sum_{p=1}^m P_p, \quad \text{with} \quad P_p = \mathbf{u}^T \left(\frac{1}{2} \mathbf{a}_p^T \mathbf{a}_p \mathbf{u} - w_p \mathbf{a}_p^T b_p \right) = \frac{1}{2} \mathbf{u}^T \mathbf{K}^{(p)} \mathbf{u} - \mathbf{u}^T \mathbf{f}^{(p)}, \quad (9.12)$$

where we have called $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and $\mathbf{f}^{(p)} = w_p \mathbf{a}_p^T \mathbf{b}_i$. P is added to the potential energy function $\Pi = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{f}$ to form the *augmented potential energy* $\Pi_a = \Pi + P$. Minimization of Π_a with respect to \mathbf{u} yields

$$(\mathbf{K} \mathbf{u} + \sum_{p=1}^m \mathbf{K}^{(p)}) \mathbf{u} = \mathbf{f} + \sum_{p=1}^m \mathbf{f}^{(p)}. \quad (9.13)$$

Each term of the sum on p , which derives from term P_p in (9.12), may be viewed as contributed by a penalty element with globalized stiffness matrix $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and globalized added force term $\mathbf{f}^{(p)} = w_p \mathbf{a}_p^T \mathbf{b}_p$.

To use a even more compact form we may write the set of multifreedom constraints as $\mathbf{A} \mathbf{u} = \mathbf{b}$. Then the penalty augmented system can be written compactly as

$$(\mathbf{K} + \mathbf{A}^T \mathbf{W} \mathbf{A}) \mathbf{u} = \mathbf{f} + \mathbf{W} \mathbf{A}^T \mathbf{b}, \quad (9.14)$$

where \mathbf{W} is a diagonal matrix of penalty weights. This compact form, however, conceals the configuration of the penalty elements.

§9.2.6. Assessment of the Penalty Method

The main advantage of the penalty function method is its straightforward computer implementation. Looking at modified systems such as (9.2), (9.7) or (9.10) it is obvious that the master equations need not be rearranged. That is, \mathbf{u} and $\hat{\mathbf{u}}$ are the same. Constraints may be programmed as “penalty elements,” and stiffness and force contributions of these elements merged through the standard assembler. In fact using this method there is no need to distinguish between unconstrained and constrained equations! Once all elements — regular and penalty — are assembled, the system can be passed to the equation solver.⁵

An important advantage with respect to the master-slave (elimination) method is its lack of sensitivity with respect to whether constraints are linearly dependent. To give a simplistic example, suppose that the constraint $u_2 = u_6$ appears twice. Then two penalty elements connecting 2 and 6 will be inserted, doubling the intended weight but not otherwise causing undue harm.

An advantage with respect to the Lagrange multiplier method described in §9.2 is that positive definiteness is not lost. Such loss can affect the performance of certain numerical processes.⁶ Finally, it is worth noting that the penalty method is easily extendible to nonlinear constraints although such extension falls outside the scope of this book.

The main disadvantage, however, is a serious one: the choice of weight values that balance solution accuracy with the violation of constraint conditions. For simple cases the square root rule previously described often works, although its effective use calls for knowledge of the magnitude of stiffness coefficients. Such knowledge may be difficult to extract from a general purpose “black box” program. For difficult cases selection of appropriate weights may require extensive numerical experimentation, wasting the user time with numerical games that have no bearing on the actual objective, which is getting a solution.

The deterioration of the condition number of the penalty-augmented stiffness matrix can have serious side effects in some solution procedures such as eigenvalue extraction or iterative solvers.

⁵ Single freedom constraints, such as those encountered in Chapter 3, are usually processed separately for efficiency.

⁶ For example, solving the master stiffness equations by Cholesky factorization or conjugate-gradients.

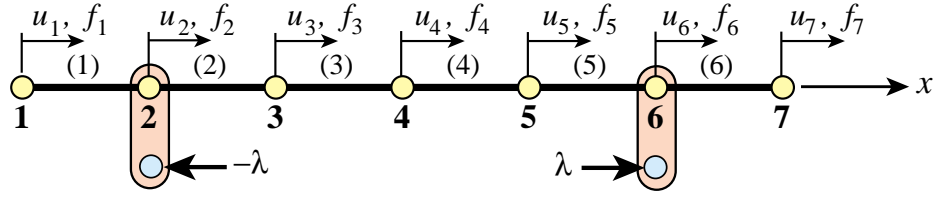


FIGURE 9.3. Physical interpretation of Lagrange multiplier adjunction to enforce the MFC $u_2 = u_6$.

Finally, even if optimal weights are selected, the combined solution error cannot be lowered beyond a threshold value.

From this assessment it is evident that penalty augmentation, although superior to the master-slave method from the standpoint of generality and ease of implementation, is no panacea.

§9.3. Lagrange Multiplier Adjunction

§9.3.1. Physical Interpretation

As in the case of the penalty function method, the method of Lagrange multipliers can be given a rigorous justification within the framework of variational calculus. But in the same spirit it will be introduced for the example structure from a physical standpoint that is particularly illuminating.

Consider again the constraint $u_2 = u_6$. Borrowing some ideas from the penalty method, imagine that nodes 2 and 6 are connected now by a *rigid* link rather than a flexible one. Thus the constraint is imposed exactly. But of course the penalty method with an infinite weight would “blow up.”

We may remove the link if it is replaced by an appropriate reaction force pair $(-\lambda, +\lambda)$, as illustrated in Figure 9.3. These are called the *constraint forces*. Incorporating these forces into the original stiffness equations (8.10) we get

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 - \lambda \\ f_3 \\ f_4 \\ f_5 \\ f_6 + \lambda \\ f_7 \end{bmatrix}. \quad (9.15)$$

This λ is called a *Lagrange multiplier*. Because λ is an unknown, let us transfer it to the *left hand side* by *appending* it to the vector of unknowns:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \quad (9.16)$$

But now we have 7 equations in 8 unknowns. To render the system determinate, the constraint condition $u_2 - u_6 = 0$ is appended as eighth equation:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \end{bmatrix}, \quad (9.17)$$

This is called the *multiplier-augmented* system. Its coefficient matrix, which is symmetric, is called the *bordered stiffness matrix*. The process by which λ is appended to the vector of original unknowns is called *adjunction*. Solving this system provides the desired solution for the degrees of freedom while also characterizing the constraint forces through λ .

§9.3.2. Lagrange Multipliers for General MFCs

The general procedure will be stated first as a recipe. Suppose that we want to solve the example structure subjected to three MFCs

$$u_2 - u_6 = 0, \quad 5u_2 - 8u_7 = 3, \quad 3u_3 + u_5 - 4u_6 = 1, \quad (9.18)$$

Adjoin these MFCs as the eighth, ninth and tenth equations:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & 0 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 & 0 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}, \quad (9.19)$$

Three Lagrange multipliers: λ_1 , λ_2 and λ_3 , are required to take care of three MFCs. Adjoin those unknowns to the nodal displacement vector. Symmetrize the coefficient matrix by appending 3 columns that are the transpose of the 3 last rows in (9.19), and filling the bottom right-hand corner

with a 3×3 zero matrix:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 & 5 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 & -8 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}. \quad (9.20)$$

§9.3.3. *The Theory Behind Lagrange Multipliers

The recipe illustrated by (9.20) comes from a well known technique of variational calculus. Using the matrix notation introduced in §8.1.3, compactly denote the set of m MFCs by $\mathbf{A}\mathbf{u} = \mathbf{b}$, where \mathbf{A} is $m \times n$. The potential energy of the unconstrained finite element model is $\Pi = \frac{1}{2}\mathbf{u}^T \mathbf{K}\mathbf{u} - \mathbf{u}^T \mathbf{f}$. To impose the constraint, adjoin m Lagrange multipliers collected in vector $\boldsymbol{\lambda}$ and form the Lagrangian

$$L(\mathbf{u}, \boldsymbol{\lambda}) = \Pi + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{u} - \mathbf{b}) = \frac{1}{2}\mathbf{u}^T \mathbf{K}\mathbf{u} - \mathbf{u}^T \mathbf{f} + \boldsymbol{\lambda}^T (\mathbf{A}\mathbf{u} - \mathbf{b}). \quad (9.21)$$

Extremizing L with respect to \mathbf{u} and $\boldsymbol{\lambda}$ yields the multiplier-augmented form

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix}. \quad (9.22)$$

The master stiffness matrix \mathbf{K} in (9.22) is said to be *bordered* with \mathbf{A} and \mathbf{A}^T . Solving this system provides \mathbf{u} and $\boldsymbol{\lambda}$. The latter can be interpreted as forces of constraint in the following sense: a removed constraint can be replaced by a system of forces characterized by $\boldsymbol{\lambda}$ multiplied by the constraint coefficients. More precisely, the constraint forces are $-\mathbf{A}^T \boldsymbol{\lambda}$.

§9.3.4. Assessment of the Lagrange Multiplier Method

In contrast to the penalty method, the method of Lagrange multipliers has the advantage of being exact (aside from computational errors due to finite precision arithmetic). It provides directly the constraint forces, which are of interest in many applications. It does not require guesses as regards weights. As the penalty method, it can be extended without difficulty to nonlinear constraints.

It is not free of disadvantages. It introduces additional unknowns, requiring expansion of the original stiffness method, and more complicated storage allocation procedures. It renders the augmented stiffness matrix indefinite, an effect that may cause grief with some linear equation solving methods that rely on positive definiteness. Finally, as the master-slave method, it is sensitive to the degree of linear independence of the constraints: if the constraint $u_2 = u_6$ is specified twice, the bordered stiffness is obviously singular.

On the whole this method appears to be the most elegant one for a general-purpose finite element program that is supposed to work as a “black box” by minimizing guesses and choices from its users. Its implementation, however, is not simple. Special care must be exercised to detect singularities due to constraint dependency and to account for the effect of loss of positive definiteness of the bordered stiffness on equation solvers.

§9.4. *The Augmented Lagrangian Method

The general matrix forms of the penalty function and Lagrangian multiplier methods are given by expressions (9.13) and (9.22), respectively. A useful connection between these methods can be established as follows.

Because the lower diagonal block of the bordered stiffness matrix in (9.22) is null, it is not possible to directly eliminate λ . To make this possible, replace this block by $\epsilon \mathbf{S}^{-1}$, where \mathbf{S} is a constraint-scaling diagonal matrix of appropriate order and ϵ is a small number. The reciprocal of ϵ is a large number called $w = 1/\epsilon$. To maintain exactness of the second equation, $\epsilon \mathbf{S}^{-1} \lambda$ is added to the right-hand side:

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \epsilon \mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \lambda \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \epsilon \mathbf{S}^{-1} \lambda^P \end{bmatrix} \quad (9.23)$$

Here superscript P (for “predicted value”) is attached to the λ on the right-hand side as a “tracer.” We can now formally solve for λ and subsequently for \mathbf{u} . The results may be presented as

$$\begin{aligned} (\mathbf{K} + w \mathbf{A}^T \mathbf{S} \mathbf{A}) \mathbf{u} &= \mathbf{f} + w \mathbf{A}^T \mathbf{S} \mathbf{b} - \mathbf{A}^T \lambda^P, \\ \lambda &= \lambda^P + w \mathbf{S}(\mathbf{b} - \mathbf{A} \mathbf{u}), \end{aligned} \quad (9.24)$$

Setting $\lambda^P = \mathbf{0}$ in the first matrix equation yields

$$(\mathbf{K} + w \mathbf{A}^T \mathbf{S} \mathbf{A}) \mathbf{u} = \mathbf{f} + w \mathbf{A}^T \mathbf{S} \mathbf{b}. \quad (9.25)$$

On taking $\mathbf{W} = w \mathbf{S}$, the general matrix equation (9.13) of the penalty method is recovered.

This relation suggests the construction of *iterative procedures* in which one tries to *improve the accuracy of the penalty function method while w is kept constant* [?]. This strategy circumvents the aforementioned ill-conditioning problems when the weight w is gradually increased. One such method is easily constructed by inspecting (9.24). Using superscript k as an iteration index and keeping w fixed, solve equations (9.24) in tandem as follows:

$$\begin{aligned} (\mathbf{K} + \mathbf{A}^T \mathbf{W} \mathbf{A}) \mathbf{u}^k &= \mathbf{f} + \mathbf{A}^T \mathbf{W} \mathbf{b} - \mathbf{A}^T \lambda^k, \\ \lambda^{k+1} &= \lambda^k + \mathbf{W}(\mathbf{b} - \mathbf{A} \mathbf{u}^k), \end{aligned} \quad (9.26)$$

for $k = 0, 1, \dots$, beginning with $\lambda^0 = \mathbf{0}$. Then \mathbf{u}^0 is the penalty solution. If the process converges one recovers the exact Lagrangian solution without having to solve the Lagrangian system (9.23) directly.

The family of iterative procedures that may be precipitated from (9.24) collectively pertains to the class of *augmented Lagrangian methods*.

§9.5. Summary

The treatment of linear MFCs in finite element systems can be carried out by several methods. Three of these: master-slave elimination, penalty augmentation and Lagrange multiplier adjunction, have been discussed. It is emphasized that no method is uniformly satisfactory in terms of generality, robustness, numerical behavior and simplicity of implementation.

Figure 9.4 gives an assessment of the three techniques in terms of seven attributes.

For a general purpose program that tries to attain “black box” behavior (that is, minimal decisions on the part of users) the method of Lagrange multipliers has the edge. This edge is unfortunately blunted by a fairly complex computer implementation and by the loss of positive definiteness in the bordered stiffness matrix.

	Master-Slave Elimination	Penalty Function	Lagrange Multipliers
Generality	fair	excellent	excellent
Ease of implementation	poor to fair	good	fair
Sensitivity to user decisions	high	high	small to none
Accuracy	variable	mediocre	excellent
Sensitivity as regards constraint dependence	high	none	high
Retains positive definiteness	yes	yes	no
Modifies unknown vector	yes	no	yes

FIGURE 9.4. Assessment summary of three MFC application methods.

Notes and Bibliography

A form of the penalty function method, quite close to that described in §9.1.5, was first proposed by Courant in the early 1940s [151]. It entered the FEM through the work of researchers in the 1960s. There is a good description in the book by Zienkiewicz and Taylor [794].

The Lagrange Multiplier method is much older. Multipliers (called initially “coefficients”) were described by Lagrange in his famous *Mécanique Analytique* monograph [416], as part of the procedure for forming the function now called the Lagrangian. Its use in FEM is more recent than penalty methods.

Augmented Lagrangian methods have received much attention since the late 1960s, when they originated in the field of constrained optimization. The original papers are by Hestenes [345] and Powell [570]. The use of the Augmented Lagrangian Multiplier method for FEM kinematic constraints is first discussed in [204], wherein the iterative algorithm (9.26) for the master stiffness equations is derived.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 9

MultiFreedom Constraints II

EXERCISE 9.1 [C+N:20] This is identical to Exercise 8.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the penalty function method. Take the weight w to be 10^k , in which k varies as $k = 3, 4, 5, \dots, 16$. For each sample w compute the Euclidean-norm solution error $e(w) = \|\mathbf{u}^p(w) - \mathbf{u}^{ex}\|_2$, where \mathbf{u}^p is the computed solution and \mathbf{u}^{ex} is the exact solution listed in (E8.1). Plot $k = \log_{10} w$ versus $\log_{10} e$ and report for which weight e attains a minimum. (See Slide #5 for a check). Does it roughly agree with the square root rule (§9.1.3) if the computations carry 16 digits of precision?

As in Exercise 8.1, use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* script solves this Exercise:

```
(* Exercise 9.1 - Penalty Method *)
(* MFC: u2-u6=1/5 variable w *)
K=MasterStiffnessOfSixElementBar[100];
Print["Stiffness K=",K//MatrixForm];
f={1,2,3,4,5,6,7}; Print["Applied forces=",f];
uexact= {0,0.27,0.275,0.25,0.185,0.07,0.14}; ew={};
For [w=100, w<=10^16, w=10*w; (* increase w by 10 every pass *)
  Khat=K; fhat=f;
  Khat[[2,2]]+=w; Khat[[6,6]]+=w; Khat[[6,2]]=Khat[[2,6]]-=w;
  fhat[[2]]+=(1/5)*w; fhat[[6]]=(1/5)*w; (*insert penalty *)
  {Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat];
  u=LinearSolve[N[Kmod],N[fmod]];
  Print["Weight w=",N[w]//ScientificForm," u=",u//InputForm];
  e=Sqrt[(u-uexact).(u-uexact)];
  (*Print["L2 solution error=",e//ScientificForm]; *)
  AppendTo[ew,{Log[10,w],Log[10,e]}];
];
ListPlot[ew,AxesOrigin->{5,-8},Frame->True, PlotStyle->
  {AbsolutePointSize[4],AbsoluteThickness[2],RGBColor[1,0,0]},
  PlotJoined->True,AxesLabel->{"Log10(w)","Log10(u error)"}];
```

Here *MasterStiffnessOfSixElementBar* and *FixLeftEndOfSixElementBar* are the same modules listed in Exercise 8.1.

Note: If you run the above program, you may get several beeps from *Mathematica* as it is processing some of the systems with very large weights. Don't be alarmed: those are only warnings. The *LinearSolve* function is alerting you that the coefficient matrices $\hat{\mathbf{K}}$ for weights of order 10^{12} or bigger are ill-conditioned.

EXERCISE 9.2 [C+N:15] Again identical to Exercise 8.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the Lagrange multiplier method. The results for the computed \mathbf{u} and the recovered force vector $\mathbf{K}\mathbf{u}$ should agree with (E8.1). Use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* script solves this Exercise:

```
(* Exercise 9.2 - Lagrange Multiplier Method *)
(* MFC: u2-u6=1/5 *)
K=MasterStiffnessOfSixElementBar[100];
Khat=Table[0,{8},{8}]; f={1,2,3,4,5,6,7}; fhat=AppendTo[f,0];
For [i=1,i<=7,i++, For[j=1,j<=7,j++, Khat[[i,j]]=K[[i,j]]]];
{Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat];
```

```

Kmod[[2,8]]=Kmod[[8,2]]= 1;
Kmod[[6,8]]=Kmod[[8,6]]=-1; fmod[[8]]=1/5;
Print["Kmod=",Kmod//MatrixForm];
Print["fmod=",fmod];
umod=LinearSolve[N[Kmod],N[fmod]]; u=Take[umod,7];
Print["Solution u=",u ,",   lambda=",umod[[8]]];
Print["Recovered node forces=",K.u];

```

Here `MasterStiffnessOfSixElementBar` and `FixLeftEndOfSixElementBar` are the same modules listed in Exercise 8.1.

Does the computed solution agree with (E8.1)?

EXERCISE 9.3 [A:10] For the example structure, show which penalty elements would implement the following MFCs:

$$\begin{aligned} \text{(a)} \quad u_2 + u_6 &= 0, \\ \text{(b)} \quad u_2 - 3u_6 &= 1/3. \end{aligned} \tag{E9.1}$$

As answer, show the stiffness equations of those two elements in a manner similar to (9.1).

EXERCISE 9.4 [A/C+N:15+15+10] Suppose that the assembled stiffness equations for a one-dimensional finite element model before imposing constraints are

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}. \tag{E9.2}$$

This system is to be solved subject to the multipoint constraint

$$u_1 = u_3. \tag{E9.3}$$

- Impose the constraint (E9.3) by the master-slave method taking u_1 as master, and solve the resulting 2×2 system of equations by hand.
- Impose the constraint (E9.3) by the penalty function method, leaving the weight w as a free parameter. Solve the equations by hand or CAS (Cramer's rule is recommended) and verify analytically that as $w \rightarrow \infty$ the solution approaches that found in (a). Tabulate the values of u_1, u_2, u_3 for $w = 0, 1, 10, 100$. *Hint 1:* the value of u_2 should not change. *Hint 2:* the solution for u_1 should be $(6w + 5)/(4w + 4)$.
- Impose the constraint (E9.3) by the Lagrange multiplier method. Show the 4×4 multiplier-augmented system of equations analogous to (9.13) and solve it by computer or calculator.

EXERCISE 9.5 [A/C:10+15+10] The left end of the cantilevered beam-column member illustrated in Figure E9.1 rests on a skew-roller that forms a 45° angle with the horizontal axis x . The member is loaded axially by a force P as shown. The finite element equations upon removing the fixed right end freedoms $\{u_{x2}, u_{y2}, \theta_2\}$, but *before* imposing the skew-roller MFC, are

$$\begin{bmatrix} EA/L & 0 & 0 \\ 0 & 12EI/L^3 & 6EI/L^2 \\ 0 & 6EI/L^2 & 4EI/L \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_1 \end{bmatrix} = \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix}, \tag{E9.4}$$

where E , A , and $I = I_{zz}$ are given member properties, θ_1 is the left end rotation, and L is the member length.⁷

⁷ The stiffness equations for a beam column are derived in Part III of this book. For now consider (E9.4) as a recipe.

To simplify the calculations set $P = \alpha EA$, and $I = \beta AL^2$, in which α and β are dimensionless parameters, and express the following solutions in terms of α and β .

- Apply the skew-roller constraint by the master-slave method (make u_{y1} slave) and solve for u_{x1} and θ_1 in terms of L , α and β . This may be done by hand or a CAS. Partial solution: $u_{x1} = \alpha L / (1 + 3\beta)$.
- Apply the skew-roller constraint with the penalty method by inserting a penalty element at node 1. Follow the rule of §9.1.4 to construct the 2×2 penalty stiffness. Compute u_{x1} from the modified equations (Cramer's rule is recommended if solved by hand). Verify that as $w \rightarrow \infty$ the answer obtained in (a) is recovered. Partial solution: $u_{x1} = \alpha L (3EA\beta + wL) / (3EA\beta + wL(1 + 3\beta))$. Can the penalty stiffness be physically interpreted in some way?

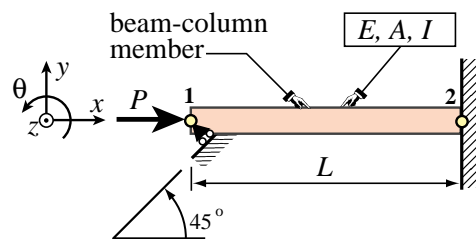


FIGURE E9.1. Cantilevered beam-column on skew-roller for Exercise 9.5.

- Apply the skew roller constraint by Lagrangian multiplier adjunction, and solve the resulting 4×4 system of equations using a CAS (by hand it will take long). Verify that you get the same solution as in (a).

EXERCISE 9.6 [A:5+5+10+10+5] A cantilever beam-column is to be joined to a plane stress plate mesh as depicted in Figure E9.2.⁸ Both pieces move in the plane $\{x, y\}$. Plane stress elements have two degrees of freedom per node: two translations u_x and u_y along x and y , respectively, whereas a beam-column element has three: two translations u_x and u_y along x and y , and one rotation (positive CCW) θ_z about z . To connect the cantilever beam to the mesh, the following “gluing” conditions are applied:

- The horizontal (u_x) and vertical (u_y) displacements of the beam at their common node (2 of beam, 4 of plate) are the same.
- The beam end rotation θ_2 and the mean rotation of the plate edge 3–5 are the same. For infinitesimal displacements and rotations the latter is $\theta_{35}^{avg} = (u_{x5} - u_{x3}) / H$.

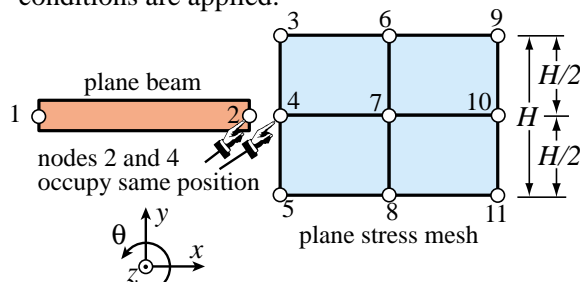


FIGURE E9.2. Beam linked to plate in plane stress for Exercise 9.6. Beam shown slightly separate from plate for visualization convenience: nodes 2 and 4 actually are at the same location.

Questions:

- Write down the three MFC conditions: two from (1) and one from (2), and state whether they are linear and homogeneous.
- Where does the above expression of θ_{35}^{avg} come from? (Geometric interpretation is OK.) Can it be made more accurate⁹ by including u_{x4} ?
- Write down the master-slave transformation matrix if $\{u_{x2}, u_{y2}, \theta_2\}$ are picked as slaves. It is sufficient to write down the transformation for the DOFs of nodes 2, 3, 4, and 5, which gives a \mathbf{T} of order 9×6 , since the transformations for the other freedoms are trivial.
- If the penalty method is used, write down the stiffness equations of the three penalty elements assuming the same weight w is used. Their stiffness matrices are of order 2×2 , 2×2 and 3×3 , respectively. (Do not proceed further)

⁸ This is extracted from a question previously given in the Aerospace Ph. D. Preliminary Exam. Technically it is not difficult once the student understand what is being asked. This can take some time, but a HW is more relaxed.

⁹ To answer the second question, observe that the displacements along 3–4 and 4–5 vary linearly. Thus the angle of rotation about z is constant for each of them, and (for infinitesimal displacements) may be set equal to the tangent.

- (e) If Lagrange multiplier adjunction is used, how many Lagrange multipliers will you need to append? (Do not proceed further).

EXERCISE 9.7 [A:30] Show that the master-slave transformation method $\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}$ can be written down as a special form of the method of Lagrange multipliers. Start from the augmented functional

$$\Pi_{MS} = \frac{1}{2}\mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{f} + \boldsymbol{\lambda}^T (\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \quad (\text{E9.5})$$

and write down the stationarity conditions of Π_{MS} with respect to \mathbf{u} , $\boldsymbol{\lambda}$ and $\hat{\mathbf{u}}$ in matrix form.

EXERCISE 9.8 [A:35] Check the matrix equations (9.23) through (9.26) quoted for the Augmented Lagrangian method.

EXERCISE 9.9 [A:40] (Advanced, close to a research paper). Show that the master-slave transformation method $\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}$ can be expressed as a limit of the penalty function method as the weights go to infinity. Start from the augmented functional

$$\Pi_P = \frac{1}{2}\mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{u}^T \mathbf{f} + \frac{1}{2}w(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}})^T (\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \quad (\text{E9.6})$$

Write down the matrix stationarity conditions with respect to \mathbf{u} and $\hat{\mathbf{u}}$ and take the limit $w \rightarrow \infty$. *Hint:* using Woodbury's formula (Appendix C, §C.5.2)

$$(\mathbf{K} + w\mathbf{T}^T \mathbf{S} \mathbf{T})^{-1} = \mathbf{K}^{-1} - \mathbf{K}^{-1} \mathbf{T}^T (\bar{\mathbf{K}} + w^{-1} \mathbf{S}^{-1})^{-1} \mathbf{T} \mathbf{K}^{-1}. \quad (\text{E9.7})$$

show that

$$\bar{\mathbf{K}}^{-1} = \mathbf{T} \mathbf{K}^{-1} \mathbf{T}^T. \quad (\text{E9.8})$$

10

Superelements and Global-Local Analysis

TABLE OF CONTENTS

	Page
§10.1. Superelement Concept	10–3
§10.1.1. Where Does the Idea Come From?	10–3
§10.1.2. Subdomains	10–5
§10.1.3. *Mathematical Requirements	10–5
§10.2. Static Condensation	10–5
§10.2.1. Condensation by Explicit Matrix Operations	10–5
§10.2.2. Condensation by Symmetric Gauss Elimination	10–6
§10.2.3. Recovery of Internal Freedoms	10–8
§10.3. Global-Local Analysis	10–8
§10. Notes and Bibliography	10–10
§10. References	10–10
§10. Exercises	10–11

§10.1. Superelement Concept

A superelement is a grouping of finite elements that, upon assembly, may be regarded as an *individual element* for computational purposes. These purposes may be driven by modeling or processing needs.

A random assortment of elements does not necessarily make up a superelement. To be considered as such, a grouping must meet certain conditions. Informally we can say that it must form a structural component on its own. This imposes certain conditions stated mathematically in §10.1.3. Inasmuch as these conditions involve advanced concepts such as rank sufficiency, which are introduced in later Chapters, the restrictions are not dwelled upon here.

As noted in Chapter 6, superelements may originate from two overlapping contexts: “bottom up” or “top down.” In a bottom up context one thinks of superelements as built from simpler elements. In a top-down context, superelements may be thought as being large pieces of a complete structure. This dual viewpoint motivates the following classification:

Macroelements. These are superelements assembled with a few primitive elements. Also called *mesh units* when they are presented to program users as individual elements.

Substructures. Complex assemblies of elements that result on breaking up a structure into distinguishable portions.

When does a substructure becomes a macroelement or vice-versa? There are no precise rules. In fact the generic term *superelement* was coined to cover the entire spectrum, ranging from *individual elements* to *complete structures*. This universality is helped by common processing features.

Both macroelements and substructures are treated exactly the same way as regards matrix processing. The basic rule is that associated with *condensation* of internal degrees of freedom. The technique is illustrated in the following section with a simple example. The reader should note, however, that condensation applies to *any* superelement, whether composed of two or a million elements.¹

§10.1.1. Where Does the Idea Come From?

Substructuring was invented by aerospace engineers in the early 1960s² to carry out a first-level breakdown of complex systems such as a complete airplane, as depicted in Figure 10.1. The decomposition may continue hierarchically through additional levels as illustrated in Figure 10.2. The concept is also natural for space vehicles operating in stages, such as the Apollo short stack depicted in Figure 10.3.

Three original motivating factors for substructuring can be cited.

1. **Facilitate division of labor.** Substructures with different functions are done by separate design groups with specialized knowledge and experience. For instance an aircraft company may set up a fuselage group, a wing group, a landing-gear group, etc. These groups are thus protected from “hurry up and wait” constraints. More specifically: a wing design group can

¹ Of course the computer implementation becomes totally different as one goes from macroelements to substructures, because efficient processing for large matrix systems requires exploitation of sparsity.

² See **Notes and Bibliography** at the end of this Chapter.

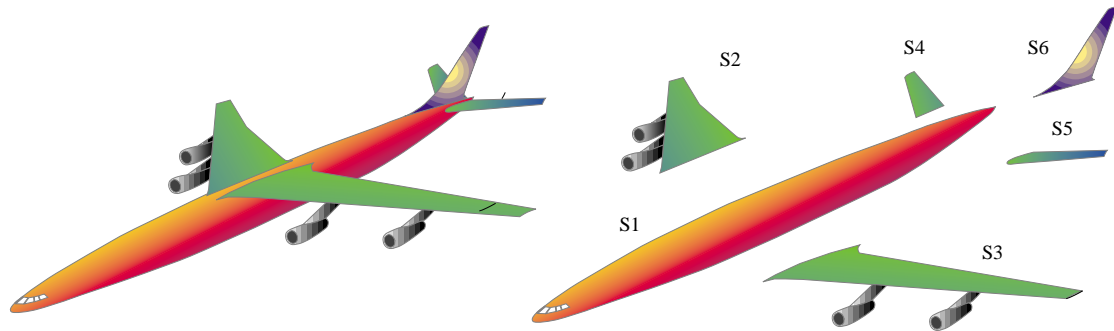


FIGURE 10.1. Complete airplane broken down into six level one substructures identified as S_1 through S_6 .

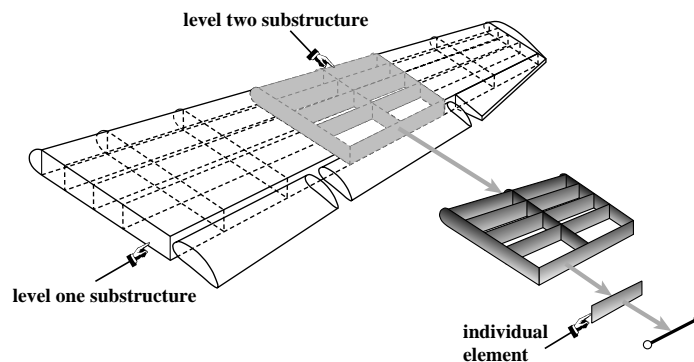


FIGURE 10.2. Further breakdown of wing structure. The decomposition process may continue down to the individual element level.

keep on working on refinements, improvements and experimental model verification as long as the interface information (the wing-fuselage intersection) stays sensibly unchanged.

2. **Take advantage of repetition.** Often structures are built of several identical or nearly identical units. For instance, the wing substructures S_2 and S_3 of Figure 10.1 are mirror images on reflection about the fuselage midplane, and so are the stabilizers S_4 and S_5 . Even if the loading is not symmetric about that midplane, recognizing repetitions saves model preparation time.
3. **Overcome computer limitations.** The computers of the 1960s operated under serious memory limitations. (For example, the first supercomputer: the Control Data 6600, had a total high-speed memory of 131072 60-bit words or 1.31 MB; that machine cost \$10M in 1966 dollars.) It was difficult to fit a complex structure such as an airplane as one entity. Substructuring permitted the complete analysis to be carried out in stages through use of auxiliary storage devices such as tapes or disks.

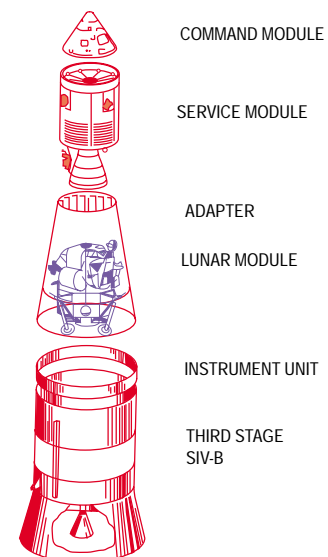


FIGURE 10.3. The Apollo short stack.

Of the three motivations, the first two still hold today. The third one has moved to a different plane:

parallel processing, as noted in §10.1.2 below.

In the late 1960s the idea was picked up and developed extensively by the offshore and shipbuilding industries, the products of which tend to be modular and repetitive to reduce fabrication costs. As noted above, repetition favors the use of substructuring techniques.

At the other end of the superelement spectrum, the mesh units herein called macroelements appeared in the mid 1960s. They were motivated by user convenience. For example, in hand preparation of models, quadrilateral and bricks involve less human labor than triangles and tetrahedra, respectively. It was therefore natural to combine the latter to assemble the former. Going a step further one can assemble components such as “box elements” for applications such as box-girder bridges.

§10.1.2. Subdomains

Applied mathematicians working on solution procedures for parallel computation have developed the concept of *subdomains*. These are groupings of finite elements that are entirely motivated by computational considerations. They are subdivisions of the finite element model done more or less automatically by a program called *domain decomposer*.

Although the concepts of substructures and subdomains overlap in many respects, it is better to keep the two separate. The common underlying theme is divide and conquer but the motivation is different.

§10.1.3. *Mathematical Requirements

A superelement is said to be *rank-sufficient* if its only zero-energy modes are rigid-body modes. Equivalently, the superelement does not possess spurious kinematic mechanisms.

Verification of the rank-sufficient condition guarantees that the static condensation procedure described below will work properly.

§10.2. Static Condensation

Degrees of freedom of a superelement are classified into two groups:

Internal Freedoms. Those that are not connected to the freedoms of another superelement. Nodes whose freedoms are internal are called *internal nodes*.

Boundary Freedoms. These are connected to at least another superelement. They usually reside at *boundary nodes* placed on the periphery of the superelement. See Figure 10.4.

The objective is to get rid of all displacement degrees of freedom associated with *internal freedoms*. This elimination process is called *static condensation*, or simply *condensation*.

Condensation may be presented in terms of explicit matrix operations, as shown in the next subsection. A more practical technique based on symmetric Gauss elimination is discussed later.

§10.2.1. Condensation by Explicit Matrix Operations

To carry out the condensation process, the assembled stiffness equations of the superelement are partitioned as follows:

$$\begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{bi} \\ \mathbf{K}_{ib} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \mathbf{f}_i \end{bmatrix}. \quad (10.1)$$

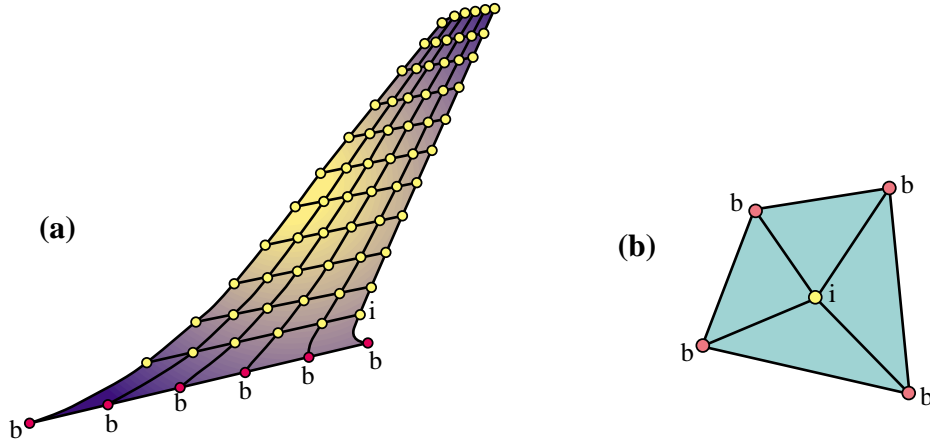


FIGURE 10.4. Classification of superelement freedoms into boundary and internal. (a) shows the vertical stabilizer substructure S_6 of Figure 10.2. (The FE mesh is pictured as two-dimensional for illustrative purposes; for an actual aircraft it will be three dimensional.) Boundary freedoms are those associated to the boundary nodes labeled b (shown in red), which are connected to the fuselage substructure. (b) shows a quadrilateral macroelement mesh-unit fabricated with 4 triangles: it has one interior and four boundary nodes.

where subvectors \mathbf{u}_b and \mathbf{u}_i collect *boundary* and *interior* degrees of freedom, respectively. Take the second matrix equation:

$$\mathbf{K}_{ib}\mathbf{u}_b + \mathbf{K}_{ii}\mathbf{u}_i = \mathbf{f}_i, \quad (10.2)$$

If \mathbf{K}_{ii} is nonsingular we can solve for the interior freedoms:

$$\mathbf{u}_i = \mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{K}_{ib}\mathbf{u}_b), \quad (10.3)$$

Replacing into the first matrix equation of (10.1) yields the *condensed stiffness equations*

$$\tilde{\mathbf{K}}_{bb}\mathbf{u}_b = \tilde{\mathbf{f}}_b. \quad (10.4)$$

In this equation,

$$\tilde{\mathbf{K}}_{bb} = \mathbf{K}_{bb} - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib}, \quad \tilde{\mathbf{f}}_b = \mathbf{f}_b - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{f}_i, \quad (10.5)$$

are called the *condensed* stiffness matrix and force vector, respectively, of the substructure.

From this point onward, the condensed superelement may be viewed, from the standpoint of further operations, as an *individual element* whose element stiffness matrix and nodal force vector are $\tilde{\mathbf{K}}_{bb}$ and $\tilde{\mathbf{f}}_b$, respectively.

Often each superelement has its own “local” coordinate system. A transformation of (10.5) to an overall global coordinate system is necessary upon condensation. In the case of multiple levels, the transformation is done with respect to the next-level superelement coordinate system. This coordinate transformation procedure automates the processing of repeated portions.

Remark 10.1. The feasibility of the condensation process (10.3)–(10.5) rests on the non-singularity of \mathbf{K}_{ii} . This matrix is nonsingular if the superelement is rank-sufficient in the sense stated in §10.1.3, and if fixing the boundary freedoms precludes all rigid body motions. If the former condition is verified but not the latter, the superelement is called *floating*. Processing floating superelements demands more advanced computational techniques, among which one may cite the use of projectors and generalized inverses [220].

```

CondenseLastFreedom[K_,f_]:=Module[{pivot,c,Kc,fc,
n=Length[K]}, If [n<=1,Return[{K,f}]];
Kc=Table[0,{n-1},{n-1}]; fc=Table[0,{n-1}];
pivot=K[[n,n]]; If [pivot==0, Print["CondenseLastFreedom:",
" Singular Matrix"]; Return[{K,f}]];
For [i=1,i<=n-1,i++, c=K[[i,n]]/pivot;
fc[[i]]=f[[i]]-c*f[[n]];
For [j=1,j<=i,j++,
Kc[[j,i]]=Kc[[i,j]]=K[[i,j]]-c*K[[n,j]]
];
];
Return[Simplify[{Kc,fc}]]
];

K={{6,-2,-1,-3},{-2,5,-2,-1},{-1,-2,7,-4},{-3,-1,-4,8}};
f={3,6,4,0};
Print["Before condensation:", " K=",K//MatrixForm, " f=",f//MatrixForm];
{K,f}=CondenseLastFreedom[K,f];Print["Upon condensing DOF 4:",
" K=",K//MatrixForm, " f=",f//MatrixForm];
{K,f}=CondenseLastFreedom[K,f];Print["Upon condensing DOF 3:",
" K=",K//MatrixForm, " f=",f//MatrixForm];

```

FIGURE 10.5. *Mathematica* module to condense the last degree of freedom from a stiffness matrix and force vector. The test statements carry out the example (10.6)–(10.10).

§10.2.2. Condensation by Symmetric Gauss Elimination

In the computer implementation of the static condensation process, calculations are not carried out as outlined above. There are two major differences. The equations of the substructure are not actually rearranged, and the explicit calculation of the inverse of \mathbf{K}_{ii} is avoided. The procedure may be in fact coded as a variant of symmetric Gauss elimination. To convey the flavor of this technique, consider the following stiffness equations of a superelement:

$$\begin{bmatrix} 6 & -2 & -1 & -3 \\ -2 & 5 & -2 & -1 \\ -1 & -2 & 7 & -4 \\ -3 & -1 & -4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \\ 0 \end{bmatrix}. \quad (10.6)$$

Suppose that the last two displacement freedoms: u_3 and u_4 , are classified as interior and are to be statically condensed out. To eliminate u_4 , perform symmetric Gauss elimination of the fourth row and column:

$$\begin{bmatrix} 6 - \frac{(-3) \times (-3)}{8} & -2 - \frac{(-1) \times (-3)}{8} & -1 - \frac{(-4) \times (-3)}{8} \\ -2 - \frac{(-3) \times (-1)}{8} & 5 - \frac{(-1) \times (-1)}{8} & -2 - \frac{(-4) \times (-1)}{8} \\ -1 - \frac{(-3) \times (-4)}{8} & -2 - \frac{(-1) \times (-4)}{8} & 7 - \frac{(-4) \times (-4)}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3 - \frac{0 \times (-3)}{8} \\ 6 - \frac{0 \times (-1)}{8} \\ 4 - \frac{0 \times (-4)}{8} \end{bmatrix}, \quad (10.7)$$

or

$$\begin{bmatrix} \frac{39}{8} & -\frac{19}{8} & -\frac{5}{2} \\ -\frac{19}{8} & \frac{39}{8} & -\frac{5}{2} \\ -\frac{5}{2} & -\frac{5}{2} & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix}. \quad (10.8)$$

Repeat the process for the third row and column to eliminate u_3 :

$$\begin{bmatrix} \frac{39}{8} - \frac{(-5/2) \times (-5/2)}{5} & -\frac{19}{8} - \frac{(-5/2) \times (-5/2)}{5} \\ -\frac{19}{8} - \frac{(-5/2) \times (-5/2)}{5} & \frac{39}{8} - \frac{(-5/2) \times (-5/2)}{5} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 3 - \frac{4 \times (-5/2)}{5} \\ 6 - \frac{4 \times (-5/2)}{5} \end{bmatrix}, \quad (10.9)$$

or

$$\begin{bmatrix} \frac{29}{8} & -\frac{29}{8} \\ -\frac{29}{8} & \frac{29}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}. \quad (10.10)$$

These are the condensed stiffness equations. Figure 10.5 shows a *Mathematica* program that carries out the foregoing steps. Module `CondenseLastFreedom` condenses the last freedom of a stiffness matrix K and a force vector f . It is invoked as $\{K_c, f_c\} = \text{CondenseLastFreedom}[K, f]$. It returns the condensed stiffness K_c and force vector f_c as new arrays. To do the example (10.6)–(10.10), the module is called twice, as illustrated in the test statements of Figure 10.5.

Obviously this procedure is much simpler than going through the explicit matrix inverse. Another important advantage of Gauss elimination is that equation rearrangement is not required even if the condensed degrees of freedom do not appear sequentially. For example, suppose that the assembled superelement contains originally eight degrees of freedom and that the freedoms to be condensed out are numbered 1, 4, 5, 6 and 8. Then Gauss elimination is carried out over those equations only, and the condensed (3×3) stiffness and (3×1) force vector extracted from rows and columns 2, 3 and 7. An implementation of this process is considered in Exercise 10.2.

Remark 10.2. The symmetric Gauss elimination procedure, as illustrated in steps (10.6)–(10.10), is primarily useful for macroelements and mesh units, since the number of stiffness equations for those typically does not exceed a few hundreds. This permits the use of full matrix storage. For substructures containing thousands or millions of degrees of freedom — such as in the airplane example — the elimination is carried out using more sophisticated sparse matrix algorithms; for example that described in [188].

Remark 10.3. The static condensation process is a matrix operation called “partial inversion” or “partial elimination” that appears in many disciplines. Here is the general form. Suppose the linear system $\mathbf{Ax} = \mathbf{y}$, where \mathbf{A} is $n \times n$ square and \mathbf{x} and \mathbf{y} are n -vectors, is partitioned as

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}. \quad (10.11)$$

Assuming the appropriate inverses to exist, then the following are easily verified matrix identities:

$$\begin{bmatrix} \mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix}. \quad (10.12)$$

We say that \mathbf{x}_1 has been eliminated or “condensed out” in the left identity and \mathbf{x}_2 in the right one. In FEM applications, it is conventional to condense out the bottom vector \mathbf{x}_2 , so the right identity is relevant. If \mathbf{A} is symmetric, to retain symmetry in (10.12) it is necessary to change the sign of one of the subvectors.

§10.2.3. Recovery of Internal Freedoms

After the boundary freedoms are obtained by the solution process, the interior freedoms can be recovered directly from (10.3). This process may be carried out either directly through a sequence of matrix operations, or equation by equation as a backsubstitution process.

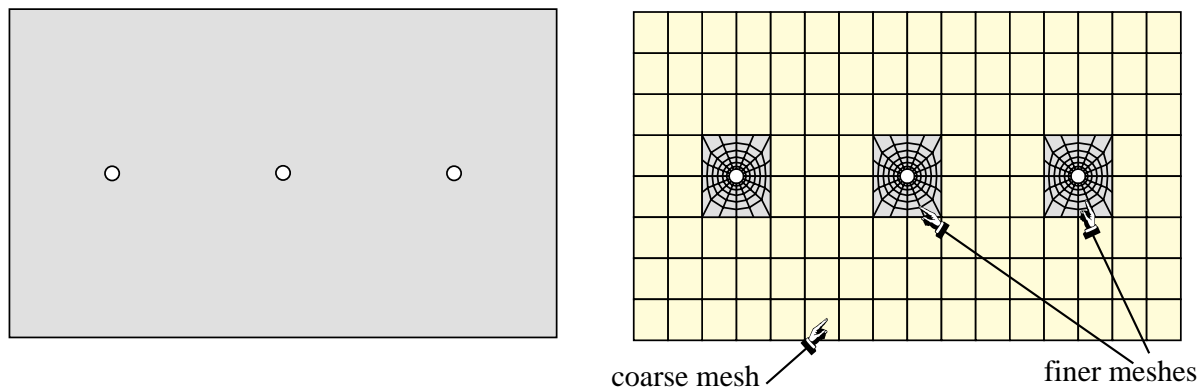


FIGURE 10.6. Left: example panel structure for global-local analysis. Right: a FEM mesh for a one-shot analysis.

§10.3. Global-Local Analysis

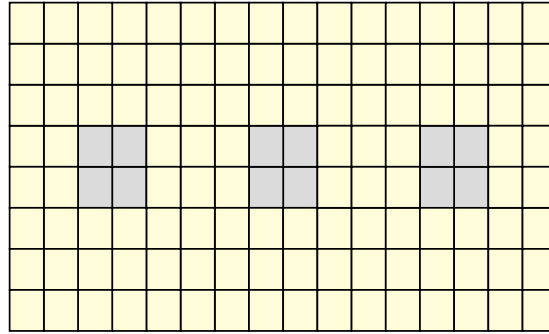
As discussed in the first Chapter, complex engineering systems are often modeled in a *multilevel* fashion following the divide and conquer approach. The superelement technique is a practical realization of that approach.

A related, but not identical, technique is *multiscale* analysis. The whole system is first analyzed as a global entity, discarding or passing over details deemed not to affect its overall behavior. Local details are then analyzed using the results of the global analysis as boundary conditions. The process can be continued into the analysis of further details of local models. And so on. When this procedure is restricted to two stages and applied in the context of finite element analysis, it is called *global-local* analysis in the FEM literature.

In the global stage the behavior of the entire structure is simulated with a finite element model that necessarily ignores details such as cutouts or joints. These details do not affect the overall behavior of the structure, but may have a bearing on safety. Such details are *a posteriori* incorporated in a series of local analyses.

The gist of the global-local approach is explained in the example illustrated in Figures 10.6 and 10.7. Although the structure is admittedly too simple to merit the application of global-local analysis, it serves to illustrate the basic ideas. Suppose one is faced with the analysis of the rectangular panel shown on the top of Figure 10.6, which contains three small holes. The bottom of that figure shows a standard (one-stage) FEM treatment using a largely regular mesh that is refined near the holes. Connecting the coarse and fine meshes usually involves using multifreedom constraints because the nodes at mesh boundaries do not match, as depicted in that figure.

Figure 10.7 illustrates the global-local analysis procedure. The global analysis is done with a coarse but regular FEM mesh that *ignores the effect of the holes*. This is followed by local analysis of the region near the holes using refined finite element meshes. The key ingredient for the local analyses is the application of boundary conditions (BCs) on the finer mesh boundaries. These BCs may be of displacement (essential) or of force (natural) type. If the former, the applied boundary displacements are interpolated from the global mesh solution. If the latter, the internal forces or stresses obtained from the global calculation are converted to nodal forces on the fine meshes through a lumping process.



Global analysis with a coarse mesh, ignoring holes, followed bylocal analysis of the vicinity of the holes with finer meshes:

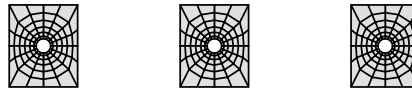


FIGURE 10.7. Global-local analysis of problem of Figure 10.6.

The BC choice noted above gives rise to two basic variations of the global-local approach. Experience accumulated over several decades³ has shown that the stress-BC approach generally gives more reliable answers.

The global-local technique can be extended to more than two levels, in which case it receives the more encompassing name *multiscale analysis*. Although this generalization is still largely in the realm of research, it is receiving increasing attention from various science and engineering communities for complex products such as the thermomechanical analysis of microelectronic components.

Notes and Bibliography

Substructuring was invented by aerospace engineers in the early 1960s. Przemieniecki's book [536] contains a fairly complete bibliography of early work. Most of this was in the form of largely inaccessible internal company or lab reports and so the actual history is difficult to trace. Macroelements appeared simultaneously in many of the early FEM codes. Quadrilateral macroelements fabricated with triangles are described in [183]. For a survey of uses of the static condensation algorithm in FEM, see [712].

The generic term *superelement* was coined in the late 1960s by the SESAM group at DNV Veritas [177]. The matrix form of static condensation for a complete structure is presented in [21], as a scheme to eliminate unloaded DOF in the displacement method. It is unclear when this idea was first applied to substructures or macroelements. The first application for reduced dynamical models is by Guyan [294].

The application of domain decomposition to parallel FEM solvers has produced an enormous and highly specialized literature. Procedures for handling floating superelements using generalized inverse methods are discussed in [210,220,508].

The global-local analysis procedure described here is also primarily used in industry and as such it is rarely mentioned in academic textbooks.

References

Referenced items have been moved to Appendix R.

³ Particularly in the aerospace industry, in which the global-local technique has been used since the early 1960s.

Homework Exercises for Chapter 10

Superelements and Global-Local Analysis

EXERCISE 10.1 [N:15] The free-free stiffness equations of a superelement are

$$\begin{bmatrix} 88 & -44 & -44 & 0 \\ -44 & 132 & -44 & -44 \\ -44 & -44 & 176 & -44 \\ 0 & -44 & -44 & 220 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 5 \\ 10 \\ 15 \\ 20 \end{bmatrix}. \quad (\text{E10.1})$$

Eliminate u_2 and u_3 from (E10.1) by static condensation, and show (but do not solve) the condensed equation system. Use either the explicit matrix inverse formulas 10.5 or the symmetric Gauss elimination process explained in §10.2.2. Hint: regardless of method the result should be

$$\begin{bmatrix} 52 & -36 \\ -36 & 184 \end{bmatrix} \begin{bmatrix} u_1 \\ u_4 \end{bmatrix} = \begin{bmatrix} 15 \\ 30 \end{bmatrix}. \quad (\text{E10.2})$$

EXERCISE 10.2 [C+N:20] Generalize the program of Figure 10.5 to a module `CondenseFreedom[K, f, k]` that is able to condense the k th degree of freedom from \mathbf{K} and \mathbf{f} , which is not necessarily the last one. That is, k may range from 1 to n , where n is the number of freedoms in $\mathbf{K}\mathbf{u} = \mathbf{f}$. Apply that program to solve the previous Exercise.

Hint: here is a possible way of organizing the inner loop:

```
ii=0; For [i=1,i<=n,i++, If [i==k, Continue[]]; ii++;
      c=K[[i,k]]/pivot; fc[[ii]]=f[[i]]-c*f[[k]]; jj=0;
      For [j=1,j<=n,j++, If [j==k, Continue[]]; jj++;
        Kc[[ii,jj]]=K[[i,j]]-c*K[[k,j]]
      ];
];
Return[{Kc,fc}]
```

EXERCISE 10.3 [D:15] Explain the similarities and differences between superelement analysis and global-local FEM analysis.

EXERCISE 10.4 [A:20] If the superelement stiffness \mathbf{K} is symmetric, the static condensation process can be viewed as a special case of the master-slave transformation method discussed in Chapter 8. To prove this, take exterior freedoms \mathbf{u}_b as masters and interior freedoms \mathbf{u}_i as slaves. Assume the master-slave transformation relation

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib} \end{bmatrix} \begin{bmatrix} \mathbf{u}_b \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ -\mathbf{K}_{ii}^{-1}\mathbf{f}_i \end{bmatrix} \stackrel{\text{def}}{=} \mathbf{T}\mathbf{u}_b - \mathbf{g}. \quad (\text{E10.3})$$

Work out $\hat{\mathbf{K}} = \mathbf{T}^T \mathbf{K} \mathbf{T}$ and $\hat{\mathbf{f}} = \mathbf{T}^T (\mathbf{f} - \mathbf{K}\mathbf{g})$, and show that $\hat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}$ coalesces with the condensed stiffness equations (10.4)–(10.5) if $\mathbf{u}_b \equiv \hat{\mathbf{u}}$. (Take advantage of the symmetry properties $\mathbf{K}_{bi}^T = \mathbf{K}_{ib}$, $(\mathbf{K}_{ii}^{-1})^T = \mathbf{K}_{ii}^{-1}$.)

EXERCISE 10.5 [D:30] (Requires thinking) Explain the conceptual and operational differences between one-stage FEM analysis and global-local analysis of a problem such as that illustrated in Figures 10.6 and 10.7. Are the answers the same? What is gained, if any, by the global-local approach over the one-stage FEM analysis?

EXERCISE 10.6 [N:20] The widely used Guyan's scheme [294] for dynamic model reduction applies the static-condensation relation (E10.3) as master-slave transformation to both the stiffness and the mass matrix of the superelement. Use this procedure to eliminate the second and third DOF of the mass-stiffness system:

$$\mathbf{M} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \quad (\text{E10.4})$$

Compute and compare the vibration frequencies of the eigensystems $(\mathbf{M} + \omega_i^2 \mathbf{K})\mathbf{v}_i = \mathbf{0}$ and $(\hat{\mathbf{M}} + \hat{\omega}_i^2 \hat{\mathbf{K}})\hat{\mathbf{v}}_i = \mathbf{0}$ before and after reduction.

Hints. The four original squared frequencies are the eigenvalues of $\mathbf{M}^{-1}\mathbf{K}$, which may be obtained, for example, with the Matlab `eig` function. The largest is 2, lowest 0. To perform the Guyan reduction is convenient to reorder \mathbf{M} and \mathbf{K} so that rows and columns 2 and 3 become 3 and 4, respectively:

$$\omega^2 \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 4 & 1 \\ 0 & 1 & 1 & 4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_4 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ -1 & 0 & 2 & -1 \\ 0 & -1 & -1 & 2 \end{bmatrix} \begin{bmatrix} v_1 \\ v_4 \\ v_2 \\ v_3 \end{bmatrix}, \quad (\text{E10.5})$$

which in partitioned matrix form is

$$\omega^2 \begin{bmatrix} \mathbf{M}_{bb} & \mathbf{M}_{bi} \\ \mathbf{M}_{ib} & \mathbf{M}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{v}_b \\ \mathbf{v}_i \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{bi} \\ \mathbf{K}_{ib} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{v}_b \\ \mathbf{v}_i \end{bmatrix}. \quad (\text{E10.6})$$

Next show that static condensation of \mathbf{K} is equivalent to a master-slave transformation

$$\mathbf{T} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{ii}^{-1} \mathbf{K}_{ib} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \quad \text{relating} \quad \begin{bmatrix} v_1 \\ v_4 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2/3 & 1/3 \\ 1/3 & 2/3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_4 \end{bmatrix}. \quad (\text{E10.7})$$

The rest is easy. Form $\hat{\mathbf{M}} = \mathbf{T}^T \mathbf{M} \mathbf{T}$ and $\hat{\mathbf{K}} = \mathbf{T}^T \mathbf{K} \mathbf{T}$, find the two squared frequencies of the condensed eigensystem and verify that they approximate the two lowest original squared frequencies.

EXERCISE 10.7 [A:20] Two beam elements: 1–2 and 2–3, each of length L and rigidity EI are connected at node 2. The macroelement has 6 degrees of freedom: $\{v_1, \theta_1, v_2, \theta_2, v_3, \theta_3\}$. Eliminate the two DOF of node 2 by condensation. Is the condensed stiffness the same as that of a beam element of length $2L$ and rigidity EI ? (For expressions of the beam stiffness matrices, see Chapter 12.)

11

Variational Formulation of Bar Element

TABLE OF CONTENTS

	Page
§11.1. A New Beginning	11–3
§11.2. Definition of Bar Member	11–3
§11.3. Variational Formulation	11–4
§11.3.1. The Total Potential Energy Functional	11–4
§11.3.2. Admissible Variations	11–6
§11.3.3. The Minimum Total Potential Energy Principle	11–6
§11.3.4. TPE Discretization	11–7
§11.3.5. Bar Element Discretization	11–8
§11.3.6. Interpolation by Shape Functions	11–9
§11.3.7. The Strain-Displacement Matrix	11–9
§11.3.8. *Trial Basis Functions	11–9
§11.4. The Finite Element Equations	11–10
§11.4.1. The Stiffness Matrix	11–10
§11.4.2. The Consistent Node Force Vector	11–11
§11.5. Weak Forms	11–13
§11.5.1. From Strong to Weak	11–13
§11.5.2. Weak Form Based Approximation Example	11–14
§11.5.3. Balanced Weak Forms	11–15
§11.5.4. Principle of Virtual Work as Balanced Weak Form	11–15
§11.5.5. *Weighted Residual Methods	11–16
§11.6. *Accuracy Analysis	11–16
§11.6.1. *Nodal Exactness and Superconvergence	11–16
§11.6.2. *Fourier Patch Analysis	11–17
§11.6.3. *Robin Boundary Conditions	11–18
§11. Notes and Bibliography	11–19
§11. References	11–20
§11. Exercises	11–21

§11.1. A New Beginning

This Chapter begins Part II of the course. This Part focuses on the construction of structural and continuum finite elements using a *variational formulation* based on the Total Potential Energy. Why only elements? Because the other synthesis steps of the DSM: globalization, merge, BC application and solution, remain the same as in Part I. Those operations are not element dependent.

Individual elements are constructed in this Part beginning with the simplest ones and progressing to more complicated ones. The formulation of 2D finite elements from a variational standpoint is discussed in Chapters 14 and following. Although the scope of that formulation is broad, exceeding structural mechanics, it is better understood by going through specific elements first.

From a geometrical standpoint the simplest finite elements are one-dimensional or *line elements*. This means that the *intrinsic dimensionality* is one, although these elements may be used in one, two or three space dimensions upon transformation to global coordinates as appropriate. The simplest one-dimensional structural element is the *two-node bar element*, which we have already encountered in Chapters 2, 3 and 5 as the truss member.

In this Chapter the bar stiffness equations are rederived using the variational formulation. For uniform properties the resulting equations are the same as those found previously using the physical or Mechanics of Materials approach. The variational method has the advantage of being readily extendible to more complicated situations, such as variable cross section or more than two nodes.

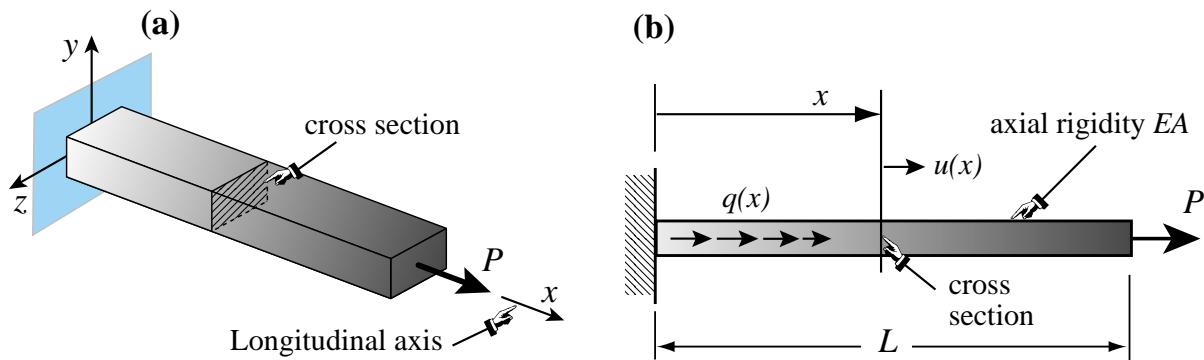


FIGURE 11.1. A fixed-free bar member: (a) 3D view showing reference frame; (b) 2D view on $\{x, y\}$ plane highlighting some quantities that are important in bar analysis.

§11.2. Definition of Bar Member

In structural mechanics a *bar* is a structural component characterized by two properties:

- (1) One preferred dimension: the *longitudinal dimension* or *axial dimension* is much larger than the other two dimensions, which are collectively known as *transverse dimensions*. The intersection of a plane normal to the longitudinal dimension and the bar defines the *cross sections*. The longitudinal dimension defines the *longitudinal axis*. See Figure 11.1(a).
- (2) The bar resists an internal axial force along its longitudinal dimension.

Table 11.1 Nomenclature for Mathematical Model of Axially Loaded Bar

<i>Quantity</i>	<i>Meaning</i>
x	Longitudinal bar axis*
$(.)'$	$d(.) / dx$
$u(x)$	Axial displacement
$q(x)$	Distributed axial force, given per unit of bar length
L	Total bar length
E	Elastic modulus
A	Cross section area; may vary with x
EA	Axial rigidity
$e = du/dx = u'$	Infinitesimal axial strain
$\sigma = Ee = Eu'$	Axial stress
$F = A\sigma = EAe = EAu'$	Internal axial force
P	Prescribed end load

* x is used in this Chapter instead of \bar{x} (as in Chapters 2–3) to simplify the notation.

In addition to trusses, bar elements are used to model cables, chains and ropes. They are also used as fictitious elements in penalty function methods, as discussed in Chapter 9.

We will consider here only *straight bars*, although their cross section may vary. Our one-dimensional mathematical model assumes that the bar material is linearly elastic obeying Hooke's law, and that displacements and strains are infinitesimal. Figure 11.1(b) pictures some relevant quantities for a fixed-free bar. Table 11.1 collects the necessary terminology for the governing equations.

Figure 11.2 displays the governing equations of the bar in a graphic format called a *Tonti diagram*. The formal similarity with the diagrams used in Chapter 5 to explain MoM elements should be noted, although the diagram of Figure 11.2 pertains to the continuum bar model rather than to the discrete one. (The qualifier “strong form” is explained in the next Chapter.)

§11.3. Variational Formulation

To illustrate the variational formulation, the finite element equations of the bar will be derived from the Minimum Potential Energy principle.

§11.3.1. The Total Potential Energy Functional

In Mechanics of Materials it is shown that the *internal energy density* at a point of a linear-elastic material subjected to a one-dimensional state of stress σ and strain e is $\mathcal{U} = \frac{1}{2}\sigma(x)e(x)$, where σ is to be regarded as linked to the displacement u through Hooke's law $\sigma = Ee$ and the strain-displacement relation $e = u' = du/dx$. This \mathcal{U} is also called the *strain energy density*. Integration over the volume of the bar gives the total internal energy

$$U = \frac{1}{2} \int_V \sigma e dV = \frac{1}{2} \int_0^L F e dx = \frac{1}{2} \int_0^L (EAu')u' dx = \frac{1}{2} \int_0^L u' EA u' dx. \quad (11.1)$$

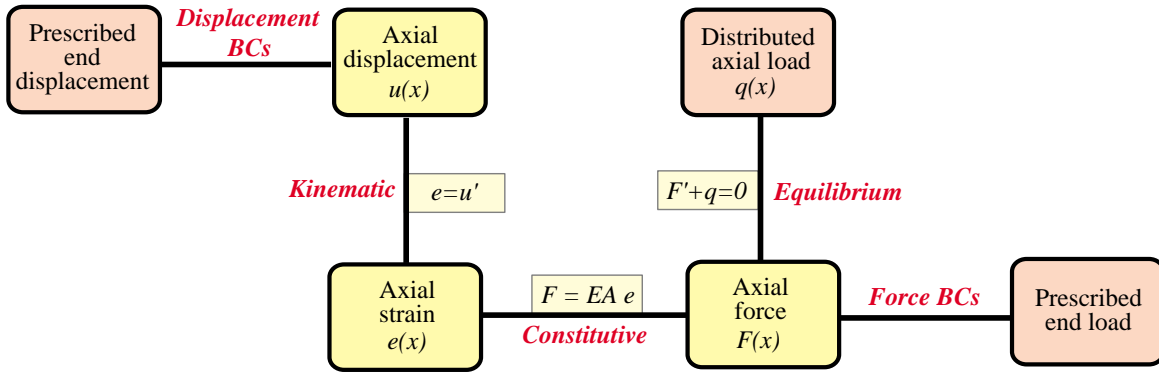


FIGURE 11.2. Strong-form Tonti diagram for the continuum model of a bar member. Field equations and BCs are represented as lines connecting the boxes. Yellow (brown) boxes contain unknown (given) quantities.

All integrand quantities in (11.1) may depend on x .

The *external work potential* is the work performed by applied mechanical loads working on the bar displacements. This potential is denoted by W . (The *external energy* V is the negative of the work potential: $V = -W$. In the ensuing derivations W will be used instead of V .) It collects contributions from two sources:

1. The distributed load $q(x)$. This contributes a cross-section density of $q(x)u(x)$ because q is assumed to be already integrated over the section.
2. Any specified axial point load(s). For the fixed-free example of Figure 11.1 the end load P would contribute $P u(L)$.

The second source may be folded into the first by conventionally writing any point load P acting at a cross section $x = a$ as a contribution $P \delta(a)$ to $q(x)$, in which $\delta(a)$ denotes the one-dimensional Dirac delta function at $x = a$. If this is done the external energy can be concisely expressed as

$$W = \int_0^L q u \, dx. \quad (11.2)$$

The total potential energy of the bar is given by

$$\boxed{\Pi = U - W} \quad (11.3)$$

Mathematically Π is a *functional*, called the *Total Potential Energy* functional or TPE. It depends only on the axial displacement $u(x)$. In Variational Calculus $u(x)$ is called the *primary variable* of the functional. When the dependence of Π on u needs to be emphasized we shall write $\Pi[u] = U[u] - W[u]$, with brackets enclosing the primary variable. To display both primary and independent variables we write, for example, $\Pi[u(x)] = U[u(x)] - W[u(x)]$.

Remark 11.1. According to the rules of Variational Calculus, the Euler-Lagrange equation for Π is

$$\mathcal{E} = \frac{\partial \Pi}{\partial u} - \frac{d}{dx} \frac{\partial \Pi}{\partial u'} = -q - (EA u')' \quad (11.4)$$

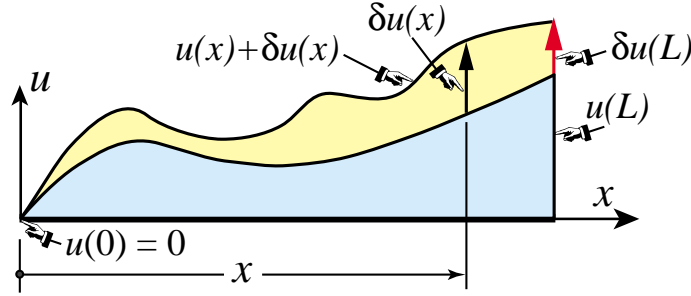


FIGURE 11.3. Concept of admissible variation of the axial displacement function $u(x)$. For convenience $u(x)$ is plotted normal to the longitudinal axis. Both $u(x)$ and $u(x) + \delta u(x)$ shown above are kinematically admissible, and so is the variation $\delta u(x)$. Note that the variation $\delta u(L)$ is not zero because the BC at $x = L$ is natural.

The stationary condition for Π is $\mathcal{E} = 0$, or

$$(EA u')' + q = 0 \quad (11.5)$$

This is the strong (pointwise) equation of equilibrium in terms of the axial displacement, which reduces to $EA u'' + q = 0$ if EA is constant. This equation is not explicitly used in the FEM development. It is instead replaced by $\delta \Pi = 0$, with the variation restricted over the class of finite element interpolation functions.

§11.3.2. Admissible Variations

The concept of *admissible variation* is fundamental in both variational calculus and the variationally formulated FEM. *Only the primary variable(s) of a functional may be varied.* For the TPE functional (11.3) this is the axial displacement $u(x)$. Suppose that $u(x)$ is changed to $u(x) + \delta u(x)$.¹ This is illustrated in Figure 11.3, where for convenience $u(x)$ is plotted normal to x . The TPE functional changes accordingly as

$$\Pi = \Pi[u] \Rightarrow \Pi + \delta \Pi = \Pi[u + \delta u]. \quad (11.6)$$

The function $\delta u(x)$ and the scalar $\delta \Pi$ are called the *variations* of $u(x)$ and Π , respectively. The variation $\delta u(x)$ should not be confused with the ordinary differential $du(x) = u'(x) dx$ since on taking the variation the independent variable x is *frozen*; that is, $\delta x = 0$.

A displacement variation $\delta u(x)$ is said to be *admissible* when both $u(x)$ and $u(x) + \delta u(x)$ are *kinematically admissible* in the sense of the Principle of Virtual Work (PVW). This agrees with the conditions of classical variational calculus, and are restated next.

A *kinematically admissible* axial displacement $u(x)$ obeys two conditions:

- (i) It is continuous over the bar length, that is, $u(x) \in C^0$ in $x \in [0, L]$.
- (ii) It satisfies exactly any displacement boundary condition, such as the fixed-end specification $u(0) = 0$ of Figure 11.1. See of Figure 11.3.

The variation $\delta u(x)$ pictured in Figure 11.3 is kinematically admissible because both $u(x)$ and $u(x) + \delta u(x)$ satisfy the foregoing conditions. Note that the variation $\delta u(L)$ at the free end $x = L$ is not necessarily zero because that boundary condition is natural; that is, not specified directly in terms of the displacement $u(L)$. On the other hand, $\delta(0) = 0$.

The physical meaning of conditions (i)–(ii) is the subject of Exercise 11.1.

¹ The symbol δ not immediately followed by a parenthesis is not a delta function but instead denotes variation with respect to the variable that follows.

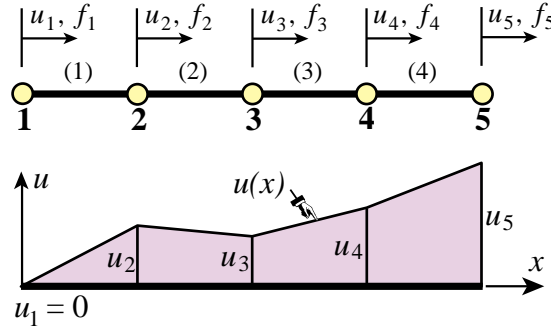


FIGURE 11.4. FEM discretization of bar member. A piecewise-linear admissible displacement trial function $u(x)$ is drawn underneath the mesh. It is assumed that the left end is fixed; thus $u_1 = 0$.

§11.3.3. The Minimum Total Potential Energy Principle

The Minimum Total Potential Energy (MTPE) principle states that the actual displacement solution $u^*(x)$ that satisfies the governing equations is that which renders Π stationary:²

$$\delta \Pi = \delta U - \delta W = 0 \quad \text{iff} \quad u = u^* \quad (11.7)$$

with respect to *admissible* variations $u = u^* + \delta u$ of the exact displacement field $u^*(x)$.

Remark 11.2. Using standard techniques of variational calculus³ it can be shown that if $EA > 0$ and kinematic boundary conditions weed out any rigid motions, the solution $u^*(x)$ of (11.7) exists, is unique, and renders $\Pi[u]$ a minimum over the class of kinematically admissible displacements. The last attribute explains the “minimum” in the name of the principle.

§11.3.4. TPE Discretization

To apply the TPE functional (11.3) to the derivation of FEM equations we replace the continuum mathematical model by a discrete one consisting of a union of bar elements. For example, Figure 11.4 illustrates the subdivision of a fixed-free bar member into four two-node elements.

Functionals are scalars. Therefore, for a discretization such as that shown in Figure 11.4, the TPE functional (11.3) may be decomposed into a sum of contributions of individual elements:

$$\Pi = \Pi^{(1)} + \Pi^{(2)} + \dots + \Pi^{(N_e)} \quad (11.8)$$

in which N_e denotes the number of elements. The same decomposition applies to both its internal energy and external work potential components:

$$\delta U = \delta U^{(1)} + \dots + \delta U^{(N_e)} = 0, \quad \delta W = \delta W^{(1)} + \dots + \delta W^{(N_e)} = 0, \quad (11.9)$$

as well as to the stationarity condition (11.7):

$$\delta \Pi = \delta \Pi^{(1)} + \delta \Pi^{(2)} + \dots + \delta \Pi^{(N_e)} = 0. \quad (11.10)$$

² The symbol “iff” in (11.7) is an abbreviation for “if and only if”.

³ See references in **Notes and Bibliography** at the end of Chapter.

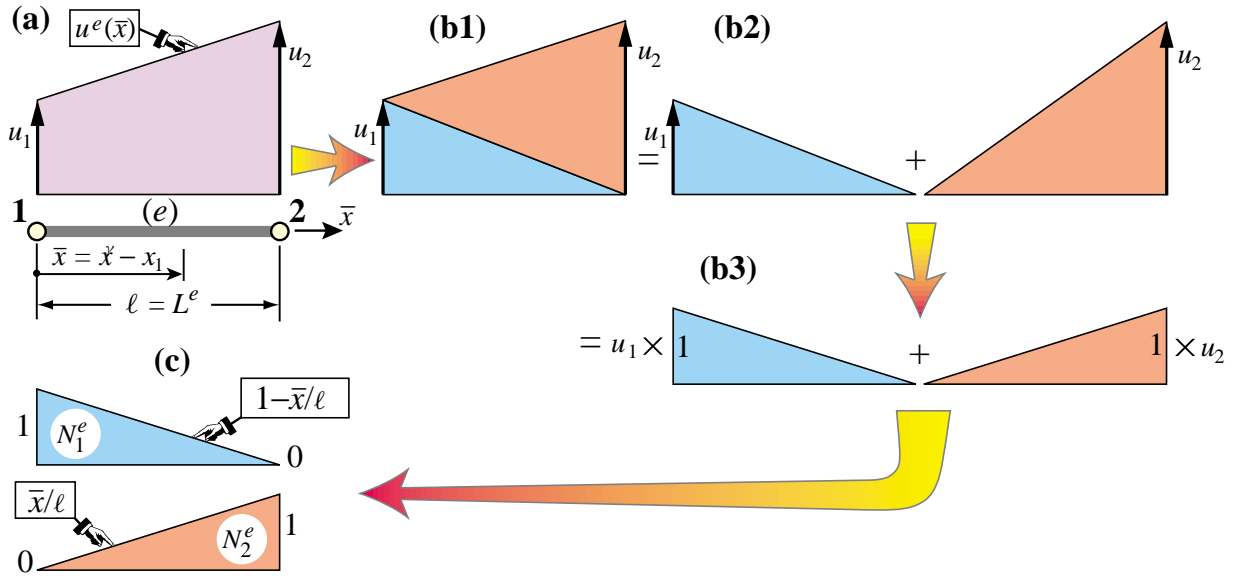


FIGURE 11.5. A two-node, TPE-based bar element: (a) element configuration and axial displacement variation (plotted normal to element axis for better visibility); (b1,b2,b3) displacement interpolation expressed in terms of linear shape functions; (c) element shape functions.

Using the fundamental lemma of variational calculus,⁴ it can be shown that (11.10) implies that for a generic element e we may write

$$\delta \Pi^e = \delta U^e - \delta W^e = 0. \quad (11.11)$$

This *variational equation* is the basis for the derivation of element stiffness equations once the displacement field has been discretized over the element, as described next.

Remark 11.3. In mathematics (11.11) is called a *first variation form*. It is a special case of a more general expression called a *weak form*, which is covered in more detail later. In mechanics it states the *Principle of Virtual Work* or PVW for each element: $\delta U^e = \delta W^e$, which says that the virtual work of internal and external forces on admissible displacement variations is equal if the element is in equilibrium [567].

§11.3.5. Bar Element Discretization

Figure 11.5(a) depicts a generic bar element e . It has two nodes, which are labeled 1 and 2. These are called the *local node numbers*.⁵ The element is referred to its local axis $\bar{x} = x - x_1$, which measures the distance from its left end. The two degrees of freedom are u_1^e and u_2^e . (Bars are not necessary since the directions of \bar{x} and x are the same.) The element length is $\ell = L^e$.

The mathematical concept of bar finite elements is based on *approximating* axial displacement $u(x)$ over the element. The exact displacement u^* is replaced by an approximate displacement

$$u^*(x) \approx u^e(x) \quad (11.12)$$

⁴ See, e.g., Chapter II of Gelfand and Fomin [289].

⁵ Note the notational change from the labels i and j of Part I. This will facilitate transition to multidimensional elements in Chapters 14ff.

over the finite element mesh. This approximate displacement, $u^e(x)$, taken over all elements $e = 1, 2, \dots, N^e$, is called the *finite element trial expansion* or simply *trial expansion*. See Figure 11.4. This FE trial expansion must belong to the class of kinematically admissible displacements defined in §. Consequently, it must be C^0 continuous over and between elements. The most common choices for u^e are polynomials in x , as in the development that follows.

§11.3.6. Interpolation by Shape Functions

In a two-node bar element the only possible polynomial choice of the displacement u^e that satisfies the interelement continuity requirement is *linear*. It can be expressed by the following interpolation formula, which is graphically developed in Figure 11.5(b1,b2,b3):

$$u^e(x) = N_1^e u_1^e + N_2^e u_2^e = [N_1^e \quad N_2^e] \begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix} = \mathbf{N}^e \mathbf{u}^e. \quad (11.13)$$

The functions N_1^e and N_2^e that multiply the node displacements u_1 and u_2 are called *shape functions*, while \mathbf{N} is called the *shape function matrix*. In this case \mathbf{N}^e reduces to a row vector.

The shape functions *interpolate* the internal displacement u^e directly from the node values. They are pictured in Figure 11.5(c). For this element, with $\bar{x} = x - x_1$ measuring the axial distance from the left node i , the shape functions are

$$N_1^e = 1 - \frac{\bar{x}}{\ell} = 1 - \zeta, \quad N_2^e = \frac{\bar{x}}{\ell} = \zeta. \quad (11.14)$$

Here

$$\zeta = \frac{x - x_1}{\ell} = \frac{\bar{x}}{\ell}, \quad (11.15)$$

is a dimensionless coordinate, also known as a *natural coordinate*, that varies from 0 through 1 over the element. Note that $dx = \ell d\zeta$ and $d\zeta = dx/\ell$. The shape function N_1^e has the value 1 at node 1 and 0 at node 2. Conversely, shape function N_2^e has the value 0 at node 1 and 1 at node 2. This is a general property of shape functions. It follows from the fact that element displacement interpolations such as (11.13) are based on physical node values.

§11.3.7. The Strain-Displacement Matrix

The axial strain associated with the trial function u^e is

$$e = \frac{du^e}{dx} = (u^e)' = \left[\frac{dN_1^e}{dx} \quad \frac{dN_2^e}{dx} \right] \begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix} = \frac{1}{\ell} [-1 \quad 1] \begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix} = \mathbf{B} \mathbf{u}^e, \quad (11.16)$$

in which

$$\mathbf{B} = \frac{1}{\ell} [-1 \quad 1], \quad (11.17)$$

is called the *strain-displacement* matrix. Note that \mathbf{B} is constant over the element.

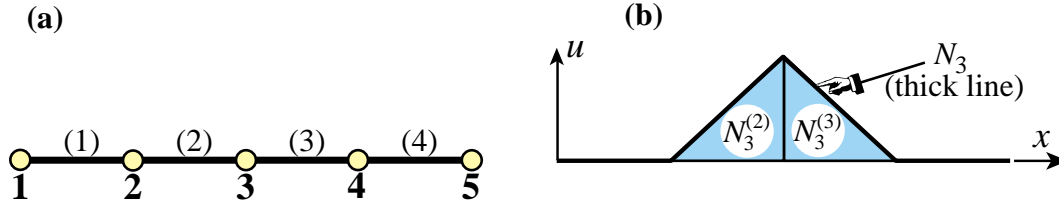


FIGURE 11.6. Trial basis function (a.k.a. hat function) for node 3 of a four-element bar discretization.

§11.3.8. *Trial Basis Functions

Shape functions are associated with elements. A *trial basis function*, or simply *basis function*, is associated with a node. Suppose node i of a bar discretization connects elements $(e1)$ and $(e2)$. The trial basis function N_i is defined as

$$N_i(x) = \begin{cases} N_i^{(e1)} & \text{if } x \in \text{element } (e1) \\ N_i^{(e2)} & \text{if } x \in \text{element } (e2) \\ 0 & \text{otherwise} \end{cases} \quad (11.18)$$

For a piecewise linear discretization, such as that used in the two-node bar, this function has the shape of a hat. Thus it is also called a *hat function* or *chapeau function*. See Figure 11.6, in which $i = 3$, $e1 = 2$, and $e2 = 3$. The concept is important in the variational interpretation of FEM as a Rayleigh-Ritz method.

Remark 11.4. In addition to continuity, shape and trial functions must satisfy a *completeness* requirement with respect to the governing variational principle. This condition is stated and discussed in later Chapters. Suffices for now to say that the shape functions (11.14), as well as the associated trial functions, do satisfy this requirement.

§11.4. The Finite Element Equations

In linear FEM the discretization process based on the TPE functional leads to the following algebraic form in the node displacements

$$\Pi^e = U^e - W^e, \quad \text{in which} \quad U^e \stackrel{\text{def}}{=} \frac{1}{2}(\mathbf{u}^e)^T \mathbf{K}^e \mathbf{u}^e \quad \text{and} \quad W^e \stackrel{\text{def}}{=} (\mathbf{u}^e)^T \mathbf{f}^e. \quad (11.19)$$

Here \mathbf{K}^e and \mathbf{f}^e are called the *element stiffness matrix* and the *element consistent nodal force vector*, respectively. The three scalars Π^e , U^e and W^e are only function of the node displacements \mathbf{u}^e . (This is a consequence of displacements being the only primary variable of the TPE functional.) Note that U^e and W^e depend *quadratically* and *linearly*, respectively, on those displacements. Taking the variation of Π^e with respect to the node displacements gives⁶

$$\delta \Pi^e = (\delta \mathbf{u}^e)^T \frac{\partial \Pi^e}{\partial \mathbf{u}^e} = (\delta \mathbf{u}^e)^T [\mathbf{K}^e \mathbf{u}^e - \mathbf{f}^e] = 0. \quad (11.20)$$

Because the variations $\delta \mathbf{u}^e$ can be arbitrary, the bracketed expression must vanish, which yields

$$\boxed{\mathbf{K}^e \mathbf{u}^e = \mathbf{f}^e.} \quad (11.21)$$

These are the familiar element stiffness equations. Hence the foregoing names given to \mathbf{K}^e and \mathbf{f}^e are justified *a posteriori*.

⁶ The $\frac{1}{2}$ factor disappears on taking the variation because U^e is quadratic in the node displacements. For a review on the calculus of discrete quadratic forms, see Appendix D.

§11.4.1. The Stiffness Matrix

We now apply the foregoing expressions to the two-node bar element. Its internal energy U^e is

$$U^e = \frac{1}{2} \int_{x_1}^{x_2} e EA e dx = \frac{1}{2} \int_0^1 e EA e \ell d\zeta. \quad (11.22)$$

Note that the integration variable x has been changed to the natural coordinate ζ defined in (11.15) that varies from 0 through 1, whence $dx = \ell d\zeta$. This form is symmetrically expanded using the strain-displacement matrix relation (11.16), by inserting $e = e^T = (\mathbf{u}^e)^T \mathbf{B}^T$ and $e = \mathbf{B} \mathbf{u}^e$ into the first and second e of (11.22), respectively, to get

$$U^e = \frac{1}{2} \int_0^1 (\mathbf{u}^e)^T \mathbf{B}^T EA \mathbf{B} \mathbf{u}^e \ell d\zeta = \frac{1}{2} \int_0^1 [u_1^e \quad u_2^e] \frac{1}{\ell} \begin{bmatrix} -1 \\ 1 \end{bmatrix} EA \frac{1}{\ell} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix} \ell d\zeta. \quad (11.23)$$

The nodal displacements do not depend on position and can be moved out of the integral. Also $\mathbf{B}^T EA \mathbf{B} = EA \mathbf{B}^T \mathbf{B}$ since EA is a scalar:

$$U^e = \frac{1}{2} (\mathbf{u}^e)^T \int_0^1 EA \mathbf{B}^T \mathbf{B} \ell d\zeta \mathbf{u}^e = \frac{1}{2} [u_1^e \quad u_2^e] \int_0^1 \frac{EA}{\ell^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \ell d\zeta \begin{bmatrix} u_1^e \\ u_2^e \end{bmatrix}. \quad (11.24)$$

By (11.19) this is expressible as $\frac{1}{2} (\mathbf{u}^e)^T \mathbf{K}^e \mathbf{u}^e$. Since \mathbf{u}^e is arbitrary, \mathbf{K}^e is extracted as

$$\boxed{\mathbf{K}^e = \int_0^1 EA \mathbf{B}^T \mathbf{B} \ell d\zeta = \int_0^1 \frac{EA}{\ell^2} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \ell d\zeta = \frac{1}{\ell} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \int_0^1 EA d\zeta.} \quad (11.25)$$

This is the bar element stiffness matrix. For a homogeneous and prismatic bar of constant rigidity, EA can be moved outside the integral, $\int_0^1 d\zeta = 1$ and (11.25) collapses to

$$\boxed{\mathbf{K}^e = \frac{EA}{\ell} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.} \quad (11.26)$$

This is the same element stiffness matrix of the prismatic truss member derived in Chapters 2 and 5 by a Mechanics of Materials approach, but now obtained through a variational argument.

§11.4.2. The Consistent Node Force Vector

The *consistent node force vector* \mathbf{f}^e defined in (11.19) comes from the element contribution to the external work potential W :

$$W^e = \int_{x_1}^{x_2} q u dx = \int_0^1 q \mathbf{N}^T \mathbf{u}^e \ell d\zeta = (\mathbf{u}^e)^T \int_0^1 q \begin{bmatrix} 1 - \zeta \\ \zeta \end{bmatrix} \ell d\zeta \stackrel{\text{def}}{=} (\mathbf{u}^e)^T \mathbf{f}^e, \quad (11.27)$$

Since \mathbf{u}^e is arbitrary,

$$\boxed{\mathbf{f}^e = \int_{x_1}^{x_2} q \begin{bmatrix} 1 - \zeta \\ \zeta \end{bmatrix} dx = \int_0^1 q \begin{bmatrix} 1 - \zeta \\ \zeta \end{bmatrix} \ell d\zeta.} \quad (11.28)$$

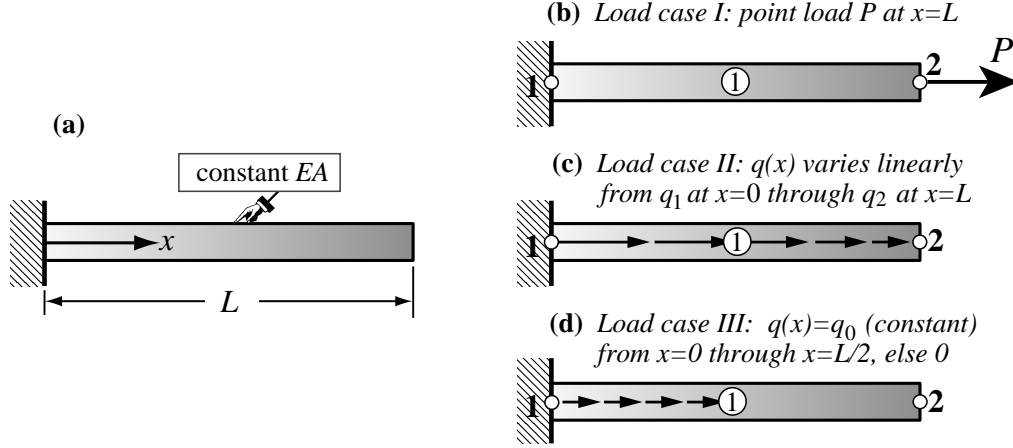


FIGURE 11.7. Fixed-free, prismatic bar example: (a) configuration; (b,c,d) FEM discretization and load cases.

in which ζ is defined by (11.15). If q is constant over the element, it may be taken out of the integral:

$$\mathbf{f}^e = q \int_0^1 \begin{bmatrix} 1 - \zeta \\ \zeta \end{bmatrix} \ell d\zeta. \quad (11.29)$$

This gives the same results as with the EbE lumping method of Chapter 7. See Exercise 11.3.

Example 11.1. The two-node bar element is tested on the benchmark problem defined in Figure 11.7. A fixed-free, homogeneous, prismatic bar of length L , elastic modulus E and cross section area A has the configuration illustrated in Figure 11.7(a). It is discretized with a *single* element as shown in Figure 11.7(b,c,d), and subjected to the three load cases pictured there. Case I involves a point load P at the free end, which may be formally represented as

$$q^I(x) = P \delta(L) \quad (11.30)$$

where $\delta()$ denotes the delta function with argument x .

Case II involves a distributed axial load that varies linearly from $q_1 = q(0)$ at the fixed end through $q_2 = q(L)$ at the free end:

$$q^{II}(x) = q_1(1 - \zeta) + q_2\zeta, \quad (11.31)$$

in which $\zeta = 1 - x/L$. Case III involves a “box” distributed load $q(x)$ that is constant and equal to q_0 from the fixed end $x = 0$ through midspan $x = L/2$, and zero otherwise:

$$q^{III}(x) = q_0 \left(H(x) - H(x - \tfrac{1}{2}L) \right), \quad (11.32)$$

in which $H()$ denotes the Heaviside unit step function with argument x . The master stiffness equations constructed using the prismatic stiffness matrix (11.26) with $\ell = L$ and $\bar{x} \rightarrow x$ are

$$\frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_1^m \\ u_2^m \end{bmatrix} = \begin{bmatrix} f_1^m \\ f_2^m \end{bmatrix} = \mathbf{f}^m. \quad (11.33)$$

Here superscript m identifies the load case. The consistent node forces computed from (11.28) with $\ell = L$ and $\bar{x} \rightarrow x$ are

$$\mathbf{f}^I = \begin{bmatrix} 0 \\ P \end{bmatrix}, \quad \mathbf{f}^{II} = \frac{L}{6} \begin{bmatrix} 2q_1 + q_2 \\ q_1 + 2q_2 \end{bmatrix}, \quad \mathbf{f}^{III} = \frac{q_0 L}{8} \begin{bmatrix} 3 \\ 1 \end{bmatrix}. \quad (11.34)$$

On applying the fixed end support condition $u_1^m = 0$ and solving for u_2^m , the free end deflections are

$$u_2^I = \frac{PL}{EA}, \quad u_2^{II} = \frac{(q_1 + 2q_2)L^2}{6EA}, \quad u_2^{III} = \frac{q_0 L^2}{8EA}. \quad (11.35)$$

The analytical solutions for $u(x)$, obtained on integrating the ODE $EAu'' + q = 0$ with boundary conditions $u(0) = 0$, $F(L) = EAu'(L) = P$ for case I and $F(L) = EAu'(L) = 0$ for cases II and III, are

$$u^I(x) = \frac{Px}{EA}, \quad u^{II}(x) = \frac{x[3(q_1 + q_2)L^2 - 3q_1Lx + (q_1 - q_2)x^2]}{6EA}, \quad u^{III}(x) = \frac{q_0}{2EA} \left(Lx - x^2 + \langle x - \frac{1}{2}L \rangle^2 \right). \quad (11.36)$$

In the expression of $u^{III}(x)$, $\langle x - \frac{1}{2}L \rangle^2$ means $(x - \frac{1}{2}L)^2$ if $x \geq \frac{1}{2}L$, else zero (Macauley's bracket notation for discontinuity functions). Evaluating (11.36) at $x = L$ and comparing to (11.35), one finds that the three computed end deflections are *exact*.

For case I this agreement is no surprise: the exact $u^I(x)$ is linear in x , which is contained in the span of the linear shape functions. But for II and III this is far from obvious since the exact solutions are cubic and piecewise quadratic, respectively, in x . The fact that the exact solution is verified at the free end node is an instance of the *nodal exactness* property discussed in §11.6.1.

Note that in cases II and III the FEM displacement solutions *inside* the element, which vary linearly, will not agree pointwise with the exact solutions, which do not. For example the exact midspan displacement is $u^{III}(\frac{1}{2}L) = q_0L^2/(8EA) = u^{III}(L)$, whereas the FEM interpolation would give $q_0L^2/(16EA)$ there, in error by 100%. To reduce such internal discrepancies the member may be divided into more elements.

§11.5. Weak Forms

Weak forms are expressions notoriously difficult to explain to newcomers. They occupy an intermediate position between differential equations and functionals. There are so many variants and procedural kinks, however, that their position in the mathematical food chain is fuzzy. Confusion is compounded by the use of disparate terminology, some generic, some application oriented. To shed some sunlight into this murky swamp, we go through a specific example: the bar member.

§11.5.1. From Strong to Weak

The governing differential equation for a bar member in terms of axial displacements is $(EAu'(x))' + q(x) = 0$, or $EAu''(x) + q(x) = 0$ if the rigidity EA is constant. Replace the zero by $r(x)$, which stands for *residual*, and move it to the left-hand side:

$$r(x) = (EAu'(x))' + q(x), \quad \text{or if } EA \text{ is constant: } r(x) = EAu''(x) + q. \quad (11.37)$$

The governing ODE may be compactly stated as $r(x) = 0$. This must hold *at each point* over the member span, say $x \in [0, L]$. Hence the term *strong form* (SF) used for this kind of mathematical model. No ambiguity so far. But suppose that insisting on $r(x) = 0$ everywhere is too demanding. We would like to relax that condition so it is satisfied only in an *average sense*. To accomplish that, multiply the residual by a function $v(x)$, integrate over the problem domain, and set the result to zero:

$$J = \int_0^L r(x) v(x) dx = 0. \quad (11.38)$$

Here $v(x)$ is supposed to be sufficiently well behaved for the integral to exist. Ignoring boundary conditions for now, (11.38) is called a *weak form*, which is often abbreviated to WF in the sequel.

Function $v(x)$ receives two names in the literature: *test function* in a general mathematical context, and *weight function* (also *weighting function*) in the context of approximation methods based on weak forms. In what follows both terms will be used within the appropriate context.

§11.5.2. Weak Form Based Approximation Example

To show how weak forms can be used to generate approximate solutions, consider again a fixed-free, prismatic, homogeneous bar member (that is, EA is constant), under uniform load $q(x) = q_0$ along its length and zero load at the free end. The WF (11.38) becomes

$$J = \int_0^L (EA u''(x) + q_0) v(x) dx = 0. \quad (11.39)$$

subject to the end conditions

$$u(0) = 0, \quad F(L) = EA u'(L) = 0. \quad (11.40)$$

We will restrict both $u(x)$ and $v(x)$ to be *quadratic* polynomials:

$$u(x) = a_0 + a_1 x + a_2 x^2, \quad v(x) = b_0 + b_1 x + b_2 x^2. \quad (11.41)$$

in which a_i and b_i are numerical coefficients, real in this case. Once assumptions such as those in (11.41) are made, more terminology kicks in. The assumed $u(x)$ is now called a *trial function*, which is spanned by the linear-space basis $\{1, x, x^2\}$ of dimension 3. The assumed $v(x)$ is called a *weight function*, which is spanned by exactly the same basis. There is a special name for the scenario when the trial and weight function bases coalesce: the *Galerkin method*.⁷ We will call the end result a *Galerkin solution*. Replacing (11.41) into (11.39) we get

$$J = \frac{L}{6} (6b_0 + 3b_1 L + 2b_2 L^2) (2EA a_2 + q_0). \quad (11.42)$$

Now J must vanish for any arbitrary value of $\{b_0, b_1, b_2\}$. On extracting the expressions that multiply those coefficients we obtain the same equation thrice: $2EA a_2 + q_0 = 0$. Thus $a_2 = -q_0/(2EA)$, whereas a_0 and a_1 remain arbitrary. Consequently the Galerkin solution *before* BC is

$$u(x) = a_0 + a_1 x - \frac{q_0}{2EA} x^2. \quad (11.43)$$

ODE aficionados would recognize this as the general solution of $EAu'' + q_0 = 0$ so Uncle Boris has done the job. Applying the end conditions (11.40) gives $a_0 = 0$ and $a_1 = q_0/(EA)$ whence the final solution is

$$u(x) = \frac{q_0}{2EA} x(2L - x). \quad (11.44)$$

Replacing into (11.37) and (11.40) it may be verified that this is the exact analytical solution.

Instead of applying the end conditions *a posteriori* we may try to incorporate them *a priori* into the trial function assumption. On enforcing (11.40) into the assumed $u(x)$ of (11.41) we find that $a_0 = 0$ and $a_1 = -2a_2 L$. The trial function becomes

$$u(x) = a_2 x(x - 2L), \quad (11.45)$$

⁷ Introduced by Boris Galerkin in 1912. For a brief account of the general methodology, see **Notes and Bibliography**

and only one free coefficient remains. Accordingly only one weight basis function is needed: either 1, x or x^2 does the job, and the exact solution (11.44) is obtained again.⁸

What happens if the load $q(x)$ varies, say, linearly and the same quadratic polynomial assumptions (11.41) are used? Then Galerkin goes gaga. See Exercise 11.8.

Even for this trivial example, several procedural choices are apparent. If we allow the trial and weight function spaces to differ, volatility zooms up. Furthermore, we can apply transformations to the residual integral as done in the next subsection. Compared to the well ordered world of variational-based FEM, confusion reigns supreme.

§11.5.3. Balanced Weak Forms

Some method in the madness can be injected by balancing. A look at (11.39) reveals an unpleasant asymmetry. Second derivatives of $u(x)$ appear, but none of $v(x)$. This places unequal restrictions on smoothness of the trial and test function spaces. Integration by parts restores derivative order balance. Replacing $\int_0^L EA u'' v dx = -\int_0^L EA u' v' dx + (EA u')v|_0^L$ and rearranging terms yields

$$J = \int_0^L EA u'(x) v'(x) dx - \int_0^L q(x) v(x) dx - (EA u'(x)) v(x)|_0^L. \quad (11.46)$$

This will be called a *balanced-derivative weak form*, or simply a *balanced weak form* (BWF). It displays obvious advantages: (i) same smoothness requirements for assumed u and v , and (ii) end BC appear explicitly in the non-integral term, neatly factored into essential and natural. A minor flaw is that the original residual is no longer clearly visible.

For a bar with variable axial rigidity replace $EA u''$ by $(EA u')'$ in the first integrand.

On repeating the Galerkin procedure of the previous subsection with the assumptions (11.41) one finds an identical J , as may be expected, and the same final solution. Again one has the choice of pre- or post-imposing the end conditions (11.40). Generally the latter choice is far more convenient in a computer implementation.

§11.5.4. Principle of Virtual Work as Balanced Weak Form

There is a close relationship between the BWF (11.46) and one of the fundamental tools of Analytical Mechanics: the Principle of Virtual Work (PVW). To exhibit it, set the test function to be an admissible variation of $u(x)$: $v(x) = \delta u(x)$, in which $\delta u(x)$ strongly satisfies all essential BC. Then assume that J is the first variation of a functional Π :

$$J = \int_0^L EA u'(x) \delta u'(x) dx - \int_0^L q(x) \delta u(x) dx - (EA u'(x)) \delta u(x)|_0^L \stackrel{\text{def}}{=} \delta \Pi. \quad (11.47)$$

Indeed this is the first variation of the TPE functional:

$$\Pi = U - W = \frac{1}{2} \int_0^L u'(x) EA u'(x) dx - \int_0^L q(x) u(x) dx \quad (11.48)$$

⁸ Some early works covering weighted residual methods, for example Crandall [155], proclaim that the trial function must satisfy *all* BC *ab initio*. Later ones, e.g., [252,253], relax that rule to BC of essential type (in Galerkin methods, this rule applies to both trial and test functions since the spaces coalesce). In practice this rule can be often relaxed further, as in the example of §11.5.2, applying essential BCs at the last moment.

Hence $J = 0$ is the same as $\delta\Pi = 0$ or $\delta U = \delta W$, which is the PVW for an elastic bar member. This relationship can be used to prove an important property: *Galerkin method is equivalent to a variational formulation* if the residual is the Euler-Lagrange equation of a functional.

Remark 11.5. Where does the boundary term $(EA u'(x)) \delta u(x) \Big|_0^L$ in (11.47) go? Actually, into δW . This immersion is a bit tricky, and depends on redefining $q(x)$ to include prescribed end point forces such as $N(L) = EA u'(L) = P$ through delta functions. This is the subject of Exercise 11.9.

§11.5.5. *Weighted Residual Methods

Galerkin method is widely used in computational mechanics, but does not exhaust all possibilities of using a weak form as source for obtaining numerical solutions. The main generalization consist of allowing trial and test (weight) functions to be different. This leads to a rich class of approximation methods unified under the name *Method of Weighted Residuals* or MWR.

The key idea is as follows. Both $u(x)$ and $v(x)$ are restricted to belong to linear function spaces of *finite dimension* N_u and N_v . These are the *trial function space* and the *test function space*, respectively. which are spanned by basis functions $\phi_i(x)$ and $\psi_i(x)$, respectively:

$$u(x) = \text{span}\{\phi_i(x), 1 \leq i \leq N_u\}, \quad v(x) = \text{span}\{\psi_i(x), 1 \leq i \leq N_v\} \quad (11.49)$$

in which usually $N_u = N_v$. Since the spaces are linear, any $u(x)$ and $v(x)$ can be represented as linear combination of the basis functions:

$$u(x) = \sum_{i=1}^{N_u} a_i \phi_i(x), \quad v(x) = \sum_{i=1}^{N_v} b_i \psi_i(x). \quad (11.50)$$

Here a_i and b_i are scalar coefficients, which may be real or complex depending on the nature of the problem. Insert these into the weak form, perform the necessary integrations, and extract the N_v expressions that are coefficients of the b_i . Solve these equations for the coefficients a_i , and replace in the first of (11.50) to get the approximate solution $u(x)$.

The MWR methodology is of course not restricted to one space dimension. It also extends to time-dependent problems. It can be merged smoothly with the FEM concept of piecewise approximation using shape functions. Some references are provided under **Notes and Bibliography**.

§11.6. *Accuracy Analysis

Low order 1D elements may give surprisingly high accuracy. In particular the lowly two-node bar element can display infinite accuracy under some conditions. This phenomenon is studied in this advanced section as it provides an introduction to modified equation methods and Fourier analysis along the way.

§11.6.1. *Nodal Exactness and Superconvergence

Suppose that the following two conditions are satisfied:

1. The bar properties are constant along the length (prismatic member).
2. The distributed load $q(x)$ is zero between nodes. The only applied loads are point forces at the nodes.

If so, a linear axial displacement $u(x)$ as defined by (11.13) and (11.14) is the exact solution over each element since constant strain and stress satisfy, element by element, all of the governing equations listed in Figure 11.2.⁹

⁹ The internal equilibrium equation $p' + q = EA u'' + q = 0$ is trivially verified because $q = 0$ from the second assumption, and $u'' = 0$ because of shape function linearity.

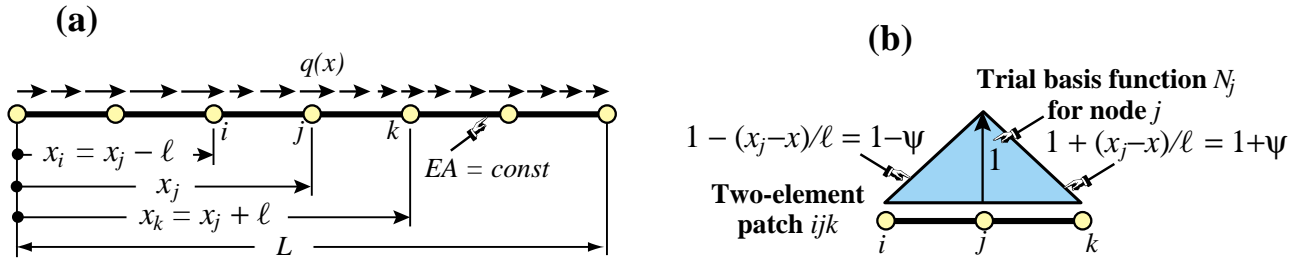


FIGURE 11.8. Superconvergence patch analysis: (a) lattice of bar elements; (b) two element patch.

It follows that if the foregoing conditions are verified the FEM solution is *exact*; that is, it agrees with the analytical solution of the mathematical model.¹⁰ Adding extra elements and nodes would not change the solution. That is the reason behind the truss discretizations used in Chapters 2–3: *one element per member is enough* if they are prismatic and loads are applied to joints. Such models are called *nodally exact*.

What happens if the foregoing assumptions are not met? Exactness is then generally lost, and several elements per member may be beneficial if spurious mechanisms are avoided.¹¹ For a 1D lattice of equal-length, prismatic two-node bar elements, an interesting and more difficult result is: *the solution is nodally exact for any loading if consistent node forces are used*. This is proven in the subsection below. This result underlies the importance of computing node forces correctly.

If conditions such as equal-length are relaxed, the solution is no longer nodally exact but convergence at the nodes is extremely rapid (faster than could be expected by standard error analysis) as long as consistent node forces are used. This phenomenon is called *superconvergence* in the FEM literature.

§11.6.2. *Fourier Patch Analysis

The following analysis is based on the modified differential equation (MoDE) method of Warming and Hyett [748] combined with the Fourier patch analysis approach of Park and Flaggs [535,536]. Consider a lattice of two-node prismatic bar elements of constant rigidity EA and equal length ℓ , as illustrated in Figure 11.8. The total length of the lattice is L . The system is subject to an arbitrary axial load $q(x)$. The only requirement on $q(x)$ is that it has a convergent Fourier series in the space direction.

From the lattice extract a patch¹² of two elements connecting nodes x_i , x_j and x_k as shown in Figure 11.8. The FEM patch equations at node j are

$$\frac{EA}{\ell} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} u_i \\ u_j \\ u_k \end{bmatrix} = f_j, \quad (11.51)$$

in which the node force f_j is obtained by consistent lumping:

$$f_j = \int_{x_i}^{x_k} q(x) N_j(x) dx = \int_{-1}^0 q(x_j + \psi\ell)(1 + \psi) \ell d\psi + \int_0^1 q(x_j + \psi\ell)(1 - \psi) \ell d\psi. \quad (11.52)$$

Here $N_j(x)$ is the “hat” trial basis function for node j , depicted in Figure 11.8, and $\psi = (x - x_j)/\ell$ is a dimensionless coordinate that takes the values $-1, 0$ and 1 at nodes i, j and k , respectively. If $q(x)$ is expanded

¹⁰ In variational language: the Green function of the $u'' = 0$ problem is included in the FEM trial space.

¹¹ These can happen when transforming such elements for 2D and 3D trusses. See Exercise E11.7.

¹² A patch is the set of all elements connected to a node; in this case j .

in Fourier series

$$q(x) = \sum_{m=1}^M q_m e^{i\beta_m x}, \quad \beta_m = m\pi/L, \quad (11.53)$$

(the term $m = 0$ requires special handling) the exact solution of the continuum equation $EA u'' + q = 0$ is

$$u^*(x) = \sum_{m=1}^M u_m^* e^{i\beta_m x}, \quad u_m^* = \frac{q_m e^{i\beta_m x}}{EA\beta_m^2}. \quad (11.54)$$

Evaluation of the consistent force using (11.52) gives

$$f_j = \sum_{m=1}^M f_{jm}, \quad f_{jm} = q_m \ell \frac{\sin^2(\frac{1}{2}\beta_m \ell)}{\frac{1}{4}\beta_m^2 \ell^2} e^{i\beta_m x_j}. \quad (11.55)$$

To construct a modified differential equation (MoDE), expand the displacement by Taylor series centered at node j . Evaluate at i and k : $u_i = u_j - \ell u_j' + \ell^2 u_j''/2! - \ell^3 u_j'''/3! + \ell^4 u_j^{iv}/4! + \dots$ and $u_k = u_j + \ell u_j' + \ell^2 u_j''/2 + \ell^3 u_j'''/3! + \ell^4 u_j^{iv}/4! + \dots$. Replace these series into (11.51) to get

$$-2EA\ell \left(\frac{1}{2!} u_j'' + \frac{\ell^2}{4!} u_j^{iv} + \frac{\ell^4}{6!} u_j^{vi} + \dots \right) = f_j. \quad (11.56)$$

This is an ODE of infinite order. It can be reduced to an algebraic equation by assuming that the response of (11.56) to $q_m e^{i\beta_m x}$ is harmonic: $u_{jm} e^{i\beta_m x}$. If so $u_{jm}'' = -\beta_m^2 u_{jm}$, $u_{jm}^{iv} = \beta_m^4 u_{jm}$, etc, and the MoDE becomes

$$2EA\ell \beta_m^2 \left(\frac{1}{2!} - \frac{\beta_m^2 \ell^2}{4!} + \frac{\beta_m^4 \ell^4}{6!} - \dots \right) u_{jm} = 4EA\ell \sin^2(\frac{1}{2}\beta_m \ell) u_{jm} = f_{jm} = q_m \ell \frac{\sin^2(\frac{1}{2}\beta_m \ell)}{\frac{1}{4}\beta_m^2 \ell^2} e^{i\beta_m x_j}. \quad (11.57)$$

Solving gives $u_{jm} = q_m e^{i\beta_m x_j} / (EA\beta_m^2)$, which compared with (11.54) shows that $u_{jm} = u_m^*$ for any $m > 0$. Consequently $u_j = u_j^*$. In other words, the MoDE (11.56) and the original ODE: $EAu'' + q = 0$ have the same value at $x = x_j$ for any load $q(x)$ developable as (11.53). This proves nodal exactness. In between nodes the two solutions will not agree.¹³

The case $m = 0$ has to be treated separately since the foregoing expressions become 0/0. The response to a uniform $q = q_0$ is a quadratic in x , and it is not difficult to prove nodal exactness.

§11.6.3. *Robin Boundary Conditions

Suppose that for a bar of length L one has the following end conditions: $u'(0) = au(0) + b$ at $x = 0$ and $u'(L) = au(L) + b$ at $x = L$, in which a and b are given coefficients. Those are called Robin BCs in the literature. Adjoining them as Courant penalty terms gives the functional

$$F(u) = \int_0^L [\frac{1}{2} EA (u')^2 - q u] dx + \frac{1}{2} [u'(0) - au(0) - b]^2 + \frac{1}{2} [u'(L) + au(L) + b]^2. \quad (11.58)$$

Divide $[0, L]$ into N_e elements and $N = N_e + 1$ nodes. Do C^0 linear interpolation over each element, insert into $F(u)$ to get $F_d(u) = \frac{1}{2} \mathbf{u}^T \mathbf{K} \mathbf{u} - \mathbf{f}^T \mathbf{v}$, in which \mathbf{u} is the vector of node values, \mathbf{K} the master stiffness matrix and \mathbf{f} the master force vector. Coefficients a and b will affect both \mathbf{K} and \mathbf{f} .

¹³ The FEM solution varies linearly between nodes whereas the exact one is generally trigonometric.

Vanishing of the first variation: $\delta F_d = 0$ yields the FEM equations $\mathbf{Ku} = \mathbf{f}$ to be solved for \mathbf{u} . The Robin BCs at $x = 0$ and $x = L$ will affect the stiffness and force contributions of the first and last elements, but not those of interior elements.

This kind of boundary value problem (i.e., with Robin BCs) is common in heat conduction and heat transfer with convection given over cooling surfaces. In that case the heat flux is proportional to the difference of the (unknown) surface temperature and that of the cooling fluid. Elements that "touch" the convection boundary are affected.

Notes and Bibliography

The foregoing account pertains to the simplest structural finite element: the two-node bar element. For bar members these developments may be generalized in several directions, three of which are mentioned next.

Refined bar models. Adding internal nodes we can pass from linear to quadratic and cubic shape functions. These elements are rarely useful on their own right, but as accessories to 2D and 3D high order continuum elements (for example, to model edge reinforcements.) For that reason they are not considered here. The 3-node bar element is developed in exercises assigned in Chapter 16.

Use in 2D and 3D truss structures. The only additional ingredients are the local-to-global transformations discussed in Chapters 3 and 6.

Curved bar elements. These can be derived using isoparametric mapping, a device introduced later.

Matrices for straight bar elements are available in any finite element book; for example Przemieniecki [575].

Tonti diagrams were introduced in the 1970s in papers now difficult to access, for example [718]. Scanned images are available, however, from <http://www.dic.units.it/perspage/discretephysics>

The fundamentals of Variational Calculus may be studied in the excellent textbook [289], which is now available in an inexpensive Dover edition. The proof of the MPE principle can be found in texts on variational methods in mechanics. For example: Langhaar [420], which is the most readable "old fashioned" treatment of the energy principles of structural mechanics, with a clear treatment of virtual work. (Out of print but used copies may be found via the web engines cited in §1.5.2.) The elegant treatment by Lanczos [419] is recommended as reading material although it is more oriented to physics than structural mechanics.

It was noted that weak forms occupy an intermediate position between two older classical areas: differential equations (introduced in the XVII Century by the Calculus founders) and variational forms (introduced by Euler in the XVIII Century). Some weak forms in disguise are also ancient; e.g., the PVW was placed on firm mathematical grounds by Lagrange in the late XVIII Century [416]. But their rapid development as tools for producing approximate solutions of ODEs and PDEs took place in the early XIX Century. Five important variants are: Galerkin (1915), subdomain (1923), least squares (1928), moments (1932), and collocation (1937). These, as well as a few others of less importance, were unified in 1956 under the label Method of Weighted Residuals or MWR, by Crandall [155]. Other attempts at unification during this period may be found in [19,145]. The use of MWR methods, especially Galerkin's, as enabling devices to generate finite element equations developed rapidly following the pioneer paper [784]. The chief motivation was to accommodate application problems where a classical variational formulation does not exist, or is inconvenient to use.

The first accuracy study of FEM discretizations using modified equation methods is by Waltz et. al. [745]; however their procedures were faulty, which led to incorrect conclusions. The first correct derivation of modified equations appeared in [748]. The topic has recently attracted interest from applied mathematicians because modified equations provide a systematic tool for *backward error analysis* of differential equations: the discrete solution is the exact solution of the modified problem. This is particularly important for the study of long term behavior of discrete dynamical systems, whether deterministic or chaotic. Recommended references along these lines are [308,316,683].

Nodal exactness of bar models for point node loads is a particular case of a theorem by Tong [716]. For arbitrary loads it was proven by Park and Flagg [535,536], who followed a variant of the scheme of §11.6.2.

Chapter 11: VARIATIONAL FORMULATION OF BAR ELEMENT

A different technique is used in Exercise 11.10. The budding concept of superconvergence, which emerged in the late 1960s, is outlined in the book of Strang and Fix [672]. There is a monograph [746] devoted to the subject; it covers only Poisson problems but provides a comprehensive reference list until 1995.

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 11

Variational Formulation of Bar Element

EXERCISE 11.1 [D:10] Explain the kinematic admissibility requirements stated in ? in terms of physics, namely ruling out the possibility of gaps or interpenetration as the bar material deforms.

EXERCISE 11.2 [A/C:15] Using (11.25), derive the stiffness matrix for a *tapered* bar element in which the cross section area varies linearly along the element length:

$$A = A_i(1 - \zeta) + A_j \zeta, \quad (\text{E11.1})$$

where A_i and A_j are the areas at the end nodes, and $\zeta = x^e/\ell$ is the dimensionless coordinate defined in §11.3.6. Show that this yields the same answer as that of a stiffness of a constant-area bar with cross section $\frac{1}{2}(A_i + A_j)$. Note: the following *Mathematica* script may be used to solve this exercise:¹⁴

```
ClearAll[Le,x,Em,A,Ai,Aj];
Be={{-1,1}}/Le; ζ=x/Le; A=Ai*(1-ζ)+Aj*ζ;
Ke=Integrate[Em*A*Transpose[Be].Be,{x,0,Le}];
Ke=Simplify[Ke];
Print["Ke for varying cross section bar: ",Ke//MatrixForm];
```

In this and following scripts Le stands for ℓ .

EXERCISE 11.3 [A:10] Find the consistent load vector \mathbf{f}^e for a bar of *constant* area A subject to a *uniform* axial force $q = \rho g A$ per unit length along the element. Show that this vector is the same as that obtained with the element-by-element (EbE) “lumping” method of §8.4, which simply assigns half of the total load: $\frac{1}{2}\rho g A \ell$, to each node. Hint: use (11.29) and $\int_0^1 \zeta d\zeta = 1/2$.

EXERCISE 11.4 [A/C:15] Repeat the previous calculation for the tapered bar element subject to a force $q = \rho g A$ per unit length, in which A varies according to (E11.1) whereas ρ and g are constant. Check that if $A_i = A_j$ one recovers $f_i = f_j = \frac{1}{2}\rho g A \ell$. Note: the following *Mathematica* script may be used to solve this exercise:¹⁵

```
ClearAll[q,A,Ai,Aj,ρ,g,Le,x];
ζ=x/Le; Ne={{1-ζ,ζ}}; A=Ai*(1-ζ)+Aj*ζ; q=ρ*g*A;
fe=Integrate[q*Ne,{x,0,Le}];
fe=Simplify[fe];
Print["fe for uniform load q: ",fe//MatrixForm];
ClearAll[A];
Print["fe check: ",Simplify[fe/.{Ai->A,Aj->A}]]//MatrixForm];
```

EXERCISE 11.5 [A/C:20] A tapered bar element of length ℓ , end areas A_i and A_j with A interpolated as per (E11.1), and constant density ρ , rotates on a plane at uniform angular velocity ω (rad/sec) about node i . Taking axis x along the rotating bar with origin at node i , the centrifugal axial force is $q(x) = \rho A \omega^2 x$ along the length, in which $x \equiv x^e$. Find the consistent node forces as functions of ρ , A_i , A_j , ω and ℓ , and specialize the result to the prismatic bar $A = A_i = A_j$. Partial result check: $f_j = \frac{1}{3}\rho \omega^2 A \ell^2$ for $A = A_i = A_j$.

¹⁴ The `ClearAll[...]` at the start of the script is recommended programming practice to initialize variables and avoid “cell crosstalk.” In a `Module` this is done by listing the local variables after the `Module` keyword.

¹⁵ The `ClearAll[A]` before the last statement is essential; else A would retain the previous assignment.

EXERCISE 11.6 [A:15] (Requires knowledge of Dirac’s delta function properties.) Find the consistent load vector \mathbf{f}^e if the bar is subjected to a concentrated axial force Q at a distance $x = a$ from its left end. Use (11.28), with $q(x) = Q \delta(x-a)$, in which $\delta(a)$ is the one-dimensional Dirac’s delta function at $x = a$. Note: the following script does it by *Mathematica*, but it is overkill:

```
ClearAll[Le,q,Q,a,x];
ζ=x/Le; Ne={{1-ζ,ζ}}; q=Q*DiracDelta[x-a];
fe=Simplify[ Integrate[q*Ne,{x,-Infinity,Infinity}] ];
Print["fe for point load Q at x=a: ",fe//MatrixForm];
```

EXERCISE 11.7 [C+D:20] In a learned paper, Dr. I. M. Clueless proposes “improving” the result for the example truss by putting three extra nodes, 4, 5 and 6, at the midpoint of members 1–2, 2–3 and 1–3, respectively. His “reasoning” is that more is better. Try Dr. C.’s suggestion using the *Mathematica* implementation of Chapter 4 and verify that the solution “blows up” because the modified master stiffness is singular. Explain physically what happens.

EXERCISE 11.8 [C+D:15] This exercise illustrates “Galerkin surprises.” Take up again the example of §11.5.2, but suppose now that the axial load varies linearly, as in (11.31). The trial and weight function assumptions are the quadratic polynomials (11.41). Show that the integral (11.39) is given by

$$12 J/L = b_0 (24EA a_2 + 6(q_1 + q_2)) + b_1 (12EA a_2 + 2(q_1 + 2q_2)) + b_2 (8EA a_2 + (q_1 + 3q_2)), \quad (\text{E11.2})$$

and that the resulting 3 equations for a_2 are *inconsistent* unless $q_1 = q_2$. Only one weight function gives the correct solution at $x = L$; which one? Note that the Galerkin method is generally viewed as the “most reliable” member of the MWR tribe. But unforeseen surprises have a silver lining: more papers can be written to explain them. Here is a partial fix: make the test function satisfy the essential BC *a priori*.

EXERCISE 11.9 [A:20]. Prove that (11.47) is the first variation of (11.48), thus linking the PVW with the TPE functional. See Remark 11.5 for a hint on how to treat the boundary term in (11.47).

EXERCISE 11.10 [A:35, close to research paper level]. Prove nodal exactness of the two-node bar element for arbitrary but Taylor expandable loading without using the Fourier series approach. Hints: expand $q(x) = q(x_j) + (\ell\psi)q'(x_j) + (\ell\psi)^2 q''(x_j)/2! + \dots$, where $\ell\psi = x - x_j$ is the distance to node j , compute the consistent force $f_j(x)$ from (11.52), and differentiate the MoDE (11.56) repeatedly in x while truncating all derivatives to a maximum order $n \geq 2$. Show that the original ODE: $EAu'' + q = 0$, emerges as an identity regardless of how many derivatives are kept.

12

Variational Formulation of Plane Beam Element

TABLE OF CONTENTS

	Page
§12.1. Introduction	12–3
§12.2. What is a Beam?	12–3
§12.2.1. Terminology	12–3
§12.2.2. Mathematical Models	12–3
§12.2.3. Assumptions of Classical Beam Theory	12–4
§12.3. The Bernoulli-Euler Beam Theory	12–4
§12.3.1. Element Coordinate Systems	12–4
§12.3.2. Kinematics	12–5
§12.3.3. Loading	12–5
§12.3.4. Support Conditions	12–5
§12.3.5. Strains, Stresses and Bending Moments	12–5
§12.4. Total Potential Energy Functional	12–6
§12.5. Beam Finite Elements	12–7
§12.5.1. Finite Element Trial Functions	12–8
§12.5.2. Shape Functions	12–8
§12.6. The Finite Element Equations	12–9
§12.6.1. The Stiffness Matrix of a Prismatic Beam	12–10
§12.6.2. Consistent Nodal Force Vector for Uniform Load	12–11
§12. Notes and Bibliography	12–15
§12. References	12–15
§12. Exercises	12–16

§12.1. Introduction

The previous Chapter introduced the TPE-based variational formulation of finite elements, which was illustrated for the bar element. This Chapter applies that technique to a more complicated one-dimensional element: the plane beam described by engineering beam theory.

Mathematically, the main difference of beams with respect to bars is the increased order of continuity required for the assumed transverse-displacement functions to be admissible. Not only must these functions be continuous but they must possess continuous x first derivatives. To meet this requirement both deflections *and* slopes are matched at nodal points. Slopes may be viewed as *rotational* degrees of freedom in the small-displacement assumptions used here.

§12.2. What is a Beam?

Beams are the most common type of structural component, particularly in Civil and Mechanical Engineering. A *beam* is a bar-like structural member whose primary function is to support *transverse loading* and carry it to the supports. See Figure 12.1.

By “bar-like” it is meant that one of the dimensions is considerably larger than the other two. This dimension is called the *longitudinal dimension* or *beam axis*. The intersection of planes normal to the longitudinal dimension with the beam member are called *cross sections*. A *longitudinal plane* is one that passes through the beam axis.

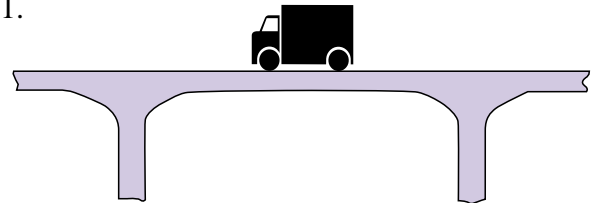


FIGURE 12.1. A beam is a structural member designed to resist transverse loads.

A beam resists transverse loads mainly through *bending action*. Bending produces compressive longitudinal stresses in one side of the beam and tensile stresses in the other.

The two regions are separated by a *neutral surface* of zero stress. The combination of tensile and compressive stresses produces an internal *bending moment*. This moment is the primary mechanism that transports loads to the supports. The mechanism is illustrated in Figure 12.2.

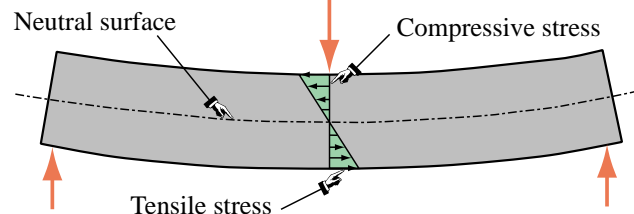


FIGURE 12.2. Beam transverse loads are primarily resisted by bending action.

§12.2.1. Terminology

A *general beam* is a bar-like member designed to resist a combination of loading actions such as biaxial bending, transverse shears, axial stretching or compression, and possibly torsion. If the internal axial force is compressive, the beam has also to be designed to resist buckling. If the beam is subject primarily to bending and axial forces, it is called a *beam-column*. If it is subjected primarily to bending forces, it is called simply a beam. A beam is *straight* if its longitudinal axis is straight. It is *prismatic* if its cross section is constant.

A *spatial beam* supports transverse loads that can act on arbitrary directions along the cross section. A *plane beam* resists primarily transverse loading on a preferred longitudinal plane. This Chapter considers only plane beams.

§12.2.2. Mathematical Models

One-dimensional mathematical models of structural beams are constructed on the basis of *beam theories*. Because beams are actually three-dimensional bodies, all models necessarily involve some form of approximation to the underlying physics. The simplest and best known models for straight, prismatic beams are based on the *Bernoulli-Euler beam theory* (also called *classical beam theory* and *engineering beam theory*), and the *Timoshenko beam theory*. The Bernoulli-Euler theory is that taught in introductory Mechanics of Materials courses, and is the one emphasized in this Chapter. The Timoshenko beam model is presented in Chapter 13, which collects advanced material.

Both models can be used to formulate beam finite elements. The Bernoulli-Euler beam theory leads to the so-called *Hermitian* beam elements.¹ These are also known as C^1 elements for the reason explained in §12.5.1. This model neglects the effect of transverse shear deformations on the internal energy. Elements based on Timoshenko beam theory, also known as C^0 elements, incorporate a first order correction for transverse shear effects. This model assumes additional importance in dynamics and vibration.

§12.2.3. Assumptions of Classical Beam Theory

The Bernoulli-Euler or classical beam theory for *plane beams* rests on the following assumptions:

1. *Planar symmetry*. The longitudinal axis is straight and the cross section of the beam has a longitudinal plane of symmetry. The resultant of the transverse loads acting on each section lies on that plane. The support conditions are also symmetric about this plane.
2. *Cross section variation*. The cross section is either constant or varies smoothly.
3. *Normality*. Plane sections originally normal to the longitudinal axis of the beam remain plane and normal to the deformed longitudinal axis upon bending.
4. *Strain energy*. The internal strain energy of the member accounts only for bending moment deformations. All other contributions, notably transverse shear and axial force, are ignored.
5. *Linearization*. Transverse deflections, rotations and deformations are considered so small that the assumptions of infinitesimal deformations apply.
6. *Material model*. The material is assumed to be elastic and isotropic. Heterogeneous beams fabricated with several isotropic materials, such as reinforced concrete, are not excluded.

§12.3. The Bernoulli-Euler Beam Theory

§12.3.1. Element Coordinate Systems

Under transverse loading one of the top surfaces shortens while the other elongates; see Figure 12.2. Therefore a *neutral surface* that undergoes no axial strain exists between the top and the bottom. The intersection of this surface with each cross section defines the *neutral axis* of that cross section.²

¹ The qualifier “Hermitian” relates to the use of a transverse-displacement interpolation formula studied by the French mathematician Hermite. The term has nothing to do with the mathematical model used.

² If the beam is homogenous, the neutral axis passes through the centroid of the cross section. If the beam is fabricated of different materials — for example, a reinforced concrete beam — the neutral axis passes through the centroid of an “equivalent” cross section. This topic is covered in Mechanics of Materials textbooks; for example Popov [567].

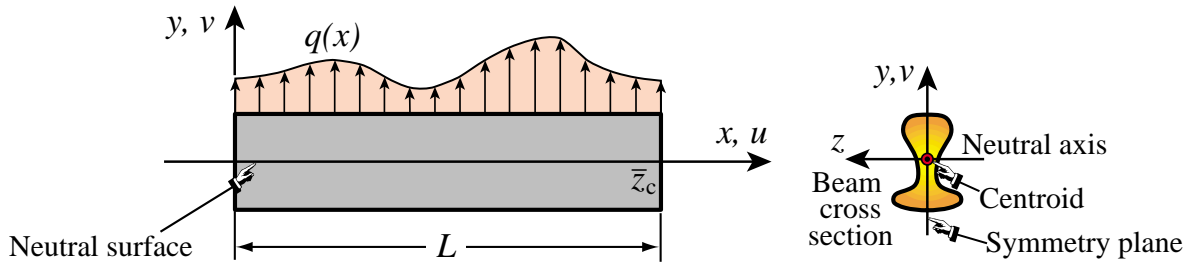


FIGURE 12.3. Terminology and choice of axes for Bernoulli-Euler model of plane beam.

The Cartesian axes for plane beam analysis are chosen as shown in Figure 12.3. Axis x lies along the longitudinal beam axis, at neutral axis height. Axis y lies in the symmetry plane and points upwards. Axis z is directed along the neutral axis, forming a RHS system with x and y . The origin is placed at the leftmost section. The total length (or span) of the beam member is called L .

§12.3.2. Kinematics

The *motion* under loading of a plane beam member in the x, y plane is described by the two dimensional displacement field

$$\begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix}, \quad (12.1)$$

where u and v are the axial and transverse displacement components, respectively, of an arbitrary beam material point. The motion in the z direction, which is primarily due to Poisson's ratio effects, is of no interest. The normality assumption of the Bernoulli-Euler model can be represented mathematically as

$$u(x, y) = -y \frac{\partial v(x)}{\partial x} = -y v' = -y \theta, \quad v(x, y) = v(x). \quad (12.2)$$

Note that the slope $v' = \partial v / \partial x = dv / dx$ of the deflection curve has been identified with the *rotation* symbol θ . This is permissible because θ represents to first order, according to the kinematic assumptions of this model, the rotation of a cross section about z positive CCW.

§12.3.3. Loading

The transverse force *per unit length* that acts on the beam in the $+y$ direction is denoted by $q(x)$, as illustrated in Figure 12.3. Concentrated loads and moments acting on isolated beam sections can be represented by the delta function and its derivative. For example, if a transverse point load F acts at $x = a$, it contributes $F\delta(a)$ to $q(x)$. If the concentrated moment C acts at $x = b$, positive CCW, it contributes $C\delta'(b)$ to $q(x)$, where δ' denotes a doublet acting at $x = b$.

§12.3.4. Support Conditions

Support conditions for beams exhibit far more variety than for bar members. Two canonical cases are often encountered in engineering practice: simple support and cantilever support. These are illustrated in Figures 12.4 and 12.5, respectively. Beams often appear as components of skeletal structures called frameworks, in which case the support conditions are of more complex type.

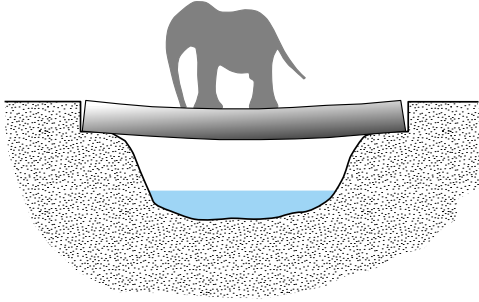


FIGURE 12.4. A simply supported beam has end supports that preclude transverse displacements but permit end rotations.

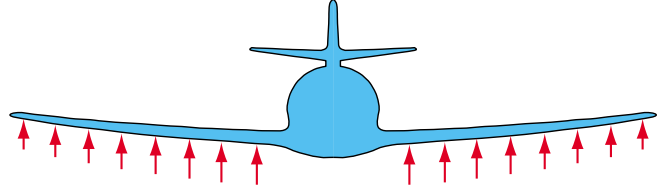


FIGURE 12.5. A cantilever beam is clamped at one end and free at the other. Airplane wings and stabilizers are examples of this configuration.

§12.3.5. Strains, Stresses and Bending Moments

The Bernoulli-Euler or classical model assumes that the internal energy of beam member is entirely due to bending strains and stresses. Bending produces axial stresses σ_{xx} , which will be abbreviated to σ , and axial strains e_{xx} , which will be abbreviated to e . The strains can be linked to the displacements by differentiating the axial displacement $u(x)$ of (12.2):

$$e = \frac{\partial u}{\partial x} = -y \frac{\partial^2 v}{\partial x^2} = -y \frac{d^2 v}{dx^2} = -y v'' = -y \kappa. \quad (12.3)$$

Here κ denotes the deformed beam axis curvature, which to first order is $\kappa \approx d^2 v / dx^2 = v''$. The bending stress $\sigma = \sigma_{xx}$ is linked to e through the one-dimensional Hooke's law

$$\sigma = E e = -E y \frac{d^2 v}{dx^2} = -E y \kappa, \quad (12.4)$$

where E is the longitudinal elastic modulus. The most important stress resultant in classical beam theory is the *bending moment* M , which is defined as the cross section integral

$$M = \int_A -y \sigma dA = E \frac{d^2 v}{dx^2} \int_A y^2 dA = E I \kappa. \quad (12.5)$$

Here $I \equiv I_{zz}$ denotes the moment of inertia $\int_A y^2 dA$ of the cross section with respect to the z (neutral) axis. The bending moment M is considered positive if it compresses the upper portion: $y > 0$, of the beam cross section, as illustrated in Figure 12.6. This convention explains the negative sign of y in the integral (12.5). The product $E I$ is called the *bending rigidity* of the beam with respect to flexure about the z axis.

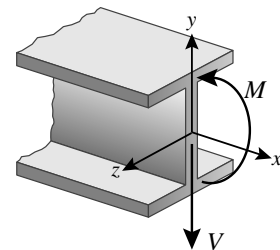


FIGURE 12.6. Positive sign convention for M and V .

The governing equations of the Bernoulli-Euler beam model are summarized in the Tonti diagram of Figure 12.7.

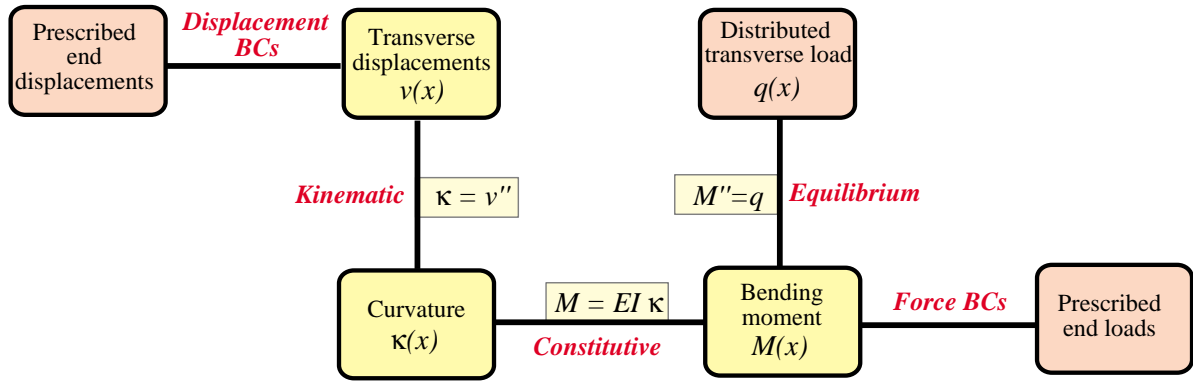


FIGURE 12.7. The Tonti diagram for the governing equations of the Bernoulli-Euler beam model.

§12.4. Total Potential Energy Functional

The total potential energy of the beam is

$$\Pi = U - W \quad (12.6)$$

where as usual U and W denote the internal and external energies, respectively. As previously explained, in the Bernoulli-Euler model U includes only the bending energy:

$$U = \frac{1}{2} \int_V \sigma e \, dV = \frac{1}{2} \int_0^L M \kappa \, dx = \frac{1}{2} \int_0^L EI \kappa^2 \, dx = \frac{1}{2} \int_0^L EI (v'')^2 \, dx = \frac{1}{2} \int_0^L v'' EI v'' \, dx. \quad (12.7)$$

The external work W accounts for the applied transverse force:

$$W = \int_0^L q v \, dx. \quad (12.8)$$

The three functionals Π , U and W must be regarded as depending on the transverse displacement $v(x)$. When this dependence needs to be emphasized we write $\Pi[v]$, $U[v]$ and $W[v]$.

Note that $\Pi[v]$ includes up to second derivatives in v , because $v'' = \kappa$ appears in U . This number is called the *variational index*. Variational calculus tells us that since the index is 2, admissible displacements $v(x)$ must be continuous, have continuous first derivatives (slopes or rotations), and satisfy the displacement BCs exactly. This continuity requirement can be succinctly stated by saying that admissible displacements must be C^1 continuous. This condition guides the construction of beam finite elements described below.

Remark 12.1. If there is an applied distributed moment $m(x)$ per unit of beam length, the external energy (12.8) must be augmented with a $\int_0^L m(x)\theta(x) \, dx$ term. This is further elaborated in Exercises 12.4 and 12.5. Such kind of distributed loading is uncommon in practice although in framework analysis occasionally the need arises for treating a concentrated moment between nodes.

§12.5. Beam Finite Elements

Beam finite elements are obtained by subdividing beam members longitudinally. The simplest Bernoulli-Euler plane beam element has two end nodes: 1 and 2, and four degrees of freedom (DOF). These are collected in the node displacement vector

$$\mathbf{u}^e = [v_1 \ \theta_1 \ v_2 \ \theta_2]^T. \quad (12.9)$$

The element is shown in Figure 12.8, which pictures the undeformed and deformed configurations.

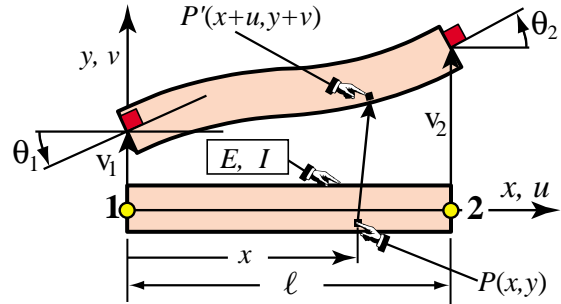


FIGURE 12.8. The two-node Bernoulli-Euler plane beam element with four DOFs.

§12.5.1. Finite Element Trial Functions

The freedoms (12.9) are used to define uniquely the variation of the transverse displacement $v^e(x)$ over the element. The C^1 continuity requirement says that both $v(x)$ and the slope $\theta = v'(x) = dv(x)/dx$ must be continuous over the entire member, and in particular between beam elements.

C^1 continuity can be trivially met *within each element* by choosing polynomial interpolation shape functions as shown below, because polynomials are C^∞ continuous. Matching nodal displacements and rotations with *adjacent elements* enforces the necessary interelement continuity.

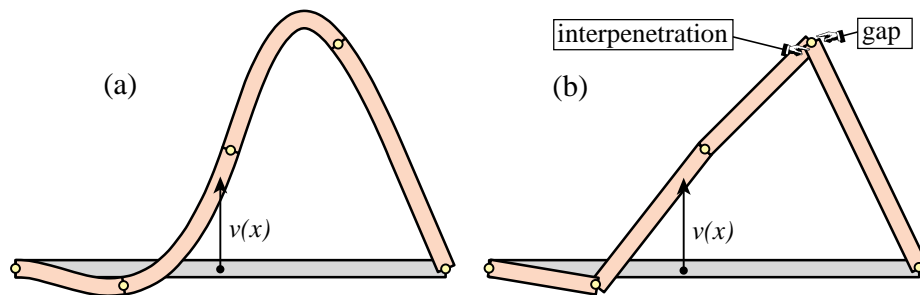


FIGURE 12.9. Deflection of a clamped-SS beam discretized with four elements, grossly exaggerated for visibility. (a) Cubic deflection elements; (b) linear deflection elements. The latter maintains only C^0 continuity, leading to unacceptable material gap and interpenetration at nodes.

Remark 12.2. The physical reason for C^1 continuity is illustrated in Figure 12.9, in which the lateral deflection curve $v(x)$ is grossly exaggerated for visibility. The left figure shows the approximation of $v(x)$ by four cubic functions, which maintain the required continuity. The right figure shows an attempt to approximate $v(x)$ by four piecewise linear functions that maintain only C^0 continuity. In this case material gap and interpenetration occur at the nodes, as well as at the clamped left end, because section rotations jump between elements.

§12.5.2. Shape Functions

The simplest shape functions that meet the C^1 continuity requirement for the nodal DOF configuration (12.9) are called the *Hermitian cubic* shape functions. The interpolation formula based on these functions is

$$v^e = [N_{v1}^e \ N_{\theta1}^e \ N_{v2}^e \ N_{\theta2}^e] \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \mathbf{N}^e \mathbf{u}^e. \quad (12.10)$$

These shape functions are conveniently expressed in terms of the dimensionless “natural” coordinate

$$\xi = \frac{2x}{\ell} - 1, \quad (12.11)$$

where ℓ is the element length. Coordinate ξ varies from $\xi = -1$ at node 1 ($x = 0$) to $\xi = +1$ at node 2 ($x = \ell$). Note that $dx/d\xi = \frac{1}{2}\ell$ and $d\xi/dx = 2/\ell$. The shape functions in terms of ξ are

$$\begin{aligned} N_{v1}^e &= \frac{1}{4}(1 - \xi)^2(2 + \xi), \\ N_{\theta1}^e &= \frac{1}{8}\ell(1 - \xi)^2(1 + \xi), \\ N_{v2}^e &= \frac{1}{4}(1 + \xi)^2(2 - \xi), \\ N_{\theta2}^e &= -\frac{1}{8}\ell(1 + \xi)^2(1 - \xi). \end{aligned} \quad (12.12)$$

These four functions are depicted in Figure 12.10.

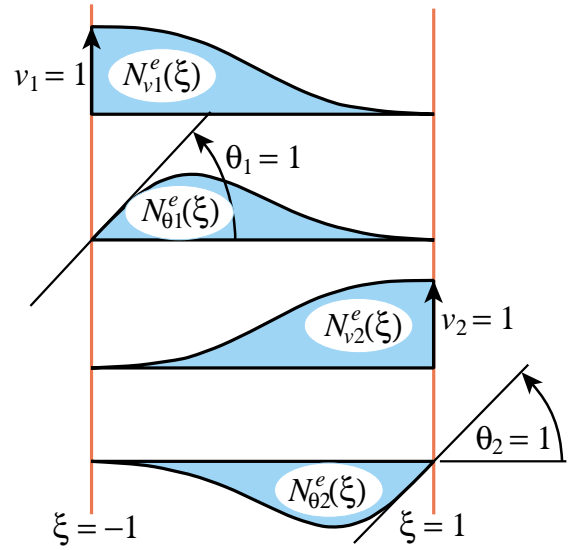


FIGURE 12.10. Cubic shape functions of plane beam element.

The curvature κ that appears in U can be expressed in terms of the nodal displacements by differentiating twice with respect to x :

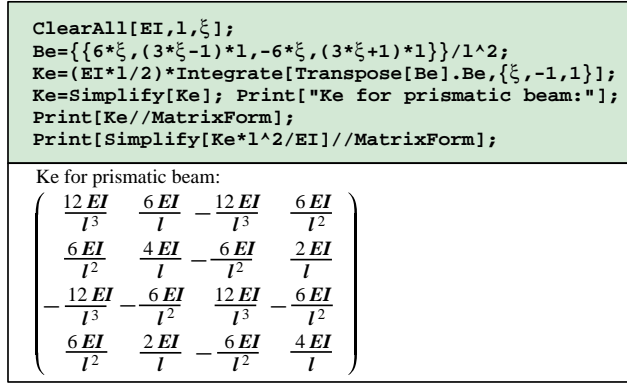
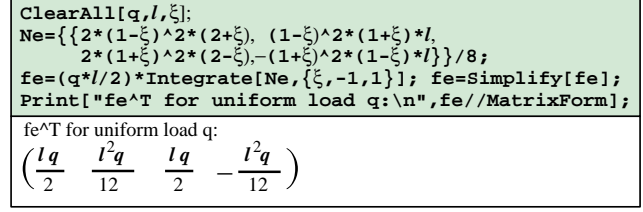
$$\kappa = \frac{d^2 v^e(x)}{dx^2} = \frac{4}{\ell^2} \frac{d^2 v^e(\xi)}{d\xi^2} = \frac{4}{\ell^2} \frac{d\mathbf{N}^e}{d\xi^2} \mathbf{u}^e = \mathbf{B} \mathbf{u}^e = \mathbf{N}'' \mathbf{u}^e. \quad (12.13)$$

Here $\mathbf{B} = \mathbf{N}''$ is the 1×4 curvature-displacement matrix

$$\mathbf{B} = \frac{1}{\ell} \begin{bmatrix} 6\frac{\xi}{\ell} & 3\xi - 1 & -6\frac{\xi}{\ell} & 3\xi + 1 \end{bmatrix}. \quad (12.14)$$

Remark 12.3. The $4/\ell^2$ factor in (12.13) comes from the differentiation chain rule. If $f(x)$ is a function of x , and $\xi = 2x/\ell - 1$, noting that $d(2/\ell)/dx = 0$ one gets

$$\frac{df(x)}{dx} = \frac{df(\xi)}{d\xi} \frac{d\xi}{dx} = \frac{2}{\ell} \frac{df(\xi)}{d\xi}, \quad \frac{d^2 f(x)}{dx^2} = \frac{d}{dx} \left(\frac{2}{\ell} \frac{df(\xi)}{d\xi} \right) = \frac{2}{\ell} \frac{d}{dx} \left(\frac{df(\xi)}{d\xi} \right) = \frac{4}{\ell^2} \frac{d^2 f(\xi)}{d\xi^2}. \quad (12.15)$$

FIGURE 12.11. Using *Mathematica* to form \mathbf{K}^e for a prismatic beam element.FIGURE 12.12. Using *Mathematica* to form \mathbf{f}^e for uniform transverse load q .

§12.6. The Finite Element Equations

Insertion of (12.12) and (12.14) into the TPE functional specialized to this element, yields the quadratic form in the nodal displacements

$$\Pi^e = \frac{1}{2}(\mathbf{u}^e)^T \mathbf{K}^e \mathbf{u}^e - (\mathbf{u}^e)^T \mathbf{f}^e, \quad (12.16)$$

where

$$\mathbf{K}^e = \int_0^\ell EI \mathbf{B}^T \mathbf{B} dx = \int_{-1}^1 EI \mathbf{B}^T \mathbf{B} \frac{1}{2} \ell d\xi, \quad (12.17)$$

is the element stiffness matrix and

$$\mathbf{f}^e = \int_0^\ell \mathbf{N}^T q dx = \int_{-1}^1 \mathbf{N}^T q \frac{1}{2} \ell d\xi, \quad (12.18)$$

is the consistent element node force vector. The calculation of the entries of \mathbf{K}^e and \mathbf{f}^e for prismatic beams and uniform load q is studied next. More complex cases are treated in the Exercises.

§12.6.1. The Stiffness Matrix of a Prismatic Beam

If the bending rigidity EI is constant over the element it can be moved out of the ξ -integral in (12.17):

$$\mathbf{K}^e = \frac{1}{2} EI \ell \int_{-1}^1 \mathbf{B}^T \mathbf{B} d\xi = \frac{EI}{2\ell} \int_{-1}^1 \begin{bmatrix} \frac{6\xi}{\ell} \\ 3\xi - 1 \\ -\frac{6\xi}{\ell} \\ 3\xi + 1 \end{bmatrix} \begin{bmatrix} \frac{6\xi}{\ell} & 3\xi - 1 & -\frac{6\xi}{\ell} & 3\xi + 1 \end{bmatrix} d\xi. \quad (12.19)$$

Expanding and integrating over the element yields

$$\mathbf{K}^e = \frac{EI}{2\ell^3} \int_{-1}^1 \begin{bmatrix} 36\xi^2 & 6\xi(3\xi-1)\ell & -36\xi^2 & 6\xi(3\xi+1)\ell \\ (3\xi-1)^2\ell^2 & -6\xi(3\xi-1)\ell & (9\xi^2-1)\ell^2 & 6\xi(3\xi+1)\ell \\ 36\xi^2 & -6\xi(3\xi+1)\ell & (3\xi+1)^2\ell^2 & 6\xi(3\xi-1)\ell \\ \text{symm} & & & \end{bmatrix} d\xi = \frac{EI}{\ell^3} \begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ & 4\ell^2 & -6\ell & 2\ell^2 \\ & & 12 & -6\ell \\ \text{symm} & & & 4\ell^2 \end{bmatrix} \quad (12.20)$$

Although the foregoing integrals can be easily carried out by hand, it is equally expedient to use a CAS such as *Mathematica* or *Maple*. For example the *Mathematica* script listed in the top box of Figure 12.11 processes (12.20) using the *Integrate* function. The output, shown in the bottom box, corroborates the hand integration result.

§12.6.2. Consistent Nodal Force Vector for Uniform Load

If q does not depend on x it can be moved out of (12.18), giving

$$\mathbf{f}^e = \frac{1}{2}q\ell \int_{-1}^1 \mathbf{N}^T d\xi = \frac{1}{2}q\ell \int_{-1}^1 \begin{bmatrix} \frac{1}{4}(1-\xi)^2(2+\xi) \\ \frac{1}{8}\ell(1-\xi)^2(1+\xi) \\ \frac{1}{4}(1+\xi)^2(2-\xi) \\ -\frac{1}{8}\ell(1+\xi)^2(1-\xi) \end{bmatrix} d\xi = \frac{1}{2}q\ell \begin{bmatrix} 1 \\ \frac{1}{6}\ell \\ 1 \\ -\frac{1}{6}\ell \end{bmatrix}. \quad (12.21)$$

This shows that a uniform load q over the beam element maps to two transverse node loads $q\ell/2$, as may be expected, plus two nodal moments $\pm q\ell^2/12$. The latter are called the *fixed-end moments* in the structural mechanics literature.³ The hand result (12.21) can be verified with the *Mathematica* script of Figure 12.12, in which \mathbf{f}^e is printed as a row vector to save space.

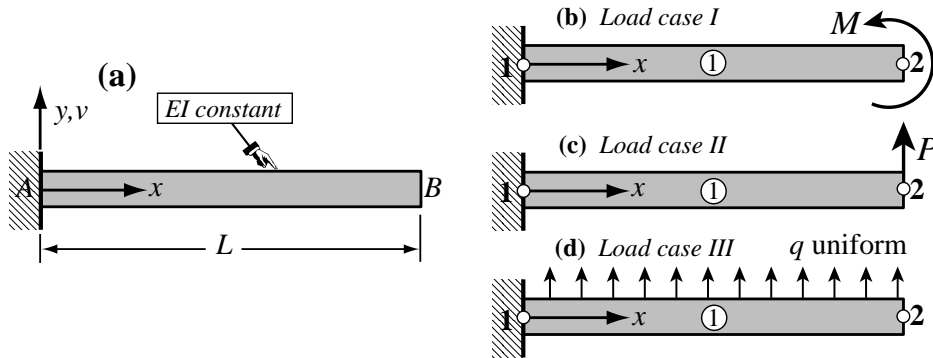


FIGURE 12.13. Cantilever beam problem for Example 12.1: (a) structure, (b-c): one-element FEM idealizations for three load cases.

Example 12.1. To see the beam element in action consider the cantilever illustrated in Figure 12.13(a). The beam is prismatic with constant rigidity EI and span L . It is discretized with a single element as shown in Figure 12.13(b,c,d), and subjected to the three load cases pictured there. Case I involves an applied end moment M , case II a transverse end force P , and case III a uniformly distributed load q over the entire beam. The FEM equations are constructed using the stiffness matrix (12.20) with $\ell = L$.

For the first two load cases, forces at end node 2 are directly set up from the given loads since no lumping is needed. Applying the support conditions $v_1 = \theta_1 = 0$ gives the reduced stiffness equations

$$\frac{EI}{L^3} \begin{bmatrix} 12 & -6L \\ -6L & 4L^2 \end{bmatrix} \begin{bmatrix} v_2^I \\ \theta_2^I \end{bmatrix} = \begin{bmatrix} 0 \\ M \end{bmatrix}, \quad \frac{EI}{L^3} \begin{bmatrix} 12 & -6L \\ -6L & 4L^2 \end{bmatrix} \begin{bmatrix} v_2^{II} \\ \theta_2^{II} \end{bmatrix} = \begin{bmatrix} P \\ 0 \end{bmatrix}, \quad (12.22)$$

³ Introduced by Hardy Cross in 1930 (long before FEM) as a key ingredient for his moment distribution method. Indeed the title of his famous paper [169] is “Analysis of continuous frames by distributing fixed-end moments.”

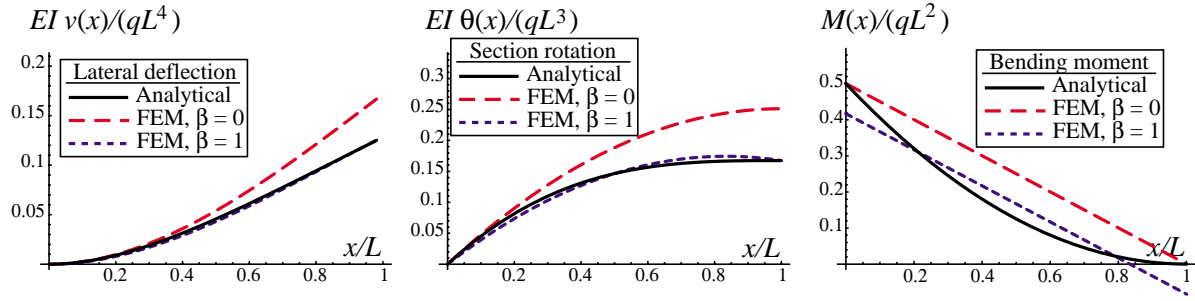


FIGURE 12.14. FEM versus analytical solutions for load case III of Example 12.1.

for load cases I and II, respectively. Solving gives the tip deflections $v_2^I = ML^2/(2EI)$ and $v_2^{II} = PL^3/(3EI)$, and the tip rotations $\theta_2^I = ML/EI$ and $\theta_2^{II} = PL^2/(2EI)$. These agree with the analytical values provided by Bernoulli-Euler beam theory. Thus a one-element idealization is sufficient for exactness. The reason is that the analytical deflection profiles $v(x)$ are quadratic and cubic polynomials in x for cases I and II, respectively. Both are included in the span of the element shape functions. Displacements $v(x)$, rotations $\theta(x)$ and moments $M(x)$ expressed as functions of x also agree with the analytical solution, as may be expected.

The results for load case III are more interesting since now the exact deflection is a quartic polynomial, which lies beyond the span of the FEM shape functions. A dimensionless parameter $0 \leq \beta \leq 1$ is introduced in the reduced stiffness equations to study the effect of load lumping method on the solution:

$$\frac{EI}{L^3} \begin{bmatrix} 12 & -6L \\ -6L & 4L^2 \end{bmatrix} \begin{bmatrix} v_2^{III} \\ \theta_2^{III} \end{bmatrix} = \frac{1}{2} q L \begin{bmatrix} 1 \\ -\frac{1}{6}\beta L \end{bmatrix}. \quad (12.23)$$

Setting $\beta = 1$ gives the energy consistent load lumping (12.21) whereas $\beta = 0$ gives the EbE (here same as NbN) load lumping $f_2^{III} = \frac{1}{2} q L$ with zero fixed-end moments. The solution of (12.23) is $v_2^{III} = q L^4(4 - \beta)/(24 EI)$ and $\theta_2^{III} = q L^3(3 - \beta)/(12 EI)$. From this one recovers the displacement, rotation and bending moment over the beam as

$$v^{III}(x) = q L^2 x^2 \frac{L(6-\beta) - 2x}{24 EI}, \quad \theta^{III}(x) = q L x \frac{L(6-\beta) - 3x}{12 EI}, \quad M^{III}(x) = \frac{q L}{12} (L(6-\beta) - 6x). \quad (12.24)$$

The analytical (exact) solution is

$$v_{ex}^{III}(x) = \frac{q x^2(3L^2 - 3Lx + x^2)}{24 EI}, \quad \theta_{ex}^{III}(x) = \frac{q x(6L^2 - 4Lx + x^2)}{6 EI}, \quad M_{ex}^{III}(x) = \frac{1}{2} q (L - x)^2. \quad (12.25)$$

The FEM and analytical solutions (12.24)-(12.25) are graphically compared in Figure 12.14. Deflections and rotations obtained with the consistent load lumping $\beta = 1$ agree better with the analytical solution. In addition the nodal values are exact (a superconvergence result further commented upon in the next Example). For the bending moment the values provided by the EbE lumping $\beta = 0$ are nodally exact but over the entire beam the $\beta = 1$ solution gives a better linear fit to the parabolic function $M_{ex}^{III}(x)$.

Example 12.2. The second example involves a simply supported beam under uniform line load q , depicted in Figure 12.15(a). It is prismatic with constant rigidity EI , span L , and discretized with two elements of length $L_1 = L(1/2 + \alpha)$ and $L_2 = L - L_1 = L(1/2 - \alpha)$, respectively. (Ordinarily two elements of the same length $1/2L$ would be used; the scalar $\alpha \in (-1/2, 1/2)$ is introduced to study the effect of unequal element sizes.)

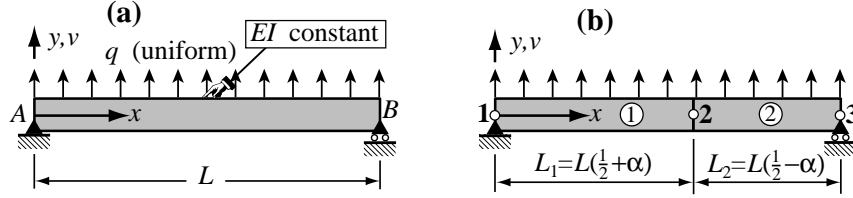


FIGURE 12.15. SS beam problem for Example 12.2: (a) structure, (b) two-element FEM idealization.

Using (12.20) and (12.21) to form the stiffness and consistent forces for both elements, assembling and applying the support conditions $v_1 = v_3 = 0$, provides the reduced stiffness equations

$$\frac{EI}{L^3} \begin{bmatrix} \frac{8L^2}{1+2\alpha} & \frac{-24L}{(1+2\alpha)^2} & \frac{4L^2}{1+2\alpha} & 0 \\ \frac{-24L}{(1+2\alpha)^2} & \frac{192(1+12\alpha^2)}{(1-4\alpha^2)^3} & \frac{192L\alpha}{(1-4\alpha^2)^2} & \frac{24L}{(1-2\alpha)^2} \\ \frac{4L^2}{1+2\alpha} & \frac{192L\alpha}{(1-4\alpha^2)^2} & \frac{16L^2}{1-4\alpha^2} & \frac{4L^2}{1-2\alpha} \\ 0 & \frac{24L}{(1-2\alpha)^2} & \frac{4L^2}{1-2\alpha} & \frac{8L^2}{1-2\alpha} \end{bmatrix} \begin{bmatrix} \theta_1 \\ v_2 \\ \theta_2 \\ \theta_3 \end{bmatrix} = \frac{qL}{2} \begin{bmatrix} \frac{L(1+2\alpha)^2}{24} \\ 1 \\ -\frac{L\alpha}{3} \\ -\frac{L(1-2\alpha)^2}{24} \end{bmatrix}. \quad (12.26)$$

Solving for the lateral displacement of node 2 gives $v_2 = qL^4(5 - 24\alpha^2 + 16\alpha^4)/(384EI)$. The exact deflection is $v(x) = qL^4(\zeta - 2\zeta^3 + \zeta^4)/(24EI)$ with $\zeta = x/L$. Replacing $x = L_1 = L(1/2 + \alpha)$ yields $v_2^{exact} = qL^4(5 - 24\alpha^2 + 16\alpha^4)/(384EI)$, which is the same as the FEM result. Likewise θ_2 is exact.

The result seems *prima facie* surprising. First, since the analytical solution is a quartic polynomial in x we have no reason to think that a cubic element will be exact. Second, one would expect accuracy deterioration as the element sizes differ more and more with increasing α . The fact that the solution at nodes is exact for any combination of element lengths is an illustration of *superconvergence*, a phenomenon already discussed in §11.5. A general proof of nodal exactness is given in §13.7, but it does require advanced mathematical tools. Note that displacements and rotations *inside* elements will not agree with the exact one; this can be observed in Figure 12.14(a,b) for load case III of the previous example.

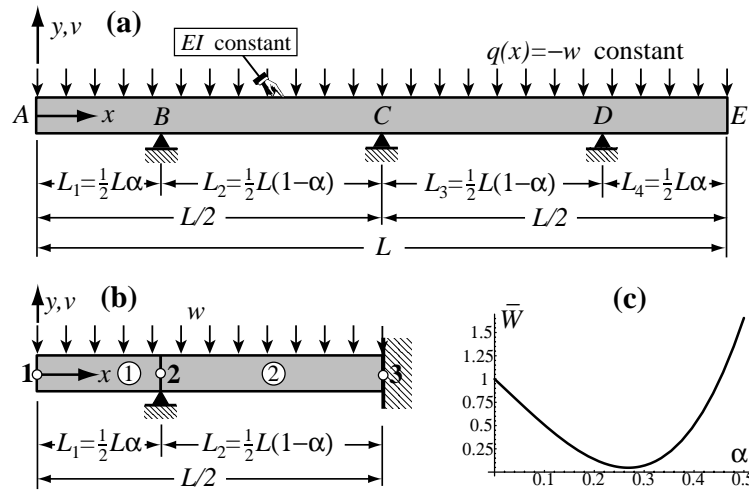


FIGURE 12.16. Continuum beam problem for Example 12.3, (a): structure, (b) two-element FEM model of half beam, (c) scaled external energy of FEM model as function of α .

Example 12.3. (Adapted from a driven-tank experiment by Patrick Weidman). This example displays the advantages of symbolic computation for solving a problem in geometric design: optimal location of supports. The prismatic continuous beam shown in Figure 12.16(a) is free at ends A and E, and simply supported at B, C and D. The beam has total span L and constant bending rigidity EI . It is loaded by a uniform distributed load $q(x) = -w$. Support C is at midspan whereas B and D are at distances $L_1 = L_4 = \frac{1}{2}L\alpha$ from the left and right free ends, respectively. Here $0 \leq \alpha < 1$ is a design parameter to be determined as discussed later.

Since the problem is symmetric about midspan C only one half of the structure, say AC, need to be discretized. The finite element model of this portion is shown in Figure 12.16(b). It has two beam elements and three nodes placed at A, B and C, respectively. Element lengths depend on the design parameter α , which is carried along as a variable. The six degrees of freedom are collected in $\mathbf{u} = [v_1 \ \theta_1 \ v_2 \ \theta_2 \ v_3 \ \theta_3]^T$. The master stiffness equations are

$$\frac{4EI}{L^3} \begin{bmatrix} \frac{24}{\alpha^3} & \frac{6L}{\alpha^2} & -\frac{24}{\alpha^3} & \frac{6L}{\alpha^2} & 0 & 0 \\ \frac{6L}{\alpha^2} & \frac{2L^2}{\alpha} & -\frac{6L}{\alpha^2} & \frac{L^2}{\alpha} & 0 & 0 \\ -\frac{24}{\alpha^3} & -\frac{6L}{\alpha^2} & \frac{24(1-3\alpha\hat{\alpha})}{\alpha^3\hat{\alpha}^3} & -\frac{6L(1-2\alpha)}{\alpha^2\hat{\alpha}^2} & -\frac{24}{\hat{\alpha}^3} & \frac{6L}{\hat{\alpha}^2} \\ \frac{6L}{\alpha^2} & \frac{L^2}{\alpha} & -\frac{6L(1-2\alpha)}{\alpha^2\hat{\alpha}^2} & \frac{2L^2}{\alpha\hat{\alpha}} & -\frac{6L}{\hat{\alpha}^2} & \frac{L^2}{\hat{\alpha}} \\ 0 & 0 & -\frac{24}{\hat{\alpha}^3} & -\frac{6L}{\hat{\alpha}^2} & \frac{24}{\hat{\alpha}^3} & -\frac{6L}{\hat{\alpha}^2} \\ 0 & 0 & \frac{6L}{\hat{\alpha}^2} & \frac{L^2}{\hat{\alpha}} & -\frac{6L}{\hat{\alpha}^2} & \frac{2L^2}{\hat{\alpha}} \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \\ v_3 \\ \theta_3 \end{bmatrix} = \frac{wL}{4} \begin{bmatrix} -\alpha \\ -\frac{L\alpha^2}{12} \\ -1 \\ \frac{L(2\alpha-1)}{12} \\ -\hat{\alpha} \\ \frac{L\hat{\alpha}^2}{12} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ f_2^r \\ 0 \\ f_3^r \\ m_3^r \end{bmatrix} \quad (12.27)$$

in which $\hat{\alpha} = 1 - \alpha$. Note that reaction forces are carefully segregated in (12.27) to simplify application of the general recovery technique discussed in §3.4.3. The support BCs are $v_2 = v_3 = \theta_3 = 0$, where the latter comes from the symmetry condition at C. Removing those freedoms provides the reduced stiffness equations

$$\frac{4EI}{L^3} \begin{bmatrix} \frac{24}{\alpha^3} & \frac{6L}{\alpha^2} & \frac{6L}{\alpha^2} \\ \frac{6L}{\alpha^2} & \frac{2L^2}{\alpha} & \frac{L^2}{\alpha} \\ \frac{6L}{\alpha^2} & \frac{L^2}{\alpha} & \frac{2L^2}{\alpha\hat{\alpha}} \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \frac{wL}{4} \begin{bmatrix} -\alpha \\ -\frac{L\alpha^2}{12} \\ \frac{L(2\alpha-1)}{12} \end{bmatrix}. \quad (12.28)$$

Solving yields

$$v_1 = -\frac{wL^4}{768EI} \alpha ((1+\alpha)^3 - 2), \quad \theta_1 = \frac{wL^3}{384EI} ((1+\alpha)^3 - 2), \quad \theta_2 = \frac{wL^3}{384EI} \hat{\alpha} (1 - 2\alpha - 5\alpha^2). \quad (12.29)$$

The complete solution is $\mathbf{u} = [v_1 \ \theta_1 \ 0 \ \theta_2 \ 0 \ 0]^T$. Inserting into (12.27) and solving for reactions gives

$$f_{r2} = \frac{wL}{16} \frac{3 + 2\alpha + \alpha^2}{\hat{\alpha}}, \quad f_{r3} = \frac{wL}{16} \frac{5 - 10\alpha - \alpha^2}{\hat{\alpha}}, \quad m_{r3} = -\frac{wL^2}{32} (1 - 2\alpha - \alpha^2). \quad (12.30)$$

whence the support reactions follow as $R_B = f_{r2}$ and $R_C = 2f_{r3}$. It remains to find the best α . Of course “best” depends on the optimality criterion. Four choices are examined below.

Minimum External Energy. The external energy at equilibrium is $W(\alpha) = \mathbf{f}^T \mathbf{u} = w^2 L^5 \bar{W}(\alpha) / (18432 EI)$, in which $\bar{W}(\alpha) = 1 - 5\alpha - 2\alpha^2 + 26\alpha^3 + 5\alpha^4 + 3\alpha^5$. Minimizing W with respect to α may be interpreted as finding the stiffest structure (in the energy sense) under the given load vector \mathbf{f} . A plot of $\bar{W}(\alpha)$ over $0 \leq \alpha \leq \frac{1}{2}$ clearly displays a minimum at $\alpha \approx 0.27$ as shown in Figure 12.16(c). Solving the quartic equation $d\bar{W}/d\alpha = 0$ gives one positive real root in the range $\alpha \in [0, 1)$, which to 5 places is $\alpha_{best} = 0.26817$.

Equal Reactions. A second choice is to require that supports at B and C take the same load: $R_B = R_C$ (note that, because of symmetry, $R_D = R_B$). Setting $f_{r2} = 2f_{r3}$ with their expressions taken from (12.30), yields $3 + 2\alpha + \alpha^2 = 10 - 20\alpha - 2\alpha^2$, or $7 - 22\alpha - 3\alpha^2 = 0$. This quadratic has the roots $\alpha = \frac{1}{3}(-11 \pm \sqrt{142})$. The positive real root $\alpha_{best} = 0.30546$ makes $R_B = R_C = R_D = wL/3$, as may be expected.

Minimum Relative Deflection. Consider two sections located at x_i and x_j , in which $\{x_i, x_j\} \in [0, \frac{1}{2}L]$, with lateral displacements $v_i = v(x_i)$ and $v_j = v(x_j)$, respectively. The maximum relative deflection is defined as $v_{ji}^{max}(\alpha) = \max |v_j - v_i|$ for a fixed α . We seek the $\alpha \in [0, 1)$ that minimizes $v_{ji}^{max}(\alpha)$. The computations are far more complex than for the previous two criteria and are the subject of Exercise 12.11. Result: the best α is the positive real root of $4 + 11\alpha - 81\alpha^2 - 49\alpha^3 - 47\alpha^4 = 0$, which to 5 places is $\alpha_{best} = 0.26681$. If this value is adopted, the relative deflection does not exceed $v_{ij}^{max} < wL^4/(67674EI)$.

Minimum Absolute Moment. Let $M(x, \alpha)$ denote the bending moment function recovered from the FEM solution for a fixed α . The maximum absolute moment is $M^{max}(\alpha) = \max |M(x, \alpha)|$ for $x \in [0, \frac{1}{2}L]$. We seek an $\alpha \in [0, 1)$ that minimizes it. This is the topic of Exercise 12.12. This problem is less well posed than the previous one because $M(x, \alpha)$ varies linearly over each element, is nonzero at node 1 and discontinuous at node 2. On the other hand, the exact bending moment varies parabolically, is zero at node 1 and continuous at node 2. Result: using the FEM-recovered $M(x, \alpha)$ and taking the average M at node 2, one finds that the best α is the positive root of $2 - 4\alpha - 15\alpha^2 = 0$, or $\alpha_{best} = 0.25540$, for which $M^{max} < wL^2/589$. The optimal solution using the exact moment distribution, however, is quite different. This is an intrinsic weakness of displacement-based FEM since internal forces are obtained by differentiation, which boosts errors. To get a better result a finer mesh would be needed.

In summary, the optimal α from the foregoing criteria varies between 0.255 to 0.306. As a reasonable compromise an engineer could pick $\alpha_{best} \approx 0.28$.

Notes and Bibliography

The Bernoulli-Euler (BE) beam model synthesizes pioneer work by Jacob and Daniel Bernoulli as well as that of Leonhard Euler in the XVIII Century. Although the model was first enunciated by 1750, it was not applied in structural design and analysis until the second half of the XIX Century. While Galileo Galilei is credited with first attempts at a theory, recent studies [42] argue that Leonardo da Vinci made crucial observations a century before Galileo. However, da Vinci lacked Hooke's law and calculus to complete the theory.

A comprehensive source of stiffness and mass matrices of plane and spatial beams is the book by Przemieniecki [575]. The derivation of stiffness matrices is carried out there using differential equilibrium equations rather than energy methods. This was in fact the common practice before 1962, as influenced by the use of transfer matrix methods [553] on the limited memory computers of the time. Results for prismatic elements, however, are identical.

Energy derivations were popularized by Archer [34,35], Martin [450] and Melosh [467,468].

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 12

Variational Formulation of Plane Beam Element

EXERCISE 12.1 [A/C:20] Use (12.17) to derive the element stiffness matrix \mathbf{K}^e of a Hermitian beam element of variable bending rigidity given by the inertia law

$$I(x) = I_1(1 - \frac{x}{\ell}) + I_2 \frac{x}{\ell} = I_1 \frac{1}{2}(1 - \xi) + I_2 \frac{1}{2}(1 + \xi). \quad (\text{E12.1})$$

Use of *Mathematica* or similar CAS tool is recommended since the integrals are time consuming and error prone. *Mathematica* hint: write

$$\text{EI} = \text{EI1}*(1-\xi)/2 + \text{EI2}*(1+\xi)/2; \quad (\text{E12.2})$$

and keep EI inside the argument of `Integrate`. Check whether you get back (12.20) if $\text{EI}=\text{EI1}=\text{EI2}$. If you use *Mathematica*, this check can be simply done after you got and printed the tapered beam \mathbf{K}^e , by writing `ClearAll[EI]; Ke=Simplify[Ke/.{EI1->EI,EI2->EI}];` and printing this matrix.⁴

EXERCISE 12.2 [A/C:20] Use (12.18) to derive the consistent node force vector \mathbf{f}^e for a Hermitian beam element under linearly varying transverse load q defined by

$$q(x) = q_1(1 - \frac{x}{\ell}) + q_2 \frac{x}{\ell} = q_1 \frac{1}{2}(1 - \xi) + q_2 \frac{1}{2}(1 + \xi). \quad (\text{E12.3})$$

Again use of a CAS is recommended, particularly since the polynomials to be integrated are quartic in ξ , and hand computations are error prone. *Mathematica* hint: write

$$q = q1*(1-\xi)/2 + q2*(1+\xi)/2; \quad (\text{E12.4})$$

and keep q inside the argument of `Integrate`. Check whether you get back (12.21) if $q_1 = q_2 = q$ (See previous Exercise for *Mathematica* procedural hints).

EXERCISE 12.3 [A:20] Obtain the consistent node force vector \mathbf{f}^e of a Hermitian beam element subject to a transverse point load P at abscissa $x = a$ where $0 \leq a \leq \ell$. Use the Dirac's delta function expression $q(x) = P \delta(a)$ and the fact that for any continuous function $f(x)$, $\int_0^\ell f(x) \delta(a) dx = f(a)$ if $0 \leq a \leq \ell$. Check the special cases $a = 0$ and $a = \ell$.

EXERCISE 12.4 [A:25] Derive the consistent node force vector \mathbf{f}^e of a Hermitian beam element subject to a linearly varying z -moment m per unit length, positive CCW, defined by the law $m(x) = m_1(1 - \xi)/2 + m_2(1 + \xi)/2$. Use the fact that the external work per unit length is $m(x)\theta(x) = m(x) v'(x) = (\mathbf{u}^e)^T (d\mathbf{N}/dx)^T m(x)$. For arbitrary $m(x)$ show that this gives

$$\mathbf{f}^e = \int_0^\ell \frac{\partial \mathbf{N}^T}{\partial x} m dx = \int_{-1}^1 \frac{\partial \mathbf{N}^T}{\partial \xi} \frac{2}{\ell} m \frac{1}{2} \ell d\xi = \int_{-1}^1 \mathbf{N}_\xi^T m d\xi, \quad (\text{E12.5})$$

where \mathbf{N}_ξ^T denote the column vectors of beam shape function derivatives with respect to ξ . Can you see a shortcut that avoids the integral altogether if m is constant?

EXERCISE 12.5 [A:20] Obtain the consistent node force vector \mathbf{f}^e of a Hermitian beam element subject to a concentrated moment ("point moment", positive CCW) C applied at $x = a$. Use the Concentrated moment load on beam element expression (E12.5) in which $m(x) = C \delta(a)$, where $\delta(a)$ denotes the Dirac's delta function at $x = a$. Check the special cases $a = 0$, $a = \ell$ and $a = \ell/2$.

⁴ `ClearAll[EI]` discards the previous definition (E12.2) of EI; the same effect can be achieved by writing $\text{EI} = .$ (dot).

EXERCISE 12.6 [A/C:25] Consider the one-dimensional Gauss integration rules.⁵

$$\text{One point : } \int_{-1}^1 f(\xi) d\xi \doteq 2f(0). \quad (\text{E12.6})$$

$$\text{Two points: } \int_{-1}^1 f(\xi) d\xi \doteq f(-1/\sqrt{3}) + f(1/\sqrt{3}). \quad (\text{E12.7})$$

$$\text{Three points: } \int_{-1}^1 f(\xi) d\xi \doteq \frac{5}{9}f(-\sqrt{3/5}) + \frac{8}{9}f(0) + \frac{5}{9}f(\sqrt{3/5}). \quad (\text{E12.8})$$

Try each rule on the monomial integrals

$$\int_{-1}^1 d\xi, \quad \int_{-1}^1 \xi d\xi, \quad \int_{-1}^1 \xi^2 d\xi, \quad \dots \quad (\text{E12.9})$$

until the rule fails. In this way verify that rules (E12.6), (E12.7) and (E12.8) are exact for polynomials of degree up to 1, 3 and 5, respectively. (*Labor-saving hint*: for odd monomial degree no computations need to be done; why?).

EXERCISE 12.7 [A/C:25] Repeat the derivation of Exercise 12.1 using the two-point Gauss rule (E12.7) to evaluate integrals in ξ . A CAS is recommended. If using *Mathematica* you may use a function definition to save typing. For example to evaluate $\int_{-1}^1 f(\xi) d\xi$ in which $f(\xi) = 6\xi^4 - 3\xi^2 + 7$, by the 3-point Gauss rule (E12.8), say

```
f[ξ_]:=6ξ^4-3ξ^2+7; int=Simplify[(5/9)*(f[-Sqrt[3/5]]+f[Sqrt[3/5]])+(8/9)*f[0]];
```

and print `int`. To form an element by Gauss integration define matrix functions in terms of ξ , for example `Be[ξ_]`, or use the substitution operator `/.`, whatever you prefer. Check whether one obtains the same answers as with analytical integration, and explain why there is agreement or disagreement. Hint for the explanation: consider the order of the ξ polynomials you are integrating over the element.

EXERCISE 12.8 [A/C:25] As above but for Exercise 12.2.

EXERCISE 12.9 [A/C:30] Derive the Bernoulli-Euler beam stiffness matrix (12.20) using the method of differential equations. To do this integrate the homogeneous differential equation $EI v'''' = 0$ four times over a cantilever beam clamped at node 1 over $x \in [0, \ell]$ to get $v(x)$. The process yields four constants of integration C_1 through C_4 , which are determined by matching the two zero-displacement BCs at node 1 and the two force BCs at node 2. This provides a 2×2 flexibility matrix relating forces and displacements at node j . Invert to get a deformational stiffness, and expand to 4×4 by letting node 1 translate and rotate.

EXERCISE 12.10 [C:20] Using *Mathematica*, repeat Example 12.2 but using EbE lumping of the distributed force q . (It is sufficient to set the nodal moments on the RHS of (12.26) to zero.) Is v_2 the same as the exact analytical solution? If not, study the ratio v_2/v_2^{exact} as function of α , and draw conclusions.

EXERCISE 12.11 [C:25] For the continuous beam of Example 12.3, verify the results given there for the optimal α that minimizes the maximum relative deflection. Plot the deflection profile when $\alpha = \alpha_{best}$.

EXERCISE 12.12 [C:25] For the continuous beam of Example 12.3, verify the results given there for the optimal α that minimizes the absolute bending moment. Plot the moment diagram when $\alpha = \alpha_{best}$.

⁵ Gauss integration is studied further in Chapter 17.

13

Advanced One-Dimensional Elements

TABLE OF CONTENTS

	Page
§13.1. Introduction	13–3
§13.2. Generalized Interpolation	13–3
§13.2.1. Legendre Polynomials	13–3
§13.2.2. Generalized Stiffnesses	13–4
§13.2.3. Transforming to Physical Freedoms: BE Model	13–4
§13.2.4. Transforming to Physical Freedoms: Shear-Flexible Model	13–5
§13.2.5. Hinged Plane Beam Element	13–5
§13.2.6. Timoshenko Plane Beam Element	13–6
§13.2.7. Shear-Curvature Recovery	13–7
§13.2.8. Beam on Elastic Supports	13–9
§13.3. Interpolation with Homogeneous ODE Solutions	13–11
§13.3.1. Exact Winkler/BE-Beam Stiffness	13–11
§13.4. Equilibrium Theorems	13–16
§13.4.1. Self-Equilibrating Force System	13–16
§13.4.2. Handling Applied Forces	13–16
§13.4.3. Flexibility Equations	13–17
§13.4.4. Rigid Motion Injection	13–19
§13.4.5. Applications	13–19
§13.5. Flexibility Based Derivations	13–20
§13.5.1. Timoshenko Plane Beam-Column	13–20
§13.5.2. Plane Circular Arch in Local System	13–21
§13.5.3. Plane Circular Arch in Global System	13–24
§13.6. *Accuracy Analysis	13–27
§13.6.1. *Accuracy of Bernoulli-Euler Beam Element	13–27
§13.6.2. *Accuracy of Timoshenko Beam Element	13–29
§13. Notes and Bibliography.	13–30
§13. References	13–31
§13. Exercises	13–32

§13.1. Introduction

This Chapter develops special one-dimensional elements, such as thick beams, arches and beams on elastic foundations, that require mathematical and modeling resources beyond those presented in Chapters 11–12. The techniques used are less elementary,¹ and may be found in books on Advanced Mechanics of Materials, e.g. [82,91]. Readers are expected to be familiar with ordinary differential equations and energy methods. The Chapter concludes with beam accuracy analysis based on the modified equation method.

All of this Chapter material would be normally bypassed in an introductory finite element course. It is primarily provided for offerings at an intermediate level, for example a first graduate FEM course in Civil Engineering. Such courses may skip most of Part I as being undergraduate material.

§13.2. Generalized Interpolation

For derivation of special and C^0 beam elements it is convenient to use a transverse-displacement cubic interpolation in which the nodal freedoms v_1, v_2, θ_1 and θ_2 are replaced by *generalized coordinates* c_1 to c_4 :

$$v(\xi) = N_{c1} c_1 + N_{c2} c_2 + N_{c3} c_3 + N_{c4} c_4 = \mathbf{N}_c \mathbf{c}. \quad (13.1)$$

Here $N_{ci}(\xi)$ are *generalized shape functions* that satisfy the completeness requirement discussed in Chapter 19. \mathbf{N}_c is a 1×4 matrix whereas \mathbf{c} is a column 4-vector. Formula (13.1) is a *generalized interpolation*. It includes the Hermite interpolation (12.10–12.12) as an instance when $c_1 = v_1, c_2 = v_1', c_3 = v_2$ and $c_4 = v_2'$.

§13.2.1. Legendre Polynomials

An obvious generalized interpolation is the ordinary cubic polynomial $v(\xi) = c_1 + c_2\xi + c_3\xi^2 + c_4\xi^3$, but this turns out not to be particularly useful. A more seminal expression is

$$v(\xi) = L_1 c_1 + L_2 c_2 + L_3 c_3 + L_4 c_4 = \mathbf{L} \mathbf{c}, \quad (13.2)$$

where the L_i are the first four Legendre polynomials

$$L_1(\xi) = 1, \quad L_2(\xi) = \xi, \quad L_3(\xi) = \frac{1}{2}(3\xi^2 - 1), \quad L_4(\xi) = \frac{1}{2}(5\xi^3 - 3\xi). \quad (13.3)$$

Here c_1 through c_4 have dimension of length. Functions (13.3) and their first two ξ -derivatives are plotted in Figure 13.1. Unlike the shape functions (12.12), the L_i have a clear physical meaning:

- L_1 Translational rigid body mode.
- L_2 Rotational rigid body mode.
- L_3 Constant-curvature deformation mode, symmetric with respect to $\xi = 0$.
- L_4 Linear-curvature deformation mode, antisymmetric with respect to $\xi = 0$.

These properties are also shared by the standard polynomial $c_1 + c_2\xi + c_3\xi^2 + c_4\xi^3$. What distinguishes the set (13.3) are the orthogonality properties

$$\begin{aligned} \mathbf{Q}_0 &= \int_0^\ell \mathbf{L}^T \mathbf{L} dx = \ell \mathbf{diag} [1 \quad 1/3 \quad 1/5 \quad 1/7], \quad \mathbf{Q}_2 = \int_0^\ell (\mathbf{L}'')^T \mathbf{L}'' dx = \frac{48}{\ell^3} \mathbf{diag} [0 \quad 0 \quad 3 \quad 25], \\ \mathbf{Q}_3 &= \int_0^\ell (\mathbf{L}''')^T \mathbf{L}''' dx = \frac{14400}{\ell^5} \mathbf{diag} [0 \quad 0 \quad 0 \quad 1], \quad \text{in which} \quad (.)' \equiv \frac{d(.)}{dx}. \end{aligned} \quad (13.4)$$

\mathbf{Q}_n is called the *covariance matrix* for the n^{th} derivative of the Legendre polynomial interpolation. The first-derivative covariance $\mathbf{Q}_1 = \int_0^\ell (\mathbf{L}')^T \mathbf{L}' dx$ is not diagonal, but this matrix is not used here.

¹ They do not reach, however, the capstone level of Advanced Finite Element Methods [?].

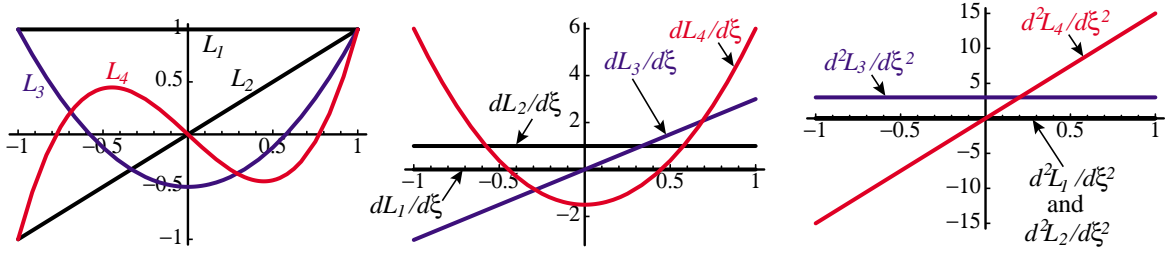


FIGURE 13.1. The Legendre polynomials and their first two ξ -derivatives shown over $\xi \in [-1, 1]$. Those interpretable as beam rigid body modes (L_1 and L_2) in black; deformational modes (L_3 and L_4) in color.

Remark 13.1. The notation (13.2)–(13.3) is FEM oriented. L_1 through L_4 are called P_0 through P_3 in the mathematical literature; e.g. Chapter 22 of the handbook [2]. The general definition for $n = 0, 1 \dots$ is

$$L_{n+1}(\xi) \equiv P_n(\xi) = \sum_{k=0}^n \binom{n}{k} \binom{-n-1}{k} \left(\frac{1-\xi}{2}\right)^k = \frac{1}{2^n} \sum_{k=0}^n \binom{n}{k}^2 (\xi-1)^{n-k} (\xi+1)^k = \frac{1}{2^n} \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{k} \binom{2n-2k}{n} \xi^{n-2k}. \quad (13.5)$$

where $\binom{n}{k}$ is the binomial coefficient. Legendre polynomials are normalized by $P_n(1) = 1$, $P_n(-1) = (-1)^n$. They can also be indirectly defined by generating functions such as

$$\sum_{k=0}^{\infty} P_n(\xi) z^n = \frac{1}{\sqrt{1-2\xi z + z^2}}, \quad \text{or alternatively} \quad \sum_{k=0}^{\infty} \frac{1}{n!} P_n(\xi) z^n = e^{xz} J_0(z\sqrt{1-\xi^2}). \quad (13.6)$$

They can also be defined through a 3-term recurrence relation: $(n+2)P_{n+2}(\xi) - (2n+3)P_{n+1}(\xi) + (n+1)P_n(\xi) = 0$ started with $P_0(\xi) = 1$ and $P_1(\xi) = \xi$. One important application of these polynomials in numerical analysis is the construction of one-dimensional Gauss integration rules: the abscissas of the n -point rule are the zeros of $L_{n+1}(\xi) = P_n(\xi)$.

§13.2.2. Generalized Stiffnesses

The beam stiffness matrix expressed in terms of the c_i is called a *generalized stiffness*. Denote the beam bending and shear rigidities by R_B and R_S , respectively. Then $\mathbf{K}_c = \mathbf{K}_{cB} + \mathbf{K}_{cS}$, where \mathbf{K}_{cB} comes from the bending energy and \mathbf{K}_{cS} from the shear energy. For the latter it is assumed that the mean shear distortion γ at a cross section is $\gamma = \Upsilon \ell^2 v'''$, where Υ is a dimensionless coefficient that depends on the mean-shear model used. Then

$$\mathbf{K}_{cB} = \int_0^\ell R_B (\mathbf{L}'')^T \mathbf{L}'' dx, \quad \mathbf{K}_{cS} = \int_0^\ell R_S \Upsilon^2 \ell^4 (\mathbf{L}''')^T \mathbf{L}''' dx. \quad (13.7)$$

In the case of a Bernoulli-Euler (BE) beam, the shear contribution is dropped: $\mathbf{K}_c = \mathbf{K}_{cB}$. Furthermore if the element is prismatic, $R_B = EI$ is constant. If so $\mathbf{K}_{cS} = R_B$ and $\mathbf{K}_{cB} = EI \mathbf{Q}_2$, where \mathbf{Q}_2 is the second diagonal matrix in (13.4). With view to future use it is convenient to differentiate between symmetric and antisymmetric bending rigidities R_{Bs} and R_{Ba} , which are associated with the responses to modes L_3 and L_4 , respectively. Assuming R_{Bs} and R_{Ba} to be uniform along the element we get

$$\mathbf{K}_c = \mathbf{K}_{cBs} + \mathbf{K}_{cBa}, \quad \mathbf{K}_{cBs} = \frac{144R_{Bs}}{\ell^3} \mathbf{diag}[0 \ 0 \ 1 \ 0], \quad \mathbf{K}_{cBa} = \frac{1200R_{Ba}}{\ell^3} \mathbf{diag}[0 \ 0 \ 0 \ 1], \quad (13.8)$$

If shear flexibility is accounted for, the contribution \mathbf{K}_{cS} of (13.7) is kept. Assuming R_S to be constant over the element, \mathbf{K}_c is split into 3 contributions (two bending and one shear):

$$\mathbf{K}_c = \mathbf{K}_{cBs} + \mathbf{K}_{cBa} + \mathbf{K}_{cS}, \quad \text{with} \quad \mathbf{K}_{cS} = R_S \Upsilon^2 \ell^4 \mathbf{Q}_3 = \frac{14400R_S \Upsilon^2}{\ell} \mathbf{diag}[0 \ 0 \ 0 \ 1]. \quad (13.9)$$

§13.2.3. Transforming to Physical Freedoms: BE Model

For a BE beam model, the generalized coordinates c_i of (13.2) can be connected to the physical DOFs by

$$\begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & 2/\ell & -6/\ell & 12/\ell \\ 1 & 1 & 1 & 1 \\ 0 & 2/\ell & 6/\ell & 12/\ell \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}, \quad \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \frac{1}{60} \begin{bmatrix} 30 & 5\ell & 30 & -5\ell \\ -36 & -3\ell & 36 & -3\ell \\ 0 & -5\ell & 0 & 5\ell \\ 6 & 3\ell & -6 & 3\ell \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix}. \quad (13.10)$$

In compact form: $\mathbf{u}^e = \mathbf{G}_B \mathbf{c}$ and $\mathbf{c} = \mathbf{H}_B \mathbf{u}^e$, with $\mathbf{H}_B = \mathbf{G}_B^{-1}$. Here $\theta_1 \equiv v'_1$ and $\theta_2 \equiv v'_2$, which reflects the fundamental “plane sections remain plane” kinematic assumption of the BE model. The physical stiffness is

$$\mathbf{K}^e = \mathbf{H}_B^T (\mathbf{K}_{cBs} + \mathbf{K}_{cBa}) \mathbf{H}_B = \frac{1}{\ell^3} \begin{bmatrix} 12R_a & 6R_a\ell & -12R_a & 6R_a\ell \\ 6R_a\ell & (3R_a + R_s)\ell^2 & -6R_a\ell & (3R_a - R_s)\ell^2 \\ -12R_a & -6R_a\ell & 12R_a & -6R_a\ell \\ 6R_a\ell & (3R_a - R_s)\ell^2 & -6R_a\ell & (3R_a + R_s)\ell^2 \end{bmatrix}. \quad (13.11)$$

If $R_s = R_a = EI$ the well known stiffness matrix (12.20) is recovered, as can be expected. The additional freedom conferred by (13.11) is exhibited later in two unconventional applications.

§13.2.4. Transforming to Physical Freedoms: Shear-Flexible Model

A shear flexible beam has mean shear distortion $\gamma = \Upsilon \ell^2 v'''$. If Υ is constant and $v(\xi)$ interpolated by (13.2), $v''' = 120 c_4/\ell^3$. Thus $\gamma = 120\Upsilon c_4/\ell$ is *constant* over the element. The end rotational freedoms become $\theta_1 = v'_1 + \gamma$ and $\theta_2 = v'_2 + \gamma$. Using $\Phi = 12\Upsilon$ to simplify the algebra, the transformations (13.10) change to

$$\begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 1 & -1 & 1 & -1 \\ 0 & \frac{2}{\ell} & -\frac{6}{\ell} & \frac{12+10\Phi}{\ell} \\ 1 & 1 & 1 & 1 \\ 0 & \frac{2}{\ell} & \frac{6}{\ell} & \frac{12+10\Phi}{\ell} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}, \quad \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix} = \frac{1}{60} \begin{bmatrix} 30 & 5\ell & 30 & -5\ell \\ -\frac{36+30\Phi}{1+\Phi} & -\frac{3\ell}{1+\Phi} & \frac{36+30\Phi}{1+\Phi} & -\frac{3\ell}{1+\Phi} \\ 0 & -5\ell & 0 & 5\ell \\ -\frac{6}{1+\Phi} & \frac{3\ell}{1+\Phi} & \frac{6}{1+\Phi} & \frac{3\ell}{1+\Phi} \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} \quad (13.12)$$

In compact form, $\mathbf{u}^e = \mathbf{G}_S \mathbf{c}$ and $\mathbf{c} = \mathbf{G}_S^{-1} \mathbf{u}^e = \mathbf{H}_S \mathbf{u}^e$. Transforming \mathbf{K}_c of (13.9) to physical freedoms yields the stiffness used to construct the Timoshenko beam element in §13.2.6:

$$\mathbf{K}^e = \mathbf{H}_S^T \mathbf{K}_c \mathbf{H}_S = \frac{R_{Bs}}{\ell} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} + \frac{12R_{Ba} + R_S \Phi^2 \ell^2}{4\ell^3 (1 + \Phi)^2} \begin{bmatrix} 4 & 2\ell & -4 & 2\ell \\ 2\ell & \ell^2 & -2\ell & \ell^2 \\ -4 & -2\ell & 4 & -2\ell \\ 2\ell & \ell^2 & -2\ell & \ell^2 \end{bmatrix}. \quad (13.13)$$

§13.2.5. Hinged Plane Beam Element

The two-node prismatic plane BE beam element depicted in Figure 13.2 has a mechanical hinge at midspan ($\xi = 0$). The cross sections on both sides of the hinge can rotate respect to each other. The top figure also sketches a fabrication method sometimes used in short-span pedestrian bridges. Gaps on either side of the hinged section cuts are filled with a bituminous material that permits slow relative rotations.

Both the curvature κ and the bending moment M must vanish at midspan. But in a element built via cubic interpolation of $v(x)$, $\kappa = v''$ must vary linearly in both ξ and x .

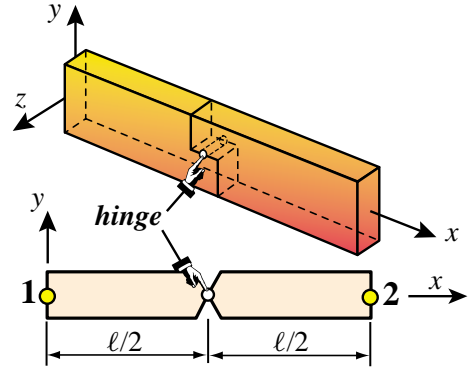


FIGURE 13.2. Beam element with hinge located at midspan. The top figure sketches a hinge fabrication method.

Consequently the mean curvature, which is controlled by the Legendre function L_2 (shown in blue on Figure 13.1) must be zero. The kinematic constraint of zero mean curvature is enforced by setting the symmetric bending rigidity $R_{Bs} = 0$ whereas the antisymmetric bending rigidity is the normal one: $R_{Ba} = EI$.

Plugging into (13.11) yields

$$\mathbf{K}^e = \frac{3EI}{\ell^3} \begin{bmatrix} 4 & 2\ell & -4 & 2\ell \\ 2\ell & \ell^2 & -2\ell & \ell^2 \\ -4 & -2\ell & 4 & -2\ell \\ 2\ell & \ell^2 & -2\ell & \ell^2 \end{bmatrix} = \frac{3EI}{\ell^3} [2 \ \ell \ -2 \ \ell] \begin{bmatrix} 2 \\ \ell \\ -2 \\ \ell \end{bmatrix}. \quad (13.14)$$

This matrix has rank one, as it can be expected from the last (dyadic) expression in (13.14). \mathbf{K}^e has one nonzero eigenvalue: $6EI(4 + \ell^2)/\ell^3$, and three zero eigenvalues. The eigenvector associated with the nonzero eigenvalue pertains² Matrix (13.14) can be derived by more sophisticated methods (e.g., mixed variational principles) but the present technique is the most expedient one.

Example 13.1.

This example deals with the prismatic continuous beam shown in Figure 13.3(a). This has two spans with lengths αL and L , respectively, where α is a design parameter, and is subjected to uniform load q_0 . The beam is free, simple supported and fixed at nodes 1, 2 and 3, respectively. There is a hinge at the center of the 2–3 span. Of interest is the design question: for which $\alpha > 0$ is the tip deflection at end 1 zero?

The beam is discretized with two elements: (1) and (2), going from 1 to 2 and 2 to 3, respectively, as shown in the figure. The stiffnesses for elements (1) and (2) are those of (12.20) and (13.14), respectively, whereas (12.21) is used to build the consistent node forces for both elements.

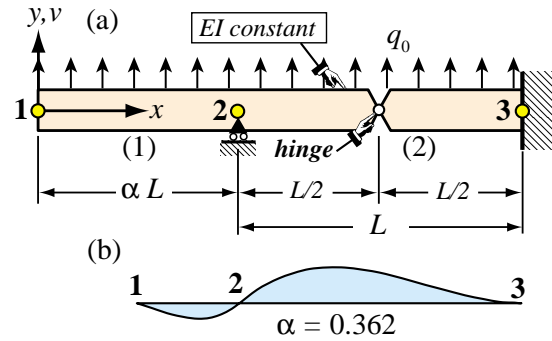


FIGURE 13.3. Beam problem for Example 13.1. (a): beam problem, (b) deflection profile when $\alpha = 0.362$.

Assembling and applying the support conditions $v_2 = v_3 = \theta_3 = 0$, provides the reduced stiffness equations

$$\frac{EI}{L^3} \begin{bmatrix} 12/\alpha^3 & 6L/\alpha^2 & 6L/\alpha^2 \\ 6L/\alpha^2 & 4L^2/\alpha & 2L^2/\alpha \\ 6L/\alpha^2 & 2L^2/\alpha & L^2(4 + 3\alpha)/\alpha \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \frac{q_0 L}{2} \begin{bmatrix} \alpha \\ L\alpha^2/6 \\ L(1 - \alpha^2)/6 \end{bmatrix}. \quad (13.15)$$

Solving for the node displacements gives $v_1 = q_0 L^4 \alpha (12\alpha^2 + 9\alpha^3 - 2)/(72EI)$, $\theta_1 = q_0 L^3 (1 - 6\alpha^2 - 6\alpha^3)/(36EI)$ and $\theta_2 = q_0 L^3 (1 - 6\alpha^2)/(36EI)$. The equation $v_1 = 0$ is quartic in α and has four roots, which to 8 places are $\alpha_1 = -1.17137898$, $\alpha_2 = -0.52399776$, $\alpha_3 = 0$, and $\alpha_4 = 0.3620434$. Since the latter is the only positive root, the solution is $\alpha = 0.362$. The deflection profile for this value is pictured in Figure 13.3(b).

§13.2.6. Timoshenko Plane Beam Element

As observed in §12.2.2, the Timoshenko beam model [657] includes a first order correction for transverse shear flexibility. The key kinematic assumption changes to “plane sections remain plane but not necessarily normal to the deformed neutral surface.” This is illustrated in Figure 13.4(a) for a 2-node plane beam element. The cross section rotation θ differs from v' by γ . Ignoring axial forces, the displacement field is analogous to that of the Bernoulli-Euler model (12.2) but with a shear correction:

$$u(x, y) = -y\theta, \quad v(x, y) = v(x), \quad \text{with} \quad \theta = \frac{\partial v}{\partial x} + \gamma = v' + \gamma, \quad \gamma = \frac{V}{GA_s}. \quad (13.16)$$

² Compare the vector in the last expression in (13.14) to the last row of \mathbf{H}_B in (13.10) to the antisymmetric deformational mode L_3 .

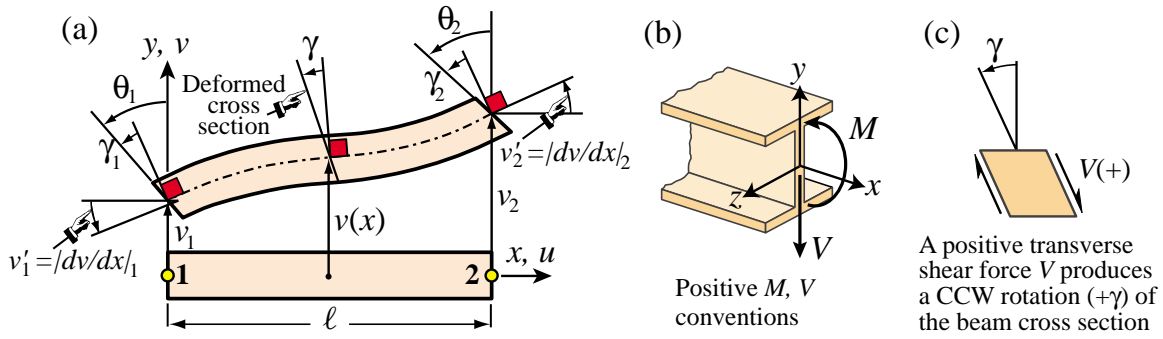


FIGURE 13.4. Two-node Timoshenko plane beam element: (a) kinematics (when developed with cubic shape functions, $\gamma_1 = \gamma_2 = \gamma$); (b) M and V sign conventions; (c) concurrence of sign conventions for V and γ .

Here V is the transverse shear force, γ the “shear rotation” (positive CCW) averaged over the cross section, G the shear modulus and A_s the effective shear area.³ The product $R_s = GA_s$ is the *shear rigidity*. To correlate with the notation of §13.2.4, note that $V = EI v'''$, $\gamma \stackrel{\text{def}}{=} \Upsilon \ell^2 v''' = V/(GA_s)$, so $\Upsilon = EI/(GA_s \ell^2)$ and

$$\Phi = 12\Upsilon = \frac{12EI}{GA_s \ell^2} \quad (13.17)$$

This dimensionless ratio characterizes the “shear slenderness” of the beam element.⁴ It is *not* an intrinsic beam property because it involves the element length. As $\Phi \rightarrow 0$ the Timoshenko model reduces to the BE model. Replacing $R_{Bs} = R_{Ba} = EI$ and $R_s = GA_s = 12EI/(\Phi \ell^2)$ into (13.13) yields the Timoshenko beam stiffness

$$\mathbf{K}^e = \frac{EI}{\ell^3(1+\Phi)} \begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & \ell^2(4+\Phi) & -6\ell & \ell^2(2-\Phi) \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & \ell^2(2-\Phi) & -6\ell & \ell^2(4+\Phi) \end{bmatrix} \quad (13.18)$$

If $\Phi = 0$ this reduces to (12.20). The *Mathematica* module `TimoshenkoBeamStiffness[Le,EI,Phi]`, listed in Figure 13.5, implements (13.18).

```
TimoshenkoBeamStiffness[Le_,EI_,Phi_]:=Module[{Ke},
  Ke=EI/(Le*(1+Phi))*{{ 12/Le^2, 6/Le,-12/Le^2, 6/Le },
    { 6/Le, 4+Phi,-6/Le, 2-Phi },
    {-12/Le^2, -6/Le, 12/Le^2,-6/Le},
    { 6/Le, 2-Phi,-6/Le, 4+Phi }};
  Return[Ke];
```

FIGURE 13.5. Module to produce stiffness matrix for Timoshenko beam element.

The calculation of the consistent node forces for uniform transverse load is covered in Exercise 13.2. A hinged Timoshenko beam is constructed in Exercise 13.3.

³ A concept defined in Mechanics of Materials; see e.g. Chapter 10 of Popov [529] or Chapter 12 of Timoshenko and Goodier [659]. A_s is calculated by equating the internal shear energy $\frac{1}{2} V \gamma = \frac{1}{2} V^2/(GA_s)$ to that produced by the shear stress distribution over the cross section. For a thin rectangular cross section and zero Poisson’s ratio, $A_s = 5A/6$.

⁴ Note that in (13.8)–(13.9), $1200R_{Bs}/\ell^3 + 14400R_s\Upsilon^2/\ell = 1200EI(1+\Phi)/\ell^3$, giving a simple interpretation for Φ .

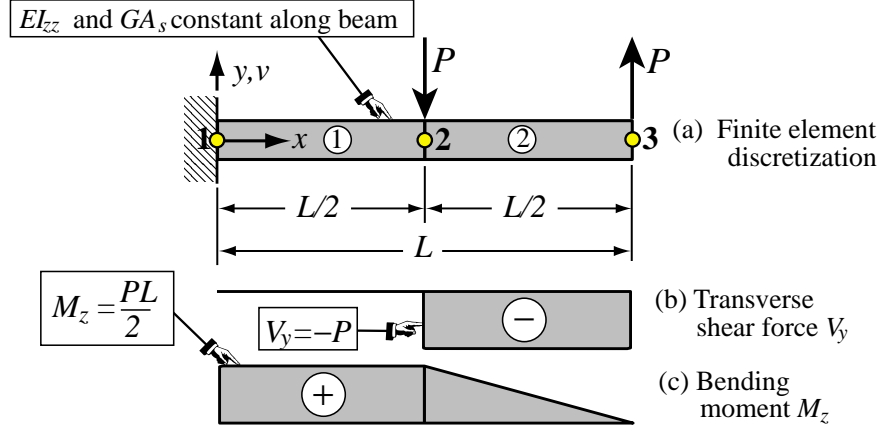


FIGURE 13.6. Example: cantilever beam discretized with two Timoshenko beam elements.

§13.2.7. Shear-Curvature Recovery

When using the Timoshenko beam model, the following arises during postprocessing. Suppose that the element node displacement vector $\mathbf{u}^e = \mathbf{u}_T^e$ is given following the solution process. Recover the mean shear distortion γ^e and the curvature κ^e over the element on the way to internal forces and stresses. The problem is not trivial because γ^e is part of the rotational freedoms. The recovery process can be effectively done by passing first to generalized coordinates: $\mathbf{c}^e = \mathbf{H}_S \mathbf{u}^e$, and then to Bernoulli-Euler node displacements: $\mathbf{u}_{BE}^e = \mathbf{G}_B \mathbf{c}^e = \mathbf{G}_B \mathbf{H}_S \mathbf{u}^e = \mathbf{T}_{BT}^e \mathbf{u}_T^e$, in which

$$\mathbf{T}_{BT} = -\mathbf{G}_B \mathbf{H}_S = \mathbf{I} - \frac{\Phi}{1 + \Phi} \begin{bmatrix} 0 & 0 & 0 & 0 \\ \frac{1}{L^e} & \frac{1}{2} & \frac{1}{L^e} & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ \frac{1}{L^e} & \frac{1}{2} & \frac{1}{L^e} & \frac{1}{2} \end{bmatrix} \quad (13.19)$$

Subtracting: $\mathbf{u}_\gamma^e \stackrel{\text{def}}{=} \mathbf{u}_T^e - \mathbf{u}_{BE}^e = (\mathbf{I} - \mathbf{G}_B \mathbf{H}_S) \mathbf{u}_T^e = [0 \ \gamma^e \ 0 \ \gamma^e]^T$, Explicitly

$$\gamma^e = \frac{1}{L^e} (v_1^e - v_2^e) + \frac{1}{2} (\theta_2^e + \theta_1^e) \quad (13.20)$$

The curvature is obtained from the Bernoulli-Euler vector: $\kappa = \mathbf{B}^e \mathbf{u}_{BE}^e$, where the curvature displacement matrix \mathbf{B}^e is that given in the previous Chapter.

Example 13.2. Consider the prismatic cantilever beam of length L pictured in Figure 13.6(a). It is subject to two point loads as shown. Shear flexibility is to be accounted for using the Timoshenko model. The bending and shear rigidities EI_{zz} and GA_s are constant along the span. The objective is to find deflections, curvatures and shear distortions and associated bending moments and shear forces.

It is sufficient to discretize the beam with two Timoshenko beam elements of length $L/2$ as shown in the figure. The stiffness matrices for both elements are given by (13.18), in which $L^e = L/2$ and $\Phi = 12 EI_{zz} / (GA_s (L/2)^2) = 48 EI_{zz} / (GA_s L^2)$. The master stiffness equations are

$$\frac{2EI_{zz}}{L^3(1+\Phi)} \begin{bmatrix} 48 & 12L & -48 & 12L & 0 & 0 \\ 12L & L^2(4+\Phi) & -12L & L^2(2-\Phi) & 0 & 0 \\ -48 & -12L & 96 & 0 & -48 & 12L \\ 12L & L^2(2-\Phi) & 0 & 2L^2(4+\Phi) & -12L & L^2(2-\Phi) \\ 0 & 0 & -48 & -12L & 48 & -12L \\ 0 & 0 & 12L & L^2(2-\Phi) & -12L & L^2(4+\Phi) \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \\ v_3 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -P \\ 0 \\ P \\ 0 \end{bmatrix}. \quad (13.21)$$

Setting the displacement B.C. $v_1 = \theta_1 = 0$ and solving yields

$$\mathbf{u} = \frac{PL^2}{4EI_{zz}} \begin{bmatrix} 0 & 0 & \frac{L}{4} & 1 & \frac{L}{24}(22+\Phi) & \frac{3}{2} \end{bmatrix}^T. \quad (13.22)$$

The mean element shear distortions are calculated from (13.22) using (13.20). This gives

$$\gamma^{(1)} = 0, \quad \gamma^{(2)} = -\frac{P L^2 \Phi}{48 E I_{zz}} = -\frac{P}{G A_s} \quad (13.23)$$

The element-level Bernoulli-Euler node displacements are obtained from (13.22) on subtracting the shear distortions (13.23) from the rotations:

$$\mathbf{u}_B^{(1)} = \frac{P L^2}{4 E I_{zz}} \begin{bmatrix} 0 & 0 & \frac{L}{4} & 1 \end{bmatrix}^T, \quad \mathbf{u}_B^{(2)} = \frac{P L^2}{4 E I_{zz}} \begin{bmatrix} \frac{L}{4} & 1 + \frac{\Phi}{12} & \frac{L}{24} (22 + \Phi) & \frac{3}{2} + \frac{\Phi}{12} \end{bmatrix}^T. \quad (13.24)$$

Note that $\theta_{B2}^{(1)} = 1 \neq \theta_{B2}^{(2)} = 1 + \frac{1}{12} \Phi$, the “kink” being due to the shear distortion jump at node 2. The curvatures are now recovered as

$$\kappa^{(1)} = \mathbf{B}^{(1)} \mathbf{u}_B^{(1)} = \frac{P L}{2 E I_{zz}}, \quad \kappa^{(2)} = \mathbf{B}^{(2)} \mathbf{u}_B^{(2)} = \frac{P L}{4 E I_{zz}} (1 - \xi^{(2)}), \quad (13.25)$$

The transverse shear force resultant and bending moment are easily recovered as $V_y^e = G A_s \gamma^{(e)}$ and $M_z^e = E I_{zz} \kappa^e$, respectively. The results are drawn in Figure 13.6(b,c).

§13.2.8. Beam on Elastic Supports

Sometimes beams, as well as other structural members, may be supported elastically along their span. Two common configurations that occur in structural engineering are:

- (i) Beam resting on a continuum medium such as soil. This is the case in foundations.
- (ii) Beam supported by discrete but closely spaced flexible supports, as in the “bed of springs” pictured in Figure 13.7. This occurs in railbeds (structurally rails are beams supported by crossties) and some types of grillworks.

The *Winkler foundation* is a simplified elastic-support model. It is an approximation for (i) because it ignores multidimensional elasticity effects as well as friction. It is a simplification of (ii) because the discrete nature of supports is smeared out. It is nonetheless popular, particularly in foundation and railway engineering, when the presence of physical uncertainties would not justify a more complicated model. Such uncertainties are inherent in soil mechanics.

The Winkler model may be viewed as a “continuification” of case (ii). Take a beam slice going from x to $x + dx$. The spring-reaction force acting on the beam is taken to be $df_F = -k_F v(x) dx$. Here $v(x)$ is the transverse deflection and k_F the Winkler foundation stiffness, which has dimension of force per length-squared. Force df_F has the opposite sign of $v(x)$, pushing up if the beam moves down and pulling down if it moves up. Beam-foundation separation effects that may occur in case (i) are ignored here because that would lead to a nonlinear contact problem.

The internal energy stored in the dx slice of Winkler springs is $-1/2 v df_F = 1/2 k_F v^2 dx$. Consequently the effect of elastic supports is to modify the internal energy U_B^e of the beam element so that it becomes

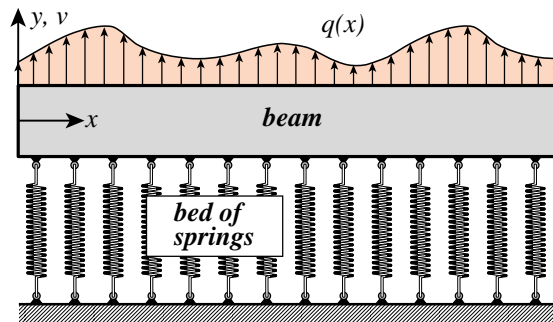


FIGURE 13.7. A beam supported by a bed of springs. Continuification of this configuration leads to the Winkler foundation model treated in this subsection.

$$U^e = U_B^e + U_F^e, \quad \text{with} \quad U_F^e = \frac{1}{2} \int_0^\ell k_F v^2 dx. \quad (13.26)$$

Therefore the total stiffness of the element is computed by *adding the foundation stiffness to the beam stiffness*. Care must be taken, however, that the *same set of nodal freedoms is used*. This is best handled by doing the generalized stiffness \mathbf{K}_{cF} first, and then using the appropriate generalized-to-physical transformation. If the transverse deflection v is interpolated with (13.2) as $v = \mathbf{L} \mathbf{c}$, the generalized Winkler foundation stiffness for constant k_F is

$$\mathbf{K}_{cF} = k_F \int_0^\ell \mathbf{L}^T \mathbf{L} dx = k_F \mathbf{Q}_0, \quad (13.27)$$

where \mathbf{Q}_0 is the first diagonal matrix in (13.4). This holds regardless of beam model. Now if the member resting on the foundation is modeled as a BE beam, one picks \mathbf{H}_B of (13.10) as generalized-to-physical transformation matrix to get

$$\mathbf{K}_F^e = k_F \mathbf{H}_B^T \mathbf{Q}_0 \mathbf{H}_B = \frac{k_F \ell}{420} \begin{bmatrix} 156 & 22\ell & 54 & -13\ell \\ 22\ell & 4\ell^2 & 13\ell & -3\ell^2 \\ 54 & 13\ell & 156 & -22\ell \\ -13\ell & -3\ell^2 & -22\ell & 4\ell^2 \end{bmatrix}, \quad (13.28)$$

If instead the supported member is modeled as a Timoshenko beam, one picks \mathbf{H}_S of (13.12) to get

$$\begin{aligned} \mathbf{K}_F^e &= k_F \mathbf{H}_S^T \mathbf{Q}_0 \mathbf{H}_S \\ &= \frac{k_F \ell}{840(1+\Phi)^2} \begin{bmatrix} 4(78+147\Phi+70\Phi^2) & \ell(44+77\Phi+35\Phi^2) & 4(27+63\Phi+35\Phi^2) & -\ell(26+63\Phi+35\Phi^2) \\ \ell(44+77\Phi+35\Phi^2) & \ell^2(8+14\Phi+7\Phi^2) & \ell(26+63\Phi+35\Phi^2) & -\ell^2(6+14\Phi+7\Phi^2) \\ 4(27+63\Phi+35\Phi^2) & \ell(26+63\Phi+35\Phi^2) & 4(78+147\Phi+70\Phi^2) & -\ell(44+77\Phi+35\Phi^2) \\ -\ell(26+63\Phi+35\Phi^2) & -\ell^2(6+14\Phi+7\Phi^2) & -\ell(44+77\Phi+35\Phi^2) & \ell^2(8+14\Phi+7\Phi^2) \end{bmatrix} \end{aligned} \quad (13.29)$$

The module `TimoshenkoWinklerStiffness[Le,kF,Φ]` listed in Figure 13.8 implements the stiffness (13.29). To get the BE-beam Winkler stiffness (13.28), invoke with $\Phi = 0$. Examples of use of this module are provided in §13.3.1.

```
TimoshenkoWinklerStiffness[Le_,kF_,Φ_]:=Module[{KeW},
  KeW={ {4*(78+147*Φ+70*Φ^2), Le*(44+77*Φ+35*Φ^2),
        4*(27+63*Φ+35*Φ^2), -Le*(26+63*Φ+35*Φ^2)},
        {Le*(44+77*Φ+35*Φ^2), Le^2*(8+14*Φ+7*Φ^2),
        Le*(26+63*Φ+35*Φ^2), -Le^2*(6+14*Φ+7*Φ^2)},
        {4*(27+63*Φ+35*Φ^2), Le*(26+63*Φ+35*Φ^2),
        4*(78+147*Φ+70*Φ^2), -Le*(44+77*Φ+35*Φ^2)},
        {-Le*(26+63*Φ+35*Φ^2), -Le^2*(6+14*Φ+7*Φ^2),
        -Le*(44+77*Φ+35*Φ^2), Le^2*(8+14*Φ+7*Φ^2)}}*
  kF*Le/(840*(1+Φ)^2); Return[KeW];
```

FIGURE 13.8. Stiffness matrix module for a Winkler foundation supporting a Timoshenko beam element.

§13.3. Interpolation with Homogeneous ODE Solutions

For both BE and Timoshenko beam models, the Legendre polynomials $L_1(\xi)$ through $L_4(\xi)$ are exact solutions of the homogeneous, prismatic, plane beam equilibrium equation $EI d^4v/dx^4 = 0$. When used as shape functions in the generalized interpolation (13.2), the resulting stiffness matrix is exact if the FEM model is loaded at the nodes, as further discussed in §13.6. The technique can be extended to more complicated one-dimensional problems. It can be used to derive exact stiffness matrices if homogeneous solutions are available in closed form, and are sufficiently simple to be amenable to analytical integration. The following subsection illustrates the method for a BE beam resting on a Winkler elastic foundation.

§13.3.1. Exact Winkler/BE-Beam Stiffness

Consider again a prismatic, plane BE beam element resting on a Winkler foundation of stiffness k_F , as pictured in Figure 13.7. The governing equilibrium equation for constant $EI > 0$ and $k_F > 0$ is $EI d^4v/dx^4 + k_F v = q(x)$. The general homogeneous solution over an element of length ℓ going from $x = 0$ to $x = \ell$ is

$$v(x) = e^{\zeta} (c_1 \sin \zeta + c_2 \cos \zeta) + e^{-\zeta} (c_3 \sin \zeta + c_4 \cos \zeta), \quad \text{with } \zeta = \chi x / \ell \text{ and } \chi = \sqrt[4]{\frac{k_F}{4EI}}. \quad (13.30)$$

Here the c_i are four integration constants to be determined from four end conditions: the nodal degrees of freedom v_1, v'_1, v_2 and v'_2 . These *constants are treated as generalized coordinates* and as before collected into vector $\mathbf{c} = [c_1 \ c_2 \ c_3 \ c_4]^T$. The solution (13.30) is used as *generalized interpolation* with $e^{\zeta} \sin \zeta$ through $e^{-\zeta} \cos \zeta$ as the four shape functions. Differentiating twice gives $v' = dv/dx$ and $v'' = d^2v/dx^2$. The TPE functional of the element in terms of the generalized coordinates can be expressed as

$$\Pi_c^e = \int_0^\ell \left(\frac{1}{2} EI (v'')^2 + \frac{1}{2} k_F v^2 - q_0 v \right) dx = \frac{1}{2} \mathbf{c}^T (\mathbf{K}_{cB} + \mathbf{K}_{cF}) \mathbf{c} - \mathbf{c}^T \mathbf{f}_c. \quad (13.31)$$

This defines \mathbf{K}_{cB} and \mathbf{K}_{cF} as generalized element stiffnesses due to beam bending and foundation springs, respectively, whereas \mathbf{f}_c is the generalized force associated with a transverse load $q(x)$. The nodal freedoms are linked to generalized coordinates by

$$\begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 \\ \chi/\ell & \chi/\ell & \chi/\ell & -\chi/\ell \\ e^{\chi} \sin \chi & e^{\chi} \cos \chi & e^{-\chi} \sin \chi & e^{-\chi} \cos \chi \\ \frac{\chi e^{\chi} (\cos \chi + \sin \chi)}{\ell} & \frac{\chi e^{\chi} (\cos \chi - \sin \chi)}{\ell} & \frac{\chi e^{-\chi} (\cos \chi - \sin \chi)}{\ell} & \frac{-\chi e^{-\chi} (\cos \chi + \sin \chi)}{\ell} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \end{bmatrix}. \quad (13.32)$$

In compact form this is $\mathbf{u}^e = \mathbf{G}_F \mathbf{c}$. Inverting gives $\mathbf{c} = \mathbf{H}_F \mathbf{u}^e$ with $\mathbf{H}_F = \mathbf{G}_F^{-1}$. The physical stiffness is $\mathbf{K}^e = \mathbf{K}_B^e + \mathbf{K}_F^e$ with $\mathbf{K}_B^e = \mathbf{H}_F^T \mathbf{K}_{cB} \mathbf{H}_F$ and $\mathbf{K}_F^e = \mathbf{H}_F^T \mathbf{K}_{cF} \mathbf{H}_F$. The consistent force vector is $\mathbf{f}^e = \mathbf{H}_F^T \mathbf{f}_c$. Computation with transcendental functions by hand is unwieldy and error-prone, and at this point it is better to leave that task to a CAS. The *Mathematica* script listed in Figure 13.9 is designed to produce \mathbf{K}_B^e , \mathbf{K}_F^e and \mathbf{f}^e for constant EI and k_F , and uniform transverse load $q(x) = q_0$. The script gives

$$\mathbf{K}_B^e = \frac{EI \chi}{4\ell^3 g^2} \begin{bmatrix} B_1 & B_2 & B_5 & -B_4 \\ & B_3 & B_4 & B_6 \\ & & B_1 & -B_2 \\ \text{symm} & & & B_3 \end{bmatrix}, \quad \mathbf{K}_F^e = \frac{k_F \ell}{16\chi^3 g^2} \begin{bmatrix} F_1 & F_2 & F_5 & -F_4 \\ & F_3 & F_4 & F_6 \\ & & F_1 & -F_2 \\ \text{symm} & & & F_3 \end{bmatrix}, \quad \mathbf{f}^e = \frac{q_0 \ell}{\chi^2 g} \begin{bmatrix} f_1 \\ f_2 \\ f_1 \\ -f_2 \end{bmatrix}, \quad (13.33)$$

```

ClearAll[EI,kF,Le,χ,q0,x]; g=2-Cos[2*χ]-Cosh[2*χ];
Nf={Exp[χ*x/Le]*Sin[χ*x/Le],Exp[χ*x/Le]*Cos[χ*x/Le],
    Exp[-χ*x/Le]*Sin[χ*x/Le],Exp[-χ*x/Le]*Cos[χ*x/Le]};
Nfd=D[Nf,x]; Nfdd=D[Nfd,x];
KgF=kF*Integrate[Transpose[{Nf}].{Nf},{x,0,Le}];
KgB=EI*Integrate[Transpose[{Nfdd}].{Nfdd},{x,0,Le}];
fg= q0*Integrate[Transpose[{Nf}].{x,0,Le}];
{KgF,KgB,fg}=Simplify[{KgF,KgB,fg}];
Print["KgF=",KgF//MatrixForm]; Print["KgB=",KgB//MatrixForm];
GF=Simplify[{Nf/.x->0,Nfd/.x->0,Nf/.x->Le,Nfd/.x->Le}];
HF=Simplify[Inverse[GF]]; HFT=Transpose[HF];
Print["GF=",GF//MatrixForm]; Print["HF=",HF//MatrixForm];
facB=(EI*χ/Le^3)/(4*g^2); facF=(kF*Le)/(16*χ^3*g^2);
KeB=Simplify[HFT.KgB.HF]; KeBfac=FullSimplify[KeB/facB];
Print["KeB=",facB," * ",KeBfac//MatrixForm];
KeF=Simplify[HFT.KgF.HF]; KeFfac=FullSimplify[KeF/facF];
Print["KeF=",facF," * ",KeFfac//MatrixForm];
facf=(q0*Le)/(χ^2*g); fe=Simplify[ExpToTrig[HFT.fg]];
fefac=Simplify[fe/facf]; Print["fe=",facf," * ",fefac];

```

FIGURE 13.9. Script to produce the exact Winkler-BE beam stiffness matrix and consistent force vector.

```

BEBeamWinklerExactStiffness[Le_,EI_,kF_,q0_]:=Module[{B1,B2,B3,B4,B5,B6,
F1,F2,F3,F4,F5,F6,f1,f2,facB,facF,facf,KeB,KeF,fe,χ},
χ=PowerExpand[Le*((kF/(4*EI))^(1/4))];
B1 =2*χ^2*(-4*Sin[2*χ]+Sin[4*χ]+4*Sin[χ]*(Cos[χ]*Cosh[2*χ]+
8*χ*Sin[χ]*Sinh[χ]^2)+2*(Cos[2*χ]-2)*Sinh[2*χ]+Sinh[4*χ]);
B2 =2*Le*χ*(4*Cos[2*χ]-Cos[4*χ]-4*Cosh[2*χ]+Cosh[4*χ]-
8*χ*Sin[2*χ]*Sinh[χ]^2+8*χ*Sin[χ]^2*Sinh[2*χ]);
B3 =-(Le^2*(8*χ*Cos[2*χ]-12*Sin[2*χ]+Cosh[2*χ]*(6*Sin[2*χ]-8*χ)+3*Sin[4*χ]+
2*(6-3*Cos[2*χ]+4*χ*Sin[2*χ])*Sinh[2*χ]-3*Sinh[4*χ]));
B4 =-4*Le*χ*(χ*Cosh[3*χ]*Sin[χ]-χ*Cosh[χ]*(-2*Sin[χ]+Sin[3*χ])+(χ*(Cos[χ]+
Cos[3*χ])+Cosh[2*χ]*(-2*χ*Cos[χ]+4*Sin[χ])+2*(-5*Sin[χ]+Sin[3*χ])*Sinh[χ]);
B5 =-4*χ^2*(2*Cos[χ]*(-2+Cos[2*χ]+Cosh[2*χ])*Sinh[χ]+Sin[3*χ]*
(Cosh[χ]-2*χ*Sinh[χ])+Sin[χ]*(-4*Cosh[χ]+Cosh[3*χ]+2*χ*Sinh[3*χ]));
B6 =2*Le^2*(Cosh[3*χ]*(-2*χ*Cos[χ]+3*Sin[χ])+Cosh[χ]*(2*χ*Cos[3*χ]+3*(Sin[3*χ]-
4*Sin[χ]))+(9*Cos[χ]-3*Cos[3*χ]-6*Cos[χ]*Cosh[2*χ]+16*χ*Sin[χ])*Sinh[χ]);
F1 =2*χ^2*(-32*χ*Sin[χ]^2*Sinh[χ]^2+6*(-2+Cos[2*χ])*
(Sin[2*χ]+Sinh[2*χ])+6*Cosh[2*χ]*(Sin[2*χ]+Sinh[2*χ]));
F2 =2*Le*χ*(4*Cos[2*χ]-Cos[4*χ]-4*Cosh[2*χ]+Cosh[4*χ]+
8*χ*Sin[2*χ]*Sinh[χ]^2-8*χ*Sin[χ]^2*Sinh[2*χ]);
F3 =Le^2*(8*χ*Cos[2*χ]+4*Sin[2*χ]-2*Cosh[2*χ]*(4*χ+Sin[2*χ])-Sin[4*χ]+
2*(Cos[2*χ]+4*χ*Sin[2*χ]-2)*Sinh[2*χ]+Sinh[4*χ]);
F4 =4*Le*χ*(χ*Cosh[3*χ]*Sin[χ]-χ*Cosh[χ]*(-2*Sin[χ]+Sin[3*χ])+(χ*Cos[χ]+
χ*Cos[3*χ]+10*Sin[χ]-2*Cosh[2*χ]*(χ*Cos[χ]+2*Sin[χ])-2*Sin[3*χ])*Sinh[χ]);
F5 =-4*χ^2*(6*Cos[χ]*(-2+Cos[2*χ]+Cosh[2*χ])*Sinh[χ]+Sin[3*χ]*
(3*Cosh[χ]+2*χ*Sinh[χ])+Sin[χ]*(-12*Cosh[χ]+3*Cosh[3*χ]-2*χ*Sinh[3*χ]));
F6 =-2*Le^2*(-(Cosh[3*χ]*(2*χ*Cos[χ]+Sin[χ]))+Cosh[χ]*(2*χ*Cos[3*χ]+
4*Sin[χ]-Sin[3*χ])+(Cos[3*χ]+Cos[χ]*(2*Cosh[2*χ]-3)+16*χ*Sin[χ])*Sinh[χ]);
f1=2*χ*(Cosh[χ]-Cos[χ])*(Sin[χ]-Sinh[χ]); f2=-(Le*(Sin[χ]-Sinh[χ])^2);
g=2-Cos[2*χ]-Cosh[2*χ]; facf=(q0*Le)/(χ^2*g);
facB=(EI*χ/Le^3)/(4*g^2); facF=(kF*Le)/(16*χ^3*g^2);
KeB=facB*{{B1,B2,B5,-B4},{B2,B3,B4,B6},{B5,B4,B1,-B2},{-B4,B6,-B2,B3}};
KeF=facF*{{F1,F2,F5,-F4},{F2,F3,F4,F6},{F5,F4,F1,-F2},{-F4,F6,-F2,F3}};
fe=facf*{f1,f2,f1,-f2}; Return[{KeB,KeF,fe}];

```

FIGURE 13.10. Module to get the exact BE-Winkler stiffness and consistent load vector.

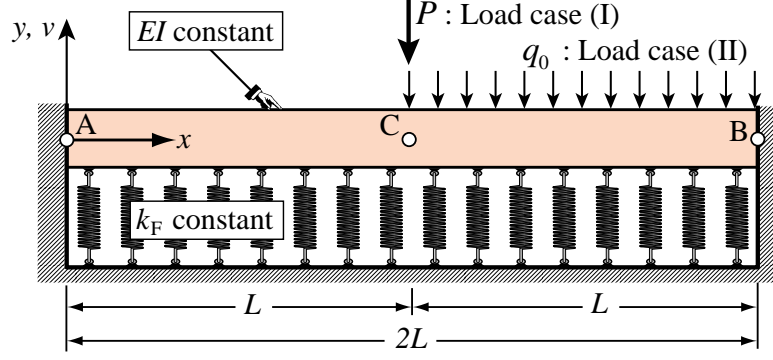


FIGURE 13.11. Example: fixed-fixed beam on Winkler elastic foundation.

in which

$$\begin{aligned}
 g &= 2 - \cos 2\chi - \cosh 2\chi, \\
 B_1 &= 2\chi^2(-4 \sin 2\chi + \sin 4\chi + 4 \sin \chi (\cos \chi \cosh 2\chi + 8\chi \sin \chi \sinh^2 \chi) + 2(\cos 2\chi - 2) \sinh 2\chi + \sinh 4\chi), \\
 B_2 &= 2\ell \chi (4 \cos 2\chi - \cos 4\chi - 4 \cosh 2\chi + \cosh 4\chi - 8\chi \sin 2\chi \sinh^2 \chi + 8\chi \sin^2 \chi \sinh 2\chi), \\
 B_3 &= -(\ell^2(8\chi \cos 2\chi - 12 \sin 2\chi + \cosh 2\chi(6 \sin 2\chi - 8\chi) + 3 \sin 4\chi + \\
 &\quad 2(6 - 3 \cos 2\chi + 4\chi \sin 2\chi) \sinh 2\chi - 3 \sinh 4\chi)), \\
 B_4 &= -4\ell \chi (\chi \cosh 3\chi \sin \chi - \chi \cosh \chi (-2 \sin \chi + \sin 3\chi) + (\chi (\cos \chi + \cos 3\chi) \\
 &\quad + \cosh 2\chi (-2\chi \cos \chi + 4 \sin \chi) + 2(-5 \sin \chi + \sin 3\chi)) \sinh \chi), \\
 B_5 &= -4\chi^2(2 \cos \chi (-2 + \cos 2\chi + \cosh 2\chi) \sinh \chi + \sin 3\chi (\cosh \chi - 2\chi \sinh \chi) \\
 &\quad + \sin \chi (-4 \cosh \chi + \cosh 3\chi + 2\chi \sinh 3\chi)), \\
 B_6 &= 2\ell^2(\cosh 3\chi (-2\chi \cos \chi + 3 \sin \chi) + \cosh \chi (2\chi \cos 3\chi + 3(-4 \sin \chi + \sin 3\chi)) \\
 &\quad + (9 \cos \chi - 3 \cos 3\chi - 6 \cos \chi \cosh 2\chi + 16\chi \sin \chi) \sinh \chi), \\
 F_1 &= 2\chi^2(-32\chi \sin^2 \chi \sinh^2 \chi + 6(-2 + \cos 2\chi)(\sin 2\chi + \sinh 2\chi) + 6 \cosh 2\chi (\sin 2\chi + \sinh 2\chi)), \\
 F_2 &= 2\ell \chi (4 \cos 2\chi - \cos 4\chi - 4 \cosh 2\chi + \cosh 4\chi + 8\chi \sin 2\chi \sinh^2 \chi - 8\chi \sin^2 \chi \sinh 2\chi), \\
 F_3 &= \ell^2(8\chi \cos 2\chi + 4 \sin 2\chi - 2 \cosh 2\chi (4\chi + \sin 2\chi) - \sin 4\chi + 2(\cos 2\chi + 4\chi \sin 2\chi - 2) \sinh 2\chi + \sinh 4\chi), \\
 F_4 &= 4\ell \chi (\chi \cosh 3\chi \sin \chi - \chi \cosh \chi (\sin 3\chi - 2 \sin \chi) + (\chi \cos \chi + \chi \cos 3\chi + 10 \sin \chi \\
 &\quad - 2 \cosh 2\chi (\chi \cos \chi + 2 \sin \chi) - 2 \sin 3\chi) \sinh \chi), \\
 F_5 &= -4\chi^2(6 \cos \chi (-2 + \cos 2\chi + \cosh 2\chi) \sinh \chi + \sin 3\chi (3 \cosh \chi + 2\chi \sinh \chi) \\
 &\quad + \sin \chi (-12 \cosh \chi + 3 \cosh 3\chi - 2\chi \sinh 3\chi)), \\
 F_6 &= -2\ell^2(-(\cosh 3\chi (2\chi \cos \chi + \sin \chi)) + \cosh \chi (2\chi \cos 3\chi + 4 \sin \chi - \sin 3\chi) \\
 &\quad + (\cos 3\chi + \cos \chi (2 \cosh 2\chi - 3) + 16\chi \sin \chi) \sinh \chi), \\
 f_1 &= 2\chi (\cosh \chi - \cos \chi) (\sin \chi - \sinh \chi), \quad f_2 = -\ell (\sin \chi - \sinh \chi)^2.
 \end{aligned} \tag{13.34}$$

These expressions are used to code module `BEBeamWinklerExactStiffness[Le,EI,kF,q0]`, which is listed in Figure 13.10.

Example 13.3. A fixed-fixed BE beam rests on a Winkler foundation as shown in Figure 13.11. The beam has span $2L$, and constant EI . The Winkler foundation coefficient k_F is constant. As usual in foundation engineering we set

$$k_F = EI \lambda^4 / L^4, \tag{13.35}$$

Table 13.1 - Results for Example of Figure 13.11 at Selected λ Values

λ	Load case (I): Central Point Load				Load case (II): Line Load Over Right Half			
	exact C_I	$N_e = 2$	$N_e = 4$	$N_e = 8$	exact C_{II}	$N_e = 2$	$N_e = 4$	$N_e = 8$
0.1	0.999997	0.999997	0.999997	0.999997	0.999997	0.999997	0.999997	0.999997
1	0.970005	0.969977	0.970003	0.970005	0.968661	0.969977	0.968742	0.968666
2	0.672185	0.668790	0.671893	0.672167	0.657708	0.668790	0.658316	0.657746
5	0.067652	0.049152	0.065315	0.067483	0.041321	0.049152	0.041254	0.041317
10	0.008485	0.003220	0.006648	0.008191	0.002394	0.003220	0.002393	0.002395
100	8.48×10^{-6}	3.23×10^{-7}	8.03×10^{-7}	1.63×10^{-6}	2.40×10^{-7}	3.23×10^{-7}	2.62×10^{-7}	2.42×10^{-7}

where λ is a dimensionless rigidity to be kept as parameter.⁵ The beam is subjected to two load cases: (I) a central point load P at $x = L$, and (II) a uniform line load q_0 over the right half $x \geq L$. See Figure 13.11.

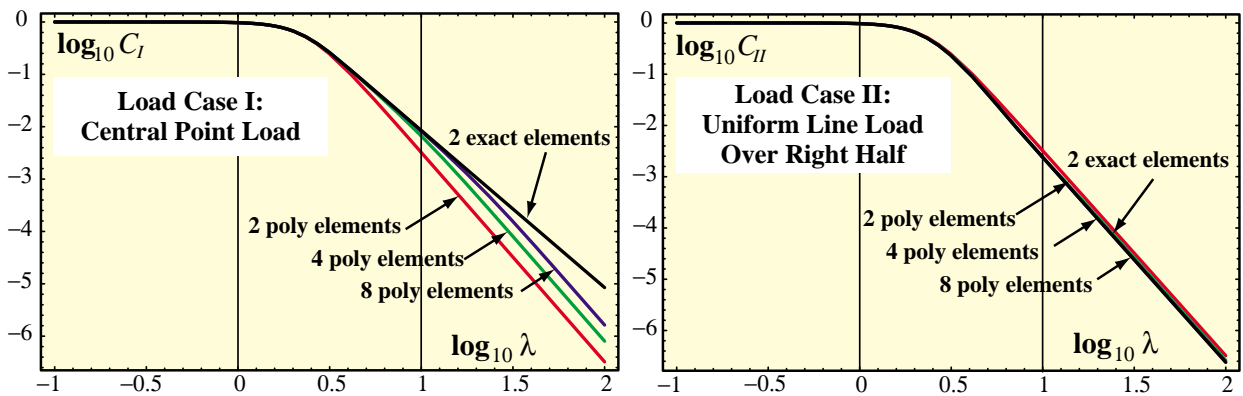
All quantities are kept symbolic. The focus of interest is the deflection v_C at midspan C ($x = L$). For convenience this is rendered dimensionless by taking $v_C^I(\lambda) = C_I(\lambda)v_{C0}^I$ and $v_C^{II}(\lambda) = C_{II}(\lambda)v_{C0}^{II}$ for load cases (I) and (II), respectively. Here $v_{C0}^I = -PL^3/(24EI)$ and $v_{C0}^{II} = -q_0L^4/(48EI)$ are the midspan deflections of cases (I) and (II) for $\lambda = 0$, that is, $k_F = 0$ (no foundation). The exact deflection factors for this model are

$$C_I(\lambda) = \frac{6\sqrt{2} \cos \sqrt{2}\lambda + \cosh \sqrt{2}\lambda - 2}{\lambda^3 \sin \sqrt{2}\lambda + \sinh \sqrt{2}\lambda} = 1 - \frac{13}{420}\lambda^4 + \frac{137}{138600}\lambda^8 \dots$$

$$C_{II}(\lambda) = \frac{48 (\cos \lambda/\sqrt{2} - \cosh \lambda/\sqrt{2})(\sin \lambda/\sqrt{2} - \sinh \lambda/\sqrt{2})}{\lambda^4 \sin \sqrt{2}\lambda + \sinh \sqrt{2}\lambda} = 1 - \frac{163}{5040}\lambda^4 + \frac{20641}{19958400}\lambda^8 + \dots$$
(13.36)

Both load cases were symbolically solved with two exact elements of length L produced by the module of Figure 13.10. As can be expected, the answers reproduce the exact solutions (13.36). Using any number of those elements would match (13.36) as long as the midspan section C is at a node. Then both cases were solved with 2, 4, and 8 elements with the stiffness (13.28) produced by cubic polynomials. The results are shown in a log-log plot in Figure 13.12. Results for selected values of λ are presented in Table 13.1.

As can be seen, for a “soft” foundation characterized by $\lambda < 1$, the cubic-polynomial elements gave satisfactory results and converged quickly to the exact answers, especially in load case (II). As λ grows over one, the deflections become rapidly smaller, and the polynomial FEM results exhibit higher relative errors. On the other hand, the absolute errors remain small. The conclusion is that exact elements are only worthwhile in highly rigid foundations (say $\lambda > 5$) and then only if results with small relative error are of interest.

FIGURE 13.12. Log-log plots of $C_I(\lambda)$ and $C_{II}(\lambda)$ for Example of Figure 13.11 over range $\lambda \in [0.1, 100]$.

⁵ Note that λ is a true physical parameter, whereas χ is discretization dependant because it involves the element length.

Remark 13.2. To correlate the exact stiffness and consistent forces with those obtained with polynomial shape functions it is illuminating to expand (13.33) as power series in χ . The rationale is that as the element size ℓ gets smaller, $\chi = \ell^4 \sqrt{k_F/(4EI)}$ goes to zero for fixed EI and k_F . *Mathematica* gives the expansions

$$\mathbf{K}_B^e = \mathbf{K}_{B0} + \chi^8 \mathbf{K}_{B8} + \chi^{12} \mathbf{K}_{B12} + \dots, \quad \mathbf{K}_F^e = \mathbf{K}_{F0} + \chi^4 \mathbf{K}_{F4} + \chi^8 \mathbf{K}_{F8} + \dots, \quad \mathbf{f}^e = \mathbf{f}_0 + \chi^4 \mathbf{f}_4 + \dots, \quad (13.37)$$

in which

$$\begin{aligned} \mathbf{K}_{B0}^e &= \text{eqn (12.20)}, \quad \mathbf{K}_{B8}^e = \frac{EI}{4365900 \ell^3} \begin{bmatrix} 25488 & 5352\ell & 23022 & -5043\ell \\ 5352\ell & 1136\ell^2 & 5043\ell & -1097\ell^2 \\ 23022 & 5043\ell & 25488 & -5352\ell \\ -5043\ell & -1097\ell^2 & -5352\ell & 1136\ell^2 \end{bmatrix}, \\ \mathbf{K}_{B12}^e &= -\frac{EI}{5959453500 \ell^3} \begin{bmatrix} 528960 & 113504\ell & 522090 & -112631\ell \\ 113504\ell & 24384\ell^2 & 112631\ell & -24273\ell^2 \\ 522090 & 112631\ell & 528960 & -113504\ell \\ -112631\ell & -24273\ell^2 & -113504\ell & 24384\ell^2 \end{bmatrix}, \quad \mathbf{K}_{F0}^e = \text{eqn (13.28)}, \\ \mathbf{K}_{F4}^e &= -\frac{k_F \ell^4}{2EI} \mathbf{K}_{B8}^e, \quad \mathbf{K}_{F8}^e = -\frac{3k_F \ell^4}{8EI} \mathbf{K}_{B12}^e, \quad \mathbf{f}_0^e = \frac{q_0 \ell}{12} [6 \ \ell \ 6 \ -\ell]^T, \quad \mathbf{f}_4^e = -\frac{q_0 \ell}{5040} [14 \ 3\ell \ 14 \ -3\ell]^T. \end{aligned} \quad (13.38)$$

Thus as $\chi \rightarrow 0$ we recover the stiffness matrices and force vector derived with polynomial shape functions, as can be expected. Note that \mathbf{K}_{B0} and \mathbf{K}_{F0} decouple, which allows them to be coded as separate modules. On the other hand the exact stiffnesses are coupled if $\chi > 0$. The foregoing expansions indicate that exactness makes little difference if $\chi < 1$.

§13.4. Equilibrium Theorems

One way to get high performance mechanical elements is to use equilibrium conditions whenever possible. These lead to *flexibility methods*. Taking advantage of equilibrium is fairly easy in one space dimension. It is more difficult in two and three, because it requires advanced variational methods that are beyond the scope of this book. This section surveys theorems that provide the theoretical basis for flexibility methods. These are applied to 1D element construction in §13.5.

§13.4.1. Self-Equilibrating Force System

First we establish a useful theorem that links displacement and force transformations. Consider a FEM discretized body such as that pictured in Figure 13.13(a). The generic potato intends to symbolize any discretized material body: an element, an element assembly or a complete structure. Partition its degrees of freedom into two types: r and s . The s freedoms (s stands for suppressed or supported) are associated with a *minimal set of supports* that control rigid body motions or RBMs. The r freedoms (r is for released) collect the rest. In the figure those freedoms are shown collected at individual points P_s and P_r for visualization convenience. Node forces, displacements and virtual displacements associated with those freedoms are partitioned accordingly. Thus

$$\mathbf{f} = \begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_r \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{u}_s \\ \mathbf{u}_r \end{bmatrix}, \quad \delta \mathbf{u} = \begin{bmatrix} \delta \mathbf{u}_s \\ \delta \mathbf{u}_r \end{bmatrix}. \quad (13.39)$$

The dimension n_s of \mathbf{f}_s , \mathbf{u}_s and $\delta \mathbf{u}_s$ is 1, 3 and 6 in one-, two- and three-dimensional space, respectively. Figure 13.13(b) shows the force system $\{\mathbf{f}_s, \mathbf{f}_r\}$ undergoing virtual displacements, which are exaggerated for visibility.⁶ Consider now the “rigid + deformational” displacement decomposition $\mathbf{u} = \mathbf{G}\mathbf{u}_s + \mathbf{d}$, in which matrix \mathbf{G} (of appropriate order) represents a rigid motion and \mathbf{d} are deformational displacements. Evaluating this at the r freedoms gives

$$\mathbf{u}_r = \mathbf{G}_r \mathbf{u}_s + \mathbf{d}_r, \quad \delta \mathbf{u}_r = \mathbf{G}_r \delta \mathbf{u}_s + \delta \mathbf{d}_r, \quad (13.40)$$

The first decomposition in (13.40), being linear in the actual displacements, is only valid only in geometrically linear analysis. That for virtual displacements is valid for a much broader class of problems.

If the supported freedom motion vanishes: $\mathbf{u}_s = \mathbf{0}$, then $\mathbf{u}_r = \mathbf{d}_r$. Thus \mathbf{d}_r represents a *relative displacement* of the unsupported freedoms with respect to the rigid motion $\mathbf{G}\mathbf{u}_s$, and likewise for the virtual displacements. Because a relative motion is necessarily associated with deformations, the alternative name *deformational displacements* is justified.

The external virtual work is $\delta W = \delta W_s + \delta W_r = \delta \mathbf{u}_s^T \mathbf{f}_s + \delta \mathbf{u}_r^T \mathbf{f}_r$. If the force system in Figure 13.13(a) is in self equilibrium *and* the virtual displacements are imparted by rigid motions $\delta \mathbf{d}_r = \mathbf{0}$ and $\delta \mathbf{u}_r = \mathbf{G}_r \delta \mathbf{u}_s$, the virtual work must vanish: $\delta W = \delta \mathbf{u}_s^T \mathbf{f}_s + \delta \mathbf{u}_s^T \mathbf{G}_r^T \mathbf{f}_r = \delta \mathbf{u}_s^T (\mathbf{f}_s + \mathbf{G}_r^T \mathbf{f}_r) = 0$. Because the $\delta \mathbf{u}_s$ are arbitrary, it follows that

$$\mathbf{f}_s + \mathbf{G}_r^T \mathbf{f}_r = \mathbf{0}, \quad \mathbf{f}_s = -\mathbf{G}_r^T \mathbf{f}_r. \quad (13.41)$$

These are the *overall static equilibrium equations* of a discrete mechanical system in self equilibrium. Sometimes it is useful to express the foregoing expressions in the complete-vector form

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_s \\ \mathbf{u}_r \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{G}_r \end{bmatrix} \mathbf{u}_s + \begin{bmatrix} \mathbf{0} \\ \mathbf{d}_r \end{bmatrix}, \quad \delta \mathbf{u} = \begin{bmatrix} \delta \mathbf{u}_s \\ \delta \mathbf{u}_r \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ \mathbf{G}_r \end{bmatrix} \delta \mathbf{u}_s + \begin{bmatrix} \mathbf{0} \\ \delta \mathbf{d}_r \end{bmatrix}, \quad \mathbf{f} = \begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_r \end{bmatrix} = \begin{bmatrix} -\mathbf{G}_r^T \\ \mathbf{I} \end{bmatrix} \mathbf{f}_r. \quad (13.42)$$

Relations in (13.42) are said to be *reciprocal*.⁷

⁶ Under virtual displacements the forces are frozen for application of the Principle of Virtual Work.

⁷ If the model is geometrically nonlinear, the first form in (13.42) does not hold.

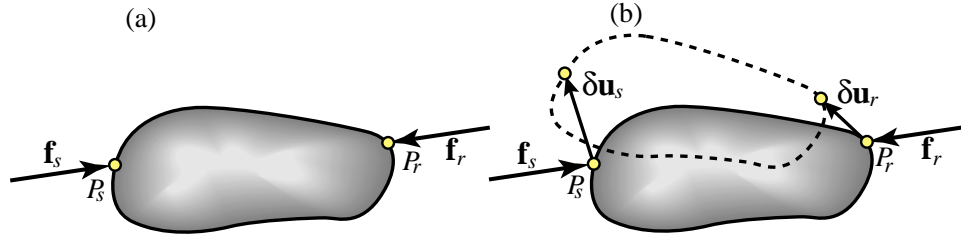


FIGURE 13.13. Body to illustrate equilibrium theorems. Nodal freedoms classified into supported (s) and released (r), each lumped to a point to simplify diagram. (a) Self equilibrated node force system. (b) Force system of (a) undergoing virtual displacements; grossly exaggerated for visibility.

Remark 13.3. The freedoms in \mathbf{u}_s are *virtual supports*, chosen for convenience in flexibility derivations. They should not be confused with actual or physical supports. For instance Civil Engineering structures tend to have redundant physical supports, whereas aircraft or orbiting satellites have none.

§13.4.2. Handling Applied Forces

Consider now a generalization of the previous scenario. An externally applied load system of surface or body forces, not necessarily in self equilibrium, acts on the body. For example, the surface tractions pictured in Figure 13.14(a). To bring this under the framework of equilibrium analysis, a series of steps are required.

First, the force system is replaced by a single resultant \mathbf{q} , as pictured in Figure 13.14(b).⁸ The point of application is P_q . Equilibrium is restored by introducing node forces \mathbf{q}_r and \mathbf{q}_s at the appropriate freedoms. The overall equilibrium condition is obtained by putting the system $\{\mathbf{q}, \mathbf{q}_r, \mathbf{q}_s\}$ through rigid-motion virtual displacements, as pictured in Figure 13.14(c). Point P_q moves through $\delta \mathbf{u}_q$, and \mathbf{G} evaluated at P_q is \mathbf{G}_q . The virtual work is $\delta W = \delta \mathbf{u}_s^T (\mathbf{q}_s + \mathbf{G}_r^T \mathbf{q}_r + \mathbf{G}_q^T \mathbf{q}) = 0$ whence

$$\mathbf{q}_s + \mathbf{G}_r^T \mathbf{q}_r + \mathbf{G}_q^T \mathbf{q} = \mathbf{0}. \quad (13.43)$$

If (13.43) is sufficient to determine \mathbf{q}_s and \mathbf{q}_r , the load system of Figure 13.14(a) can be effectively replaced by the nodal forces $-\mathbf{q}_s$ and $-\mathbf{q}_r$, as depicted in Figure 13.14(d). These are called the *equivalent node forces*. But in general (13.43) is insufficient to fully determine \mathbf{q}_s and \mathbf{q}_r . The remaining equations to construct the equivalent forces must come from a theorem that accounts for the internal energy, as discussed in §13.4.3.

Adding (13.41) and (13.43) gives the *general overall equilibrium condition*

$$\mathbf{f}_s + \mathbf{q}_s + \mathbf{G}_r^T (\mathbf{f}_r + \mathbf{q}_r) + \mathbf{G}_q^T \mathbf{q} = \mathbf{0}, \quad (13.44)$$

which is applied in §13.4.4 to the recovery of supported freedoms.

Remark 13.4. The replacement of the applied force by a resultant is not strictly necessary, as it is always possible to write out the virtual work by appropriately integrating distributed effects. The resultant is primarily useful as an instructional tool, because matrix \mathbf{G}_q is not position dependent.

Remark 13.5. Conditions (13.41) and (13.43), which were derived through the PVW, hold for general mechanical systems under mild reversibility requirements [670, §231], including geometric nonlinearities. From now on we restrict attention to systems *linear in the actual displacements*.

§13.4.3. Flexibility Equations

The first step in FEM equilibrium analysis is obtaining discrete *flexibility equations*. The stiffness equations introduced in Chapter 2 relate forces to displacements. At the element level they are $\mathbf{f}^e = \mathbf{K}^e \mathbf{u}^e$. By definition, flexibility equations relate displacements to forces: $\mathbf{u}^e = \mathbf{F}^e \mathbf{f}^e$, where \mathbf{F}^e is the element flexibility matrix. So the expectation is that the flexibility can be obtained as the inverse of the stiffness: $\mathbf{F}^e = (\mathbf{K}^e)^{-1}$. Right?

⁸ Although the figure shows a resultant point force, in general it may include a point moment that is not shown for simplicity. See also Remark 13.3.

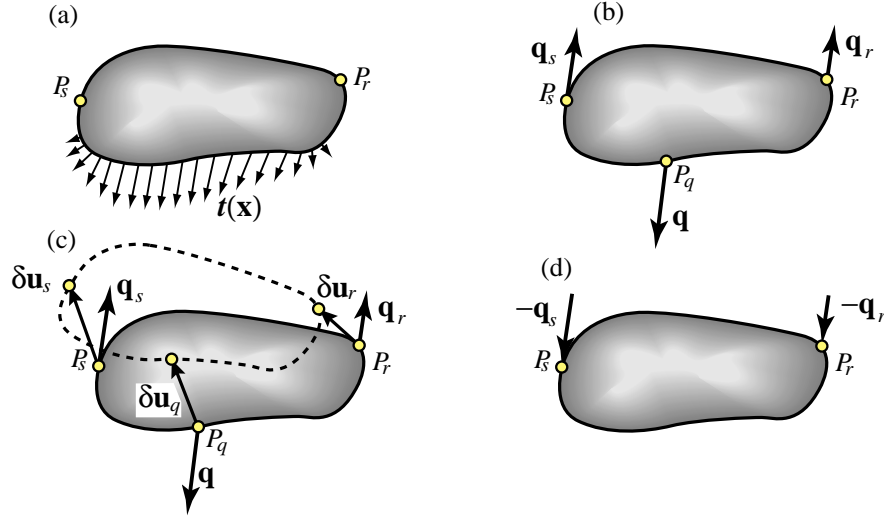


FIGURE 13.14. Processing non-self-equilibrated applied loads with flexibility methods. (a) Body under applied distributed load. (b) Substitution by resultant and self equilibration. (c) Deriving overall equilibrium conditions through the PVW. (d) Replacing the applied loads by equivalent nodal forces.

Wrong. Recall that \mathbf{K}^e for a disconnected free-free element is *singular*. Its ordinary inverse does not exist. Expectations go up in smoke. The same difficulty holds for a superelement or complete structure.

To get a conventional flexibility matrix⁹ it is necessary to *remove all rigid body motions* in advance. This can be done through the *virtual supports* introduced in §13.4.1. The support motions \mathbf{u}_s are fixed, say $\mathbf{u}_s = \mathbf{0}$. Flexibility equations are sought between what is left of the kinematics. Dropping the element superscript for brevity, for a linear problem one gets

$$\mathbf{F}_{rr} \mathbf{f}_r = \mathbf{d}_r. \quad (13.45)$$

Note that \mathbf{u}_r does not appear: only the deformational or relative displacements. To recover \mathbf{u}_r it is necessary to release the supports, but if that is naively done \mathbf{F}_{rr} ceases to exist. This difficulty is overcome in §13.4.4.

There is another key difference with stiffness methods. The DSM assembly procedure covered in Chapter 3 (and extended in Chapter 27 to general structures) *does not translate into a similar technique for flexibility methods*. Although it is possible to assemble flexibilities of MoM elements, the technique is neither simple nor elegant. And it becomes dauntingly complex when tried on continuum-based elements [211].

So one of the main uses of flexibility equations today is as a stepping stone on the way to derive element stiffness equations, starting from (13.45). The procedural steps are explained in §13.4.4. But how should (13.45) be derived? There are several methods but only one, based on the Total Complementary Potential Energy (TCPE) principle of variational mechanics is described here.

To apply TCPE, the complementary energy Π^* of the body must be expressed as a function of the nodal forces \mathbf{f}_r . For fixed supports ($\mathbf{u}_s = \mathbf{0}$) and a linear system, the functional can be expressed as

$$\Pi^*(\mathbf{f}_r) = U^*(\mathbf{f}_r) - \mathbf{f}_r^T \mathbf{d}_r = \frac{1}{2} \mathbf{f}_r^T \mathbf{F}_{rr} \mathbf{f}_r + \mathbf{f}_r^T \mathbf{b}_r - \mathbf{f}_r^T \mathbf{d}_r + \Pi_0^*. \quad (13.46)$$

Here U^* is the internal complementary energy, also called the stress energy by many authors, e.g., [292], \mathbf{b}_r is a term resulting from loading actions such as as thermal effects, body or surface forces, and Π_0^* is independent of \mathbf{f}_r . Calculation of U^* in 1D elements involves expressing the internal forces (axial force, shear forces, bending

⁹ In the FEM literature it is often called simply *the* flexibility. The reason is that for a long time it was believed that getting a flexibility matrix required a supported structure. With the recent advent of the free-free flexibility (see **Notes and Bibliography**) it becomes necessary to introduce a “deformational” or “conventional” qualifier.

moments, torque, etc.) in terms of \mathbf{f}_r from statics. Application examples are given in the next section.¹⁰ The TCPE principle states that Π^* is stationary with respect to variations in \mathbf{f}_r when kinematic compatibility is satisfied:

$$\frac{\partial \Pi^*}{\partial \mathbf{f}_r} = \mathbf{F}_{rr} \mathbf{f}_r + \mathbf{b}_r - \mathbf{d}_r = \mathbf{0}, \quad \text{whence} \quad \mathbf{d}_r = \mathbf{F}_{rr} \mathbf{f}_r + \mathbf{b}_r. \quad (13.47)$$

By hypothesis the deformational flexibility \mathbf{F}_{rr} is nonsingular. Solving for \mathbf{f}_r gives the *deformational stiffness* equations

$$\mathbf{f}_r = \mathbf{K}_{rr} \mathbf{d}_r - \mathbf{q}_r, \quad \text{with} \quad \mathbf{K}_{rr} = \mathbf{F}_{rr}^{-1} \quad \text{and} \quad \mathbf{q}_r = \mathbf{K}_{rr} \mathbf{b}_r. \quad (13.48)$$

The matrix \mathbf{K}_{rr} is the *deformational stiffness* matrix, whereas \mathbf{q}_r is the *equivalent load* vector.

§13.4.4. Rigid Motion Injection

Suppose that \mathbf{F}_{rr} and \mathbf{q}_r of (13.48) have been found, for example from the TPCE principle (13.47). The goal is to arrive at the free-free stiffness equations, which are partitioned in accordance with (13.39) as

$$\begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_r \end{bmatrix} = \begin{bmatrix} \mathbf{K}_{ss} & \mathbf{K}_{sr} \\ \mathbf{K}_{rs} & \mathbf{K}_{rr} \end{bmatrix} \begin{bmatrix} \mathbf{u}_s \\ \mathbf{u}_r \end{bmatrix} - \begin{bmatrix} \mathbf{q}_s \\ \mathbf{q}_r \end{bmatrix}, \quad (13.49)$$

To justify the presence of \mathbf{K}_{rr} and \mathbf{q}_r here, set $\mathbf{u}_s = \mathbf{0}$, whence $\mathbf{u}_r = \mathbf{d}_r$. Consequently the second equation reduces to $\mathbf{f}_r = \mathbf{K}_{rr} \mathbf{d}_r - \mathbf{q}_r$, which matches (13.48). Inserting \mathbf{f}_s and \mathbf{f}_r into (13.44) yields

$$(\mathbf{K}_{ss} + \mathbf{G}_r^T \mathbf{K}_{rs}) \mathbf{u}_s + (\mathbf{K}_{sr} + \mathbf{G}_r^T \mathbf{K}_{rr}) \mathbf{u}_r + [\mathbf{q}_s + \mathbf{G}_r^T \mathbf{q}_r + \mathbf{G}_q^T \mathbf{q}] = \mathbf{0}, \quad (13.50)$$

and replacing $\mathbf{u}_r = \mathbf{G}_r \mathbf{u}_s + \mathbf{d}_r$,

$$[\mathbf{K}_{ss} + \mathbf{G}_r^T \mathbf{K}_{rs} + \mathbf{K}_{sr} \mathbf{G}_r + \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r] \mathbf{u}_s + [\mathbf{K}_{sr} + \mathbf{G}_r^T \mathbf{K}_{rr}] \mathbf{d}_r + [\mathbf{q}_s + \mathbf{G}_r^T \mathbf{q}_r + \mathbf{G}_q^T \mathbf{q}] = \mathbf{0}. \quad (13.51)$$

Because \mathbf{u}_s , \mathbf{d}_r and \mathbf{q} can be arbitrarily varied, each bracket in (13.51) must vanish identically, giving

$$\begin{aligned} \mathbf{q}_s &= -\mathbf{G}_r^T \mathbf{q}_r - \mathbf{G}_q^T \mathbf{q}, & \mathbf{K}_{sr} &= -\mathbf{G}_r^T \mathbf{K}_{rr}, & \mathbf{K}_{rs} &= \mathbf{K}_{sr}^T = -\mathbf{K}_{rr} \mathbf{G}_r, \\ \mathbf{K}_{ss} &= -\mathbf{G}_r^T \mathbf{K}_{rs} - \mathbf{K}_{sr} \mathbf{G}_r - \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r = \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r + \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r - \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r = \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r. \end{aligned} \quad (13.52)$$

Inserting these into (13.49) yields

$$\begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_r \end{bmatrix} = \begin{bmatrix} \mathbf{G}_r^T \mathbf{K}_{rr} \mathbf{G}_r & -\mathbf{G}_r^T \mathbf{K}_{rr} \\ -\mathbf{K}_{rr} \mathbf{G}_r & \mathbf{K}_{rr} \end{bmatrix} \begin{bmatrix} \mathbf{u}_s \\ \mathbf{u}_r \end{bmatrix} - \begin{bmatrix} \mathbf{G}_r^T \mathbf{q}_r - \mathbf{G}_q^T \mathbf{q} \\ \mathbf{q}_r \end{bmatrix}. \quad (13.53)$$

This can be put in the more compact form

$$\begin{bmatrix} \mathbf{f}_s \\ \mathbf{f}_r \end{bmatrix} = \begin{bmatrix} -\mathbf{G}_r^T \\ \mathbf{I} \end{bmatrix} \mathbf{K}_{rr} \begin{bmatrix} -\mathbf{G}_r & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}_s \\ \mathbf{u}_r \end{bmatrix} - \begin{bmatrix} -\mathbf{G}_r^T \\ \mathbf{I} \end{bmatrix} \mathbf{q}_r + \begin{bmatrix} \mathbf{G}_q \\ \mathbf{0} \end{bmatrix} \mathbf{q} = \mathbf{T}^T \mathbf{K}_{rr} \mathbf{T} \mathbf{u} - \mathbf{T}^T \mathbf{q}_r + \mathbf{T}_q^T \mathbf{q}, \quad (13.54)$$

with $\mathbf{T} = \begin{bmatrix} -\mathbf{G}_r & \mathbf{I} \end{bmatrix}$ and $\mathbf{T}_q = \begin{bmatrix} \mathbf{G}_q & \mathbf{0} \end{bmatrix}$.

The end result is that the free-free stiffness is $\mathbf{T}^T \mathbf{F}_{rr}^{-1} \mathbf{T}$. Alternatively (13.54) may be derived by plugging $\mathbf{d}_r = \mathbf{u}_r - \mathbf{G}_r \mathbf{u}_s$ into (13.48) and then into (13.44).

Remark 13.6. Let \mathbf{S} be a $n_s \times n_s$ nonsingular matrix. The matrix \mathbf{R} built by the prescription

$$\mathbf{R} = \begin{bmatrix} \mathbf{S} \\ \mathbf{GS} \end{bmatrix} \quad (13.55)$$

is called a *rigid body motion matrix* or simply *RBM matrix*. The columns of \mathbf{R} represent nodal values of rigid motions, hence the name. The scaling provided by \mathbf{S} may be adjusted to make \mathbf{R} simpler. The key property is $\mathbf{TR} = \mathbf{0}$ and thus $\mathbf{KR} = \mathbf{T}^T \mathbf{K}_{rr} \mathbf{TR} = \mathbf{0}$. Other properties are studied in [?].

¹⁰ For 2D and 3D elements the process is more delicate and demands techniques, such as hybrid variational principles, that lie beyond the scope of this material.

§13.4.5. Applications

Stiffness Equilibrium Tests. If one injects $\mathbf{u}_r = \mathbf{G}\mathbf{u}_s$ and $\mathbf{q}_r = \mathbf{0}$ into (13.54) the result is $\mathbf{f}_r = \mathbf{0}$ and $\mathbf{f}_s = \mathbf{0}$. That is, all node forces must vanish for arbitrary \mathbf{u}_s . This test is useful at any level (element, superelement, full structure) to verify that a directly generated \mathbf{K} (that is, a \mathbf{K} constructed independently of overall equilibrium) is “clean” as regards rigid body modes.

Element Stiffness from Flexibility. Here \mathbf{F}_{rr} is constructed at the supported element level, inverted to get \mathbf{K}_{rr} and rigid motions injected through (13.54). Applications to element construction are illustrated in §13.5.

Experimental Stiffness from Flexibility. In this case \mathbf{F}_{rr} is obtained through experimental measurements on a supported structure or substructure.¹¹ To insert this as a “user defined superelement” in a DSM code, it is necessary to produce a stiffness matrix. This is done again by inversion and RBM injection.

§13.5. Flexibility Based Derivations

The equilibrium theorems of the foregoing section are applied to the flexibility derivation of several one-dimensional elements.

§13.5.1. Timoshenko Plane Beam-Column

A beam-column member combines axial and bending effects. A 2-node, straight beam-column has three DOFs at each node: the axial displacement, the transverse displacement and a rotation. If the cross section is doubly symmetric, axial and bending effects are decoupled. A prismatic, plane element of this kind is shown in Figure 13.15(a). End nodes are 1–2. The bending component is modeled as a Timoshenko beam. The element is subjected to the six node forces shown, and to a uniformly distributed load q_0 . To suppress rigid motions node 1 is fixed as shown in Figure 13.15(b), making the beam a cantilever. Following the notation of §13.4.1–§13.4.2,

$$\mathbf{u}_s = \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_1 \end{bmatrix}, \quad \mathbf{d}_r = \mathbf{u}_r = \begin{bmatrix} u_{x2} \\ u_{y2} \\ \theta_2 \end{bmatrix}, \quad \mathbf{f}_s = \begin{bmatrix} f_{x1} \\ f_{y1} \\ m_1 \end{bmatrix}, \quad \mathbf{f}_r = \begin{bmatrix} f_{x2} \\ f_{y2} \\ m_2 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ q\ell \\ 0 \end{bmatrix}, \quad \mathbf{G}(x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & x \\ 0 & 0 & 0 \end{bmatrix}, \quad (13.56)$$

Further, $\mathbf{G}_r = \mathbf{G}(\ell)$ and $\mathbf{G}_q = \mathbf{G}(\ell/2)$. The internal forces are the axial force $F(x)$, the transverse shear $V(x)$ and the bending moment $M(x)$. These are directly obtained from statics by doing a free-body diagram at distance x from the left end as illustrated in Figure 13.15(c). With the positive convention as shown we get

$$N(x) = -f_{x2}, \quad V(x) = -f_{y2} - q_0(\ell - x), \quad M(x) = m_2 + f_{y2}(\ell - x) + \frac{1}{2}q_0(\ell - x)^2. \quad (13.57)$$

Useful check: $dM/dx = V$. Assuming a doubly symmetric section so that N and M are decoupled, the element TCPE functional is

$$\begin{aligned} \Pi^* &= \frac{1}{2} \int_0^\ell \left(\frac{N^2}{EA} + \frac{M^2}{EI} + \frac{V^2}{GA_s} \right) dx - \mathbf{f}_r^T \mathbf{d}_r = \frac{1}{2} \mathbf{f}_r^T \mathbf{F}_{rr} \mathbf{f}_r + \mathbf{f}_r^T \mathbf{b}_r - \mathbf{f}_r^T \mathbf{d}_r + \Pi_0^*, \\ \text{in which } \mathbf{F}_{rr} &= \frac{\partial^2 \Pi^*}{\partial \mathbf{f}_r \partial \mathbf{f}_r} = \begin{bmatrix} \frac{\ell}{EA} & 0 & 0 \\ 0 & \frac{\ell^3(4 + \Phi)}{24EI} & \frac{\ell^2}{2EI} \\ 0 & \frac{\ell^2}{2EI} & \frac{\ell}{EI} \end{bmatrix}, \quad \mathbf{b}_r = q_0 \begin{bmatrix} 0 \\ \frac{\ell^3(4 + \Phi)}{12EI} \\ \frac{\ell^2}{6EI} \end{bmatrix}. \end{aligned} \quad (13.58)$$

¹¹ The classical static tests on an airplane wing are performed by applying transverse forces and torques to the wing tip with the airplane safely on the ground. These experimental influence coefficients can be used for model validation.

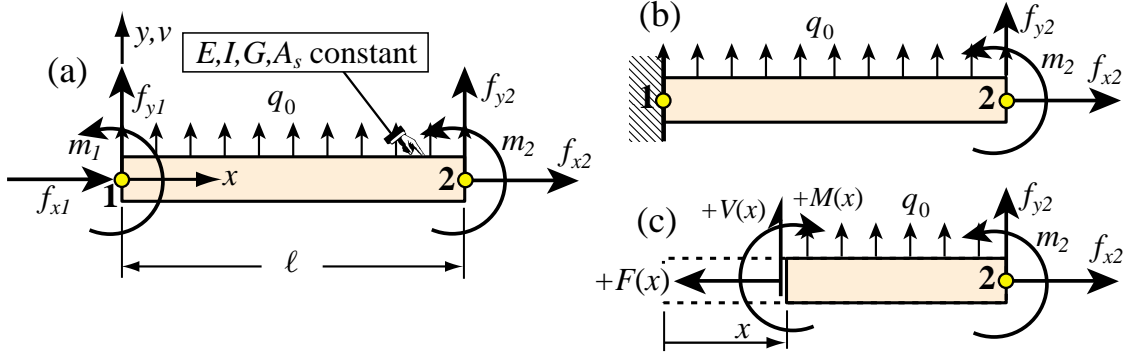


FIGURE 13.15. Flexibility derivation of Timoshenko plane beam-column stiffness: (a) element and node forces, (b) removal of RBMs by fixing left node, (c) FBD that gives internal forces at varying x .

Term Π_0^* is inconsequential, since it disappears on differentiation.

Applying the TCPE principle yields $\mathbf{F}_{rr} \mathbf{f}_r = \mathbf{b}_r - \mathbf{d}_r$. This is inverted to produce the deformational stiffness relation $\mathbf{f}_r = \mathbf{K}_{rr} \mathbf{d}_r + \mathbf{q}_r$, in which

$$\mathbf{K}_{rr} = \mathbf{F}_{rr}^{-1} = \begin{bmatrix} \frac{EA}{\ell} & 0 & 0 \\ 0 & \frac{12EI}{\ell^3(1+\Phi)} & -\frac{6EI}{\ell^2(1+\Phi)} \\ 0 & -\frac{6EI}{\ell^2(1+\Phi)} & \frac{EI(4+\Phi)}{\ell(1+\Phi)} \end{bmatrix}, \quad \mathbf{q}_r = q_0 \mathbf{K}_{rr} \mathbf{b}_r = \begin{bmatrix} 0 \\ q_0 \ell / 2 \\ -q_0 \ell^2 / 12 \end{bmatrix}. \quad (13.59)$$

To use (13.54) the following transformation matrices are required:

$$\mathbf{T}^T = \begin{bmatrix} -\mathbf{G}_r^T \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & -\ell & -1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_q = \begin{bmatrix} \mathbf{G}_q \\ \mathbf{0} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \ell/2 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} 0 \\ q_0 \ell \\ 0 \end{bmatrix}. \quad (13.60)$$

Injecting the rigid body modes from (13.54), $\mathbf{K}^e = \mathbf{T}^T \mathbf{K}_{rr} \mathbf{T}$ and $\mathbf{f}^e = \mathbf{T}^T \mathbf{q}_r$, yields

$$\mathbf{K}^e = \frac{EA}{\ell} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \frac{EI}{\ell^3(1+\Phi)} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 12 & 6\ell & 0 & -12 & 6\ell \\ 0 & 6\ell & \ell^2(4+\Phi) & 0 & -6\ell & \ell^2(2-\Phi) \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -12 & -6\ell & 0 & 12 & -6\ell \\ 0 & 6\ell & \ell^2(2-\Phi) & 0 & -6\ell & \ell^2(4+\Phi) \end{bmatrix},$$

$$\mathbf{f}^e = q_0 \ell [0 \quad 1/2 \quad \ell/12 \quad 0 \quad 1/2 \quad -\ell/12]^T. \quad (13.61)$$

The bending component is the same stiffness found previously in §13.2.6; compare with (13.18). The node force vector is the same as the consistent one constructed in an Exercise. A useful verification technique is to support the beam element at end 2 and recompute \mathbf{K}^e and \mathbf{f}^e . This should reproduce (13.61).

All of the foregoing computations were carried out by the *Mathematica* script shown in Figure 13.16.

§13.5.2. Plane Circular Arch in Local System

In this and next subsection, the flexibility method is used to construct the stiffness matrix of a curved, prismatic, plane beam-column element with circular profile, pictured in Figure 13.17(a). The local system $\{x, y\}$ is defined

```

ClearAll[Le,EI,GAs,Φ,q0,fx2,fy2,m2]; GAs=12*EI/(Φ*Le^2);
F=fx2; V=-fy2-q0*(Le-x); M=m2+fy2*(Le-x)+(1/2)*q0*(Le-x)^2;
Print["check dM/dx=V: ",Simplify[D[M,x]-V]];
Ucd=F^2/(2*EA)+M^2/(2*EI)+V^2/(2*GAs);
Uc=Simplify[Integrate[Ucd,{x,0,Le}]]; Print["Uc=",Uc];
u2=D[Uc,fx2]; v2=D[Uc,fy2]; θ2=D[Uc,m2];
Frr={{ D[u2,fx2], D[u2,fy2], D[u2,m2]},
      { D[v2,fx2], D[v2,fy2], D[v2,m2]},
      { D[θ2,fx2], D[θ2,fy2], D[θ2,m2]}};
br={D[Uc,fx2],D[Uc,fy2],D[Uc,m2]}/.{fx2->0,fy2->0,m2->0}; Print["br=",br];
Frr=Simplify[Frr]; Print["Frr=",Frr//MatrixForm];
Krr=Simplify[Inverse[Frr]]; Print["Krr=",Krr//MatrixForm];
qr=Simplify[-Krr.br]; Print["qr=",qr];
TT={{-1,0,0},{0,-1,0},{0,-Le,-1},{1,0,0},{0,1,0},{0,0,1}};
T=Transpose[TT]; Simplify[Ke=TT.Krr.T]; Print["Ke=",Ke//MatrixForm];
GrT={{1,0,0},{0,1,0},{0,Le,1}}; Gr=Transpose[GrT];
GqT={{1,0,0},{0,1,0},{0,Le/2,1}}; Gq=Transpose[GqT];
Print["Gr=",Gr//MatrixForm," Gq=",Gq//MatrixForm];
qv={0,q0*Le,0}; Print["qs=",Simplify[-GrT.qr-GqT.qv]];

```

FIGURE 13.16. Script to derive the stiffness matrix and consistent load vector of the prismatic, plane Timoshenko beam element of Figure 13.15 by flexibility methods.

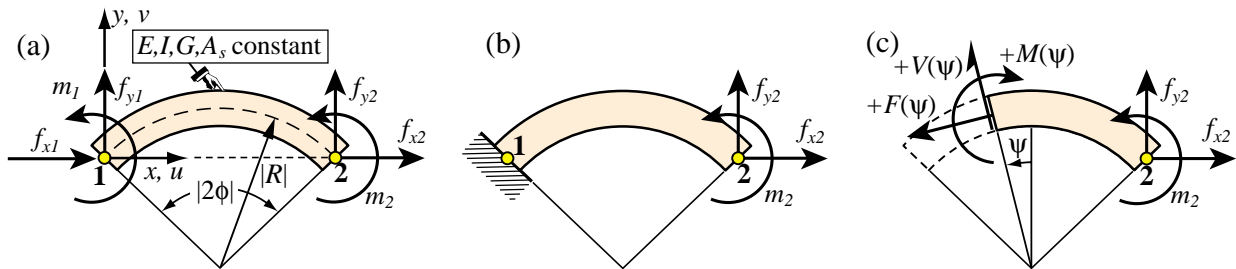


FIGURE 13.17. Flexibility derivation of plane circular arch element: (a) element and node forces, (b) removal of RBMs by fixing left node, (c) free body diagram of varying cross section.

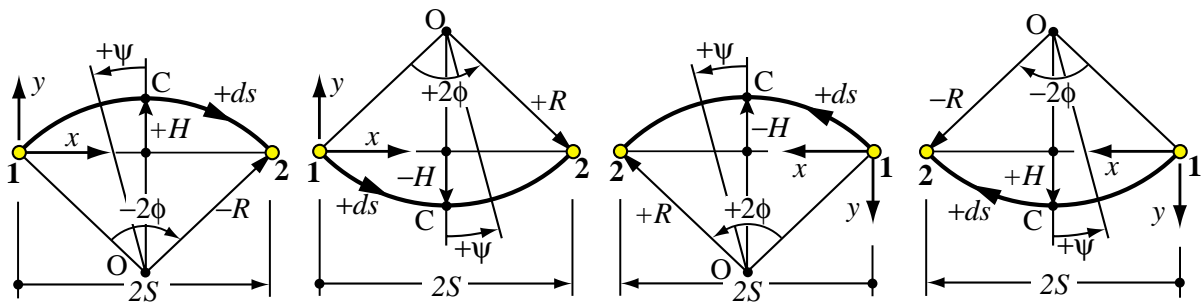


FIGURE 13.18. Sign conventions in derivation of circular arch element.

as shown there: x is a “chord axis” that passes through end nodes 1–2, and goes from 1 to 2. Axis y is placed at $+90^\circ$ from x . No load acts between nodes.

In a curved plane element of this nature, axial extension and bending are intrinsically coupled. Thus consideration of three freedoms per node is mandatory. These are the translations along x and y , and the rotation θ about z . This element can be applied to the analysis of plane arches and ring stiffeners (as in airplane fuselages

and submarine pressure hulls). If the arch curvature varies along the member, it should be subdivided into sufficiently small elements over each of which the radius magnitude is sensibly constant.

Care must be taken as regards sign conventions to ensure correct results when the arch convexity and node numbering changes. Various cases are pictured in Figure 13.18. The conventions are as follows:

- (1) The local node numbers define a *positive arclength traversal* along the element midline as going from 1 to 2. The curved length ℓ (not shown in figure) and the (chord) spanlength $2S$ are *always positive*.
- (2) The arch rise H , the angular span 2ϕ and the arch radius R are *signed quantities* as illustrated in Figure 13.18. The rise is the distance from chord midpoint to arch crown C: it has the sign of its projection on y . The angular span 2ϕ is that subtended by the arch on moving from 1 to 2: it is positive if CCW. Finally, the radius R has the sign of ϕ so $\ell = 2R\phi$ is always positive.

Some signed geometric relations:

$$S = R \sin \phi, \quad H = R(\cos \phi - 1) \quad (13.62)$$

The location of an arch section is defined by the “tilt angle” ψ measured from the circle center-to-crown symmetry line OC, positive CCW. See Figure 13.18. The differential arclength $ds = R d\psi$ always points in the positive traversal sense $1 \rightarrow 2$.

The rigid motions are removed by fixing the left end as shown in in Figure 13.17(b). The internal forces $F(\psi)$, $V(\psi)$ and $M(\psi)$ at an arbitrary cross section are obtained from the FBD of Figure 13.17(c) to be

$$F = f_{x2} \cos \psi + f_{y2} \sin \psi, \quad V = f_{x2} \sin \psi - f_{y2} \cos \psi, \quad M = m_2 + f_{x2} R(\cos \phi - \cos \psi) + f_{y2} R(\sin \psi - \sin \phi). \quad (13.63)$$

For typical straight beam-column members there are only two practically useful models: Bernoulli-Euler (BE) and Timoshenko. For *curved members* there are many more. These range from simple corrections to BE through theory-of-elasticity-based models. The model selected here is one of intermediate complexity. It is defined by the internal complementary energy functional

$$U^* = \int_0^\ell \left(\frac{(F - M/R)^2}{2EA} + \frac{M^2}{2EI} \right) ds = \int_{-\phi}^\phi \left(\frac{(F - M/R)^2}{2EA} + \frac{M^2}{2EI} \right) R d\psi. \quad (13.64)$$

The assumptions embodied in this formula are: (1) the shear energy density $V^2/(2GA_s)$ is neglected; (2) the cross section area A and moment of inertia I are unchanged with respect of those of the straight member. These assumptions are reasonable if $|R| > 10r$, where $r = +\sqrt{I/A}$ is the radius of gyration of the cross section. Further corrections are treated in Exercises. To simplify the ensuing formulas it is convenient to take

$$EA = \frac{EI}{\Psi^2 R^2 \phi^2} = \frac{4EI}{\Psi^2 \ell^2}, \quad \text{or} \quad \Psi = \frac{2r}{\ell} \quad \text{with} \quad r^2 = \frac{I}{A}. \quad (13.65)$$

This defines Ψ as a dimensionless geometric parameter. Note that this is not an intrinsic measure of arch slenderness, because it involves the element length. (In that respect it is similar to Φ of the Timoshenko beam element, defined by (13.17).) The necessary calculations are carried out by the *Mathematica* script of Figure 13.21, which has been pared down to essentials to save space. The deformational flexibility and stiffness computed are

$$\mathbf{F}_{rr} = \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ \text{symm} & F_{22} & F_{23} \\ & & F_{33} \end{bmatrix}, \quad \mathbf{K}_{rr} = \mathbf{F}_{rr}^{-1} = \begin{bmatrix} K_{11} & K_{12} & K_{13} \\ \text{symm} & K_{22} & K_{23} \\ & & K_{33} \end{bmatrix},$$

in which (first expression is exact value, second a Taylor series expansion in ϕ)

$$\begin{aligned}
 F_{11} &= \frac{\ell^3}{16EI\phi^3} (2\phi(2 + \phi^2\Psi^2) + 2\phi(1 + \phi^2\Psi^2) \cos 2\phi - 3 \sin 2\phi) = \frac{\ell^3}{60EI} (15\Psi^2 + \phi^2(2 - 15\Psi^2)) + O(\phi^4) \\
 F_{12} &= \frac{\ell^3 \sin \phi}{4EI\phi^3} (\sin \phi - \phi(1 + \phi^2\Psi^2) \cos \phi) = \frac{\ell^3 \phi}{12EI} (1 - 3\Psi^2) + O(\phi^3), \\
 F_{13} &= \frac{\ell^2}{2EI\phi^2} (\sin \phi - \phi(1 + \phi^2\Psi^2) \cos \phi) = \frac{\ell^3 \phi}{6EI} (1 - 3\Psi^2) + O(\phi^3), \\
 F_{22} &= \frac{\ell^3}{16EI\phi^3} (2\phi(2 + \phi^2\Psi^2) - 2\phi(1 + \phi^2\Psi^2) \cos 2\phi - \sin 2\phi) = \frac{\ell^3}{60EI} (20 + 3\phi^2(5\Psi^2 - 2)) + O(\phi^4), \\
 F_{23} &= \frac{\ell^2 \sin \phi}{2EI\phi} (1 + \phi^2\Psi^2) = \frac{\ell^2}{12EI} (6 + \phi^2(6\Psi^2 - 1)) + O(\phi^4), \quad F_{33} = \frac{\ell}{EI} (1 + \phi^2\Psi^2).
 \end{aligned} \tag{13.66}$$

Introduce $d_1 = \phi(1 + \phi^2\Psi^2)(\phi + \sin \phi \cos \phi) - 2 \sin^2 \phi$ and $d_2 = \phi - \sin \phi \cos \phi$. Then

$$\begin{aligned}
 K_{11} &= \frac{4EI\phi^4}{\ell^3 d_1} (1 + \phi^2\Psi^2) = \frac{EI}{45\ell^3} (45\Psi^2 + \phi^2(15\Psi^2 + 45\Psi^4 - 1)) + O(\phi^4), \quad K_{12} = 0, \\
 K_{13} &= \frac{4EI\phi^2}{\ell^2 d_1} (\phi(1 + \phi^2\Psi^2) \cos \phi - \sin \phi) = \frac{6EI\phi}{\ell^2 \Psi^2} (3\Psi^2 - 1) + O(\phi^3), \\
 K_{22} &= \frac{8EI\phi^3}{\ell^3 d_2} = \frac{12EI}{5\ell^3} (5 + \phi^2) + O(\phi^4), \quad K_{23} = -\frac{\ell \sin \phi}{2\phi} K_{22} = -\frac{EI}{5\ell^3} (30 + \phi^2) + O(\phi^4), \\
 K_{33} &= \frac{EI\phi}{8\ell d_1 d_2} (8\phi^2(3 + 2\phi^2\Psi^2) - 9 + 16 \cos 2\phi - 7 \cos 4\phi - 8\phi \sin 2\phi (2 + (1 + \phi^2\Psi^2) \cos 2\phi)) \\
 &= \frac{EI}{45\ell \Psi^2} (180\Psi^2 + \phi^2(5 - 48\Psi^2)) + O(\phi^4),
 \end{aligned} \tag{13.67}$$

The constraint $K_{23} = -K_{22}\ell \sin \phi / (2\phi)$ must be verified by any arch stiffness, regardless of the TCPE form used. The necessary transformation matrix to inject the rigid body modes

$$\mathbf{T} = [-\mathbf{G} \quad \mathbf{I}] = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & -2R \sin \phi & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -1 & -\ell \sin \phi / \phi & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}. \tag{13.68}$$

Applying the congruent transformation gives the free-free stiffness

$$\mathbf{K}^e = \mathbf{T}^T \mathbf{K}_{rr} \mathbf{T} = \begin{bmatrix} K_{11} & 0 & K_{13} & -K_{11} & 0 & -K_{13} \\ 0 & K_{22} & -K_{23} & 0 & -K_{22} & -K_{23} \\ K_{13} & -K_{23} & K_{33} & -K_{13} & K_{23} & K_{36} \\ -K_{11} & 0 & -K_{13} & K_{11} & 0 & K_{13} \\ 0 & -K_{22} & K_{23} & 0 & K_{22} & K_{23} \\ -K_{13} & -K_{23} & K_{36} & K_{13} & K_{23} & K_{33} \end{bmatrix} \tag{13.69}$$

The only new entry not in (13.67) is

$$K_{36} = -K_{33} - K_{23}\ell \sin \phi / \phi = \frac{EI}{45\ell} (90\Psi^2 + \phi^2(12\Psi^2 - 5)) + O(\phi^4)$$

As a check, if $\phi \rightarrow 0$ the entries reduce to that of the beam-column modeled with Bernoulli-Euler. For example $K_{11} \rightarrow 4EI\Psi^2/\ell^3 = EA/\ell$, $K_{22} \rightarrow 12EI/\ell^3$, $K_{33} \rightarrow 4EI/\ell$, $K_{36} \rightarrow 2EI/\ell$, etc.

```

ClearAll[ $\phi, \psi, \Psi, \Gamma, R, F, M, V, Le, EA, EI$ ];
EA=4*EI/( $\Psi^2 Le^2$ );
V=-fy2*Cos[ $\psi$ ]+fx2*Sin[ $\psi$ ]; F=fx2*Cos[ $\psi$ ]+fy2*Sin[ $\psi$ ];
M=m2+fx2*R*(Cos[ $\psi$ ]-Cos[ $\phi$ ])+fy2*R*(Sin[ $\psi$ ]+Sin[ $\phi$ ]);
Ucd=(F-M/R)^2/(2*EA)+ M^2/(2*EI);
Uc=Simplify[Integrate[Ucd*R,{ $\psi, -\phi, \phi$ }]]; Print["Uc=",Uc];
u2=D[Uc,fx2]; v2=D[Uc,fy2];  $\theta 2$ =D[Uc,m2];
Frr=Simplify[{ { D[u2,fx2], D[u2,fy2], D[u2,m2] },
{ D[v2,fx2], D[v2,fy2], D[v2,m2] },
{ D[ $\theta 2$ ,fx2], D[ $\theta 2$ ,fy2], D[ $\theta 2$ ,m2] } }];
Frr=FullSimplify[Frr/.{R->Le/(2* $\phi$ )}]; Print["Frr=",Frr//MatrixForm];
Krr=FullSimplify[Inverse[Frr]]; Print["Krr=",Krr//MatrixForm];
TT={{-1,0,0},{0,-1,0},{0,-2*R*Sin[ $\phi$ ],-1},{1,0,0},{0,1,0},{0,0,1}};
TT=TT/.{R->Le/(2* $\phi$ )}; T=Transpose[TT];
Ke=Simplify[TT.Krr.T]; Print["Ke=",Ke//MatrixForm];

```

FIGURE 13.19. Script to produce circular arch element stiffness in local coordinates.

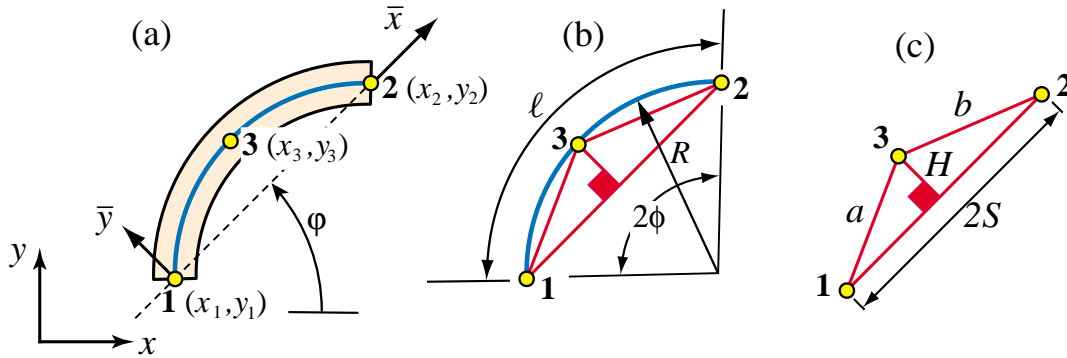
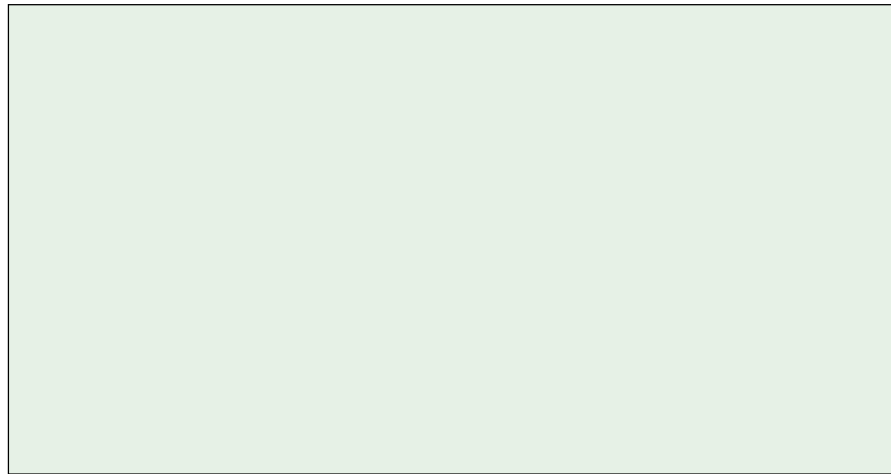


FIGURE 13.20. Plane circular arch element in global coordinates: (a) geometric id (b) intrinsic geometry recovery.

FIGURE 13.21. Module to produce plane circular arch element stiffness in global coordinates.
(To be done)

§13.5.3. Plane Circular Arch in Global System

To use the circular arch element in a 2D finite element program it is necessary to specify its geometry in the $\{x, y\}$ plane and then to transform the stiffness (13.69) to global coordinates. The first requirement can be handled by providing the coordinates of three nodes: $\{x_i, y_i\}, i = 1, 2, 3$. Node 3 (see Figure) is a geometric node that serves to define the element mean curvature but has no associated freedoms.

(Section to be completed).

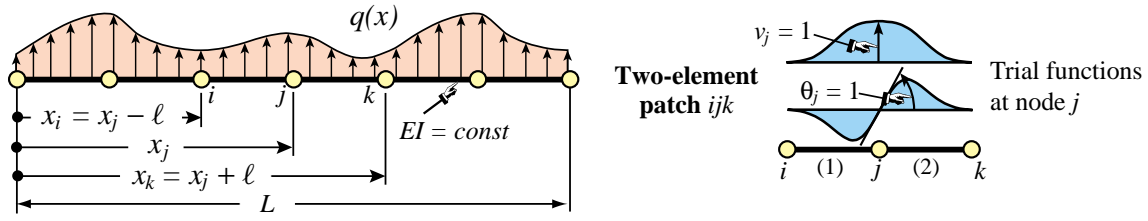


FIGURE 13.22. Repeating beam lattice for accuracy analysis.

§13.6. *Accuracy Analysis

This section presents the accuracy analysis of a repeating lattice of beam elements, analogous to that done for the bar element in §12.5. The analysis uses the method of modified differential equations (MoDE) mentioned in the **Notes and Bibliography** of Chapter 12. It is performed using a repeated differentiations scheme similar to that used in solving Exercise 12.8. Only the case of the Bernoulli-Euler model is worked out in detail.

§13.6.1. *Accuracy of Bernoulli-Euler Beam Element

Consider a lattice of repeating two-node, prismatic, plane Bernoulli-Euler beam elements of rigidity EI and length ℓ , as illustrated in Figure 13.22. The system is subject to an arbitrary lateral load $q(x)$, which is assumed infinitely differentiable in x . From the lattice extract a patch of two elements: (1) and (2), connecting nodes $i-j$ and $j-k$, respectively, as shown in Figure 13.22.

The FEM patch equations at node j are

$$\frac{EI}{\ell^3} \begin{bmatrix} -12 & -6\ell & 24 & 0 & -12 & 6\ell \\ 6\ell & 2\ell^2 & 0 & 8\ell^2 & -6\ell & 2\ell^2 \end{bmatrix} \begin{bmatrix} v_i \\ \theta_i \\ v_j \\ \theta_j \\ v_k \\ \theta_k \end{bmatrix} = \begin{bmatrix} f_j \\ m_j \end{bmatrix} \quad (13.70)$$

Expand $v(x)$, $\theta(x)$ and $q(x)$ in Taylor series about $x = x_j$, truncating at $n+1$ terms for v and θ , and $m+1$ terms for q . Using $\psi = (x - x_j)/\ell$, the series are $v(x) = v_j + \psi \ell v'_j + (\psi^2 \ell^2/2!)v''_j + \dots + (\psi^n \ell^n/n!)v_j^{[n]}$, $\theta(x) = \theta_j + \psi \ell \theta'_j + (\psi^2 \ell^2/2!)\theta''_j + \dots + (\psi^n \ell^n/n!)\theta_j^{[n]}$, and $q(x) = q_j + \psi \ell q'_j + (\psi^2 \ell^2/2!)q''_j + \dots + (\psi^m \ell^m/m!)q_j^{[m]}$. Here $v_j^{[n]}$, etc., is an abbreviation for $d^n v(x_j)/dx^n$.¹² Evaluate the $v(x)$ and $\theta(x)$ series at i and k by setting $\psi = \pm 1$, and insert in (13.70). Use the $q(x)$ series evaluated over elements (1) and (2), to compute the consistent forces f_j and m_j as

$$f_j = \frac{\ell}{2} \left(\int_{-1}^1 N_3^{(1)} q^{(1)} d\xi^{(1)} + \int_{-1}^1 N_1^{(2)} q^{(2)} d\xi^{(2)} \right), \quad m_j = \frac{\ell}{2} \left(\int_{-1}^1 N_4^{(1)} q^{(1)} d\xi^{(1)} + \int_{-1}^1 N_2^{(2)} q^{(2)} d\xi^{(2)} \right). \quad (13.71)$$

Here $N_3^{(1)} = \frac{1}{4}(1 + \xi^{(1)})^2(2 - \xi^{(1)})$, $N_4^{(1)} = -\frac{\ell}{8}(1 + \xi^{(1)})^2(1 - \xi^{(1)})$, $N_1^{(2)} = \frac{1}{4}(1 - \xi^{(2)})^2(2 + \xi^{(2)})$, and $N_2^{(2)} = \frac{\ell}{8}(1 - \xi^{(2)})^2(1 + \xi^{(2)})$ are the Hermitian shape functions components of the j node trial function, whereas $q^{(1)} = q(\psi^{(1)})$, $\psi^{(1)} = -\frac{1}{2}(1 - \xi^{(1)})$ and $q^{(2)} = q(\psi^{(2)})$, $\psi^{(2)} = \frac{1}{2}(1 + \xi^{(2)})$ denote the lateral loads.

¹² Brackets are used instead of parentheses to avoid confusion with element superscripts. If derivatives are indexed by primes or roman numerals the brackets are omitted.

To show the resulting system in compact matrix form it is convenient to collect the derivatives at node j into vectors:

$$\mathbf{v}_j = [v_j \ v'_j \ v''_j \ \dots v_j^{[n]}]^T, \quad \boldsymbol{\theta}_j = [\theta_j \ \theta'_j \ \theta''_j \ \dots \theta_j^{[n]}]^T, \quad \mathbf{q}_j = [q_j \ q'_j \ q''_j \ \dots q_j^{[n]}]^T. \quad (13.72)$$

The resulting differential system can be compactly written

$$\begin{bmatrix} \mathbf{S}_{vv} & \mathbf{S}_{v\theta} \\ \mathbf{S}_{\theta v} & \mathbf{S}_{\theta\theta} \end{bmatrix} \begin{bmatrix} \mathbf{v}_j \\ \boldsymbol{\theta}_j \end{bmatrix} = \begin{bmatrix} \mathbf{P}_v \\ \mathbf{P}_\theta \end{bmatrix} \mathbf{q}_j. \quad (13.73)$$

Here \mathbf{S}_{vv} , $\mathbf{S}_{v\theta}$, $\mathbf{S}_{\theta v}$ and $\mathbf{S}_{\theta\theta}$ are triangular Toeplitz matrices of order $(n+1) \times (n+1)$ whereas \mathbf{P}_v and \mathbf{P}_θ are generally rectangular matrices of order $(n+1) \times (m+1)$. Here is the expression of these matrices for $n = 8$, $m = 4$:

$$\begin{aligned} \mathbf{S}_{vv} = EI \begin{bmatrix} 0 & 0 & \frac{-12}{\ell} & 0 & -\ell & 0 & \frac{-\ell^3}{30} & 0 & \frac{-\ell^5}{1680} \\ 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -\ell & 0 & \frac{-\ell^3}{30} & 0 \\ 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -\ell & 0 & \frac{-\ell^3}{30} \\ 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -\ell & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -\ell \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{S}_{v\theta} = EI \begin{bmatrix} 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{10} & 0 & \frac{\ell^5}{420} & 0 \\ 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{10} & 0 & \frac{\ell^5}{420} \\ 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{10} & 0 \\ 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{10} \\ 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \\ \mathbf{S}_{\theta v} = EI \begin{bmatrix} 0 & \frac{-12}{\ell} & 0 & -2\ell & 0 & \frac{-\ell^3}{10} & 0 & \frac{-\ell^5}{420} & 0 \\ 0 & 0 & \frac{-12}{\ell} & 0 & -2\ell & 0 & \frac{-\ell^3}{10} & 0 & \frac{-\ell^5}{420} \\ 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -2\ell & 0 & \frac{-\ell^3}{10} & 0 \\ 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -2\ell & 0 & \frac{-\ell^3}{10} \\ 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -2\ell & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 & -2\ell \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{-12}{\ell} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{S}_{\theta\theta} = EI \begin{bmatrix} \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{6} & 0 & \frac{\ell^5}{180} & 0 & \frac{\ell^7}{10080} \\ 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{6} & 0 & \frac{\ell^5}{180} & 0 \\ 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{6} & 0 & \frac{\ell^5}{180} \\ 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{6} & 0 \\ 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 & \frac{\ell^3}{6} \\ 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 & 2\ell \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{12}{\ell} \end{bmatrix} \end{aligned} \quad (13.74)$$

$$\mathbf{P}_v = \begin{bmatrix} \ell & 0 & \frac{\ell^3}{15} & 0 & \frac{\ell^5}{560} \\ 0 & \ell & 0 & \frac{\ell^3}{15} & 0 \\ 0 & 0 & \ell & 0 & \frac{\ell^3}{15} \\ 0 & 0 & 0 & \ell & 0 \\ 0 & 0 & 0 & 0 & \ell \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{P}_\theta = \begin{bmatrix} 0 & \frac{\ell^3}{15} & 0 & \frac{\ell^5}{315} & 0 \\ 0 & 0 & \frac{\ell^3}{15} & 0 & \frac{\ell^5}{315} \\ 0 & 0 & 0 & \frac{\ell^3}{15} & 0 \\ 0 & 0 & 0 & 0 & \frac{\ell^3}{15} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13.75)$$

Elimination of θ_j gives the condensed system

$$\mathbf{Sv}_j = \mathbf{Pq}_j, \quad \text{with} \quad \mathbf{S} = \mathbf{S}_{vv} - \mathbf{S}_{v\theta} \mathbf{S}_{\theta\theta}^{-1} \mathbf{S}_{\theta v}, \quad \mathbf{P} = \mathbf{P}_v - \mathbf{S}_{v\theta} \mathbf{S}_{\theta\theta}^{-1} \mathbf{P}_\theta. \quad (13.76)$$

$$\mathbf{S} = \mathbf{S}_{vv} - \mathbf{S}_{v\theta} \mathbf{S}_{\theta\theta}^{-1} \mathbf{S}_{\theta v} = EI \begin{bmatrix} 0 & 0 & 0 & 0 & \ell & 0 & 0 & 0 & \frac{-\ell^5}{720} \\ 0 & 0 & 0 & 0 & 0 & \ell & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \ell & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ell & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ell \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{P} = \mathbf{P}_v - \mathbf{S}_{v\theta} \mathbf{S}_{\theta v}^{-1} \mathbf{P}_\theta = \begin{bmatrix} \ell & 0 & 0 & 0 & \frac{-\ell^5}{720} \\ 0 & \ell & 0 & 0 & 0 \\ 0 & 0 & \ell & 0 & 0 \\ 0 & 0 & 0 & \ell & 0 \\ 0 & 0 & 0 & 0 & \ell \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (13.77)$$

The nontrivial differential relations¹³ given by $\mathbf{S} \mathbf{v}_j = \mathbf{P} \mathbf{q}_j$ are $EI(v_j^{iv} - \ell^4 v_j^{viii}/720) = q_j - \ell^4 q_j^{iv}/720$, $EI v_j^v = q_j'$, $EI v_j^{vi} = q_j''$, $EI v_j^{vii} = q_j'''$, and $EI v_j^{viii} = q_j^{iv}$. The first one is a truncation of the infinite order MoDE. Elimination of all v_j derivatives but v_j^{iv} yields

$$EI v_j^{iv} = q_j, \quad (13.78)$$

exactly. That this is not a fluke can be confirmed by increasing n and while taking $m = n - 4$. The first 4 columns and last 4 rows of \mathbf{S} , which are always zero, are removed to get $\hat{\mathbf{S}}$. The last 4 rows of \mathbf{P} , which are also zero, are removed to get $\hat{\mathbf{P}}$. With $m = n - 4$ both matrices are of order $n - 3 \times n - 3$. Then $\hat{\mathbf{S}} = \hat{\mathbf{P}}$ for any n , which leads immediately to (13.78). This was confirmed with *Mathematica* running n up to 24.

The foregoing analysis shows that the BE cubic element is *nodally exact for any smooth load* over a repeating lattice if consistent load computation is used. Exercise 13.18 verifies that the property is lost if elements are not identical. Numerical experiments confirm these conclusions. The Laplace transform method only works part way: it gives a different infinite order MoDE but recovers (13.78) as $n \rightarrow \infty$.

These accuracy properties are not widely known. If all beam elements are prismatic and subjected only to point loads at nodes, overall exactness follows from a theorem by Tong [665], which is not surprising since the exact solution is contained in the FEM approximation. For general distributed loads the widespread belief is that the cubic element incurs $O(\ell^4)$ errors. The first study of this nature by Waltz et. al. [692] gave the modified differential equation (MoDE) for a uniform load q as

$$v^{iv} - \frac{\ell^4}{720} v^{viii} + \dots = \frac{q}{EI}. \quad (13.79)$$

The above terms are correct. In fact a more complete expression, obtained in this study, is

$$EI \left(v^{iv} - \frac{\ell^4}{720} v^{viii} + \frac{\ell^6}{3024} v^{x} - \frac{7\ell^8}{259200} v^{xii} + \dots \right) = q - \frac{\ell^4}{720} q^{iv} + \frac{\ell^6}{3024} q^{vi} - \frac{7\ell^8}{259200} q^{viii} + \dots \quad (13.80)$$

But the conclusion that “the principal error term” is of order ℓ^4 [692,p. 1009] is incorrect. The misinterpretation is due to (13.80) being an ODE of infinite order. Truncation is fine *if followed by elimination of higher derivatives*. If this is done, the finite order MoDE (13.78) emerges regardless of where (13.80) is truncated; an obvious clue being the repetition of coefficients in both sides. The moral is that conclusions based on infinite order ODEs should be viewed with caution, unless corroborated by independent means.

¹³ Derivatives of order 4 and higher are indicated by Roman numeral superscripts instead of primes.

§13.6.2. *Accuracy of Timoshenko Beam Element

Following the same procedure it can be shown that the infinite order MoDE for a Timoshenko beam element repeating lattice with the stiffness (13.18) and consistent node forces is

$$EI \left(v_j^{iv} + \frac{\ell^2 \Phi}{12} v_j^{vi} - \frac{\ell^4 (1 + 5\Phi - 5\Phi^2)}{720} v_j^{viii} + \frac{\ell^6 (20 + 7\Phi - 70\Phi^2 + 35\Phi^3)}{60480} v_j^x + \dots \right) \\ = q_j - \frac{\ell^4 (1 + 5\Phi)}{720} q_j^{iv} + \frac{\ell^6 (20 + 14\Phi - 35\Phi^2)}{60480} q_j^{vi} + \dots, \quad \text{in which } \Phi = 12EI/(GA_s \ell^2). \quad (13.81)$$

This reduces to (13.80) if $\Phi = 0$. Elimination of higher order derivatives gives the finite order MoDE (aka FOMoDE)

$$EI v_j^{iv} = q_j - \frac{\ell^2 \Phi}{12} q_j'' = q_j - \frac{EI}{GA_s} q_j''. \quad (13.82)$$

which repeats for any $n > 8$. This happens to be the exact governing differential equation for a statically loaded Timoshenko beam [240, p. 23]. Consequently the Timoshenko beam is *nodally exact* under the same conditions previously stated for the Bernoulli-Euler model.

Notes and Bibliography

The material in this Chapter intertwines the very old and the very new. Before energy methods came to the FEM forefront by 1960 (see historical sketch in §1.7), ordinary differential equations (ODE) and flexibility methods were essential part of the toolbox repertoire of the professional structural engineer. Traces of that dominance may be found in the books by Przemieniecki [536], Pestel and Leckie [515], and the survey by Gallagher [264]. Energy derivations were popularized by Archer [31,32], Martin [416], and Melosh [433,434]. For one-dimensional elements, however, results are often identical. This can provide a valuable crosscheck.

The Legendre interpolation (13.2) was introduced in [212] [215] to study optimal mass-stiffness combinations for beam elements in the context of finite element templates [206]. The diagonal covariance matrices \mathbf{Q}_n given in (13.4) play a key role in model customization. The hinged element stiffness (13.14) is rederived in the Advanced FEM Lecture Notes [?]. using a mixed variational principle. The separation of uncoupled rigidity effects in stiffness forms such as (13.8) and (13.9) is suggested by template theory [223].

The Timoshenko beam model was originally proposed in [657]. Timoshenko cleverly packaged the model with miscellaneous ingredients introduced earlier by Bresse and Hencky. It has become important as a tool for transient response and control simulations because its dynamic form is strictly hyperbolic.¹⁴ The Timoshenko beam element stiffness (13.18) first appeared in [676] in the guise of a spar element for use in aircraft structures; the end node freedoms of that element differing from the classical set used here. The particular form (13.18) is derived in Section 5.6 of [536] using ODEs. This beam model pertains to the class of “ C^0 elements” that have been extensively studied in the FEM literature after 1968. The book of Hughes [350] provides a comprehensive treatment of such methods for beams and plates.

The classical work on beams on elastic foundations is by Hetenyi [326]. Useful solutions are tabulated in Roark-Young’s handbook [575].

That the use of homogeneous solutions of governing differential equations yields nodally-exact stiffness equations was first proven generally by Tong [665] in a Galerkin context. This derivation procedure, however, was rarely used after the 1960s. Two obstacles: (1) it is largely restricted to either one dimensional elements, or to problems with special symmetries that can be modeled with ODEs;¹⁵ (2) rapidly increasing solution

¹⁴ The Bernoulli-Euler beam dynamic model is parabolic and thus exhibits an infinite transverse wave speed. Such a model is unsuitable for wave propagation problems.

¹⁵ For example, symmetrically loaded circular plates or shells of revolution.

complexity in complicated problems discouraged hand derivations. Whereas the first limitation still holds, the increasing availability of CAS allows timely consideration of more difficult problems. The construction of the exact beam-on-Winkler-foundation element in §13.3 offers a case in point. Using *Mathematica* the complete derivation, checking and fully-symbolic testing took about 6 hours, whereas a hand derivation, coding and numerical testing would likely take weeks or months. The main application of exact elements appears to be *a priori* error estimation: how many simpler elements are needed to do the job of an exact one?

The construction of stiffness matrices from flexibility information was historically one of the first techniques by which stiffness equations of MoM members were derived. The rigid body injection method of §13.4.4 largely follows Section 6.6 of [536]. The presentation of discrete-system equilibrium theorems in §13.4.2 includes a new ingredient missing from previous work: handling non-self-equilibrated loading systems. This extension removes the 40-year-old objection that flexibility methods (or more generally, schemes based on the TCPE principle) are unable to produce equivalent or consistent node forces.

The use of equilibrium methods for multidimensional finite elements was pioneered by Fraeijs de Veubeke in the 1960s and early 1970s. His obsession with solution bounding got these methods seriously stuck, however, because of difficulties in interelement connections that maintain system-level equilibrium, as well as avoidance of spurious modes. More practical extensions lead to the so-called Trefftz and equilibrium hybrid methods. These are presently the topic of active research¹⁶ but require advanced variational techniques beyond the scope of this book. Another recent advance is the discovery of the free-free flexibility as the true dual of the free-free stiffness [210,219,211]. This extension relies heavily on projection operators.

That Hermitian BE element models are nodally exact if consistent loads are used is stated in [359, Sec. 8.3] as the “beam theorem.” Despite the name, no proof is given; only anecdotal evidence — it was likely discovered by numerical experimentation. A proof based on Fourier series appears in [240].

As the study in §13.6 illustrates, modified equation methods in boundary value problems¹⁷ are delicate and should be used with care. Their intricacies baffled Strang and Fix who, upon doing Fourier analysis of a cubic beam element, incorrectly stated [624, p. 171] that only one of the discrete equations — that for $v(x)$ — is consistent “and the others are completely inconsistent.” The alternative is the variational approach to error analysis. Although more robust and forgiving, predictions are often so conservative as to be of little value.

References

Referenced items have been moved to Appendix R.

¹⁶ Along with discontinuous Galerkin methods, a reinvention of Fraeijs de Veubeke’s weakly diffusive models.

¹⁷ They are more forgiving in initial value problems.

Homework Exercises for Chapter 13 Advanced One-Dimensional Elements

EXERCISE 13.1 [A:15] Evaluate the strain and stress fields associated with the Timoshenko beam displacement field (13.16).

EXERCISE 13.2 [A/C:20] Find the consistent node forces for a Timoshenko beam element under a uniform transverse load q . Hint: find \mathbf{f}_c for the Legendre interpolation, then premultiply by \mathbf{H}_s^T .

EXERCISE 13.3 [A/C:20] Construct the stiffness matrix a Timoshenko plane beam element with a hinge at the center. Hint: set $R_{Bs} = 0$, $R_{Ba} = EI$ and $R_s = 12EI/(\Phi\ell^2)$ in (13.13).

EXERCISE 13.4 [A/C:15=5+10] Consider a cantilever beam of constant bending rigidity EI , constant shear rigidity GA_s and length L , which is modeled with one Timoshenko element. The beam is end loaded by a transverse force P . The support conditions are $v_1 = \theta_1 = 0$.

- (a) Find the end displacement v_2 and end rotation θ_2 in terms of P , E , G , I , A_s and L . Compare with the analytical values $PL^3/(3EI) + PL/(GA_s)$ and $PL^2/(2EI)$, respectively.
- (b) Why does the finite element model provides the exact answer with one element?

EXERCISE 13.5 [A:25] (Requires math ability). Discuss what happens in (13.18) if $\Phi \rightarrow \infty$. Is the result useful for a shear-only “spar” element? Hint: eliminate θ_1 and θ_2 by a master-slave MFC.

EXERCISE 13.6 [A:10] For a given number of elements N^e of length $\ell = 2L/N^e$, relate χ and λ in Example 13.2.

EXERCISE 13.7 [A/C:40] (research paper level). Derive an exact Timoshenko-beam-on-Winkler-foundation equation method.element using the differential equation method.

EXERCISE 13.8 [C:30] Write *Mathematica* code to verify the nodal exactness conclusion of §13.6.1 using the repeated differentiation approach.

EXERCISE 13.9 [C:30] As above, but using the Laplace transform. Show that this only does half the job.

EXERCISE 13.10 [A/C:35] Find the general symbolic expression of the terms in

EXERCISE 13.11 [A/C:40] (research paper level) Analyze nodal accuracy if the length of the beam elements $(1 \pm \alpha)\ell$, where $0 \leq \alpha \leq \frac{1}{2}$.

EXERCISE 13.12 [A/C:35] Using *Mathematica*, verify the results (13.81) and (13.82) in §13.6.2.

14

The Plane Stress Problem

TABLE OF CONTENTS

	Page
§14.1. Introduction	14–3
§14.2. Plate in Plane Stress	14–3
§14.2.1. Behavioral Assumptions	14–3
§14.2.2. Mathematical Model	14–4
§14.2.3. Problem Data	14–5
§14.2.4. Problem Unknowns	14–5
§14.3. Plane Stress Governing Equations	14–6
§14.3.1. Governing Equations	14–6
§14.3.2. Boundary Conditions	14–7
§14.3.3. Weak versus Strong Form	14–8
§14.3.4. Total Potential Energy	14–9
§14.4. Finite Element Equations	14–10
§14.4.1. Displacement Interpolation	14–10
§14.4.2. Element Energy	14–11
§14.4.3. Element Stiffness Equations	14–12
§14. Notes and Bibliography.	14–12
§14. References	14–12
§14. Exercises	14–13

§14.1. Introduction

We now pass to the variational formulation of two-dimensional *continuum* finite elements. The problem of *plane stress* will serve as the vehicle for illustrating such formulations. As narrated in Appendix O, continuum-based structural finite elements were invented in the aircraft industry (at Boeing during the early 1950s) to solve this kind of problem when it arose in the design and analysis of delta wing panels [727].

The problem is presented here within the framework of the linear theory of elasticity.

§14.2. Plate in Plane Stress

In structural mechanics, a flat thin sheet of material is called a *plate*.¹ The distance between the plate faces is the *thickness*, denoted by h . The *midplane* lies halfway between the two faces.

The direction normal to the midplane is the *transverse* direction. Directions parallel to the midplane are called *in-plane* directions. The global axis z is oriented along the transverse direction. Axes x and y are placed in the midplane, forming a right-handed Rectangular Cartesian Coordinate (RCC) system. Thus the equation of the midplane is $z = 0$. The $+z$ axis conventionally defines the *top surface* of the plate as the one that it intersects, whereas the opposite surface is called the *bottom surface*. See Figure 14.1(a).

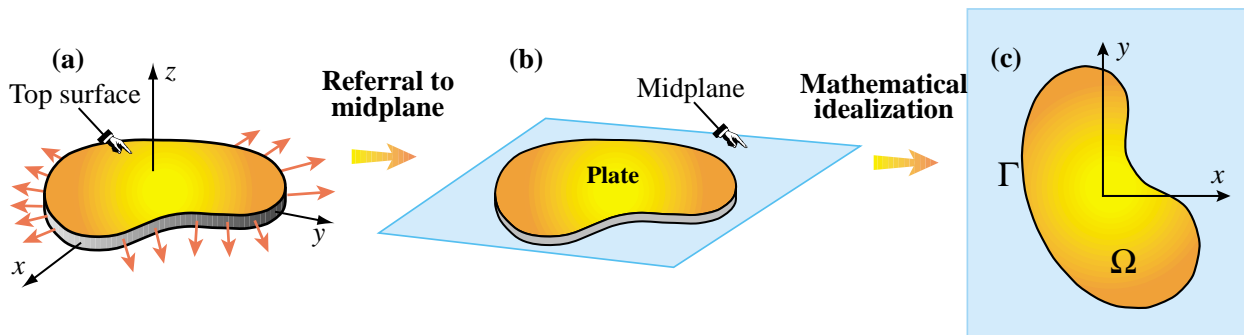


FIGURE 14.1. A plate structure in plane stress: (a) configuration; (b) referral to its midplane; (c) 2D mathematical idealization as boundary value problem.

§14.2.1. Behavioral Assumptions

A plate loaded in its midplane is said to be in a state of *plane stress*, or a *membrane state*, if the following assumptions hold:

1. All loads applied to the plate act in the midplane direction, and are symmetric with respect to the midplane.
2. All support conditions are symmetric about the midplane.
3. In-plane displacements, strains and stresses can be taken to be uniform through the thickness.
4. The normal and shear stress components in the z direction are zero or negligible.

¹ If it is relatively thick, as in concrete pavements or Argentinian beefsteaks, the term *slab* is also used but not usually for plane stress conditions.

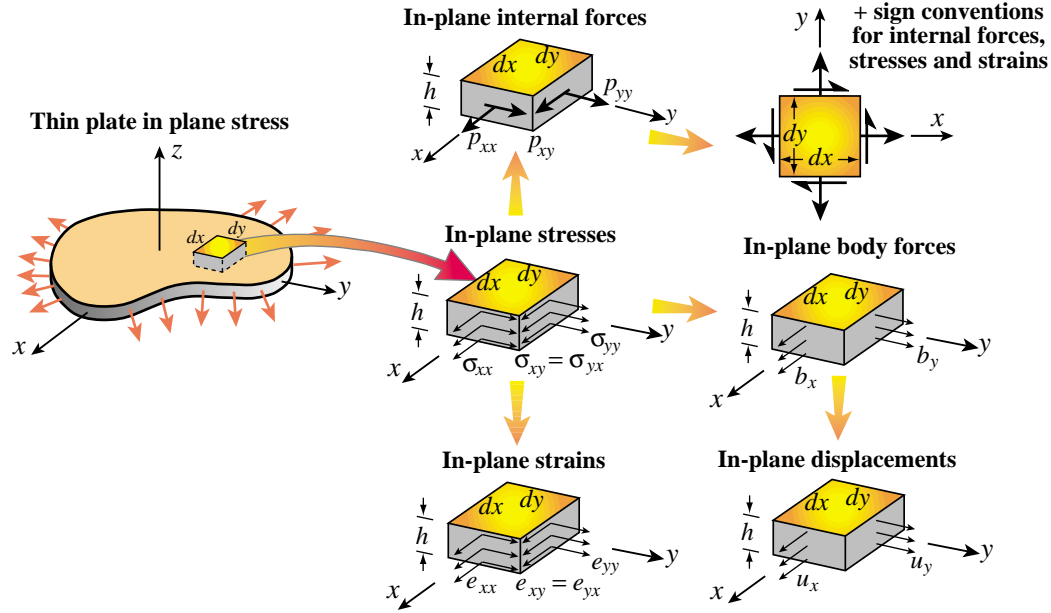


FIGURE 14.2. Notational conventions for in-plane stresses, strains, displacements and internal forces of a thin plate in plane stress.

The last two assumptions are not necessarily consequences of the first two. For the latter to hold, the thickness h should be small, typically 10% or less, than the shortest in-plane dimension. If the plate thickness varies it should do so gradually. Finally, the plate fabrication must exhibit symmetry with respect to the midplane.

To these four assumptions we add the following restriction:

5. The plate is fabricated of the same material through the thickness. Such plates are called *transversely homogeneous* or (in aerospace) *monocoque* plates.

The last assumption excludes wall constructions of importance in aerospace, in particular composite and honeycomb sandwich plates. The development of mathematical models for such configurations requires a more complicated integration over the thickness as well as the ability to handle coupled bending and stretching effects, and will not be considered here.

Remark 14.1. Selective relaxation from assumption 4 leads to the so-called *generalized plane stress state*, in which z stresses are accepted. The *plane strain state* is obtained if strains in the z direction are precluded. Although the construction of finite element models for those states has many common points with plane stress, we shall not consider those models here. For isotropic materials the plane stress and plane strain problems can be mapped into each other through a fictitious-property technique; see Exercise 14.1.

Remark 14.2. Transverse loading on a plate produces *plate bending*, which is associated with a more complex configuration of internal forces and deformations. This subject is studied in [245].

§14.2.2. Mathematical Model

The mathematical model of the plate in plane stress is set up as a two-dimensional boundary value problem (BVP), in which the plate is projected onto its midplane; see Figure 14.1(b). This allows to formulate the BVP over a plane domain Ω with a boundary Γ , as illustrated in Figure 14.1(c).

In this idealization the third dimension is represented as functions of x and y that are *integrated through the plate thickness*. Engineers often work with internal plate forces, which result from integrating the in-plane stresses through the thickness. See Figure 14.2.

§14.2.3. Problem Data

The following summarizes the givens in the plate stress problem.

Domain geometry. This is defined by the boundary Γ illustrated in Figure 14.1(c).

Thickness. Most plates used as structural components have constant thickness. If the thickness does vary, in which case $h = h(x, y)$, it should do so gradually to maintain the plane stress state. Sudden changes in thickness may lead to stress concentrations.

Material data. This is defined by the constitutive equations. Here we shall assume that the plate material is linearly elastic but not necessarily isotropic.

Specified Interior Forces. These are known forces that act in the interior Ω of the plate. There are of two types. *Body forces* or *volume forces* are forces specified per unit of plate volume; for example the plate weight. *Face forces* act tangentially to the plate faces and are transported to the midplane. For example, the friction or drag force on an airplane skin is of this type if the skin is modeled to be in plane stress.

Specified Surface Forces. These are known forces that act on the boundary Γ of the plate. In elasticity they are called *surface tractions*. In actual applications it is important to know whether these forces are specified per unit of surface area or per unit length. The former may be converted to the latter by multiplying through the appropriate thickness value.

Displacement Boundary Conditions. These specify how the plate is supported. Points subject to support conditions may be fixed, allowed to move in one direction, or subject to multipoint constraints. Also symmetry and antisymmetry lines may be identified as discussed in Chapter 8 of IFEM [247].

If no displacement boundary conditions are imposed, the plate is said to be *free-free* or *floating*.

§14.2.4. Problem Unknowns

The unknown fields are displacements, strains and stresses. Because of the assumed wall fabrication homogeneity the in-plane components are assumed to be *uniform through the plate thickness*. Thus the dependence on z disappears and all such components become functions of x and y only.

Displacements. The in-plane displacement field is defined by two components:

$$\mathbf{u}(x, y) = \begin{bmatrix} u_x(x, y) \\ u_y(x, y) \end{bmatrix} \quad (14.1)$$

The transverse displacement component $u_z(x, y, z)$ component is generally nonzero because of Poisson's ratio effects, and depends on z . However, this displacement does not appear in the governing equations.

Strains. The in-plane strain field forms a tensor defined by three independent components: e_{xx} , e_{yy} and e_{xy} . To allow stating the FE equations in matrix form, these components are cast to form a

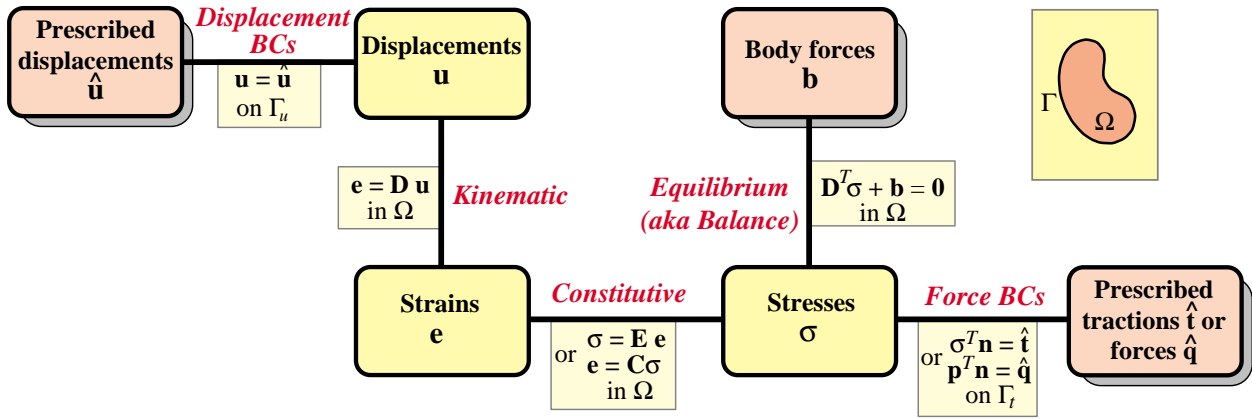


FIGURE 14.3. The Strong Form of the plane stress equations of linear elastostatics displayed as a Tonti diagram. Yellow boxes identify prescribed fields whereas orange boxes denote unknown fields. The distinction between Strong and Weak Forms is explained in §14.3.3.

3-component “strain vector”

$$\mathbf{e}(x, y) = \begin{bmatrix} e_{xx}(x, y) \\ e_{yy}(x, y) \\ 2e_{xy}(x, y) \end{bmatrix} \quad (14.2)$$

The factor of 2 in e_{xy} shortens strain energy expressions. The shear strain components e_{xz} and e_{yz} vanish. The transverse normal strain e_{zz} is generally nonzero because of Poisson’s ratio effects. This strain does not enter the governing equations as unknown, however, because the associated stress σ_{zz} is zero. This eliminates the contribution of $\sigma_{zz}e_{zz}$ to the internal energy.

Stresses. The in-plane stress field forms a tensor defined by three independent components: σ_{xx} , σ_{yy} and σ_{xy} . As in the case of strains, to allow stating the FE equations in matrix form, these components are cast to form a 3-component “stress vector”

$$\boldsymbol{\sigma}(x, y) = \begin{bmatrix} \sigma_{xx}(x, y) \\ \sigma_{yy}(x, y) \\ \sigma_{xy}(x, y) \end{bmatrix} \quad (14.3)$$

The remaining three stress components: σ_{zz} , σ_{xz} and σ_{yz} , are assumed to vanish.

The *plate internal forces* are obtained on integrating the stresses through the thickness. Under the assumption of uniform stress distribution,

$$p_{xx} = \sigma_{xx}h, \quad p_{yy} = \sigma_{yy}h, \quad p_{xy} = \sigma_{xy}h. \quad (14.4)$$

These p ’s also form a tensor. They are called *membrane forces* in the literature. See Figure 14.2.

§14.3. Plane Stress Governing Equations

We shall develop plane stress finite elements in the framework of classical linear elasticity. The necessary governing equations are presented below. They are graphically represented in the Strong Form Tonti diagram of Figure 14.3.

§14.3.1. Governing Equations

The three internal fields: displacements, strains and stresses (14.1)–(14.3) are connected by three field equations: kinematic, constitutive and internal-equilibrium equations. If initial strain effects are ignored, these equations read

$$\begin{aligned} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} &= \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix}, \\ \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} &= \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \\ \begin{bmatrix} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix}. \end{aligned} \quad (14.5)$$

The compact matrix version of (14.5) is

$$\boxed{\mathbf{e} = \mathbf{D} \mathbf{u}, \quad \boldsymbol{\sigma} = \mathbf{E} \mathbf{e}, \quad \mathbf{D}^T \boldsymbol{\sigma} + \mathbf{b} = \mathbf{0},} \quad (14.6)$$

Here $\mathbf{E} = \mathbf{E}^T$ is the 3×3 stress-strain matrix of plane stress elastic moduli, \mathbf{D} is the 3×2 symmetric-gradient operator and its transpose the 2×3 tensor-divergence operator.²

If the plate material is isotropic with elastic modulus E and Poisson's ratio ν , the moduli in the constitutive matrix \mathbf{E} reduce to $E_{11} = E_{22} = E/(1 - \nu^2)$, $E_{33} = \frac{1}{2}E/(1 + \nu) = G$, $E_{12} = \nu E_{11}$ and $E_{13} = E_{23} = 0$. See also Exercise 14.1.

§14.3.2. Boundary Conditions

Boundary conditions prescribed on Γ may be of two types: displacement BC or force BC (the latter is also called stress BC or traction BC). To write down those conditions it is conceptually convenient to break up Γ into two subsets: Γ_u and Γ_t , over which displacements and force or stresses, respectively, are specified. See Figure 14.4.

Displacement boundary conditions are prescribed on Γ_u in the form

$$\boxed{\mathbf{u} = \hat{\mathbf{u}}.} \quad (14.7)$$

Here $\hat{\mathbf{u}}$ are prescribed displacements. Often $\hat{\mathbf{u}} = \mathbf{0}$. This happens in fixed portions of the boundary, as the ones illustrated in Figure 14.4.

Force boundary conditions (also called stress BCs and traction BCs in the literature) are specified on Γ_t . They take the form

$$\boxed{\boldsymbol{\sigma}_n = \hat{\mathbf{t}}.} \quad (14.8)$$

Here $\hat{\mathbf{t}}$ are prescribed surface tractions specified as a force per unit area (that is, not integrated through the thickness), and $\boldsymbol{\sigma}_n$ is the stress vector shown in Figure 14.4.

² The dependence on (x, y) has been omitted to reduce clutter.

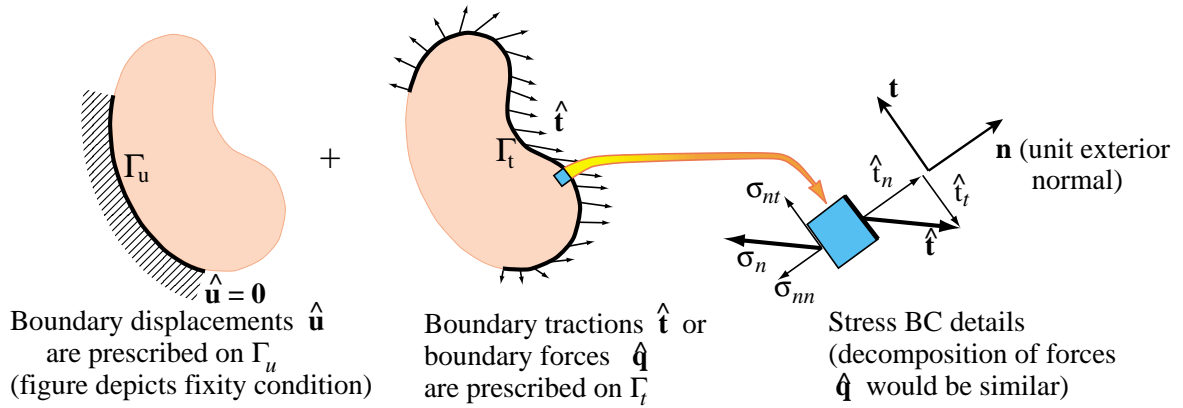


FIGURE 14.4. Displacement and force (stress, traction) boundary conditions for the plane stress problem.

An alternative form of (14.8) uses the internal plate forces:

$$\mathbf{p}_n = \hat{\mathbf{q}}. \quad (14.9)$$

Here $\mathbf{p}_n = \sigma_n h$ and $\hat{\mathbf{q}} = \hat{\mathbf{t}} h$. This form is used more often than (14.8) in structural design, particularly when the plate wall construction is inhomogeneous.

The components of σ_n in Cartesian coordinates follow from Cauchy's stress transformation formula

$$\sigma_n = \begin{bmatrix} \sigma_{xx}n_x + \sigma_{xy}n_y \\ \sigma_{xy}n_x + \sigma_{yy}n_y \end{bmatrix} = \begin{bmatrix} n_x & 0 & n_y \\ 0 & n_y & n_x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \quad (14.10)$$

in which n_x and n_y denote the Cartesian components of the unit normal vector \mathbf{n}^e (also called the direction cosines of the normal). Thus (14.8) splits into two scalar conditions: $\hat{t}_x = \sigma_{nx}$ and $\hat{t}_y = \sigma_{ny}$. The derivation of (14.10) is the subject of Exercise 14.4.

It is sometimes convenient to write the condition (14.8) in terms of normal n and tangential t directions:

$$\sigma_{nn} = \hat{t}_n, \quad \sigma_{nt} = \hat{t}_t \quad (14.11)$$

in which $\sigma_{nn} = \sigma_{nx}n_x + \sigma_{ny}n_y$ and $\sigma_{nt} = -\sigma_{nx}n_y + \sigma_{ny}n_x$. See Figure 14.4.

Remark 14.3. The separation of Γ into Γ_u and Γ_t is useful for conciseness in the mathematical formulation, such as the energy integrals presented below. It does not exhaust, however, all BC possibilities. Frequently at points of Γ one specifies a displacement in one direction and a force (or stress) in the other. An example of these are roller and sliding conditions as well as lines of symmetry and antisymmetry. These are called *mixed displacement-traction* BC. To cover these situations one needs either a generalization of the boundary split, in which Γ_u and Γ_t are permitted to overlap, or to define another portion Γ_m for 'mixed conditions'. Such generalizations will not be presented here, as they become unimportant once the FE discretization is done.

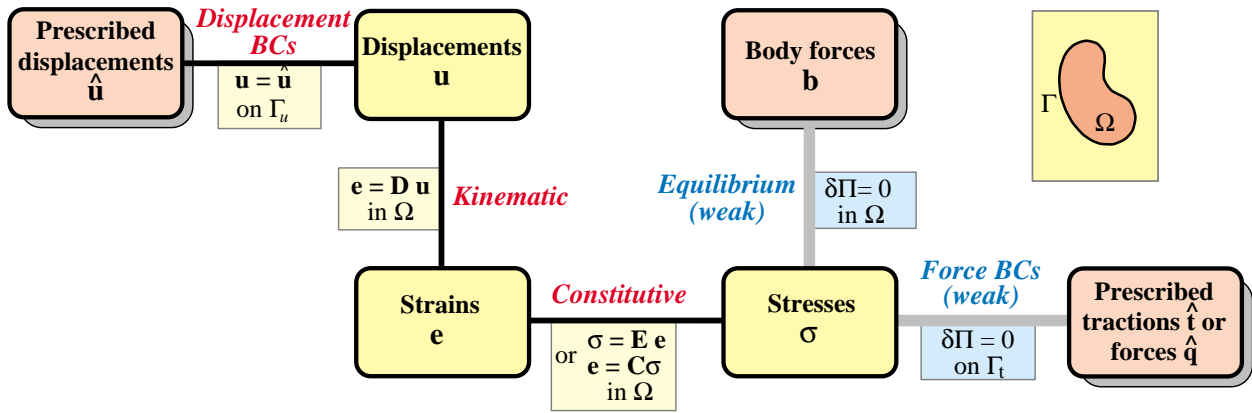


FIGURE 14.5. The TPE-based Weak Form of the plane stress equations of linear elastostatics. Weak links are marked with grey lines.

§14.3.3. Weak versus Strong Form

We introduce now some further terminology from variational calculus. The Tonti diagram of Figure 14.3 is said to display the *Strong Form* of the governing equations because all relations are verified point by point. These relations, called *strong links*, are shown in the diagram with black lines.

A Weak Form is obtained by *relaxing* one or more strong links, as briefly described in Chapter 11. Those are replaced by *weak links*, which enforce relations in an average or integral sense rather than point by point. The weak links are then provided by the variational formulation chosen for the problem. Because in general many variational forms of the same problem are possible, there are many possible Weak Forms. On the other hand the Strong Form is unique.

The Weak Form associated with the Total Potential Energy (TPE) variational form is illustrated in Figure 14.5. The internal equilibrium equations and stress BC become weak links, which are drawn by gray lines. These equations are given by the variational statement $\delta\Pi = 0$, where the TPE functional Π is given in the next subsection. The FEM displacement formulation discussed below is based on this particular Weak Form.

§14.3.4. Total Potential Energy

The Total Potential Energy functional for the plane stress problem is given by

$$\Pi = U - W. \quad (14.12)$$

The internal energy can be expressed in terms of the strains only as

$$U = \frac{1}{2} \int_{\Omega} h \boldsymbol{\sigma}^T \mathbf{e} d\Omega = \frac{1}{2} \int_{\Omega} h \mathbf{e}^T \mathbf{E} \mathbf{e} d\Omega. \quad (14.13)$$

in which $\frac{1}{2} \mathbf{e}^T \mathbf{E} \mathbf{e}$ is the *strain energy density*. The derivation details are relegated to Exercise 14.5. The external energy (potential of the applied forces) is the sum of contributions from the given

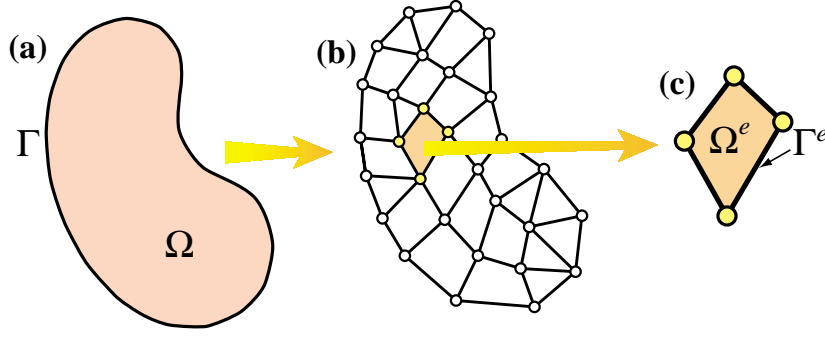


FIGURE 14.6. Finite element discretization and extraction of generic element.

interior (body) and exterior (boundary) forces:

$$W = \int_{\Omega} h \mathbf{u}^T \mathbf{b} d\Omega + \int_{\Gamma_t} h \mathbf{u}^T \hat{\mathbf{t}} d\Gamma. \quad (14.14)$$

Note that the boundary integral over Γ is taken only over Γ_t . That is, the portion of the boundary over which tractions or forces are specified.

§14.4. Finite Element Equations

The necessary equations to apply the finite element method to the plane stress problem are collected here and expressed in matrix form. The domain of Figure 14.6(a) is discretized by a finite element mesh as illustrated in Figure 14.6(b). From this mesh we extract a generic element labeled e with $n \geq 3$ node points. In subsequent derivations the number n is kept *arbitrary*. Therefore, the formulation is applicable to arbitrary two-dimensional elements, for example those sketched in Figure 14.7.

To comfortably accommodate general element types, the node points will be labeled 1 through n . These are called *local node numbers*. Numbering will always start with corners.

The element domain and boundary are denoted by Ω^e and Γ^e , respectively. The element has $2n$ degrees of freedom. These are collected in the element node displacement vector in a node by node arrangement:

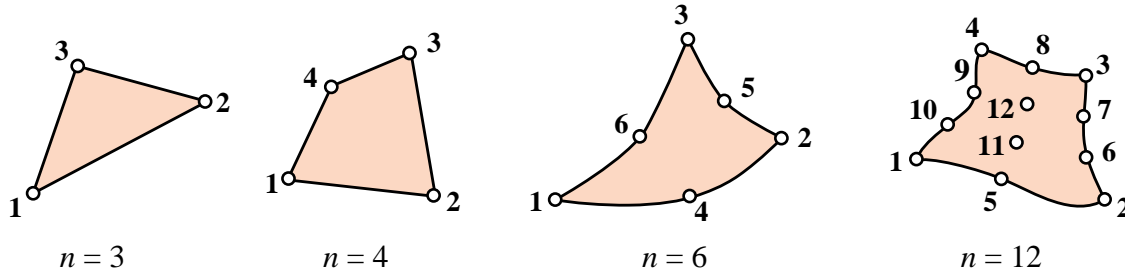
$$\mathbf{u}^e = [u_{x1} \quad u_{y1} \quad u_{x2} \quad \dots \quad u_{xn} \quad u_{yn}]^T. \quad (14.15)$$

§14.4.1. Displacement Interpolation

The displacement field $\mathbf{u}^e(x, y)$ over the element is interpolated from the node displacements. We shall assume that the same interpolation functions are used for both displacement components.³ Thus

$$u_x(x, y) = \sum_{i=1}^n N_i^e(x, y) u_{xi}, \quad u_y(x, y) = \sum_{i=1}^n N_i^e(x, y) u_{yi}, \quad (14.16)$$

³ This is the so called *element isotropy* condition, which is studied and justified in advanced FEM courses.

FIGURE 14.7. Example plane stress finite elements, characterized by their number of nodes n .

in which $N_i^e(x, y)$ are the element shape functions. In matrix form:

$$\mathbf{u}(x, y) = \begin{bmatrix} u_x(x, y) \\ u_y(x, y) \end{bmatrix} = \begin{bmatrix} N_1^e & 0 & N_2^e & 0 & \cdots & N_n^e & 0 \\ 0 & N_1^e & 0 & N_2^e & \cdots & 0 & N_n^e \end{bmatrix} \mathbf{u}^e = \mathbf{N} \mathbf{u}^e. \quad (14.17)$$

This \mathbf{N} (with superscript e omitted to reduce clutter) is called the *shape function matrix*. It has dimensions $2 \times 2n$. For example, if the element has 4 nodes, \mathbf{N} is 2×8 .

The *interpolation condition* on the element shape function $N_i^e(x, y)$ states that it must take the value one at the i^{th} node and zero at all others. This ensures that the interpolation (14.17) is correct at the nodes. Additional requirements on the shape functions are stated in later Chapters.

Differentiating the finite element displacement field yields the strain-displacement relations:

$$\mathbf{e}(x, y) = \begin{bmatrix} \frac{\partial N_1^e}{\partial x} & 0 & \frac{\partial N_2^e}{\partial x} & 0 & \cdots & \frac{\partial N_n^e}{\partial x} & 0 \\ 0 & \frac{\partial N_1^e}{\partial y} & 0 & \frac{\partial N_2^e}{\partial y} & \cdots & 0 & \frac{\partial N_n^e}{\partial y} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_2^e}{\partial x} & \cdots & \frac{\partial N_n^e}{\partial y} & \frac{\partial N_n^e}{\partial x} \end{bmatrix} \mathbf{u}^e = \mathbf{B} \mathbf{u}^e. \quad (14.18)$$

This $\mathbf{B} = \mathbf{D} \mathbf{N}$ is called the *strain-displacement matrix*. It is dimensioned $3 \times 2n$. For example, if the element has 6 nodes, \mathbf{B} is 3×12 . The stresses are given in terms of strains and displacements by $\boldsymbol{\sigma} = \mathbf{E} \mathbf{e} = \mathbf{E} \mathbf{B} \mathbf{u}^e$, which is assumed to hold at all points of the element.

§14.4.2. Element Energy

To obtain finite element stiffness equations, the variation of the TPE functional is decomposed into contributions from individual elements:

$$\delta \Pi^e = \delta U^e - \delta W^e = 0. \quad (14.19)$$

in which

$$U^e = \frac{1}{2} \int_{\Omega^e} h \boldsymbol{\sigma}^T \mathbf{e} d\Omega^e = \frac{1}{2} \int_{\Omega^e} h \mathbf{e}^T \mathbf{E} \mathbf{e} d\Omega^e \quad (14.20)$$

and

$$W^e = \int_{\Omega^e} h \mathbf{u}^T \mathbf{b} d\Omega^e + \int_{\Gamma^e} h \mathbf{u}^T \hat{\mathbf{t}} d\Gamma^e \quad (14.21)$$

Note that in (14.21) Γ_l^e has been taken equal to the *complete boundary* Γ^e of the element. This is a consequence of the fact that displacement boundary conditions are applied *after* assembly, to a free-free structure. Consequently it does not harm to assume that all boundary conditions are of stress type insofar as forming the element equations.

§14.4.3. Element Stiffness Equations

Inserting the relations $\mathbf{u} = \mathbf{N}\mathbf{u}^e$, $\mathbf{e} = \mathbf{B}\mathbf{u}^e$ and $\boldsymbol{\sigma} = \mathbf{E}\mathbf{e}$ into Π^e yields the quadratic form in the nodal displacements

$$\Pi^e = \frac{1}{2} \mathbf{u}^{eT} \mathbf{K}^e \mathbf{u}^e - \mathbf{u}^{eT} \mathbf{f}^e. \quad (14.22)$$

Here the element stiffness matrix is

$$\mathbf{K}^e = \int_{\Omega^e} h \mathbf{B}^T \mathbf{E} \mathbf{B} d\Omega^e, \quad (14.23)$$

and the consistent element nodal force vector is

$$\mathbf{f}^e = \int_{\Omega^e} h \mathbf{N}^T \mathbf{b} d\Omega^e + \int_{\Gamma^e} h \mathbf{N}^T \hat{\mathbf{t}} d\Gamma^e. \quad (14.24)$$

In the second integral of (14.24) the matrix \mathbf{N} is evaluated on the element boundary only.

The calculation of the entries of \mathbf{K}^e and \mathbf{f}^e for several elements of historical or practical interest is described in subsequent Chapters.

Notes and Bibliography

The plane stress problem is well suited for introducing continuum finite elements, from both historical and technical standpoints. Some books use the Poisson equation for this purpose, but problems such as heat conduction cannot illustrate features such as vector-mixed boundary conditions and shear effects.

The first continuum structural finite elements were developed at Boeing in the early 1950s to model delta-wing skin panels [143,727]. A plane stress model was naturally chosen for the panels. The paper that gave the method its name [134] used the plane stress problem as application driver.

The technical aspects of plane stress can be found in any book on elasticity. A particularly readable one is the excellent textbook by Fung [278], which is unfortunately out of print.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 14

The Plane Stress Problem

EXERCISE 14.1 [A+C:15] Suppose that the structural material is isotropic, with elastic modulus E and Poisson's ratio ν . The in-plane stress-strain relations for plane stress ($\sigma_{zz} = \sigma_{xz} = \sigma_{yz} = 0$) and plane strain ($e_{zz} = e_{xz} = e_{yz} = 0$) as given in any textbook on elasticity, are

$$\begin{aligned} \text{plane stress: } \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} &= \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \\ \text{plane strain: } \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} &= \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1}{2}(1-2\nu) \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}. \end{aligned} \quad (\text{E14.1})$$

Show that the constitutive matrix of plane strain can be formally obtained by replacing E by a fictitious modulus E^* and ν by a fictitious Poisson's ratio ν^* in the plane stress constitutive matrix. Find the expression of E^* and ν^* in terms of E and ν .

You may also chose to answer this exercise by doing the inverse process: go from plane strain to plain stress by replacing a fictitious modulus and Poisson's ratio in the plane strain constitutive matrix.

This device permits “reusing” a plane stress FEM program to do plane strain, or vice-versa, as long as the material is isotropic.

Partial answer to go from plane stress to plane strain: $\nu^* = \nu/(1-\nu)$.

EXERCISE 14.2 [A:25] In the finite element formulation of near incompressible isotropic materials (as well as plasticity and viscoelasticity) it is convenient to use the so-called *Lamé constants* λ and μ instead of E and ν in the constitutive equations. Both λ and μ have the physical dimension of stress and are related to E and ν by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = G = \frac{E}{2(1+\nu)}. \quad (\text{E14.2})$$

Conversely

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \quad \nu = \frac{\lambda}{2(\lambda + \mu)}. \quad (\text{E14.3})$$

Substitute (E14.3) into both of (E14.1) to express the two stress-strain matrices in terms of λ and μ . Then split the stress-strain matrix \mathbf{E} of plane strain as

$$\mathbf{E} = \mathbf{E}_\mu + \mathbf{E}_\lambda \quad (\text{E14.4})$$

in which \mathbf{E}_μ and \mathbf{E}_λ contain only μ and λ , respectively, with \mathbf{E}_μ diagonal and $E_{\lambda 33} = 0$. This is the Lamé or $\{\lambda, \mu\}$ splitting of the plane strain constitutive equations, which leads to the so-called **B**-bar formulation of near-incompressible finite elements.⁴ Express \mathbf{E}_μ and \mathbf{E}_λ also in terms of E and ν .

For the plane stress case perform a similar splitting in which where \mathbf{E}_λ contains only $\bar{\lambda} = 2\lambda\mu/(\lambda + 2\mu)$ with $E_{\lambda 33} = 0$, and \mathbf{E}_μ is a diagonal matrix function of μ and $\bar{\lambda}$.⁵ Express \mathbf{E}_μ and \mathbf{E}_λ also in terms of E and ν .

⁴ Equation (E14.4) is sometimes referred to as the deviatoric+volumetric splitting of the stress-strain law, on account of its physical meaning in plane strain. That interpretation is not fully accurate, however, for plane stress.

⁵ For the physical significance of $\bar{\lambda}$ see [658, pp. 254ff].

EXERCISE 14.3 [A:20] Include thermoelastic effects in the plane stress constitutive field equations, assuming a thermally isotropic material with coefficient of linear expansion α . Hint: start from the two-dimensional Hooke's law including temperature:

$$e_{xx} = \frac{1}{E}(\sigma_{xx} - \nu\sigma_{yy}) + \alpha \Delta T, \quad e_{yy} = \frac{1}{E}(\sigma_{yy} - \nu\sigma_{xx}) + \alpha \Delta T, \quad 2e_{xy} = \sigma_{xy}/G, \quad (\text{E14.5})$$

in which $\Delta T = \Delta T(x, y)$ and $G = \frac{1}{2}E/(1 + \nu)$. Solve for stresses and collect effects of ΔT in one vector of “thermal stresses.”

EXERCISE 14.4 [A:15] Derive the Cauchy stress-to-traction equations (?) using force equilibrium along x and y and the geometric relations shown in Figure E14.1. (This is the “wedge method” in Mechanics of Materials.)

Hint: $t_x ds = \sigma_{xx} dy + \sigma_{xy} dx$, etc.

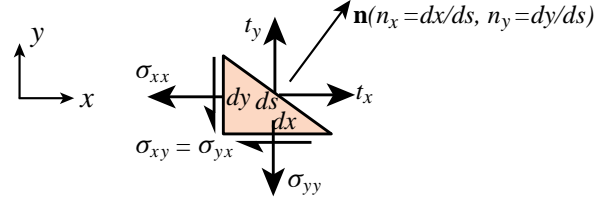


FIGURE E14.1. Geometry for deriving (?).

EXERCISE 14.5 [A:25=5+5+15] A linearly elastic plate is in plane stress. It is shown in courses in elasticity that the internal strain energy density stored per unit volume of the plate expressed in terms of stresses and strains is the bilinear form

$$\mathcal{U} = \frac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + \sigma_{xy}e_{xy} + \sigma_{yx}e_{yx}) = \frac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + 2\sigma_{xy}e_{xy}) = \frac{1}{2}\boldsymbol{\sigma}^T \mathbf{e}. \quad (\text{E14.6})$$

- (a) Show that (E14.6) can be written in terms of strains only as

$$\mathcal{U} = \frac{1}{2}\mathbf{e}^T \mathbf{E} \mathbf{e}, \quad (\text{E14.7})$$

thus justifying the strain energy density expression given in (14.13) for the plane stress problem.

- (b) Show that (E14.6) can be written in terms of stresses only as

$$\mathcal{U} = \frac{1}{2}\boldsymbol{\sigma}^T \mathbf{C} \boldsymbol{\sigma}, \quad (\text{E14.8})$$

in which $\mathbf{C} = \mathbf{E}^{-1}$ is the elastic compliance (strain-stress) matrix.

- (c) Suppose you want to write (E14.6) in terms of the extensional strains $\{e_{xx}, e_{yy}\}$ and of the shear stress $\sigma_{xy} = \sigma_{yx}$. This is known as a *mixed* representation. Show that

$$\mathcal{U} = \frac{1}{2} \begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}^T \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ A_{13} & A_{23} & A_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}, \quad (\text{E14.9})$$

and explain how the entries A_{ij} of the kernel matrix \mathbf{A} that appears in (E14.9) can be calculated⁶ in terms of the elastic moduli E_{ij} .

⁶ The process of computing \mathbf{A} is an instance of “partial inversion” of the elasticity matrix \mathbf{E} . It is closely related to the *Schur complement* concept covered in Appendix P.

Note the following Table list relations between commonly used moduli for isotropic linear elastic material. Here K is the bulk modulus whereas M is the P-wave modulus used in seismology.

	(λ, μ)	(E, μ)	(K, λ)	(K, μ)	(λ, ν)	(μ, ν)	(E, ν)	(K, ν)	(K, E)
$K =$	$\lambda + \frac{2\mu}{3}$	$\frac{E\mu}{3(3\mu-E)}$			$\lambda \frac{1+\nu}{3\nu}$	$\frac{2\mu(1+\nu)}{3(1-2\nu)}$	$\frac{E}{3(1-2\nu)}$		
$E =$	$\mu \frac{3\lambda+2\mu}{\lambda+\nu}$		$9K \frac{K-\lambda}{3K-\lambda}$	$\frac{9K\mu}{3K+\mu}$		$\frac{\lambda(1+\nu)(1-2\nu)}{\nu}$	$2\mu(1+\nu)$		$3K(1-2\nu)$
$\lambda =$		$\mu \frac{E-2\mu}{3\mu-E}$		$K - \frac{2\mu}{3}$		$\frac{2\mu\nu}{1-2\nu}$	$\frac{E\nu}{(1+\nu)(1-2\nu)}$	$\frac{3K}{1+\nu}$	$\frac{3K(3K-E)}{9K-E}$
$\mu = G =$			$\mu \frac{K-\lambda}{2}$		$\lambda \frac{K-\lambda}{3K-\lambda}$	$\frac{9K\mu}{3K+\mu}$	$\frac{\lambda(1-2\nu)}{2\nu}$	$\frac{E}{2(1+\nu)}$	$3K \frac{(1-2\nu)}{2(1+\nu)}$
$\nu =$		$\frac{\lambda}{2(\lambda+\mu)}$	$\frac{E}{2\mu} - 1$	$\frac{\lambda}{3K-\lambda}$	$\frac{3K-2\mu}{2(3K+\mu)}$				$\frac{3K-E}{6K}$
$M =$	$\lambda+2\mu$	$\mu \frac{4\mu-E}{3\mu-E}$	$3K-2\lambda$	$K + \frac{4\mu}{3}$	$\lambda \frac{1-\nu}{\nu}$	$\mu \frac{2-2\nu}{1-2\nu}$	$\frac{E(1-\nu)}{(1+\nu)(1-2\nu)}$	$3K \frac{1-\nu}{1+\nu}$	$3K \frac{3K+E}{9K-E}$

(E14.10)

15

Three-Node Plane Stress Triangles

TABLE OF CONTENTS

	Page
§15.1. Introduction	15-3
§15.2. Background	15-3
§15.2.1. Parametric Representation of Functions	15-3
§15.2.2. Geometry	15-4
§15.2.3. Triangular Coordinates	15-4
§15.2.4. Linear Interpolation	15-5
§15.2.5. Coordinate Transformations	15-5
§15.2.6. Partial Derivatives	15-6
§15.2.7. *Homogeneous Polynomials in Triangular Coordinates	15-7
§15.2.8. *Interesting Points and Lines	15-7
§15.3. The Turner Triangle	15-8
§15.3.1. Strain-Displacement Equations	15-8
§15.3.2. Stress-Strain Equations	15-8
§15.3.3. The Stiffness Matrix	15-9
§15.3.4. The Consistent Nodal Force Vector	15-9
§15.3.5. Implementation	15-10
§15.3.6. *Consistency Verification	15-11
§15.3.7. *Checking Continuity	15-11
§15.3.8. *Checking Completeness	15-12
§15.3.9. *Tonti Matrix Diagram	15-12
§15.4. *Derivation Using Natural Strains and Stresses	15-12
§15.4.1. *Natural Strains and Stresses	15-12
§15.4.2. *Covariant Node Displacements	15-14
§15.4.3. *The Natural Stiffness Matrix	15-14
§15.5. *The Veubeke Equilibrium Triangle	15-14
§15.5.1. *Kinematic Relations	15-15
§15.5.2. *Stiffness Matrix	15-15
§15.5.3. *Implementation	15-16
§15.5.4. *Spurious Kinematic Modes	15-17
§15.6. *Shear Locking in Turner Triangles	15-18
§15.6.1. *The Inplane Bending Test	15-18
§15.6.2. *Energy Ratios	15-19
§15.6.3. *Convergence as Mesh is Refined	15-19
§15. Notes and Bibliography.	15-20
§15. References	15-21
§15. Exercises	15-23

§15.1. Introduction

This Chapter derives element stiffness equations of three-node triangles constructed with linear displacements for the plane stress problem formulated in Chapter 14. These elements have six displacement degrees of freedom, which are placed at the *connection nodes*. There are two main versions that differ on where the connection nodes are located:

1. The *Turner triangle* has connection nodes located at the corners. Turner triangle
2. The *Veubeke equilibrium triangle* has connection nodes located at the side midpoints. Veubeke equilibrium triangle

The triangle geometry is defined by the corner locations or *geometric nodes* in both cases. Of the two versions, the Turner triangle is by far the most practically important one in solid and structural mechanics.¹ Thus most of the material in this Chapter is devoted to it. It enjoys several important properties:

- (i) It belongs to both the isoparametric and subparametric element families, which are introduced in the next Chapter.
- (ii) It allows closed form derivations for the stiffness matrix and consistent force vector without need for numerical integration.
- (iii) It cannot be improved by the addition of internal degrees of freedom.

Properties (ii) and (iii) are shared by the Veubeke equilibrium triangle. Since this model is rarely used in structural applications it is covered only as advanced material in §15.5.

The Turner triangle is not a good performer for structural stress analysis. It is still used in problems that do not require high accuracy, as well as in non-structural applications such as thermal and electromagnetic analysis. One reason is that triangular meshes are easily generated over arbitrary two-dimensional domains using techniques such as Delaunay triangulation.

§15.2. Background

§15.2.1. Parametric Representation of Functions

The concept of *parametric representation* of functions is crucial in modern FEM. Together with multidimensional numerical integration, it is a key enabling tool for developing elements in two and three space dimensions.² Without these tools the developer would become lost in an algebraic maze as element geometry and shape functions get more complicated. The essentials of parametric representation can be illustrated through a simple example. Consider the following alternative representations of the unit-circle function, $x^2 + y^2 = 1$:

$$(I) \ y = \sqrt{1 - x^2}, \quad (II) \ x = \cos \theta \text{ and } y = \sin \theta. \quad (15.1)$$

The direct representation (I) fits the conventional function notation, i.e., $y = f(x)$. Given a value of x , it returns one or more y . On the other hand, the parametric representation (II) is indirect: both x

¹ The triangle was one of the two plane-stress continuum elements presented by Turner, Clough, Martin and Topp in their 1956 paper [727]. This publication is widely regarded as the start of the present FEM. The derivation was not done, however, with assumed displacements. See **Notes and Bibliography** at the end of this Chapter.

² Numerical integration is not useful for the triangular elements covered here, but essential in the more complicated iso-P models covered in Chapters 16ff.

and y are given in terms of one parameter, the angle θ . Elimination of θ through the trigonometric identity $\cos^2 \theta + \sin^2 \theta = 1$ recovers $x^2 + y^2 = 1$. But there are situations in which working with the parametric form throughout the development is more convenient. Continuum finite elements provide a striking illustration of this point.

§15.2.2. Geometry

The geometry of the 3-node triangle shown in Figure 15.1(a) is specified triangle geometry by the location of its three corner nodes on the $\{x, y\}$ plane. Nodes are labelled 1, 2, 3 while traversing the sides in *counterclockwise* fashion. Their location is defined by their Cartesian coordinates: $\{x_i, y_i\}$ for $i = 1, 2, 3$. The Turner triangle has six degrees of freedom, defined by the six corner displacement components $\{u_{xi}, u_{yi}\}$, for $i = 1, 2, 3$. The interpolation of the internal displacements $\{u_x, u_y\}$ from these six values is studied in §15.3, after triangular coordinates are introduced. The triangle area can be obtained as

$$2A = \det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = (x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) + (x_1 y_2 - x_2 y_1). \quad (15.2)$$

The area given by (15.2) is a *signed* area of triangle quantity. It is positive if the corners are numbered in cyclic counterclockwise order (when looking down from the $+z$ axis), as illustrated in Figure 15.1(b). This convention is followed in the sequel.

§15.2.3. Triangular Coordinates

Points of the triangle may also be located in terms of a triangular coordinates *parametric* coordinate system:

$$\zeta_1, \zeta_2, \zeta_3. \quad (15.3)$$

In the literature these 3 parameters receive an astonishing number of names, as the list collected in Table 15.1 shows. In the sequel the name *triangular coordinates* will be used to emphasize the close association with this particular geometry.

Equations

$$\zeta_i = \text{constant} \quad (15.4)$$

represent a set of straight lines parallel to the side opposite to the i^{th} corner, as depicted in Figure 15.2. The equations of sides 2–3, 3–1 and 1–2 are $\zeta_1 = 0$, $\zeta_2 = 0$ and $\zeta_3 = 0$, respectively. The three corners have coordinates $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$. The three midpoints of the sides have coordinates $(\frac{1}{2}, \frac{1}{2}, 0)$, $(0, \frac{1}{2}, \frac{1}{2})$ and $(\frac{1}{2}, 0, \frac{1}{2})$, the centroid has coordinates $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, and so on. The coordinates are not independent because their sum is unity:

$$\zeta_1 + \zeta_2 + \zeta_3 = 1. \quad (15.5)$$

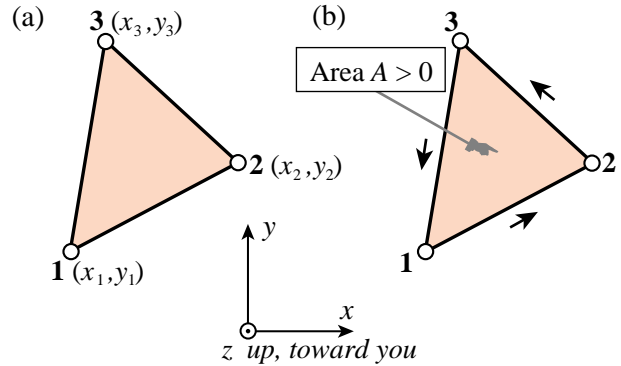
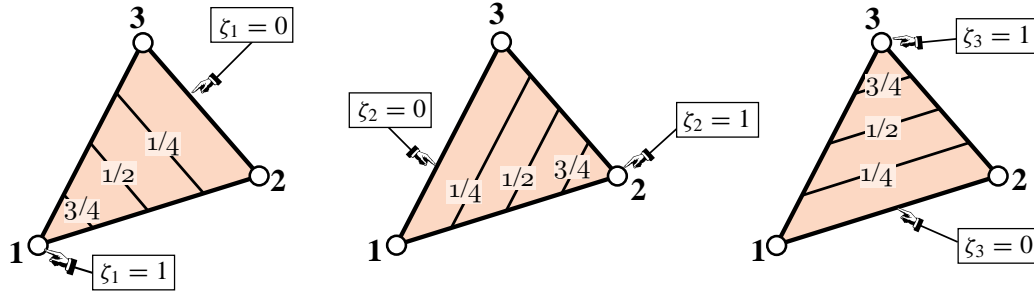


FIGURE 15.1. The three-node, linear-displacement plane stress triangular element: (a) geometry; (b) area and positive boundary traversal.

FIGURE 15.2. Triangular coordinates $\zeta_1, \zeta_2, \zeta_3$.**Table 15.1** Names of element parametric coordinates

Name	Applicable to
natural coordinates	all elements
isoparametric coordinates	isoparametric elements
shape function coordinates	isoparametric elements
barycentric coordinates	simplices (triangles, tetrahedra, ...)
Möbius coordinates	triangles
triangular coordinates	all triangles
area (also written “areal”) coordinates	straight-sided triangles
Triangular coordinates normalized as per $\zeta_1 + \zeta_2 + \zeta_3 = 1$ are often qualified as “homogeneous” in the mathematical literature.	

Remark 15.1. In pre-1970 FEM publications, triangular coordinates were often called *area coordinates*, and occasionally *areal coordinates*. This comes from the following interpretation: area coordinates $\zeta_i = A_{jk}/A$, where A_{jk} is the area subtended by the subtriangle formed by the point P and corners j and k , in which j and k are 3-cyclic permutations of i . Historically this was the way coordinates were defined in 1960s papers. However this relation does not carry over to general isoparametric triangles with curved sides and thus it is not used here.

§15.2.4. Linear Interpolation

Consider a function $f(x, y)$ that varies *linearly* over the linear interpolation over triangle triangle domain. In terms of Cartesian coordinates it may be expressed as

$$f(x, y) = a_0 + a_1x + a_2y, \quad (15.6)$$

where a_0, a_1 and a_2 are coefficients to be determined from three conditions. In finite element work such conditions are often the *nodal values* taken by f at the corners:

$$f_1, f_2, f_3. \quad (15.7)$$

The expression in triangular coordinates makes direct use of those three values:

$$f(\zeta_1, \zeta_2, \zeta_3) = f_1\zeta_1 + f_2\zeta_2 + f_3\zeta_3 = [f_1 \ f_2 \ f_3] \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = [\zeta_1 \ \zeta_2 \ \zeta_3] \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}. \quad (15.8)$$

Formula (15.8) is called a *linear interpolant* for f .

§15.2.5. Coordinate Transformations

Quantities that are closely linked with the element geometry are triangle coordinate transformations best expressed in triangular coordinates. On the other hand, quantities such as displacements, strains and stresses are usually expressed in the Cartesian system $\{x, y\}$. Thus we need transformation equations through which it is possible to pass from one coordinate system to the other.

Cartesian and triangular coordinates are linked by the relation

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}. \quad (15.9)$$

The first equation says that the sum of the three coordinates is one. The next two express x and y linearly as homogeneous forms in the triangular coordinates. These are obtained by applying the linear interpolant (15.8) to the Cartesian coordinates: $x = x_1\zeta_1 + x_2\zeta_2 + x_3\zeta_3$ and $y = y_1\zeta_1 + y_2\zeta_2 + y_3\zeta_3$. Assuming $A \neq 0$, inversion of (15.9) yields

$$\begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} 2A_{23} & y_{23} & x_{32} \\ 2A_{31} & y_{31} & x_{13} \\ 2A_{12} & y_{12} & x_{21} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}. \quad (15.10)$$

Here $x_{jk} = x_j - x_k$, $y_{jk} = y_j - y_k$, A is the triangle area given by (15.2) and A_{jk} denotes the area subtended by corners j, k and the origin of the x - y system. If this origin is taken at the centroid of the triangle, $A_{23} = A_{31} = A_{12} = A/3$.

§15.2.6. Partial Derivatives

From equations (15.9) and (15.10) we immediately obtain the following relations between partial derivatives:

$$\frac{\partial x}{\partial \zeta_i} = x_i, \quad \frac{\partial y}{\partial \zeta_i} = y_i, \quad (15.11)$$

$$2A \frac{\partial \zeta_i}{\partial x} = y_{jk}, \quad 2A \frac{\partial \zeta_i}{\partial y} = x_{kj}. \quad (15.12)$$

In (15.12) j and k denote the 3-cyclic permutations of i . For example, if $i = 2$, then $j = 3$ and $k = 1$. The derivatives of a function $f(\zeta_1, \zeta_2, \zeta_3)$ with respect to x or y follow immediately from (15.12) and application of the chain rule:

$$\begin{bmatrix} \frac{\partial f}{\partial x} = \frac{1}{2A} \left(\frac{\partial f}{\partial \zeta_1} y_{23} + \frac{\partial f}{\partial \zeta_2} y_{31} + \frac{\partial f}{\partial \zeta_3} y_{12} \right) \\ \frac{\partial f}{\partial y} = \frac{1}{2A} \left(\frac{\partial f}{\partial \zeta_1} x_{32} + \frac{\partial f}{\partial \zeta_2} x_{13} + \frac{\partial f}{\partial \zeta_3} x_{21} \right) \end{bmatrix} \quad (15.13)$$

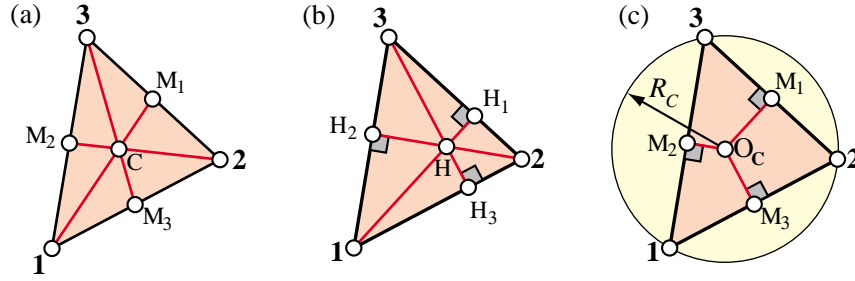


FIGURE 15.3. Interesting points and lines of a triangle.

which in matrix form is

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} y_{23} & y_{31} & y_{12} \\ x_{32} & x_{13} & x_{21} \end{bmatrix} \begin{bmatrix} \frac{\partial f}{\partial \zeta_1} \\ \frac{\partial f}{\partial \zeta_2} \\ \frac{\partial f}{\partial \zeta_3} \end{bmatrix}. \quad (15.14)$$

With these mathematical ingredients in place we are now in a position to handle the derivation of straight-sided triangular elements, and in particular the Turner and Veubeke triangles.

§15.2.7. *Homogeneous Polynomials in Triangular Coordinates

Because ζ_1 , ζ_2 and ζ_3 are not independent, polynomial functions in those variables are not unique. For example $3 - 2\zeta_1 + \zeta_2 - 3\zeta_3$ and $\zeta_1 + 4\zeta_2$ are identical, since they differ by $3 - 3(\zeta_1 + \zeta_2 + \zeta_3) = 0$. To achieve uniqueness it is necessary to write the function as a *homogeneous* polynomial, as in the second form of this example.

To reduce the general linear polynomial $c_{000} + c_{100}\zeta_1 + c_{010}\zeta_2 + c_{001}\zeta_3$ to homogeneous form, subtract $c_{000}(1 - \zeta_1 - \zeta_2 - \zeta_3)$, which is zero, to get $P_1 = (c_{100} - c_{000})\zeta_1 + (c_{010} - c_{000})\zeta_2 + (c_{001} - c_{000})\zeta_3$.

To reduce the general quadratic polynomial $c_{000} + c_{100}\zeta_1 + c_{010}\zeta_2 + c_{001}\zeta_3 + c_{200}\zeta_1^2 + c_{020}\zeta_2^2 + c_{002}\zeta_3^2 + c_{110}\zeta_1\zeta_2 + c_{011}\zeta_2\zeta_3 + c_{101}\zeta_3\zeta_1$ to homogeneous form, subtract $(c_{000} + c_{100}\zeta_1 + c_{010}\zeta_2 + c_{001}\zeta_3)(1 - \zeta_1 - \zeta_2 - \zeta_3)$.

And so on. All polynomial expressions used in this book for triangles are expressed in homogeneous form.

§15.2.8. *Interesting Points and Lines

Some distinguished lines and points of a straight-sided triangle are briefly described here for use in other developments as well as in Exercises. The *triangle medians* are three lines that join the corners to triangle medians the midpoints of the opposite sides, as pictured in Figure 15.3(a). The midpoint opposite corner i is labeled M_i .

The medians $1-M_1$, $2-M_2$ and $3-M_3$ have equations $\zeta_2 = \zeta_3$, $\zeta_3 = \zeta_1$ and $\zeta_1 = \zeta_2$, respectively, in triangular coordinates. They intersect at the centroid C of coordinates $\{\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\}$. Other names for the centroid are *barycenter* and *center of gravity*. If you make a real triangle out of cardboard, you can balance the triangle at this point. It can be shown that the centroid trisects the medians, that is to say, the distance from a corner to the centroid is twice the distance from the centroid to the opposite side of the triangle.

The *altitudes* are three lines that connect each corner with triangle altitudes their projections onto the opposing sides, as depicted in Figure 15.3(b). The projection of corner i is identified H_i , so the altitudes are $1-H_1$, $2-H_2$ and $3-H_3$. Locations H_i are called *altitude feet*. The altitudes intersect at the triangle *orthocenter* H . The

lengths of those segments triangle orthocenter triangle heights triangle altitude feet are the *triangle heights*. The triangular coordinates of H_i and H , as well as the altitude equations, are worked out in an Exercise.

Another interesting point is the center O_C of the circumscribed circle, or circumcircle. This is the unique circle that passes through the three corners, as shown in Figure 15.3(c). It can be geometrically constructed by drawing the normal to each side at the midpoints. Those three lines, called the perpendicular side bisectors, intersect at O_C . A famous theorem by Euler asserts that the centroid, the orthocenter and the circumcircle center fall on a straight line, called the Euler line. Furthermore, C lies between O_C and H , and the distance O_C-H is three times the distance $H-C$.

§15.3. The Turner Triangle

The simplest triangular element for plane stress (and in general, for 2D problems of variational index $m = 1$) is the three-node triangle with *linear shape functions*, with degrees of freedom located at the corners. The shape functions are simply the triangular coordinates. That is, $N_i^e = \zeta_i$ for $i = 1, 2, 3$. When applied to the plane stress problem, this element is called the Turner triangle. For the plane stress problem we select the linear interpolation linear displacement interpolation over triangle (15.8) for the displacement components u_x and u_y at an arbitrary point $P(\zeta_1, \zeta_2, \zeta_3)$:

$$u_x = u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3, \quad u_y = u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3. \quad (15.15)$$

The interpolation is illustrated in Figure 15.4. The two expressions in (15.15) can be combined in a matrix form that befits the expression (14.17) for an arbitrary plane stress element:

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \zeta_1 & 0 & \zeta_2 & 0 & \zeta_3 & 0 \\ 0 & \zeta_1 & 0 & \zeta_2 & 0 & \zeta_3 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{N} \mathbf{u}^e, \quad (15.16)$$

where \mathbf{N} is the matrix of shape functions.

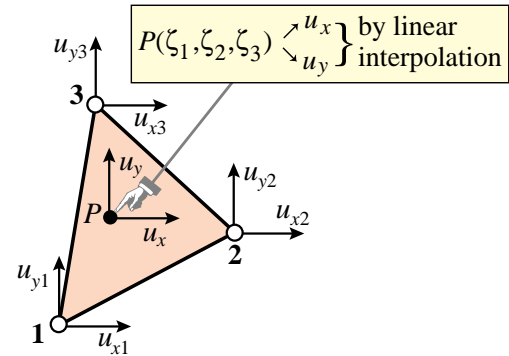


FIGURE 15.4. Displacement interpolation over triangle.

§15.3.1. Strain-Displacement Equations

The strains within the elements are obtained by differentiating the shape functions with respect to x and y . Using (15.14), (15.16) and the strains in Turner triangle general form (14.18) we get

$$\mathbf{e} = \mathbf{D} \mathbf{N} \mathbf{u}^e = \frac{1}{2A} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{B} \mathbf{u}^e, \quad (15.17)$$

in which \mathbf{D} denotes the symbolic strain-to-displacement differentiation operator given in (14.6), and \mathbf{B} is the strain-displacement matrix. Note that the strains are *constant* over the element. This is the origin of the name *constant strain triangle* constant strain triangle (CST) given it in many finite element publications.

§15.3.2. Stress-Strain Equations

The stress field $\boldsymbol{\sigma}$ is related to the strain field by the elastic constitutive equation in (14.5), which is repeated here for convenience:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \mathbf{E} \mathbf{e}, \quad (15.18)$$

where E_{ij} are plane stress elastic moduli. The constitutive matrix \mathbf{E} will be assumed to be constant over the element. Because the strains are constant, so are the stresses.

§15.3.3. The Stiffness Matrix

The element stiffness matrix is given by the general formula (14.23), stiffness matrix of Turner triangle which is repeated here

$$\mathbf{K}^e = \int_{\Omega^e} h \mathbf{B}^T \mathbf{E} \mathbf{B} d\Omega, \quad (15.19)$$

where Ω^e is the triangle domain, and h the plate thickness that appears in the plane stress problem. Since \mathbf{B} and \mathbf{E} are constant, they can be taken out of the integral:

$$\mathbf{K}^e = \mathbf{B}^T \mathbf{E} \mathbf{B} \int_{\Omega^e} h d\Omega \quad (15.20)$$

If h is uniform over the element the remaining integral in (15.20) is simply hA , and we obtain the closed form

$$\mathbf{K}^e = A h \mathbf{B}^T \mathbf{E} \mathbf{B} = \frac{h}{4A} \begin{bmatrix} y_{23} & 0 & x_{32} \\ 0 & x_{32} & y_{23} \\ y_{31} & 0 & x_{13} \\ 0 & x_{13} & y_{31} \\ y_{12} & 0 & x_{21} \\ 0 & x_{21} & y_{12} \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}. \quad (15.21)$$

Exercise 15.1 deals with the case of a linearly varying plate thickness.

§15.3.4. The Consistent Nodal Force Vector

For simplicity we consider here only internal body forces³ defined by the vector field

$$\mathbf{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \quad (15.22)$$

which is specified per unit of volume. The consistent nodal force vector consistent node forces of Turner triangle \mathbf{f}^e is given by the general formula (14.23) of the previous Chapter:

$$\mathbf{f}^e = \int_{\Omega^e} h \mathbf{N}^T \mathbf{b} d\Omega = \int_{\Omega^e} h \begin{bmatrix} \zeta_1 & 0 \\ 0 & \zeta_1 \\ \zeta_2 & 0 \\ 0 & \zeta_2 \\ \zeta_3 & 0 \\ 0 & \zeta_3 \end{bmatrix} \mathbf{b} d\Omega. \quad (15.23)$$

³ For consistent force computations corresponding to distributed boundary loads over a side, see Exercise 15.4.

```

Trig3TurnerMembraneStiffness[ncoor_,Emat_,h_,numer_]:=Module[{
  x1,x2,x3,y1,y2,y3,x21,x32,x13,y12,y23,y31,A,Be,Ke},
  {{x1,y1},{x2,y2},{x3,y3}}=ncoor;
  A=Simplify[(x2*y3-x3*y2+(x3*y1-x1*y3)+(x1*y2-x2*y1))/2];
  {x21,x32,x13,y12,y23,y31}={x2-x1,x3-x2,x1-x3,y1-y2,y2-y3,y3-y1};
  Be={{y23,0,y31,0,y12,0},{0,x32,0,x13,0,x21},
      {x32,y23,x13,y31,x21,y12}}/(2*A);
  If[numer, Be=N[Be]]; Ke=A*h*Transpose[Be].Emat.Be;
  Return[Ke];

```

FIGURE 15.5. Implementation of Turner triangle stiffness matrix calculation as a *Mathematica* module.

The simplest case is when the body force components (15.22) as well as the thickness h are constant over the element. Then we need the integrals

$$\int_{\Omega^e} \zeta_1 d\Omega = \int_{\Omega^e} \zeta_2 d\Omega = \int_{\Omega^e} \zeta_3 d\Omega = \frac{1}{3}A \quad (15.24)$$

which replaced into (15.23) gives

$$\mathbf{f}^e = \frac{Ah}{3} [b_x \quad b_y \quad b_x \quad b_y \quad b_x \quad b_y]^T. \quad (15.25)$$

This agrees with the simple element-by-element force-lumping procedure, which assigns one third of the total force along the $\{x, y\}$ directions: Ahb_x and Ahb_y , to each corner.

Remark 15.2. The integrals (15.24) are particular cases of the general integration formula of monomials in triangular coordinates:

$$\frac{1}{2A} \int_{\Omega^e} \zeta_1^i \zeta_2^j \zeta_3^k d\Omega = \frac{i!j!k!}{(i+j+k+2)!}, \quad i \geq 0, j \geq 0, k \geq 0. \quad (15.26)$$

which can be derived through the Beta function. Here i, j, k are integer exponents. This formula *only holds for triangles with straight sides*, and thus does not apply for higher order elements with curved sides. Formulas (15.24) are obtained by setting exponents $i = 1, j = k = 0$ in (15.26), and permuting $\{i, j, k\}$ cyclically.

§15.3.5. Implementation

The implementation of the Turner triangle in any programming language is very simple. A *Mathematica* module that returns \mathbf{K}^e is shown in Figure 15.5. The module needs only 8 lines of code. It is invoked as

$$\mathbf{Ke} = \text{Trig3TurnerMembraneStiffness}[\text{ncoor}, \text{Emat}, h, \text{numer}]; \quad (15.27)$$

The arguments are

ncoor	Element node coordinates, arranged as a list: $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$.
Emat	A two-dimensional list storing the 3×3 plane stress matrix of elastic moduli as $\{\{E_{11}, E_{12}, E_{13}\}, \{E_{12}, E_{22}, E_{23}\}, \{E_{13}, E_{23}, E_{33}\}\}$.
h	Plate thickness, assumed uniform over the triangle.

```

ncoor={{0,0},{3,1},{2,2}}; Emat=8*{{8,2,0},{2,8,0},{0,0,3}};
Ke=Trig3TurnerMembraneStiffness[ncoor,Emat,1,False];
Print["Ke=",Ke//MatrixForm];
Print["eigs of Ke=",Chop[Eigenvalues[N[Ke]]]];
Show[Graphics[RGBColor[1,0,0]],Graphics[AbsoluteThickness[2]],
Graphics[Polygon[ncoor]],Axes->True];

```

$$Ke = \begin{pmatrix} 11 & 5 & -10 & -2 & -1 & -3 \\ 5 & 11 & 2 & 10 & -7 & -21 \\ -10 & 2 & 44 & -20 & -34 & 18 \\ -2 & 10 & -20 & 44 & 22 & -54 \\ -1 & -7 & -34 & 22 & 35 & -15 \\ -3 & -21 & 18 & -54 & -15 & 75 \end{pmatrix}$$

eigs of Ke = {139.33, 60., 20.6704, 0, 0, 0}

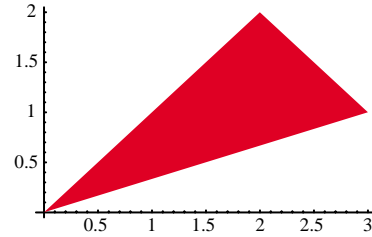


FIGURE 15.6. Test statements to exercise the module of Figure 15.5, and outputs.

numer A logical flag: True to request floating-point computation, else False.

This module is exercised by the statements listed at the top of Figure 15.6, which form a triangle with corner coordinates $\{\{0,0\}, \{3,1\}, \{2,2\}\}$, isotropic material matrix with $E_{11} = E_{22} = 64$, $E_{12} = 16$, $E_{33} = 24$, others zero, (that is, $E = 60$ and $\nu = \frac{1}{4}$) and unit thickness. The results are shown at the bottom of Figure 15.6. The computation of stiffness matrix eigenvalues is always a good programming test, since 3 eigenvalues must be exactly zero and the other 3 real and positive, as explained in Chapter 19. The last test statement draws the triangle (this plot was moved to the right of the numeric output to save space.)

§15.3.6. *Consistency Verification

It remains to check whether the interpolation (15.15) for element displacements consistency verification for Turner triangle meets the completeness and continuity criteria studied in Chapter 19 for finite element trial functions. Such *consistency* conditions are sufficient to insure convergence toward the exact solution of the mathematical model as the mesh is refined.

The variational index for the plane stress problem is $m = 1$. According to the rules stated in §19.3, the trial functions should be 1-complete, C^0 continuous, and C^1 piecewise differentiable.

§15.3.7. *Checking Continuity

Along any triangle side, the variation of u_x and u_y is *linear and uniquely determined by the value at the nodes on that side*. continuity verification for Turner triangle For example, over side 1–2 of an individual triangle, which has equation $\zeta_3 = 0$:

$$\begin{aligned} u_x &= u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3 = u_{x1}\zeta_1 + u_{x2}\zeta_2, \\ u_y &= u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3 = u_{y1}\zeta_1 + u_{y2}\zeta_2. \end{aligned} \quad (15.28)$$

An identical argument holds for that side when it belongs to an adjacent triangle, such as elements (e1) and (e2) shown in Figure 15.7. Since the node values on all elements that meet at a node are the same, u_x and u_y match

Section 15: THREE-NODE PLANE STRESS TRIANGLES

along the side, and the trial function is C^0 interelement continuous. Because the functions are continuous inside the elements, it follows that the continuity requirement is met.

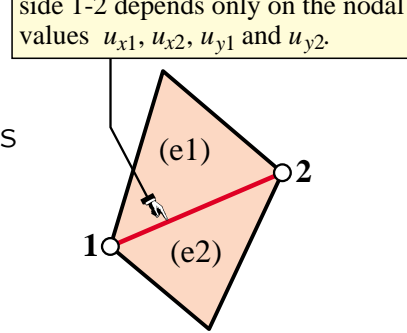


FIGURE 15.7. Interelement continuity check.

§15.3.8. *Checking Completeness

The completeness condition for variational order $m = 1$ requires that the shape functions $N_i = \zeta_i$ be able to represent exactly completeness verification for Turner triangle any linear displacement field:

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y, \quad u_y = \beta_0 + \beta_1 x + \beta_2 y. \quad (15.29)$$

To check this we obtain the nodal values associated with the motion (15.29): $u_{xi} = \alpha_0 + \alpha_1 x_i + \alpha_2 y_i$ and $u_{yi} = \beta_0 + \beta_1 x_i + \beta_2 y_i$ for $i = 1, 2, 3$. Replace these in (15.16) and see if (15.29) is recovered. Here are the detailed calculations for component u_x :

$$\begin{aligned} u_x &= \sum_i u_{xi} \zeta_i = \sum_i (\alpha_0 + \alpha_1 x_i + \alpha_2 y_i) \zeta_i = \sum_i (\alpha_0 \zeta_i + \alpha_1 x_i \zeta_i + \alpha_2 y_i \zeta_i) \\ &= \alpha_0 \sum_i \zeta_i + \alpha_1 \sum_i (x_i \zeta_i) + \alpha_2 \sum_i (y_i \zeta_i) = \alpha_0 + \alpha_1 x + \alpha_2 y. \end{aligned} \quad (15.30)$$

Component u_y can be similarly verified. Consequently (15.16) satisfies the completeness requirement for the plane stress problem — and in general, for any problem of variational index 1. Finally, a piecewise linear trial function is obviously C^1 piecewise differentiable and consequently has finite energy. Thus the two completeness requirements are satisfied.

§15.3.9. *Tonti Matrix Diagram

For further developments covered in more advanced Tonti diagram for Turner triangle courses, it is convenient to split the governing equations of the element. In the case of the Turner triangle they are, omitting element superscripts:

$$\mathbf{e} = \mathbf{B}\mathbf{u}, \quad \boldsymbol{\sigma} = \mathbf{E}\mathbf{e}, \quad \mathbf{f} = \mathbf{A}^T \boldsymbol{\sigma} = V\mathbf{B}^T \boldsymbol{\sigma}. \quad (15.31)$$

Here $V = h_m A$ is the volume of the element, h_m being the mean thickness. The equations (15.31) may be graphically represented with the diagram shown in Figure 15.8. This is a discrete Tonti diagram similar to those of Chapter 6.

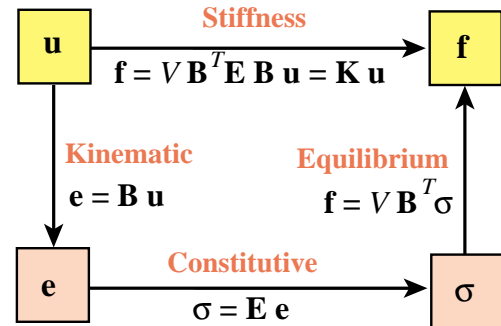


FIGURE 15.8. Tonti matrix diagram for Turner triangle.

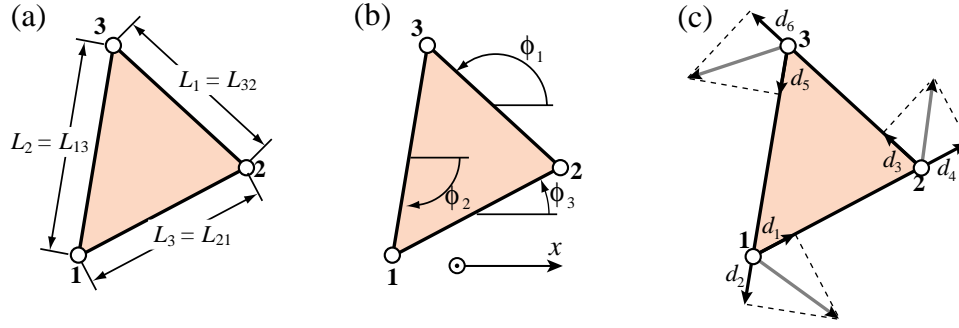


FIGURE 15.10. Additional quantities appearing in natural strain and stress calculations: (a) side lengths, (b) side directions, (c) covariant node displacements.

§15.4. *Derivation Using Natural Strains and Stresses

The foregoing derivation of the Turner triangle uses Cartesian strains and stresses, as well as $\{x, y\}$ displacements. The only intrinsic quantities are the triangle coordinates. This advanced section examines the derivation of the element stiffness matrix through natural strains, natural stresses and covariant displacements.

Although the procedure does not offer obvious shortcuts over the previous derivation, it becomes important in the construction of more complicated high performance elements. It also helps reading recent literature in assumed strain elements.

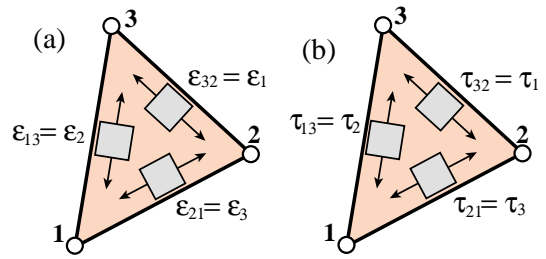


FIGURE 15.9. Geometry-intrinsic fields for the Turner triangle: (a) natural strains ϵ_i , (b) natural stresses τ_i .

§15.4.1. *Natural Strains and Stresses

Natural strains are extensional strains directed parallel to the natural strains triangle sides, as shown in Figure 15.9(a). Natural strains are denoted by $\epsilon_{21} \equiv \epsilon_3$, $\epsilon_{32} \equiv \epsilon_1$, and $\epsilon_{13} \equiv \epsilon_2$.

Similarly, *natural stresses* are normal stresses directed parallel to the triangle sides, natural stresses as shown in Figure 15.9(b). Natural stresses are denoted by $\tau_{21} \equiv \tau_3$, $\tau_{32} \equiv \tau_1$, and $\tau_{13} \equiv \tau_2$.

Because both natural stresses and strains are constant over the triangle, no node value association is needed.

The natural strains can be related to Cartesian strains by the following tensor transformation⁴

$$\epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} c_1^2 & s_1^2 & s_1 c_1 \\ c_2^2 & s_2^2 & s_2 c_2 \\ c_3^2 & s_3^2 & s_3 c_3 \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \mathbf{T}_e^{-1} \mathbf{e}. \quad (15.32)$$

Here $c_1 = x_{32}/L_1$, $s_1 = y_{32}/L_1$, $c_2 = x_{13}/L_2$, $s_2 = y_{13}/L_2$, $c_3 = x_{21}/L_3$, and $s_3 = y_{21}/L_3$, are sines and cosines of the side directions with respect to $\{x, y\}$, as illustrated in Figure 15.10(a,b). The inverse of this relation is

$$\mathbf{e} = \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \frac{1}{4A^2} \begin{bmatrix} y_{31}y_{21}L_1^2 & y_{12}y_{32}L_2^2 & y_{23}y_{13}L_3^2 \\ x_{31}x_{21}L_1^2 & x_{12}x_{32}L_2^2 & x_{23}x_{13}L_3^2 \\ (y_{31}x_{12} + x_{13}y_{21})L_1^2 & (y_{12}x_{23} + x_{21}y_{32})L_2^2 & (y_{23}x_{31} + x_{32}y_{13})L_3^2 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \mathbf{T}_e \epsilon. \quad (15.33)$$

⁴ This is the “straingage rosette” transformation studied in Mechanics of Materials books.

Note that \mathbf{T}_e is constant over the triangle. From the invariance of the strain energy density $\boldsymbol{\sigma}^T \mathbf{e} = \boldsymbol{\tau}^T \boldsymbol{\epsilon}$ it follows that the stresses transform as $\boldsymbol{\tau} = \mathbf{T}_e \boldsymbol{\sigma}$ and $\boldsymbol{\sigma} = \mathbf{T}_e^{-1} \boldsymbol{\tau}$. That strain energy density may be expressed as

$$\mathcal{U} = \frac{1}{2} \mathbf{e}^T \mathbf{E} \mathbf{e} = \frac{1}{2} \boldsymbol{\epsilon}^T \mathbf{E}_n \boldsymbol{\epsilon}, \quad \mathbf{E}_n = \mathbf{T}_e^T \mathbf{E} \mathbf{T}_e. \quad (15.34)$$

Here \mathbf{E}_n is a stress-strain matrix that relates natural stresses to natural strains as $\boldsymbol{\tau} = \mathbf{E}_n \boldsymbol{\epsilon}$. It may be therefore called the natural constitutive matrix.

§15.4.2. *Covariant Node Displacements

Covariant node displacements d_i are directed along the side directions, as shown in Figure 15.10(c), which defines the notation used for them. They are related to the Cartesian node displacements by

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \begin{bmatrix} c_3 & s_3 & 0 & 0 & 0 & 0 \\ c_2 & s_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & s_1 & 0 & 0 \\ 0 & 0 & c_3 & s_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_2 & s_2 \\ 0 & 0 & 0 & 0 & c_1 & s_1 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{T}_d \mathbf{u}. \quad (15.35)$$

The inverse relation is

$$\mathbf{u} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} L_3 y_{31} & L_2 y_{21} & 0 & 0 & 0 & 0 \\ L_3 x_{13} & L_2 x_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & L_1 y_{12} & L_3 y_{32} & 0 & 0 \\ 0 & 0 & L_1 x_{21} & L_3 x_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & L_2 y_{23} & L_1 y_{13} \\ 0 & 0 & 0 & 0 & L_2 x_{32} & L_1 x_{31} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \mathbf{T}_d^{-1} \mathbf{d}. \quad (15.36)$$

The natural strains are evidently given by the relations $\epsilon_1 = (d_6 - d_3)/L_1$, $\epsilon_2 = (d_2 - d_5)/L_2$ and $\epsilon_3 = (d_4 - d_1)/L_3$. Collecting these in matrix form:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1/L_1 & 0 & 0 & 1/L_1 \\ 0 & 1/L_2 & 0 & 0 & -1/L_2 & 0 \\ -1/L_3 & 0 & 0 & 1/L_3 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \mathbf{B}_\epsilon \mathbf{d}. \quad (15.37)$$

§15.4.3. *The Natural Stiffness Matrix

The natural stiffness matrix for constant thickness h is

$$\mathbf{K}_n^e = (Ah) \mathbf{B}_\epsilon^T \mathbf{E}_n \mathbf{B}_\epsilon, \quad \mathbf{E}_n = \mathbf{T}_e^T \mathbf{E} \mathbf{T}_e. \quad (15.38)$$

The Cartesian stiffness matrix is

$$\mathbf{K}^e = \mathbf{T}_d^T \mathbf{K}_n \mathbf{T}_d. \quad (15.39)$$

Comparing with $\mathbf{K}^e = (Ah) \mathbf{B}^T \mathbf{E} \mathbf{B}$ we see that

$$\mathbf{B} = \mathbf{T}_e \mathbf{B}_\epsilon \mathbf{T}_d, \quad \mathbf{B}_\epsilon = \mathbf{T}_e^{-1} \mathbf{B} \mathbf{T}_d^{-1}. \quad (15.40)$$

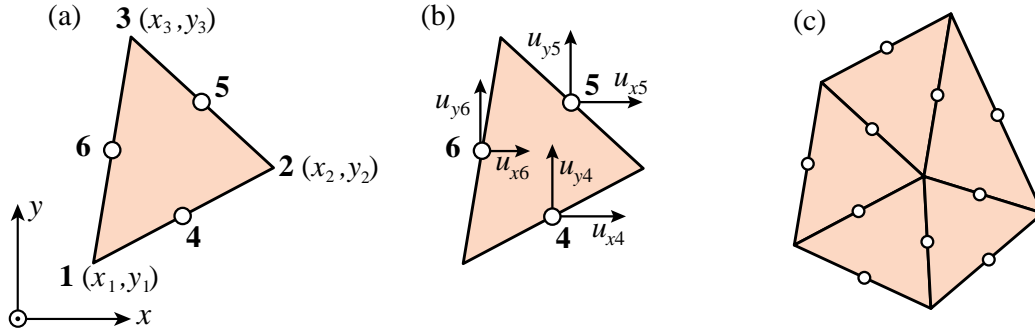


FIGURE 15.11. The Veubeke equilibrium triangle: (a) geometric definition; (b) degree-of-freedom configuration; (c) element patch showing how triangles are connected at the midpoints.

§15.5. *The Veubeke Equilibrium Triangle

The Veubeke equilibrium triangle⁵ differs from the Turner triangle in the degree-of-freedom configuration. Veubeke equilibrium triangle As illustrated in Figure 15.11, those are moved to the midpoints {4, 5, 6} while the corner nodes {1, 2, 3} still define the geometry of the element. In the FEM terminology introduced in Chapter 6, the geometric nodes {1, 2, 3} and the connection nodes {4, 5, 6} no longer coincide. The node displacement vector collects the freedoms shown in Figure 15.11(b):

$$\mathbf{u}^e = [u_{x4} \quad u_{y4} \quad u_{x5} \quad u_{y5} \quad u_{x6} \quad u_{y6}]^T. \quad (15.41)$$

The quickest way to formulate the stiffness matrix of this element is to relate 15.41 to the node displacements of the Turner triangle, renamed for convenience as

$$\mathbf{u}_T^e = [u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3}]^T. \quad (15.42)$$

§15.5.1. *Kinematic Relations

The node freedom vectors 15.41 and 15.42 are easily related since by linear interpolation along the sides one obviously has $u_{x4} = \frac{1}{2}(u_{x1} + u_{x2})$, $u_{y4} = \frac{1}{2}(u_{y1} + u_{y2})$, etc. Expressing those links in matrix form gives

$$\begin{bmatrix} u_{x4} \\ u_{y4} \\ u_{x5} \\ u_{y5} \\ u_{x6} \\ u_{y6} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}, \quad \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 & 1 \\ 1 & 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x4} \\ u_{y4} \\ u_{x5} \\ u_{y5} \\ u_{x6} \\ u_{y6} \end{bmatrix}. \quad (15.43)$$

In compact form: $\mathbf{u}^e = \mathbf{T}_{VT} \mathbf{u}_T^e$ and $\mathbf{u}_T^e = \mathbf{T}_{TV} \mathbf{u}^e$, with $\mathbf{T}_{VT} = \mathbf{T}_{TV}^{-1}$. The shape functions are

$$N_4 = \zeta_1 + \zeta_2 - \zeta_3, \quad N_5 = -\zeta_1 + \zeta_2 + \zeta_3, \quad N_6 = \zeta_1 - \zeta_2 + \zeta_3. \quad (15.44)$$

Renaming the Turner triangle strain-displacement matrix of (15.17) as \mathbf{B}_T , the corresponding matrix that relates $\mathbf{e} = \mathbf{B} \mathbf{u}^e$ in the Veubeke equilibrium triangle becomes

$$\mathbf{B} = \mathbf{B}_T \mathbf{T}_{TV} = \frac{1}{A} \begin{bmatrix} y_{21} & 0 & y_{32} & 0 & y_{13} & 0 \\ 0 & x_{12} & 0 & x_{23} & 0 & x_{31} \\ x_{12} & y_{21} & x_{23} & y_{32} & x_{31} & y_{13} \end{bmatrix} \quad (15.45)$$

⁵ The qualifier *equilibrium* distinguishes this element from others created by Fraeijs de Veubeke, including the 6-node plane stress conforming triangle. See **Notes and Bibliography** for the original derivation from an equilibrium field.


```

Trig3VeubekeMembraneStiffness[ncoor_,Emat_,h_,numer_]:=Module[{
  x1,x2,x3,y1,y2,y3,x12,x23,x31,y21,y32,y13,A,Be,Te,Ke},
  {{x1,y1},{x2,y2},{x3,y3}}=ncoor;
  A=Simplify[(x2*y3-x3*y2+(x3*y1-x1*y3)+(x1*y2-x2*y1))/2];
  {x12,x23,x31,y21,y32,y13}={x1-x2,x2-x3,x3-x1,y2-y1,y3-y2,y1-y3};
  Be={{y21,0,y32,0,y13,0},{0,x12,0,x23,0,x31},
      {x12,y21,x23,y32,x31,y13}}/A;
  If [numer,Be=N[Be]]; Ke=A*h*Transpose[Be].Emat.Be;
  Return[Ke]];

```

FIGURE 15.12. Implementation of Veubeke equilibrium triangle stiffness matrix as a *Mathematica* module.

§15.5.2. *Stiffness Matrix

The element stiffness matrix is given by the general formula (14.23). For constant plate thickness h one obtains the closed form

$$\mathbf{K}^e = A h \mathbf{B}^T \mathbf{E} \mathbf{B} = \frac{h}{A} \begin{bmatrix} y_{21} & 0 & x_{12} \\ 0 & x_{12} & y_{21} \\ y_{32} & 0 & x_{23} \\ 0 & x_{23} & y_{32} \\ y_{13} & 0 & x_{31} \\ 0 & x_{31} & y_{13} \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} y_{21} & 0 & y_{32} & 0 & y_{13} & 0 \\ 0 & x_{12} & 0 & x_{23} & 0 & x_{31} \\ x_{12} & y_{21} & x_{23} & y_{32} & x_{31} & y_{13} \end{bmatrix}. \quad (15.46)$$

stiffness matrix of Veubeke equilibrium triangle The computation of consistent body forces is left as an Exercise.

§15.5.3. *Implementation

The implementation of the Veubeke equilibrium triangle as a *Mathematica* module that returns \mathbf{K}^e is shown in Figure 15.12. It needs only 8 lines of code. It is invoked as

$$\mathbf{K}^e = \text{Trig3VeubekeMembraneStiffness}[\text{ncoor}, \text{Emat}, h, \text{numer}]; \quad (15.47)$$

The arguments have the same meaning as those of the module Trig3TurnerMembraneStiffness described in §15.3.6.

```

ncoor={{0,0},{3,1},{2,2}}; Emat=8*{{8,2,0},{2,8,0},{0,0,3}};
Ke=Trig3VeubekeMembraneStiffness[ncoor,Emat,1,False];
Print["Ke=",Ke//MatrixForm];
Print["eigs of Ke=",Chop[Eigenvalues[N[Ke]]]];

```

$$\mathbf{K}^e = \begin{pmatrix} 140 & -60 & -4 & -28 & -136 & 88 \\ -60 & 300 & -12 & -84 & 72 & -216 \\ -4 & -12 & 44 & 20 & -40 & -8 \\ -28 & -84 & 20 & 44 & 8 & 40 \\ -136 & 72 & -40 & 8 & 176 & -80 \\ 88 & -216 & -8 & 40 & -80 & 176 \end{pmatrix}$$

$$\text{eigs of Ke} = \{557.318, 240., 82.6816, 0, 0, 0\}$$

FIGURE 15.13. Test statements to exercise the module of Figure 15.12, and outputs.

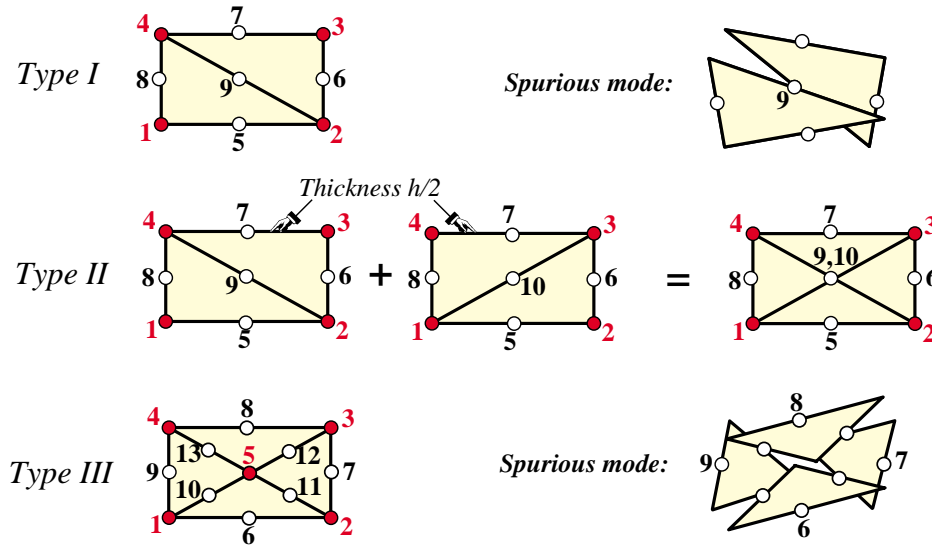


FIGURE 15.14. Three macroelement assemblies fabricated with Veubeke equilibrium triangles to investigate spurious kinematic modes. Red-filled and white-filled circles mark geometric and connection nodes, respectively.

This module is exercised by the statements listed at the top of Figure 15.13, which form a triangle with corner coordinates $\{\{0, 0\}, \{3, 1\}, \{2, 2\}\}$, isotropic material matrix with $E_{11} = E_{22} = 64$, $E_{12} = 16$, $E_{33} = 24$, others zero, and unit thickness. The results are shown at the bottom of Figure 15.13. This is the same triangle used to test module `Trig3TurnerMembraneStiffness` in §15.3.6. Note that the element is rank sufficient.

§15.5.4. *Spurious Kinematic Modes

Although an individual Veubeke equilibrium triangle is rank sufficient, assemblies are prone to the appearance of spurious mechanisms. That is, kinematic modes that produce no strain energy although they are not rigid body modes. These will be illustrated by studying the three macroelements pictured in Figure 15.14. For simplicity the macroelements are of rectangular shape, but the conclusions apply to more general geometries.

Type I macroelement is built with two triangles. It has four geometric nodes: 1–4, five connection nodes: 5–9, and 10 degrees of freedom. The eigenvalue analysis of the assembled stiffness \mathbf{K} is given as an Exercise. It shows that \mathbf{K} has 4 zero eigenvalues. Since there are 3 rigid body modes in 2D, one is spurious. It is easily shown that the spurious mode corresponds to the relative rotation of the two triangles with center node 9 as pivot, as pictured to the right of the macroelement.

Type II macroelement is built with four crisscrossed triangles of thickness $h/2$ as illustrated in the Figure. It has four geometric nodes: 1–4, six connection nodes: 5–10, and 12 degrees of freedom. (Note that although 9 and 10 occupy the same location for this geometry, they should be considered as two separate nodes.) The eigenvalue analysis of the assembled stiffness \mathbf{K} is given as an Exercise. It shows that \mathbf{K} has 3 zero eigenvalues and therefore this macroelement has no spurious modes.

Type III macroelement is of Union-Jack type and is built with 4 triangles. It has five geometric nodes: 1–5, eight connection nodes: 6–13, and 16 degrees of freedom. The eigenvalue analysis of the assembled stiffness \mathbf{K} is given as an Exercise. It shows that \mathbf{K} has 4 zero eigenvalues and consequently one spurious mode. This correspond to the triangles rotating about the midpoints 6–9 as pivots, as pictured to the right of the macroelement.

These examples show that this element, when used in a stiffness code, is prone to *spurious pivot modes* where sides of adjacent triangles rotate relatively from each other about the midpoint connector. This is a consequence

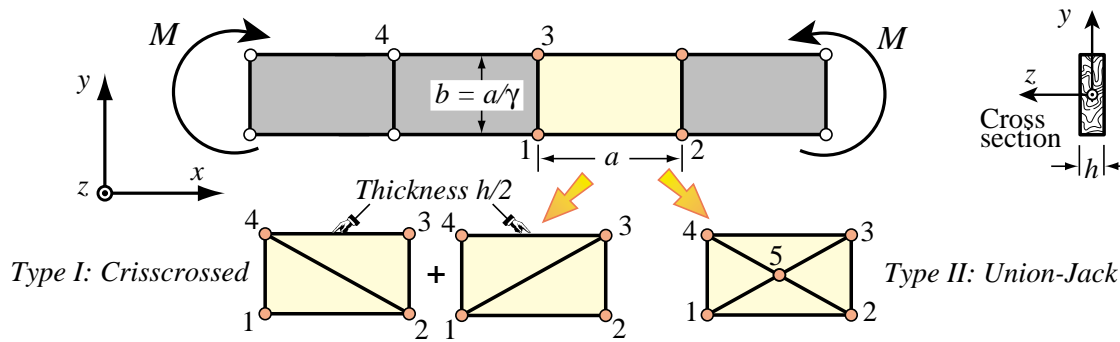


FIGURE 15.15. The bending test with two macroelement types.

of the element being nonconforming: full determination of linearly varying side displacements requires two nodes over that side, and there is only one. Even if a rank sufficiently macroelement mesh unit such as Type II of Figure 15.14 can be constructed, there is no guarantee that spurious pivot modes will not occur when those mesh units are connected. For this reason this element is rarely used in DSM-based structural programs, but acquires importance in applications where flux conservation is important.

§15.6. *Shear Locking in Turner Triangles

A well known deficiency of the 3-node Turner triangle is inability to follow rapidly varying stress fields. This is understandable since stresses within the element, for uniform material properties, are constant. But its 1D counterpart: the 2-node bar element, is nodally exact for displacements under some mild assumptions stated in Chapter 11, and correctly solves loaded-at-joints trusses with one element per member. On the other hand, the triangle can be *arbitrarily way off* under unhappy combinations of loads, geometry and meshing.

What happens in going from 1D to 2D? New effects emerge, notably shear energy and inplane bending. These two can combine to produce *shear locking*: shear locking spurious shear energy elongated triangles can become extraordinarily stiff under inplane bending because of *spurious shear energy*.⁶ The bad news for engineers is that wrong answers caused by locking are *non-conservative*: deflections and stresses can be so grossly underestimated that safety margins are overwhelmed.

To characterize shear locking quantitatively it is convenient to use macroelements in which triangles are combined to form a 4-node rectangle. This simplifies repetition to form regular meshes. The rectangle response under in-plane bending is compared to that of a Bernoulli-Euler beam segment. It is well known that the latter is exact under constant moment. The response ratio of macroelement to beam is a good measure of triangle performance under bending. Such benchmarks are technically called *higher order patch tests*. Test results can be summarized higher order patch test by one number: the *energy ratio*, which gives a scalar measure of relative stiffness.

§15.6.1. *The Inplane Bending Test

The test is defined in Figure 15.15. A Bernoulli-Euler plane beam of thin rectangular cross-section of height b and thickness h is bent under applied end moments M . The beam is fabricated of isotropic material with elastic modulus E and Poisson's ratio ν . Except for possible end effects the exact solution of the beam problem (from both the theory-of-elasticity and beam-theory standpoints) is a constant bending moment $M(x) = M$

⁶ The deterioration can be even more pronounced for its spatial counterpart: the 4-node tetrahedron element, because shear effects are even more important in three dimensions.

along the span. The associated curvature is $\kappa = M/(EI_{zz}) = 12M/(Eb^3h)$. The exact energy taken by a beam segment of length a is $U_{\text{beam}} = \frac{1}{2}M\kappa a = 6M^2 a/(Eb^3h) = \frac{1}{24}Eb^3h\kappa^2 a = \frac{1}{24}Eb^3h\theta_a^2/a$. In the latter $\theta_a = \kappa a$ is the relative rotation of two cross sections separated by a .

To study the bending performance of triangles the beam is modeled with one layer of identical rectangular macroelements dimensioned $a \times b$ and made up of triangles, as illustrated in Figure 15.15. The rectangle aspect ratio is $\gamma = a/b$. All rectangles undergo the same deformations and thus it is enough to study a individual macroelement 1-2-3-4. Two types are considered here:

Crisscrossed (CC). Formed by overlaying triangles 1-2-4, 3-4-2, 2-3-1 and 4-1-2, each with thickness $h/2$. Using 4 triangles instead of 2 makes the macroelement geometrically and physically symmetric since 2 triangles are attached to each corner.

Union-Jack (UJ). Formed by placing a fifth node at the center and dividing the rectangle into 4 triangles: 1-2-5, 2-3-5, 3-4-5, 4-1-5. By construction this element is also geometrically and physically symmetric.

§15.6.2. *Energy Ratios

The assembled macroelement stiffnesses are \mathbf{K}_{CC} and \mathbf{K}_{UJ}^+ , of orders 8×8 and 10×10 , respectively. For the latter the internal node 5 is statically condensed producing an 8×8 stiffness \mathbf{K}_U . To test performance we apply four alternating corner loads as shown in Figure 15.16. The resultant bending moment is $M = Pb$.

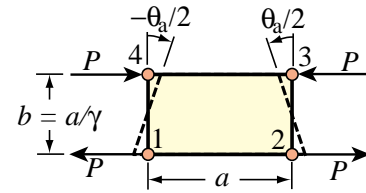


FIGURE 15.16. Bending a macroelement by applying a relative edge rotation.

Although triangles cannot copy curvatures pointwise,⁷ macroelement edges can rotate since constituent triangles can expand or contract. Because of symmetries, the rotations of sides 1-2 and 3-4 are $-\theta_a/2$ and $\theta_a/2$, as illustrated in Figure 15.16. The corresponding corner x displacements are $\pm b\theta_a/4$ whereas the y displacements are zero. Assemble these into a node displacement 8-vector \mathbf{u}_M .

$$\mathbf{u}_M = \frac{1}{4}b\theta_a [-1 \ 0 \ 1 \ 0 \ -1 \ 0 \ 1 \ 0]^T \quad (15.48)$$

The internal energy taken by a macroelement of 8×8 stiffness \mathbf{K}_M under (15.48) is $U_M = \frac{1}{2}\mathbf{u}_M^T \mathbf{K}_M \mathbf{u}_M$, which can be expressed as a function of E , ν , a , b , h and θ_a .⁸

The ratio $r_M = U_M/U_{\text{beam}}$ is called the *energy ratio*. If $r_M > 1$ the macroelement is stiffer than the beam because it takes more energy to bend it to conform to the same edge rotations, and the 2D model is said to be *overstiff*. Results for zero Poisson's ratio, computed with the script of Figure 15.17, are

$$r_{CC} = 3 + \frac{3}{2}\gamma^2, \quad r_{UJ} = \frac{3(1 + \gamma^2)^2}{2 + 4\gamma^2}. \quad (15.49)$$

If for example $\gamma = a/b = 10$, which is an elongated rectangular shape of 10:1 aspect ratio, $r_{CC} = 153$ and the crisscrossed macroelement is 153 times stiffer than the beam. For the Union-Jack configuration $r_{UJ} = 10201/134 = 76.13$; about twice better but still way overstiff. If $\gamma = 1$, $r_{CC} = 4.5$ and $r_{UJ} = 2$: overstiff but not dramatically so. The effect of a nonzero Poisson's ratio is studied in Exercise 15.10.

⁷ That is the reason why they can be so stiff under bending.

⁸ The load P could be recovered via $\mathbf{K}_M \mathbf{u}_M$, but this value is not needed to compute energy ratios.

```

ClearAll[a,b,Em,h, $\gamma$ ];
b=a/ $\gamma$ ; Iz=h*b^3/12; Ubeam=Simplify[(1/2)*Em*Iz* $\theta$ a^2/a];
Emat=Em*{{1,0,0},{0,1,0},{0,0,1/2}};
nc={{-a,-b},{a,-b},{a,b},{-a,b},{0,0}}/2;
enCC={{1,2,4},{3,4,2},{2,3,1},{4,1,3}};
enUJ={{1,2,5},{2,3,5},{3,4,5},{4,1,5}}; r={0,0};
For [m=1,m<=2,m++, mtype={"CC","UJ"}[[m]];
  nF={8,10}[[m]]; K=Table[0,{nF},{nF}]; f=Table[0,{nF}];
  For [e=1,e<=4,e++,
    If [mtype=="CC", enl=enCC[[e]], enl=enUJ[[e]]];
    {n1,n2,n3}=enl; encoor={nc[[n1]],nc[[n2]],nc[[n3]]];
    ht=h; If [mtype=="CC", ht=h/2];
    Ke=Trig3TurnerMembraneStiffness[encoor,Emat,ht,False];
    eft={2*n1-1,2*n1,2*n2-1,2*n2,2*n3-1,2*n3};
    For [i=1,i<=6,i++, For [j=1,j<=6,j++, ii=eft[[i]];
      jj=eft[[j]]; K[[ii,jj]]+=Ke[[i,j]] ];
    ]; KM=K=Simplify[K];
    If [mtype=="UJ",
      {K,f}=Simplify[CondenseLastFreedom[K,f]];
      {KM,f}=Simplify[CondenseLastFreedom[K,f]];
    ];
    Print["KM=",KM//MatrixForm];
    uM={1,0,-1,0,1,0,-1,0}* $\theta$ a*b/4;
    UM=uM.KM.uM/2; rM=Simplify[UM/Ubeam];
    Print["rM=",rM]; r[[m]]=rM;
  ];
Plot[Evaluate[r],{ $\gamma$ ,0,10}];

```

FIGURE 15.17. Script to compute energy ratios for the two macroelements of Figure 15.15.

§15.6.3. *Convergence as Mesh is Refined

Note that if $\gamma = a/b \rightarrow 0$, $r_{CC} \rightarrow 3$ and $r_{UJ} \rightarrow 1.5$. So even if the beam of Figure 15.15 is divided into an infinite number of macroelements along x the solution will not converge. It is necessary to subdivide also along the height. If $2n$ ($n \geq 1$) identical macroelement layers are placed along the beam height while γ is kept fixed, the energy ratio becomes

$$r^{(2n)} = \frac{2^{2n} - 1 + r^{(1)}}{2^{2n}} = 1 + \frac{r^{(1)} - 1}{2^{2n}}, \quad (15.50)$$

where $r^{(1)}$ is the ratio (15.49) for one layer. If $r^{(1)} = 1$, $r^{(2n)} = 1$ for all $n \geq 1$, so bending exactness is maintained as expected. If $n = 1$ (two layers), $r^{(2)} = (3 + r^{(1)})/4$ and if $n = 2$ (four layers), $r^{(4)} = (7 + r^{(1)})/8$.

If $n \rightarrow \infty$, $r^{(2n)} \rightarrow 1$, but convergence can be slow. For example, suppose that $\gamma = 1$ (unit aspect ratio $a = b$) and that $r^{(1)} = r_{CC} = 4.5$. To get within 1% of the exact solution, $1 + 3.5/2^{2n} < 1.01$. This is satisfied if $n \geq 5$, meaning 10 layers of elements along y . If the beam span is 10 times the height, 1000 macroelements or 4000 triangles are needed for this simple problem, which is exactly solvable by one beam element.

The stress accuracy of triangles is examined in Chapter 28.

Notes and Bibliography

As a plane stress structural element, the Turner triangle was first developed in the 1956 paper by Turner et. al. [727]. The target application was modeling of delta wing skin panels. Arbitrary quadrilaterals were formed by assembling triangles as macroelements. Because of its geometric flexibility, the element was soon adopted in aircraft structural analysis codes in the late 1950's. It moved to Civil Engineering applications through the research and teaching at Berkeley of Ray Clough, who gave the method its name in [134].

The derivation method of [727] would look unfamiliar to present FEM practitioners used to the displacement method. It was based on assumed stress modes. More precisely: the element, referred to a local Cartesian system $\{x, y\}$, is put under three constant stress states: σ_{xx} , σ_{yy} and σ_{xy} , collected in array σ . Lumping the

stress field to the nodes gives the node forces: $\mathbf{f} = \mathbf{L}\boldsymbol{\sigma}$. The strain field computed from stresses is $\mathbf{e} = \mathbf{E}^{-1}\boldsymbol{\sigma}$. This is integrated to get a deformation-displacement field, to which 3 rigid-body modes are added as integration constants. Evaluating at the nodes produces $\mathbf{e} = \mathbf{A}\mathbf{u}$, and the stiffness matrix follows on eliminating $\boldsymbol{\sigma}$ and \mathbf{e} : $\mathbf{K} = \mathbf{L}\mathbf{E}\mathbf{A}$. For constant thickness and material properties it happens that $\mathbf{L} = \mathbf{V}\mathbf{A}^T$ and so $\mathbf{K} = \mathbf{V}\mathbf{A}^T\mathbf{E}\mathbf{A}$ happily turned out to be symmetric. This \mathbf{A} is the \mathbf{B} of (15.17) times $2A$, so in the end the stiffness matrix (for constant plate thickness) turns out to be the same as (15.21).

The derivation from assumed displacements evolved later. It is not clear who worked it out first, although it is mentioned in [134,765]. The equivalence of the two forms, through energy principles, had been noted by Gallagher [280]. Early displacement derivations typically started from linear polynomials in Cartesian coordinates. For example Przemieniecki [575] begins with

$$u_x = c_1x + c_2y + c_3, \quad u_y = c_4x + c_5y + c_6. \quad (15.51)$$

Here the c_i play the role of generalized coordinates, which have to be eventually eliminated in favor of node displacements. The same approach is used by Clough in a widely disseminated 1965 article [136]. Even for this simple element the approach is unnecessarily complicated and leads to long hand computations. The elegant derivation in triangular coordinates was popularized by Argyris [27].

The idea of using piecewise linear interpolation over a triangular mesh actually precedes [727] by 13 years. As noted in Chapter 1, it appears in an article by Courant [151], where it is applied to a Poisson's equation modeling St. Venant's torsion. The idea did not influence early work in FEM, however, since as noted above the derivation in [727] was not based on displacement interpolation.

The Veubeke equilibrium triangle appears in [266, p. 170] and is further elaborated in [267, p. 176]. It is constructed there as an *equilibrium element*, that is, the stress field inside the triangle is assumed to be $\sigma_{xx} = \beta_1$, $\sigma_{yy} = \beta_2$ and $\sigma_{xy} = \beta_3$, where $\{\beta_1, \beta_2, \beta_3\}$ are stress parameters. (A field of constant stresses satisfies identically the plane-stress differential equilibrium equations for zero body forces.) Stress parameters can be uniquely expressed in terms of generalized edge loads, which turn out to be virtual-work conjugate to midside displacements.⁹ The direct displacement derivation given here as a "Turner triangle mapping" is new. As previously noted, this element is rarely used in structural mechanics because of the danger of spurious kinematic modes discussed in §15.5.4. It has importance, however, in some non-structural applications.

The completeness check worked out in §15.4.2 is a specialization case of a general proof developed by Irons in the mid 1960s (see [384, §3.9] and references therein) for general isoparametric elements. The check works because the Turner triangle *is* isoparametric.

What are here called triangular coordinates were introduced by Möbius in his 1827 book [478].¹⁰ They are often called barycentric coordinates on account on the interpretation discussed in [153]. Other names are listed in Table 15.1. Triangles possess many fascinating geometric properties studied even before Euclid. An exhaustive development can be found, in the form of solved exercises, in [660].

It is unclear when the monomial integration formula (15.26) was first derived. As an expression for integrands expressed in triangular coordinates it was first stated in [197].

The natural strain derivation of §15.4 is patterned after that developed for the so-called ANDES (Assumed Natural Deviatoric Strain) elements [475]. For the Turner triangle it provides nothing new aside of fancy terminology. Energy ratios of the form used in §15.6 were introduced in [86] as a way to tune up the stiffness of Free-Formulation elements.

⁹ The initial step of assuming stresses exactly mimics that of [727] a decade earlier. What is fundamentally different in Fraeijs de Veubeke's derivation is the use of energy theorems (in this case, PVW) to pass from generalized edge loads to mean edge displacements. The approach is characteristic of FEM Generation 2.

¹⁰ He is better remembered for the "Möbius strip" or "Möbius band," the first one-sided 3D surface in mathematics.

Section 15: THREE-NODE PLANE STRESS TRIANGLES

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 15

The Linear Plane Stress Triangle

EXERCISE 15.1 [A:15] Assume that the 3-node plane stress triangle has *variable* thickness defined over the element by the linear interpolation formula

$$h(\zeta_1, \zeta_2, \zeta_3) = h_1\zeta_1 + h_2\zeta_2 + h_3\zeta_3, \quad (\text{E15.1})$$

where h_1 , h_2 and h_3 are the thicknesses at the corner nodes. Show that the element stiffness matrix is still given by (15.21) but with h replaced by the mean thickness $h_m = (h_1 + h_2 + h_3)/3$. *Hint*: use (15.20) and (15.26).

EXERCISE 15.2 [A:20] The exact integrals of triangle-coordinate monomials over a straight-sided triangle are given by the formula (15.26), where A denotes the area of the triangle, and i , j and k are nonnegative integers. Tabulate the right-hand side for combinations of exponents i , j and k such that $i + j + k \leq 3$, beginning with $i = j = k = 0$. Remember that $0! = 1$. (*Labor-saving hint*: don't bother repeating exponent permutations; for example $i = 2, j = 1, k = 0$ and $i = 1, j = 2, k = 0$ are permutations of the same thing. Hence one needs to tabulate only cases in which $i \geq j \geq k$).

EXERCISE 15.3 [A/C:20] Compute the consistent node force vector \mathbf{f}^e for body loads over a Turner triangle, if the element thickness varies as per (E15.1), $b_x = 0$, and $b_y = b_{y1}\zeta_1 + b_{y2}\zeta_2 + b_{y3}\zeta_3$. Check that for $h_1 = h_2 = h_3 = h$ and $b_{y1} = b_{y2} = b_{y3} = b_y$ you recover (15.25). For area integrals use (15.26). Partial result: $f_{y1} = (A/60)[b_{y1}(6h_1 + 2h_2 + 2h_3) + b_{y2}(2h_1 + 2h_2 + h_3) + b_{y3}(2h_1 + h_2 + 2h_3)]$.

EXERCISE 15.4 [A/C:20] Derive the formula for the consistent force vector \mathbf{f}^e of a Turner triangle of constant thickness h , if side 1–2 ($\zeta_3 = 0, \zeta_2 = 1 - \zeta_1$), is subject to a linearly varying boundary force $\mathbf{q} = h\hat{\mathbf{t}}$ such that

$$\begin{aligned} q_x &= q_{x1}\zeta_1 + q_{x2}\zeta_2 = q_{x1}(1 - \zeta_2) + q_{x2}\zeta_2, \\ q_y &= q_{y1}\zeta_1 + q_{y2}\zeta_2 = q_{y1}(1 - \zeta_2) + q_{y2}\zeta_2. \end{aligned} \quad (\text{E15.2})$$

This “line boundary force” \mathbf{q} has dimension of force per unit of side length.

Procedural Hint. Use the last term of the line integral (14.21), in which $\hat{\mathbf{t}}$ is replaced by \mathbf{q}/h , and show that since the contribution of sides 2-3 and 3-1 to the line integral vanish,

$$W^e = (\mathbf{u}^e)^T \mathbf{f}^e = \int_{\Gamma^e} \mathbf{u}^T \mathbf{q} d\Gamma^e = \int_0^1 \mathbf{u}^T \mathbf{q} L_{21} d\zeta_2, \quad (\text{E15.3})$$

where L_{21} is the length of side 1–2. Replace $u_x(\zeta_2) = u_{x1}(1 - \zeta_2) + u_{x2}\zeta_2$; likewise for u_y , q_x and q_y , integrate and identify with the inner product shown as the second term in (E15.3). Partial result: $f_{x1} = L_{21}(2q_{x1} + q_{x2})/6$, $f_{x3} = f_{y3} = 0$.

Note. The following *Mathematica* script solves this Exercise. If you decide to use it, explain the logic.

```
ClearAll[ux1, uy1, ux2, uy2, ux3, uy3, z2, L12];
ux=ux1*(1-z2)+ux2*z2; uy=uy1*(1-z2)+uy2*z2;
qx=qx1*(1-z2)+qx2*z2; qy=qy1*(1-z2)+qy2*z2;
We=Simplify[L12*Integrate[qx*ux+qy*uy,{z2,0,1}]];
fe=Table[Coefficient[We,{ux1,uy1,ux2,uy2,ux3,uy3}][[i]],{i,1,6}];
fe=Simplify[fe]; Print["fe=",fe];
```

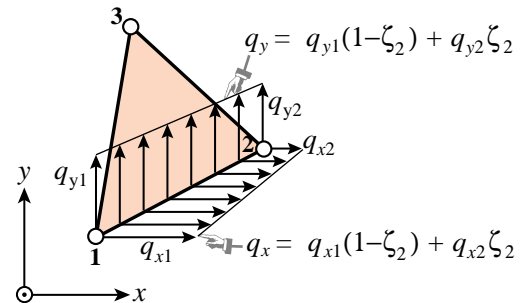


FIGURE E15.1. Line force on triangle side 1–2 for Exercise 15.4.

EXERCISE 15.5 [C+N:15] Compute the entries of \mathbf{K}^e for the following plane stress triangle:

$$x_1 = 0, y_1 = 0, x_2 = 3, y_2 = 1, x_3 = 2, y_3 = 2, \quad (E15.4)$$

$$\mathbf{E} = \begin{bmatrix} 100 & 25 & 0 \\ 25 & 100 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \quad h = 1.$$

This may be done by hand (it is a good exercise in matrix multiplication) or (more quickly) using the script of Figure 15.5. Partial result: $K_{11} = 18.75$, $K_{66} = 118.75$.

EXERCISE 15.6 [A+C:15] Show that the sum of the rows (and columns) 1, 3 and 5 of \mathbf{K}^e as well as the sum of rows (and columns) 2, 4 and 6 must vanish, and explain why. Check it with the foregoing script.

EXERCISE 15.7 [A:10]. Consider two triangles T and T^* , both with positive area. The corner coordinates of T are $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$ and those of T^* are $\{\{x_1^*, y_1^*\}, \{x_2^*, y_2^*\}, \{x_3^*, y_3^*\}\}$. A point P in T has Cartesian coordinates $\{x, y\}$ and triangular coordinates $\{\zeta_1, \zeta_2, \zeta_3\}$. A point P^* in T^* has Cartesian coordinates $\{x^*, y^*\}$ and the same triangular coordinates. Show that $\{x^*, y^*\}$ and $\{x, y\}$ are connected by the affine transformation

$$\begin{bmatrix} 1 \\ x^* \\ y^* \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1^* & x_2^* & x_3^* \\ y_1^* & y_2^* & y_3^* \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} \quad (E15.5)$$

(The indicated inverse exists if T has positive area, as assumed.)

EXERCISE 15.8 [A:15]. Let point P have triangular coordinates $\{\zeta_1^P, \zeta_2^P, \zeta_3^P\}$, as shown in Figure E15.2. Find the distances h_{P1} , h_{P2} and h_{P3} of P to the three triangle sides, and the triangular coordinates of points P_1 , P_2 and P_3 shown in the Figure (P_i is projection on the side opposite to corner i .) Show that $h_{Pi} = \zeta_{Pi} h_i = 2\zeta_{Pi} A/L_{ji}$, for $i = 1, 2, 3$, $j = 2, 3, 1$ and $k = 3, 1, 2$, in which L_{ji} denotes the length of the side that joins corners i and j and h_i is the distance from corner i to the opposite side, as illustrated in Figure E15.2. (Note: the distances $\{h_{P1}, h_{P2}, h_{P3}\}$ are called the *trilinear coordinates* of a point P with respect to the vertices of the triangle. They were introduced by Plücker in 1835. They are essentially scaled versions of the triangular coordinates.)

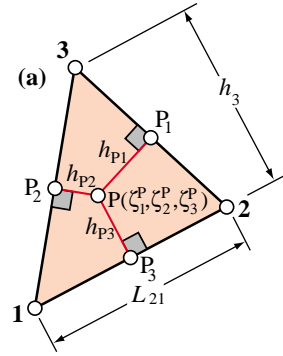


FIGURE E15.2. Distances of arbitrary point P to three triangle sides.

EXERCISE 15.9 [A:10]. Express the distances from the triangle centroid to the 3 sides in term of the triangle area and the side lengths. Answer: $\frac{2}{3}A/L_{21}$, $\frac{2}{3}A/L_{32}$ and $\frac{2}{3}A/L_{13}$, where A is the area of the triangle assumed positive and L_{ji} is the length of side that joins corners i and j , cf. Figure E15.2, Hint: the area of each subtriangle subtended by the centroid and two corners is $\frac{1}{3}A$.

EXERCISE 15.10 [A:20] Find the triangular coordinates of the altitude feet points H_1 , H_2 and H_3 pictured in Figure 15.3. Once these are obtained, find the equations of the altitudes in triangular coordinates, and the coordinates of the orthocenter H . Answer for H_3 : $\zeta_1 = \frac{1}{2} + (L_{13}^2 - L_{32}^2)/(2L_{21}^2)$, where L_{ji} is the length of side that joins corners i and j ; cf. Figure E15.2.

EXERCISE 15.11 [C+D:20] Let $p(\zeta_1, \zeta_2, \zeta_3)$ represent a *polynomial* expression in the natural coordinates. The integral

$$\int_{\Omega^e} p(\zeta_1, \zeta_2, \zeta_3) d\Omega \quad (E15.6)$$

over a straight-sided triangle can be computed symbolically by the following *Mathematica* module:

```

IntegrateOverTriangle[expr_, tcoord_, A_, max_] := Module [{p, i, j, k, z1, z2, z3, c, s = 0},
  p = Expand[expr]; {z1, z2, z3} = tcoord;
  For [i = 0, i <= max, i++, For [j = 0, j <= max, j++, For [k = 0, k <= max, k++,
    c = Coefficient[Coefficient[Coefficient[p, z1, i], z2, j], z3, k];
    s += 2*c*(i!*j!*k!)/((i+j+k+2)!);
  ]]];
Return[Simplify[A*s]] ];

```

This is referenced as `int=IntegrateOverTriangle[p,{z1,z2,z3},A,max]`. Here p is the polynomial to be integrated, $z1$, $z2$ and $z3$ denote the symbols used for the triangular coordinates, A is the triangle area and max the highest exponent appearing in a triangular coordinate. The module name returns the integral. For example, if $p=16+5*b*z2^2+z1^3+z2*z3*(z2+z3)$ the call `int=IntegrateOverTriangle[p,{z1,z2,z3},A,3]` returns `int=A*(97+5*b)/6`. Explain how the module works.

EXERCISE 15.12 [C+D:25] Explain the logic of the script listed in Figure 15.17. Then extend it to account for isotropic material with arbitrary Poisson's ratio ν . Obtain the macroelement energy ratios as functions of γ and ν . Discuss whether the effect of a nonzero ν makes much of a difference if $\gamma \gg 1$.

EXERCISE 15.13 [A/C:25] Verify the conclusions of §15.5.4 as regards rank sufficiency or deficiency of the three Veubeke macroelement assemblies pictured in Figure 15.14. Carry out tests with rectangular macroelements dimensioned $a \times b$, constant thickness h , elastic modulus E and Poisson's ratio 0.

EXERCISE 15.14 [C+D:25] To find whether shear is the guilty party in the poor performance of elongated triangles (as alleged in §15.6) run the script of Figure 15.17 with a zero shear modulus. This can be done by setting `Emat=Em*{{1,0,0},{0,1,0},{0,0,0}}` in the third line. Discuss the result. Can Em be subsequently reduced to a smaller (fictitious) value so that $r \equiv 1$ for all aspect ratios γ ? Is this practical?

```

HomogenizedLinTrigCoorFunction[expr_, {ζ1_, ζ2_, ζ3_}] := Module[
  {f = expr, repζ0, C0}, repζ0 = {ζ1 -> 0, ζ2 -> 0, ζ3 -> 0};
  C0 = Simplify[f /. repζ0]; f = Simplify[f - C0(1 - ζ1 - ζ2 - ζ3)];
  Return[f];

HomogenizedQuadTrigCoorFunction[expr_, {ζ1_, ζ2_, ζ3_}] := Module[
  {f, repζ0, C0, C1, C2, C3}, repζ0 = {ζ1 -> 0, ζ2 -> 0, ζ3 -> 0};
  f = HomogenizedLinTrigCoorFunction[expr, {ζ1, ζ2, ζ3}];
  C1 = Coefficient[f, ζ1] /. repζ0; C2 = Coefficient[f, ζ2] /. repζ0;
  C3 = Coefficient[f, ζ3] /. repζ0; {C1, C2, C3} = Simplify[{C1, C2, C3}];
  f = Simplify[Expand[f - (C1*ζ1 + C2*ζ2 + C3*ζ3)(1 - ζ1 - ζ2 - ζ3)]];
  Return[f];

```

FIGURE E15.3. Two *Mathematica* modules that homogenize linear and quadratic polynomials expressed in triangular coordinates.

EXERCISE 15.15 [C:15] The two *Mathematica* modules listed in Figure E15.3 homogenize linear and quadratic polynomials, respectively, expressed in triangular coordinates. Explain their logic.

Section 15: THREE-NODE PLANE STRESS TRIANGLES

EXERCISE 15.16 [C+D:25] Access the file `Trig3PlaneStress.nb` from the course Web site by clicking on the appropriate link in Chapter 15 Index. This is a *Mathematica* Notebook that does plane stress FEM analysis using the 3-node Turner triangle.

Download the Notebook into your directory. Load into *Mathematica*. Execute the top 7 input cells (which are actually initialization cells) so the necessary modules are compiled. Each cell is preceded by a short comment cell which outlines the purpose of the modules it holds. Notes: (1) the plot-module cell may take a while to run through its tests; be patient; (2) to get rid of unsightly messages and silly beeps about similar names, initialize each cell twice.

After you are satisfied everything works fine, run the cantilever beam problem, which is defined in the last input cell.

After you get a feel of how this code operate, study the source. Prepare a hierarchical diagram of the modules,¹¹ beginning with the main program of the last cell. Note which calls what, and briefly explain the purpose of each module. Return this diagram as answer to the homework. You do not need to talk about the actual run and results; those will be discussed in Part III.

Hint: a hierarchical diagram for `Trig3PlaneStress.nb` begins like

```
Main program in Cell 8 - drives the FEM analysis
  GenerateNodes - generates node coordinates of regular mesh
  GenerateTriangles - generate element node lists of regular mesh
  .....
```

EXERCISE 15.17 [A:10] Consider the Veubeke triangle with 3 midside nodes 4, 5 and 6. Show that three possible shape functions are $1 - 2\zeta_3$, $1 - 2\zeta_1$ and $1 - 2\zeta_2$, respectively. Show that these functions satisfy the interpolation and completeness conditions, but fail the compatibility condition.

¹¹ A hierarchical diagram is a list of modules and their purposes, with indentation to show dependence, similar to the table of contents of a book. For example, if module AAAA calls BBBB and CCCC, and BBBB calld DDDD, the hierarchical diagram may look like:

```
AAAA - purpose of AAAA
  BBBB - purpose of BBBB
    DDDD - purpose of DDDD
  CCCC - purpose of CCCC
```

Hint on Exercise 15.3 (added October 19, 2011)

If doing this Exercise by hand, you should process as follows.

First, multiply \mathbf{N}^T by \mathbf{b} :

$$\mathbf{N}^T \cdot \mathbf{b} = \begin{bmatrix} \zeta_1 & 0 \\ 0 & \zeta_1 \\ \zeta_2 & 0 \\ 0 & \zeta_2 \\ \zeta_3 & 0 \\ 0 & \zeta_3 \end{bmatrix} \begin{bmatrix} 0 \\ b_{y1}\zeta_1 + b_{y2}\zeta_2 + b_{y3}\zeta_3 \end{bmatrix}$$

to get a 6-vector. Entries 1,3 and 5 are zero. Entry 2 is $(b_{y1}\zeta_1 + b_{y2}\zeta_2 + b_{y3}\zeta_3)\zeta_1$, and so on for entries 4 and 6. Next, scale this vector by $h = h_1\zeta_1 + h_2\zeta_2 + h_3\zeta_3$. Entries 1,3 and 5 remain zero, whereas entries 2, 4 and 6 become cubic polynomials in the ζ_i . For example, the second entry is

$$(h_1\zeta_1 + h_2\zeta_2 + h_3\zeta_3)(b_{y1}\zeta_1 + b_{y2}\zeta_2 + b_{y3}\zeta_3)\zeta_1$$

Expand these in term of cubic monomials. For example, the expanded second entry becomes

$$h_1 b_{y1} \zeta^3 + h_1 b_{y2} \zeta_1^2 \zeta_2 + 7 \text{ more terms}$$

Next, collect the ζ_i monomials that appear in entries 2, 4 and 6. The 10 possible monomials are $\zeta_1^3, \zeta_2^3, \zeta_3^3, \zeta_1^2\zeta_2, \zeta_1^2\zeta_3, \zeta_2^2\zeta_1, \zeta_2^2\zeta_3, \zeta_3^2\zeta_1, \zeta_3^2\zeta_2$, and $\zeta_1\zeta_2\zeta_3$. Move all monomial coefficients such as $b_{y1} h_1$, etc., outside the area integral, and apply the formula (15.26) to the monomial integrals. Three cases:

$$\begin{aligned} \int_{\Omega^e} \zeta_1^3 d\Omega &= \int_{\Omega^e} \zeta_2^3 d\Omega = \int_{\Omega^e} \zeta_3^3 d\Omega = \frac{A}{10} \\ \int_{\Omega^e} \zeta_1^2 \zeta_2 d\Omega &= \int_{\Omega^e} \zeta_1^2 \zeta_3 d\Omega = \int_{\Omega^e} \zeta_2^2 \zeta_1 d\Omega = \dots = \frac{A}{20} \\ \int_{\Omega^e} \zeta_1 \zeta_2 \zeta_3 &= \frac{A}{60} \end{aligned}$$

Finally, collect the common factor A , collect the h factors of the b_{yi} as in (E15.2) and you are done. Well, not quite. It is instructive to check your results for the special cases $h_1 = h_2 = h_3 = h$ (constant thickness), and $b_{y1} = b_{y2} = b_{y3} = b_y$ (constant body force). If both the thickness h and the body force b_y are constant, the total force on the element, which is then $b_y h A$, should divide equally in 3 for each node. This would agree with the element-by-element force lumping recipe of Section 7).

If you are good in *Mathematica*, the result can be obtained in milliseconds, but you need to use the module listed under Exercise 15.11.

16

The Isoparametric Representation

TABLE OF CONTENTS

	Page
§16.1. Introduction	16-3
§16.2. Isoparametric Representation	16-3
§16.2.1. Motivation	16-3
§16.2.2. Equalizing Geometry and Displacements	16-4
§16.3. General Isoparametric Formulation	16-5
§16.4. Triangular Elements	16-5
§16.4.1. The Linear Triangle	16-6
§16.4.2. The Quadratic Triangle	16-6
§16.4.3. *The Cubic Triangle	16-6
§16.5. Quadrilateral Elements	16-6
§16.5.1. Quadrilateral Coordinates and Iso-P Mappings	16-6
§16.5.2. The Bilinear Quadrilateral	16-7
§16.5.3. The Biquadratic Quadrilateral	16-7
§16.6. Completeness Properties of Iso-P Elements	16-8
§16.6.1. *Completeness Analysis	16-8
§16.6.2. Completeness Checks	16-9
§16.6.3. *Completeness for Higher Variational Index	16-11
§16.7. Iso-P Elements in One and Three Dimensions	16-11
§16. Notes and Bibliography	16-11
§16. References	16-11
§16. Exercises	16-12

§16.1. Introduction

The procedure used in Chapter 15 to formulate the stiffness equations of the linear triangle can be formally extended to quadrilateral elements as well as higher order triangles. But one quickly encounters technical difficulties:

1. The construction of shape functions that satisfy consistency requirements for higher order elements with curved boundaries becomes increasingly complicated.
2. Integrals that appear in the expressions of the element stiffness matrix and consistent nodal force vector can no longer be evaluated in simple closed form.

These two obstacles can be overcome through the concepts of *isoparametric elements* and *numerical quadrature*, respectively. The combination of these two ideas transformed the field of finite element methods in the late 1960s. Together they support a good portion of what is presently used in production finite element programs.

In the present Chapter the concept of isoparametric representation is introduced for two dimensional elements. This representation is illustrated on specific elements. In the next Chapter these techniques, combined with numerical integration, are applied to quadrilateral elements.

§16.2. Isoparametric Representation

§16.2.1. Motivation

The linear triangle presented in Chapter 15 is an isoparametric element although was not originally derived as such. The two key equations are (15.10), which defines the triangle geometry, and (15.16), which defines the primary variable, in this case the displacement field. These equations are reproduced here for convenience:

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}, \quad (16.1)$$

$$\begin{aligned} u_x &= u_{x1}N_1^e + u_{x2}N_2^e + u_{x3}N_3^e = u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3, \\ u_y &= u_{y1}N_1^e + u_{y2}N_2^e + u_{y3}N_3^e = u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3. \end{aligned} \quad (16.2)$$

The interpretation of these equations is as follows. The triangular coordinates define the element geometry via (16.1). The displacement expansion (16.2) is defined by the shape functions, which are in turn expressed in terms of the triangular coordinates. For the linear triangle, shape functions and triangular coordinates coalesce.

These relations are diagrammed in Figure 16.1. Evidently geometry and displacements are not treated equally. If we proceed to higher order triangular elements while keeping straight sides, only the displacement expansion is refined whereas the geometry definition remains the same.

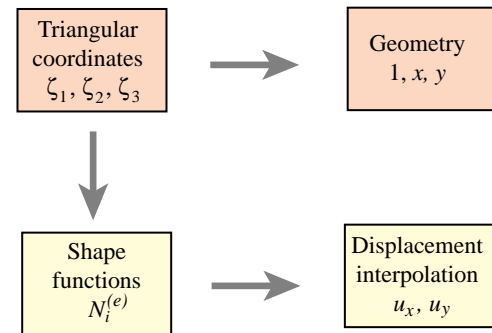


FIGURE 16.1. Superparametric representation of triangular element.

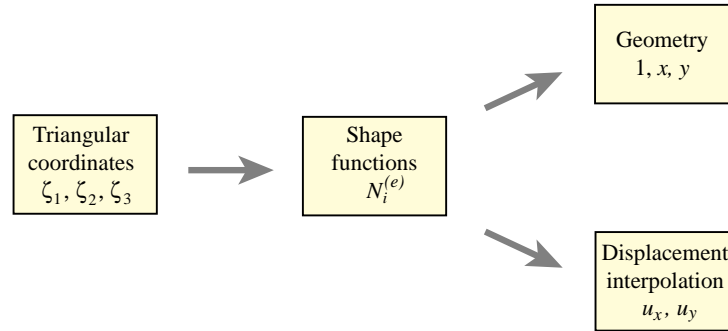


FIGURE 16.2. Isoparametric representation of triangular elements.

Elements built according to the foregoing prescription are called *superparametric*, a term that emphasizes that unequal treatment.

§16.2.2. Equalizing Geometry and Displacements

On first inspection (16.2) and (16.1) do not look alike. Their inherent similarity can be displayed, however, if the second one is rewritten and adjoined to (16.1) to look as follows:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{y3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{y3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \end{bmatrix}. \quad (16.3)$$

This form emphasizes that geometry and displacements are given by the *same* parametric representation, as shown in Figure 16.2.

The key idea is to use the shape functions to represent *both the element geometry and the problem unknowns*, which in structural mechanics are displacements. Hence the name *isoparametric element* (“iso” means equal), often abbreviated to *iso-P element*. This property may be generalized to arbitrary elements by replacing the term “triangular coordinates” by the more general one “natural coordinates.” This generalization is illustrated in Figure 16.3.

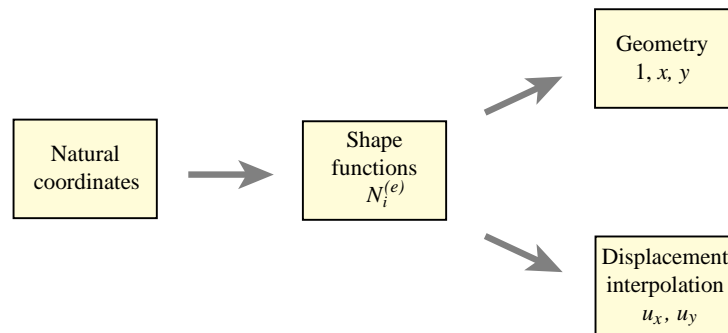


FIGURE 16.3. Isoparametric representation of arbitrary two-dimensional elements: triangles or quadrilaterals. For 3D elements, expand the geometry list to $\{1, x, y, z\}$ and the displacements to $\{u_x, u_y, u_z\}$.

Under this generalization, natural coordinates (triangular coordinates for triangles, quadrilateral coordinates for quadrilaterals) appear as *parameters* that define the shape functions. The shape functions connect the geometry with the displacements.

Remark 16.1. The terms *isoparametric* and *superparametric* were introduced by Irons and coworkers at Swansea in 1966. See **Notes and Bibliography** at the end of this Chapter. There are also *subparametric* elements whose geometry is more refined than the displacement expansion.

§16.3. General Isoparametric Formulation

The generalization of (16.3) to an arbitrary two-dimensional element with n nodes is straightforward. Two set of relations, one for the element geometry and the other for the element displacements, are required. Both sets exhibit the same interpolation in terms of the shape functions.

Geometric relations:

$$\boxed{1 = \sum_{i=1}^n N_i^e, \quad x = \sum_{i=1}^n x_i N_i^e, \quad y = \sum_{i=1}^n y_i N_i^e.} \quad (16.4)$$

Displacement interpolation:

$$\boxed{u_x = \sum_{i=1}^n u_{xi} N_i^e, \quad u_y = \sum_{i=1}^n u_{yi} N_i^e.} \quad (16.5)$$

These two sets of equations may be combined in matrix form as

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ u_{x1} & u_{x2} & \dots & u_{xn} \\ u_{y1} & u_{y2} & \dots & u_{yn} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ \vdots \\ N_n^e \end{bmatrix}. \quad (16.6)$$

The first three scalar equations in (16.6) express the geometry definition, and the last two the displacement expansion. Note that additional rows may be added to this matrix expression if more variables are interpolated by the same shape functions. For example, suppose that the thickness h and a temperature field T are both interpolated from the n node values:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \\ h \\ T \end{bmatrix} = \begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ y_1 & y_2 & \dots & y_n \\ u_{x1} & u_{x2} & \dots & u_{xn} \\ u_{y1} & u_{y2} & \dots & u_{yn} \\ h_1 & h_2 & \dots & h_n \\ T_1 & T_2 & \dots & T_n \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ \vdots \\ N_n^e \end{bmatrix}. \quad (16.7)$$

Note that the column of shape functions does not change.

To illustrate the use of the isoparametric concept, we take a look at specific 2D isoparametric elements that are commonly used in structural and non-structural applications. These are separated into triangles and quadrilaterals because different natural coordinates are used.

§16.4. Triangular Elements

§16.4.1. The Linear Triangle

The three-noded linear triangle, studied in Chapter 15 and pictured in Figure 16.4, may be presented as an isoparametric element:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \end{bmatrix}. \quad (16.8)$$

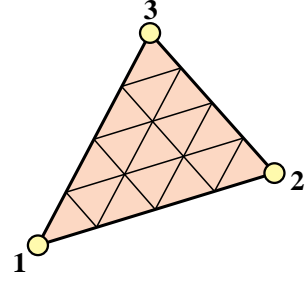


FIGURE 16.4. The 3-node linear triangle.

The shape functions are simply the triangular coordinates:

$$N_1^e = \zeta_1, \quad N_2^e = \zeta_2, \quad N_3^e = \zeta_3. \quad (16.9)$$

The linear triangle is the only triangular element that is both superparametric and isoparametric.

§16.4.2. The Quadratic Triangle

The six node triangle shown in Figure 16.5 is the next complete-polynomial member of the isoparametric triangle family. The isoparametric definition is

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \\ N_4^e \\ N_5^e \\ N_6^e \end{bmatrix} \quad (16.10)$$

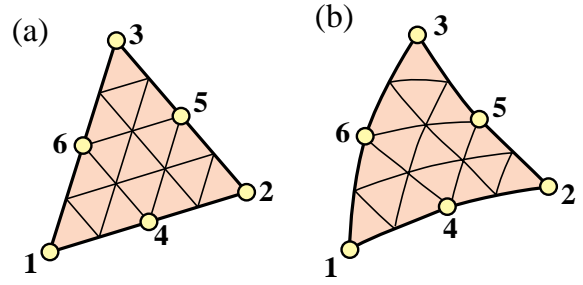


FIGURE 16.5. The 6-node quadratic triangle: (a) the superparametric version, with straight sides and midside nodes at midpoints; (b) the isoparametric version.

The shape functions are

$$\begin{aligned} N_1^e &= \zeta_1(2\zeta_1 - 1), & N_2^e &= \zeta_2(2\zeta_2 - 1), & N_3^e &= \zeta_3(2\zeta_3 - 1), \\ N_4^e &= 4\zeta_1\zeta_2, & N_5^e &= 4\zeta_2\zeta_3, & N_6^e &= 4\zeta_3\zeta_1. \end{aligned} \quad (16.11)$$

The element may have parabolically curved sides defined by the location of the midnodes 4, 5 and 6. The triangular coordinates for a curved triangle are no longer straight lines, but form a curvilinear system as can be observed in Figure 16.5(b).

§16.4.3. *The Cubic Triangle

The cubic triangle has ten nodes. This shape functions of this element are the subject of an Exercise in Chapter 18. The implementation is studied in Chapter 24.

§16.5. Quadrilateral Elements

§16.5.1. Quadrilateral Coordinates and Iso-P Mappings

Before presenting examples of quadrilateral elements, we must introduce the appropriate *natural coordinate system* for that geometry. The natural coordinates for a triangular element are the triangular coordinates ζ_1 , ζ_2 and ζ_3 . The natural coordinates for a quadrilateral element are ξ and η , which are illustrated in Figure 16.6 for both straight sided and curved side quadrilaterals. These are called *quadrilateral coordinates*.

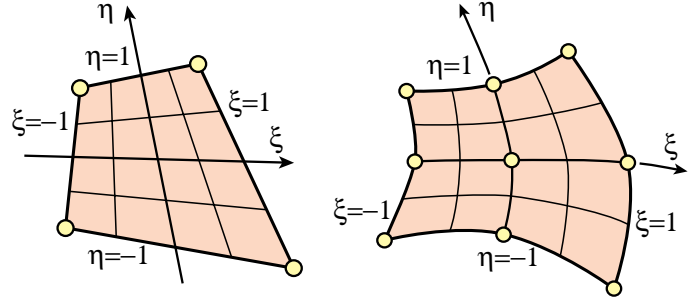


FIGURE 16.6. Quadrilateral coordinates.

These coordinates vary from -1 on one side to $+1$ at the other, taking the value zero over the quadrilateral medians. This particular variation range (instead of taking, say, 0 to 1) was chosen by Irons and coworkers to facilitate use of the standard Gauss integration formulas. Those formulas are discussed in the next Chapter.

Remark 16.2. In some FEM derivations it is convenient to visualize the quadrilateral coordinates plotted as Cartesian coordinates in the $\{\xi, \eta\}$ plane. This is called the *reference plane*. All quadrilateral elements in the reference plane become a square of side 2, called the *reference element*, which extends over $\xi \in [-1, 1]$, $\eta \in [-1, 1]$. The transformation between $\{\xi, \eta\}$ and $\{x, y\}$ dictated by the second and third equations of (16.4), is called the *isoparametric mapping*. A similar version exists for triangles. An important application of this mapping is discussed in §16.6; see Figure 16.9 there.

§16.5.2. The Bilinear Quadrilateral

The four-node quadrilateral shown in Figure 16.7 is the simplest member of the quadrilateral family. It is defined by

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \\ N_4^e \end{bmatrix}. \quad (16.12)$$

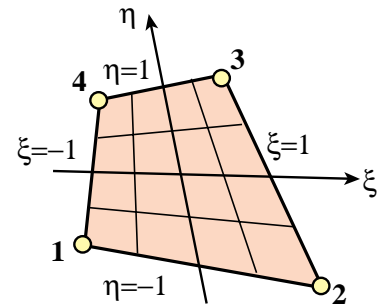


FIGURE 16.7. The 4-node bilinear quadrilateral.

The shape functions are

$$\begin{aligned} N_1^e &= \frac{1}{4}(1 - \xi)(1 - \eta), & N_2^e &= \frac{1}{4}(1 + \xi)(1 - \eta), \\ N_3^e &= \frac{1}{4}(1 + \xi)(1 + \eta), & N_4^e &= \frac{1}{4}(1 - \xi)(1 + \eta). \end{aligned} \quad (16.13)$$

These functions vary *linearly* on quadrilateral coordinate lines $\xi = \text{const}$ and $\eta = \text{const}$, but are not linear polynomials as in the case of the three-node triangle.

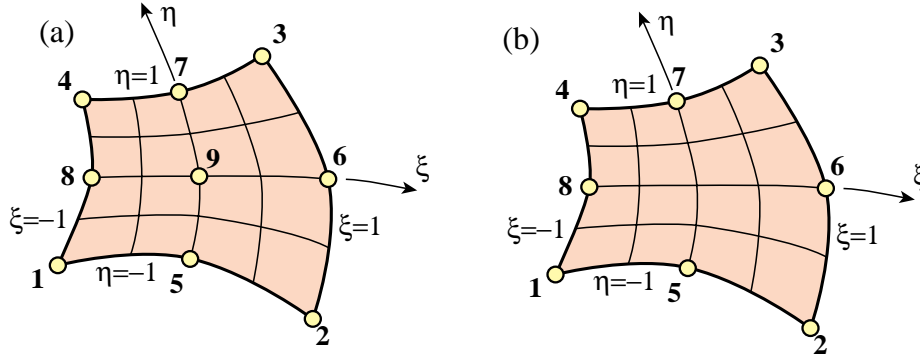


FIGURE 16.8. Two widely used higher order quadrilaterals: (a) the nine-node biquadratic quadrilateral; (b) the eight-node “serendipity” quadrilateral.

§16.5.3. The Biquadratic Quadrilateral

The nine-node quadrilateral shown in Figure 16.8(a) is the next *complete* member of the quadrilateral family. It has eight external nodes and one internal node. It is defined by

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} & u_{x9} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} & u_{y9} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ \vdots \\ N_9^e \end{bmatrix}. \quad (16.14)$$

This element is often referred to as the *Lagrangian quadrilateral* in the FEM literature, a term explained in the **Notes and Bibliography**. Its shape functions are

$$\begin{aligned} N_1^e &= \frac{1}{4}(1-\xi)(1-\eta)\xi\eta, & N_5^e &= -\frac{1}{2}(1-\xi^2)(1-\eta)\eta, \\ N_2^e &= -\frac{1}{4}(1+\xi)(1-\eta)\xi\eta, & N_6^e &= \frac{1}{2}(1+\xi)(1-\eta^2)\xi, & N_9^e &= (1-\xi^2)(1-\eta^2) \\ &\dots & &\dots & \end{aligned} \quad (16.15)$$

These functions vary *quadratically* along the coordinate lines $\xi = \text{const}$ and $\eta = \text{const}$. The shape function associated with the internal node 9 is called a *bubble function* because of its geometric shape, which is pictured in §18.4.2.

Figure 16.8(a) depicts a widely used eight-node variant called the “serendipity” quadrilateral. (A name that originated from circumstances surrounding the element discovery.) The internal node is eliminated by kinematic constraints as worked out in an Exercise of Chapter 18.

§16.6. Completeness Properties of Iso-P Elements

Some general conclusions as regards the range of applications of isoparametric elements can be obtained from a *completeness analysis*. More specifically, whether the general prescription (16.6) that combines (16.4) and (16.5) satisfies the *completeness* criterion of finite element trial expansions. This is one of the conditions for convergence to the analytical solution. The requirement is treated generally in Chapter 19, and is stated here in recipe form.

§16.6.1. *Completeness Analysis

The plane stress problem has variational index $m = 1$. A set of shape functions is complete for this problem if they can represent exactly any *linear* displacement motions such as

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y, \quad u_y = \beta_0 + \beta_1 x + \beta_2 y. \quad (16.16)$$

To carry out the check, evaluate (16.16) at the nodes

$$u_{xi} = \alpha_0 + \alpha_1 x_i + \alpha_2 y_i, \quad u_{yi} = \beta_0 + \beta_1 x_i + \beta_2 y_i, \quad i = 1, \dots, n. \quad (16.17)$$

Insert this into the displacement expansion (16.5) to see whether the linear displacement field (16.16) is recovered. Here are the computations for the displacement component u_x :

$$u_x = \sum_{i=1}^n (\alpha_0 + \alpha_1 x_i + \alpha_2 y_i) N_i^e = \alpha_0 \sum_i N_i^e + \alpha_1 \sum_i x_i N_i^e + \alpha_2 \sum_i y_i N_i^e = \alpha_0 + \alpha_1 x + \alpha_2 y. \quad (16.18)$$

For the last step we have used the geometry definition relations (16.4), reproduced here for convenience:

$$\boxed{1 = \sum_{i=1}^n N_i^e, \quad x = \sum_{i=1}^n x_i N_i^e, \quad y = \sum_{i=1}^n y_i N_i^e.} \quad (16.19)$$

A similar calculation may be made for u_y . It appears that the isoparametric displacement expansion represents (16.18) for *any* element, and consequently meets the completeness requirement for variational order $m = 1$. The derivation carries without essential change to three dimensions.¹

Can you detect a flaw in this conclusion? The fly in the ointment is the last replacement step of (16.18), which assumes that the geometry relations (16.19) *are identically satisfied*. Indeed they are for all the example elements presented in the previous sections. But if the new shape functions are constructed directly by the methods of Chapter 18, *a posteriori* checks of those identities are necessary.

§16.6.2. Completeness Checks

The first check in (16.19) is easy: *the sum of shape functions must be unity*. This is also called the *unit sum condition*. It can be easily verified by hand for simple elements. Here are two examples.

Example 16.1. Check for the linear triangle: directly from the definition of triangular coordinates,

$$N_1^e + N_2^e + N_3^e = \zeta_1 + \zeta_2 + \zeta_3 = 1. \quad (16.20)$$

¹ This derivation is due to B. M. Irons. See for example [218, p. 75]. The property was known since the mid 1960s and contributed substantially to the rapid acceptance of iso-P elements.

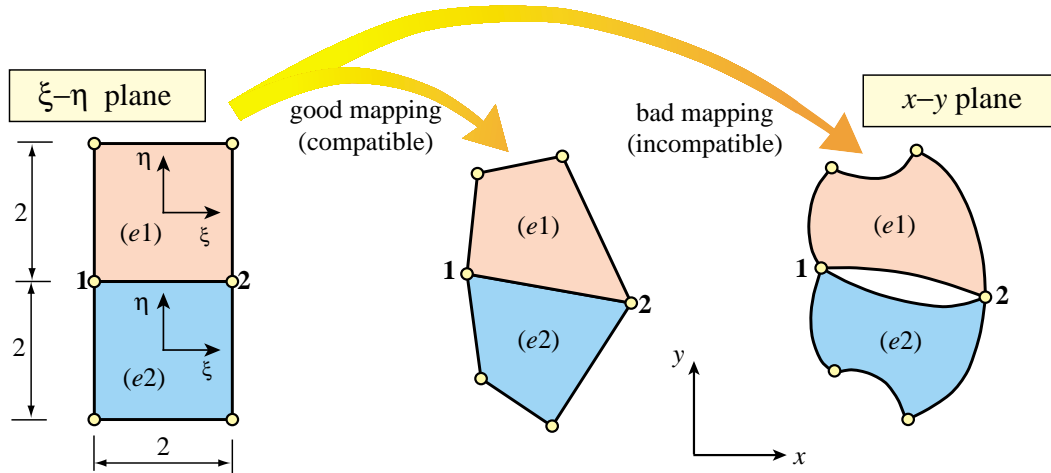


FIGURE 16.9. Good and bad isoparametric mappings of 4-node quadrilateral from the $\{\xi, \eta\}$ reference plane onto the $\{x, y\}$ physical plane.

Example 16.2. Check for the 4-node bilinear quadrilateral:

$$N_1^e + N_2^e + N_3^e + N_4^e = \frac{1}{4}(1 - \xi - \eta + \xi\eta) + \frac{1}{4}(1 + \xi - \eta - \xi\eta) + \frac{1}{4}(1 + \xi + \eta + \xi\eta) + \frac{1}{4}(1 - \xi + \eta - \xi\eta) = 1 \quad (16.21)$$

For more complicated elements see Exercises 16.2 and 16.3.

The other two checks are less obvious. For specificity consider the 4-node bilinear quadrilateral. The geometry definition equations are

$$x = \sum_{i=1}^4 x_i N_i^e(\xi, \eta), \quad y = \sum_{i=1}^4 y_i N_i^e(\xi, \eta). \quad (16.22)$$

Given the corner coordinates, $\{x_i, y_i\}$ and a point $P(x, y)$ one can try to solve for $\{\xi, \eta\}$. This solution requires nontrivial work because it involves two coupled quadratics, but can be done. Reinserting into (16.22) simply gives back x and y , and nothing is gained.²

The correct question to pose is: is the correct geometry of the quadrilateral preserved by the mapping from $\{\xi, \eta\}$ to $\{x, y\}$? In particular, are the sides straight lines? Figure 16.9 illustrate these questions. Two side-two squares: (e1) and (e2), contiguous in the $\{\xi, \eta\}$ reference plane, are mapped to quadrilaterals (e1) and (e2) in the $\{x, y\}$ physical plane through (16.22). The common side 1-2 must remain a straight line to preclude interelement gaps or interpenetration.

We are therefore lead to consider *geometric compatibility* upon mapping. But this is equivalent to the question of *interelement displacement compatibility*, which is stipulated as item (C) in §18.1. The statement “the displacement along a side must be uniquely determined by nodal displacements on that side” translates to “the coordinates of a side must be uniquely determined by nodal coordinates on that side.” Summarizing:

² This tautology is actually a blessing, since finding explicit expressions for the natural coordinates in terms of x and y rapidly becomes impossible for higher order elements. See, for example, the complications that already arise for the bilinear quadrilateral in §23.3.

Unit-sum condition + interelement compatibility \rightarrow completeness.	(16.23)
---	---------

This subdivision of work significantly reduces the labor involved in element testing.

§16.6.3. *Completeness for Higher Variational Index

The completeness conditions for variational index 2 are far more demanding because they involve quadratic motions. No simple isoparametric configurations satisfy those conditions. Consequently isoparametric formulations have limited importance in the finite element analysis of plate and shell bending.

§16.7. Iso-P Elements in One and Three Dimensions

The reader should not think that the concept of isoparametric representation is confined to two-dimensional elements. It applies without conceptual changes to one and three dimensions *as long as the variational index remains one*.³ Three-dimensional solid elements are covered in an advanced course. The use of the isoparametric formulation to construct a 3-node bar element is the topic of Exercises 16.4 through 16.7.

Notes and Bibliography

A detailed presentation of the isoparametric concept, with annotated references to the original 1960 papers may be found in the textbook [218].

This matrix representation for isoparametric elements used here was introduced in [101].

The term *Lagrangian element* in the mathematical FEM literature identifies quadrilateral and hexahedra (brick) elements that include all polynomial terms $\xi^i \eta^j$ (in 2D) or $\xi^i \eta^j \mu^k$ (in 3D) with $i \leq n$, $j \leq n$ and $k \leq n$, as part of the shape function interpolation. Such elements have $(n + 1)^2$ nodes in 2D and $(n + 1)^3$ nodes in 3D, and the interpolation is said to be n -bicomplete. For example, if $n = 2$, the biquadratic quadrilateral with $(2 + 1)^2 = 9$ nodes is Lagrangian and 2-bicomplete. (The qualifier “Lagrangian” in this context refers to Lagrange’s interpolation formula, not to Lagrange multipliers.)

References

Referenced items have been moved to Appendix R

³ A limitation explained in §16.6.3.

Homework Exercises for Chapter 16

The Isoparametric Representation

EXERCISE 16.1 [D:10] What is the physical interpretation of the shape-function unit-sum condition discussed in §16.6? Hint: the element must respond exactly in terms of displacements to rigid-body translations in the x and y directions.

EXERCISE 16.2 [A:15] Check by algebra that the sum of the shape functions for the six-node quadratic triangle (16.11) is exactly one regardless of natural coordinates values. Hint: show that the sum is expressible as $2S_1^2 - S_1$, where $S_1 = \zeta_1 + \zeta_2 + \zeta_3$.

EXERCISE 16.3 [A/C:15] Complete the table of shape functions (16.23) of the nine-node biquadratic quadrilateral. Verify that their sum is exactly one.

EXERCISE 16.4 [A:20] Consider a three-node bar element referred to the natural coordinate ξ . The two end nodes and the midnode are identified as 1, 2 and 3, respectively. The natural coordinates of nodes 1, 2 and 3 are $\xi = -1$, $\xi = 1$ and $\xi = 0$, respectively. The variation of the shape functions $N_1(\xi)$, $N_2(\xi)$ and $N_3(\xi)$ is sketched in Figure E16.1. These functions must be quadratic polynomials in ξ :

$$N_1^e(\xi) = a_0 + a_1\xi + a_2\xi^2, \quad N_2^e(\xi) = b_0 + b_1\xi + b_2\xi^2, \quad N_3^e(\xi) = c_0 + c_1\xi + c_2\xi^2. \quad (\text{E16.1})$$

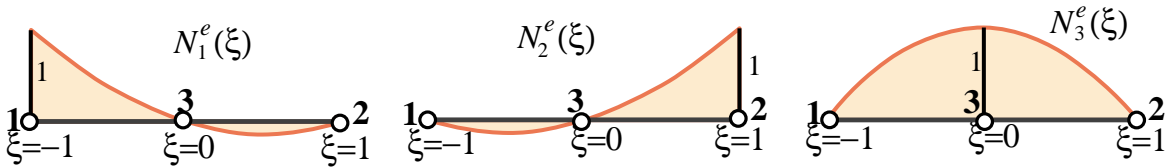


FIGURE E16.1. Isoparametric shape functions for 3-node bar element (sketch). Node 3 has been drawn at the 1–2 midpoint but it may be moved away from it, as in Exercises E16.5 and E16.6.

Determine the coefficients a_0 , through c_2 using the node value conditions depicted in Figure E16.1; for example $N_1^e = 1, 0$ and 0 for $\xi = -1, 0$ and 1 at nodes 1, 3 and 2, respectively. Proceeding this way show that

$$N_1^e(\xi) = -\frac{1}{2}\xi(1 - \xi), \quad N_2^e(\xi) = \frac{1}{2}\xi(1 + \xi), \quad N_3^e(\xi) = 1 - \xi^2. \quad (\text{E16.2})$$

Verify that their sum is identically one.

EXERCISE 16.5

[A/C:15+10+15+5] A 3-node straight bar element is defined by 3 nodes: 1, 2 and 3, with axial coordinates x_1 , x_2 and x_3 , respectively, as illustrated in Figure E16.2. The element has axial rigidity EA and length $\ell = x_2 - x_1$. The axial displacement is $u(x)$. The 3 degrees of freedom are the axial node displacements u_1 , u_2 and u_3 . The isoparametric definition of the element is

$$\begin{bmatrix} 1 \\ x \\ u \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \end{bmatrix}, \quad (\text{E16.3})$$

in which $N_i^e(\xi)$ are the shape functions (E16.2) of the previous Exercise. Node 3 lies between 1 and 2 but is not necessarily at the midpoint $x = \frac{1}{2}\ell$. For convenience define

$$x_1 = 0, \quad x_2 = \ell, \quad x_3 = \left(\frac{1}{2} + \alpha\right)\ell, \quad (\text{E16.4})$$

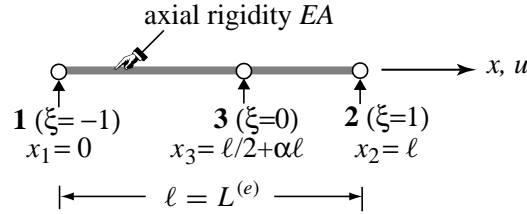


FIGURE E16.2. The 3-node bar element in its local system.

where $-\frac{1}{2} < \alpha < \frac{1}{2}$ characterizes the location of node 3 with respect to the element center. If $\alpha = 0$ node 3 is located at the midpoint between 1 and 2. See Figure E16.2.

- From (E16.4) and the second equation of (E16.3) get the Jacobian $J = dx/d\xi$ in terms of ℓ , α and ξ . Show that: (i) if $-\frac{1}{4} < \alpha < \frac{1}{4}$ then $J > 0$ over the whole element $-1 \leq \xi \leq 1$; (ii) if $\alpha = 0$, $J = \ell/2$ is constant over the element.
- Obtain the 1×3 strain-displacement matrix \mathbf{B} relating $e = du/dx = \mathbf{B} \mathbf{u}^e$, where \mathbf{u}^e is the column 3-vector of node displacements u_1 , u_2 and u_3 . The entries of \mathbf{B} are functions of ℓ , α and ξ . Hint: $\mathbf{B} = d\mathbf{N}/dx = J^{-1}d\mathbf{N}/d\xi$, where $\mathbf{N} = [N_1 \ N_2 \ N_3]$ and J comes from item (a).
- Show that the element stiffness matrix is given by

$$\mathbf{K}^e = \int_0^\ell EA \mathbf{B}^T \mathbf{B} dx = \int_{-1}^1 EA \mathbf{B}^T \mathbf{B} J d\xi. \quad (\text{E16.5})$$

Evaluate the rightmost integral for arbitrary α but constant EA using the 2-point Gauss quadrature rule (E13.7). Specialize the result to $\alpha = 0$, for which you should get $K_{11} = K_{22} = 7EA/(3\ell)$, $K_{33} = 16EA/(3\ell)$, $K_{12} = EA/(3\ell)$ and $K_{13} = K_{23} = -8EA/(3\ell)$, with eigenvalues $\{8EA/\ell, 2EA/\ell, 0\}$. Note: use of a CAS is recommended for this item to save time.

- What is the minimum number of Gauss points needed to integrate \mathbf{K}^e exactly if $\alpha = 0$?

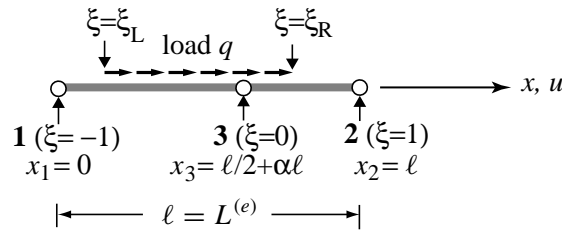
EXERCISE 16.6 [A/C:20] This Exercise is a continuation of the foregoing one, and addresses the question of why \mathbf{K}^e was computed by numerical integration in item (c). Why not use exact integration? The answer is that the exact stiffness for arbitrary α is numerically useless. To see why, try the following script in *Mathematica*:

```

ClearAll[EA,L,alpha,xi]; (* Define J and B={{B1,B2,B3}} here *)
Ke=Simplify[Integrate[EA*Transpose[B].B*J,{xi,-1,1},
  Assumptions->alpha>0&&alpha<1/4&&EA>0&&L>0]];
Print["exact Ke=",Ke//MatrixForm];
Print["exact Ke for alpha=0",Simplify[Ke/.alpha->0]//MatrixForm];
Keseries=Normal[Series[Ke,{alpha,0,2}]];
Print["Ke series about alpha=0:",Keseries//MatrixForm];
Print["Ke for alpha=0",Simplify[Keseries/.alpha->0]//MatrixForm];

```

At the start of this script define J and B with the results of items (a) and (b), respectively. Then run the script. The line `Print["exact Ke for alpha=0",Simplify[Ke/.alpha->0]//MatrixForm]` will trigger error messages. Comment on why the exact stiffness cannot be evaluated directly at $\alpha = 0$ (look at the printed expression before this one). A Taylor series expansion about $\alpha = 0$ circumvents these difficulties but the 2-point Gauss integration rule gives the correct answer without the gyrations.

FIGURE E16.3. The 3-node bar element under a “box” axial load q .

EXERCISE 16.7 [A/C:20] Construct the consistent force vector for the 3-node bar element of the foregoing exercise, if the bar is loaded by a uniform axial force q (given per unit of x length) that extends from $\xi = \xi_L$ through $\xi = \xi_R$, and is zero otherwise. Here $-1 \leq \xi_L < \xi_R \leq 1$. See Figure E16.3. Use

$$\mathbf{f}^e = \int_{-\xi_L}^{\xi_R} q \mathbf{N}^T J d\xi, \quad (\text{E16.6})$$

with the $J = dx/d\xi$ found in Exercise 16.5(a) and analytical integration. The answer is quite complicated and nearly hopeless by hand. Specialize the result to $\alpha = 0$, $\xi_L = -1$ and $\xi_R = 1$.

17

Isoparametric Quadrilaterals

TABLE OF CONTENTS

	Page
§17.1. Introduction	17-3
§17.2. Partial Derivative Computation	17-3
§17.2.1. The Jacobian	17-3
§17.2.2. Shape Function Derivatives	17-4
§17.2.3. Computing the Jacobian Matrix	17-4
§17.2.4. The Strain-Displacement Matrix	17-5
§17.2.5. *A Shape Function Implementation	17-5
§17.3. Numerical Integration by Gauss Rules	17-6
§17.3.1. One Dimensional Rules	17-6
§17.3.2. Implementation of 1D Rules	17-8
§17.3.3. Two Dimensional Rules	17-8
§17.3.4. Implementation of 2D Gauss Rules	17-9
§17.4. The Stiffness Matrix	17-10
§17.5. *Integration Variants	17-11
§17.5.1. *Weighted Integration	17-11
§17.5.2. *Selective Integration	17-12
§17. Notes and Bibliography	17-12
§17. References	17-12
§17. Exercises	17-13

§17.1. Introduction

In this Chapter the isoparametric representation of element geometry and shape functions discussed in the previous Chapter is used to construct *quadrilateral* elements for the plane stress problem. Formulas given in Chapter 14 for the stiffness matrix and consistent load vector of general plane stress elements are of course applicable to these elements. For a practical implementation, however, we must go through more specific steps:

1. Construction of shape functions.
2. Computations of shape function derivatives to form the strain-displacement matrix.
3. Numerical integration over the element by Gauss quadrature rules.

The first topic was dealt in the previous Chapter in recipe form, and is systematically covered in the next one. Assuming the shape functions have been constructed (or readily found in the FEM literature) the second and third items are combined in an algorithm suitable for programming any isoparametric quadrilateral. The implementation of the algorithm in the form of element modules is partly explained in the Exercises of this Chapter, and covered more systematically in Chapter 23.

We shall not deal with isoparametric triangles here to keep the exposition focused. Triangular coordinates, being linked by a constraint, require “special handling” techniques that would complicate and confuse the exposition. Chapter 24 discusses isoparametric triangular elements in detail.

§17.2. Partial Derivative Computation

Partial derivatives of shape functions with respect to the Cartesian coordinates x and y are required for the strain and stress calculations. Because shape functions are not directly functions of x and y but of the natural coordinates ξ and η , the determination of Cartesian partial derivatives is not trivial. The derivative calculation procedure is presented below for the case of an arbitrary isoparametric quadrilateral element with n nodes.

§17.2.1. The Jacobian

In quadrilateral element derivations we will need the Jacobian of two-dimensional transformations that connect the differentials of $\{x, y\}$ to those of $\{\xi, \eta\}$ and vice-versa. Using the chain rule:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \mathbf{J}^T \begin{bmatrix} d\xi \\ d\eta \end{bmatrix}, \quad \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = \mathbf{J}^{-T} \begin{bmatrix} dx \\ dy \end{bmatrix}. \quad (17.1)$$

Here \mathbf{J} denotes the Jacobian matrix of (x, y) with respect to (ξ, η) , whereas \mathbf{J}^{-1} is the Jacobian matrix of (ξ, η) with respect to (x, y) :

$$\mathbf{J} = \frac{\partial(x, y)}{\partial(\xi, \eta)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \quad \mathbf{J}^{-1} = \frac{\partial(\xi, \eta)}{\partial(x, y)} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial y} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix} = \frac{1}{J} \begin{bmatrix} J_{22} & -J_{12} \\ -J_{21} & J_{11} \end{bmatrix}, \quad (17.2)$$

where $J = |\mathbf{J}| = \det(\mathbf{J}) = J_{11}J_{22} - J_{12}J_{21}$. In FEM work \mathbf{J} and \mathbf{J}^{-1} are called simply the *Jacobian* and *inverse Jacobian*, respectively; the fact that it is a matrix being understood. The scalar symbol

J is reserved for the determinant of \mathbf{J} . In one dimension \mathbf{J} and J coalesce. Jacobians play a crucial role in differential geometry. For the general definition of Jacobian matrix of a differential transformation, see Appendix D.

Remark 17.1. Observe that the matrices relating the differentials in (17.1) are the *transposes* of what we call \mathbf{J} and \mathbf{J}^{-1} . The reason is that coordinate differentials transform as contravariant quantities: $dx = (\partial x/\partial \xi) d\xi + (\partial x/\partial \eta) d\eta$, etc. But Jacobians are arranged as in (17.2) because of earlier use in covariant transformations: $\partial \phi/\partial x = (\partial \xi/\partial x)(\partial \phi/\partial \xi) + (\partial \eta/\partial x)(\partial \phi/\partial \eta)$, as in (17.5) below.

The reader is cautioned that notations vary among application areas. As quoted in Appendix D, one author puts it this way: “When one does matrix calculus, one quickly finds that there are two kinds of people in this world: those who think the gradient is a row vector, and those who think it is a column vector.”

Remark 17.2. To show that \mathbf{J} and \mathbf{J}^{-1} are in fact inverses of each other we form their product:

$$\mathbf{J}^{-1}\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial x}{\partial \eta} \frac{\partial \eta}{\partial x} & \frac{\partial y}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial y}{\partial \eta} \frac{\partial \eta}{\partial x} \\ \frac{\partial x}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial x}{\partial \eta} \frac{\partial \eta}{\partial y} & \frac{\partial y}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial y}{\partial \eta} \frac{\partial \eta}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial y}{\partial x} \\ \frac{\partial x}{\partial y} & \frac{\partial y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad (17.3)$$

where we have taken into account that $x = x(\xi, \eta)$, $y = y(\xi, \eta)$ and the fact that x and y are independent coordinates. This proof would collapse, however, if instead of $\{\xi, \eta\}$ we had the triangular coordinates $\{\zeta_1, \zeta_2, \zeta_3\}$ because rectangular matrices have no conventional inverses. This case requires special handling and is covered in Chapter 24.

§17.2.2. Shape Function Derivatives

The shape functions of a quadrilateral element are expressed in terms of the quadrilateral coordinates ξ and η introduced in §16.5.1. The derivatives with respect to x and y are given by the chain rule:

$$\frac{\partial N_i^e}{\partial x} = \frac{\partial N_i^e}{\partial \xi} \frac{\partial \xi}{\partial x} + \frac{\partial N_i^e}{\partial \eta} \frac{\partial \eta}{\partial x}, \quad \frac{\partial N_i^e}{\partial y} = \frac{\partial N_i^e}{\partial \xi} \frac{\partial \xi}{\partial y} + \frac{\partial N_i^e}{\partial \eta} \frac{\partial \eta}{\partial y}. \quad (17.4)$$

This can be put in matrix form as

$$\begin{bmatrix} \frac{\partial N_i^e}{\partial x} \\ \frac{\partial N_i^e}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} \\ \frac{\partial N_i^e}{\partial \eta} \end{bmatrix} = \frac{\partial(\xi, \eta)}{\partial(x, y)} \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} \\ \frac{\partial N_i^e}{\partial \eta} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \frac{\partial N_i^e}{\partial \xi} \\ \frac{\partial N_i^e}{\partial \eta} \end{bmatrix}. \quad (17.5)$$

where \mathbf{J}^{-1} is defined in (17.2). The computation of \mathbf{J} is addressed in the next subsection.

§17.2.3. Computing the Jacobian Matrix

To compute the entries of \mathbf{J} at any quadrilateral location we make use of the last two geometric relations in (16.4), which are repeated here for convenience:

$$x = \sum_{i=1}^n x_i N_i^e, \quad y = \sum_{i=1}^n y_i N_i^e. \quad (17.6)$$

Differentiating with respect to the quadrilateral coordinates,

$$\frac{\partial x}{\partial \xi} = \sum_{i=1}^n x_i \frac{\partial N_i^e}{\partial \xi}, \quad \frac{\partial y}{\partial \xi} = \sum_{i=1}^n y_i \frac{\partial N_i^e}{\partial \xi}, \quad \frac{\partial x}{\partial \eta} = \sum_{i=1}^n x_i \frac{\partial N_i^e}{\partial \eta}, \quad \frac{\partial y}{\partial \eta} = \sum_{i=1}^n y_i \frac{\partial N_i^e}{\partial \eta}. \quad (17.7)$$

because the x_i and y_i do not depend on ξ and η . In matrix form:

$$\mathbf{J} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \mathbf{P}\mathbf{X} = \begin{bmatrix} \frac{\partial N_1^e}{\partial \xi} & \frac{\partial N_2^e}{\partial \xi} & \cdots & \frac{\partial N_n^e}{\partial \xi} \\ \frac{\partial N_1^e}{\partial \eta} & \frac{\partial N_2^e}{\partial \eta} & \cdots & \frac{\partial N_n^e}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}. \quad (17.8)$$

Given a quadrilateral point of coordinates ξ, η we calculate the entries of \mathbf{J} using (17.8). The inverse Jacobian \mathbf{J}^{-1} is then obtained by numerically inverting this 2×2 matrix.

Remark 17.3. The symbolic inversion of \mathbf{J} for arbitrary ξ, η in general leads to extremely complicated expressions unless the element has a particularly simple geometry, (for example rectangles as in Exercises 17.1–17.3). This was one of the difficulties that motivated the use of Gaussian numerical quadrature, as discussed in §17.3 below.

§17.2.4. The Strain-Displacement Matrix

The strain-displacement matrix \mathbf{B} that appears in the computation of the element stiffness matrix is given by the general expression (14.18), which is reproduced here for convenience:

$$\mathbf{e} = \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1^e}{\partial x} & 0 & \frac{\partial N_2^e}{\partial x} & 0 & \cdots & \frac{\partial N_n^e}{\partial x} & 0 \\ 0 & \frac{\partial N_1^e}{\partial y} & 0 & \frac{\partial N_2^e}{\partial y} & \cdots & 0 & \frac{\partial N_n^e}{\partial y} \\ \frac{\partial N_1^e}{\partial y} & \frac{\partial N_1^e}{\partial x} & \frac{\partial N_2^e}{\partial y} & \frac{\partial N_2^e}{\partial x} & \cdots & \frac{\partial N_n^e}{\partial y} & \frac{\partial N_n^e}{\partial x} \end{bmatrix} \mathbf{u}^e = \mathbf{B}\mathbf{u}^e. \quad (17.9)$$

The nonzero entries of \mathbf{B} are partials of the shape functions with respect to x and y . The calculation of those partials is done by computing \mathbf{J} via (17.8), inverting and using the chain rule (17.5).

```

Quad4IsoPShapeFunDer[ncoor_,qcoor_]:= Module[
  {Nf,dNx,dNy,dNxi,dNxi,i,J11,J12,J21,J22,Jdet,xi,eta,x,y},
  {xi,eta}=qcoor;
  Nf={ (1-xi)*(1-eta), (1+xi)*(1-eta), (1+xi)*(1+eta), (1-xi)*(1+eta) }/4;
  dNxi={ -(1-eta), (1-eta), (1+eta), -(1+eta) }/4;
  dNxi={ -(1-xi), -(1+xi), (1+xi), (1-xi) }/4;
  x=Table[ncoor[[i,1]],{i,4}]; y=Table[ncoor[[i,2]],{i,4}];
  J11=dNxi.x; J12=dNxi.y; J21=dNxi.x; J22=dNxi.y;
  Jdet=Simplify[J11*J22-J12*J21];
  dNx= ( J22*dNxi-J12*dNxi)/Jdet; dNx=Simplify[dNx];
  dNy= (-J21*dNxi+J11*dNxi)/Jdet; dNy=Simplify[dNy];
  Return[ {Nf,dNx,dNy,Jdet} ]
];

```

FIGURE 17.1. A shape function module for the 4-node bilinear quadrilateral.

§17.2.5. *A Shape Function Implementation

To make the foregoing discussion more specific, Figure 17.1 shows the *shape function module* for the 4-node bilinear quadrilateral. This is a code fragment that returns the value of the shape functions and their $\{x, y\}$ derivatives at a given point of quadrilateral coordinates $\{\xi, \eta\}$. The module is invoked by saying

$$\{Nf, Nfx, Nfy, Jdet\} = \text{Quad4IsoPShapeFunDer}[ncoor, qcoor] \quad (17.10)$$

where the arguments are

ncoor Quadrilateral node coordinates arranged in two-dimensional list form:
 $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}\}$.

qcoor Quadrilateral coordinates $\{\xi, \eta\}$ of the point.

The module returns:

Nf Value of shape functions, arranged as list $\{Nf_1, Nf_2, Nf_3, Nf_4\}$.

Nfx Value of x -derivatives of shape functions, arranged as list $\{Nfx_1, Nfx_2, Nfx_3, Nfx_4\}$.

Nfy Value of y -derivatives of shape functions, arranged as list $\{Nfy_1, Nfy_2, Nfy_3, Nfy_4\}$.

Jdet Jacobian determinant.

Example 17.1. Consider a 4-node bilinear quadrilateral shaped as an axis-aligned 2:1 rectangle, with $2a$ and a as the x and y dimensions, respectively. The node coordinate array is $ncoor = \{\{0, 0\}, \{2a, 0\}, \{2a, a\}, \{0, a\}\}$. The shape functions and their $\{x, y\}$ derivatives are to be evaluated at the rectangle center $\xi = \eta = 0$. The appropriate call is

$$\{Nf, Nfx, Nfy, Jdet\} = \text{Quad4IsoPShapeFunDer}[ncoor, \{0, 0\}]$$

This returns $Nf = \{1/8, 1/8, 3/8, 3/8\}$, $Nfx = \{-1/(8a), 1/(8a), 3/(8a), -3/(8a)\}$, $Nfy = \{-1/(2a), -1/(2a), 1/(2a), 1/(2a)\}$ and $Jdet = a^2/2$.

§17.3. Numerical Integration by Gauss Rules

Numerical integration is essential for practical evaluation of integrals over isoparametric element domains. The standard practice has been to use *Gauss integration* because such rules use a *minimal number of sample points to achieve a desired level of accuracy*. This economy is important for efficient element calculations, since a *matrix product* is evaluated at each sample point. The fact that the location of the sample points in Gauss rules is usually given by non-rational numbers is of no concern in digital computation.

§17.3.1. One Dimensional Rules

The classical Gauss integration rules are defined by

$$\int_{-1}^1 F(\xi) d\xi \approx \sum_{i=1}^p w_i F(\xi_i). \quad (17.11)$$

Here $p \geq 1$ is the number of Gauss integration points (also known as sample points), w_i are the integration weights, and ξ_i are sample-point abscissae in the interval $[-1, 1]$. The use of the canonical interval $[-1, 1]$ is no restriction, because an integral over another range, say from a to b , can be

Table 17.1 - One-Dimensional Gauss Rules with 1 through 5 Sample Points

Points	Rule
1	$\int_{-1}^1 F(\xi) d\xi \approx 2F(0)$
2	$\int_{-1}^1 F(\xi) d\xi \approx F(-1/\sqrt{3}) + F(1/\sqrt{3})$
3	$\int_{-1}^1 F(\xi) d\xi \approx \frac{5}{9}F(-\sqrt{3/5}) + \frac{8}{9}F(0) + \frac{5}{9}F(\sqrt{3/5})$
4	$\int_{-1}^1 F(\xi) d\xi \approx w_{14}F(\xi_{14}) + w_{24}F(\xi_{24}) + w_{34}F(\xi_{34}) + w_{44}F(\xi_{44})$
5	$\int_{-1}^1 F(\xi) d\xi \approx w_{15}F(\xi_{15}) + w_{25}F(\xi_{25}) + w_{35}F(\xi_{35}) + w_{45}F(\xi_{45}) + w_{55}F(\xi_{55})$

For the 4-point rule, $\xi_{34} = -\xi_{24} = \sqrt{(3 - 2\sqrt{6/5})/7}$, $\xi_{44} = -\xi_{14} = \sqrt{(3 + 2\sqrt{6/5})/7}$, $w_{14} = w_{44} = \frac{1}{2} - \frac{1}{6}\sqrt{5/6}$, and $w_{24} = w_{34} = \frac{1}{2} + \frac{1}{6}\sqrt{5/6}$.

For the 5-point rule, $\xi_{55} = -\xi_{15} = \frac{1}{3}\sqrt{5 + 2\sqrt{10/7}}$, $\xi_{45} = -\xi_{35} = \frac{1}{3}\sqrt{5 - 2\sqrt{10/7}}$, $\xi_{35} = 0$, $w_{15} = w_{55} = (322 - 13\sqrt{70})/900$, $w_{25} = w_{45} = (322 + 13\sqrt{70})/900$ and $w_{35} = 512/900$.

transformed to $[-1, +1]$ via a simple linear transformation of the independent variable, as shown in the Remark below.

The first five one-dimensional Gauss rules, illustrated in Figure 17.2, are listed in Table 17.1. These integrate exactly polynomials in ξ of orders up to 1, 3, 5, 7 and 9, respectively. In general a one-dimensional Gauss rule with p points integrates exactly polynomials of order up to $2p - 1$. This is called the *degree* of the formula.

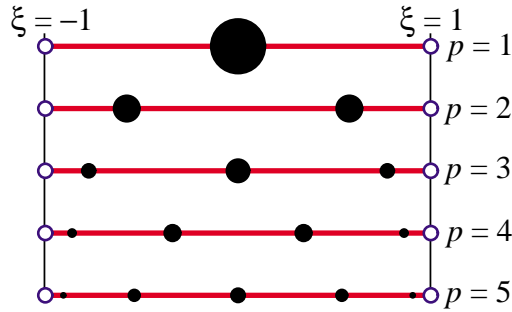


FIGURE 17.2. The first five one-dimensional Gauss rules $p = 1, 2, 3, 4, 5$ depicted over the line segment $\xi \in [-1, +1]$. Sample point locations are marked with black circles. The radii of those circles are proportional to the integration weights.

Remark 17.4. A more general integral, such as $F(x)$ over $[a, b]$ in which $\ell = b - a > 0$, is transformed to the canonical interval $[-1, 1]$ through the mapping $x = \frac{1}{2}a(1 - \xi) + \frac{1}{2}b(1 + \xi) = \frac{1}{2}(a + b) + \frac{1}{2}\ell\xi$, or $\xi = (2/\ell)(x - \frac{1}{2}(a + b))$. The Jacobian of this mapping is $J = dx/d\xi = 1/2\ell$. Thus

$$\int_a^b F(x) dx = \int_{-1}^1 F(\xi) J d\xi = \int_{-1}^1 F(\xi) \frac{1}{2}\ell d\xi. \quad (17.12)$$

```

LineGaussRuleInfo[{rule_,number_},point_]:= Module[
{g2={-1,1}/Sqrt[3],w3={5/9,8/9,5/9},
g3={-Sqrt[3/5],0,Sqrt[3/5]},
w4={(1/2)-Sqrt[5/6]/6, (1/2)+Sqrt[5/6]/6,
(1/2)+Sqrt[5/6]/6, (1/2)-Sqrt[5/6]/6},
g4={-Sqrt[(3+2*Sqrt[6/5])/7],-Sqrt[(3-2*Sqrt[6/5])/7],
Sqrt[(3-2*Sqrt[6/5])/7], Sqrt[(3+2*Sqrt[6/5])/7]},
g5={-Sqrt[5+2*Sqrt[10/7]],-Sqrt[5-2*Sqrt[10/7]],0,
Sqrt[5-2*Sqrt[10/7]], Sqrt[5+2*Sqrt[10/7]]/3,
w5={322-13*Sqrt[70],322+13*Sqrt[70],512,
322+13*Sqrt[70],322-13*Sqrt[70]}/900,
i=point,p=rule,info={Null,Null},0}},
If [p==1, info={0,2}];
If [p==2, info={g2[[i]],1}];
If [p==3, info={g3[[i]],w3[[i]]}];
If [p==4, info={g4[[i]],w4[[i]]}];
If [p==5, info={g5[[i]],w5[[i]]}];
If [number, Return[N[info]], Return[Simplify[info]]];
1;

```

FIGURE 17.3. A *Mathematica* module that returns the first five one-dimensional Gauss rules.

Remark 17.5. Higher order Gauss rules are tabulated in standard manuals for numerical computation. For example, the widely used Handbook of Mathematical Functions [1] lists (in Table 25.4) rules with up to 96 points. For $p > 6$ the abscissas and weights of sample points are not expressible as rational numbers or radicals, and can only be given as floating-point numbers.

§17.3.2. Implementation of 1D Rules

The *Mathematica* module shown in Figure 17.3 returns either exact or floating-point information for the first five unidimensional Gauss rules. To get information for the i^{th} point of the p^{th} rule, in which $1 \leq i \leq p$ and $p = 1, 2, 3, 4, 5$, call the module as

$$\{x_{ii}, w_i\} = \text{LineGaussRuleInfo}[\{p, \text{number}\}, i] \quad (17.13)$$

Logical flag *number* is *True* to get numerical (floating-point) information, or *False* to get exact information. The module returns the sample point abscissa ξ_i in x_{ii} and the weight w_i in w_i . If p is not in the implemented range 1 through 5, the module returns $\{\text{Null}, 0\}$.

Example 17.2. $\{x_i, w\} = \text{LineGaussRuleInfo}[\{3, \text{False}\}, 2]$ returns $x_i = 0$ and $w = 8/9$, whereas $\{x_i, w\} = \text{LineGaussRuleInfo}[\{3, \text{True}\}, 2]$ returns (to 16 places) $x_i = 0.$ and $w = 0.8888888888888889$.

§17.3.3. Two Dimensional Rules

The simplest two-dimensional Gauss rules are called *product rules*. They are obtained by applying the one-dimensional rules to each independent variable in turn. To apply these rules we must first reduce the integrand to the canonical form:

$$\int_{-1}^1 \int_{-1}^1 F(\xi, \eta) d\xi d\eta = \int_{-1}^1 d\eta \int_{-1}^1 F(\xi, \eta) d\xi. \quad (17.14)$$

Once this is done we can process numerically each integral in turn:

$$\int_{-1}^1 \int_{-1}^1 F(\xi, \eta) d\xi d\eta = \int_{-1}^1 d\eta \int_{-1}^1 F(\xi, \eta) d\xi \approx \sum_{i=1}^{p_1} \sum_{j=1}^{p_2} w_i w_j F(\xi_i, \eta_j). \quad (17.15)$$

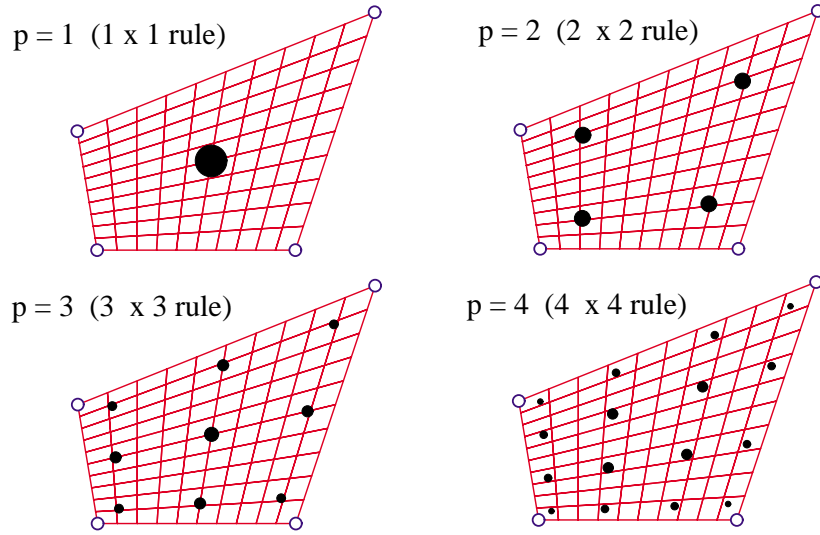


FIGURE 17.4. The first four two-dimensional Gauss product rules $p = 1, 2, 3, 4$ depicted over a straight-sided quadrilateral region. Sample points are marked with black circles. The areas of these circles are proportional to the integration weights.

where p_1 and p_2 are the number of Gauss points in the ξ and η directions, respectively. Usually the same number $p = p_1 = p_2$ is chosen if the shape functions are taken to be the same in the ξ and η directions. This is in fact the case for all quadrilateral elements presented here. The first four two-dimensional Gauss product rules with $p = p_1 = p_2$ are illustrated in Figure 17.4.

§17.3.4. Implementation of 2D Gauss Rules

The *Mathematica* module listed in Figure 17.5 implements two-dimensional product Gauss rules having 1 through 5 points in each direction. The number of points in each direction may be the same or different. If the rule has the same number of points p in both directions the module is called in either of two ways:

$$\begin{aligned} \{\{x_{ii}, etaj\}, wij\} &= \text{QuadGaussRuleInfo}[\{p, \text{numer}\}, \{i, j\}] \\ \{\{x_{ii}, etaj\}, wij\} &= \text{QuadGaussRuleInfo}[\{p, \text{numer}\}, k] \end{aligned} \quad (17.16)$$

The first form is used to get information for point $\{i, j\}$ of the $p \times p$ rule, in which $1 \leq i \leq p$ and $1 \leq j \leq p$. The second form specifies that point by a “visiting counter” k that runs from 1 through p^2 ; if so $\{i, j\}$ are internally extracted¹ as $j = \text{Floor}[(k-1)/p] + 1$; $i = k - p*(j-1)$.

If the integration rule has p_1 points in the ξ direction and p_2 points in the η direction, the module may be called also in two ways:

$$\begin{aligned} \{\{x_{ii}, etaj\}, wij\} &= \text{QuadGaussRuleInfo}[\{p_1, p_2\}, \text{numer}, \{i, j\}] \\ \{\{x_{ii}, etaj\}, wij\} &= \text{QuadGaussRuleInfo}[\{p_1, p_2\}, \text{numer}, k] \end{aligned} \quad (17.17)$$

The meaning of the second argument is as follows. In the first form i runs from 1 to p_1 and j from 1 to p_2 . In the second form k runs from 1 to $p_1 p_2$; if so i and j are extracted by $j = \text{Floor}[(k-1)/p_1] + 1$;

¹ Indices i and j are denoted by $i1$ and $i2$, respectively, inside the module.

```

QuadGaussRuleInfo[{rule_,number_},point_]:=Module[
{ξ,η,p1,p2,i,j,w1,w2,m,info={{Null,Null},0}},
If [Length[rule]==2, {p1,p2}=rule, p1=p2=rule];
If [p1<0, Return[QuadNonProductGaussRuleInfo[
{-p1,number},point]]];
If [Length[point]==2, {i,j}=point, m=point;
j=Floor[(m-1)/p1]+1; i=m-p1*(j-1)];
{ξ,w1}= LineGaussRuleInfo[{p1,number},i];
{η,w2}= LineGaussRuleInfo[{p2,number},j];
info={{ξ,η},w1*w2};
If [number, Return[N[info]], Return[Simplify[info]]];
];

```

FIGURE 17.5. A *Mathematica* module that returns two-dimensional product Gauss rules.

$i=k-p1*(i-1)$. In all four forms, logical flag *number* is set to *True* if numerical information is desired and to *False* if exact information is desired.

The module returns ξ_i and η_j in *xii* and *etaj*, respectively, and the weight product $w_i w_j$ in *wij*. This code is used in the Exercises at the end of the chapter. If the inputs are not in range, the module returns $\{\{Null,Null\},0\}$.

Example 17.3. $\{\{xi,eta\},w\}=QuadGaussRuleInfo[\{3,False\},\{2,3\}]$ returns $xi=0$, $eta=Sqrt[3/5]$ and $w=40/81$.

Example 17.4. $\{\{xi,eta\},w\}=QuadGaussRuleInfo[\{3,True\},\{2,3\}]$ returns (to 16-place precision) $xi=0.$, $eta=0.7745966692414834$ and $w=0.49382716049382713$.

§17.4. The Stiffness Matrix

The stiffness matrix of a general plane stress element is given by the expression (14.23), which is reproduced here:

$$\mathbf{K}^e = \int_{\Omega^e} h \mathbf{B}^T \mathbf{E} \mathbf{B} d\Omega^e \quad (17.18)$$

Of the terms that appear in (17.18) the strain-displacement matrix \mathbf{B} has been discussed previously. The thickness h , if variable, may be interpolated via the shape functions. The stress-strain matrix \mathbf{E} is usually constant in elastic problems, but we could in principle interpolate it as appropriate should it vary over the element. To integrate (17.18) numerically by a two-dimensional product Gauss rule, we have to reduce it to the canonical form (17.14), that is

$$\mathbf{K}^e = \int_{-1}^1 \int_{-1}^1 \mathbf{F}(\xi, \eta) d\xi d\eta. \quad (17.19)$$

If ξ and η are the quadrilateral coordinates, everything in (17.19) already fits this form, except the element of area $d\Omega^e$.

To complete the reduction we need to express $d\Omega^e$ in terms of the differentials $d\xi$ and $d\eta$. The desired relation is (see Remark below)

$$d\Omega^e = dx dy = \det \mathbf{J} d\xi d\eta = J d\xi d\eta. \quad (17.20)$$

The idea behind (17.23) is that $\mathbf{K}_{1 \times 1}^e$ is rank-deficient and too soft whereas $\mathbf{K}_{2 \times 2}^e$ is rank-sufficient but too stiff. A combination of too-soft and too-stiff hopefully “balances” the stiffness. An application of this idea to the mitigation of *shear locking* for modeling in-plane bending is the subject of Exercise E17.4.

§17.5.2. *Selective Integration

In the FEM literature the term *selective integration* is used to describe a scheme for forming \mathbf{K}^e as the sum of two or more matrices computed with different integration rules *and* different constitutive properties.³ We consider here the case of a two-way decomposition. Split the plane stress constitutive matrix \mathbf{E} into two:

$$\mathbf{E} = \mathbf{E}_I + \mathbf{E}_{II} \quad (17.24)$$

This is called a *stress-strain splitting*. Inserting (17.24) into (17.13) the expression of the stiffness matrix becomes

$$\mathbf{K}^e = \int_{\Omega^e} h \mathbf{B}^T \mathbf{E}_I \mathbf{B} d\Omega^e + \int_{\Omega^e} h \mathbf{B}^T \mathbf{E}_{II} \mathbf{B} d\Omega^e = \mathbf{K}_I^e + \mathbf{K}_{II}^e. \quad (17.25)$$

If these two integrals were done through the same integration rule, the stiffness would be identical to that obtained by integrating $h \mathbf{B}^T \mathbf{E} \mathbf{B} d\Omega^e$. The trick is to use two different rules: rule (I) for the first integral and rule (II) for the second.

In practice selective integration is mostly useful for the 4-node bilinear quadrilateral. For this element rules (I) and (II) are the 1×1 and 2×2 Gauss product rules, respectively. Exercises E17.5–7 investigate stress-strain splittings (17.24) that improve the in-plane bending performance of rectangular elements.

Notes and Bibliography

The 4-node quadrilateral has a checkered history. It was first derived as a rectangular panel with edge reinforcements (not included here) by Argyris in his 1954 *Aircraft Engineering* series [12, p. 49 in the Butterworths reprint]. Argyris used bilinear displacement interpolation in Cartesian coordinates.⁴

After much flailing, a conforming generalization to arbitrary geometry was published in 1964 by Taig and Kerr [371] using quadrilateral-fitted coordinates called $\{\xi, \eta\}$ but running from 0 to 1. (Reference [371] cites an 1961 English Electric Aircraft internal report as original source but [218, p. 520] remarks that the work goes back to 1957.) Bruce Irons, who was aware of Taig’s work while at Rolls Royce, changed the $\{\xi, \eta\}$ range to $[-1, 1]$ to fit Gauss quadrature tables. He proceeded to create the seminal isoparametric family as a far-reaching extension upon moving to Swansea [33,96,214–218].

Gauss integration is also called Gauss-Legendre quadrature. Gauss presented these rules, derived from first principles, in 1814; cf. Sec 4.11 of [165]. Legendre’s name is often adjoined because the abscissas of the 1D sample points turned out to be the zeros of Legendre polynomials. A systematic description is given in [362]. For references in multidimensional numerical integration, see **Notes and Bibliography** in Chapter 24.

Selective and reduced integration in FEM developed in the early 1970s, and by now there is a huge literature. An excellent textbook source is [210].

References

Referenced items have been moved to Appendix R.

³ This technique is also called “selective reduced integration” to reflect the fact that one of the rules (the “reduced rule”) underintegrates the element.

⁴ This work is probably the first derivation of a continuum-based finite element by assumed displacements. As noted in §1.7.1, Argyris was aware of the ongoing work in stiffness methods at Turner’s group in Boeing, but the plane stress models presented in [392] were derived by interelement flux assumptions. Argyris used the unit displacement theorem, displacing each DOF in turn by one. The resulting displacement pattern is what is now called a shape function.

Homework Exercises for Chapter 17

Isoparametric Quadrilaterals

The *Mathematica* module `Quad4IsoPMembraneStiffness` listed in Figure E17.1 computes the element stiffness matrix of the 4-node bilinear quadrilateral. This module is useful as a tool for the Exercises that follow.

```

Quad4IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
Module[{i,k,p=2,number=False,h=th,qcoor,c,w,Nf,
  dNx,dNy,Jdet,Be,Ke=Table[0,{8},{8}]},
If [Length[options]==2, {number,p}=options,{number}=options];
If [p<1|p>4, Print["p out of range"]; Return[Null]];
For [k=1, k<=p*p, k++,
  {qcoor,w}= QuadGaussRuleInfo[{p,number},k];
  {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
  If [Length[th]==4, h=th.Nf]; c=w*Jdet*h;
  Be={Flatten[Table[{dNx[[i]], 0},{i,4}]],
    Flatten[Table[{0, dNy[[i]]},{i,4}]],
    Flatten[Table[{dNy[[i]],dNx[[i]]},{i,4}]]};
  Ke+=Simplify[c*Transpose[Be].(Emat.Be)];
]; Return[Simplify[Ke]]
1;

```

FIGURE E17.1. Mathematica module to compute the stiffness matrix of a 4-node bilinear quadrilateral in plane stress.

The module makes use of the shape function module `Quad4IsoPShapeFunDer` listed in Figure 17.1, and of the Gauss integration modules `QuadGaussRuleInfo` and (indirectly) `LineGaussRuleInfo`, listed in Figures 17.5 and are included in the web-posted Notebook `Quad4Stiffness.nb`.⁵ The module is invoked as

$$\text{Ke} = \text{Quad4IsoPMembraneStiffness}[\text{ncoor}, \text{Emat}, \text{thick}, \text{options}] \quad (\text{E17.1})$$

The arguments are:

- ncoor** Quadrilateral node coordinates arranged in two-dimensional list form:
 $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}\}.$
- Emat** A two-dimensional list storing the 3×3 plane stress matrix of elastic moduli:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \quad (\text{E17.2})$$

arranged as $\{\{E_{11}, E_{12}, E_{33}\}, \{E_{12}, E_{22}, E_{23}\}, \{E_{13}, E_{23}, E_{33}\}\}.$ Must be symmetric. If the material is isotropic with elastic modulus E and Poisson's ratio ν , this matrix becomes

$$\mathbf{E} = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1 - \nu) \end{bmatrix} \quad (\text{E17.3})$$

- thick** The plate thickness specified either as a four-entry list: $\{h_1, h_2, h_3, h_4\}$ or as a scalar: $h.$

⁵ This Notebook does not include scripts for doing the Exercises below, although it has some text statements at the bottom of the cell. You will need to enter the Exercise scripts yourself.

The first form is used to specify an element of variable thickness, in which case the entries are the four corner thicknesses and h is interpolated bilinearly. The second form specifies uniform thickness.

options Processing options. This list may contain two items: { **numer**, **p** } or one: { **numer** }.

numer is a logical flag with value **True** or **False**. If **True**, the computations are done in floating point arithmetic. For symbolic or exact arithmetic work set **numer** to **False**.⁶

p specifies the Gauss product rule to have **p** points in each direction. **p** may be 1 through 4. For rank sufficiency, **p** must be 2 or higher. If **p** is 1 the element will be rank deficient by two.⁷ If omitted **p** = 2 is assumed.

The module returns \mathbf{K}^e as an 8×8 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2} \ u_{x3} \ u_{y3} \ u_{x4} \ u_{y4}]^T. \quad (\text{E17.4})$$

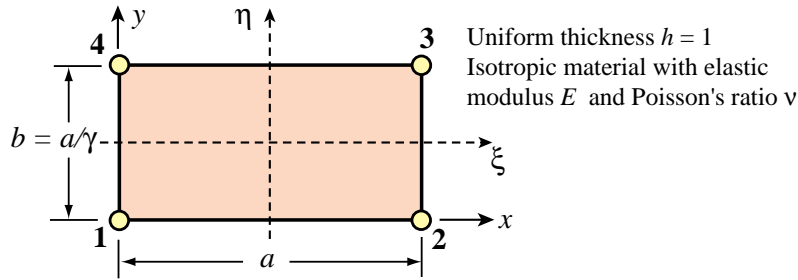


FIGURE E17.2. Element for Exercises 17.1 to 17.3.

For the following three exercises we consider the specialization of the general 4-node bilinear quadrilateral to a *rectangular* element dimensioned a and b in the x and y directions, respectively, as depicted in Figure E17.2. The element has uniform unit thickness h . The material is isotropic with elastic modulus E and Poisson's ratio ν and consequently \mathbf{E} reduces to (E17.3). The stiffness matrix of this element can be expressed in closed form.⁸ For convenience define $\gamma = a/b$ (rectangle aspect ratio), $\psi_1 = (1 + \nu)\gamma$, $\psi_2 = (1 - 3\nu)\gamma$, $\psi_3 = 2 + (1 - \nu)\gamma^2$, $\psi_4 = 2\gamma^2 + (1 - \nu)$, $\psi_5 = (1 - \nu)\gamma^2 - 4$, $\psi_6 = (1 - \nu)\gamma^2 - 1$, $\psi_7 = 4\gamma^2 - (1 - \nu)$ and $\psi_8 = \gamma^2 - (1 - \nu)$. Then the stiffness matrix in closed form is

$$\mathbf{K}^e = \frac{Eh}{24\gamma(1 - \nu^2)} \begin{bmatrix} 4\psi_3 & 3\psi_1 & 2\psi_5 & -3\psi_2 & -2\psi_3 & -3\psi_1 & -4\psi_6 & 3\psi_2 \\ & 4\psi_4 & 3\psi_2 & 4\psi_8 & -3\psi_1 & -2\psi_4 & -3\psi_2 & -2\psi_7 \\ & & 4\psi_3 & -3\psi_1 & -4\psi_6 & -3\psi_2 & -2\psi_3 & 3\psi_1 \\ & & & 4\psi_4 & 3\psi_2 & -2\psi_7 & 3\psi_1 & -2\psi_4 \\ & & & & 4\psi_3 & 3\psi_1 & 2\psi_5 & -3\psi_2 \\ & & & & & 4\psi_4 & 3\psi_2 & 4\psi_8 \\ & & & & & & 4\psi_3 & -3\psi_1 \\ \text{symm} & & & & & & & 4\psi_4 \end{bmatrix}. \quad (\text{E17.5})$$

⁶ The reason for this option is speed. A symbolic or exact computation can take orders of magnitude more time than a floating-point evaluation. This becomes more pronounced as elements get more complicated.

⁷ The rank of an element stiffness is discussed in Chapter 19.

⁸ This closed form can be obtained by either exact integration, or numerical integration with a 2×2 or higher Gauss rule.

EXERCISE 17.1 [C:20] Exercise the *Mathematica* module of Figure E17.1 with the following script:

```
ClearAll[Em,nu,a,b,h]; Em=48; h=1; a=4; b=2; nu=0;
ncoor={{0,0},{a,0},{a,b},{0,b}};
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
For [p=1, p<=4, p++,
  Ke= Quad4IsoPMembraneStiffness[ncoor,Emat,h,{True,p}];
  Print["Gauss integration rule: ",p," x ",p];
  Print["Ke=",Chop[Ke]//MatrixForm];
  Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]]]]
];
```

Verify that for integration rules $p=2,3,4$ the stiffness matrix does not change and has three zero eigenvalues, which correspond to the three two-dimensional rigid body modes. On the other hand, for $p=1$ the stiffness matrix is different and displays five zero eigenvalues, which is physically incorrect. (This phenomenon is analyzed further in Chapter 19.) Question: why does the stiffness matrix stays exactly the same for $p \geq 2$? Hint: take a look at the entries of the integrand $h \mathbf{B}^T \mathbf{E} \mathbf{B} J$; for a *rectangular geometry* are those polynomials in ξ and η , or rational functions? If the former, of what polynomial order in ξ and η are the entries?

EXERCISE 17.2 [C:20] Check the rectangular element stiffness closed form given in (E17.5). This may be done by hand (takes a while) or (quicker) running the script of Figure E17.3, which calls the *Mathematica* module of Figure E17.1.

```
ClearAll[Em,v,a,b,h,gamma]; b=a/gamma;
ncoor={{0,0},{a,0},{a,b},{0,b}};
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};
Ke= Quad4IsoPMembraneStiffness[ncoor,Emat,h,{False,2}];
scaledKe=Simplify[Ke*(24*(1-v^2)*gamma/(Em*h))];
Print["Ke=",Em*h/(24*gamma*(1-v^2)), "\n", scaledKe//MatrixForm];
```

FIGURE E17.3. Script suggested for Exercise E17.2.

The scaling introduced in the last two lines is for matrix visualization convenience. Verify (E17.5) by printout inspection and report any typos to instructor.

EXERCISE 17.3 [A/C:25=5+10+10] A Bernoulli-Euler plane beam of thin rectangular cross-section with span L , height b and thickness h (normal to the plane of the figure) is bent under end moments M as illustrated in Figure E17.4. The beam is fabricated of isotropic material with elastic modulus E and Poisson's ratio ν . The *exact* solution of the beam problem (from both the theory-of-elasticity and beam-theory standpoints) is a constant bending moment M along the span. Consequently the beam deforms with uniform curvature $\kappa = M/(EI_z)$, in which $I_z = \frac{1}{12}hb^3$ is the cross-section second moment of inertia about z .

The beam is modeled with *one layer* of identical 4-node iso-P bilinear quadrilaterals through its height. These are rectangles with horizontal dimension a ; in the Figure $a = L/4$. The aspect ratio b/a is denoted by γ . By analogy with the exact solution, all rectangles in the finite element model will undergo the same deformation. We can therefore isolate a typical element as illustrated in Figure E17.4.

The exact displacement field for the beam segment referred to the $\{x, y\}$ axes placed at the element center as shown in the bottom of Figure E17.4, are

$$u_x = -\kappa xy, \quad u_y = \frac{1}{2}\kappa(x^2 + \nu y^2), \quad (\text{E17.6})$$

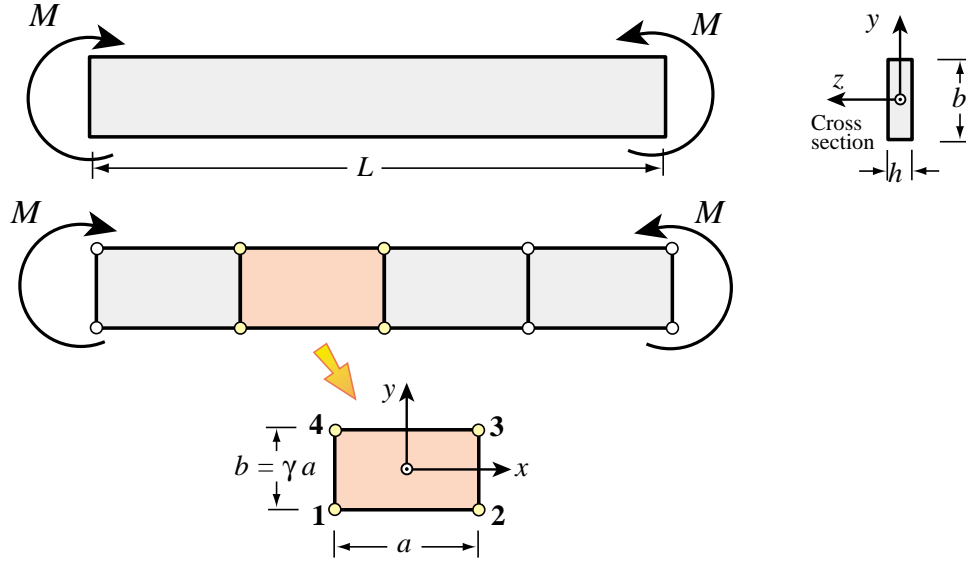


FIGURE E17.4. Pure bending of Bernoulli-Euler plane beam of thin rectangular cross section, for Exercises 17.3–7. The beam is modeled by one layer of 4-node iso-P bilinear quadrilaterals through its height.

where κ is the deformed beam curvature M/EI . The stiffness equations of the typical rectangular element are given by the close form expression (E17.5).

The purpose of this Exercise is to compare the in-plane bending response of the 4-node iso-P bilinear rectangle to that of a Bernoulli-Euler beam element (which would be exact for this configuration). The quadrilateral element will be called *x-bending exact* if it reproduces the beam solution for all $\{\gamma, \nu\}$. This comparison is distributed into three items.

- Check that (E17.6), as a plane stress 2D elasticity solution, is in full agreement with Bernoulli-Euler beam theory. This can be done by computing the strains $e_{xx} = \partial u_x / \partial x$, $e_{yy} = \partial u_y / \partial y$ and $2e_{xy} = \partial u_y / \partial x + \partial u_x / \partial y$. Then get the stresses σ_{xx} , σ_{yy} and σ_{xy} through the plane stress constitutive matrix (E17.3) of an isotropic material. Verify that both σ_{yy} and σ_{xy} vanish for any ν , and that $\sigma_{xx} = -E\kappa y = -My/I_z$, which agrees with equation (13.4) in Chapter 13.
- Compute the strain energy $U_{\text{quad}} = \frac{1}{2}(\mathbf{u}_{\text{beam}})^T \mathbf{K}^e \mathbf{u}_{\text{beam}}$ absorbed by the 4-node element under nodal displacements \mathbf{u}_{beam} constructed by evaluating (E17.6) at the nodes 1,2,3,4. To simplify this calculation, it is convenient to decompose that vector as follows:

$$\mathbf{u}_{\text{beam}} = \mathbf{u}_{\text{beam}}^x + \mathbf{u}_{\text{beam}}^y = \frac{1}{4}\kappa ab \begin{bmatrix} -1 & 0 & 1 & 0 & -1 & 0 & 1 & 0 \end{bmatrix}^T + \frac{1}{8}\kappa(a^2 + \nu b^2) \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}^T \quad (\text{E17.7})$$

Explain why $\mathbf{K}^e \mathbf{u}_{\text{beam}}^y$ must vanish and consequently

$$U_{\text{quad}} = \frac{1}{2}(\mathbf{u}_{\text{beam}}^x)^T \mathbf{K}^e \mathbf{u}_{\text{beam}}^x. \quad (\text{E17.8})$$

This energy can be easily computed by *Mathematica* by using the first 4 lines of the script of the previous Exercise, except that here `ncoor={ {-a,-b},{a,-b},{a,b},{-a,b}}/2`. If vector $\mathbf{u}_{\text{beam}}^x$ is formed in `u` as a one-dimensional list, `Uquad=Simplify[u.Ke.u/2]`. This should come out as a function of M , E , ν , h , a and γ because $\kappa = M/(EI_z) = 12M/(Eha^3\gamma^3)$.

- From Mechanics of Materials, or equation (13.7) of Chapter 13, the strain energy absorbed by the beam segment of length a under a constant bending moment M is $U_{\text{beam}} = \frac{1}{2}M\kappa a = M^2a/(2EI_z) =$

$6M^2/(Eha^2\gamma^3)$. Form the *energy ratio* $r = U_{quad}/U_{beam}$ and show that it is a function of the rectangle aspect ratio $\gamma = b/a$ and of Poisson's ratio ν only:

$$r = r(\gamma, \nu) = \frac{1 + 2/\gamma^2 - \nu}{(2/\gamma^2)(1 - \nu^2)}. \quad (\text{E17.9})$$

This happens to be the ratio of the 2D model solution to the exact (beam) solution. Hence $r = 1$ means that we get the exact answer, that is the 2D model is *x*-bending exact. If $r > 1$ the 2D model is overstiff, and if $r < 1$ the 2D model is overflexible. Evidently $r > 1$ for all γ if $0 \leq \nu \leq \frac{1}{2}$. Moreover if $b \ll a$, $r \gg 1$; for example if $a = 10b$ and $\nu = 0$, $r \approx 50$ and the 2D model gives only about 2% of the correct solution. This phenomenon is referred to in the FEM literature as *shear locking*, because over stiffness is due to the bending motion triggering spurious shear energy in the element. Remedies to shear locking at the element level are studied in advanced FEM courses. Draw conclusions as to the adequacy or inadequacy of the 2D model to capture inplane bending effects, and comment on how you might improve results by modifying the discretization of Figure E17.4.⁹

EXERCISE 17.4 [A+C:20] A naive remedy to shear locking can be attempted with the weighted integration methodology outlined in §17.6.1. Let $\mathbf{K}_{1 \times 1}^e$ and $\mathbf{K}_{2 \times 2}^e$ denote the element stiffnesses produced by 1×1 and 2×2 Gauss product rules, respectively. Take

$$\mathbf{K}_\beta^e = (1 - \beta)\mathbf{K}_{1 \times 1}^e + \beta\mathbf{K}_{2 \times 2}^e \quad (\text{E17.10})$$

where β is adjusted so that shear locking is reduced or eliminated. It is not difficult to find β if the element is rectangular and isotropic. For the definition of *x*-bending exact please read the previous Exercise. Inserting \mathbf{K}_β^e into the test introduced there verify that

$$r = \frac{\beta(1 + 2\gamma^2 - \nu)}{(2/\gamma^2)(1 - \nu^2)}. \quad (\text{E17.11})$$

Whence show that if

$$\beta = \frac{2/\gamma^2(1 - \nu^2)}{1 + 2/\gamma^2 - \nu}, \quad (\text{E17.12})$$

then $r \equiv 1$ for all $\{\gamma, \nu\}$ and the element is *x*-bending exact. A problem with this idea is that it does not make it *y*-bending exact because $r(\gamma) \neq r(1/\gamma)$ if $\gamma \neq 1$. Moreover the device is not easily extended to non-rectangular geometries or non-isotropic material.

EXERCISE 17.5 [A+C:35] (Advanced) To understand this Exercise please begin by reading Exercise 17.3, and the concept of shear locking. The material is again assumed isotropic with elastic modules E and Poisson's ratio ν . The 4-node rectangular element will be said to be *bending exact* if $r = 1$ for any $\{\gamma, \nu\}$ if the bending test described in Exercise 17.3 is done in both *x* and *y* directions. A bending-exact element is completely shear-lock-free.

The selective integration scheme outlined in §17.6.2 is more effective than weighted integration (covered in the previous exercise) to fully eliminate shear locking. Let the integration rules (I) and (II) be the 1×1 and 2×2 product rules, respectively. However the latter is generalized so the sample points are located at $\{-\chi, \chi\}$, $\{\chi, -\chi\}$, $\{\chi, \chi\}$ and $\{-\chi, \chi\}$, with weight 1.¹⁰ Consider the stress-strain splitting

$$\mathbf{E} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} \alpha & \beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} + \frac{E}{1-\nu^2} \begin{bmatrix} 1-\alpha & \nu-\beta & 0 \\ \nu-\beta & 1-\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{E}_I + \mathbf{E}_{II}, \quad (\text{E17.13})$$

⁹ Note that even if we make $a \rightarrow 0$ and $\gamma = b/a \rightarrow \infty$ by taking an infinite number of rectangular elements along *x*, the energy ratio r remains greater than one if $\nu > 0$ since $r \rightarrow 1/(1 - \nu^2)$. Thus the 2D model would not generally converge to the correct solution if we keep one layer through the height.

¹⁰ For a rectangular geometry these sample points lie on the diagonals. In the case of the standard 2-point Gauss product rule $\chi = 1/\sqrt{3}$.

where α and β are scalars. Show that if

$$\chi = \sqrt{\frac{1 - \nu^2}{3(1 - \alpha)}} \quad (\text{E17.14})$$

the resulting element stiffness $\mathbf{K}_I^e + \mathbf{K}_{II}^e$ is bending exact for any $\{\alpha, \beta\}$. As a corollary show that if $\alpha = \nu^2$, which corresponds to the splitting

$$\mathbf{E} = \frac{E}{1 - \nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1 - \nu}{2} \end{bmatrix} = \frac{E}{1 - \nu^2} \begin{bmatrix} \nu^2 & \beta & 0 \\ \beta & \nu^2 & 0 \\ 0 & 0 & \frac{1 - \nu}{2} \end{bmatrix} + \frac{E}{1 - \nu^2} \begin{bmatrix} 1 - \nu^2 & \nu - \beta & 0 \\ \nu - \beta & 1 - \nu^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{E}_I + \mathbf{E}_{II}, \quad (\text{E17.15})$$

then $\chi = 1/\sqrt{3}$ and rule (II) becomes the standard 2×2 Gauss product rule. What are two computationally convenient settings for β ?

EXERCISE 17.6 [A+C:35] (Advanced) A variation on the previous exercise on selective integration to make the isotropic rectangular 4-node element bending exact. Integration rule (I) is not changed. However rule (II) has four sample points located at $\{0, -\chi\}$, $\{\chi, 0\}$, $\{0, \chi\}$ and $\{-\chi, 0\}$ each with weight 1.¹¹ Show that if one selects the stress-strain splitting (E17.13) and

$$\chi = \sqrt{\frac{2(1 - \nu^2)}{3(1 - \alpha)}} \quad (\text{E17.16})$$

the resulting element stiffness $\mathbf{K}_I^e + \mathbf{K}_{II}^e$ is bending exact for any $\{\alpha, \beta\}$. Discuss which choices of α reduce χ to $1/\sqrt{3}$ and $\sqrt{2/3}$, respectively.

EXERCISE 17.7 [A+C:40] (Advanced, research paper level, requires a CAS to be tractable) Extend Exercise 17.5 to consider the case of general anisotropic material:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \quad (\text{E17.17})$$

The rules for the selective integration scheme are as described in Exercise 17.5. The appropriate stress-strain splitting is

$$\mathbf{E} = \mathbf{E}_I + \mathbf{E}_{II} = \begin{bmatrix} E_{11} \alpha_1 & E_{12} \beta & E_{13} \\ E_{12} \beta & E_{22} \alpha_2 & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} + \begin{bmatrix} E_{11}(1 - \alpha_1) & E_{12}(1 - \beta) & 0 \\ E_{12}(1 - \beta) & E_{22}(1 - \alpha_2) & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (\text{E17.18})$$

in which β is arbitrary and

$$\begin{aligned} 1 - \alpha_1 &= \frac{|\mathbf{E}|}{3\chi^2 E_{11}(E_{22}E_{33} - E_{23}^2)} = \frac{1}{3\chi^2 C_{11}}, & 1 - \alpha_2 &= \frac{|\mathbf{E}|}{3\chi^2 E_{22}(E_{11}E_{33} - E_{13}^2)} = \frac{1}{3\chi^2 C_{22}}, \\ |\mathbf{E}| &= \det(\mathbf{E}) = E_{11}E_{22}E_{33} + 2E_{12}E_{13}E_{23} - E_{11}E_{23}^2 - E_{22}E_{13}^2 - E_{33}E_{12}^2, \\ C_{11} &= E_{11}(E_{22}E_{33} - E_{23}^2)/|\mathbf{E}|, & C_{22} &= E_{22}(E_{11}E_{33} - E_{13}^2)/|\mathbf{E}|. \end{aligned} \quad (\text{E17.19})$$

Show that the resulting rectangular element is bending exact for any \mathbf{E} and $\chi \neq 0$. (In practice one would select $\chi = 1/\sqrt{3}$.)

¹¹ This is called a 4-point median rule, since the four points are located on the quadrilateral medians.

18

Shape Function Magic

TABLE OF CONTENTS

	Page
§18.1. Requirements	18-3
§18.2. Direct Fabrication of Shape Functions	18-3
§18.3. Triangular Element Shape Functions	18-4
§18.3.1. The Three-Node Linear Triangle	18-4
§18.3.2. The Six-Node Quadratic Triangle	18-5
§18.4. Quadrilateral Element Shape Functions	18-6
§18.4.1. The Four-Node Bilinear Quadrilateral	18-6
§18.4.2. The Nine-Node Biquadratic Quadrilateral	18-7
§18.4.3. The Eight-Node “Serendipity” Quadrilateral	18-9
§18.5. Does the Magic Wand Always Work?	18-10
§18.5.1. Hierarchical Corrections	18-10
§18.5.2. Transition Element Example	18-11
§18.6. *Mathematica Modules to Plot Shape Functions	18-12
§18. Notes and Bibliography	18-14
§18. References	18-14
§18. Exercises	18-15

§18.1. Requirements

This Chapter explains, through a series of examples, how isoparametric shape functions can be directly constructed by geometric considerations. For a problem of variational index 1, the isoparametric shape function N_i^e associated with node i of element e must satisfy the following conditions:

- (A) *Interpolation condition.* Takes a unit value at node i , and is zero at all other nodes.
- (B) *Local support condition.* Vanishes over any element boundary (a side in 2D, a face in 3D) that does not include node i .
- (C) *Interelement compatibility condition.* Satisfies C^0 continuity between adjacent elements over any element boundary that includes node i .
- (D) *Completeness condition.* The interpolation is able to represent exactly any displacement field which is a linear polynomial in x and y ; in particular, a constant value.

Requirement (A) follows directly by interpolation from node values. Conditions (B), (C) and (D) are consequences of the *convergence* requirements discussed further in the next Chapter.¹ For the moment these three conditions may be viewed as recipes.

One can readily verify that all isoparametric shape function sets listed in Chapter 16 satisfy the first two conditions from construction. Direct verification of condition (C) is also straightforward for those examples. A statement equivalent to (C) is that the value of the shape function over a side (in 2D) or face (in 3D) common to two elements must uniquely depend only on its nodal values on that side or face.

Completeness is a property of *all* element isoparametric shape functions taken together, rather than of an individual one. If the element satisfies (B) and (C), in view of the discussion in §16.6 it is sufficient to check that the *sum of shape functions is identically one*.

§18.2. Direct Fabrication of Shape Functions

Contrary to the what the title of this Chapter implies, the isoparametric shape functions listed in Chapter 16 did not come out of a magician's hat. They can be derived systematically by a judicious inspection process. By "inspection" it is meant that the *geometric* visualization of shape functions plays a crucial role.

The method is based on the following observation. In all examples given so far the isoparametric shape functions are given as *products* of fairly simple polynomial expressions in the natural coordinates. This is no accident but a direct consequence of the definition of natural coordinates. All shape functions of Chapter 16 can be expressed as the product of m factors:

$$N_i^e = c_i L_1 L_2 \dots L_m, \quad (18.1)$$

where

$$L_j = 0, \quad j = 1, \dots, m. \quad (18.2)$$

are the homogeneous equation of lines or curves expressed as *linear* functions in the natural coordinates, and c_i is a normalization coefficient.

¹ Convergence means that the discrete FEM solution approaches the exact analytical solution as the mesh is refined.

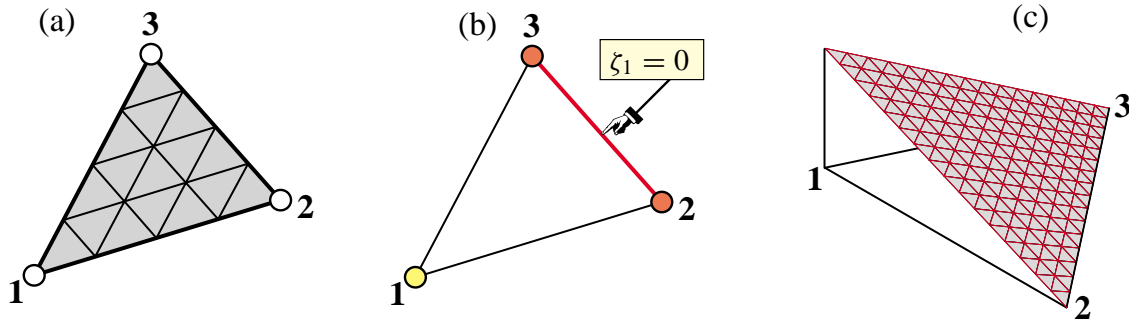


FIGURE 18.1. The three-node linear triangle: (a) element geometry; (b) equation of side opposite corner 1; (c) perspective view of the shape function $N_1 = \zeta_1$.

For two-dimensional isoparametric elements, the ingredients in (18.1) are chosen according to the following five rules.

- R1 Select the L_j as the minimal number of lines or curves linear in the natural coordinates that cross all nodes except the i^{th} node. (A *sui generis* “cross the dots” game.) Primary choices in 2D are the element sides and medians.
- R2 Set coefficient c_i so that N_i^e has the value 1 at the i^{th} node.
- R3 Check that N_i^e vanishes over all element sides that do not contain node i .
- R4 Check the polynomial order over each side that contains node i . If the order is n , there must be exactly $n + 1$ nodes on the side for compatibility to hold.
- R5 If local support (R3) and interelement compatibility (R4) are satisfied, check that the sum of shape functions is identically one.

The examples that follow show these rules in action for two-dimensional elements. Essentially the same technique is applicable to one- and three-dimensional elements.

§18.3. Triangular Element Shape Functions

This section illustrates the use of (18.1) in the construction of shape functions for the linear and the quadratic triangle. The cubic triangle is dealt with in Exercise 18.1.

§18.3.1. The Three-Node Linear Triangle

Figure 18.1 shows the three-node linear triangle that was studied in detail in Chapter 15. The three shape functions are simply the triangular coordinates: $N_i = \zeta_i$, for $i = 1, 2, 3$. Although this result follows directly from the linear interpolation formula of §15.2.4, it can be also quickly derived from the present methodology as follows.

The equation of the triangle side opposite to node i is $L_{j-k} = \zeta_i = 0$, where j and k are the cyclic permutations of i . Here symbol L_{j-k} denotes the left hand side of the homogeneous equation of the natural coordinate line that passes through node points j and k . See Figure 18.1(b) for $i = 1$, $j = 2$ and $k = 3$. Hence the obvious guess is

$$N_i^e \stackrel{\text{guess}}{=} c_i L_i. \quad (18.3)$$

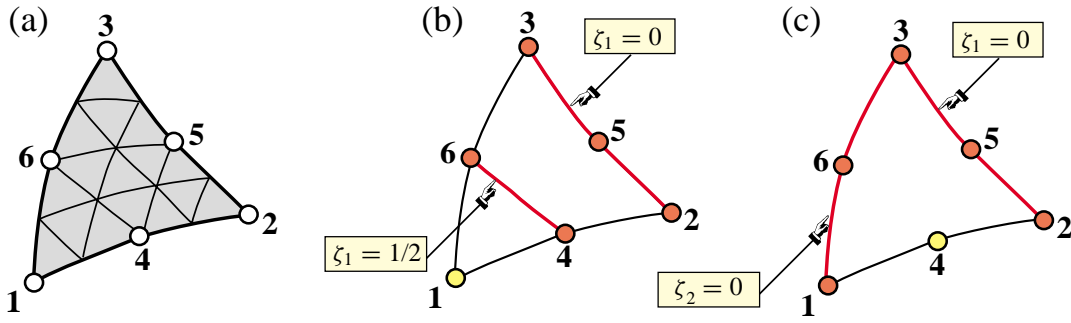


FIGURE 18.2. The six-node quadratic triangle: (a) element geometry; (b) lines (in red) whose product yields N_1^e ; (c) lines (in red) whose product yields N_4^e .

This satisfies conditions (A) and (B) except the unit value at node i ; this holds if $c_i = 1$. The local support condition (B) follows from construction: the value of ζ_i is zero over side $j-k$. Inter-element compatibility follows from R4: the variation of ζ_i along the 2 sides meeting at node i is linear and that there are two nodes on each side; cf. §15.4.2. Completeness follows since $N_1^e + N_2^e + N_3^e = \zeta_1 + \zeta_2 + \zeta_3 = 1$. Figure 18.1(c) depicts $N_1^e = \zeta_1$, drawn normal to the element in perspective view.

§18.3.2. The Six-Node Quadratic Triangle

The geometry of the six-node quadratic triangle is shown in Figure 18.2(a). Inspection reveals two types of nodes: corners (1, 2 and 3) and midside nodes (4, 5 and 6). Consequently we can expect two types of associated shape functions. We select nodes 1 and 4 as representative cases.

For both cases we try the product of *two* linear functions in the triangular coordinates because we expect the shape functions to be quadratic. These functions are illustrated in Figures 18.2(b,c) for corner node 1 and midside node 4, respectively.

For corner node 1, inspection of Figure 18.2(b) suggests trying

$$N_1^e \stackrel{\text{guess}}{=} c_1 L_{2-3} L_{4-6}, \quad (18.4)$$

Why is (18.4) expected to work? Clearly N_1^e will vanish over 2-5-3 and 4-6. This makes the function zero at nodes 2 through 6, as is obvious upon inspection of Figure 18.2(b), while being nonzero at node 1. This value can be adjusted to be unity if c_1 is appropriately chosen. The equations of the lines that appear in (18.4) are

$$L_{2-3}: \quad \zeta_1 = 0, \quad L_{4-6}: \quad \zeta_1 - \frac{1}{2} = 0. \quad (18.5)$$

Replacing into (18.3) we get

$$N_1^e = c_1 \zeta_1 (\zeta_1 - \frac{1}{2}), \quad (18.6)$$

To find c_1 , evaluate $N_1^e(\zeta_1, \zeta_2, \zeta_3)$ at node 1. The triangular coordinates of this node are $\zeta_1 = 1$, $\zeta_2 = \zeta_3 = 0$. We require that it takes a unit value there: $N_1^e(1, 0, 0) = c_1 \times 1 \times \frac{1}{2} = 1$ whence $c_1 = 2$ and finally

$$N_1^e = 2\zeta_1 (\zeta_1 - \frac{1}{2}) = \zeta_1 (2\zeta_1 - 1), \quad (18.7)$$

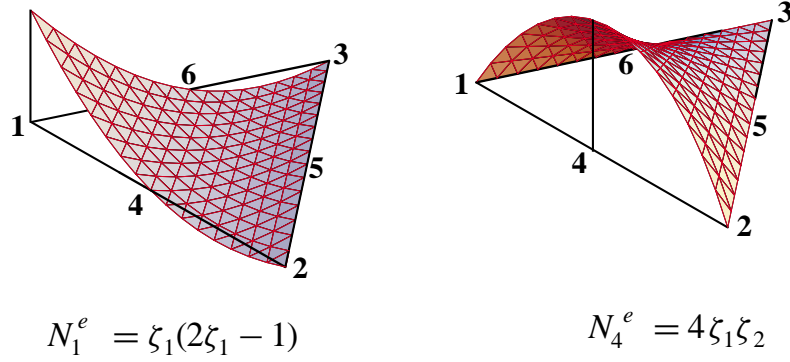


FIGURE 18.3. Perspective view of shape functions N_1^e and N_4^e for the quadratic triangle. The plot is done over a straight side triangle for programming simplicity.

as listed in §16.5.2. Figure 18.3 shows a perspective view. The other two corner shape functions follow by cyclic permutations of the corner index.

For midside node 4, inspection of Figure 18.2(c) suggests trying

$$N_4^e \stackrel{\text{guess}}{=} c_4 L_{2-3} L_{1-3} \quad (18.8)$$

Evidently (18.8) satisfies requirements (A) and (B) if c_4 is appropriately normalized. The equation of sides L_{2-3} and L_{1-3} are $\zeta_1 = 0$ and $\zeta_2 = 0$, respectively. Therefore $N_4^e(\zeta_1, \zeta_2, \zeta_3) = c_4 \zeta_1 \zeta_2$. To find c_4 , evaluate this function at node 4, the triangular coordinates of which are $\zeta_1 = \zeta_2 = \frac{1}{2}$, $\zeta_3 = 0$. We require that it takes a unit value there: $N_4^e(\frac{1}{2}, \frac{1}{2}, 0) = c_4 \times \frac{1}{2} \times \frac{1}{2} = 1$. Hence $c_4 = 4$, which gives

$$N_4^e = 4\zeta_1\zeta_2 \quad (18.9)$$

as listed in §16.5.2. Figure 18.3 shows a perspective view of this shape function. The other two midside shape functions follow by cyclic permutations of the node indices.

It remains to carry out the interelement continuity check. Consider node 1. The boundaries containing node 1 and common to adjacent elements are 1–2 and 1–3. Over each one the variation of N_1^e is quadratic in ζ_1 . Therefore the polynomial order over each side is 2. Because there are three nodes on each boundary, the compatibility condition (C) of §18.1 is verified. A similar check can be carried out for midside node shape functions. Exercise 16.1 verified that the sum of the N_i is unity. Therefore the element is complete.

§18.4. Quadrilateral Element Shape Functions

Three quadrilateral elements, with 4, 9 and 8 nodes, respectively, which are commonly used in computational mechanics serve as examples to illustrate the construction of shape functions. Elements with more nodes, such as the bicubic quadrilateral, are not treated as they are rarely used.

§18.4.1. The Four-Node Bilinear Quadrilateral

The element geometry and natural coordinates are shown in Figure 18.4(a). Only one type of node (corner) and associated shape function is present. Consider node 1 as typical. Inspection of

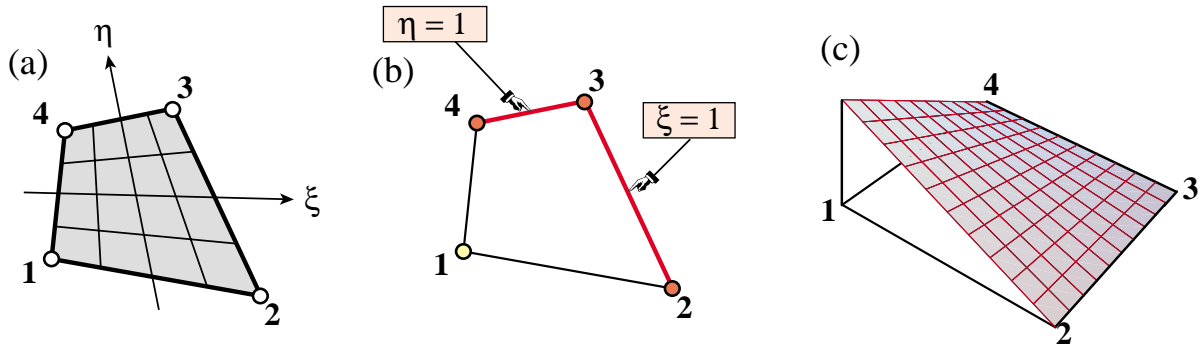


FIGURE 18.4. The four-node bilinear quadrilateral: (a) element geometry; (b) sides (in red) that do not contain corner 1; (c) perspective view of the shape function N_1^e .

Figure 18.4(b) suggests trying

$$N_1^e \stackrel{\text{guess}}{=} c_1 L_{2-3} L_{3-4} \quad (18.10)$$

This plainly vanishes over nodes 2, 3 and 4, and can be normalized to unity at node 1 by adjusting c_1 . By construction it vanishes over the sides 2-3 and 3-4 that do not belong to 1. The equation of side 2-3 is $\xi = 1$, or $\xi - 1 = 0$. The equation of side 3-4 is $\eta = 1$, or $\eta - 1 = 0$. Replacing in (18.10) yields

$$N_1^e(\xi, \eta) = c_1(\xi - 1)(\eta - 1) = c_1(1 - \xi)(1 - \eta). \quad (18.11)$$

To find c_1 , evaluate at node 1, the natural coordinates of which are $\xi = \eta = -1$:

$$N_1^e(-1, -1) = c_1 \times 2 \times 2 = 4c_1 = 1. \quad (18.12)$$

Hence $c_1 = \frac{1}{4}$ and the shape function is

$$N_1^e = \frac{1}{4}(1 - \xi)(1 - \eta), \quad (18.13)$$

as listed in §16.6.2. Figure 18.4(c) shows a perspective view.

For the other three nodes the procedure is the same, traversing the element cyclically. It can be verified that the general expression of the shape functions for this element is

$$N_i^e = \frac{1}{4}(1 + \xi_i \xi)(1 + \eta_i \eta). \quad (18.14)$$

The continuity check proceeds as follows, using N_1^e as example. Node 1 belongs to interelement boundaries 1-2 and 1-3. Over side 1-2, $\eta = -1$ is constant and N_1^e is a *linear* function of ξ . To see this, replace $\eta = -1$ in (18.13). Over side 1-3, $\xi = -1$ is constant and N_1^e is a *linear* function of η . Consequently the polynomial variation order is 1 over both sides. Because there are two nodes on each side the compatibility condition is satisfied. The sum of the shape functions is one, as shown in (16.21); thus the element is complete.

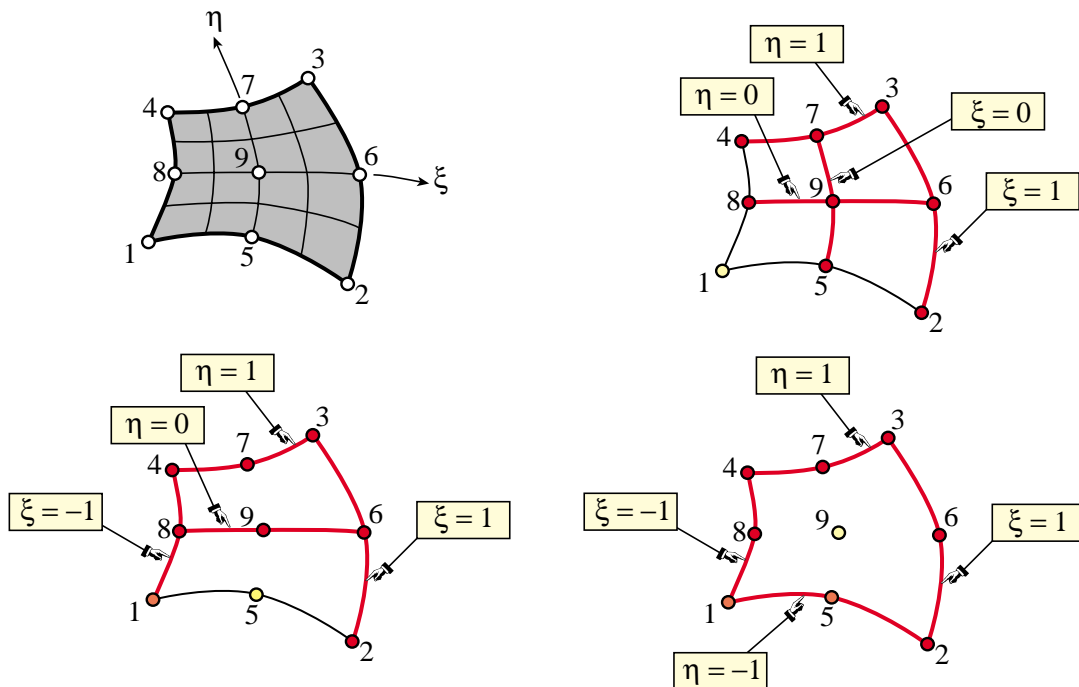


FIGURE 18.5. The nine-node biquadratic quadrilateral: (a) element geometry; (b,c,d): lines (in red) whose product makes up the shape functions N_1^e , N_5^e and N_9^e , respectively.

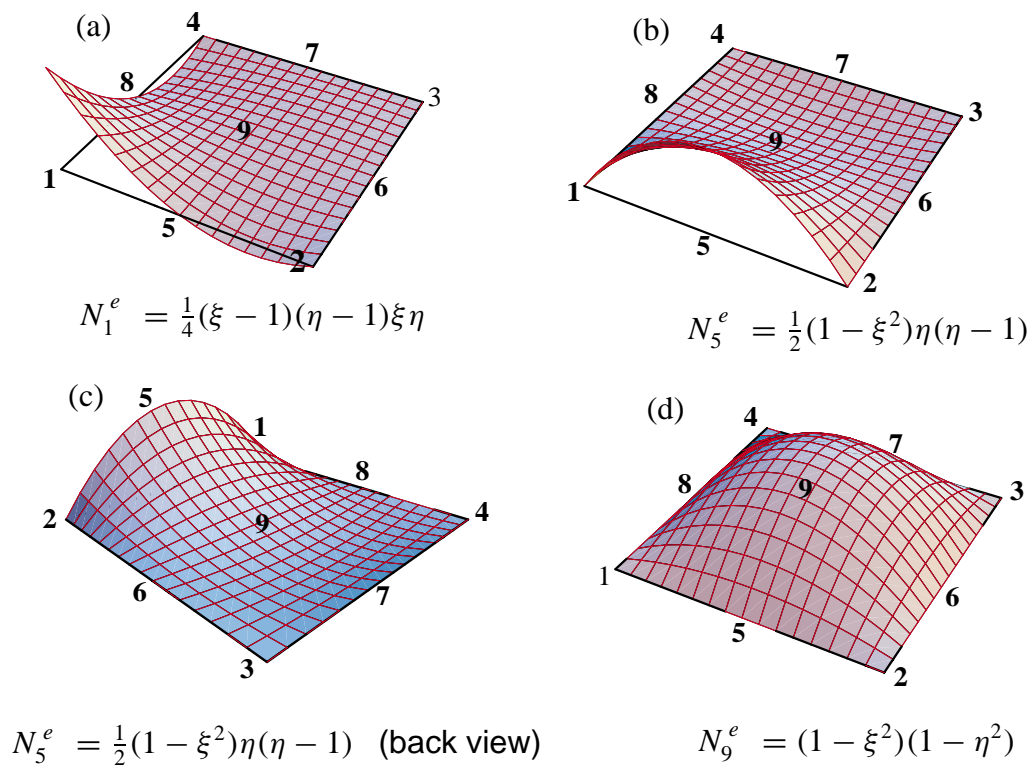


FIGURE 18.6. Perspective view of the shape functions for nodes 1, 5 and 9 of the nine-node biquadratic quadrilateral.

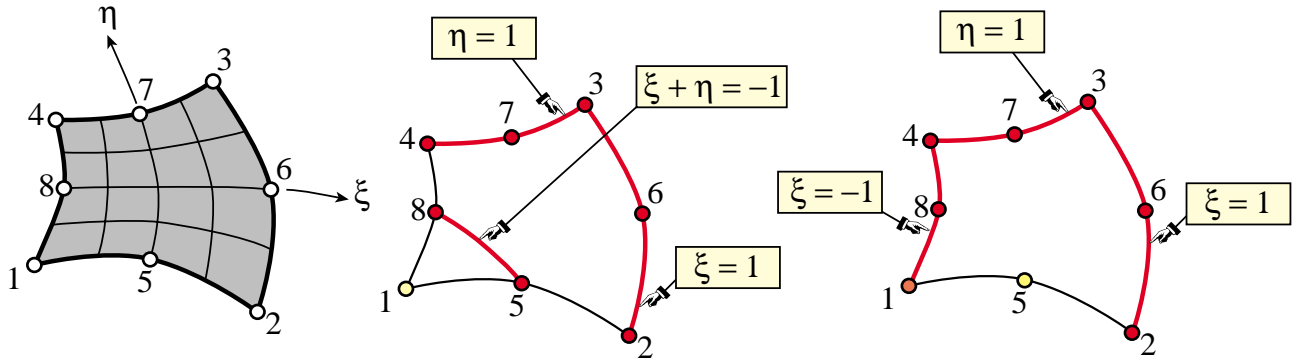


FIGURE 18.7. The eight-node serendipity quadrilateral: (a) element geometry; (b,c): lines (in red) whose product make up the shape functions N_1^e and N_5^e , respectively.

§18.4.2. The Nine-Node Biquadratic Quadrilateral

The element geometry is shown in Figure 18.5(a). This element has three types of shape functions, which are associated with corner nodes, midside nodes and center node, respectively.

The lines whose product is used to construct three types of shape functions are illustrated in Figure 18.5(b,c,d) for nodes 1, 5 and 9, respectively. The technique has been sufficiently illustrated in previous examples. Here we summarize the calculations for nodes 1, 5 and 9, which are taken as representatives of the three types:

$$N_1^e = c_1 L_{2-3} L_{3-4} L_{5-7} L_{6-8} = c_1 (\xi - 1)(\eta - 1)\xi\eta. \quad (18.15)$$

$$N_5^e = c_5 L_{2-3} L_{1-4} L_{6-8} L_{3-4} = c_5 (\xi - 1)(\xi + 1)\eta(\eta - 1) = c_5 (1 - \xi^2)\eta(1 - \eta). \quad (18.16)$$

$$N_9^e = c_9 L_{1-2} L_{2-3} L_{3-4} L_{4-1} = c_9 (\xi - 1)(\eta - 1)(\xi + 1)(\eta + 1) = c_9 (1 - \xi^2)(1 - \eta^2) \quad (18.17)$$

Imposing the normalization conditions we find

$$c_1 = \frac{1}{4}, \quad c_5 = -\frac{1}{2}, \quad c_9 = 1, \quad (18.18)$$

and we obtain the shape functions listed in §16.6.3. Perspective views are shown in Figure 18.6. The remaining N_i 's are constructed through a similar procedure.

Verification of the interelement continuity condition is immediate: the polynomial variation order of N_i^e over any side that belongs to node i is two and there are three nodes on each side. Exercise 16.2 checks that the sum of shape function is unity. Thus the element is complete.

§18.4.3. The Eight-Node “Serendipity” Quadrilateral

This is an eight-node quadrilateral element that results when the center node 9 of the biquadratic quadrilateral is eliminated by kinematic constraints. The geometry and node configuration is shown in Figure 18.7(a). This element has been widely used in commercial codes since the 70s for static problems. It is gradually being phased out in favor of the 9-node quadrilateral for dynamic problems.

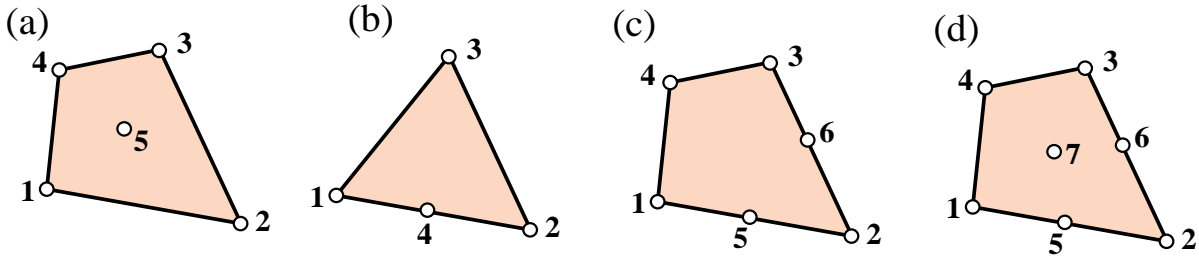


FIGURE 18.8. Node configurations for which the magic recipe does not work.

The 8-node quadrilateral has two types of shape functions, which are associated with corner nodes and midside nodes. Lines whose products yields the shape functions for nodes 1 and 5 are shown in Figure 18.7(b,c).

Here are the calculations for shape functions of nodes 1 and 5, which are taken again as representative cases.

$$N_1^e = c_1 L_{2-3} L_{3-4} L_{5-8} = c_1 (\xi - 1)(\eta - 1)(1 + \xi + \eta) = c_1 (1 - \xi)(1 - \eta)(1 + \xi + \eta), \quad (18.19)$$

$$N_5^e = c_5 L_{2-3} L_{3-4} L_{4-1} = c_5 (\xi - 1)(\xi + 1)(\eta - 1) = c_5 (1 - \xi^2)(1 - \eta). \quad (18.20)$$

Imposing the normalization conditions we find

$$c_1 = -\frac{1}{4}, \quad c_5 = \frac{1}{2} \quad (18.21)$$

The other shape functions follow by appropriate permutation of nodal indices. The interelement continuity and completeness verification are similar to that carried out for the nine-node element, and are relegated to exercises.

§18.5. Does the Magic Wand Always Work?

The “cross the dots” recipe (18.1)–(18.2) is not foolproof. It fails for certain node configurations although it is a reasonable way to start. It runs into difficulties, for instance, in the problem posed in Exercise 18.6, which deals with the 5-node quadrilateral depicted in Figure 18.8(a). If for node 1 one tries the product of side 2–3, side 3–4, and the diagonal 2–5–4, the shape function is easily worked out to be $N_1^e = -\frac{1}{8}(1 - \xi)(1 - \eta)(\xi + \eta)$. This satisfies conditions (A) and (B). However, it violates (C) along sides 1–2 and 4–1, because it varies quadratically over them with only two nodes per side.

§18.5.1. Hierarchical Corrections

A more robust technique relies on a *correction approach*, which employs a combination of terms such as (18.1). For example, a combination of two patterns, one with m factors and one with n factors, is

$$N_i^e = c_i L_1^c L_2^c \dots L_m^c + d_i L_1^d L_2^d \dots L_n^d, \quad (18.22)$$

Here two normalization coefficients: c_i and d_i , appear. In practice trying forms such as (18.22) from scratch becomes cumbersome. The development is best done *hierarchically*. The first term is

taken to be that of a lower order element, called the *parent element*, for which the one-shot approach works. The second term is then a corrective shape function that vanishes at the nodes of the parent element. If this is insufficient one more corrective term is added, and so on.

The technique is best explained through examples. Exercise 18.6 illustrates the procedure for the element of Figure 18.8(a). The next subsection works out the element of Figure 18.8(b).

§18.5.2. Transition Element Example

The hierarchical correction technique is useful for *transition elements*, which have corner nodes but midnodes only over certain sides. Three examples are pictured in Figure 18.8(b,c,d). Shape functions that work can be derived with one, two and three hierarchical corrections, respectively.

As an example, let us construct the shape function N_1^e for the 4-node transition triangle shown in Figure 18.8(b). Candidate lines for the recipe (18.1) are obviously the side 2–3: $\zeta_1 = 0$, and the median 3–4: $\zeta_1 = \zeta_2$. Accordingly we try

$$N_1^e \stackrel{\text{guess}}{=} c_1 \zeta_1 (\zeta_1 - \zeta_2), \quad N_1(1, 0, 0) = 1 = c_1. \quad (18.23)$$

This function $N_1^e = \zeta_1(\zeta_1 - \zeta_2)$ satisfies conditions (A) and (B) but fails compatibility: over side 1–3 of equation $\zeta_2 = 0$, because $N_1^e(\zeta_1, 0, \zeta_3) = \zeta_1^2$. This varies quadratically but there are only 2 nodes on that side. Thus (18.23) is no good.

To proceed hierarchically we start from the shape function for the 3-node linear triangle: $N_1^e = \zeta_1$. This will not vanish at node 4, so apply a correction that vanishes at all nodes but 4. From knowledge of the quadratic triangle midpoint functions, that is obviously $\zeta_1 \zeta_2$ times a coefficient to be determined. The new guess is

$$N_1^e \stackrel{\text{guess}}{=} \zeta_1 + c_1 \zeta_1 \zeta_2. \quad (18.24)$$

Coefficient c_1 is determined by requiring that N_1^e vanish at 4: $N_1^e(\frac{1}{2}, \frac{1}{2}, 0) = \frac{1}{2} + c_1 \frac{1}{4} = 0$, whence $c_1 = -2$ and the shape function is

$$N_1^e = \zeta_1 - 2\zeta_1 \zeta_2. \quad (18.25)$$

This is easily checked to satisfy compatibility on all sides. The verification of completeness is left to Exercise 18.8.

Note that since $N_1^e = \zeta_1(1 - 2\zeta_2)$, (18.25) can be constructed as the normalized product of lines $\zeta_1 = 0$ and $\zeta_2 = 1/2$. The latter passes through 4 and is parallel to 1–3. As part of the opening moves in the shape function game this would be a lucky guess indeed. If one goes to a more complicated element no obvious factorization is possible.

Cell 18.1 *Mathematica* Module to Draw a Function over a Triangle Region

```

PlotTriangleShapeFunction[xytrig_,f_,Nsub_,aspect_]:=Module[
  {Ni,line3D={},poly3D={},zc1,zc2,zc3,xyf1,xyf2,xyf3,
   xc,yc, x1,x2,x3,y1,y2,y3,z1,z2,z3,iz1,iz2,iz3,d},
  {{x1,y1,z1},{x2,y2,z2},{x3,y3,z3}}=Take[xytrig,3];
  xc={x1,x2,x3}; yc={y1,y2,y3}; Ni=Nsub*3;
  Do [ Do [iz3=Ni-iz1-iz2; If [iz3<=0, Continue[]]; d=0;
    If [Mod[iz1+2,3]==0&&Mod[iz2-1,3]==0, d= 1];
    If [Mod[iz1-2,3]==0&&Mod[iz2+1,3]==0, d=-1];
    If [d==0, Continue[]];
    zc1=N[{iz1+d,d,iz2-d,iz3-d}/Ni];
    zc2=N[{iz1-d,iz2+d,d,iz3-d}/Ni];
    zc3=N[{iz1-d,iz2-d,iz3+d,d}/Ni];
    xyf1={xc.zc1,yc.zc1,f[zc1[[1]],zc1[[2]],zc1[[3]]]};
    xyf2={xc.zc2,yc.zc2,f[zc2[[1]],zc2[[2]],zc2[[3]]]};
    xyf3={xc.zc3,yc.zc3,f[zc3[[1]],zc3[[2]],zc3[[3]]]};
    AppendTo[poly3D,Polygon[{xyf1,xyf2,xyf3}]];
    AppendTo[line3D,Line[{xyf1,xyf2,xyf3,xyf1}]],
  {iz2,1,Ni-iz1}],{iz1,1,Ni}];
  Show[ Graphics3D[RGBColor[1,0,0]],Graphics3D[poly3D],
    Graphics3D[Thickness[.002]],Graphics3D[line3D],
    Graphics3D[RGBColor[0,0,0]],Graphics3D[Thickness[.005]],
    Graphics3D[Line[xytrig]],PlotRange->All,
    BoxRatios->{1,1,aspect},Boxed->False]
];
ClearAll[f1,f4];
xyc1={0,0,0}; xyc2={3,0,0}; xyc3={Sqrt[3],3/2,0};
xytrig=N[{xyc1,xyc2,xyc3,xyc1}]; Nsub=16;
f1[zeta1_,zeta2_,zeta3_]:=zeta1*(2*zeta1-1);
f4[zeta1_,zeta2_,zeta3_]:=4*zeta1*zeta2;
PlotTriangleShapeFunction[xytrig,f1,Nsub,1/2];
PlotTriangleShapeFunction[xytrig,f4,Nsub,1/2.5];

```

§18.6. *Mathematica Modules to Plot Shape Functions

A *Mathematica* module called `PlotTriangleShape Functions`, listed in Cell 18.1, has been developed to draw perspective plots of shape functions $N_i(\zeta_1, \zeta_2, \zeta_3)$ over a triangular region. The region is assumed to have straight sides to simplify the logic. The test statements that follow the module produce the shape function plots shown in Figure 18.3 for the 6-node quadratic triangle. Argument `Nsub` controls the plot resolution while `aspect` controls the xyz box aspect ratio. The remaining arguments are self explanatory.

Another *Mathematica* module called `PlotQuadrilateralShape Functions`, listed in Cell 18.2, has been developed to produce perspective plots of shape functions $N_i(\xi, \eta)$ over a quadrilateral region. The region is assumed to have straight sides to simplify the logic. The test statements that follow the module produce

Cell 18.2 *Mathematica* Module to Draw a Function over a Quadrilateral Region

```

PlotQuadrilateralShapeFunction[xyquad_,f_,Nsub_,aspect_]:=Module[
{Ne,Nev,line3D={},poly3D={},xyf1,xyf2,xyf3,i,j,n,ixi,ieta,
xi,eta,x1,x2,x3,x4,y1,y2,y3,y4,z1,z2,z3,z4,xc,yc},
{{x1,y1,z1},{x2,y2,z2},{x3,y3,z3},{x4,y4,z4}}=Take[xyquad,4];
xc={x1,x2,x3,x4}; yc={y1,y2,y3,y4};
Ne[xi_,eta_]:=N[{(1-xi)*(1-eta),(1+xi)*(1-eta),
(1+xi)*(1+eta),(1-xi)*(1+eta)}/4]; n=Nsub;
Do [ Do [ ixi=(2*i-n-1)/n; ieta=(2*j-n-1)/n;
{xi,eta}=N[{ixi-1/n,ieta-1/n}]; Nev=Ne[xi,eta];
xyf1={xc.Nev,yc.Nev,f[xi,eta]};
{xi,eta}=N[{ixi+1/n,ieta-1/n}]; Nev=Ne[xi,eta];
xyf2={xc.Nev,yc.Nev,f[xi,eta]};
{xi,eta}=N[{ixi+1/n,ieta+1/n}]; Nev=Ne[xi,eta];
xyf3={xc.Nev,yc.Nev,f[xi,eta]};
{xi,eta}=N[{ixi-1/n,ieta+1/n}]; Nev=Ne[xi,eta];
xyf4={xc.Nev,yc.Nev,f[xi,eta]};
AppendTo[poly3D,Polygon[{xyf1,xyf2,xyf3,xyf4}]];
AppendTo[line3D,Line[{xyf1,xyf2,xyf3,xyf4,xyf1}]],
{i,1,Nsub}],{j,1,Nsub}];
Show[ Graphics3D[RGBColor[1,0,0]],Graphics3D[poly3D],
Graphics3D[Thickness[.002]],Graphics3D[line3D],
Graphics3D[RGBColor[0,0,0]],Graphics3D[Thickness[.005]],
Graphics3D[Line[xyquad]], PlotRange->All,
BoxRatios->{1,1,aspect},Boxed->False]
];
ClearAll[f1,f5,f9];
xyc1={0,0,0}; xyc2={3,0,0}; xyc3={3,3,0}; xyc4={0,3,0};
xyquad=N[{xyc1,xyc2,xyc3,xyc4,xyc1}]; Nsub=16;
f1[xi_,eta_]:= (1/2)*(xi-1)*(eta-1)*xi*eta;
f5[xi_,eta_]:= (1/2)*(1-xi^2)*eta*(eta-1);
f9[xi_,eta_]:= (1-xi^2)*(1-eta^2);
PlotQuadrilateralShapeFunction[xyquad,f1,Nsub,1/2];
PlotQuadrilateralShapeFunction[xyquad,f5,Nsub,1/2.5];
PlotQuadrilateralShapeFunction[xyquad,f9,Nsub,1/3];

```

the shape function plots shown in Figure 18.6(a,b,d) for the 9-node biquadratic quadrilateral. Argument *Nsub* controls the plot resolution while *aspect* controls the *xyz* box aspect ratio. The remaining arguments are self explanatory.

Notes and Bibliography

The name “shape functions” for interpolation functions directly expressed in terms of physical coordinates (the node displacements in the case of isoparametric elements) was coined by Irons. The earliest published reference seems to be the paper [33]. This was presented in 1965 at the first Wright-Patterson conference, the first all-FEM meeting that strongly influenced the development of computational mechanics in Generation 2. The key connection to numerical integration was presented in [214], although it is mentioned in prior internal reports. A comprehensive exposition is given in the textbook by Irons and Ahmad [218].

The quick way of developing shape functions presented here was used in the writer’s 1966 thesis [100] for triangular elements. The qualifier “magic” arose from the timing for covering this Chapter in a Fall Semester course: the lecture falls near Halloween.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 18

Shape Function Magic

EXERCISE 18.1 [A/C:10+10] The complete cubic triangle for plane stress has 10 nodes located as shown in Figure E18.1, with their triangular coordinates listed in parentheses.

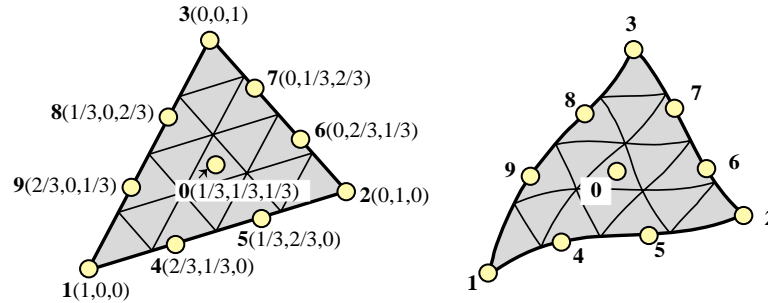


FIGURE E18.1. Ten-node cubic triangle for Exercise 18.1. The left picture shows the superparametric element whereas the right one shows the isoparametric version with curved sides.

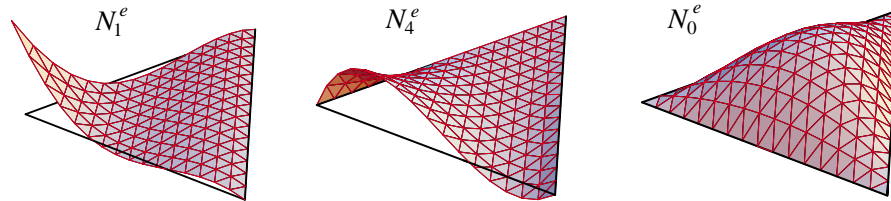


FIGURE E18.2. Perspective plots of the shape functions N_1^e , N_4^e and N_0^e for the 10-node cubic triangle.

- Construct the cubic shape functions N_1^e , N_4^e and N_0^e for nodes 1, 4, and 0 (the interior node is labeled as zero, not 10) using the line-product technique. [Hint: each shape function is the product of 3 and only 3 lines.] Perspective plots of those 3 functions are shown in Figure E18.2.
- Construct the missing 7 shape functions by appropriate node number permutations, and verify that the sum of the 10 functions is identically one. For the unit sum check use the fact that $\zeta_1 + \zeta_2 + \zeta_3 = 1$.

EXERCISE 18.2 [A:15] Find an alternative shape function N_1^e for corner node 1 of the 9-node quadrilateral of Figure 18.5(a) by using the diagonal lines 5–8 and 2–9–4 in addition to the sides 2–3 and 3–4. Show that the resulting shape function violates the compatibility condition (C) stated in §18.1.

EXERCISE 18.3 [A/C:15] Complete the above exercise for all nine nodes. Add the shape functions (use a CAS and simplify) and verify whether their sum is unity.

EXERCISE 18.4 [A/C:20] Verify that the shape functions N_1^e and N_5^e of the eight-node serendipity quadrilateral discussed in §18.4.3 satisfy the interelement compatibility condition (C) stated in §18.1. Obtain all 8 shape functions and verify that their sum is unity.

EXERCISE 18.5 [C:15] Plot the shape functions N_1^e and N_5^e of the eight-node serendipity quadrilateral studied in §18.4.3 using the module `PlotQuadrilateralShapeFunction` listed in Cell 18.2.

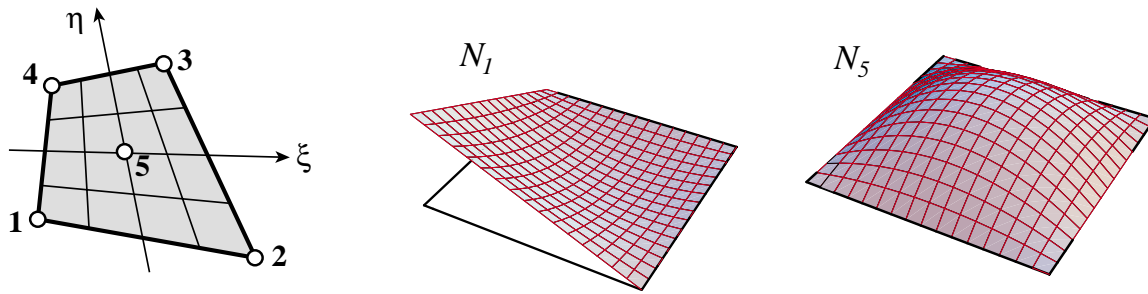


FIGURE E18.3. Five node quadrilateral element for Exercise 18.6.

EXERCISE 18.6 [A:15]. A five node quadrilateral element has the nodal configuration shown in Figure E18.3. Perspective views of N_1^e and N_5^e are shown in that Figure.² Find five shape functions N_i^e , $i = 1, 2, 3, 4, 5$ that satisfy compatibility, and also verify that their sum is unity.

Hint: develop $N_5(\xi, \eta)$ first for the 5-node quad using the line-product method; then the corner shape functions $\bar{N}_i(\xi, \eta)$ ($i = 1, 2, 3, 4$) for the 4-node quad (already given in the Notes); finally combine $N_i = \bar{N}_i + \alpha N_5$, determining α so that all N_i vanish at node 5. Check that $N_1 + N_2 + N_3 + N_4 + N_5 = 1$ identically.

EXERCISE 18.7 [A:15]. An eight-node “brick” finite element for three dimensional analysis has three isoparametric natural coordinates called ξ, η and μ . These coordinates vary from -1 at one face to $+1$ at the opposite face, as sketched in Figure E18.4.

Construct the (trilinear) shape function for node 1 (follow the node numbering of the figure). The equations of the brick faces are:

1485 : $\xi = -1$	2376 : $\xi = +1$
1265 : $\eta = -1$	4378 : $\eta = +1$
1234 : $\mu = -1$	5678 : $\mu = +1$

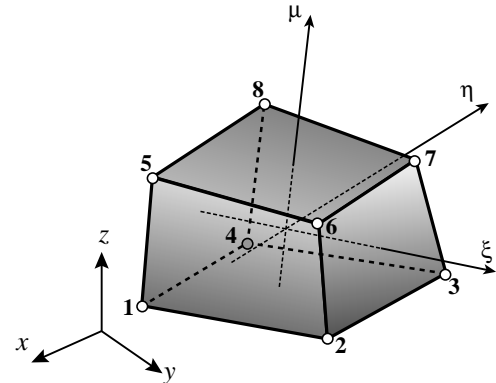


FIGURE E18.4. Eight-node isoparametric “brick” element for Exercise 18.7.

EXERCISE 18.8 [A:15]. Consider the 4-node transition triangular element of Figure 18.8(b). The shape function for node 1, $N_1 = \zeta_1 - 2\zeta_1\zeta_2$ was derived in §18.5.2 by the correction method. Show that the others are $N_2 = \zeta_2 - 2\zeta_1\zeta_2$, $N_3 = \zeta_3$ and $N_4 = 4\zeta_1\zeta_2$. Check that compatibility and completeness are verified.

EXERCISE 18.9 [A:15]. Construct the six shape functions for the 6-node transition quadrilateral element of Figure 18.8(c). Hint: for the corner nodes, use two corrections to the shape functions of the 4-node bilinear quadrilateral. Check compatibility and completeness. Partial result: $N_1 = \frac{1}{4}(1 - \xi)(1 - \eta) - \frac{1}{4}(1 - \xi^2)(1 - \eta)$.

EXERCISE 18.10 [A:20]. Consider a 5-node transition triangle in which midnode 6 on side 1–3 is missing. Show that $N_1^e = \zeta_1 - 2\zeta_1\zeta_2 - 2\zeta_2\zeta_3$. Can this be expressed as a line product like (18.1)?

² Although this N_1^e resembles the N_1^e of the 4-node quadrilateral depicted in Figure 18.4, they are not the same. That in Figure E18.3 must vanish at node 5 ($\xi = \eta = 0$). On the other hand, the N_1^e of Figure 18.4 takes the value $\frac{1}{4}$ there.

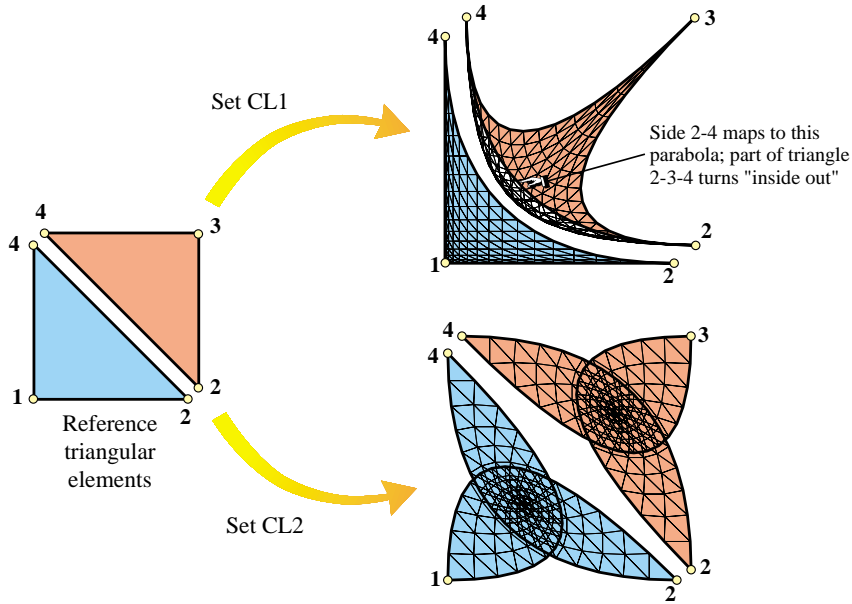


FIGURE E18.5. Mapping of reference triangles under sets (E18.1) and (E18.2). Triangles are slightly separated at the diagonal 2–4 for visualization convenience.

EXERCISE 18.11 [A:30]. The three-node linear triangle is known to be a poor performer for stress analysis. In an effort to improve it, Dr. I. M. Clueless proposes two sets of quadratic shape functions:

$$\text{CL1:} \quad N_1 = \zeta_1^2, \quad N_2 = \zeta_2^2, \quad N_3 = \zeta_3^2. \quad (\text{E18.1})$$

$$\text{CL2:} \quad N_1 = \zeta_1^2 + 2\zeta_2\zeta_3, \quad N_2 = \zeta_2^2 + 2\zeta_3\zeta_1, \quad N_3 = \zeta_3^2 + 2\zeta_1\zeta_2. \quad (\text{E18.2})$$

Dr. C. writes a learned paper claiming that both sets satisfy the interpolation condition, that set CL1 will work because it is conforming and that set CL2 will work because $N_1 + N_2 + N_3 = 1$. He provides no numerical examples. You get the paper for review. Show that the claims are false, and both sets are worthless. Hint: study §16.6 and Figure E18.5.

EXERCISE 18.12 [A:25]. Another way of constructing shape functions for “incomplete” elements is through kinematic multifreedom constraints (MFCs) applied to a “parent” element that contains the one to be derived. Suppose that the 9-node biquadratic quadrilateral is chosen as parent, with shape functions called N_i^P , $i = 1, \dots, 9$ given in §18.4.2. To construct the shape functions of the 8-node serendipity quadrilateral, the motions of node 9 are expressed in terms of the motions of the corner and midside nodes by the interpolation formulas

$$\begin{aligned} u_{x9} &= \alpha(u_{x1} + u_{x2} + u_{x3} + u_{x4}) + \beta(u_{x5} + u_{x6} + u_{x7} + u_{x8}), \\ u_{y9} &= \alpha(u_{y1} + u_{y2} + u_{y3} + u_{y4}) + \beta(u_{y5} + u_{y6} + u_{y7} + u_{y8}), \end{aligned} \quad (\text{E18.3})$$

where α and β are scalars to be determined. (In the terminology of Chapter 9, u_{x9} and u_{y9} are slaves while boundary DOFs are masters.) Show that the shape functions of the 8-node quadrilateral are then $N_i = N_i^P + \alpha N_9^P$ for $i = 1, \dots, 4$ and $N_i = N_i^P + \beta N_9^P$ for $i = 5, \dots, 8$. Furthermore, show that α and β can be determined by two conditions:

1. The unit sum condition: $\sum_{i=1}^8 N_i = 1$, leads to $4\alpha + 4\beta = 1$.
2. Exactness of displacement interpolation for ξ^2 and η^2 leads to $2\alpha + \beta = 0$.

Solve these two equations for α and β , and verify that the serendipity shape functions given in §18.4.3 result.

EXERCISE 18.13 [A:25] Construct the 16 shape functions of the bicubic quadrilateral.

19

FEM Convergence Requirements

TABLE OF CONTENTS

	Page
§19.1. Overview	19-3
§19.2. The Variational Index	19-3
§19.3. Consistency Requirements	19-4
§19.3.1. Completeness	19-4
§19.3.2. Compatibility	19-4
§19.3.3. Matching and Non-Matching Meshes	19-6
§19.4. Stability	19-7
§19.4.1. Rank Sufficiency	19-8
§19.4.2. Jacobian Positiveness	19-9
§19. Notes and Bibliography	19-11
§19. References	19-11
§19. Exercises	19-12

§19.1. Overview

Chapters 11 through 18 have discussed, in piecemeal fashion, requirements for shape functions of isoparametric elements. These are motivated by *convergence*: as the mesh is refined, the FEM solution should approach the analytical solution of the mathematical model.¹ This attribute is obviously necessary to instill confidence in FEM results from the standpoint of mathematics.

This Chapter provides unified information on convergence requirements. These requirements can be grouped into three:

Completeness. The elements must have enough *approximation power* to capture the analytical solution in the limit of a mesh refinement process. This intuitive statement is rendered more precise below.

Compatibility. The shape functions should provide *displacement continuity* between elements. Physically these insure that no material gaps appear as the elements deform. As the mesh is refined, such gaps would multiply and may absorb or release spurious energy.

Stability. The system of finite element equations must satisfy certain *well posedness* conditions that preclude nonphysical zero-energy modes in elements, as well as the absence of excessive element distortion.

Completeness and compatibility are two aspects of the so-called **consistency** condition between the discrete and mathematical models. A finite element model that passes both completeness and continuity requirements is called *consistent*. This is the FEM analog of the famous Lax-Wendroff theorem,² which says that consistency and stability imply convergence.

Remark 19.1. A deeper mathematical analysis done in more advanced courses shows that completeness is *necessary* for convergence whereas failure of the other requirements does not necessarily precludes it. There are, for example, FEM models in common use that do not satisfy compatibility. Furthermore, numerically unstable models may be used (with caution) in situations where that property is advantageous, as in the modeling of local singularities. Nonetheless, the satisfaction of the three criteria guarantees convergence and may therefore be regarded as a safe choice for the beginner user.

§19.2. The Variational Index

For the mathematical statement of the completeness and continuity conditions, the variational index alluded to in previous sections plays a fundamental role.

The FEM is based on the direct discretization of an energy functional $\Pi[u]$, where u (displacements for the elements considered in this book) is the primary variable, or (equivalently) the function to be varied. Let m be the highest spatial derivative order of u that appears in Π . This m is called the *variational index*.

¹ Of course FEM convergence does not guarantee the correctness of the mathematical model in capturing the physics. As discussed in Chapter 1, *model verification* against experiments is a different and far more difficult problem.

² Proven originally for classical finite difference discretizations in fluid mechanics. More precisely, it states that a numerical scheme for the scalar conservation law, $du/dt + df/dx = 0$ converges to a unique (weak) solution, if it is consistent, stable and conservative. There is no equivalent theorem for systems of conservation laws.

Example 19.1. In the bar problem discussed in Chapter 11,

$$\Pi[u] = \int_0^L \left(\frac{1}{2} u' E A u' - q u \right) dx. \quad (19.1)$$

The highest derivative of the displacement $u(x)$ is $u' = du/dx$, which is first order in the space coordinate x . Consequently $m = 1$. This is also the case on the plane stress problem studied in Chapter 14, because the strains are expressed in terms of first order derivatives of the displacements.

Example 19.2. In the plane beam problem discussed in Chapter 12,

$$\Pi[v] = \int_0^L \left(\frac{1}{2} v'' E I v'' - q v \right) dx. \quad (19.2)$$

The highest derivative of the transverse displacement is the curvature $\kappa = v'' = d^2v/dx^2$, which is of second order in the space coordinate x . Consequently $m = 2$.

§19.3. Consistency Requirements

Using the foregoing definition of variational index, we can proceed to state the two key requirements for finite element shape functions.

§19.3.1. Completeness

The element shape functions must represent exactly all polynomial terms of order $\leq m$ in the Cartesian coordinates. A set of shape functions that satisfies this condition is called m -complete.

Note that this requirement applies *at the element level* and involves *all* shape functions of the element.

Example 19.3. Suppose a displacement-based element is for a plane stress problem, in which $m = 1$. Then 1-completeness requires that the linear displacement field

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y, \quad u_y = \alpha_0 + \alpha_1 x + \alpha_2 y \quad (19.3)$$

be exactly represented for any value of the α coefficients. This is done by evaluating (19.3) at the nodes to form a displacement vector \mathbf{u}^e and then checking that $\mathbf{u} = \mathbf{N}^e \mathbf{u}^e$ recovers exactly (19.3). Section 16.6 presents the details of this calculation for an arbitrary isoparametric plane stress element. That analysis shows that completeness is satisfied if the *sum of the shape functions is unity* and the *element is compatible*.

Example 19.4. For the plane beam problem, in which $m = 2$, the quadratic transverse displacement

$$v = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \quad (19.4)$$

must be exactly represented over the element. This is easily verified in for the 2-node beam element developed in Chapter 13, because the assumed transverse displacement is a complete cubic polynomial in x . A complete cubic contains the quadratic (19.4) as special case.

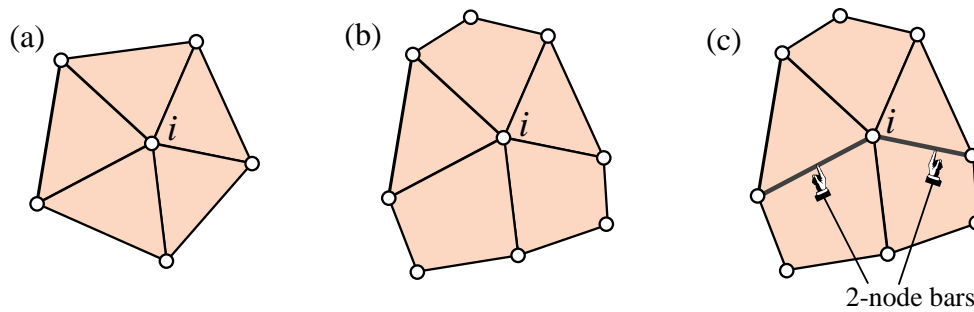


FIGURE 19.1. An element patch is the set of all elements attached to a patch node, labeled i . (a) illustrates a patch of triangles; (b) a mixture of triangles and quadrilaterals; (c) a mixture of triangles, quadrilaterals, and bars.

§19.3.2. Compatibility

To state this requirement succinctly, it is convenient to introduce the concept of *element patch*, or simply *patch*. This is the set of all elements attached to a given node, called the *patch node*. The definition is illustrated in Figure 19.1, which shows three different kind of patches attached to patch node i in a plane stress problem. The patch of Figure 19.1(a) contains only one type of element: 3-node linear triangles. The patch of Figure 19.1(b) mixes two plane stress element types: 3-node linear triangles and 4-node bilinear quadrilaterals. The patch of Figure 19.1(c) combines three element types: 3-node linear triangles, 4-node bilinear quadrilaterals, and 2-node bars.

We define a finite element *patch trial function* as the union of shape functions activated by setting a degree of freedom at the patch node to unity, while all other freedoms are zero.

A patch trial function “propagates” only over the patch, and is zero beyond it. This property follows from the local-support requirement stated in §18.1: a shape function for node i should vanish on all sides or faces that do not include i .

With the help of these definitions we can enunciate the compatibility requirement as follows.

Patch trial functions must be $C^{(m-1)}$ continuous between interconnected elements, and C^m piecewise differentiable inside each element.

If the variational index is $m = 1$, the patch trial functions must be C^0 continuous between elements, and C^1 inside elements.

A set of shape functions that satisfies the first requirement is called *conforming*. A conforming expansion that satisfies the second requirement is said to be of *finite energy*. Note that this condition applies at two levels: individual element, and element patch. An element endowed with conforming shape functions is said to be *conforming*. A conforming element that satisfies the finite energy requirement is said to be *compatible*.³

³ The FEM literature is a bit fuzzy as regards these terms. It seems better to leave the qualifier “conforming” to denote interelement compatibility; informally “an element that gets along with its neighbors.” The qualifier “compatible” is used in the stricter sense of conforming while possessing sufficient internal smoothness.

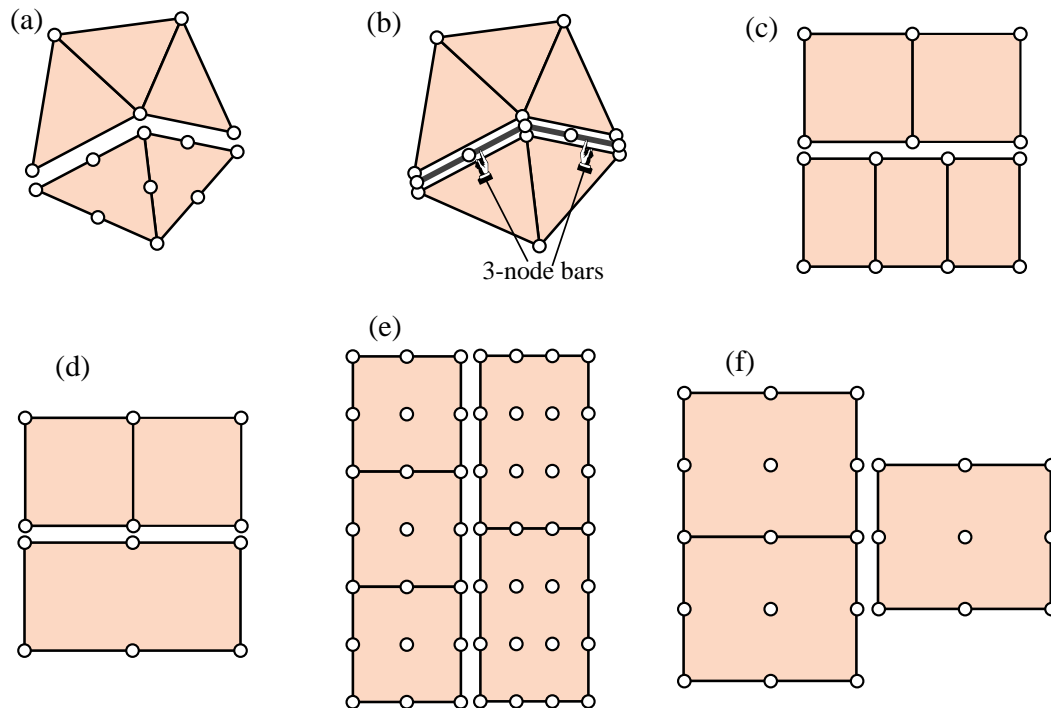


FIGURE 19.2. Examples of 2D non-matching meshes. Interelement boundaries that fail matching conditions are shown offset for visualization convenience. In (a,b,c) some nodes do not match. In (d,e,f) nodes and DOFs match but some sides do not, leading to violations of C^0 continuity.

Figures 19.1(b,c) illustrates the fact that one needs to check the possible connection of *matching elements* of different types and possibly different dimensionality.

§19.3.3. Matching and Non-Matching Meshes

As stated, compatibility refers to the *complete finite element mesh* because mesh trial functions are a combination of patch trial functions, which in turn are the union of element shape functions. This generality poses some logistical difficulties because the condition is necessarily mesh dependent. Compatibility can be checked at the *element level* by restricting attention to *matching meshes*. A matching mesh is one in which adjacent elements share sides, nodes and degrees of freedom, as in the patches shown in Figure 19.1.

For a matching mesh it is sufficient to restrict consideration first to a pair of adjacent elements, and then to the side shared by these elements. Suppose that the variation of a shape function *along that side* is controlled by k nodal values. Then a polynomial variation of order up to $k - 1$ in the natural coordinate(s) can be specified uniquely over the side. This is sufficient to verify interelement compatibility for $m = 1$, implying C^0 continuity, if the shape functions are polynomials.

This simplified criterion is the one used in previous Chapters. Specific 2D examples were given in Chapters 15 through 18.

Remark 19.2. If the variational index is $m = 2$ and the problem is multidimensional, as in the case of plates and shells, the check is far more involved and trickier because continuity of *normal derivatives along a side* is involved. This practically important scenario is examined in advanced FEM treatments. The case of non-polynomial shape functions is, on the other hand, of little practical interest.

A mesh that does not satisfy the matching criteria stated above is called a *nonmatching mesh*. Several two-dimensional examples are shown in Figure 19.2. As can be seen there is a wide range of possibilities: nonmatching nodes, matching nodes but different element types, etc. Figure 19.3 depicts a three-dimensional example, in which case even more variety can be expected.

Nonmatching meshes are the rule rather than the exception in contact and impact problems (which, being geometrically nonlinear, are outside the scope of this book). See Figure 19.4 illustrates what happens in a problem of slipping contact.

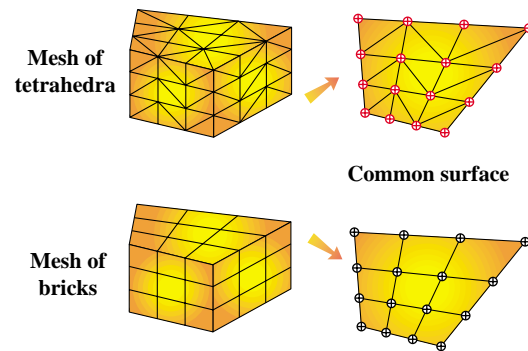


FIGURE 19.3. Example of a 3D non-matching mesh. Top portion discretized with tetrahedra, lower portion with bricks. Nodes and boundary-quadrant edges and DOFs match, but element types are different, leading to violation of C^0 continuity.

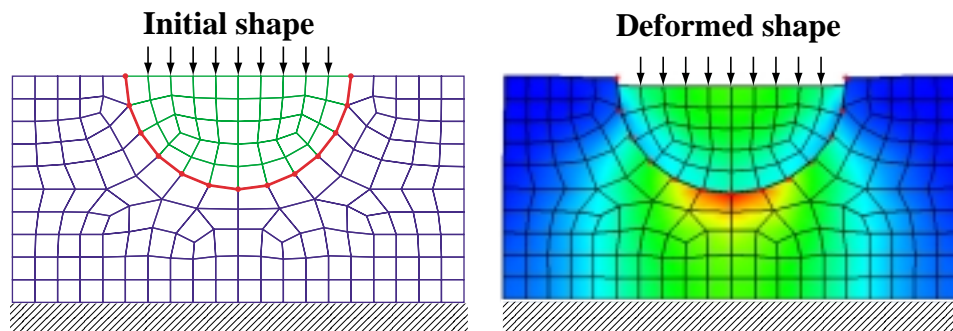


FIGURE 19.4. In contact and impact problems, matching meshes are the exception rather than the rule. Even if the meshes match at initial contact, slipping may produce a nonmatching mesh in the deformed configuration, as illustrated in the figure.

In multiphysics simulations nonmatching meshes are common, since they are often prepared separately for the different physical components, as illustrated in Figure 19.5.

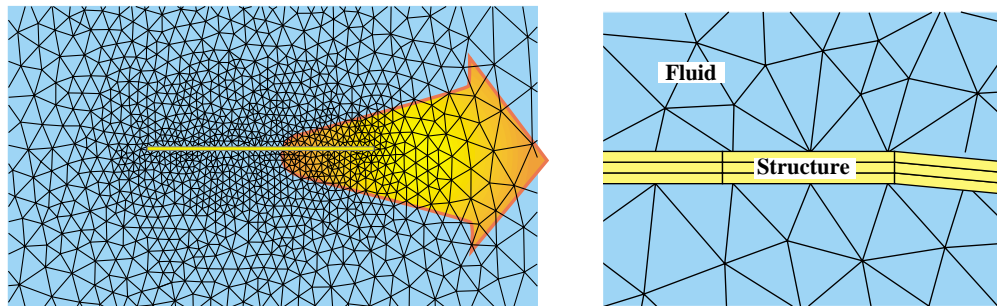


FIGURE 19.5. Nonmatching meshes are common in multiphysics problems, as in this example of fluid-structure interaction (FSI). Two-dimensional model to simulate flow around a thin plate. If the meshes are independently prepared node locations will not generally match.

§19.4. Stability

Stability may be informally characterized as ensuring that the finite element model enjoys the same solution uniqueness properties of the analytical solution of the mathematical model. For example, if the only motions that produce zero internal energy in the mathematical model are rigid body motions, the finite element model must inherit that property. Since FEM can handle arbitrary assemblies of elements, including individual elements, this property is required to hold at the element level.

In the present outline we are concerned with stability at the element level. Stability is not a property of shape functions *per se* but of the implementation of the element as well as its geometrical definition. It involves two subordinate requirements: rank sufficiency, and Jacobian positiveness. Of these, rank sufficiency is the most important one.

§19.4.1. Rank Sufficiency

The element stiffness matrix must not possess any zero-energy kinematic mode other than rigid body modes.

This can be mathematically expressed as follows. Let n_F be the number of element degrees of freedom, and n_R be the number of independent rigid body modes. Let r denote the rank of \mathbf{K}^e . The element is called *rank sufficient* if $r = n_F - n_R$ and *rank deficient* if $r < n_F - n_R$. In the latter case, the *rank deficiency* is defined by

$$d = (n_F - n_R) - r \quad (19.5)$$

If an isoparametric element is numerically integrated, let n_G be the number of Gauss points, while n_E denotes the order of the stress-strain matrix \mathbf{E} . Two additional assumptions are made:

- (i) The element shape functions satisfy completeness in the sense that the rigid body modes are exactly captured by them.
- (ii) Matrix \mathbf{E} is of full rank.

Then each Gauss point adds n_E to the rank of \mathbf{K}^e , up to a maximum of $n_F - n_R$. Hence the rank of \mathbf{K}^e will be

$$r = \min(n_F - n_R, n_E n_G) \quad (19.6)$$

To attain rank sufficiency, $n_E n_G$ must equal or exceed $n_F - n_R$:

$$\boxed{n_E n_G \geq n_F - n_R} \quad (19.7)$$

from which the appropriate Gauss integration rule can be selected.

In the plane stress problem, $n_E = 3$ because \mathbf{E} is a 3×3 matrix of elastic moduli; see equation (14.5)₂. Also $n_R = 3$. Consequently $r = \min(n_F - 3, 3n_G)$ and $3n_G \geq n_F - 3$.

Remark 19.3. The fact that each Gauss point adds $n_E n_G$ to the rank can be proven considering the following property. Let \mathbf{B} be a $n_E \times n_F$ rectangular real matrix with $\text{rank } r_B \leq n_E$, and \mathbf{E} an $n_E \times n_E$ positive-definite (p.d.) symmetric matrix. Then the rank of $\mathbf{B}^T \mathbf{E} \mathbf{B}$ is r_B . Proof: let $\mathbf{u} \neq \mathbf{0}$ be a non-null n_F -vector. If $\mathbf{B}^T \mathbf{E} \mathbf{B} \mathbf{u} = \mathbf{0}$ then $0 = \mathbf{u}^T \mathbf{B}^T \mathbf{E} \mathbf{B} \mathbf{u} = \|\mathbf{E}^{1/2} \mathbf{B} \mathbf{u}\|^2$. Therefore $\mathbf{B} \mathbf{u} = \mathbf{0}$. Identify now \mathbf{B} and \mathbf{E} with the strain-displacement and stress-strain (constitutive) matrix, respectively. In the plane stress case $n_E = 3$, $n_F = 2n > 3$ is the number of element freedoms. Thus \mathbf{B} has rank 3 and *a fortiori* $\mathbf{B}^T \mathbf{E} \mathbf{B}$ must also have rank 3 since \mathbf{E} is p.d. At each Gauss point i a contribution of $w_i \mathbf{B}^T \mathbf{E} \mathbf{B}$, which has rank 3 if $w_i > 0$, is added to \mathbf{K}^e . By a theorem of linear algebra, the rank of \mathbf{K}^e increases by 3 until it reaches $n_F - n_R$.

Table 19.1 Rank-sufficient Gauss Rules for Some Plane Stress Elements

Element	n	n_F	$n_F - 3$	Min n_G	Recommended rule
3-node triangle	3	6	3	1	centroid*
6-node triangle	6	12	9	3	3-point rules*
10-node triangle	10	20	17	6	6-point rule*
4-node quadrilateral	4	8	5	2	2×2
8-node quadrilateral	8	16	13	5	3×3
9-node quadrilateral	9	18	15	5	3×3
16-node quadrilateral	16	32	29	10	4×4

* These triangle integration rules are introduced in §24.2.

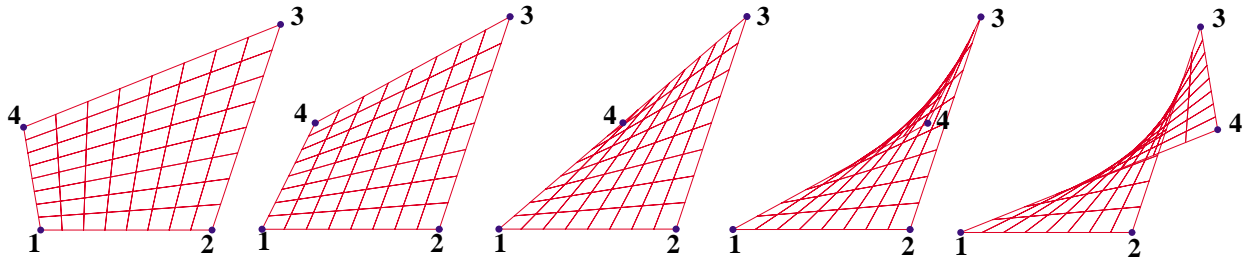


FIGURE 19.6. Effect of displacing node 4 of the four-node bilinear quadrilateral shown on the leftmost picture, to the right.

Example 19.5. Consider a plane stress 6-node quadratic triangle. Then $n_F = 2 \times 6 = 12$. To attain the proper rank of $12 - n_R = 12 - 3 = 9$, $n_G \geq 3$. A 3-point Gauss rule, such as the midpoint rule defined in §24.2, makes the element rank sufficient.

Example 19.6. Consider a plane stress 9-node biquadratic quadrilateral. Then $n_F = 2 \times 9 = 18$. To attain the proper rank of $18 - n_R = 18 - 3 = 15$, $n_G \geq 5$. The 2×2 product Gauss rule is insufficient because $n_G = 4$. Hence a 3×3 rule, which yields $n_G = 9$, is required to attain rank sufficiency.

Table 19.1 collects rank-sufficient Gauss integration rules for some widely used plane stress elements with n nodes and $n_F = 2n$ freedoms.

§19.4.2. Jacobian Positiveness

The geometry of the element should be such that the determinant $J = \det \mathbf{J}$ of the Jacobian matrix defined⁴ in §17.2, is positive everywhere. As illustrated in Equation (17.20), J characterizes the local metric of the element natural coordinates.

⁴ This definition applies to quadrilateral elements. The Jacobian determinant of an arbitrary triangular element is defined in §24.2.

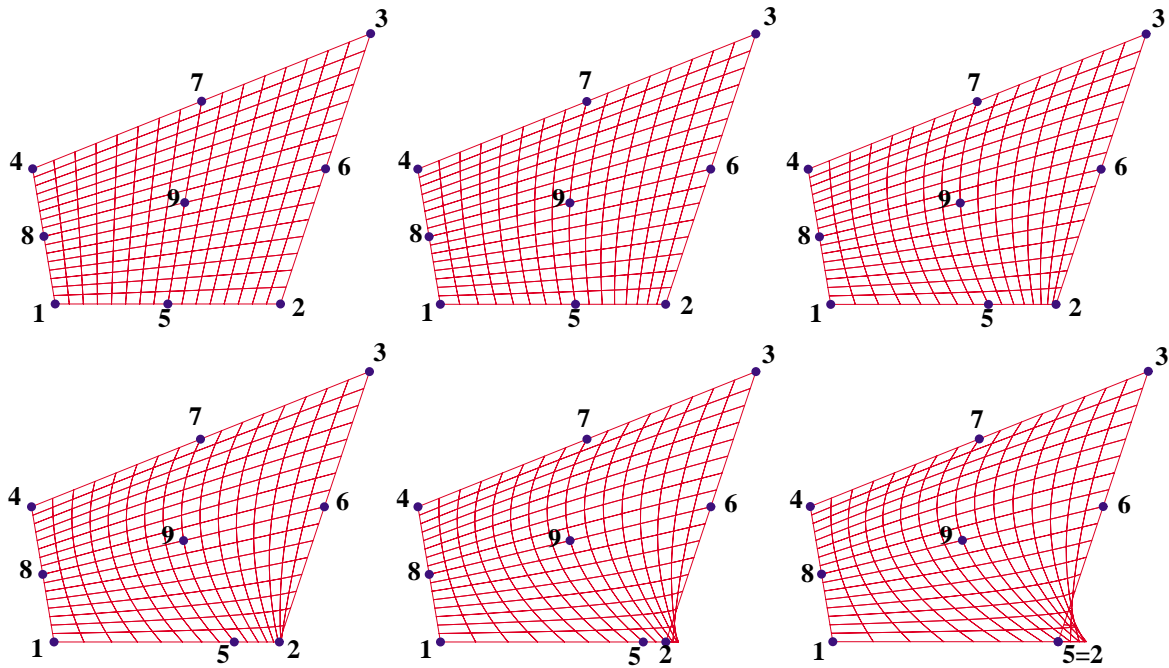


FIGURE 19.7. Effect of moving midpoint 5 of a 9-node biquadratic quadrilateral tangentially toward corner 2.

For a three-node triangle J is constant and in fact equal to $2A$. The requirement $J > 0$ is equivalent to saying that corner nodes must be positioned and numbered so that a positive area $A > 0$ results. This is called a *convexity condition*. It is easily checked by a finite element program.

But for 2D elements with more than 3 nodes distortions may render *portions* of the element metric negative. This is illustrated in Figure 19.6 for a 4-node quadrilateral in which node 4 is gradually moved to the right. The quadrilateral gradually morphs from a convex figure into a nonconvex one. The center figure is a triangle; note that the metric near node 4 is badly distorted (in fact $J = 0$ there) rendering the element unacceptable. This clearly contradicts the erroneous advice of some FE books, which state that quadrilaterals can be reduced to triangles as special cases, thereby rendering triangular elements unnecessary.

For higher order elements proper location of corner nodes is not enough. The non-corner nodes (midside, interior, etc.) must be placed sufficiently close to their natural locations (midpoints, centroids, etc.) to avoid violent local distortions. The effect of midpoint motions in quadratic elements is illustrated in Figures 19.7 and 19.8.

Figure 19.7 depicts the effect of moving midside node 5 tangentially in a 9-node quadrilateral element while keeping all other 8 nodes fixed. When the location of 5 reaches the quarter-point of side 1-2, the metric at corner 2 becomes singular in the sense that $J = 0$ there. Although this is disastrous in ordinary FE work, it has applications in the construction of special “crack” elements for linear fracture mechanics.

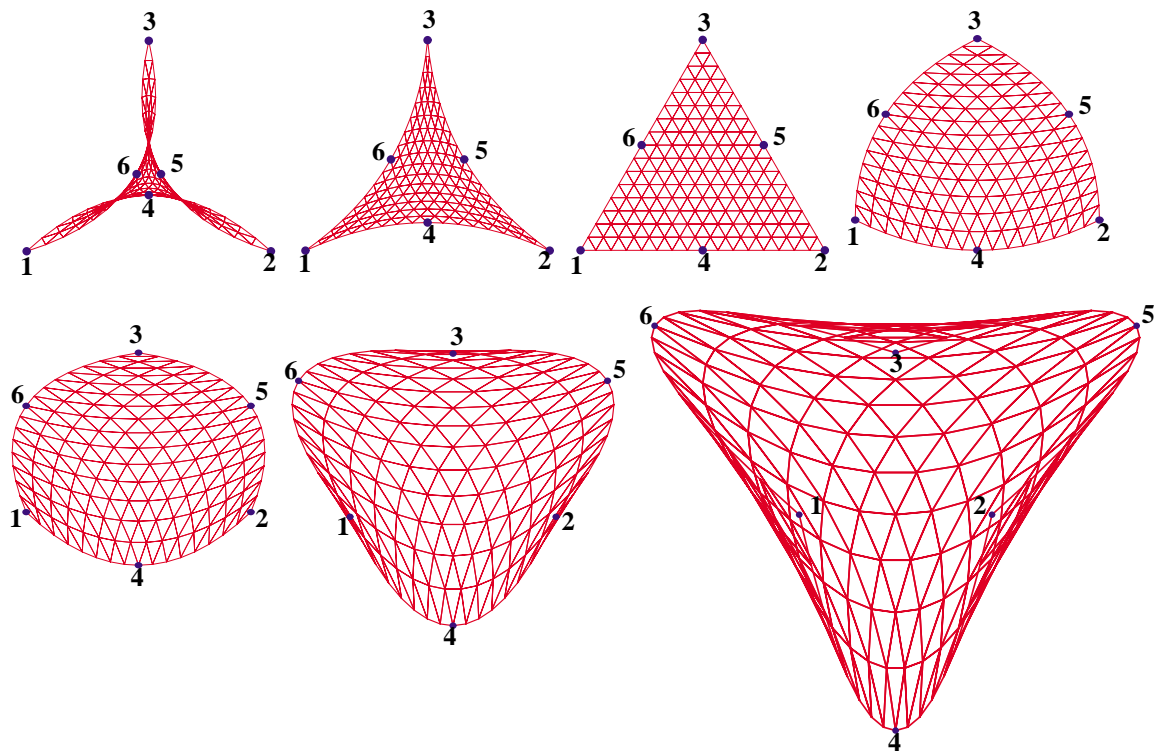


FIGURE 19.8. Effect of displacing midpoints 4, 5 and 6 of an equilateral 6-node triangle along the midpoint normals. Motion is inwards in first two top frames, outwards in the last four. In the lower leftmost picture nodes 1 through 6 lie on a circle.

Displacing midside nodes normally to the sides is comparatively more forgiving, as illustrated in Figure 19.8. This depicts a 6-node equilateral triangle in which midside nodes 4, 5 and 6 are moved inwards and outwards along the normals to the midpoint location. As shown in the lower left picture, the element may be even morphed into a “parabolic circle” (meaning that nodes 1 through 6 lie on a circle) without the metric breaking down.

Notes and Bibliography

The literature on the mathematics of finite element methods has grown exponentially since the monograph of Strang and Fix [360]. This is very readable but out of print. A more up-to-date exposition is the textbook by Szabo and Babuska [370]. The subjects collected in this Chapter tend to be dispersed in recent monographs and obscured by overuse of functional analysis.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 19

FEM Convergence Requirements

EXERCISE 19.1 [D:20] Explain why the two-dimensional meshes pictured in Figure 19.2(d,e,f) fail interelement compatibility although nodes and DOFs match.

EXERCISE 19.2 [A:20] The isoparametric definition of the straight 3-node bar element in its local system \bar{x} is

$$\begin{bmatrix} 1 \\ \bar{x} \\ \bar{v} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \\ \bar{u}_1 & \bar{u}_2 & \bar{u}_3 \end{bmatrix} \begin{bmatrix} N_1^e(\xi) \\ N_2^e(\xi) \\ N_3^e(\xi) \end{bmatrix}. \quad (\text{E19.1})$$

Here ξ is the isoparametric coordinate that takes the values -1 , 1 and 0 at nodes 1, 2 and 3, respectively, while N_1^e , N_2^e and N_3^e are the shape functions found in Exercise 16.3 and listed in (E16.2).

For simplicity, take $\bar{x}_1 = 0$, $\bar{x}_2 = L$, $\bar{x}_3 = \frac{1}{2}L + \alpha L$. Here L is the bar length and α a parameter that characterizes how far node 3 is away from the midpoint location $\bar{x} = \frac{1}{2}L$. Show that the minimum α 's (minimal in absolute value sense) for which $J = d\bar{x}/d\xi$ vanishes at a point in the element are $\pm 1/4$ (the quarter-points). Interpret this result as a singularity by showing that the axial strain becomes infinite at an end point. (This result has application in fracture mechanics modeling.)

EXERCISE 19.3 [A:15] Consider one dimensional bar-like elements with n nodes and 1 degree of freedom per node so $n_F = n$. The correct number of rigid body modes is 1. Each Gauss integration point adds 1 to the rank; that is $N_E = 1$. By applying (19.7), find the minimal rank-preserving Gauss integration rules with p points in the longitudinal direction if the number of node points is $n = 2, 3$ or 4 .

EXERCISE 19.4 [A:20] Consider three dimensional solid “brick” elements with n nodes and 3 degrees of freedom per node so $n_F = 3n$. The correct number of rigid body modes is 6. Each Gauss integration point adds 6 to the rank; that is, $N_E = 6$. By applying (19.7), find the minimal rank-preserving Gauss integration rules with p points in each direction (that is, $1 \times 1 \times 1$, $2 \times 2 \times 2$, etc) if the number of node points is $n = 8, 20, 27$, or 64 . Partial answer: for $n = 27$ the minimal rank preserving rule is $3 \times 3 \times 3$.

EXERCISE 19.5 [A/C:35] (Requires use of a CAS help to be tractable). Repeat Exercise 19.2 for a 9-node plane stress element. The element is initially a perfect square, nodes 5,6,7,8 are at the midpoint of the sides 1–2, 2–3, 3–4 and 4–1, respectively, and 9 at the center of the square. Displace 5 tangentially towards 2 until the Jacobian determinant at 2 vanishes. This result is important in the construction of “singular elements” for fracture mechanics.

EXERCISE 19.6 [A/C:35] Repeat Exercise 19.5 but moving node 5 along the normal to the side. Discuss the range of motion for which $\det \mathbf{J} > 0$ within the element.

EXERCISE 19.7 [A:20] Discuss whether the deVeubeke triangle presented in Chapter 15 satisfies completeness and interelement-compatibility requirements.

20

Implementation of One-Dimensional Elements

TABLE OF CONTENTS

	Page
§20.1. The Plane Bar Element	20-3
§20.1.1. Stiffness Matrix	20-3
§20.1.2. Stiffness Module	20-3
§20.1.3. Testing the Plane Bar Module	20-4
§20.2. The Space Bar Element	20-5
§20.2.1. Stiffness Matrix	20-6
§20.2.2. Stiffness Module	20-6
§20.2.3. Testing the Space Bar Module	20-7
§20.3. The Plane Beam-Column Element	20-8
§20.3.1. Stiffness Matrix	20-8
§20.3.2. Stiffness Module	20-9
§20.3.3. Testing the Plane Beam-Column Module	20-9
§20.4. *Plane Beam With Offset Nodes	20-11
§20.4.1. Plate Reinforced With Edge Beams	20-11
§20.4.2. Rigid Link Transformation	20-12
§20.4.3. Direct Fabrication of Offset Beam Stiffness	20-13
§20.5. Layered Beam Configurations	20-14
§20.5.1. Layered Beam With No Interlayer Slip	20-15
§20.5.2. Beam Stacks Allowing Interlayer Slip	20-17
§20.6. The Space Beam Element	20-18
§20.6.1. Stiffness Matrix	20-18
§20.6.2. Stiffness Module	20-20
§20. Notes and Bibliography	20-20
§20. Exercises	20-21

This Chapter begins Part III of the course. This Part deals with the computer implementation of the Finite Element Method for static analysis. It is organized in “bottom up” fashion. It begins with simple topics, such as programming of bar and beam elements, and gradually builds up toward more complex models and calculations.

Specific examples of this Chapter illustrate the programming of one-dimensional elements: bars and beams, using *Mathematica* as implementation language.

§20.1. The Plane Bar Element

The two-node, prismatic, two-dimensional bar element was studied in Chapters 2-3 for modeling plane trusses. It is reproduced in Figure 20.1 for convenience. It has two nodes and four degrees of freedom. The element node displacements and conjugate forces are

$$\mathbf{u}^e = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \end{bmatrix}, \quad \mathbf{f}^e = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \end{bmatrix}. \quad (20.1)$$

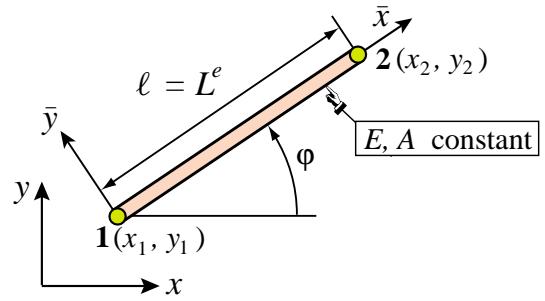


FIGURE 20.1. Plane bar element.

The element geometry is described by the coordinates $\{x_i, y_i\}$, $i = 1, 2$ of the two end nodes. For stiffness computations, the only material and fabrication properties required are the modulus of elasticity $E = E^e$ and the cross section area $A = A^e$, respectively. Both are taken to be constant over the element.

§20.1.1. Stiffness Matrix

The element stiffness matrix in global $\{x, y\}$ coordinates is given by the explicit expression derived in §3.1:

$$\mathbf{K}^e = \frac{EA}{\ell} \begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix} = \frac{EA}{\ell^3} \begin{bmatrix} x_{21}x_{21} & x_{21}y_{21} & -x_{21}x_{21} & -x_{21}y_{21} \\ x_{21}y_{21} & y_{21}y_{21} & -x_{21}y_{21} & -y_{21}y_{21} \\ -x_{21}x_{21} & -x_{21}y_{21} & x_{21}x_{21} & x_{21}y_{21} \\ -x_{21}y_{21} & -y_{21}y_{21} & x_{21}y_{21} & y_{21}y_{21} \end{bmatrix}. \quad (20.2)$$

Here $c = \cos \varphi = x_{21}/\ell$, $s = \sin \varphi = y_{21}/\ell$, in which $x_{21} = x_2 - x_1$, $y_{21} = y_2 - y_1$, $\ell = \sqrt{x_{21}^2 + y_{21}^2}$, and φ is the angle formed by \bar{x} and x , measured from x positive counterclockwise (see Figure 20.1). The second expression in (20.2) is preferable in a computer algebra system because it enhances simplification possibilities when doing symbolic work, and is the one actually implemented in the module described below.

§20.1.2. Stiffness Module

The computation of the stiffness matrix \mathbf{K}^e of the two-node, prismatic plane bar element is done by *Mathematica* module `PlaneBar2Stiffness`. This is listed in Figure 20.2. The module is invoked as

$$\text{Ke} = \text{PlaneBar2Stiffness}[\text{ncoor}, \text{Em}, \text{A}, \text{options}] \quad (20.3)$$

```

PlaneBar2Stiffness[ncoor_,Em_,A_,options_]:= Module[
{ x1,x2,y1,y2,x21,y21,EA,numer,L,LL,LLL,Ke},
{{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
EA=Em*A; {numer}=options; LL=x21^2+y21^2; L=Sqrt[LL];
If [numer,{x21,y21,EA,LL,L}=N[{x21,y21,EA,LL,L}]];
If [!numer, L=PowerExpand[L]]; LLL=Simplify[LL*L];
Ke=(Em*A/LLL)*{{ x21*x21, x21*y21,-x21*x21,-x21*y21},
{ y21*x21, y21*y21,-y21*x21,-y21*y21},
{ -x21*x21,-x21*y21, x21*x21, x21*y21},
{ -y21*x21,-y21*y21, y21*x21, y21*y21}};

Return[Ke]];

```

FIGURE 20.2. *Mathematica* stiffness module for a two-node, prismatic plane bar element.

The arguments are

- ncoor Node coordinates of element arranged as $\{\{x1,y1\},\{x2,y2\}\}$.
- Em Elastic modulus.
- A Cross section area.
- options A list of processing options. For this element is has only one entry: {numer}. This is a logical flag with the value True or False. If True the computations are carried out in floating-point arithmetic. If False symbolic processing is assumed.

The module returns the 4×4 element stiffness matrix as function value.

```

ClearAll[A,Em,L];
ncoor={{0,0},{30,40}}; Em=1000; A=5;
Ke= PlaneBar2Stiffness[ncoor,Em,A,{True}];
Print["Numerical Elem Stiff Matrix: "];
Print[Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]]]];
Print["Symmetry check=",Simplify[Chop[Transpose[Ke]-Ke]]];

```

Numerical Elem Stiff Matrix:

$$\begin{pmatrix} 36. & 48. & -36. & -48. \\ 48. & 64. & -48. & -64. \\ -36. & -48. & 36. & 48. \\ -48. & -64. & 48. & 64. \end{pmatrix}$$

Eigenvalues of Ke = {200., 0, 0, 0}

Symmetry check={ {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0}, {0, 0, 0, 0} }

FIGURE 20.3. Test of plane bar stiffness module with numerical inputs.

§20.1.3. Testing the Plane Bar Module

The modules are tested by the scripts listed in Figures 20.3 and 20.4. The script shown on the top of Figure 20.3 tests a numerically defined element with end nodes located at (0, 0) and (30, 40), with $E = 1000$, $A = 5$, and numer set to True. Executing the script produces the results listed in the bottom of that figure.

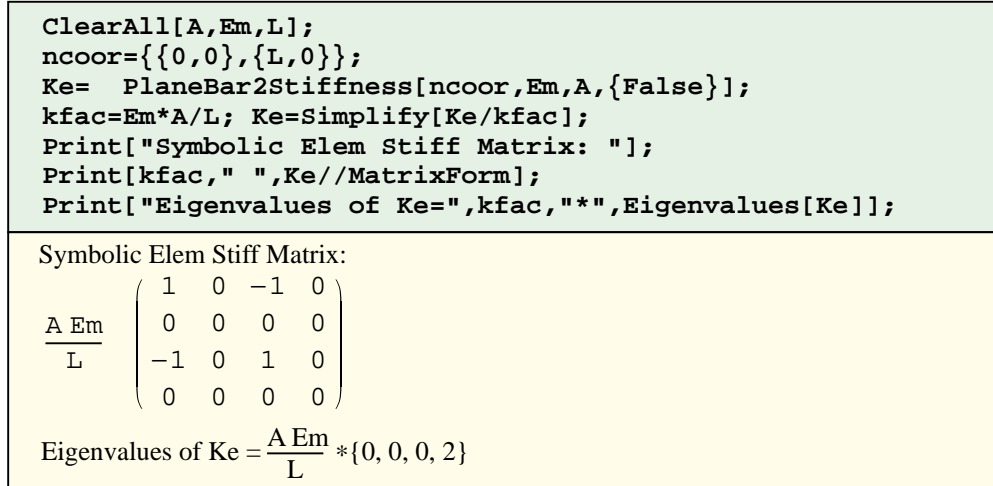


FIGURE 20.4. Test of plane bar stiffness module with symbolic inputs.

On return from `PlaneBar2Stiffness`, the stiffness matrix returned in \mathbf{K}_e is printed. Its four eigenvalues are computed and printed. As expected three eigenvalues, which correspond to the three independent rigid body motions of the element, are zero. The remaining eigenvalue is positive and equal to $E A / \ell$. The symmetry of \mathbf{K}_e is checked by printing $(\mathbf{K}^e)^T - \mathbf{K}^e$ upon simplification and chopping.

The script of Figure 20.4 tests a symbolically defined bar element with end nodes located at $(0, 0)$ and $(L, 0)$, which is aligned with the x axis. Properties E and A are kept symbolic. Executing the script shown in the top of Figure 20.4 produces the results shown in the bottom of that figure. One thing to be noticed is the use of the stiffness scaling factor $E A / \ell$, called `kfac` in the script. This is a symbolic quantity that can be extracted as factor of matrix \mathbf{K}^e . The effect is to clean up matrix and vector output, as can be observed in the printed results.

§20.2. The Space Bar Element

To show how the previous implementation extends easily to three dimensions, this section describes the implementation of the space bar element.

The two-node, prismatic, space bar element is pictured in Figure 20.5. The element has two nodes and six degrees of freedom. The element node displacements and conjugate forces are arranged as

$$\mathbf{u}^e = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{z1} \\ u_{x2} \\ u_{y2} \\ u_{z2} \end{bmatrix}, \quad \mathbf{f}^e = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{z1} \\ f_{x2} \\ f_{y2} \\ f_{z2} \end{bmatrix}. \quad (20.4)$$

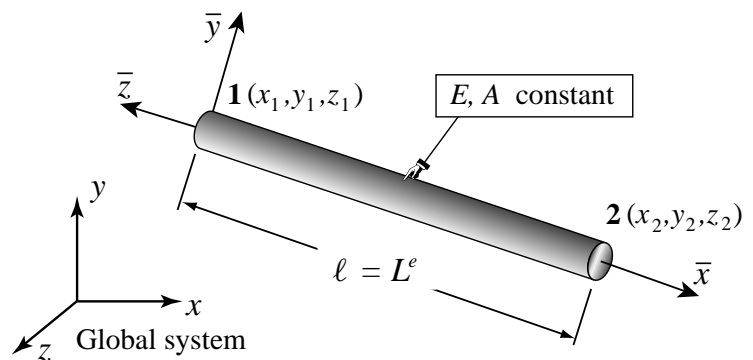


FIGURE 20.5. The space (3D) bar element.

```

SpaceBar2Stiffness[ncoor_, Em_, A_, options_] := Module[
  {x1, x2, y1, y2, z1, z2, x21, y21, z21, EA, numer, L, LL, LLL, Ke},
  {{x1, y1, z1}, {x2, y2, z2}} = ncoor; {x21, y21, z21} = {x2 - x1, y2 - y1, z2 - z1};
  EA = Em * A; {numer} = options; LL = x21^2 + y21^2 + z21^2; L = Sqrt[LL];
  If [numer, {x21, y21, z21, EA, LL, L} = N[{x21, y21, z21, EA, LL, L}]];
  If [!numer, L = PowerExpand[L]]; LLL = Simplify[LL * L];
  Ke = (Em * A / LLL) *
    { { x21*x21, x21*y21, x21*z21, -x21*x21, -x21*y21, -x21*z21},
      { y21*x21, y21*y21, y21*z21, -y21*x21, -y21*y21, -y21*z21},
      { z21*x21, z21*y21, z21*z21, -z21*x21, -z21*y21, -z21*z21},
      { -x21*x21, -x21*y21, -x21*z21, x21*x21, x21*y21, x21*z21},
      { -y21*x21, -y21*y21, -y21*z21, y21*x21, y21*y21, y21*z21},
      { -z21*x21, -z21*y21, -z21*z21, z21*x21, z21*y21, z21*z21} };
  Return[Ke];
];

```

FIGURE 20.6. Module to form the stiffness of the space (3D) bar element.

The element geometry is described by the coordinates $\{x_i, y_i, z_i\}$, $i = 1, 2$ of the two end nodes. As in the case of the plane bar, the two properties required for the stiffness computations are the modulus of elasticity E and the cross section area A . Both are assumed to be constant over the element.

§20.2.1. Stiffness Matrix

For the space bar element, introduce the notation $x_{21} = x_2 - x_1$, $y_{21} = y_2 - y_1$, $z_{21} = z_2 - z_1$ and $\ell = \sqrt{x_{21}^2 + y_{21}^2 + z_{21}^2}$. It can be shown¹ that the element stiffness matrix in global coordinates is given by

$$\mathbf{K}^e = \frac{E^e A^e}{\ell^3} \begin{bmatrix} x_{21}x_{21} & x_{21}y_{21} & x_{21}z_{21} & -x_{21}x_{21} & -x_{21}y_{21} & -x_{21}z_{21} \\ x_{21}y_{21} & y_{21}y_{21} & y_{21}z_{21} & -x_{21}y_{21} & -y_{21}y_{21} & -y_{21}z_{21} \\ x_{21}z_{21} & y_{21}z_{21} & z_{21}z_{21} & -x_{21}z_{21} & -y_{21}z_{21} & -z_{21}z_{21} \\ -x_{21}x_{21} & -x_{21}y_{21} & -x_{21}z_{21} & x_{21}x_{21} & x_{21}y_{21} & x_{21}z_{21} \\ -x_{21}y_{21} & -y_{21}y_{21} & -y_{21}z_{21} & x_{21}y_{21} & y_{21}y_{21} & y_{21}z_{21} \\ -x_{21}z_{21} & -y_{21}z_{21} & -z_{21}z_{21} & x_{21}z_{21} & y_{21}z_{21} & z_{21}z_{21} \end{bmatrix}. \quad (20.5)$$

This matrix expression in terms of coordinate differences is useful in symbolic work, because it enhances simplification possibilities.

§20.2.2. Stiffness Module

The computation of the stiffness matrix \mathbf{K}^e of the two-node, prismatic space bar element, is done by *Mathematica* module *SpaceBar2Stiffness*. This is listed in Figure 20.6. The module is invoked as

$$\mathbf{K}^e = \text{SpaceBar2Stiffness}[\text{ncoor}, \text{Em}, \text{A}, \text{options}] \quad (20.6)$$

The arguments are

ncoor Node coordinates of element arranged as $\{\{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}\}$.

¹ The derivation was the subject of Exercise 6.10.

```

ClearAll[A,Em];
ncoor={{0,0,0},{2,3,6}}; Em=343; A=10;
Ke= SpaceBar2Stiffness[ncoor,Em,A,{True}];
Print["Numerical Elem Stiff Matrix: "];
Print[Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[Ke]]];

```

Numerical Elem Stiff Matrix:

$$\begin{pmatrix} 40. & 60. & 120. & -40. & -60. & -120. \\ 60. & 90. & 180. & -60. & -90. & -180. \\ 120. & 180. & 360. & -120. & -180. & -360. \\ -40. & -60. & -120. & 40. & 60. & 120. \\ -60. & -90. & -180. & 60. & 90. & 180. \\ -120. & -180. & -360. & 120. & 180. & 360. \end{pmatrix}$$

Eigenvalues of Ke = {980., 0, 0, 0, 0, 0}

FIGURE 20.7. Testing the space bar stiffness module with numerical inputs.

Em Elastic modulus.

A Cross section area.

options A list of processing options. For this element is has only one entry: {numer}. This is a logical flag with the value True or False. If True the computations are carried out in floating-point arithmetic. If False symbolic processing is assumed.

The module returns the 6×6 element stiffness matrix as function value.

```

ClearAll[A,Em,L];
ncoor={{0,0,0},{L,2*L,2*L}}/3;
Ke= SpaceBar2Stiffness[ncoor,Em,A,{False}];
kfac=Em*A/(9*L); Ke=Simplify[Ke/kfac];
Print["Symbolic Elem Stiff Matrix: "];
Print[kfac," ",Ke//MatrixForm];
Print["Eigenvalues of Ke=",kfac,"*",Eigenvalues[Ke]];

```

Symbolic Elem Stiff Matrix:

$$\frac{A \text{ Em}}{9 L} \begin{pmatrix} 1 & 2 & 2 & -1 & -2 & -2 \\ 2 & 4 & 4 & -2 & -4 & -4 \\ 2 & 4 & 4 & -2 & -4 & -4 \\ -1 & -2 & -2 & 1 & 2 & 2 \\ -2 & -4 & -4 & 2 & 4 & 4 \\ -2 & -4 & -4 & 2 & 4 & 4 \end{pmatrix}$$

Eigenvalues of Ke = $\frac{A \text{ Em}}{9 L} * \{0, 0, 0, 0, 0, 18\}$

FIGURE 20.8. Testing the space bar stiffness module with symbolic inputs.

§20.2.3. Testing the Space Bar Module

The modules are tested by the scripts listed in Figures 20.7 and 20.8. As these are similar to previous tests done on the plane bar they need not be described in detail.

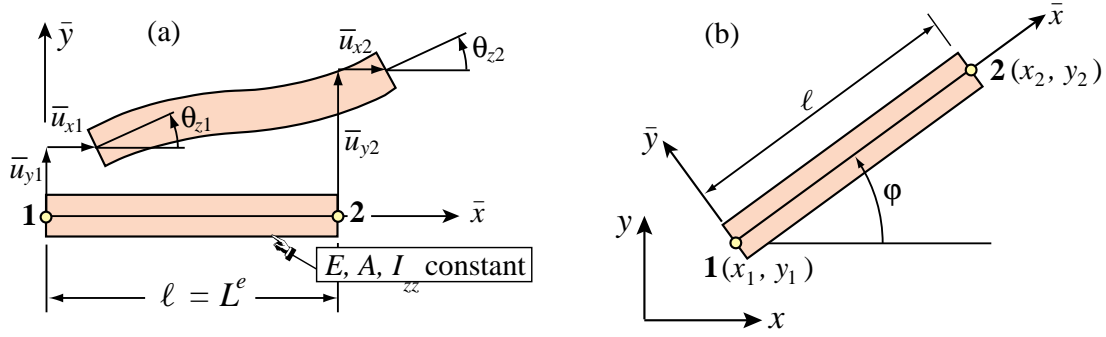


FIGURE 20.9. Plane beam-column element: (a) in its local system; (b) in the global system.

The script of Figure 20.7 tests a numerically defined space bar with end nodes located at $(0, 0, 0)$ and $(30, 40, 0)$, with $E = 1000$, $A = 5$ and `numer` set to `True`. Executing the script produces the results listed in the bottom of that Figure.

The script of Figure 20.8 tests a symbolically defined bar element with end nodes located at $(0, 0, 0)$ and $(L, 2L, 2L)/3$, which has length L and is not aligned with the x axis. The element properties E and A are kept symbolic. Executing the script produces the results shown in the bottom of that Figure. Note the use of a stiffness factor `kfac` of $EA/(9\ell)$ to get cleaner printouts.

§20.3. The Plane Beam-Column Element

Beam-column elements model structural members that resist both axial and bending actions. This is the case in skeletal structures such as frameworks which are common in steel and reinforced-concrete building construction. A plane beam-column element is a combination of a plane bar (such as that considered in §20.1), and a plane beam.

We consider a beam-column element in its local system (\bar{x}, \bar{y}) as shown in Figure 20.9(a), and then in the global system (x, y) as shown in Figure 20.9(b). The six degrees of freedom and conjugate node forces of the elements are:

$$\bar{\mathbf{u}}^e = \begin{bmatrix} \bar{u}_{x1} \\ \bar{u}_{y1} \\ \theta_{z1} \\ \bar{u}_{x2} \\ \bar{u}_{y2} \\ \theta_{z2} \end{bmatrix}, \quad \bar{\mathbf{f}}^e = \begin{bmatrix} \bar{f}_{x1} \\ \bar{f}_{y1} \\ m_{z1} \\ \bar{u}_{x2} \\ \bar{u}_{y2} \\ m_{z2} \end{bmatrix}, \quad \mathbf{u}^e = \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_{z1} \\ u_{x2} \\ u_{y2} \\ \theta_{z2} \end{bmatrix}, \quad \mathbf{f}^e = \begin{bmatrix} f_{x1} \\ f_{y1} \\ m_{z1} \\ f_{x2} \\ f_{y2} \\ m_{z2} \end{bmatrix}. \quad (20.7)$$

The rotation angles θ and the nodal moments m are the same in the local and the global systems because they are about the z axis, which does not change in passing from local to global.

The element geometry is described by the coordinates $\{x_i, y_i\}$, $i = 1, 2$ of the two end nodes. The element length is $\ell = L^e$. Properties involved in the stiffness calculations are: the modulus of elasticity E , the cross section area A and the moment of inertia $I = I_{zz}$ about the neutral axis. All properties are taken to be constant over the element.

§20.3.1. Stiffness Matrix

To obtain the plane beam-column stiffness in the local system we simply add the stiffness matrices derived in Chapters 11 and 12, respectively, to get

$$\bar{\mathbf{K}}^e = \frac{EA}{\ell} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & & 0 & 0 \\ \text{symm} & & & & & 0 \end{bmatrix} + \frac{EI}{\ell^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & 12 & 6\ell & 0 & -12 & 6\ell \\ & & 4\ell^2 & 0 & -6\ell & 2\ell^2 \\ & & & 0 & 0 & 0 \\ & & & & 12 & -6\ell \\ \text{symm} & & & & & 4\ell^2 \end{bmatrix} \quad (20.8)$$

The two matrices on the right of (20.8) come from the bar stiffness (12.22) and the Bernoulli-Euler bending stiffness (13.20), respectively. Before adding them, rows and columns have been rearranged in accordance with the nodal freedoms (20.7).

The displacement transformation matrix between local and global systems is

$$\bar{\mathbf{u}}^e = \begin{bmatrix} \bar{u}_{x1} \\ \bar{u}_{y1} \\ \theta_{z1} \\ u_{x2} \\ \bar{u}_{y2} \\ \theta_{z2} \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & s & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_{z1} \\ u_{x2} \\ u_{y2} \\ \theta_{z2} \end{bmatrix} = \mathbf{T} \mathbf{u}^e, \quad (20.9)$$

where $c = \cos \varphi = (x_2 - x_1)/\ell$, $s = \sin \varphi = (y_2 - y_1)/\ell$, and φ is the angle between \bar{x} and x , measured positive-counterclockwise from x ; see Figure 20.9. The stiffness matrix in the global system is obtained through the congruent transformation

$$\mathbf{K}^e = \mathbf{T}^T \bar{\mathbf{K}}^e \mathbf{T}. \quad (20.10)$$

Explicit expressions of the entries of \mathbf{K}^e are messy. Unlike the bar, it is better to let the program do the transformation.

§20.3.2. Stiffness Module

The computation of the stiffness matrix \mathbf{K}^e of the two-node, prismatic plane beam-column element is done by *Mathematica* module `PlaneBeamColumn2Stiffness`. This is listed in Figure 20.10. The module is invoked as

$$\mathbf{K}^e = \text{PlaneBeamColumn2Stiffness}[\text{ncoor}, \text{Em}, \{\text{A}, \text{Izz}\}, \text{options}] \quad (20.11)$$

The arguments are

- `ncoor` Node coordinates of element arranged as $\{\{x_1, y_1\}, \{x_2, y_2\}\}$.
- `Em` Elastic modulus.
- `A` Cross section area.
- `Izz` Moment of inertia of cross section area respect to axis z .
- `options` A list of processing options. For this element is has only one entry: `{numer}`. This is a logical flag with the value `True` or `False`. If `True` the computations are carried out in floating-point arithmetic. If `False` symbolic processing is assumed.

The module returns the 6×6 element stiffness matrix as function value.

```

PlaneBeamColumn2Stiffness[ncoor_,Em_,{A_,Izz_},options_]:=Module[
{
x1,x2,y1,y2,x21,y21,EA,EI,numer,L,LL,LLL,Te,Kebar,Ke},
{
{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
EA=Em*A; EI=Em*Izz; {numer}=options;
LL=Simplify[x21^2+y21^2]; L=Sqrt[LL];
If [numer,{x21,y21,EA,EI,LL,L}=N[{x21,y21,EA,EI,LL,L}]];
If [!numer, L=PowerExpand[L]]; LLL=Simplify[LL*L];
Kebar= (EA/L)*{
{ 1,0,0,-1,0,0},{0,0,0,0,0,0},{0,0,0,0,0,0},
{-1,0,0, 1,0,0},{0,0,0,0,0,0},{0,0,0,0,0,0}} +
(2*EI/LLL)*{
{ 0,0,0,0,0,0},{0, 6, 3*L,0,-6, 3*L},{0,3*L,2*LL,0,-3*L, LL},
{ 0,0,0,0,0,0},{0,-6,-3*L,0, 6,-3*L},{0,3*L, LL,0,-3*L,2*LL}};
Te={{x21,y21,0,0,0,0}/L,{-y21,x21,0,0,0,0}/L,{0,0,1,0,0,0},
{0,0,0,x21,y21,0}/L,{0,0,0,-y21,x21,0}/L,{0,0,0,0,0,1}};
Ke=Transpose[Te].Kebar.Te;
Return[Ke] ];

```

FIGURE 20.10. *Mathematica* module to form the stiffness matrix of a two-node, prismatic plane beam-column element.

```

ClearAll[L,Em,A,Izz];
ncoor={{0,0},{3,4}}; Em=100; A=125; Izz=250;
Ke= PlaneBeamColumn2Stiffness[ncoor,Em,{A,Izz},{True}];
Print["Numerical Elem Stiff Matrix: "];
Print[Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[Ke]]];

```

Numerical Elem Stiff Matrix:

$$\begin{pmatrix} 2436. & 48. & -4800. & -2436. & -48. & -4800. \\ 48. & 2464. & 3600. & -48. & -2464. & 3600. \\ -4800. & 3600. & 20000. & 4800. & -3600. & 10000. \\ -2436. & -48. & 4800. & 2436. & 48. & 4800. \\ -48. & -2464. & -3600. & 48. & 2464. & -3600. \\ -4800. & 3600. & 10000. & 4800. & -3600. & 20000. \end{pmatrix}$$

Eigenvalues of Ke = {34800., 10000., 5000., 0, 0, 0}

FIGURE 20.11. Test of two-node plane beam-column element with numeric inputs.

§20.3.3. Testing the Plane Beam-Column Module

The beam-column stiffness are tested by the scripts shown in Figures 20.11 and 20.12.

The script at the top of Figure 20.11 tests a numerically defined element of length $\ell = 5$ with end nodes located at $(0, 0)$ and $(3, 4)$, respectively, with $E = 100$, $A = 125$ and $I_{zz} = 250$. The output is shown at the bottom of that figure. The stiffness matrix returned in Ke is printed. Its six eigenvalues are computed and printed. As expected three eigenvalues, which correspond to the three independent rigid body motions of the element, are zero. The remaining three eigenvalues are positive.

The script at the top of Figure 20.12 tests a plane beam-column of length L with end nodes at $(0, 0)$ and $(3L/5, 4L/5)$. The properties E , A and I_{zz} are kept in symbolic form. The output is shown at the bottom of that figure. The printed matrix looks complicated because bar and beam coupling

occurs when the element is not aligned with the global axes. The eigenvalues are obtained in closed symbolic form, and their simplicity provides a good check that the transformation matrix (20.9) is orthogonal. Three eigenvalues are exactly zero; one is associated with the axial (bar) stiffness and two with the flexural (beam) stiffness.

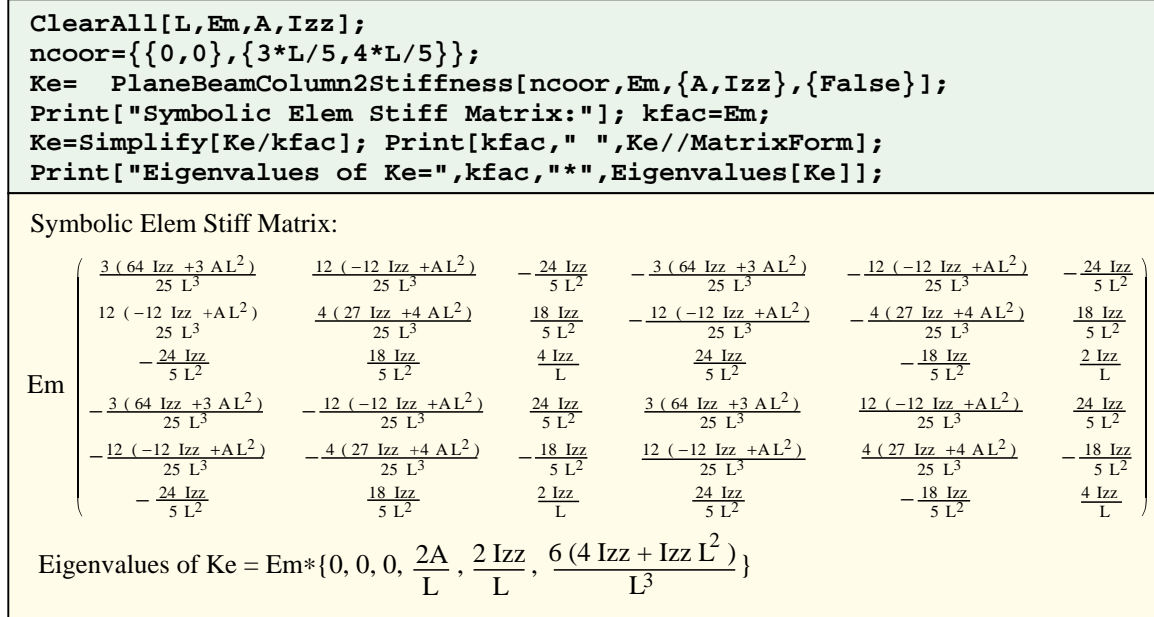


FIGURE 20.12. Test of two-node plane beam-column element with symbolic inputs.

§20.4. *Plane Beam With Offset Nodes

§20.4.1. Plate Reinforced With Edge Beams

Consider a plate reinforced with edge beams, as shown in Figure 20.13(a). The conventional placement of the nodes is at the plate midsurface and beam longitudinal (centroidal) axis. But those *element centered* locations do not coincide. To assemble the structure it is necessary to refer both the plate and beam stiffness equations to common locations, because such equations are only written at nodes. We assume that those *connection nodes*, or simply *connectors*, will be placed at the plate midsurface, as sketched in Figure 20.13(b). With that choice there is no need to change the plate equations. The beam connectors have been moved, however, from their original centroidal positions. For the beam these connectors are also known as *offset nodes*.

Structural configurations such as that of Figure 20.13(a) are common in aerospace, civil and mechanical engineering when shells or plates are reinforced with eccentric stiffeners.

The process of moving the beam stiffness equation to the offset nodes is called *offsetting*. It relies on setting up multifreedom constraints (MFC) between centered and offset node freedoms, and applying the master-slave congruential transformation introduced in Chapter 8. For simplicity we discuss only this process assuming that the beam of Figure 20.13(b) is a plane beam whose freedoms are to be moved upwards by a distance d , which is positive if going upward from beam centroid. Freedoms at connection and centroidal nodes are declared to be master and slaves, respectively. They are labeled as shown in Figure 20.13(c,d). The original stiffness equations referred to centroidal (slave) freedoms are

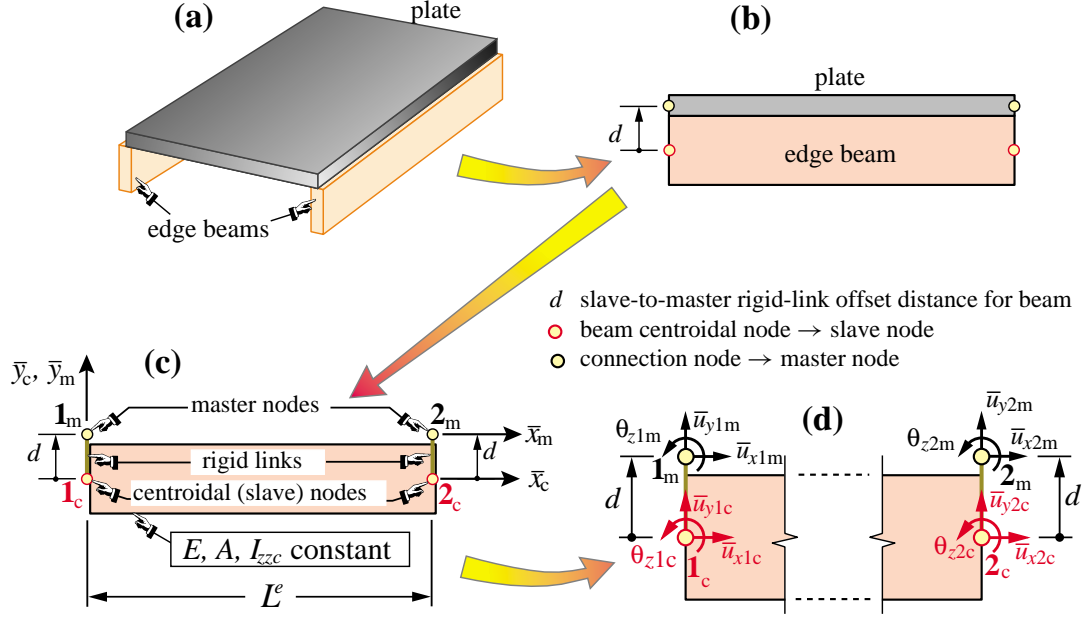


FIGURE 20.13. Plane beam with nodes offset for a rigid-link connection to plate.

$$\frac{E}{L^e} \begin{bmatrix} A & 0 & 0 & -A & 0 & 0 \\ & \frac{12I_{zzc}}{(L^e)^2} & \frac{6I_{zzc}}{(L^e)^2} & 0 & -\frac{12I_{zzc}}{(L^e)^2} & -\frac{6I_{zzc}}{(L^e)^2} \\ & & \frac{4I_{zzc}}{L^e} & 0 & -\frac{6I_{zzc}}{(L^e)^2} & \frac{2I_{zzc}}{L^e} \\ & & & A & 0 & 0 \\ & & & & \frac{12I_{zzc}}{(L^e)^2} & -\frac{6I_{zzc}}{(L^e)^2} \\ & & & & & \frac{4I_{zzc}}{L^e} \end{bmatrix} \begin{bmatrix} \bar{u}_{x1s} \\ \bar{u}_{y1s} \\ \theta_{z1s} \\ \bar{u}_{x2s} \\ \bar{u}_{y2s} \\ \theta_{z2s} \end{bmatrix} = \begin{bmatrix} \bar{f}_{x1s} \\ \bar{f}_{y1s} \\ m_{z1s} \\ \bar{f}_{x2s} \\ \bar{f}_{y2s} \\ m_{z2s} \end{bmatrix}, \quad \text{or} \quad \mathbf{K}_c^e \mathbf{u}_c^e = \mathbf{f}_c^e, \quad (20.12)$$

symm

in which A is the beam cross section area while I_{zzc} denotes the section moment of inertia with respect to the centroidal axis \bar{z}_c .

§20.4.2. Rigid Link Transformation

Kinematic constraints between master and centroidal (slave) freedoms are obtained assuming that they are connected by *rigid links* as pictured in Figure 20.13(c,d). This gives the centroidal(slave)-to-master transformation

$$\begin{bmatrix} \bar{u}_{x1c} \\ \bar{u}_{y1c} \\ \theta_{z1c} \\ \bar{u}_{x2c} \\ \bar{u}_{y2c} \\ \theta_{z2c} \end{bmatrix} = \begin{bmatrix} 1 & 0 & d & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & d \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_{x1m} \\ \bar{u}_{y1m} \\ \theta_{z1m} \\ \bar{u}_{x2m} \\ \bar{u}_{y2m} \\ \theta_{z2m} \end{bmatrix} \quad \text{or} \quad \mathbf{u}_s^e = \mathbf{T}_{sm} \mathbf{u}_m^e. \quad (20.13)$$

The inverse transformation: $\mathbf{T}_{mc} = \mathbf{T}_{cm}^{-1}$ is obtained on replacing d with $-d$, as is physically obvious. The modified stiffness equations are obtained by the congruent transformation: $\mathbf{T}_{cm}^T \mathbf{K}_c^e \mathbf{T}_{cm} = \mathbf{T}_{cm}^T \mathbf{f}_c^e = \mathbf{f}_m^e$, which yields

$$\frac{E}{L^e} \begin{bmatrix} A & 0 & d & -A & 0 & -d \\ 0 & \frac{12 I_{zzc}}{(L^e)^2} & \frac{6 I_{zzc}}{L^e} & 0 & -\frac{12 I_{zzc}}{(L^e)^2} & \frac{6 I_{zzc}}{L^e} \\ d & \frac{6 I_{zzc}}{L^e} & 4 I_{zzc} + A d^2 & -d & -\frac{6 I_{zzc}}{L^e} & 2 I_{zzc} - A d^2 \\ -A & 0 & -d & A & 0 & d \\ 0 & -\frac{12 I_{zzc}}{(L^e)^2} & -\frac{6 I_{zzc}}{L^e} & 0 & \frac{12 I_{zzc}}{(L^e)^2} & -\frac{6 I_{zzc}}{L^e} \\ -d & \frac{6 I_{zzc}}{L^e} & 2 I_{zzc} - A d^2 & d & -\frac{6 I_{zzc}}{L^e} & 4 I_{zzc} + A d^2 \end{bmatrix} \begin{bmatrix} \bar{u}_{x1m} \\ \bar{u}_{y1m} \\ \theta_{z1m} \\ \bar{u}_{x2m} \\ \bar{u}_{y2m} \\ \theta_{z2m} \end{bmatrix} = \begin{bmatrix} \bar{f}_{x1m} \\ \bar{f}_{y1m} \\ m_{z1m} \\ \bar{f}_{x2m} \\ \bar{f}_{y2m} \\ m_{z2m} \end{bmatrix} \quad (20.14)$$

Note that the modified equations are still referred to the local system $\{\bar{x}_m, \bar{y}_m\}$ pictured in Figure 20.13(c). Prior to assembly they should be transformed to the global system $\{x, y\}$ prior to assembly.

The foregoing transformation procedure has a flaw: for standard plate elements it will introduce compatibility errors at the interface between plate and beam. This may cause the beam stiffness to be significantly underestimated. See the textbook by Cook et. al. [72] for an explanation, and references therein. The following subsections describes a different scheme that builds \mathbf{K}_m^e directly and cures that incompatibility.

§20.4.3. Direct Fabrication of Offset Beam Stiffness

This approach directly interpolates displacements and strains from master nodes placed at distance d from the beam longitudinal (centroidal) axis, as pictured in Figure 20.14. As usual the isoparametric coordinate ξ along the beam element varies from -1 at node 1 through $+1$ at node 2. The following cross section geometric properties are defined for use below:

$$A = \int_{A^e} dA, \quad S_z = \int_{A^e} \bar{y} dA = A d, \quad I_{zzc} = \int_{A^e} \bar{y}_c^2 dA, \quad I_{zzm} = \int_{A^e} \bar{y}^2 dA = I_{zzc} + A d^2, \quad (20.15)$$

The inplane displacements are expressed in term of the master freedoms at nodes 1_m and 2_m . Using the Bernoulli-Euler model gives

$$\begin{bmatrix} \bar{u}_{xm} \\ \bar{u}_{ym} \end{bmatrix} = \begin{bmatrix} N_{ux1} & -\bar{y} \frac{\partial N_{uy1}}{\partial \bar{x}} & -\bar{y} \frac{\partial N_{\theta z1}}{\partial \bar{x}} & N_{ux2} & -\bar{y} \frac{\partial N_{uy2}}{\partial \bar{x}} & -\bar{y} \frac{\partial N_{\theta z2}}{\partial \bar{x}} \\ 0 & N_{uy1} & N_{\theta z1} & 0 & N_{uy2} & N_{\theta z2} \end{bmatrix} \begin{bmatrix} \bar{u}_{x1m} \\ \bar{u}_{y1m} \\ \theta_{z1m} \\ \bar{u}_{x2m} \\ \bar{u}_{y2m} \\ \theta_{z2m} \end{bmatrix} \quad (20.16)$$

where $N_{ux1} = \frac{1}{2}(1 - \xi)/2$, $N_{ux2} = \frac{1}{2}(1 + \xi)/2$, $N_{uy1} = \frac{1}{4}(1 - \xi)^2(2 + \xi)$, $N_{\theta z1} = \frac{1}{8}\ell(1 - \xi)^2(1 + \xi)$, $N_{uy2} = \frac{1}{4}(1 + \xi)^2(2 - \xi)$, $N_{\theta z2} = -\frac{1}{8}\ell(1 + \xi)^2(1 - \xi)$ are the usual bar and beam shape functions, but here referred to the offset axis \bar{x}_m .

The axial strain is $e_{xx} = \partial u_x / \partial \bar{x}$ and the strain energy $U^e = \frac{1}{2} \int_{V^e} E e_{xx}^2 dV$ where $dV = A d\bar{x} = A d\xi$. Carrying

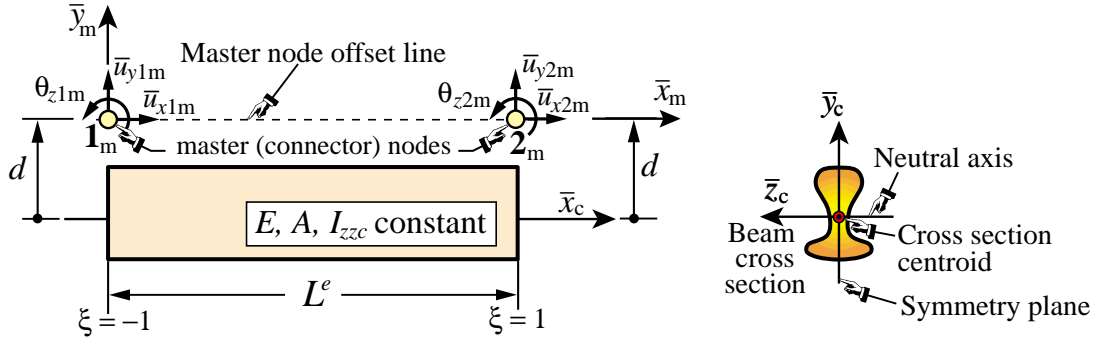


FIGURE 20.14. Plane beam fabricated directly from offset master node freedoms.

out the integral and differentiating twice with respect to the degrees of freedom yields the stiffness matrix

$$\begin{aligned}
 \mathbf{K}^e &= \frac{E}{L^e} \begin{bmatrix} A & 0 & -Ad & -A & 0 & Ad \\ 0 & \frac{12(I_{zzc} + Ad^2)}{(L^e)^2} & \frac{6(I_{zzc} + Ad^2)}{L^e} & 0 & \frac{-12(I_{zzc} + Ad^2)}{(L^e)^2} & \frac{6(I_{zzc} + Ad^2)}{L^e} \\ -Ad & \frac{6(I_{zzc} + Ad^2)}{L^e} & 4(I_{zzc} + Ad^2) & Ad & \frac{-6(I_{zzc} + Ad^2)}{L^e} & 2(I_{zzc} + Ad^2) \\ -A & 0 & Ad & A & 0 & -Ad \\ 0 & \frac{-12(I_{zzc} + Ad^2)}{(L^e)^2} & \frac{-6(I_{zzc} + Ad^2)}{L^e} & 0 & \frac{12(I_{zzc} + Ad^2)}{(L^e)^2} & \frac{-6(I_{zzc} + Ad^2)}{L^e} \\ Ad & \frac{6(I_{zzc} + Ad^2)}{L^e} & 2(I_{zzc} + Ad^2) & -Ad & \frac{-6(I_{zzc} + Ad^2)}{L^e} & 4(I_{zzc} + Ad^2) \end{bmatrix} \\
 &= \frac{EA}{L^e} \begin{bmatrix} 1 & 0 & -d & -1 & 0 & d \\ 0 & \frac{12(r_G^2 + d^2)}{(L^e)^2} & \frac{6(r_G^2 + d^2)}{L^e} & 0 & \frac{-12(r_G^2 + d^2)}{(L^e)^2} & \frac{6(r_G^2 + d^2)}{L^e} \\ -d & \frac{6(r_G^2 + d^2)}{L^e} & 4(r_G^2 + d^2) & d & \frac{-6(r_G^2 + d^2)}{L^e} & 2(r_G^2 + d^2) \\ -1 & 0 & d & 1 & 0 & -d \\ 0 & \frac{-12(r_G^2 + d^2)}{(L^e)^2} & \frac{-6(r_G^2 + d^2)}{L^e} & 0 & \frac{12(r_G^2 + d^2)}{(L^e)^2} & \frac{-6(r_G^2 + d^2)}{L^e} \\ d & \frac{6(r_G^2 + d^2)}{L^e} & 2(r_G^2 + d^2) & -d & \frac{-6(r_G^2 + d^2)}{L^e} & 4(r_G^2 + d^2) \end{bmatrix} \quad (20.17)
 \end{aligned}$$

In the second form, $r_G^2 = I_{zzc}/A$ is the squared radius of gyration of the cross section about z .

Comparing the first form of \mathbf{K}^e in (20.17) with (20.14) shows that the bending terms are significantly different if $d \neq 0$. These differences underscore the limitations of the rigid link assumption.

§20.5. Layered Beam Configurations

Another application of rigid link constraints is to modeling of *layered beams*, also called *composite beams* as well as *beam stacks*. These are beam-columns fabricated with beam layers that are linked to operate collectively as a beam member. In this section we study how to form the stiffness equations of beam stacks under the following assumptions:

1. The overall cross section of the beam member is rectangular and prismatic (that is, the cross section is constant along the longitudinal direction).
2. Both the beam stack and each of its constituent layers are prismatic.

3. The layers are of homogeneous isotropic material. Layer material, however, may vary from layer to layer.

The key modeling assumption is: is interlayer slip allowed or not? The two cases are studied next.

§20.5.1. Layered Beam With No Interlayer Slip

The main modeling constraint here is: *if all layers are of the same material, the stiffness equations should reduce to those of a homogenous beam.* To discuss how to meet this requirement it is convenient to introduce a *beam template* that separates the stiffness matrix into basic and higher order components. Consider a homogeneous, isotropic prismatic beam column element with elastic modulus E , cross section area A and cross section second moment of inertia I_{xxc} with respect to its neutral (centroidal) axis. The template form of the stiffness matrix in the local system is

$$\bar{\mathbf{K}}^e = \mathbf{K}_b^e + \mathbf{K}_h^e = \begin{bmatrix} K_{b1} & 0 & 0 & -K_{b1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{b2} & 0 & 0 & -K_{b2} \\ -K_{b1} & 0 & 0 & K_{b1} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -K_{b2} & 0 & 0 & K_{b2} \end{bmatrix} + \beta_h \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{h3} & K_{h4} & 0 & -K_{h3} & K_{h4} \\ 0 & K_{h4} & K_{h5} & 0 & -K_{h4} & K_{h5} \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -K_{h3} & -K_{h4} & 0 & K_{h3} & -K_{h4} \\ 0 & K_{h4} & K_{h5} & 0 & -K_{h4} & K_{h5} \end{bmatrix} \quad (20.18)$$

in which $K_{b1} = EA/L^e$, $K_{b2} = EI_{zzc}/L^e$, $K_{h3} = 12EI_{zzc}/(L^e)^3$, $K_{h4} = 6EI_{zzc}/(L^e)^2$ and $K_{h5} = 3EI_{zzc}/L^e$. Here β_h is a free parameter that scales the higher order stiffness \mathbf{K}_h . If $\beta_h = 1$ we recover the standard beam column stiffness (20.8). For a rectangular cross section of height H and width h , $A = Hh$ and $I_{zzc} = H^3 h/12$, and the template (20.18) becomes

$$\bar{\mathbf{K}}^e = \frac{Eh}{L^e} \begin{bmatrix} H & 0 & 0 & -H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{12}H^3 & 0 & 0 & -\frac{1}{12}H^3 \\ -H & 0 & 0 & H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{12}H^3 & 0 & 0 & \frac{1}{12}H^3 \end{bmatrix} + \frac{\beta_h E H^3 h}{4(L^e)^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2L^e & 0 & -4 & 2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & -2L^e & 0 & 4 & -2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \end{bmatrix} \quad (20.19)$$

Next, cut the foregoing beam into two identical layers of height $H_k = H/2$, where $k = 1, 2$ is used as layer index. See Figure 20.15(b). The layers have area $A_k = H_k h = Hh/2$ and self inertia $I_{xxk} = H_k^3 h/12 = H^3 h/96$. The layer stiffness matrices in template form are

$$\bar{\mathbf{K}}_k^e = \frac{Eh}{L^e} \begin{bmatrix} H_k & 0 & 0 & -H_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{12}H_k^3 & 0 & 0 & -\frac{1}{12}H_k^3 \\ -H_k & 0 & 0 & H_k & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{12}H_k^3 & 0 & 0 & \frac{1}{12}H_k^3 \end{bmatrix} + \frac{\beta_{hk} E H_k^3 h}{4(L^e)^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2L^e & 0 & -4 & 2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & -2L^e & 0 & 4 & -2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \end{bmatrix}, \quad k = 1, 2. \quad (20.20)$$

Beacuse the layers are identical it is reasonable to assume the same higher order free parameter for both layers, that is, $\beta_{h1} = \beta_{h2} = \beta_h$. The offset distances from each layer to the centroid of the full beam to the centroid of each layer are $d_1 = -H/4$ and $d_2 = H/4$. The rigid-link transformation matrices for (20.20) are the same as those found in the previous Section:

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & -H/4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & -H/4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} 1 & 0 & H/4 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & H/4 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \quad (20.21)$$

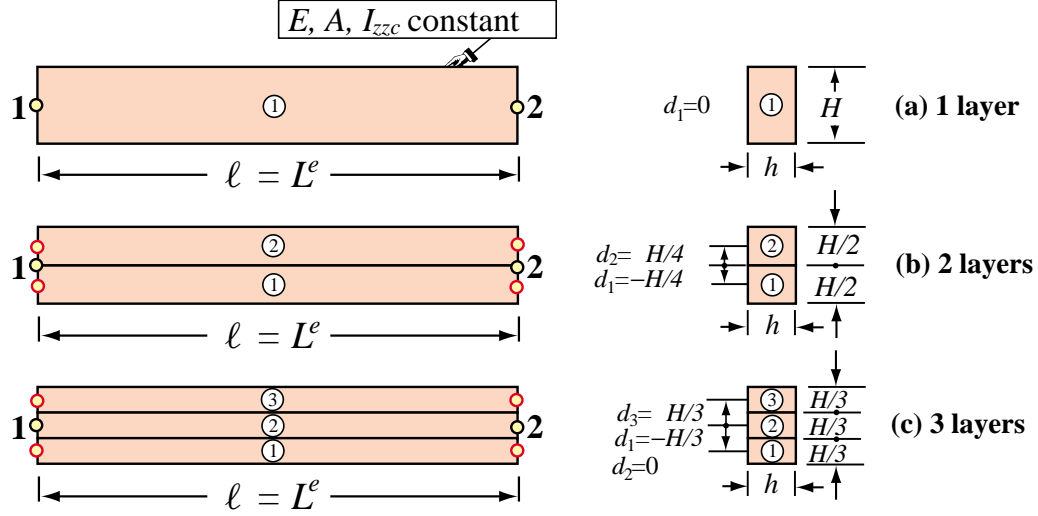


FIGURE 20.15. Plane beam divided into identical "sticking" layers.

Transforming and adding layers contributions as $\bar{\mathbf{K}}^e = \mathbf{T}_1^T \bar{\mathbf{K}}_1^e \mathbf{T}_1 + \mathbf{T}_2^T \bar{\mathbf{K}}_2^e \mathbf{T}_2$ gives

$$\bar{\mathbf{K}}^e = \frac{Eh}{L^e} \begin{bmatrix} H & 0 & 0 & -H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{12}H^3 & 0 & 0 & -\frac{1}{12}H^3 \\ -H & 0 & 0 & H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{12}H^3 & 0 & 0 & \frac{1}{12}H^3 \end{bmatrix} + \frac{\beta_h E H^3 h}{16(L^e)^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2L^e & 0 & -4 & 2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & -2L^e & 0 & 4 & -2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \end{bmatrix}. \quad (20.22)$$

This becomes identical to (20.19) if we set $\beta_h = 4$.

Carrying out the same exercise for three identical layers of height $H/3$, as shown in Figure 20.15(c), yields

$$\bar{\mathbf{K}}^e = \frac{Eh}{L^e} \begin{bmatrix} H & 0 & 0 & -H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{12}H^3 & 0 & 0 & -\frac{1}{12}H^3 \\ -H & 0 & 0 & H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\frac{1}{12}H^3 & 0 & 0 & \frac{1}{12}H^3 \end{bmatrix} + \frac{\beta_h E H^3 h}{36(L^e)^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 4 & 2L^e & 0 & -4 & 2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -4 & -2L^e & 0 & 4 & -2L^e \\ 0 & 2L^e & (L^e)^2 & 0 & -2L^e & (L^e)^2 \end{bmatrix}, \quad (20.23)$$

which becomes identical to (20.19) if we set $\beta_h = 9$. It is not difficult to show that if the beam is divided into $N \geq 2$ layers of height H/N , the correct beam stiffness is recovered if we take $\beta_h = N^2$.

It is not difficult to prove the following generalization. Suppose that the beam is cut into N layers of heights $H_k = \gamma_k H$, $k = 1, \dots, N$ that satisfy $\sum_1^N H_k = H$ or $\sum_1^N \gamma_k = 1$. To get the correct stiffness of the layered beam take

$$\beta_{hk} = 1/\gamma_k^2. \quad (20.24)$$

For example, suppose that the beam is divided into 3 layers of thicknesses $H_1 = H_3 = H/4$ and $H_2 = H/2$. Then pick $\beta_{h1} = \beta_{h3} = 1/(\frac{1}{4})^2 = 16$ and $\beta_{h2} = 1/(\frac{1}{2})^2 = 4$.

What is the interpretation of this boost? A spectral analysis of the combined stiffness shows that taking $\beta_{hk} = 1$ lowers the rigidity associated with the antisymmetric bending mode of the element. But this mode is associated with shear-slippage between layers. Boosting β_{hk} as found above compensates exactly for this rigidity decay.

§20.5.2. Beam Stacks Allowing Interlayer Slip

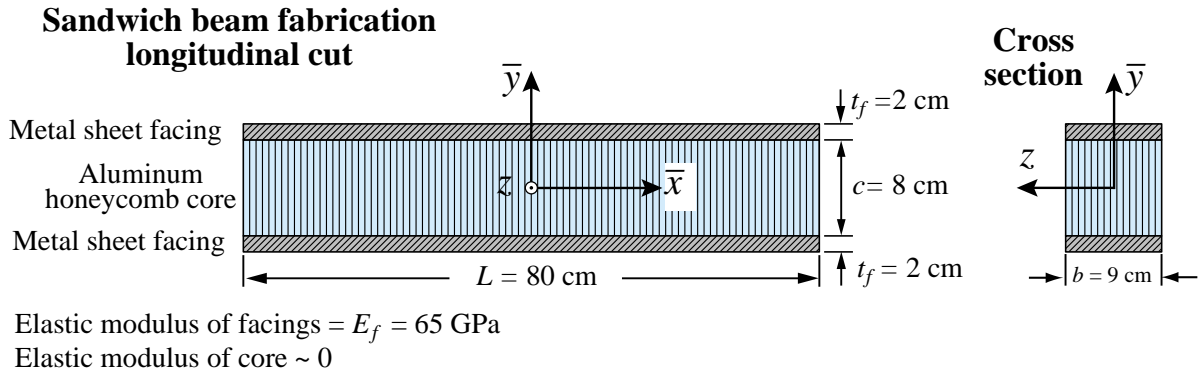


FIGURE 20.16. Plane sandwich beam.

There are beam fabrications where layers can slip longitudinally past each other. One important example is the sandwich beam illustrated in Figure 20.16. The beam is divided into three layers: 2 metal sheet facings and a honeycomb core. The core stiffness can be neglected since its effective elastic modulus is very low compared to the facings modulus E_f . In addition, the facings are not longitudinally bonded since they are separated by the weaker core. To form the beam stiffness by the rigid link method it is sufficient to form the faces self-stiffness, which are identical, and the rigid-link transformation matrices:

$$\mathbf{K}_k^e = \frac{E_f}{L^e} \begin{bmatrix} A_f & 0 & 0 & -A_f & 0 & 0 \\ 0 & \frac{12 I_{zzf}}{(L^e)^2} & \frac{6 I_{zzf}}{L^e} & 0 & -\frac{12 I_{zzf}}{(L^e)^2} & \frac{6 I_{zzf}}{L^e} \\ 0 & \frac{6 I_{zzf}}{L^e} & 4 I_{zzf} & -d & -\frac{6 I_{zzf}}{L^e} & 2 I_{zzf} \\ -A_f & 0 & 0 & A_f & 0 & 0 \\ 0 & -\frac{12 I_{zzf}}{(L^e)^2} & -\frac{6 I_{zzf}}{L^e} & 0 & \frac{12 I_{zzf}}{(L^e)^2} & -\frac{6 I_{zzf}}{L^e} \\ 0 & \frac{6 I_{zzf}}{L^e} & 2 I_{zzf} & 0 & -\frac{6 I_{zzf}}{L^e} & 4 I_{zzf} \end{bmatrix}, \quad \mathbf{T}_k = \begin{bmatrix} 1 & 0 & d_f & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & d_f \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad k = 1, 2. \quad (20.25)$$

where, in the notation of Figure 20.16, $A_f = bt_f$, $I_{xxf} = bt_f^3/12$, $d_1 = -(c + t_f)/2$ and $d_2 = (c + t_f)/2$. The stiffness of the sandwich beam is $\mathbf{K}^e = \mathbf{T}_1^T \mathbf{K} \mathbf{T}_1 + \mathbf{T}_2^T \mathbf{K} \mathbf{T}_2$ into which the numerical values given in Figure 20.16 may be inserted. There is no need to use here the template form and of adjusting the higher order stiffness.

§20.6. The Space Beam Element

A second example in 3D is the general beam element shown in Figure 20.17. The element is prismatic and has two end nodes: 1 and 2, placed at the centroid of the end cross sections.

These define the local \bar{x} axis as directed from 1 to 2. For simplicity the cross section will be assumed to be doubly symmetric, as is the case in commercial I and double-T profiles. The principal moments of inertia are defined by these symmetries. The local \bar{y} and \bar{z} axes are aligned with the symmetry lines of the cross section forming a RH system with \bar{x} . Consequently the principal moments of inertia are I_{yy} and I_{zz} , the bars being omitted for convenience.

The global coordinate system is $\{x, y, z\}$. To define the orientation of $\{\bar{y}, \bar{z}\}$ with respect to the global system, a third orientation node 3, which must not be colinear with 1–2, is introduced. See Figure 20.17. Axis \bar{y} lies in the 1–2–3 plane and \bar{z} is normal to 1–2–3.

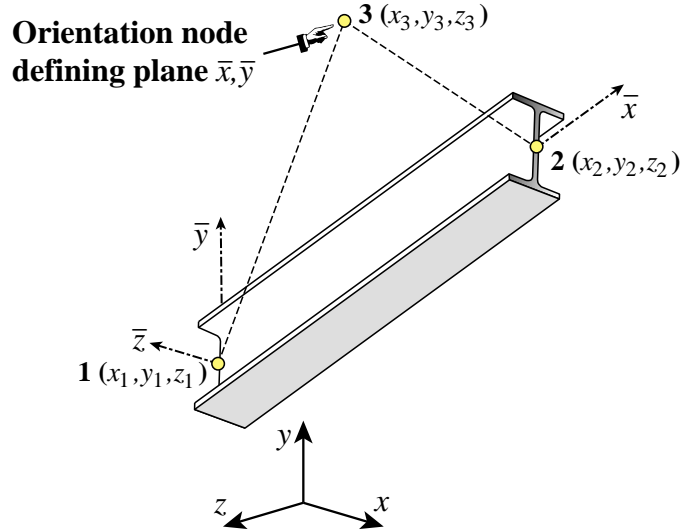


FIGURE 20.17. The space (3D) beam element.

Six global DOF are defined at each node i : the 3 translations u_{xi} , u_{yi} , u_{zi} and the 3 rotations θ_{xi} , θ_{yi} , θ_{zi} .

§20.6.1. Stiffness Matrix

The element global node displacements and conjugate forces are arranged as

$$\begin{aligned} \mathbf{u}^e &= [u_{x1} \quad u_{y1} \quad u_{z1} \quad \theta_{x1} \quad \theta_{y1} \quad \theta_{z1} \quad u_{x2} \quad u_{y2} \quad u_{z2} \quad \theta_{x2} \quad \theta_{y2} \quad \theta_{z2}]^T, \\ \mathbf{f}^e &= [f_{x1} \quad f_{y1} \quad f_{z1} \quad m_{x1} \quad m_{y1} \quad m_{z1} \quad f_{x2} \quad f_{y2} \quad f_{z2} \quad m_{x2} \quad m_{y2} \quad m_{z2}]^T. \end{aligned} \quad (20.26)$$

The beam material is characterized by the elastic modulus E and the shear modulus G (the latter appears in the torsional stiffness). Four cross section properties are needed: the cross section area A , the moment of inertia J that characterizes torsional rigidity,² and the two principal moments of inertia I_{yy} and I_{zz} taken with respect to \bar{y} and \bar{z} , respectively. The length of the element is denoted by L . The Bernoulli-Euler model is used; thus the effect of transverse shear on the beam stiffness is neglected.

To simplify the following expressions, define the following “rigidity” combinations by symbols: $R^a = EA/L$, $R^t = GJ/L$, $R_{y3}^b = EI_{yy}/L^3$, $R_{y2}^b = EI_{yy}/L^2$, $R_y^b = EI_{yy}/L$, $R_{z3}^b = EI_{zz}/L^3$, $R_{z2}^b = EI_{zz}/L^2$, $R_z^b = EI_{zz}/L$. Note that R^a is the axial rigidity, R^t the torsional rigidity, while

² For circular and annular cross sections, J is the polar moment of inertia of the cross section wrt \bar{x} . For other sections J has dimensions of (length)⁴ but must be calculated according to St. Venant’s theory of torsion, or approximate theories.

```

SpaceBeamColumn2Stiffness[ncoor_, {Em_, Gm_}, {A_, Izz_, Iyy_, Jxx_},
options_] := Module[
{
x1, x2, y1, y2, z1, z2, x21, y21, z21, xm, ym, zm, x0, y0, z0, dx, dy, dz,
EA, EIyy, EIzz, GJ, numer, ra, ry, ry2, ry3, rz, rz2, rz3, rx,
L, LL, LLL, yL, txx, txy, txz, tyx, tyy, tyz, tzx, tzy, tzz, T, Kebar, Ke},
{
x1, y1, z1} = ncoor[[1]]; {x2, y2, z2} = ncoor[[2]];
{x0, y0, z0} = {xm, ym, zm} = {x1 + x2, y1 + y2, z1 + z2} / 2;
If [Length[ncoor] <= 2, {x0, y0, z0} = {0, 1, 0}];
If [Length[ncoor] == 3, {x0, y0, z0} = ncoor[[3]]];
{x21, y21, z21} = {x2 - x1, y2 - y1, z2 - z1}; {numer} = options;
EA = Em * A; EIzz = Em * Izz; EIyy = Em * Iyy; GJ = Gm * Jxx;
LL = Simplify[x21^2 + y21^2 + z21^2]; L = Sqrt[LL];
If [numer, {x21, y21, z21, EA, EIyy, EIzz, GJ, LL, L} =
N[{x21, y21, z21, EA, EIyy, EIzz, GJ, LL, L}]];
If [!numer, L = PowerExpand[L]]; LLL = Simplify[LL * L];
ra = EA / L; rx = GJ / L;
ry = 2 * EIyy / L; ry2 = 6 * EIyy / LL; ry3 = 12 * EIyy / LLL;
rz = 2 * EIzz / L; rz2 = 6 * EIzz / LL; rz3 = 12 * EIzz / LLL;
Kebar = {
{
ra, 0, 0, 0, 0, 0, -ra, 0, 0, 0, 0, 0},
{
0, rz3, 0, 0, 0, rz2, 0, -rz3, 0, 0, 0, rz2},
{
0, 0, ry3, 0, -ry2, 0, 0, 0, -ry3, 0, -ry2, 0},
{
0, 0, 0, rx, 0, 0, 0, 0, 0, -rx, 0, 0},
{
0, 0, -ry2, 0, 2 * ry, 0, 0, 0, ry2, 0, ry, 0},
{
0, rz2, 0, 0, 0, 2 * rz, 0, -rz2, 0, 0, 0, rz},
{
-ra, 0, 0, 0, 0, 0, ra, 0, 0, 0, 0, 0},
{
0, -rz3, 0, 0, 0, -rz2, 0, rz3, 0, 0, 0, -rz2},
{
0, 0, -ry3, 0, ry2, 0, 0, 0, ry3, 0, ry2, 0},
{
0, 0, 0, -rx, 0, 0, 0, 0, 0, rx, 0, 0},
{
0, 0, -ry2, 0, ry, 0, 0, 0, ry2, 0, 2 * ry, 0},
{
0, rz2, 0, 0, 0, rz, 0, 0, -rz2, 0, 0, 2 * rz}};
{dx, dy, dz} = {x0 - xm, y0 - ym, z0 - zm}; If [numer, {dx, dy, dz} = N[{dx, dy, dz}]];
txx = dz * y21 - dy * z21; tzy = dx * z21 - dz * x21; tzz = dy * x21 - dx * y21;
zL = Sqrt[txx^2 + tzy^2 + tzz^2];
If [!numer, zL = Simplify[PowerExpand[zL]]];
{txx, tzy, tzz} = {txx, tzy, tzz} / zL; {txx, txy, txz} = {x21, y21, z21} / L;
tyx = tzy * txz - tzz * txy; tyy = tzz * txx - txz * txz; tyz = txz * txy - tzy * txx;
Te = {
{txx, txy, txz, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{tyx, tyy, tyz, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{txx, tzy, tzz, 0, 0, 0, 0, 0, 0, 0, 0, 0},
{0, 0, 0, txx, txy, txz, 0, 0, 0, 0, 0, 0},
{0, 0, 0, tyx, tyy, tyz, 0, 0, 0, 0, 0, 0},
{0, 0, 0, txx, tzy, tzz, 0, 0, 0, 0, 0, 0},
{0, 0, 0, 0, 0, 0, txx, txy, txz, 0, 0, 0},
{0, 0, 0, 0, 0, 0, tyx, tyy, tyz, 0, 0, 0},
{0, 0, 0, 0, 0, 0, txx, tzy, tzz, 0, 0, 0},
{0, 0, 0, 0, 0, 0, 0, 0, 0, txx, txy, txz},
{0, 0, 0, 0, 0, 0, 0, 0, 0, tyx, tyy, tyz},
{0, 0, 0, 0, 0, 0, 0, 0, 0, txx, tzy, tzz}};
Ke = Transpose[Te].Kebar.Te;
Return[Ke]
];

```

FIGURE 20.18. Module to form stiffness of space (3D) beam.

the R^b 's are bending rigidities scaled by the length in various ways. Then the 12×12 local stiffness matrix can be written as³

³ Cf. page 79 of Pzremieniecki [321]. The presentation in this book includes transverse shear effects as per Timoshenko's beam theory. The form (20.27) results from neglecting those effects.

$$\bar{\mathbf{K}}^e = \begin{bmatrix} R^a & 0 & 0 & 0 & 0 & 0 & -R^a & 0 & 0 & 0 & 0 & 0 \\ 0 & 12R_{z3}^b & 0 & 0 & 0 & 6R_{z2}^b & 0 & -12R_{z3}^b & 0 & 0 & 0 & 6R_{z2}^b \\ 0 & 0 & 12R_{y3}^b & 0 & -6R_{y2}^b & 0 & 0 & 0 & -12R_{y3}^b & 0 & -6R_{y2}^b & 0 \\ 0 & 0 & 0 & R^t & 0 & 0 & 0 & 0 & 0 & -R^t & 0 & 0 \\ 0 & 0 & -6R_{y2}^b & 0 & 4R_y^b & 0 & 0 & 0 & 6R_{y2}^b & 0 & 2R_y^b & 0 \\ 0 & 6R_{z2}^b & 0 & 0 & 0 & 4R_z^b & 0 & -6R_{z2}^b & 0 & 0 & 0 & 2R_z^b \\ -R^a & 0 & 0 & 0 & 0 & 0 & R^a & 0 & 0 & 0 & 0 & 0 \\ 0 & -12R_{z3}^b & 0 & 0 & 0 & -6R_{z2}^b & 0 & 12R_{z3}^b & 0 & 0 & 0 & -6R_{z2}^b \\ 0 & 0 & -12R_{y3}^b & 0 & 6R_{y2}^b & 0 & 0 & 0 & 12R_{y3}^b & 0 & 6R_{y2}^b & 0 \\ 0 & 0 & 0 & -R^t & 0 & 0 & 0 & 0 & 0 & R^t & 0 & 0 \\ 0 & 0 & -6R_{y2}^b & 0 & 2R_y^b & 0 & 0 & 0 & 6R_{y2}^b & 0 & 4R_y^b & 0 \\ 0 & 6R_{z2}^b & 0 & 0 & 0 & 2R_z^b & 0 & -6R_{z2}^b & 0 & 0 & 0 & 4R_z^b \end{bmatrix} \quad (20.27)$$

The transformation to the global system is the subject of Exercise 20.8.

§20.6.2. Stiffness Module

The computation of the stiffness matrix \mathbf{K}^e of the two-node, prismatic space beam-column element is done by *Mathematica* module `SpaceBeamColumn2Stiffness`. This is listed in Figure 20.18. The module is invoked as

```
Ke = SpaceBeamColumn2Stiffness[ncoor, {Em,Gm}, {A,Izz,Iyy,Jxx}, options]
```

(20.28)

The arguments are

- | | |
|---------|--|
| ncoor | Node coordinates of element arranged as $\{\{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \{x_3, y_3, z_3\}\}$, in which $\{x_3, y_3, z_3\}$ specifies an orientation node 3 that defines the local frame. See §20.4.1. |
| Em | Elastic modulus. |
| Gm | Shear modulus. |
| A | Cross section area. |
| Izz | Moment of inertia of cross section area respect to axis \bar{z} . |
| Iyy | Moment of inertia of cross section area respect to axis \bar{y} . |
| Jxx | Inertia with respect to \bar{x} that appears in torsional rigidity GJ . |
| options | A list of processing options. For this element is has only one entry: <code>{numer}</code> . This is a logical flag with the value <code>True</code> or <code>False</code> . If <code>True</code> the computations are carried out in floating-point arithmetic. If <code>False</code> symbolic processing is assumed. |

The module returns the 12×12 element stiffness matrix as function value.

The implementation logic and testing of this element is the subject of Exercises 20.8 and 20.9.

Notes and Bibliography

All elements implemented here are formulated in most books dealing with matrix structural analysis. Przemieniecki [321] has been recommended in Chapter 1 on account of being inexpensive. The implementation and testing procedures are rarely covered.

The use of rigid links for offsetting degrees of freedom is briefly covered in §7.8 of the textbook [72].

Homework Exercises for Chapter 20

Implementation of One-Dimensional Elements

EXERCISE 20.1 [C:15] Download the plane bar stiffness module and their testers and verify the test results reported here. Comment on whether the stiffness matrix K_e has the correct rank of 1.

EXERCISE 20.2 [C:15] Download the space bar stiffness module and their testers and verify the test results reported here. Comment on whether the computed stiffness matrix K_e has the correct rank of 1.

EXERCISE 20.3 [C:15] Download the plane beam-column stiffness module and their testers and verify the test results reported here. Comment on whether the computed stiffness matrix K_e has the correct rank of 3.

EXERCISE 20.4 [A+C:30] Explain why the space bar element has rank 1 although it has 6 degrees of freedom and 6 rigid body modes. (According to the formula given in Chapter 19, the correct rank should be $6 - 6 = 0$.)

EXERCISE 20.5 [C:25] Implement the plane bar, plane beam-column and space bar stiffness element module in a lower level programming language and check them by writing a short test driver. [Do not bother about the mass modules.] Your choices are C, Fortran 77 or Fortran 90. (C++ is overkill for this kind of software).

EXERCISE 20.6 [A:25] Explain why the eigenvalues of \mathbf{K}^e of any the elements given here do not change if the $\{x, y, z\}$ global axes change.

EXERCISE 20.7 [A+C:30] (Advanced) Implement a 3-node space bar element. *Hint:* use the results of Exercise 16.5 and transform the local stiffness to global coordinates via a 3×9 transformation matrix. Test the element and verify that it has two nonzero eigenvalues.

EXERCISE 20.8 [D+A:25] Explain the logic of the space beam module listed in Figure 20.18. Assume that the local stiffness matrix $\tilde{\mathbf{K}}^e$ stored in Kebar is correct (it has been transcribed from [321]). Instead, focus on how the local to global transformation is built and applied.

EXERCISE 20.9 [C:25] Test the space beam element of Figure 20.18 using the scripts given in Figures E20.1 and E20.2, and report results. Comment on whether the element exhibits the correct rank of 6.

```
ClearAll[L,Em,Gm,A,Izz,Iyy,Jxx];
ncoor={{0,0,0},{1,8,4}}; Em=54; Gm=30;
A=18; Izz=36; Iyy=72; Jxx=27;
Ke=SpaceBeamColumn2Stiffness[ncoor,{Em,Gm},{A,Izz,Iyy,Jxx},{True}];
Print["Numerical Elem Stiff Matrix: "];
Print[SetPrecision[Ke,4]//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[Ke]]];
```

FIGURE E20.1. Script for numeric testing of the space beam module of Figure 20.18.

```
ClearAll[L,Em,Gm,A,Izz,Iyy,Jxx];
ncoor={{0,0,0},{2*L,2*L,L}/3};
Ke=SpaceBeamColumn2Stiffness[ncoor,{Em,Gm},{A,Izz,Iyy,Jxx},{False}];
kfac=Em; Ke=Simplify[Ke/kfac];
Print["Numerical Elem Stiff Matrix: "];
Print[kfac," ",Ke//MatrixForm];
Print["Eigenvalues of Ke=",kfac,"*",Eigenvalues[Ke]];
```

FIGURE E20.2. Script for symbolic testing of the space beam module of Figure 20.18.

21

FEM Program for Space Trusses

TABLE OF CONTENTS

	Page
§21.1. Introduction	21-3
§21.2. Analysis Stages	21-3
§21.3. Analysis Support Modules	21-3
§21.3.1. Defining Boundary Conditions	21-5
§21.3.2. Modifying the Master Stiffness Equations	21-6
§21.3.3. Displacement Solution and Reaction Recovery	21-8
§21.3.4. Flattening and Partitioning Node-Freedom Vectors	21-9
§21.3.5. Internal Force Recovery	21-10
§21.3.6. The Solution Driver	21-11
§21.4. Utility Print Modules	21-12
§21.5. Utility Graphic Modules	21-13
§21.5.1. Plot Module Calls	21-13
§21.5.2. Plot View Specification	21-15
§21.6. Example 1: Bridge Plane Truss Example	21-15
§21.7. Example 2: An Orbiting Truss Structure	21-19
§21. Notes and Bibliography.	21-20
§21. Exercises	21-21

§21.1. Introduction

This Chapter presents a complete FEM program for analysis of space trusses. Why not start with plane trusses? Three reasons. First, plane truss code already appeared in Chapter 4, although most components were tailored to the example truss of Chapters 2–3. Second, the difference between 2D and 3D implementation logic for truss analysis is insignificant. Finally, space trusses are more interesting in engineering applications, particularly for orbiting structures in aerospace.

The overall organization of the space truss analysis program is diagrammed in Figure 21.1.

The description is done in “bottom up” fashion. That means the element level modules are presented first, followed by midlevel modules, ending with the driver program. This means traversing the diagram of Figure 21.1 left to right and bottom to top.

The program includes some simple minded graphics, including animation. The graphic modules are provided in the posted Notebook but not described in detail. Data structures are explained along the way as they arise.

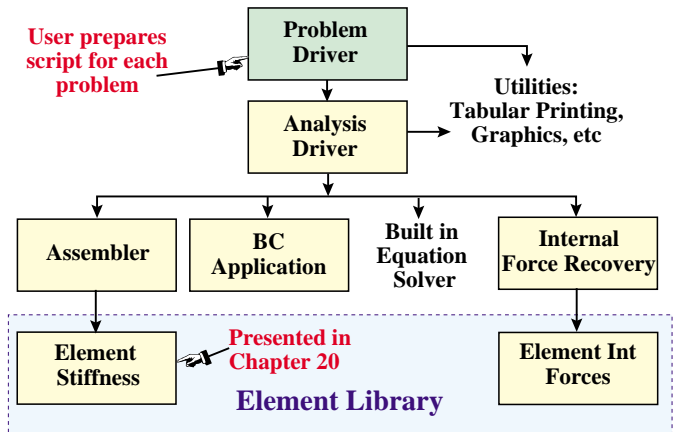


FIGURE 21.1. Overall organization of space truss analysis program.

§21.2. Analysis Stages

The analysis of a structure by the Direct Stiffness Method involves three major stages: preprocessing or model definition, processing, and postprocessing. This is true for toy programs such as the one presented here, through huge commercial codes. Of course the stages here are very short.

The *preprocessing* portion of the space truss analysis is done by a driver script, which directly sets the data structures for the problem at hand.

The *processing* stage involves three steps:

- Assembly of the master stiffness matrix, with a subordinate element stiffness module.
- Modification of master stiffness matrix and node force vector for displacement boundary conditions.
- Solution of the modified equations for displacements. For the program presented here the built in *Mathematica* function `LinearSolve` is used.

Upon executing the processing steps, the displacements are available. The following *postprocessing* steps follow.

- Recovery of forces including reactions, done through a \mathbf{Ku} matrix multiplication.
- Computation of internal (axial) forces and stresses in truss members.
- Plotting deflected shapes and member stress levels.

```

SpaceTrussMasterStiffness[nodxyz_,elenod_,
  elemat_,elefab_,prcopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodxyz],neldof,
  e,eftab,ni,nj,i,j,ii,jj,ncoor,Em,A,options,Ke,K},
  K=Table[0,{3*numnod},{3*numnod}];
  For [e=1, e<=numele, e++, {ni,nj}=elenod[[e]];
    eftab={3*ni-2,3*ni-1,3*ni,3*nj-2,3*nj-1,3*nj};
    ncoor={nodxyz[[ni]],nodxyz[[nj]]};
    Em=elemat[[e]]; A=elefab[[e]]; options=prcopt;
    Ke=SpaceBar2Stiffness[ncoor,Em,A,options];
    neldof=Length[Ke];
    For [i=1, i<=neldof, i++, ii=eftab[[i]];
      For [j=1, j<=neldof, j++, jj=eftab[[j]];
        K[[jj,ii]]=K[[ii,jj]]+Ke[[i,j]] ];
      ];
    ]; Return[K];
  ];
SpaceBar2Stiffness[ncoor_,Em_,A_,options_]:=Module[
  {x1,x2,y1,y2,z1,z2,x21,y21,z21,EA,numer,L,LL,LLL,Ke},
  {{x1,y1,z1},{x2,y2,z2}}=ncoor;{x21,y21,z21}={x2-x1,y2-y1,z2-z1};
  EA=Em*A; {numer}=options; LL=x21^2+y21^2+z21^2; L=Sqrt[LL];
  If [numer,{x21,y21,z21,EA,LL,L}=N[{x21,y21,z21,EA,LL,L}]];
  If [!numer, L=PowerExpand[L]]; LLL=Simplify[LL*L];
  Ke=(Em*A/LLL)*
    {{ x21*x21, x21*y21, x21*z21,-x21*x21,-x21*y21,-x21*z21},
    { y21*x21, y21*y21, y21*z21,-y21*x21,-y21*y21,-y21*z21},
    { z21*x21, z21*y21, z21*z21,-z21*x21,-z21*y21,-z21*z21},
    {-x21*x21,-x21*y21,-x21*z21, x21*x21, x21*y21, x21*z21},
    {-y21*x21,-y21*y21,-y21*z21, y21*x21, y21*y21, y21*z21},
    {-z21*x21,-z21*y21,-z21*z21, z21*x21, z21*y21, z21*z21}};
  Return[Ke];

```

FIGURE 21.2. Master stiffness assembly module for a space truss. The element stiffness module SpaceBar2Stiffness, already discussed in Chapter 20, is listed for convenience.

§21.3. Analysis Support Modules

We begin by listing here modules that support processing steps. These are put into separate cells for display and testing convenience.

HAssembling the Master Stiffness

Module SpaceTrussMasterStiffness, listed in Figure 21.2, assembles the master stiffness matrix of a space truss. It uses the element stiffness formation module SpaceBar2Stiffness discussed in the previous Chapter. That module is also listed in Figure 21.2 for convenience of the reader. The assembler is invoked by

$$K = \text{SpaceTrussMasterStiffness}[\text{nodxyz}, \text{elenod}, \text{elemat}, \text{elefab}, \text{prcopt}] \quad (21.1)$$

The arguments are

- nodxyz Nodal coordinates placed in a node-by-node list $\{\{x_1, y_1, z_1\}, \{x_2, y_2, z_2\}, \dots, \{x_n, y_n, z_n\}\}$, where n is the total number of nodes of the truss.
- elenod Element end nodes placed in an element-by-element list: $\{\{i_1, j_1\}, \{i_2, j_2\}, \dots, \{i_e, j_e\}\}$, where e is the total number of elements of the truss.

```

ClearAll[nodxyz,elemat,elefab,eleopt];
nodxyz={{0,0,0},{10,0,0},{10,10,0}};
elenod= {{1,2},{2,3},{1,3}};
elemat= Table[100,{3}]; elefab= {1,1/2,2*Sqrt[2]}; prcopt= {False};
K=SpaceTrussMasterStiffness[nodxyz,elenod,elemat,elefab,prcopt];
Print["Master Stiffness of Example Truss in 3D:\n",K//MatrixForm];
Print["eigs of K:",Chop[Eigenvalues[N[K]]]];

```

Master Stiffness of Example Truss in 3D:

$$\begin{pmatrix} 20 & 10 & 0 & -10 & 0 & 0 & -10 & -10 & 0 \\ 10 & 10 & 0 & 0 & 0 & 0 & -10 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 10 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 & 0 & -5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 0 & 0 & 10 & 10 & 0 \\ -10 & -10 & 0 & 0 & -5 & 0 & 10 & 15 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

eigs of K: {45.3577, 16.7403, 7.902, 0, 0, 0, 0, 0, 0}

FIGURE 21.3. Testing the space truss assembler module.

- elemat** Element material properties. The only such property required for this analysis is the elastic modulus E of each element. These are put in an element-by-element list: $\{E_1, E_2, \dots, E_e\}$, where e is the total number of elements of the truss.
- elefab** Element fabrication properties. The only such property required for this analysis is the cross section area A of each element. These are put in an element-by-element list: $\{A_1, A_2, \dots, A_e\}$, where e is the total number of elements of the truss.
- prcopt** Processing option list. Only one option is required in the assembler and so **prcopt** is simply $\{\text{numer}\}$. Here **numer** is a logical flag set to $\{\text{True}\}$ to tell the assembler to carry out element stiffness computations in floating-point arithmetic. Else set to $\{\text{False}\}$ to keep computations in exact arithmetic or symbolic form.

The module returns the assembled stiffness, stored as a full $3n \times 3n$ matrix, as function value.

Details of the assembly process are discussed in Chapter 25 for more general scenarios. Here we note that the module uses the freedom-pointer-table technique described in §3.4 for merging each element stiffness matrix into the master stiffness.

The assembler is tested by the script shown in the top cell of Figure 21.3. The script defines the example truss of Chapters 2–3 as a 3D structure with its three members placed in the $\{x, y\}$ plane. See Figure 21.4. The axial rigidity values $EA = 100, 50$ and $200\sqrt{2}$ for elements 1, 2 and 3, respectively, have to be untangled because E is placed in **elemat** whereas A goes to **elefab**.

To split EA we take $E = 100$ for the three elements. Thus **elemat** = $\{100, 100, 100\} = \text{Table}[100, \{3\}]$ whereas **elefab** = $\{1, 1/2, 2\sqrt{2}\}$.

Running the assembler in exact arithmetic gives the 9×9 master stiffness shown in the bottom cell of Figure 21.3. Taking its eigenvalues gives 6 zeros, which is the expected number in three dimensions.

§21.3.1. Defining Boundary Conditions

The modification process described here refers to the application of displacement boundary conditions on the master stiffness equations. These are assumed to be single-freedom constraints, either homogeneous such as $u_{x3} = 0$, or nonhomogeneous such as $u_{z6} = -0.72$.

Boundary condition data in FEM programs is usually specified in two levels: nodes at the first level, and freedoms assigned to that node at the second level. (The reason is that nodes are highly visible to casual users, whereas direct access to freedom numbers is difficult.) This space truss program is no exemption. This data is organized into two lists: node freedom tags and node freedom values. Their configuration is best specified through an example.

Consider again the example truss in 3D shown in Figure 21.4, which has 3 nodes and 9 freedoms. The node freedom tag list, internally called `nodtag`, is

$$\text{nodtag} = \{ \{1,1,1\}, \{0,1,1\}, \{0,0,1\} \} \quad (21.2)$$

Each first-level entry of this list pertains to a node. The second level is associated with freedoms: displacements in the x , y and z directions. Freedom activity is marked by tag 0 or 1. A 1-tag means that the displacement is prescribed, whereas a 0-tag indicates that the force is known. Thus $\{1,1,1\}$ for node 1 means that the node is fixed in the three directions.

The node freedom value list, internally called `nodval`, gives the prescribed value of the force or the displacement. For the example truss it is

$$\text{nodval} = \{ \{0,0,0\}, \{0,0,0\}, \{2,1,0\} \} \quad (21.3)$$

For node 1, the tag entry is $\{1,1,1\}$ and the value entry is $\{0,0,0\}$. This says that $u_{x1} = u_{y1} = u_{z1} = 0$. For node 2 it says that $f_{x2} = 0$ and $u_{y2} = u_{z2} = 0$. For node 3 it says that $f_{x3} = 2$, $f_{y3} = 1$ and $u_{z3} = 0$. The entries of `nodval` can be integers, floating point numbers, or symbols, whereas those in `nodtag` can only be 0 or 1.¹

§21.3.2. Modifying the Master Stiffness Equations

The modification of the master stiffness equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ for displacement BCs produces the modified system $\hat{\mathbf{K}}\mathbf{u} = \hat{\mathbf{f}}$. This is done by the two modules listed in Figure 21.5. These modules are not restricted to space trusses, and may in fact be used for more general problems.

The stiffness modifier is invoked by

$$\text{Kmod} = \text{ModifiedMasterStiffness}[\text{nodtag}, \mathbf{K}] \quad (21.4)$$

The arguments are:

¹ Other tag values may be implemented in more complicated programs to mark multifreedom constraints, for example.

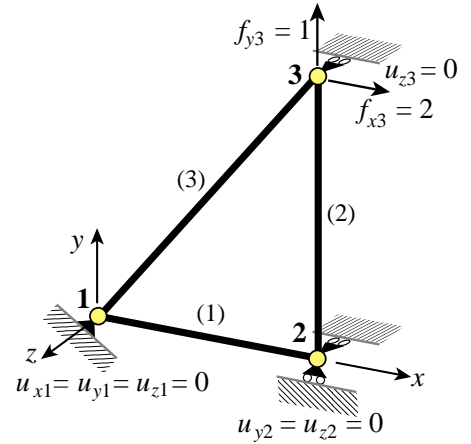


FIGURE 21.4. The example truss in three dimensions, used as module tester.

```

ModifiedMasterStiffness[nodtag_,K_] := Module[
  {i,j,k,n=Length[K],pdof,np,Kmod=K},
  pdof=PrescDisplacementDOFTags[nodtag]; np=Length[pdof];
  For [k=1,k<=np,k++, i=pdof[[k]];
    For [j=1,j<=n,j++, Kmod[[i,j]]=Kmod[[j,i]]=0];
    Kmod[[i,i]]=1];
  Return[Kmod]];

ModifiedNodeForces[nodtag_,nodval_,K_,f_] := Module[
  {i,j,k,n=Length[K],pdof,pval,np,d,c,fmod=f},
  pdof=PrescDisplacementDOFTags[nodtag]; np=Length[pdof];
  pval=PrescDisplacementDOFValues[nodtag,nodval]; c=Table[1,{n}];
  For [k=1,k<=np,k++, i=pdof[[k]]; c[[i]]=0];
  For [k=1,k<=np,k++, i=pdof[[k]]; d=pval[[k]];
    fmod[[i]]=d; If [d==0, Continue[]];
    For [j=1,j<=n,j++, fmod[[j]]-=K[[i,j]]*c[[j]]*d];
  ];
  Return[fmod]];

```

FIGURE 21.5. Modules to modify the master stiffness matrix and node force vector to apply the displacement boundary conditions.

nodtag A node by node list of freedom tags, as defined in the previous subsection.

K The master stiffness matrix **K** produced by the assembler module.

The modified stiffness matrix, which has the same order as **K**, is returned as function value.

The force modifier is invoked by

$$\text{fmod} = \text{ModifiedNodeForces}[\text{pdof}, \text{pval}, \text{K}, \text{f}] \quad (21.5)$$

The arguments are:

nodtag The node freedom tag list defined in the previous subsection.

nodval The node freedom value list defined in the previous subsection.

K The master stiffness matrix **K** produced by the assembler module (before modification). This is only used if at least one of the displacement BCs is non-homogeneous.

f The force vector before application of the displacement BCs.

The modified force vector, which has the same order as **f**, is returned as function value.

The modules are tested by the script listed in the top cell of Figure 21.6. It uses symbolic master stiffness equations of order 6. The test illustrates a not well known feature of *Mathematica*: use of `Array` function to generate subscripted symbolic arrays of one and two dimensions. The results, shown in the bottom cell of Figure 21.6, should be self explanatory.

Remark 21.1. On entry, the modification modules of Figure 21.5 build auxiliary lists **pdof** and **pval** to simplify the modification logic. **pdof** is a list of the prescribed degrees of freedom identified by their equation number in the master stiffness equations. For example, if freedoms 4, 7, 9 and 15 are specified, **pdof** = {4,7,9,15}. These indices are stored in ascending order. **pval** is a list of the prescribed displacement values listed in **pdof**. These lists are constructed by the modules listed in Figure 21.7. The calls are **pdof** = `PrescribedDOFTags[nodtag]` and **pval** = `PrescribedDOFValues[nodtag,nodval]`.

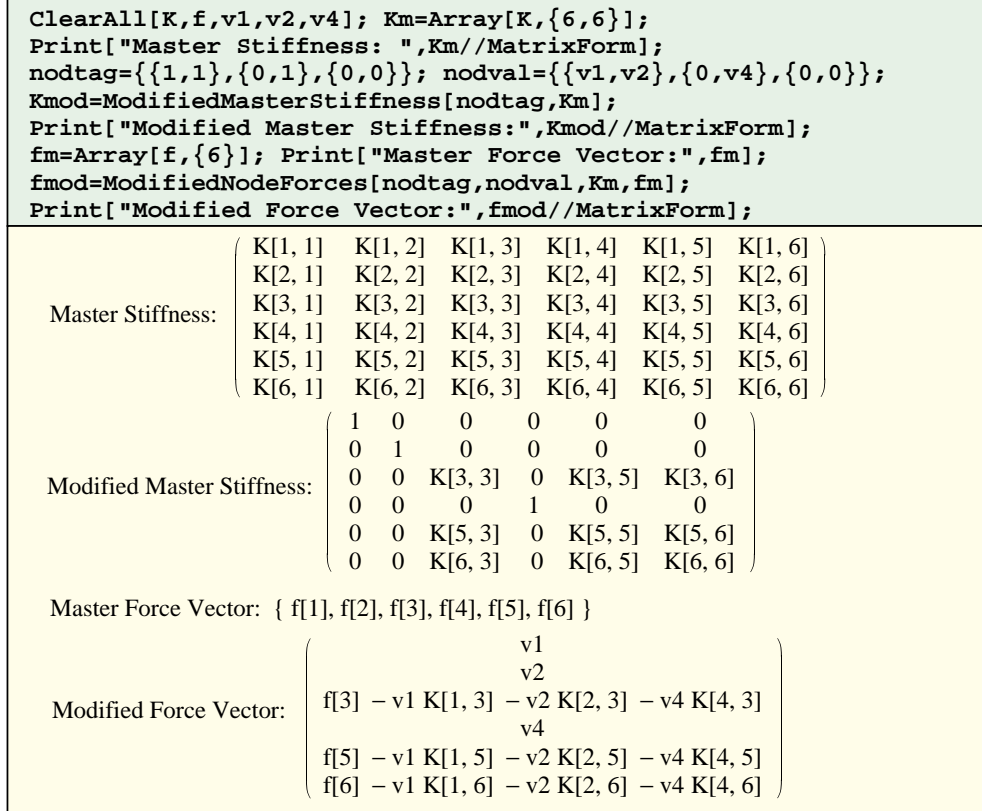


FIGURE 21.6. Testing the displacement boundary condition application modules.

Remark 21.2. The logic of `ModifiedMasterStiffness` is straightforward. Construct `pdof`, then clear appropriate rows and columns of \mathbf{K} and place ones on the diagonal. Note the use of the *Mathematica* function `Length` to control loops: `np=Length[pdof]` sets `np` to the number of prescribed freedoms. Similarly `n=Length[K]` sets `n` to the order of the master stiffness matrix \mathbf{K} , which is used to bound the row and column clearing loop. These statements may be placed in the list that declares local variables.

Remark 21.3. `ModifiedNodalForces` has more complicated logic because it accounts for nonhomogeneous BCs. On entry it constructs `pdof` and `pval`. If nonzero values appear in `pval`, the original entries of \mathbf{f} are modified as described in §4.1.2, and the end result is the effective force vector. Force vector entries corresponding to the prescribed displacement values are replaced by the latter in accordance with the prescription (4.13). If there are nonhomogeneous BCs it is important that the stiffness matrix provided as third argument be the master stiffness *before* modification by `ModifiedStiffnessMatrix`. This is because stiffness coefficients that are cleared by `ModifiedStiffnessMatrix` are needed for modifying the force vector.

§21.3.3. Displacement Solution and Reaction Recovery

The linear system $\hat{\mathbf{K}} \mathbf{u} = \hat{\mathbf{f}}$, where $\hat{\mathbf{K}}$ and $\hat{\mathbf{f}}$ are the modified stiffness and force matrices, respectively, is solved for displacements by a built-in linear algebraic solver. In *Mathematica* `LinearSolve` is available for this purpose:

$$\mathbf{u} = \text{LinearSolve}[\mathbf{Kmod}, \mathbf{fmod}] \quad (21.6)$$

At this point postprocessing begins. The node forces including reactions are obtained from $\mathbf{f} = \mathbf{K} \mathbf{u}$.

```

PrescDisplacementDOFTags[nodtag_]:= Module [
  {j,n,numnod=Length[nodtag],pdof={},k=0,m},
  For [n=1,n<=numnod,n++, m=Length[nodtag[[n]]];
    For [j=1,j<=m,j++, If [nodtag[[n,j]]>0,
      AppendTo[pdof,k+j]];
    ]; k+=m;
  ]; Return[pdof];
PrescDisplacementDOFValues[nodtag_,nodval_]:= Module [
  {j,n,numnod=Length[nodtag],pval={},k=0,m},
  For [n=1,n<=numnod,n++, m=Length[nodtag[[n]]];
    For [j=1,j<=m,j++, If [nodtag[[n,j]]>0,
      AppendTo[pval,nodval[[n,j]]]];
    ]; k+=m;
  ]; Return[pval];

```

FIGURE 21.7. Modules to build auxiliary lists pdof and pval from node-by-node BC data.

This can be done simply as a matrix product:

$$f = K.u \quad (21.7)$$

where K is the original master stiffness matrix before modification, and u the displacement vector computed by LinearSolve.

§21.3.4. Flattening and Partitioning Node-Freedom Vectors

In the analysis process one often needs displacement and force vectors in two different list formats. For example, the computed node displacement vector produced by (21.6) for the example truss in 3D is

$$u = \{0,0,0,0,0,0,0.4,-0.2,0\} \quad (21.8)$$

Following the terminology of Mathematica this will be called the *flat* form of the displacement vector. For postprocessing purposes (especially printing and plotting) it is convenient to rearrange to the node by node form

$$\text{nodd} = \{\{0,0,0\},\{0,0,0\},\{0.4,-0.2,0\}\} \quad (21.9)$$

This will be called the *node-partitioned* form of the displacement vector. Similar dual formats exist for the node force vector. In flat form this is called f and in node-partitioned form nodfor.

```

FlatNodePartVector[nv_]:=Flatten[nv];
NodePartFlatVector[nfc_,v_]:= Module [
  {i,k,m,n,nv={},numnod},
  If [Length[nfc]==0, nv=Partition[v,nfc]];
  If [Length[nfc]>0, numnod=Length[nfc]; m=0;
    nv=Table[0,{numnod}];
    For [n=1,n<=numnod,n++, k=nfc[[n]];
      nv[[n]]=Table[v[[m+i]],{i,1,k}];
      m+=k];
  ]; Return[nv];

```

FIGURE 21.8. Utility modules to flatten and node-partition node-DOF vectors.


```

SpaceTrussIntForces[nodxyz_,elenod_,elemat_,elefab_,
  noddis_,prcopt_]:= Module[{ numnod=Length[nodxyz],
  numele=Length[elenod],e,ni,nj,ncoor,Em,A,options,ue,p},
  p=Table[0,{numele}];
  For [e=1, e<=numele, e++, {ni,nj}=elenod[[e]];
    ncoor={nodxyz[[ni]],nodxyz[[nj]]};
    ue=Flatten[{ noddis[[ni]],noddis[[nj]] }];
    Em=elemat[[e]]; A=elefab[[e]]; options=prcopt;
    p[[e]]=SpaceBar2IntForce[ncoor,Em,A,ue,options]
  ];
  Return[p]];

SpaceBar2IntForce[ncoor_,Em_,A_,ue_,options_]:= Module[
  {x1,x2,y1,y2,z1,z2,x21,y21,z21,EA,numer,LL,pe},
  {{x1,y1,z1},{x2,y2,z2}}=ncoor; {x21,y21,z21}={x2-x1,y2-y1,z2-z1};
  EA=Em*A; {numer}=options; LL=x21^2+y21^2+z21^2;
  If [numer,{x21,y21,z21,EA,LL}=N[{x21,y21,z21,EA,LL}]];
  pe=(EA/LL)*(x21*(ue[[4]]-ue[[1]])+y21*(ue[[5]]-ue[[2]])+
    +z21*(ue[[6]]-ue[[3]]));
  Return[pe]];

SpaceTrussStresses[elefab_,elefor_,prcopt_]:= Module[
  {numele=Length[elefab],e,elesig}, elesig=Table[0,{numele}];
  For [e=1, e<=numele, e++, elesig[[e]]=elefor[[e]]/elefab[[e]] ];
  Return[elesig]];

```

FIGURE 21.9. Modules to compute internal forces and stresses in a space truss.

The utility modules listed in Figure 21.8 can be used to pass from one format to the other. To flatten the node-partitioned form of a vector, say nv , say

$$v = \text{FlatNodePartVector}[nv] \quad (21.10)$$

To node-partition a flat vector v say

$$nv = \text{NodePartFlatVector}[nfc,v] \quad (21.11)$$

where nfc is the number of freedoms per node. For space trusses this is 3 so appropriate conversion calls are $\text{noddis} = \text{NodePartFlatVector}[3,u]$ and $\text{nodfor} = \text{NodePartFlatVector}[3,f]$.

Remark 21.4. $\text{FlatNodePartVector}$ can be directly done by the built-in function Flatten whereas $\text{NodePartFlatVector}$ — for a fixed number of freedoms per node — can be done by Partition . The reason for the “wrappers” is to guide the conversion of *Mathematica* code to a lower level language such as C, where such built-in list functions are missing.

Remark 21.5. The additional code in $\text{NodePartFlatVector}$ caters to the case where the number of freedoms can vary from node to node, in which case nfc is a list (not a number) called the *node freedom count*, hence the abbreviation. That facility is useful in more advanced courses.

§21.3.5. Internal Force Recovery

The calculation of internal forces and stresses in a space truss involves computing axial forces in the bar elements. The modules that do those calculations given the displacement solution are listed


```

ClearAll[nodxyz,elenod,elemat,elefab,noddis];
nodxyz={{0,0,0},{10,0,0},{10,10,0}}; elenod={{1,2},{2,3},{1,3}};
elemat= Table[100,{3}]; elefab= {1,1/2,2*Sqrt[2]};
noddis={{0,0,0}, {0,0,0}, {4/10,-2/10,0}}; prcopt={False};
elefor=SpaceTrussIntForces[nodxyz,elenod,elemat,elefab,noddis,prcopt];
Print["Int Forces of Example Truss:",elefor];
Print["Stresses:",SpaceTrussStresses[elefab,elefor,prcopt]];

```

```

Int Forces of Example Truss: { 0, -1, 2*Sqrt[2] }
Stresses: { 0, -2, 1 }

```

FIGURE 21.10. Test of the internal force recovery module.

in Figure 21.9. Module `SpaceTrussIntForces` computes the internal forces (axial forces) in all truss members. It is invoked by

```
elefor = SpaceTrussIntForces[nodxyz,elenod,elemat, elefab,noddis,prcopt] (21.12)
```

Five of the arguments: `nodxyz`, `elenod`, `elemat`, `elefab` and `prcopt`, are the same used in the call (21.1) to the stiffness assembler. The additional argument, `noddis`, contains the computed node displacements arranged in node-partitioned form

```
noddis = {{ux1,uy1,uz1},{ux2,uy2,uz2}, ... {uxn,uy1,uzn}} (21.13)
```

This form can be obtained from the computed displacement solution through the utility module (21.11). As function value `SpaceTrussIntForces` returns a list of element axial forces

```
{p1, p2 ... pe } (21.14)
```

`SpaceTrussIntForces` makes use of `SpaceBar2IntForce`, which computes the internal force in an individual bar element. This is invoked as `SpaceBar2IntForce`

```
p = SpaceBar2IntForces[ncoor,Em,A,ue,options] (21.15)
```

Arguments `ncoor`, `Em`, `A` and `options` are the same as in the call to `SpaceBar2Stiffness` described in §20.2.2. The additional argument, `ue`, contains the flat list of the six element node displacements in the global system arranged as `{ux1,uy1,ux1,ux2,ux2,uz2}`. The recovery equation is the subject of Exercise 21.4.

The last module in Figure 21.9 computes the member stresses simply by dividing the internal forces by the cross section areas. It is invoked as

```
elesig = SpaceTrussStresses[elefab,elefor,prcopt] (21.16)
```

Here `elefab` and `prcopt` are as before, and `elefor` contains the element forces computed by `SpaceTrussIntForces`. The element axial stresses are returned as function value.

The statements of the top cell of Figure 21.9 exercise the internal force recovery for the example truss in 3D, requesting exact calculations. Array `p` is printed in the bottom cell. The axial forces of 0, -1 and $2\sqrt{2}$ agree with those determined in Chapter 3.

§21.3.6. The Solution Driver

It is convenient to package the sequence of operations described in the previous subsections, namely assembly, modification, solution, force and stress recovery, into one module called the *solution driver*. This is listed in Figure 21.11. It is invoked by saying

```
{noddis,nodfor,elefor,elesig}= SpaceTrussSolution[nodxyz,elenod,
                                         elemat,elefab,nodtag,nodval,prcopt] (21.17)
```

All arguments: nodxyz, elenod, elemat, elefab, nodtag, nodval and prcopt, have been described in previous subsections. The module returns four lists:

noddis Computed node displacement in node partitioned format.
 nodfor Recovered node forces including reactions in node partitioned format.
 elefor Element internal forces.
 elesig Element stresses.

Note that the listing of SpaceTrussSolution in Figure 21.11 has two commented out eigenvalue computations, one for the master stiffness K and one for the modified stiffness Kmod. Decomenting those commands comes in handy when setting up and running a new problem if errors are detected.

```
SpaceTrussSolution[nodxyz_,elenod_,elemat_,elefab_,nodtag_,nodval_,
prcopt_] := Module[{K,Kmod,f,fmod,u,noddis,nodfor,elefor,elesig},
K=SpaceTrussMasterStiffness[nodxyz,elenod,elemat,elefab,prcopt];
(* Print["eigs of K=",Chop[Eigenvalues[N[K]]]]; *)
Kmod=ModifiedMasterStiffness[nodtag,K];
f=FlatNodePartVector[nodval];
fmod=ModifiedNodeForces[nodtag,nodval,K,f];
(* Print["eigs of Kmod=",Chop[Eigenvalues[N[Kmod]]]]; *)
u=LinearSolve[Kmod,fmod]; u=Chop[u]; f=Chop[K.u, 10.0^(-8)];
nodfor=NodePartFlatVector[3,f]; noddis=NodePartFlatVector[3,u];
elefor=Chop[SpaceTrussIntForces[nodxyz,elenod,elemat,elefab,
noddis,prcopt]];
elesig=SpaceTrussStresses[elefab,elefor,prcopt];
Return[{noddis,nodfor,elefor,elesig}];
];
```

FIGURE 21.11. The analysis driver module.

§21.4. Utility Print Modules

Utility print modules are used to display input and output data in tabular form. The following six modules are provided in Cell 6 of the SpaceTruss.nb notebook.

To print the node coordinates in nodxyz:

```
PrintSpaceTrussNodeCoordinates[nodxyz,title,digits] (21.18)
```

To print the element nodes in elenod, element materials in elemat and element fabrications in elefab:

```
PrintSpaceTrussElementData[elenod,elemat,elefab,title,digits] (21.19)
```

To print the freedom activity data in `nodtag` and `nodval`:

```
PrintSpaceTrussFreedomActivity[nodtag,nodval,title,digits] (21.20)
```

To print the node displacements in `noddis` (configured in node-partitioned form):

```
PrintSpaceTrussNodeDisplacements[noddis,title,digits] (21.21)
```

To print the node forces in `nodfor` (configured in node-partitioned form):

```
PrintSpaceTrussNodeForces[nodfor,title,digits] (21.22)
```

To print the element internal forces in `elefor` and element stresses in `elesig`:

```
PrintSpaceTrussElemForcesAndStresses[elefor,elesig,title,digits] (21.23)
```

In all cases, `title` is an optional character string to be printed as a title before the table; for example "Node coordinates of bridge truss". To eliminate the title, specify "" (two quote marks together).

The last argument of the print modules: `digits`, is optional. If set to `{d,f}` it specifies that floating point numbers are to be printed with room for at least `d` digits, with `f` digits after the decimal point.

If `digits` is specified as a void list: `{ }`, a preset default is used for `d` and `f`.

§21.5. Utility Graphic Modules

Graphic modules that support preprocessing are placed in Cells 4 and 5 of the `SpaceTruss.nb` notebook. These display unlabeled elements, elements and nodes with labels, deformed shapes and element stress levels.

§21.5.1. Plot Module Calls

To plot elements only:

```
PlotSpaceTrussElements[nodxyz,elenod,title,{view,aspect,{}}] (21.24)
```

To plot element and nodes with optional labeling:

```
PlotSpaceTrussElementsAndNodes[nodxyz,elenod,title,{view,aspect,labels}] (21.25)
```

To plot deformed shape of the truss under computed node displacements:

```
PlotSpaceTrussDeformedShape[nodxyz,elenod,noddis,amplif,box,title,
                             {view,aspect,colors}] (21.26)
```

To plot element axial stress levels using a coloring scheme:

```
PlotSpaceTrussStresses[nodxyz,elenod,elesig,sigfac,box,title,
                       {view,aspect,{}}] (21.27)
```

In the foregoing `nodxyz`, `elenod`, `noddis`, `noddfor` `elefor` and `elesig` have been described above. The other arguments are as follows.

view A list configured as a list of two 3D vectors: `{{Vhat1,Vhat2,Vhat3},{W1,W2,W3}}`. Vector **W** with global components `{W1,W2,W3}` specifies the view direction whereas vector **V̂**, with global components `{V1,V2,V3}`, specifies the up direction for the plot view, as discussed in §21.5.2. If a void list is provided for the argument, the default is `{{0,1,0},{0,0,1}}`; this means that view direction is along the $-z$ axis whereas the up direction is the y axis.

- aspect** Vertical-to-horizontal aspect ratio of the plot as it appears on a Notebook cell. Three possibilities. If set to -1 , the *Mathematica* default `Aspect->Automatic` is chosen. If set to zero, an aspect ratio is computed by the plot module as appropriate. If set to a positive number, the aspect ratio is set to that number.
- labels** Only used in `PlotSpaceTrussElementsAndNodes`. A list with the following configuration: `{{nlabels,frn,fex,fey},{elabels,fre},{fntfam,fntsiz,fntwgt,fntslt}}}`.
- nlabels** A logical flag. Set to `True` to get node labels in plot.
- frn** Radius of circle drawn around each node expressed as percentage of plot size.
- fex** Horizontal eccentricity in points of node label from node location.
- fey** Vertical eccentricity in points of node label from node location.
- elabels** A logical flag. Set to `True` to get element labels in plot.
- fre** Radius of circle drawn around each element number expressed as percentage of plot size. from node location.
- fntfam** Font family used for labels, for example `"Times"`
- fntsiz** Size in points of font used for labels; usually 10 through 14.
- fntwgt** Font weight used for element labels. Typical settings are `"Plain"` or `"Bold"`. Node labels are always drawn in boldface.
- fntslt** Font slant used for element labels. Typical settings are `"Plain"`, `"Italics"` or `"Slanted"`. Node labels are always drawn in Plain.
- amplif** The displacement amplification factor to be used by `PlotSpaceTrussDeformedShape`. Usually a value much larger than one (say 100 or 1000) is necessary to visualize displacements in actual structures. Becomes a list if the call is to draw several shapes (for example undeformed and deformed). For example setting `amplif` to `{0,100}` will draw two shapes: the undeformed configuration and one with magnification of 100. If more than one shape is to be drawn the `colors` specification comes in handy.
- box** A list of points, specified by their $\{x, y, z\}$ global coordinates, that forms a box that encloses the plot. Used in `PlotSpaceTrussDeformedShape` and `PlotSpaceTrussStresses`. The box is converted to a frame by the view projector. This is useful for various purposes, one being to do realistic animations by drawing a sequence of deformed shapes moving inside this box.
- colors** Defines element colors to be used by `PlotSpaceTrussDeformedShape`, specified as lower case character string. Legal ones are `"black"`, `"red"`, `"blue"`, `"green"` and `"white"`. If the call is to draw several shapes (for example undeformed and deformed), this argument can be a list, such as `{ "black", "red" }`, in which case the colors are in one to one correspondence with the amplification values in `amplif`. If no color is specified, black is assumed.
- sigfac** A stress scaling factor for `PlotSpaceTrussStresses`. Normally set to 1.

Because of the confusing way *Mathematica* handles plots, (some features do scale with plot size while others, such as font sizes, do not), some interactive experimentation with dimension specs seems inevitable.

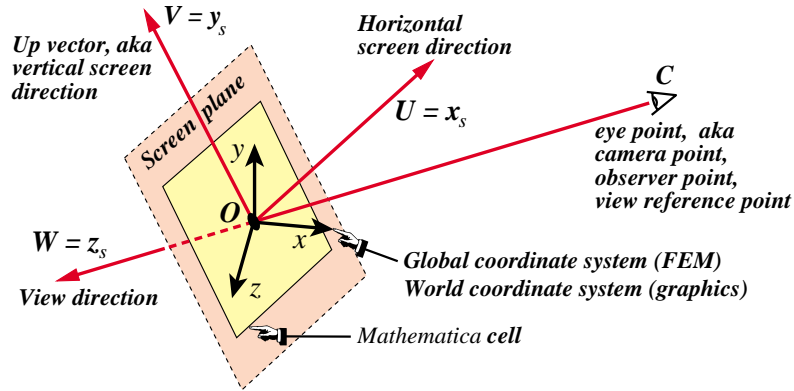


FIGURE 21.12. Plot view in 3D as mapping of world to screen coordinates.

§21.5.2. Plot View Specification

Plotting 3D objects, such as the space trusses considered here, involves mapping the coordinates given in the FEM global system $\{x, y, z\}$ into *screen coordinates* $\{x_s, y_s, z_s\}$ in which $z_s = 0$. Objects are rendered using screen coordinates. Construction of this mapping is based on the *view specification*, which appears as argument of all plotting routines described in the foregoing subsection.

The viewing ingredients are shown in Figure 21.12. In computer graphics, the 3D space spanned by the FEM global system $\{x, y, z\}$ is called the *world space* for obvious reasons. The screen coordinates $\{x_s, y_s, z_s\}$ are defined by two vectors, which in computer graphics are typically identified as \mathbf{W} and \mathbf{V} : the *view direction* and *up direction*, respectively.

The *view direction* is the line defined by joining the eye position at C with the origin O of $\{x, y, z\}$. The *screen plane* passes through O and is normal to \mathbf{W} . Screen coordinates x_s and y_s are in the screen plane and going along the horizontal and vertical directions, respectively. In computer graphics the x_s and y_s directions are called \mathbf{U} and \mathbf{V} , respectively. When the plot is rendered in a *Mathematica* cell, $\mathbf{U} \equiv x_s$ goes horizontally from left to right. Axes $\{\mathbf{u} \equiv x_s, \mathbf{V} \equiv y_s, \mathbf{W} \equiv z_s\}$ form a RHS Cartesian coordinate system. Mappings from screen to pixel coordinates are handled by the plotting system, and need not be discussed here.

In the plot modules used here, the eye point C is assumed to be at infinity.² Thus only the view direction \mathbf{W} , as specified by three direction numbers, is used. For example, the specification $\{1, 1, 1\}$ says that \mathbf{W} is the trisector of the $\{x, y, z\}$ octant. To define the up direction $\mathbf{V} \equiv y_s$ one has to enter a second “indication” vector: $\hat{\mathbf{V}}$, which must not be parallel to \mathbf{W} . Since $\hat{\mathbf{V}}$ is not necessarily normal to \mathbf{W} , \mathbf{V} is constructed by orthogonalization: $\mathbf{V} = \hat{\mathbf{V}} - (\hat{\mathbf{V}}^T \mathbf{W}_n) \mathbf{W}_n$, where \mathbf{W}_n is \mathbf{W} normalized to length one. For example if $\hat{\mathbf{V}}$ and \mathbf{W} are specified by view argument $\{\{0, 1, 0\}, \{2, 2, 1\}\}$, then $\mathbf{V} = [0, 1, 0] - ([0, 1, 0]^T [2/3, 2/3, 1/3] / [2/3, 2/3, 1/3]^T [2/3, 2/3, 1/3]) [2/3, 2/3, 1/3] = [0, 1, 0] - (2/3) [2/3, 2/3, 1/3] = [-4/9, 5/9, -2/9]$. Note that $\mathbf{V}^T \mathbf{W} = 0$.

§21.6. Example 1: Bridge Plane Truss Example

This example deals with the analysis of the 6-bay bridge truss problem defined in Figure 21.13. This truss has 12 nodes and 17 elements. It is contained in the $\{x, y\}$ plane and can only move in that plane. It is fixed at node 1 and on rollers at node 12.

² Graphics where the eye point C is at a finite distance produce *perspective plots*. The Graphics3D system of *Mathematica* allows perspective plotting. However the plots described here use only the 2D graphics subset.

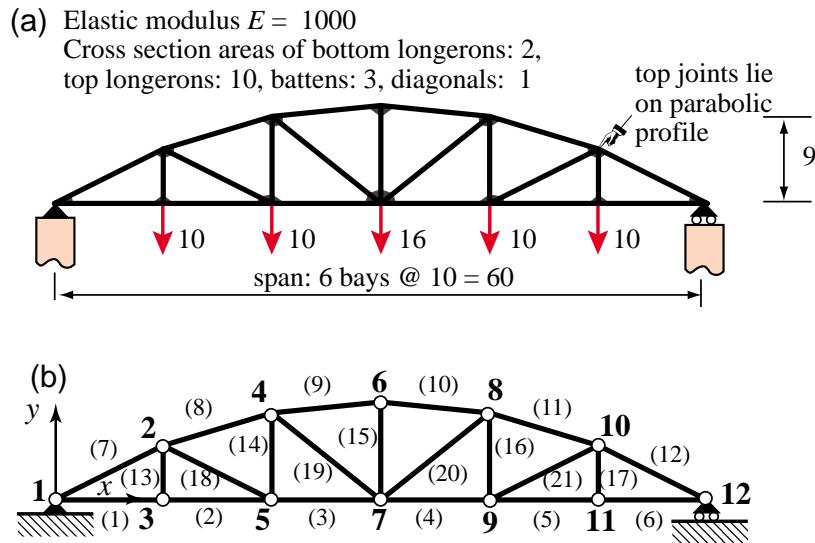


FIGURE 21.13. Six-bay bridge plane truss used as example problem: (a) truss structure showing supports and applied loads; (b) finite element idealization as pin-jointed truss.

The driver is listed in Figure 21.14. Preprocessing statements appear on top, with light green background. These define the problem through specification of the following data structures.³

NodeCoordinates	Same configuration as nodxyz
ElemNodes	Same configuration as elenod
ElemMaterials	Same configuration as elemat
ElemFabrications	Same configuration as elefab. This list is built up from four repeating cross sectional areas: Abot, Atop, Abat and Adia, for the areas of bottom longerons, top longerons, battens and diagonals, respectively.
NodeDOFTags	Same configuration as nodtag. Initialized to $\{0, 0, 1\}$ for all nodes on creation so as to fix all z displacements. Then the support conditions in the $\{x, y\}$ plane are specified at supported nodes. For example, $\text{NodeDOFTags}[[1]] = \{1, 1, 1\}$ says that the three node displacement components u_{x1} , u_{y1} and u_{z1} of node 1 are specified.
NodeDOFValues	Same configuration as nodval. Initialized to $\{0, 0, 0\}$ on creation for all nodes. Then the value of nonzero applied loads is set at nodes 3, 5, 7, 9 and 11. For example $\text{NodeDOFValues}[[7]] = \{0, -16, 0\}$ specifies $f_{x7} = 0$, $f_{y7} = -16$ and $u_{z7} = 0$.

The input data structures can be shown in tabular form for convenient inspection using print utility modules. Printed tables are shown on the left of Figure 21.15.

Running the solution module returns computed displacements, node forces including reactions, internal forces and member stress. These are printed with utility modules. These results are shown

³ Note the use of longer mnemonic names in the problem driver. For example `NodeCoordinates` instead of `nodxyz`. This simplifies the preparation of problem solving assignments since driver scripts are more self-documenting. It also helps grading returned assignments.

```

NodeCoordinates={ {0,0,0},{10,5,0},{10,0,0},{20,8,0},{20,0,0},{30,9,0},
                  {30,0,0},{40,8,0},{40,0,0},{50,5,0},{50,0,0},{60,0,0}};
ElemNodes={ {1,3},{3,5},{5,7},{7,9},{9,11},{11,12},
             {1,2},{2,4},{4,6},{6,8},{8,10},{10,12},
             {2,3},{4,5},{6,7},{8,9},{10,11},
             {2,5},{4,7},{7,8},{9,10}};
PrintSpaceTrussNodeCoordinates[NodeCoordinates,"Node coordinates:",{}];
numnod=Length[NodeCoordinates]; numele=Length[ElemNodes];
Em=1000; Abot=2; Atop=10; Abat=3; Adia=1;
ElemMaterials= Table[Em,{numele}];
ElemFabrications={Abot,Abot,Abot,Abot,Abot,Abot,Abot,Atop,Atop,Atop,Atop,
                  Atop,Atop,Abat,Abat,Abat,Abat,Abat,Adia,Adia,Adia,Adia};
PrintSpaceTrussElementData[ElemNodes,ElemMaterials,ElemFabrications,
                           "Element data:",{}];
ProcessOptions= {True};

view={ {0,1,0},{0,0,1}};
labels={ {True,0.06,-1.5,1.5},{True,0.12},{ "Times",11,"Roman"}};
PlotSpaceTrussElementsAndNodes[NodeCoordinates,ElemNodes,
                               "bridge mesh",{view,-1,labels}];

NodeDOFTags= Table[{0,0,1},{numnod}];
NodeDOFValues=Table[{0,0,0},{numnod}];
NodeDOFValues[[3]]= {0,-10,0}; NodeDOFValues[[5]]= {0,-10,0};
NodeDOFValues[[7]]= {0,-16,0};
NodeDOFValues[[9]]= {0,-10,0}; NodeDOFValues[[11]]= {0,-10,0};
NodeDOFTags[[1]]= {1,1,1}; (* fixed node 1 *)
NodeDOFTags[[numnod]]= {0,1,1}; (* hroller @ node 12 *)
PrintSpaceTrussFreedomActivity[NodeDOFTags,NodeDOFValues,
                              "DOF Activity:",{}];

{NodeDisplacements,NodeForces,ElemForces,ElemStresses}=
  SpaceTrussSolution[ NodeCoordinates,ElemNodes,ElemMaterials,
                    ElemFabrications,NodeDOFTags,NodeDOFValues,ProcessOptions ];

PrintSpaceTrussNodeDisplacements[NodeDisplacements,
                                "Computed node displacements:",{}];
PrintSpaceTrussNodeForces[NodeForces,
                          "Node forces including reactions:",{}];
PrintSpaceTrussElemForcesAndStresses[ElemForces,ElemStresses,
                                     "Int Forces and Stresses:",{}];

view={ {0,1,0},{0,0,1}}; box={ {0,-4,0},{60,-4,0},{60,10,0},{0,10,0}};
PlotSpaceTrussDeformedShape[NodeCoordinates,ElemNodes,NodeDisplacements,
                             {0,1},box,"deformed shape (unit magnif)",{view,-1,{ "black","blue"}}];
PlotSpaceTrussStresses[NodeCoordinates,ElemNodes,ElemStresses,1,box,
                       "axial stresses in truss members",{view,0,labels}];

```

FIGURE 21.14. Driver script for analysis of the 6-bay plane bridge truss. Preprocessing statements in light green background. Processing and postprocessing statements in light blue.

on the right of Figure 21.15.

Output plot results are collected in Figure 21.16.

Node coordinates:			
node	x-coor	y-coor	z-coor
1	0.000000	0.000000	0.000000
2	10.000000	5.000000	0.000000
3	10.000000	0.000000	0.000000
4	20.000000	8.000000	0.000000
5	20.000000	0.000000	0.000000
6	30.000000	9.000000	0.000000
7	30.000000	0.000000	0.000000
8	40.000000	8.000000	0.000000
9	40.000000	0.000000	0.000000
10	50.000000	5.000000	0.000000
11	50.000000	0.000000	0.000000
12	60.000000	0.000000	0.000000

Element data:			
elem	nodes	modulus	area
1	81, 3<	1000.00	2.00
2	83, 5<	1000.00	2.00
3	85, 7<	1000.00	2.00
4	87, 9<	1000.00	2.00
5	89, 11<	1000.00	2.00
6	811, 12<	1000.00	2.00
7	81, 2<	1000.00	10.00
8	82, 4<	1000.00	10.00
9	84, 6<	1000.00	10.00
10	86, 8<	1000.00	10.00
11	88, 10<	1000.00	10.00
12	810, 12<	1000.00	10.00
13	82, 3<	1000.00	3.00
14	84, 5<	1000.00	3.00
15	86, 7<	1000.00	3.00
16	88, 9<	1000.00	3.00
17	810, 11<	1000.00	3.00
18	82, 5<	1000.00	1.00
19	84, 7<	1000.00	1.00
20	87, 8<	1000.00	1.00
21	89, 10<	1000.00	1.00

DOF Activity:						
node	x-tag	y-tag	z-tag	x-value	y-value	z-value
1	1	1	1	0	0	0
2	0	0	1	0	0	0
3	0	0	1	0	-10	0
4	0	0	1	0	0	0
5	0	0	1	0	-10	0
6	0	0	1	0	0	0
7	0	0	1	0	-16	0
8	0	0	1	0	0	0
9	0	0	1	0	-10	0
10	0	0	1	0	0	0
11	0	0	1	0	-10	0
12	0	1	1	0	0	0

Computed node displacements:			
node	x-displ	y-displ	z-displ
1	0.000000	0.000000	0.000000
2	0.809536	-1.775600	0.000000
3	0.280000	-1.792260	0.000000
4	0.899001	-2.291930	0.000000
5	0.560000	-2.316600	0.000000
6	0.847500	-2.385940	0.000000
7	0.847500	-2.421940	0.000000
8	0.795999	-2.291930	0.000000
9	1.135000	-2.316600	0.000000
10	0.885464	-1.775600	0.000000
11	1.415000	-1.792260	0.000000
12	1.695000	0.000000	0.000000

Node forces including reactions:			
node	x-force	y-force	z-force
1	0.0000	28.0000	0.0000
2	0.0000	0.0000	0.0000
3	0.0000	-10.0000	0.0000
4	0.0000	0.0000	0.0000
5	0.0000	-10.0000	0.0000
6	0.0000	0.0000	0.0000
7	0.0000	-16.0000	0.0000
8	0.0000	0.0000	0.0000
9	0.0000	-10.0000	0.0000
10	0.0000	0.0000	0.0000
11	0.0000	-10.0000	0.0000
12	0.0000	28.0000	0.0000

Int Forces and Stresses:		
elem	axial force	axial stress
1	56.0000	28.0000
2	56.0000	28.0000
3	57.5000	28.7500
4	57.5000	28.7500
5	56.0000	28.0000
6	56.0000	28.0000
7	-62.6100	-6.2610
8	-60.0300	-6.0030
9	-60.3000	-6.0300
10	-60.3000	-6.0300
11	-60.0300	-6.0030
12	-62.6100	-6.2610
13	10.0000	3.3330
14	9.2500	3.0830
15	12.0000	4.0000
16	9.2500	3.0830
17	10.0000	3.3330
18	1.6770	1.6770
19	3.2020	3.2020
20	3.2020	3.2020
21	1.6770	1.6770

FIGURE 21.15. Bridge truss example: tabular printed output. On the left: node, element and freedom data. On the right: computed displacements, node forces, internal forces and member stresses.

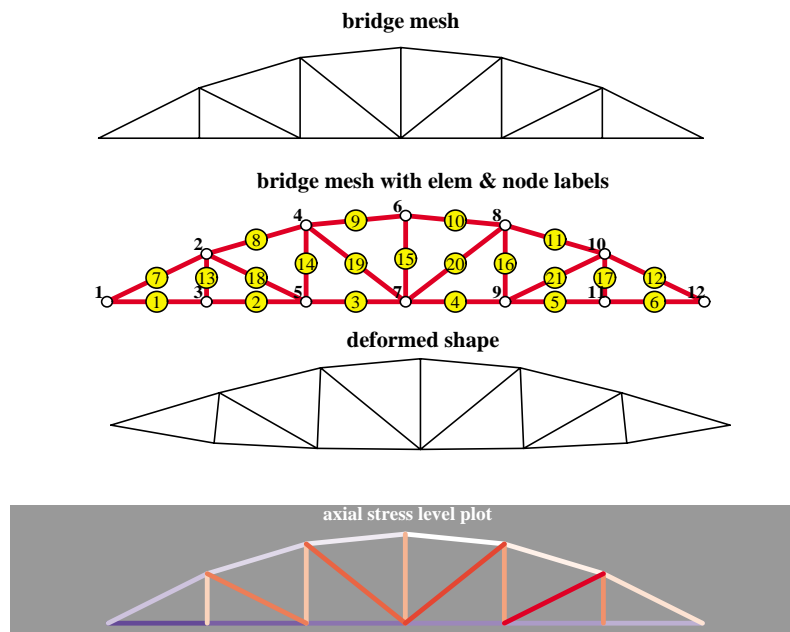


FIGURE 21.16. Bridge truss example: graphic output collected in one figure.

§21.7. Example 2: An Orbiting Truss Structure

To be included in the final version of the Chapter.

Notes and Bibliography

The dominant philosophy in FEM implementation is to construct general purpose programs that can solve a wide range of problems. For example, static and dynamic response of arbitrary structures with linear or nonlinear behavior. This path was naturally taken once the Direct Stiffness Method became widely accepted in the mid 1960s. It is reflected in the current crop of commercial FEM programs. Their source code by now has reached into millions of lines.

These codes do have a place in undergraduate engineering education, starting at the junior level. At this level students should be taught rudiments of modeling and how to use those black box programs as tools. This exposure provides also basic knowledge for capstone senior projects that require finite element analysis.

At the graduate level, however, students should understand what goes on behind the scene. But access to innards of commercial programs is precluded (and even if it were, it would be difficult to follow given their complexity). The philosophy followed here is to use special purpose codes written in a high level language. These may be collectively called *grey level codes*. A high level language such as *Mathematica* conceals utility operations such as matrix products, linear solvers and graphics, but permits the application code logic to be seen and studied.

As a result a complete FEM program is tiny (typically a few hundreds lines), it can be built and debugged in a few hours, and may be understood as a whole by one person. On the down side of course these toy programs can do only very limited problems, but for instructional use simplicity outweighs generality.

Homework Exercises for Chapter 21

FEM Program for Space Trusses

EXERCISE 21.1 [D:10] The logic of `SpaceTrussMasterStiffness` cannot be used for structures other than space trusses. Justify this assertion.

EXERCISE 21.2 [D:10] The logic of `ModifiedMasterStiffness` and `ModifiedNodeForces` is not restricted to space trusses, but can be used for any FEM program that stores \mathbf{K} and \mathbf{f} as full arrays. Justify this assertion.

EXERCISE 21.3 [D:20] The logic of `PrescDisplacementDOFTags` and `PrescDisplacementDOFValues` is not restricted to a fixed number of DOF per node. Justify this assertion.

EXERCISE 21.4 [A:15] Show that the longitudinal elongation of a space bar can be computed directly from the global displacements $u_{x1}, u_{y1}, \dots, u_{z2}$ from

$$d = (x_{21} u_{x21} + y_{21} u_{y21} + z_{21} u_{z21})/\ell, \quad (\text{E21.1})$$

in which $x_{21} = x_2 - x_1$, $u_{x21} = u_{x2} - u_{x1}$, etc, and ℓ is the bar length. Hence justify the formula used in module `SpaceBar2IntForce` listed in Figure 21.9 to recover the axial force $p = (EA/\ell)d$.

EXERCISE 21.5 [C:25] Analyze the structure shown in Figure E21.1. This is a pin-jointed truss model of a 200-in-high (5m) transmission tower originally proposed by Fox and Schmit in 1964 [146] as a test for early automated-synthesis codes based on FEM. It became a standard benchmark for structural optimization.

The truss has 10 joints (nodes) and 25 members (elements). The truss geometry and node numbering are defined in Figure E21.1(a). Joints 1 and 2 at the top of the tower lie on the $\{x, z\}$ plane. The truss (but not the loads) is symmetric about the $\{y, z\}$ and $\{x, z\}$ planes. Figure E21.1(b) gives the element numbers.

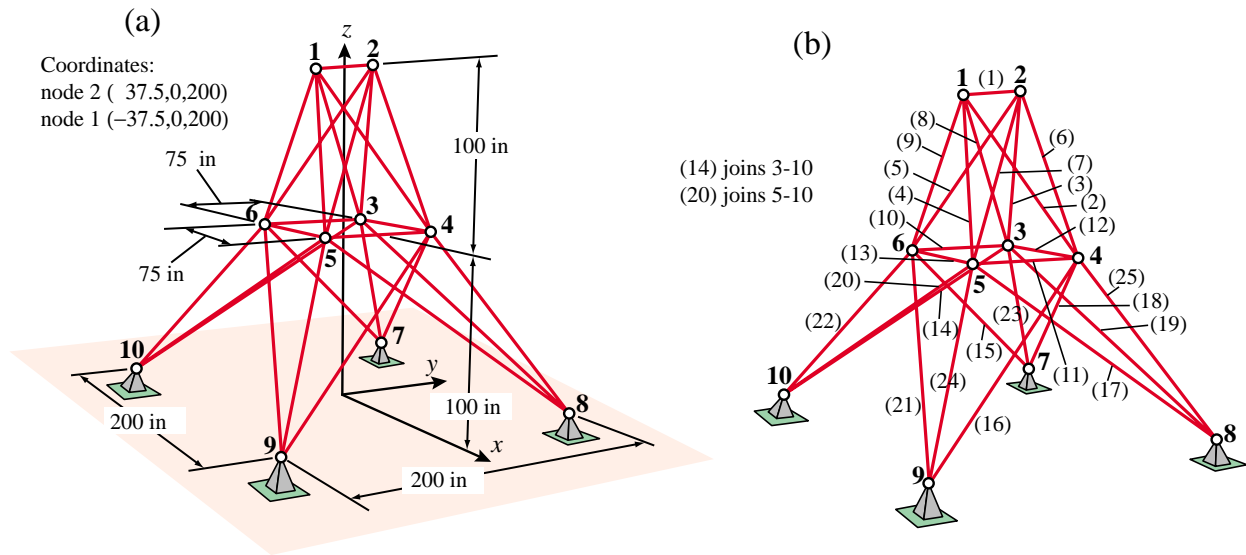


FIGURE E21.1. 25-member space truss model of a transmission tower. (a): Geometry definition and node numbers; (b) element numbers. For member properties and loads see Tables E21.1 and E21.2.

Table E21.1 Cross section areas of transmission tower members

Element	A (sq in)	Element	A (sq in)	Element	A (sq in)
1	0.033	10	0.010	19	1.760
2	2.015	11	0.010	20	1.760
3	2.015	12	0.014	21	1.760
4	2.015	13	0.014	22	2.440
5	2.015	14	0.980	23	2.440
6	2.823	15	0.980	24	2.440
7	2.823	16	0.980	25	2.440
8	2.823	17	0.980		
9	2.823	18	1.760		

Table E21.2 Applied load case for transmission tower

Node	x-load (lb)	y-load (lb)	z-load (lb)
1	1000	10000	−5000
2	0	10000	−5000
3	500	0	0
6	500	0	0
Applied forces at all other nodes are zero. Own-weight loads not considered.			

The members are aluminum tubes with the cross sections listed in Table E21.1.⁴ The modulus of elasticity is $E = 10^7$ psi for all members. The specific weight is 0.1 lb/in³. The applied load case to be studied is given in Table E21.2.

Analyze the transmission tower using the program provided in the `SpaceTruss.nb` Notebook (downloadable from Chapter 21 Index). Results to report: driver program cell, node displacements and element stresses. (More details on HW assignment sheet.) Note: as a quick check on model preparation, the total weight of the tower should be 555.18 lb.

A Fix for Version 6. Cell 0 added to Notebook `SpaceTruss.nb` on November 1, 2008 so that plots will display correctly under *Mathematica* Version 6.0 and later versions. If plots don't show up correctly please notify instructor.

⁴ Data taken from an optimal design reported by Venkayya, Khot and Reddy in 1968 [399].

22

FEM Programs for Plane Trusses and Frames

TABLE OF CONTENTS

	Page
§22.1. Introduction	22-3
§22.2. Analysis Stages	22-3
§22.3. Analysis Support Modules	22-3
§22.3.1. Assembling the Master Stiffness	22-3
§22.3.2. Modifying the Master Stiffness Equations	22-5
§22.3.3. Internal Force Recovery	22-6
§22.3.4. Graphic Modules	22-6
§22.4. A Bridge Truss Example	22-6
§22. Exercises	22-11

§22.1. Introduction

This Chapter presents a complete FEM program for analysis of plane trusses, programmed in *Mathematica*. The program includes some simple minded graphics, including animation. Exercises 22.2-22.4 discuss a complete FEM program for frame analysis for homework assignments.

The description is done in “bottom up” fashion. That means the basic modules are presented, then the driver program. Graphic modules are provided in posted Notebooks but not explained.

§22.2. Analysis Stages

As in all FEM programs, the analysis of a structure by the Direct Stiffness Method involves three major stages: (I) preprocessing or model definition, (II) processing, and (III) postprocessing.

The *preprocessing* portion of the plane truss analysis is done by the driver program, which directly sets the data structures.

I.1 Model definition by direct setting of the data structures.

I.2 Plot of the FEM mesh, including nodes and element labels.

The *processing* stage involves three steps:

II.1 Assembly of the master stiffness matrix, with a subordinate element stiffness module.

II.2 Modification of master stiffness matrix and node force vector for displacement boundary conditions.

II.3 Solution of the modified equations for displacements. For the programs presented here the built in *Mathematica* function `LinearSolve` is used.

Upon executing these three processing steps, the displacements are available. The following *post-processing* steps may follow:

III.1 Recovery of forces including reactions, done through a \mathbf{Ku} matrix multiplication.

III.2 Computation of internal (axial) forces in truss members.

III.3 Plotting deflected shapes and member stress levels.

These steps will be demonstrated in class from a laptop computer.

§22.3. Analysis Support Modules

We begin by listing here the modules that support various analysis steps, and which are put into separate cells for testing convenience.

§22.3.1. Assembling the Master Stiffness

The function `PlaneTrussMasterStiffness`, listed in Figure 22.1, assembles the master stiffness matrix of a plane truss. It uses the element stiffness formation module `PlaneBar2Stiffness` described in the previous Chapter. The statements at the end of the cell test this module by forming and printing \mathbf{K} of the example truss.

The arguments of `PlaneTrussMasterStiffness` are

```

PlaneTrussMasterStiffness[nodcoor_,elenod_,
  elemat_,elefab_,eleopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodcoor],
  e,eNL,eftab,ni,nj,i,j,ncoor,mprop,fprop,opt,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];
  For [e=1, e<=numele, e++,
    eNL=elenod[[e]]; {ni,nj}=eNL;
    eftab={2*ni-1,2*ni,2*nj-1,2*nj};
    ncoor={nodcoor[[ni]],nodcoor[[nj]]};
    mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
    Ke=PlaneBar2Stiffness[ncoor,mprop,fprop,opt];
    neldof=Length[Ke];
    For [i=1, i<=neldof, i++, ii=eftab[[i]];
      For [j=i, j<=neldof, j++, jj=eftab[[j]];
        K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]]
      ];
    ];
  ]; Return[K];
];

PlaneBar2Stiffness[ncoor_,mprop_,fprop_,opt_]:= Module[
  {x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,numer,L,LL,LLL,Ke},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A}=fprop; {numer}=opt;
  If [numer,{x21,y21,Em,A}=N[{x21,y21,Em,A}]];
  LL=x21^2+y21^2; L=PowerExpand[Sqrt[LL]]; LLL=Simplify[LL*L];
  Ke=(Em*A/LLL)*{{ x21*x21, x21*y21,-x21*x21,-x21*y21},
    { y21*x21, y21*y21,-y21*x21,-y21*y21},
    {-x21*x21,-x21*y21, x21*x21, x21*y21},
    {-y21*x21,-y21*y21, y21*x21, y21*y21}};

  Return[Ke]
];

nodcoor={{0,0},{10,0},{10,10}};
elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{100,0,0,0},{3}];
elefab= {{1},{1/2},{2*Sqrt[2]}};
eleopt= {True};
K=PlaneTrussMasterStiffness[nodcoor,elenod,
  elemat,elefab,eleopt];
Print["Master Stiffness of Example Truss:"];
Print[K/MatrixForm];

```

FIGURE 22.1. Master stiffness assembly module, with test statements in red.

- nodcoor** Nodal coordinates arranged as a two-dimensional list:
 $\{\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_n, y_n\}\}$,
 where n is the total number of nodes.
- elenod** Element end nodes arranged as a two-dimensional list:
 $\{\{i_1, j_1\}, \{i_2, j_2\}, \dots, \{i_e, j_e\}\}$,
 where e is the total number of elements.
- elemat** Element material properties arranged as a two-dimensional list:
 $\{\{Em_1, Gm_1, rho_1, alpha_1\}, \dots, \{Em_e, Gm_e, rho_e, alpha_e\}\}$,
 where e is the total number of elements. Only the elastic modulus Em is used in this program. For the other properties zeros may be stored as placeholders.


```

ModifiedMasterStiffness[pdof_,K_] := Module[
  {i,j,k,n=Length[K],np=Length[pdof],Kmod}, Kmod=K;
  For [k=1,k<=np,k++, i=pdof[[k]];
    For [j=1,j<=n,j++, Kmod[[i,j]]=Kmod[[j,i]]=0];
    Kmod[[i,i]]=1
  ];
  Return[Kmod]
];
ModifiedNodeForces[pdof_,f_] := Module[
  {i,k,np=Length[pdof],fmod}, fmod=f;
  For [k=1,k<=np,k++, i=pdof[[k]]; fmod[[i]]=0];
  Return[fmod]
];
K=Array[Kij,{6,6}];
Print["Assembled Master Stiffness:"];Print[K//MatrixForm];
K=ModifiedMasterStiffness[{1,2,4},K];
Print["Master Stiffness Modified For Displacement B.C.:"];
Print[K//MatrixForm];
f=Array[fi,{6}];
Print["Node Force Vector:"]; Print[f];
f=ModifiedNodeForces[{1,2,4},f];
Print["Node Force Vector Modified For Displacement B.C.:"];
Print[f];

```

FIGURE 22.2. Modifying the master stiffness and node force vector for displacement boundary conditions, with test statements in red.

- elefab** Element fabrication properties arranged as a two-dimensional list:
 $\{\{A_1\}, \dots \{A_e\}\}$,
 where e is the total number of elements, and A the cross section area.
- eleopt** Element processing option: set to $\{\text{True}\}$ to tell `PlaneBar2Stiffness` to carry out element stiffness computations in floating-point arithmetic. Else set to $\{\text{False}\}$ to keep computations in exact arithmetic or symbolic form.

The assembler uses the freedom-pointer-table technique described in §3.4 for merging the element stiffness matrix into the master stiffness. The module returns the master stiffness matrix \mathbf{K} in list K , which is stored as a full matrix.

The statements at the end of Figure 22.1 test this module by forming and printing the master stiffness matrix of the example truss. These statements are executed when the cell is initialized.

§22.3.2. Modifying the Master Stiffness Equations

Following the assembly process the master stiffness equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ must be modified to account for displacement boundary conditions. This is done through the computer-oriented equation modification process described in §3.4.2. Modules that perform this operation are listed in Figure 22.2, along with test statements.

Module `ModifiedMasterStiffness` carries out this process for the master stiffness matrix \mathbf{K} , whereas `ModifiedNodalForces` does this for the nodal force vector \mathbf{f} . The logic of `ModifiedNodalForces` is considerably simplified by assuming that *all prescribed displacements are zero*, that is, the BCs are homogeneous. This is the case in the implementation shown here.

Module `ModifiedMasterStiffness` receives two arguments:

- pdof A list of the prescribed degrees of freedom identified by their global number. For the example truss of Chapters 2-3 this list would have three entries: {1, 2, 4}, which are the freedom numbers of u_{x1} , u_{y1} and u_{y2} in \mathbf{u} .
- K The master stiffness matrix \mathbf{K} produced by the assembler module described in the previous subsection.

The module clears appropriate rows and columns of \mathbf{K} , places ones on the diagonal, and returns the thus modified \mathbf{K} as function value. Note the use of the *Mathematica* function `Length` to control loops: `np=Length[pdof]` sets `np` to the number of prescribed freedoms. Similarly `n=Length[K]` sets `n` to the order of the master stiffness matrix \mathbf{K} , which is used to bound the row and column clearing loop. These statements may be placed in the list that declares local variables.

Module `ModifiedNodalForces` has a very similar structure and logic and need not be described in detail. It is important to note, however, that for homogeneous BCs the two module are independent of each other and may be called in any order. On the other hand, if there were nonzero prescribed displacements the force modification must be done *before* the stiffness modification. This is because stiffness coefficients that are cleared in the latter are needed for modifying the force vector.

The test statements at the bottom of Figure 22.2 are chosen to illustrate another feature of *Mathematica*: the use of the `Array` function to generate subscripted symbolic arrays of one and two dimensions. The test output should be self explanatory and is not shown here. Both the force vector and its modified form are printed as row vectors to save space.

§22.3.3. Internal Force Recovery

Module `PlaneTrussIntForces` listed in Figure 22.3 computes the internal forces (axial forces) in all truss members. The first five arguments are the same as for the assembler routine described previously. The last argument, `u`, contains the computed node displacements arranged as a flat, one dimensional list:

$$\{ux1, uy1 \dots uxn, uyn\} \quad (22.1)$$

`PlaneTrussIntForces` makes use of `PlaneBar2IntForce`, which computes the internal force in an *individual member*. `PlaneBar2IntForce` is similar in argument sequence and logic to `PlaneBar2Stiffness` of Figure 22.1. The first four arguments are identical. The last argument, `ue`, contains the list of the four element node displacements in the global system. The logic of the recovery module is straightforward and follows the method outlined in §3.2.

The statements at the bottom of Figure 22.3 test the internal force recovery for the example truss of Chapters 2-3, a and should return forces of 0, -1 and $2\sqrt{2}$ for members 1, 2 and 3, respectively.

§22.3.4. Graphic Modules

Graphic modules that support preprocessing are placed in Cells 4A, 4B and 4C of the *Mathematica* notebook. These plot unlabeled elements, elements and nodes with labels, and boundary conditions. Graphic modules that support postprocessing are placed in Cells 5A and 5B of the Notebook. These plot deformed shapes and axial stress levels in color.

These modules are not listed since they are still undergoing modifications at the time of this writing. One unresolved problem is to find a way for absolute placement of supported nodes for correct deflected-shape animations.

```

PlaneTrussIntForces[nodcoor_,elenod_,elemat_,elefab_,
  eleopt_,u_]:= Module[{numele=Length[elenod],
  numnod=Length[nodcoor],e,eNL,eftab,ni,nj,i,
  ncoor,mprop,fprop,opt,ue,p},
  p=Table[0,{numele}]; ue=Table[0,{4}];
  For [e=1, e<=numele, e++,
    eNL=elenod[[e]]; {ni,nj}=eNL;
    eftab={2*ni-1,2*ni,2*nj-1,2*nj};
    ncoor={nodcoor[[ni]],nodcoor[[nj]]};
    mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
    For [i=1,i<=4,i++, ii=eftab[[i]]; ue[[i]]=u[[ii]]];
    p[[e]]=PlaneBar2IntForce[ncoor,mprop,fprop,opt,ue]
  ];
  Return[p]
];
PlaneBar2IntForce[ncoor_,mprop_,fprop_,opt_,ue_]:= Module[
  {x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,numer,LL,pe},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A}=fprop; {numer}=opt;
  (*If [numer,{x21,y21,Em,A}]=N[{x21,y21,Em,A}];*)
  LL=x21^2+y21^2;
  pe=Em*A*(x21*(ue[[3]]-ue[[1]])+y21*(ue[[4]]-ue[[2]]))/LL;
  Return[pe]
];
nodcoor={{0,0},{10,0},{10,10}}; elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{100,0,0,0},{3}]; elefab= {{1},{1/2},{2*Sqrt[2]}};
eleopt= {True}; u={0,0,0,0,0.4,-0.2};
p=PlaneTrussIntForces[nodcoor,elenod,elemat,elefab,eleopt,u];
Print["Int Forces of Example Truss:"];
Print[p];

```

FIGURE 22.3. Calculation of truss internal forces, with test statements in red.

§22.4. A Bridge Truss Example

The driver program in Cell 6 defines and runs the analysis of the 6-bay bridge truss problem defined in Figure 22.4. This truss has 12 nodes and 17 elements. It is fixed at node 1 and on rollers at node 12.

The driver is listed in Figures 22.5 and 22.6. It begins by defining the problem through specification of the following data structures:

NodeCoordinates Same configuration as nodcoor

ElemNodeLists Same configuration as elenod

ElemMaterials Same configuration as elemat

ElemFabrication Same configuration as elefab. This list is built up from four repeating cross sectional areas: Abot, Atop, Abat and Adia, for the areas of bottom longerons, top longerons, battens and diagonals, respectively.

ProcessOptions Same configuration as eleopt

FreedomValue. This is a two-dimensional list that specifies freedom values, node by node. It is initialized to zero on creation, then the value of nonzero applied loads is set at nodes 3, 5, 7, 9 and 11. For example FreedomValue[[7]] = {0,-16} specifies $f_{x7} = 0$ and $f_{y7} = -16$.

FreedomTag. This is a two-dimensional list that specifies, for each nodal freedom, whether the value

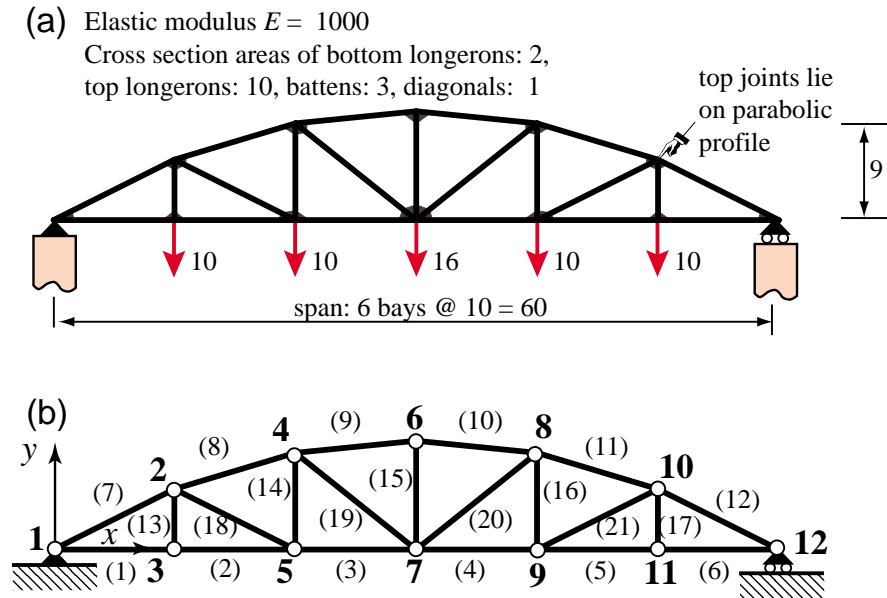


FIGURE 22.4. Six-bay bridge truss used as example problem: (a) truss structure showing supports and applied loads; (b) finite element idealization as pin-jointed truss.

of the force (tag 0) or displacement (tag 1) is prescribed. It is initialized to zero on creation, then the support tags at nodes 1 (fixed) and 12 (horizontal roller) are set. For example, `FreedomTag[[1]] = {1, 1}` says that both node displacement components u_{x1} and u_{y1} of node 1 are specified.

Processing commands are listed in Figure 22.6. The master stiffness matrix is assembled by the module listed in Figure 22.1. The stiffness matrix is placed in `K`. The applied force vector stored as a one-dimensional list, is placed in `f`, which results from the application of built-in function `Flatten` to `Freedom Value`.

The prescribed degree-of-freedom array `pdof` is constructed by scanning `FreedomTag` for nonzero values. For this problem `pdof={1, 2, 23}`. The displacement boundary conditions are applied by the modules of Figure 22.2, which return the modified master stiffness `Kmod` and the modified node force vector `fmod`. Note that the modified stiffness matrix is stored into `Kmod` rather than `K` to save the original form of the master stiffness for the recovery of reaction forces later.

The complete displacement vector is obtained by the matrix calculation

$$u = \text{LinearSolve}[Kmod, fmod] \quad (22.2)$$

which takes advantage of the built-in linear solver provided by `Mathematica`.

The remaining calculations recover the node vector including reactions by the matrix-vector multiply $f = K \cdot u$ (recall that `K` contains the unmodified master stiffness matrix). The member internal forces `p` are obtained through the module listed in Figure 22.3. The program prints `u`, `f` and `p` as row vectors to conserve space.

Running the program of Figures 22.5–6 produces the output shown in Figure 22.7. Output plot results are collected in Figure 22.8.

```

ClearAll[];
NodeCoordinates={{0,0},{10,5},{10,0},{20,8},{20,0},{30,9},
                {30,0},{40,8},{40,0},{50,5},{50,0},{60,0}};
ElemNodeLists= {{1,3},{3,5},{5,7},{7,9},{9,11},{11,12},
                {1,2},{2,4},{4,6},{6,8},{8,10},{10,12},
                {2,3},{4,5},{6,7},{8,9},{10,11},
                {2,5},{4,7},{7,8},{9,10}};
numnod=Length[NodeCoordinates];
numele=Length[ElemNodeLists]; numdof=2*numnod;
ElemMaterial= Table[{1000,0,0,0},{numele}];
Abot=2; Atop=10; Abat=3; Adia=1;
ElemFabrication=Join[Table[{Abot},{6}],Table[{Atop},{6}],
                    Table[{Abat},{5}],Table[{Adia},{4}]];
ProcessOptions= {True}; aspect=0;
PlotLineElements[NodeCoordinates,ElemNodeLists,aspect,
                "test mesh"];
PlotLineElementsAndNodes[NodeCoordinates,ElemNodeLists,aspect,
                "test mesh with elem & node labels",{True,0.12},{True,0.05}];

FreedomTag=FreedomValue=Table[{0,0},{numnod}];
FreedomValue[[3]]= {0,-10}; FreedomValue[[5]]= {0,-10};
FreedomValue[[7]]= {0,-16};
FreedomValue[[9]]= {0,-10}; FreedomValue[[11]]= {0,-10};
Print["Applied node forces="]; Print[FreedomValue];
FreedomTag[[1]]= {1,1}; (* fixed node 1 *)
FreedomTag[[numnod]]= {0,1}; (* hroller @ node 12 *)

```

FIGURE 22.5. Driver for analysis of the bridge truss: preprocessing.

```

f=Flatten[FreedomValue];
K=PlaneTrussMasterStiffness[NodeCoordinates,
    ElemNodeLists,ElemMaterial,ElemFabrication,ProcessOptions];
pdof={}; For[n=1,n<=numnod,n++, For[j=1,j<=2,j++,
    If[FreedomTag[[n,j]]>0, AppendTo[pdof,2*(n-1)+j]]];
Kmod=ModifiedMasterStiffness[pdof,K];
fmod=ModifiedNodeForces [pdof,f];
u=LinearSolve[Kmod,fmod]; u=Chop[u];
Print["Computed Nodal Displacements:"]; Print[u];
f=Simplify[K.u]; f=Chop[f];
Print["External Node Forces Including Reactions:"]; Print[f];
p=PlaneTrussIntForces[NodeCoordinates,ElemNodeLists,
    ElemMaterial,ElemFabrication,eleopt,u]; p=Chop[p];
sigma=Table[p[[i]]/ElemFabrication[[i,1]],{i,1,numele}];
Print["Internal Member Forces:"]; Print[p];
PlotTrussDeformedShape[NodeCoordinates,ElemNodeLists,u,
    1.0,aspect,"Deformed shape"];
PlotAxialStressLevel[NodeCoordinates,ElemNodeLists,sigma,
    1.0,aspect,"Axial stress level"];

```

FIGURE 22.6. Driver for analysis of the bridge truss: processing and postprocessing.

```

Applied node forces:
{ {0, 0}, {0, 0}, {0, -10}, {0, 0}, {0, -10}, {0, 0},
  {0, -16}, {0, 0}, {0, -10}, {0, 0}, {0, -10}, {0, 0} }

Computed Nodal Displacements:
{0, 0, 0.809536, -1.7756, 0.28, -1.79226, 0.899001,
 -2.29193, 0.56, -2.3166, 0.8475, -2.38594,
 0.8475, -2.42194, 0.795999, -2.29193, 1.135, -2.3166,
 0.885464, -1.7756, 1.415, -1.79226, 1.695, 0}

External Node Forces Including Reactions:
{0, 28., 0, 0, 0, -10., 0, 0, 0, -10., 0, 0, 0,
 -16., 0, 0, 0, -10., 0, 0, 0, -10., 0, 28.}

Internal Member Forces:
{56., 56., 57.5, 57.5, 56., 56., -62.6099,
 -60.0318, -60.2993, -60.2993,
 -60.0318, -62.6099, 10., 9.25, 12., 9.25,
 10., 1.67705, 3.20156, 3.20156, 1.67705}

```

FIGURE 22.7. Printed output from the bridge truss analysis.

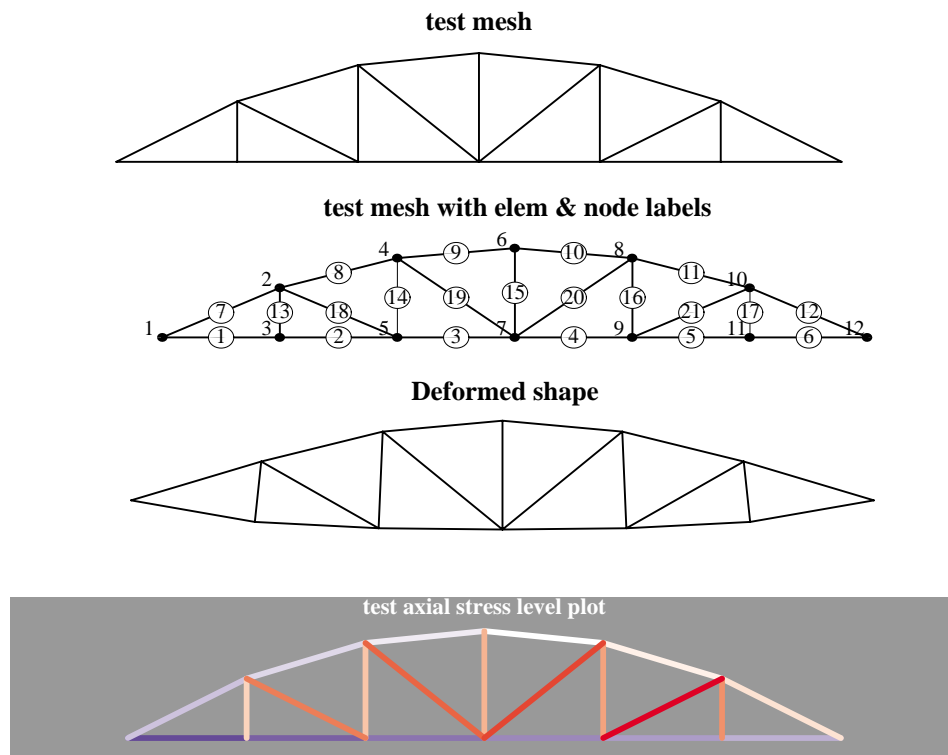


FIGURE 22.8. Graphics output from the bridge truss analysis.

Homework Exercises for Chapter 22

FEM Programs for Plane Trusses and Frames

EXERCISE 22.1

Placeholder.

EXERCISE 22.2 [C:25] Using the `PlaneTruss.nb` Notebook posted on the web site as guide, complete the `PlaneFrame.nb` Notebook that implements the analysis of an arbitrary plane frame using the plane beam stiffness presented in Chapter 21. To begin this homework, download the `PlaneFrame.nb` Notebook file from the web site.

The modification involves selected Cells of the Notebook, as described below. For additional details refer to the Notebook; some of the modifications involve only partially filled Modules.

Cell 1: Assembler. The element stiffness routine is the beam-column stiffness presented in Chapter 21. This module, called `PlaneBeamColumn2Stiffness`, is included in Cell 1 of the `PlaneFrame.nb` notebook linked from the Chapter 22 index.¹ In modifying the assembler module, remember that there are three degrees of freedom per node: u_{xi} , u_{yi} and θ_i , not just two.

The incomplete assembler is shown in Figure E22.1. The test statements are shown in red after the module. The lower cell shows the test output produced by a correctly completed module.

Cell 2: BC Applicator. No modifications are necessary. The two modules should work correctly for this problem since they don't assume anything about freedoms per nodes. The same test statements can be kept.

Cell 3: Internal Force Recovery. Both modules require modifications because the beam has 3 internal forces: (i) the axial force (which is recovered exactly as for bars), (ii) the bending moment $m_i = EI\kappa_i$ at end node i , and (iii) the bending moment $m_j = EI\kappa_j$ at end node j .² Furthermore the element displacement vector has six degrees of freedom, not just four. Test the modules on a simple beam problem.

The incomplete internal force module is shown in Figure E22.2. The test statements are shown in red after the module. The lower cell shows the test output produced by a correctly completed module.

Cell 4-5: Graphic Modules. These are now provided ready to use, and should not be touched.

Cell 6: Driver Program: use this for Exercise 22.3. Do not touch for this one.

As solution to this Exercise return a listing of the completed Cells 1 and 3, along with the output of the test statements for those cells.

EXERCISE 22.3

[C:25] Use the program developed in the previous Exercise to analyze the plane frame shown in Figure E22.3. The frame is fixed³ at A , B and C . It is loaded by a downward point load P at G and by a horizontal point load $\frac{1}{2}P$ at D . All members have the same cross section $a \times a$ for simplicity, and the material is the same.

¹ If you want to extract an individual cell from a Notebook to work on a separate file (a good idea for working on groups) select the cell, then pick `SaveSelectionAs -> Plain Text` from the Edit menu, give it a file name in the pop-up dialogue box, and save it.

² Two end moments are required because the beam element used here has linearly varying curvatures and hence moments. To recover the end moments refer to the pertinent equations in Chapter 13. The steps are as follows. For element e recover the transverse displacements \bar{u}_{yi} and \bar{u}_{yj} in the local element system $\bar{x}^{(e)}$, $\bar{y}^{(e)}$. Complete these with rotations θ_i and θ_j (which do not need to be transformed) to form the 4×1 beam local node displacement vector. Then apply the curvature-displacement relation (13.13) at $\xi = -1$ and $\xi = 1$ to get the two end curvatures $\kappa_i^{(e)}$ and $\kappa_j^{(e)}$, respectively. Finally multiply those curvatures by $E^{(e)}I_{zz}^{(e)}$ to get the bending moments.

³ For a plane frame, a fixed condition, also known as clamped condition, means that the x , y displacements and the rotation about z are zero.

```
(* PlaneBeamColumn2Stiffness module goes here *)

PlaneFrameMasterStiffness[nodcoor_,elenod_,
  elemat_,elefab_,eleopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodcoor],
  e,eNL,eftab,ni,nj,i,j,ncoor,mprop,fprop,opt,Ke,K},
  K=Table[0,{3*numnod},{3*numnod}];
  For [e=1, e<=numele, e++,
    (* missing statements *)
    Ke=PlaneBeamColumn2Stiffness[ncoor,mprop,fprop,opt];
    (* missing statements *)
  ];
  Return[K]
];

nodcoor={{0,0},{10,0},{10,10}};
elenod= {{1,2},{2,3},{1,3}}; elemat= Table[{100,0,0,0},{3}];
elefab= {{1,10},{1/2,10},{2*Sqrt[2],10}}; eleopt= {True};
K=PlaneFrameMasterStiffness[nodcoor,elenod,elemat,elefab,eleopt];
Print["Master Stiffness of Example Frame:"];
Print[K//MatrixForm]; Print[Chop[Eigenvalues[K]]];
```

Master Stiffness of Example Frame:

22.1213	7.87868	-21.2132	-10.	0.	0.	-12.1213	-7.87868	-21.2132
7.87868	24.1213	81.2132	0.	-12.	60.	-7.87868	-12.1213	21.2132
-21.2132	81.2132	682.843	0.	-60.	200.	21.2132	-21.2132	141.421
-10.	0.	0.	22.	0.	-60.	-12.	0.	-60.
0.	-12.	-60.	0.	17.	-60.	0.	-5.	0.
0.	60.	200.	-60.	-60.	800.	60.	0.	200.
-12.1213	-7.87868	21.2132	-12.	0.	60.	24.1213	7.87868	81.2132
-7.87868	-12.1213	-21.2132	0.	-5.	0.	7.87868	17.1213	-21.2132
-21.2132	21.2132	141.421	-60.	0.	200.	81.2132	-21.2132	682.843

{1121.7, 555.338, 531.652, 46.6129, 22.4971, 14.366, 0, 0, 0}

FIGURE E22.1. The incomplete assembler module for Exercise 22.2, with test statements in red. Module PlaneBeamColumn2Stiffness, which goes in the upper portion of the cell is omitted to save space; that module was listed in Figure 21.9. Output cell results are produced by a correct module.

The SI physical units to be used are: mm for lengths, N for forces, and $\text{MPa}=\text{N}/\text{mm}^2$ for elastic moduli. For the calculations use the following numerical data: $L = 10,000$ mm (10 m), $H = 6,000$ (6 m), $a = 500$ mm (0.5 m), $P = 4,800$ N, $E = 35,000$ MPa (high strength concrete). The member cross section area is $A = a^2$, and the flexural moment of inertia about the neutral axis is $I_{zz} = a^4/12$.

The recommended finite element discretization of two elements per member is shown in Figure E22.4.

As solution to this exercise list the driver program you used and the displacement and internal force outputs. The results of primary interest are:

1. The horizontal displacement at D , and the vertical displacement at G (mm).
2. The axial forces (N) in columns AD , BE and CF , with appropriate sign⁴.
3. The maximum bending moment (N.mm) over the floor members DE and EF , and the maximum bending moment in the columns AD , BE and CF .⁵

Provide deformed shape plot (but not the animation) and the frame stress level plots as part of the homework results. Note: the Notebook on the web contains some of the actual plots produced by the complete Notebook,

⁴ Plus for tension, minus for compression

⁵ The bending moment varies linearly over each element. It should be continuous at all joints except E .


```

PlaneFrameIntForces[nodcoor_,elenod_,elemat_,elefab_,
  eleopt_,u_]:= Module[{numele=Length[elenod],
    numnod=Length[nodcoor],e,eNL,eftab,ni,nj,i,
    ncoor,mprop,fprop,opt,ue,p},
  p=Table[0,{numele}]; ue=Table[0,{6}];
  For [e=1, e<=numele, e++,
    eNL=elenod[[e]]; {ni,nj}=eNL;
    ncoor={nodcoor[[ni]],nodcoor[[nj]]};
    mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
    eftab={3*ni-2,3*ni-1,3*ni,3*nj-2,3*nj-1,3*nj};
    For [i=1,i<=6,i++, ii=eftab[[i]]; ue[[i]]=u[[ii]]];
    p[[e]]=PlaneBeamColumn2IntForces[ncoor,mprop,fprop,opt,ue]
  ];
  Return[p]
];

PlaneBeamColumn2IntForces[ncoor_,mprop_,fprop_,opt_,ue_]:=
Module[{x1,x2,y1,y2,x21,y21,Em,Gm,rho,alpha,A,Izz,num,LL,L,
  dv1,cv1,cv2,pe=Table[0,{3}]],
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,rho,alpha}=mprop; {A,Izz}=fprop; {num}=opt;
  If [num,{x21,y21,Em,A,Izz}=N[{x21,y21,Em,A,Izz}]];
  (* missing statements for Exercise 22.2 *)
  Return[pe]
];

ClearAll[L,Em,A1,A2,Izz1,Izz2,Izz3,uz1,uz2,uz3];
nodcoor={{0,0},{L,0},{L,L}}; elenod= {{1,2},{2,3},{1,3}};
elemat= Table[{Em,0,0,0},{3}];
elefab= {{A1,Izz1},{A2,Izz2},{A3,Izz3}}; eleopt= {False};
u={0,uy1,0, 0,uy2,0, 0,uy3,0};
p=PlaneFrameIntForces[nodcoor,elenod,elemat,elefab,eleopt,u];
Print["Int Forces of Example Frame:"]; Print[p];

```

Int Forces of Example Frame:

$$\left\{ \left\{ 0, \frac{6 \text{ Em Izz1 } (-uy1 + uy2)}{L^2}, -\frac{6 \text{ Em Izz1 } (-uy1 + uy2)}{L^2} \right\}, \left\{ \frac{A2 \text{ Em } (-uy2 + uy3)}{L}, 0, 0 \right\}, \right. \\ \left. \left\{ \frac{A3 \text{ Em } (-uy1 + uy3)}{2 L}, \frac{3 \text{ Em Izz3 } (-uy1 + uy3)}{\sqrt{2} L^2}, -\frac{3 \text{ Em Izz3 } (-uy1 + uy3)}{\sqrt{2} L^2} \right\} \right\}$$

FIGURE E22.2. Figure E22.2. Incomplete element internal force recovery module for Exercise 22.2. Test statements in red. Results in output cell are those produced by a correct module.

to be used as targets.

Partial answers for this exercise:

Largest vertical displacement: -0.232 mm (\Downarrow) at node 4

Largest negative bending moment: -6.8×10^6 N.mm, at node 5 of element (4)

Axial forces: -2651 N over (3) and (4)

EXERCISE 22.4 [D:20] Explain why the solution given by the FEM model of Figure E22.3 is exact for the Bernoulli-Euler bending model; that is, cannot be improved by subdividing each element into more elements.

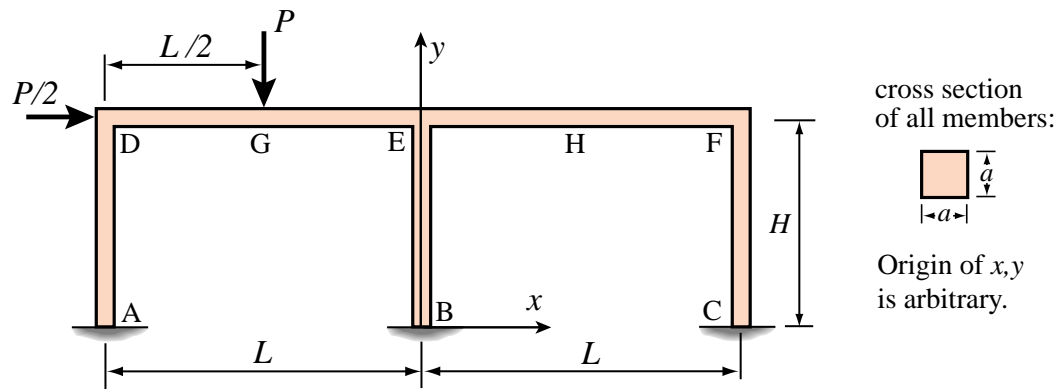


FIGURE E22.3. Plane frame structure for Exercise 22.3.

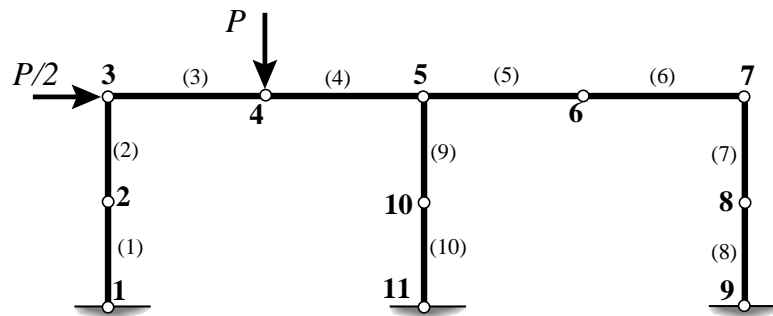


FIGURE E22.4. Recommended FEM discretization for the plane frame of the previous figure.

23

Implementation of Iso-P Quadrilateral Elements

TABLE OF CONTENTS

	Page
§23.1. Introduction	23-3
§23.2. Bilinear Quadrilateral Stiffness Matrix	23-3
§23.2.1. Gauss Quadrature Rule Information	23-3
§23.2.2. Shape Function Evaluation	23-5
§23.2.3. Element Stiffness	23-5
§23.3. Test of Bilinear Quadrilateral	23-7
§23.3.1. Test of Rectangular Geometry	23-7
§23.3.2. Test of Trapezoidal Geometry	23-8
§23.4. *Consistent Node Forces for Body Force Field	23-10
§23.5. *Recovery of Corner Stresses	23-10
§23.6. *Quadrilateral Coordinates of Given Point	23-12
§23. Notes and Bibliography	23-12
§23. References	23-13
§23. Exercises	23-14

§23.1. Introduction

This Chapter illustrates, through a specific example, the computer implementation of isoparametric *quadrilateral* elements for the plane stress problem. Triangles, which present some programming quirks, are covered in the next Chapter.

The programming example is that of the four-node bilinear quadrilateral. It covers the computation of the element stiffness matrix, the consistent node force vector for a body forces, and stress recovery at selected points. The organization of the computations is typical of isoparametric-element modules in any number of space dimensions.

§23.2. Bilinear Quadrilateral Stiffness Matrix

We consider here the implementation of the 4-node bilinear quadrilateral for plane stress, depicted in Figure 23.1. The element stiffness matrix of simple one-dimensional elements, such as the ones discussed in Chapters 20–22, can be easily packaged in a simple module. For multidimensional elements, however, it is convenient to break up the implementation into *application dependent* and *application independent* modules, as flowcharted in Figure 23.2. The application independent modules can be “reused” in other FEM applications, for example to construct thermal, fluid or electromagnetic elements.

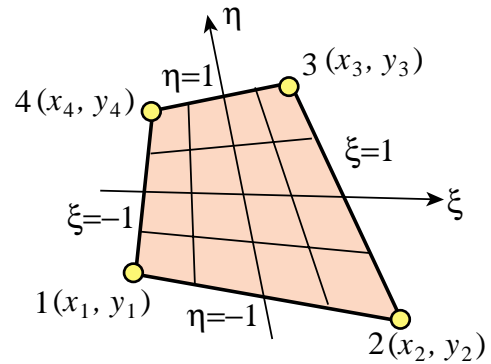


FIGURE 23.1. The 4-node bilinear quadrilateral element.

For the bilinear quadrilateral stiffness computations, the separation of Figure 23.2 is done by dividing the work into three modules:

Quad4IsoPMembraneStiffness. Computes the element stiffness matrix \mathbf{K}^e of a four-node isoparametric quadrilateral element in plane stress.

QuadGuassRuleInfo. Returns two-dimensional Gauss quadrature formula of product type.

Quad4IsoPShapeFunDer. Evaluates the shape functions of a four-node isoparametric quadrilateral and their x/y derivatives, at a specific point.

These modules are described in further detail in the following subsections, in a “bottom up” fashion.

§23.2.1. Gauss Quadrature Rule Information

Recall from §17.3 that Gauss quadrature rules for isoparametric quadrilateral elements have the canonical form

$$\int_{-1}^1 \int_{-1}^1 \mathbf{F}(\xi, \eta) d\xi d\eta = \int_{-1}^1 d\eta \int_{-1}^1 \mathbf{F}(\xi, \eta) d\xi \doteq \sum_{i=1}^{p_1} \sum_{j=1}^{p_2} w_i w_j \mathbf{F}(\xi_i, \eta_j). \quad (23.1)$$

Here $\mathbf{F} = h\mathbf{B}^T \mathbf{E} \mathbf{B} J$ is the matrix to be integrated, whereas p_1 and p_2 are the number of Gauss points in the ξ and η directions, respectively. Often, but not always, the same number $p = p_1 = p_2$ is chosen in both directions. A formula with $p_1 = p_2$ is called an *isotropic integration rule* because directions ξ and η are treated alike.

QuadGaussRuleInfo is an application independent module that implements the two-dimensional product Gauss rules with 1 through 5 points in each direction. The number of points in each direction may be the same or different. Usage of this module was described in detail in §17.3.4. For the readers convenience it is listed, along with its subordinate module LineGaussRuleInfo, in Figure 23.3.

This module is classified as application independent since it can reused for any quadrilateral element.

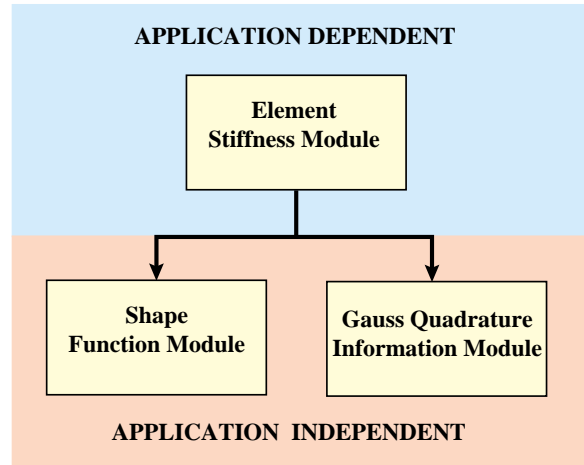


FIGURE 23.2. Separation of element stiffness modules into application-dependent and application-independent levels.

```

QuadGaussRuleInfo[{rule_,numer_},point_]:= Module[
{ξ,η,p1,p2,i,j,w1,w2,m,info={{Null,Null},0}},
If [Length[rule]==2, {p1,p2}=rule, p1=p2=rule];
If [p1<0, Return[QuadNonProductGaussRuleInfo[
{-p1,numer_},point]]];
If [Length[point]==2, {i,j}=point, m=point;
j=Floor[(m-1)/p1]+1; i=m-p1*(j-1)];
{ξ,w1}= LineGaussRuleInfo[{p1,numer_},i];
{η,w2}= LineGaussRuleInfo[{p2,numer_},j];
info={{ξ,η},w1*w2};
If [numer, Return[N[info]], Return[Simplify[info]]];
];

LineGaussRuleInfo[{rule_,numer_},point_]:= Module[
{g2={-1,1}/Sqrt[3],w3={5/9,8/9,5/9},
g3={-Sqrt[3/5],0,Sqrt[3/5]},
w4={{(1/2)-Sqrt[5/6]}/6, (1/2)+Sqrt[5/6]}/6,
(1/2)+Sqrt[5/6]}/6, (1/2)-Sqrt[5/6]}/6},
g4={-Sqrt[(3+2*Sqrt[6/5])/7],-Sqrt[(3-2*Sqrt[6/5])/7],
Sqrt[(3-2*Sqrt[6/5])/7], Sqrt[(3+2*Sqrt[6/5])/7]},
g5={-Sqrt[5+2*Sqrt[10/7]],-Sqrt[5-2*Sqrt[10/7]],0,
Sqrt[5-2*Sqrt[10/7]], Sqrt[5+2*Sqrt[10/7]]}/3,
w5={322-13*Sqrt[70],322+13*Sqrt[70],512,
322+13*Sqrt[70],322-13*Sqrt[70]}/900,
i=point,p=rule,info={{Null,Null},0}},
If [p==1, info={0,2}];
If [p==2, info={g2[[i]],1}];
If [p==3, info={g3[[i]],w3[[i]]}];
If [p==4, info={g4[[i]],w4[[i]]}];
If [p==5, info={g5[[i]],w5[[i]]}];
If [numer, Return[N[info]], Return[Simplify[info]]];
];
  
```

FIGURE 23.3. Module to get Gauss-product quadrature information for a quadrilateral.

```

Quad4IsoPShapeFunDer[ncoor_,qcoor_]:= Module[
{Nf,dNx,dNy,dNξ,dNη,i,J11,J12,J21,J22,Jdet,ξ,η,x,y},
{ξ,η}=qcoor;
Nf={ (1-ξ)*(1-η), (1+ξ)*(1-η), (1+ξ)*(1+η), (1-ξ)*(1+η) }/4;
dNξ = { -(1-η), (1-η), (1+η), -(1+η) }/4;
dNη= { -(1-ξ), -(1+ξ), (1+ξ), (1-ξ) }/4;
x=Table[ncoor[[i,1]],{i,4}]; y=Table[ncoor[[i,2]],{i,4}];
J11=dNξ.x; J12=dNξ.y; J21=dNη.x; J22=dNη.y;
Jdet=Simplify[J11*J22-J12*J21];
dNx= ( J22*dNξ-J12*dNη)/Jdet; dNx=Simplify[dNx];
dNy= (-J21*dNξ+J11*dNη)/Jdet; dNy=Simplify[dNy];
Return[ {Nf,dNx,dNy,Jdet} ]
];

```

FIGURE 23.4. Shape function module for 4-node bilinear quadrilateral.

§23.2.2. Shape Function Evaluation

Quad4IsoPShapeFunDer is an application independent module that computes the shape functions N_i^e , $i = 1, 2, 3, 4$ and its x - y partial derivatives at the sample integration points. The logic, listed in Figure 23.4, is straightforward and follows closely the description of Chapter 17.

The arguments of the module are the $\{x, y\}$ quadrilateral corner coordinates, which are passed in `ncoor`, and the two quadrilateral coordinates $\{\xi, \eta\}$, which are passed in `qcoor`. The former have the same configuration as described for the element stiffness module below.

The quadrilateral coordinates define the element location at which the shape functions and their derivatives are to be evaluated. For the stiffness formation these are Gauss points, but for strain and stress computations these may be other points, such as corner nodes.

Quad4IsoPShapeFunDer returns the two-level list $\{Nf, Nx, Ny, Jdet\}$, in which the first three are 4-entry lists. List `Nf` collects the shape function values, `Nx` the shape function x -derivatives, `Ny` the shape function y -derivatives, and `Jdet` is the Jacobian determinant called J in Chapters 17.

§23.2.3. Element Stiffness

Module Quad4IsoPMembraneStiffness computes the stiffness matrix of a four-noded isoparametric quadrilateral element in plane stress. The module configuration is typical of isoparametric elements in any number of dimensions. It follows closely the procedure outlined in Chapter 17. The module logic is listed in Figure 23.5. The statements at the bottom of the module box (not shown in that Figure) test it for specific configurations.

The module is invoked as

$$Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,options] \quad (23.2)$$

The arguments are:

`ncoor` Quadrilateral node coordinates arranged in two-dimensional list form:
 $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}\}.$

```

Quad4IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
Module[{i,k,p=2,numer=False,h=th,qcoor,c,w,Nf,
  dNx,dNy,Jdet,Be,Ke=Table[0,{8},{8}]},
If [Length[options]==2, {numer,p}=options,{numer}=options];
If [p<1||p>4, Print["p out of range"]; Return[Null]];
For [k=1, k<=p*p, k++,
  {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
  {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
  If [Length[th]==4, h=th.Nf]; c=w*Jdet*h;
  Be={Flatten[Table[{dNx[[i]], 0},{i,4}]],
    Flatten[Table[{0, dNy[[i]]},{i,4}]],
    Flatten[Table[{dNy[[i]],dNx[[i]]},{i,4}]]};
  Ke+=Simplify[c*Transpose[Be].(Emat.Be)];
]; Return[Simplify[Ke]]
];

```

FIGURE 23.5. Element stiffness formation module for 4-node bilinear quadrilateral.

Emat A two-dimensional list storing the 3×3 plane stress matrix of elastic moduli:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \quad (23.3)$$

arranged as $\{\{E_{11}, E_{12}, E_{33}\}, \{E_{12}, E_{22}, E_{23}\}, \{E_{13}, E_{23}, E_{33}\}\}$. This matrix must be symmetric. If the material is isotropic with elastic modulus E and Poisson's ratio ν , it reduces to

$$\mathbf{E} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix} \quad (23.4)$$

th The plate thickness specified either as a four-entry list: $\{h_1, h_2, h_3, h_4\}$ or as a scalar: h .

The first form is used to specify an element of variable thickness, in which case the entries are the four corner thicknesses and h is interpolated bilinearly. The second form specifies uniform thickness.

options Processing options. This list may contain two items: $\{\text{numer}, p\}$ or one: $\{\text{numer}\}$. **numer** is a logical flag with value **True** or **False**. If **True**, the computations are done in floating point arithmetic. For symbolic or exact arithmetic work set **numer** to **False**.¹

p specifies the Gauss product rule to have p points in each direction. p may be 1 through 4. For rank sufficiency, p must be 2 or higher. If p is 1 the element will be rank deficient by two.² If omitted $p = 2$ is assumed.

¹ The reason for this option is speed. A symbolic or exact computation can take orders of magnitude more time than a floating-point evaluation. This becomes more pronounced as elements get more complicated.

² The rank of an element stiffness is discussed in Chapter 19.

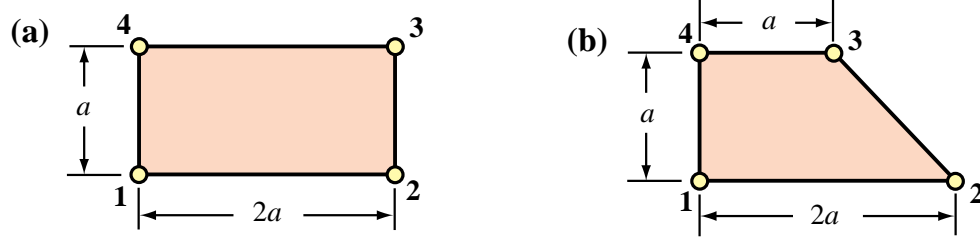


FIGURE 23.6. Test quadrilateral element geometries.

The module returns \mathbf{K}^e as an 8×8 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^e = [u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4}]^T. \quad (23.5)$$

§23.3. Test of Bilinear Quadrilateral

The stiffness module is tested on the two quadrilateral geometries shown in Figure 23.6. Both elements have unit thickness and isotropic material. The left one is a rectangle of base $2a$ and height a . The right one is a right trapezoid with base $2a$, top width a and height a .

The two geometries will be used to illustrate the effect of the numerical integration rule.

§23.3.1. Test of Rectangular Geometry

The script listed in Figure 23.7 computes and displays the stiffness of the rectangular element shown in Figure 23.6(a). This is a rectangle of base $2a$ and height a . The plate has unit thickness and isotropic material with $E = 96$ and $\nu = 1/3$, giving the stress-strain constitutive matrix

$$\mathbf{E} = \begin{bmatrix} 108 & 36 & 0 \\ 36 & 108 & 0 \\ 0 & 0 & 36 \end{bmatrix} \quad (23.6)$$

Using a 2×2 Gauss integration rule returns the stiffness matrix

$$\mathbf{K}^e = \begin{bmatrix} 42 & 18 & -6 & 0 & -21 & -18 & -15 & 0 \\ 18 & 78 & 0 & 30 & -18 & -39 & 0 & -69 \\ -6 & 0 & 42 & -18 & -15 & 0 & -21 & 18 \\ 0 & 30 & -18 & 78 & 0 & -69 & 18 & -39 \\ -21 & -18 & -15 & 0 & 42 & 18 & -6 & 0 \\ -18 & -39 & 0 & -69 & 18 & 78 & 0 & 30 \\ -15 & 0 & -21 & 18 & -6 & 0 & 42 & -18 \\ 0 & -69 & 18 & -39 & 0 & 30 & -18 & 78 \end{bmatrix} \quad (23.7)$$

Note that the rectangle dimension a does not appear in (23.7). This is a general property: *the stiffness matrix of plane stress elements is independent of in-plane dimension scalings*. This follows from the fact that entries of the strain-displacement matrix \mathbf{B} have dimensions $1/L$, where L denotes a

```

ClearAll[Em,nu,a,b,e,h,p,number]; h=1;
Em=96; nu=1/3; (* isotropic material *)
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
Print["Emat=",Emat//MatrixForm];
ncoor={{0,0},{2*a,0},{2*a,a},{0,a}}; (* 2:1 rectangular geometry *)
p=2;(* 2 x 2 Gauss rule *)number=False;(* exact symbolic arithmetic *)
Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,h,{number,p}];
Ke=Simplify[Chop[Ke]]; Print["Ke=",Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]];

```

FIGURE 23.7. Driver for stiffness calculation of rectangular element of Figure 23.6(a) for 2×2 Gauss rule.

characteristic inplane length. Consequently entries of $\mathbf{B}^T \mathbf{B}$ have dimension $1/L^2$. Integration over the element area cancels out L^2 .

Using a higher order Gauss integration rule, such as 3×3 and 4×4 , reproduces exactly (23.7). This is a property characteristic of the rectangular geometry, since in that case the entries of \mathbf{B} vary linearly in ξ and η , and J is constant. Therefore the integrand $h \mathbf{B}^T \mathbf{E} \mathbf{B} J$ is at most quadratic in ξ and η , and 2 Gauss points in each direction suffice to compute the integral exactly. Using a 1×1 rule yields a rank-deficiency matrix, a result illustrated in detail in §23.2.2.

The stiffness matrix (23.7) has the eigenvalues

$$[223.64 \quad 90 \quad 78 \quad 46.3603 \quad 42 \quad 0 \quad 0 \quad 0] \quad (23.8)$$

This verifies that \mathbf{K}^e has the correct rank of five (8 total DOFs minus 3 rigid body modes).

§23.3.2. Test of Trapezoidal Geometry

```

ClearAll[Em,nu,h,a,p]; h=1;
Em=48*63*13*107; nu=1/3;
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
ncoor={{0,0},{2*a,0},{a,a},{0,a}};
For [p=1,p<=4,p++,
  Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,h,{True,p}];
  Ke=Rationalize[Ke,0.0000001]; Print["Ke=",Ke//MatrixForm];
  Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]]
];

```

FIGURE 23.8. Driver for stiffness calculation of trapezoidal element of Figure 23.6(b) for four Gauss integration rules.

The trapezoidal element geometry of Figure 23.6(b) is used to illustrate the effect of changing the $p \times p$ Gauss integration rule. Unlike the rectangular case, the element stiffness keeps changing as p is varied from 1 to 4. The element is rank sufficient, however, for $p \geq 2$ in agreement with the analysis of Chapter 19.

The computations are driven with the script shown in Figure 23.8. The value of p is changed in a loop. The flag `number` is set to `True` to use floating-point computation for speed (see Remark 23.1). The computed entries of \mathbf{K}^e are transformed to the nearest rational number (exact integers in this case) using the built-in function `Rationalize`. The strange value of $E = 48 \times 63 \times 13 \times 107 = 4206384$,

in conjunction with $\nu = 1/3$, makes all entries of \mathbf{K}^e exact integers when computed with the first 4 Gauss rules. This device facilitates visual comparison between the computed stiffness matrices:

$$\mathbf{K}_{1 \times 1}^e = \begin{bmatrix} 1840293 & 1051596 & -262899 & -262899 & -1840293 & -1051596 & 262899 & 262899 \\ 1051596 & 3417687 & -262899 & 1314495 & -1051596 & -3417687 & 262899 & -1314495 \\ -262899 & -262899 & 1051596 & -525798 & 262899 & 262899 & -1051596 & 525798 \\ -262899 & 1314495 & -525798 & 1051596 & 262899 & -1314495 & 525798 & -1051596 \\ -1840293 & -1051596 & 262899 & 262899 & 1840293 & 1051596 & -262899 & -262899 \\ -1051596 & -3417687 & 262899 & -1314495 & 1051596 & 3417687 & -262899 & 1314495 \\ 262899 & 262899 & -1051596 & 525798 & -262899 & -262899 & 1051596 & -525798 \\ 262899 & -1314495 & 525798 & -1051596 & -262899 & 1314495 & -525798 & 1051596 \end{bmatrix} \quad (23.9)$$

$$\mathbf{K}_{2 \times 2}^e = \begin{bmatrix} 2062746 & 1092042 & -485352 & -303345 & -1395387 & -970704 & -182007 & 182007 \\ 1092042 & 3761478 & -303345 & 970704 & -970704 & -2730105 & 182007 & -2002077 \\ -485352 & -303345 & 1274049 & -485352 & -182007 & 182007 & -606690 & 606690 \\ -303345 & 970704 & -485352 & 1395387 & 182007 & -2002077 & 606690 & -364014 \\ -1395387 & -970704 & -182007 & 182007 & 2730105 & 1213380 & -1152711 & -424683 \\ -970704 & -2730105 & 182007 & -2002077 & 1213380 & 4792851 & -424683 & -60669 \\ -182007 & 182007 & -606690 & 606690 & -1152711 & -424683 & 1941408 & -364014 \\ 182007 & -2002077 & 606690 & -364014 & -424683 & -60669 & -364014 & 2426760 \end{bmatrix} \quad (23.10)$$

$$\mathbf{K}_{3 \times 3}^e = \begin{bmatrix} 2067026 & 1093326 & -489632 & -304629 & -1386827 & -968136 & -190567 & 179439 \\ 1093326 & 3764046 & -304629 & 968136 & -968136 & -2724969 & 179439 & -2007213 \\ -489632 & -304629 & 1278329 & -484068 & -190567 & 179439 & -598130 & 609258 \\ -304629 & 968136 & -484068 & 1397955 & 179439 & -2007213 & 609258 & -358878 \\ -1386827 & -968136 & -190567 & 179439 & 2747225 & 1218516 & -1169831 & -429819 \\ -968136 & -2724969 & 179439 & -2007213 & 1218516 & 4803123 & -429819 & -70941 \\ -190567 & 179439 & -598130 & 609258 & -1169831 & -429819 & 1958528 & -358878 \\ 179439 & -2007213 & 609258 & -358878 & -429819 & -70941 & -358878 & 2437032 \end{bmatrix} \quad (23.11)$$

$$\mathbf{K}_{4 \times 4}^e = \begin{bmatrix} 2067156 & 1093365 & -489762 & -304668 & -1386567 & -968058 & -190827 & 179361 \\ 1093365 & 3764124 & -304668 & 968058 & -968058 & -2724813 & 179361 & -2007369 \\ -489762 & -304668 & 1278459 & -484029 & -190827 & 179361 & -597870 & 609336 \\ -304668 & 968058 & -484029 & 1398033 & 179361 & -2007369 & 609336 & -358722 \\ -1386567 & -968058 & -190827 & 179361 & 2747745 & 1218672 & -1170351 & -429975 \\ -968058 & -2724813 & 179361 & -2007369 & 1218672 & 4803435 & -429975 & -71253 \\ -190827 & 179361 & -597870 & 609336 & -1170351 & -429975 & 1959048 & -358722 \\ 179361 & -2007369 & 609336 & -358722 & -429975 & -71253 & -358722 & 2437344 \end{bmatrix} \quad (23.12)$$

As can be seen entries change substantially in going from $p = 1$ to $p = 2$, then more slowly. The eigenvalues of these matrices are:

Rule	Eigenvalues (scaled by 10^{-6}) of \mathbf{K}^e							
1×1	8.77276	3.68059	2.26900	0	0	0	0	0
2×2	8.90944	4.09769	3.18565	2.64521	1.54678	0	0	0
3×3	8.91237	4.11571	3.19925	2.66438	1.56155	0	0	0
4×4	8.91246	4.11627	3.19966	2.66496	1.56199	0	0	0

(23.13)

The stiffness matrix computed by the one-point rule is rank deficient by two. The eigenvalues do not change appreciably after $p = 2$. Because the nonzero eigenvalues measure the internal energy taken up by the element in deformation eigenmodes, it can be seen that raising the order of the integration makes the element stiffer.

```

Quad4IsoPMembraneBodyForces[ncoor_,rho_,th_,options_,bfor_]:=
Module[{i,k,p=2,numer=False,h=th,
  bx,by,bx1,by1,bx2,by2,bx3,by3,bx4,by4,bxc,byc,qcoor,
  c,w,Nf,dNx,dNy,Jdet,B,qctab,fe=Table[0,{8}]},
  If [Length[options]==2, {numer,p}=options, {numer}=options];
  If
  [Length[bfor]==2,{bx,by}=bfor;bx1=bx2=bx3=bx4=bx;by1=by2=by3=by4=by];
  If [Length[bfor]==4,{bx1,by1},{bx2,by2},{bx3,by3},{bx4,by4}=bfor];
  If [p<1||p>4, Print["p out of range"]; Return[Null]];
  bxc={bx1,bx2,bx3,bx4}; byc={by1,by2,by3,by4};
  For [k=1, k<=p*p, k++,
    {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
    {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
    bx=Nf.bxc; by=Nf.byc; If [Length[th]==4, h=th.Nf];
    c=w*Jdet*h;
    bk=Flatten[Table[{Nf[[i]]*bx,Nf[[i]]*by},{i,4}]];
    fe+=c*bk;
  ]; Return[fe]
];

```

FIGURE 23.9. Module for computation of consistent node forces from a given body force field.

Remark 23.1.

The formation of the trapezoidal element stiffness using floating-point computation by setting `numer=True` took 0.017, 0.083, 0.15 and 0.25 seconds for $p = 1, 2, 3, 4$, respectively, on a Mac G4/867. Changing `numer=False` to do exact computation increases the formation time to 0.033, 1.7, 4.4 and 44.6 seconds, respectively. (The unusually large value for $p = 4$ is due to the time spent in the simplification of the highly complex exact expressions produced by the Gauss quadrature rule.) This underscores the speed advantage of using floating-point arithmetic when exact symbolic and algebraic calculations are not required.

§23.4. *Consistent Node Forces for Body Force Field

The module `Quad4IsoPMembraneBodyForces` listed in Figure 23.9 computes the consistent force associated with a body force field $\vec{\mathbf{b}} = \{b_x, b_y\}$ given over a 4-node iso-P quadrilateral in plane stress. The field is specified per unit of volume in componentwise form. For example if the element is subjected to a gravity acceleration field (self-weight) in the $-y$ direction, $b_x = 0$ and $b_y = -\rho g$, where ρ is the mass density.

The arguments of the module are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

<code>mprop</code>	Not used; retained as placeholder.
<code>bfor</code>	Body forces per unit volume. Specified as a two-item one-dimensional list: $\{b_x, b_y\}$, or as a four-entry two-dimensional list: $\{b_{x1}, b_{y1}\}, \{b_{x2}, b_{y2}\}, \{b_{x3}, b_{y3}\}, \{b_{x4}, b_{y4}\}$. In the first form the body force field is taken to be constant over the element. The second form assumes body forces to vary over the element and specified by values at the four corners, from which the field is interpolated bilinearly.

The module returns `fe` as an 8×1 one dimensional array arranged $\{f_{x1}, f_{y1}, f_{x2}, f_{y2}, f_{x3}, f_{y3}, f_{x4}, f_{y4}\}$ to represent the vector

$$\mathbf{f}^e = [f_{x1} \ f_{y1} \ f_{x2} \ f_{y2} \ f_{x3} \ f_{y3} \ f_{x4} \ f_{y4}]^T. \quad (23.14)$$

```

Quad4IsoPMembraneStresses[ncoor_,Emat_,th_,options_,udis_]:=
Module[{i,k,numer=False,qcoor,Nf,
  dNx,dNy,Jdet,Be,qctab,ue=udis,sige=Table[0,{4},{3}]},
qctab={{-1,-1},{1,-1},{1,1},{-1,1}};
numer=options[[1]];
If [Length[udis]==4, ue=Flatten[udis]];
For [k=1, k<=Length[sige], k++,
  qcoor=qctab[[k]]; If [numer, qcoor=N[qcoor]];
  {Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
  Be={ Flatten[Table[{dNx[[i]], 0},{i,4}]],
    Flatten[Table[{0, dNy[[i]]},{i,4}]],
    Flatten[Table[{dNy[[i]],dNx[[i]]},{i,4}]]};
  sige[[k]]=Emat.(Be.ue);
]; Return[sige]
];

```

FIGURE 23.10. Module for calculation of corner stresses.

§23.5. *Recovery of Corner Stresses

Although the subject of stress recovery is treated in further detail in a later chapter, for completeness a stress computation module called `Quad4IsoPMembraneStresses` for the 4-node quad is listed in Figure 23.10.

The arguments of the module are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

- | | |
|--------------------|--|
| <code>fprop</code> | Not used; retained as placeholder. |
| <code>udis</code> | The 8 corner displacements components. these may be specified as a 8-entry one-dimensional list form:
$\{ux1, uy1, ux2, uy2, ux3, uy3, ux4, uy4\}$,
or as a 4-entry two-dimensional list:
$\{ux1, uy1\}, \{ux2, uy2\}, \{ux3, uy3\}, \{ux4, uy4\}$. |

The module returns the corner stresses stored in a 4-entry, two-dimensional list:

$\{\{sigxx1, sigyy1, sigxy1\}, \{sigxx2, sigyy2, sigxy2\}, \{sigxx3, sigyy3, sigxy3\}, \{sigxx4, sigyy4, sigxy4\}\}$ to represent the stress array

$$\sigma^e = \begin{bmatrix} \sigma_{xx1} & \sigma_{xx2} & \sigma_{xx3} & \sigma_{xx4} \\ \sigma_{yy1} & \sigma_{yy2} & \sigma_{yy3} & \sigma_{yy4} \\ \sigma_{xy1} & \sigma_{xy2} & \sigma_{xy3} & \sigma_{xy4} \end{bmatrix} \quad (23.15)$$

The stresses are directly evaluated at the corner points without invoking any smoothing procedure. A more elaborated recovery scheme is presented in a later Chapter.

§23.6. *Quadrilateral Coordinates of Given Point

The following inverse problem arises in some applications. Given a 4-node quadrilateral, defined by the Cartesian coordinates $\{x_i, y_i\}$ of its corners, and an arbitrary point $P(x_P, y_P)$, find the quadrilateral coordinates ξ_P, η_P of P . In answering this question it is understood that the quadrilateral coordinates can be extended outside the element, as illustrated in Figure 23.11.

The governing equations are $x_P = x_1 N_1 + x_2 N_2 + x_3 N_3 + x_4 N_4$ and $y_P = y_1 N_1 + y_2 N_2 + y_3 N_3 + y_4 N_4$, where $N_1 = \frac{1}{4}(1 - \xi_P)(1 - \eta_P)$, etc. These bilinear equations are to be solved for $\{\xi_P, \eta_P\}$. Elimination of say, ξ_P , leads to a quadratic equation in η_P : $a\eta_P^2 + b\eta_P + c = 0$. It can be shown that $b^2 \geq 4ac$ so there are two real roots: η_1 and η_2 . These can be back substituted to get ξ_1 and ξ_2 . Of the two solutions: $\{\xi_1, \eta_1\}$ and $\{\xi_2, \eta_2\}$ the one closest to $\xi = 0, \eta = 0$ is to be taken.

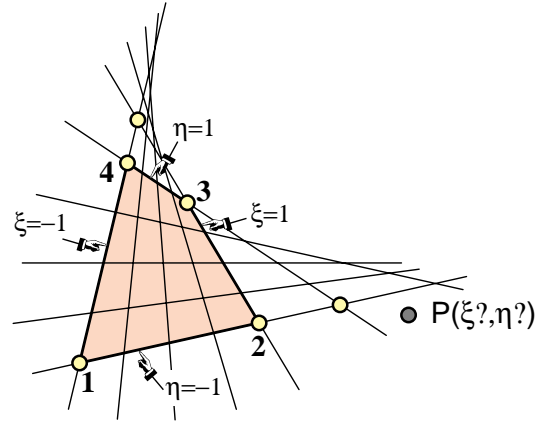


FIGURE 23.11. Quadrilateral coordinates can be extended outside the element to answer the problem posed in §23.6. The six yellow-filled circles identify the four corners plus the intersections of the opposite sides. This six-point set defines the so-called complete quadrilateral, which is important in projective geometry.

The evolute of the coordinate lines is a parabola.

Although seemingly straightforward, the process is prone to numerical instabilities. For example, if the quadrilateral becomes a rectangle or parallelogram, the quadratic equations degenerate to linear, and one of the roots takes off to ∞ . In floating point arithmetic severe cancellation can occur in the other root. A robust numerical algorithm, which works stably for any geometry, is obtained by eliminating ξ and η in turn, getting the minimum-modulus root of $a\eta_P^2 + b\eta_P + c = 0$ with the stable formula.³ $\eta_P^{min} = b/(b + \sqrt{b^2 - 4ac})$, forming the other quadratic equation, and computing its minimum-modulus root the same way. In addition, x_P and y_P are referred to the quadrilateral center as coordinate origin. The resulting algorithm can be presented as follows. Given $\{x_1, y_1, \dots, x_4, y_4\}$ and $\{x_P, y_P\}$, compute

$$\begin{aligned}
 x_b &= x_1 - x_2 + x_3 - x_4, & y_b &= y_1 - y_2 + y_3 - y_4, & x_{cx} &= x_1 + x_2 - x_3 - x_4, & y_{cx} &= y_1 + y_2 - y_3 - y_4, \\
 x_{ce} &= x_1 - x_2 - x_3 + x_4, & y_{ce} &= y_1 - y_2 - y_3 + y_4, & A &= \frac{1}{2}((x_3 - x_1)(y_4 - y_2) - (x_4 - x_2)(y_3 - y_1)), \\
 J_1 &= (x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4), & J_2 &= (x_2 - x_3)(y_1 - y_4) - (x_1 - x_4)(y_2 - y_3), \\
 x_0 &= \frac{1}{4}(x_1 + x_2 + x_3 + x_4), & y_0 &= \frac{1}{4}(y_1 + y_2 + y_3 + y_4), & x_{P0} &= x_P - x_0, & y_{P0} &= y_P - y_0, \\
 b_\xi &= A - x_{P0} y_b + y_{P0} x_b, & b_\eta &= -A - x_{P0} y_b + y_{P0} x_b, & c_\xi &= x_{P0} y_{cx} - y_{P0} x_{cx}, \\
 c_\eta &= x_{P0} y_{ce} - y_{P0} x_{ce}, & \xi_P &= \frac{2c_\xi}{-\sqrt{b_\xi^2 - 2J_1 c_\xi} - b_\xi}, & \eta_P &= \frac{2c_\eta}{\sqrt{b_\eta^2 + 2J_2 c_\eta} - b_\eta}.
 \end{aligned}$$

(23.16)

One common application is to find whether P is inside the quadrilateral: if both ξ_P and η_P are in the range $[-1, 1]$ the point is inside, else outside. This occurs, for example, in relating experimental data from given sensor locations⁴ to an existing FEM mesh.

A *Mathematica* module that implements (23.16) is listed in Figure 23.12.

³ See Section 5.6 of [320].

⁴ While at Boeing in 1969 the writer had to solve a collocation problem of this nature, although in three dimensions. Pressure data measured at a wind tunnel had to be transported to an independently constructed FEM quadrilateral mesh modeling the wing skin.

```

QuadCoordinatesOfPoint[{{x1_,y1_},{x2_,y2_},{x3_,y3_},
  {x4_,y4_}},{x_,y_}]:= Module[{A,J0,J1,J2,
  xb=x1-x2+x3-x4,yb=y1-y2+y3-y4,xcξ=x1+x2-x3-x4,ycξ=y1+y2-y3-y4,
  xcη=x1-x2-x3+x4,ycη=y1-y2-y3+y4,bξ,bη,cξ,cη,
  x0=(x1+x2+x3+x4)/4,y0=(y1+y2+y3+y4)/4,dx,dy,ξ,η},
  J0=(x3-x1)*(y4-y2)-(x4-x2)*(y3-y1); A=J0/2;
  J1=(x3-x4)*(y1-y2)-(x1-x2)*(y3-y4);
  J2=(x2-x3)*(y1-y4)-(x1-x4)*(y2-y3);
  dx=x-x0; dy=y-y0;
  bξ=A-dx*yb+dy*xb; bη=-A-dx*yb+dy*xb;
  cξ= dx*ycξ-dy*xcξ; cη=dx*ycη-dy*xcη;
  ξ=2*cξ/(-Sqrt[bξ^2-2*J1*cξ]-bξ);
  η=2*cη/( Sqrt[bη^2+2*J2*cη]-bη);
  Return[{ξ,η}]]];

```

FIGURE 23.12. A *Mathematica* module that implements the algorithm (23.16).

Notes and Bibliography

For an outline of the history of the 4-node quadrilateral, see **Notes and Bibliography** in Chapter 17. The element is called the Taig quadrilateral in the early FEM literature, recognizing his developer [371]. This paper actually uses the exactly integrated stiffness matrix.

Gauss numerical integration for quadrilaterals was advocated by Irons [214,218], who changed the range of the quadrilateral coordinates to $[-1, +1]$ to fit tabulated Gauss rules.

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 23

Implementation of Iso-P Quadrilateral Elements

EXERCISE 23.1 [C:15] Figures E23.1–2 show the *Mathematica* implementation of the stiffness modules for the 5-node, “bilinear+bubble” iso-P quadrilateral of Figure E18.3. Module `Quad5IsoPMembraneStiffness` returns the 10×10 stiffness matrix whereas module `Quad5IsoPShapeFunDer` returns shape function values and Cartesian derivatives. (The Gauss quadrature module is reused.) Both modules follow the style of the 4-node quadrilateral implementation listed in Figures 23.4–5. The only differences in argument lists is that `ncoor` has five node coordinates: $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \{x_4, y_4\}, \{x_5, y_5\}\}$, and that a variable plate thickness in `fprop` (one of the two possible formats) is specified as $\{h_1, h_2, h_3, h_4, h_5\}$.

```
Quad5IsoPMembraneStiffness[ncoor_, Emat_, th_, options_] :=
Module[{i, j, k, p=2, numer=False, h=th, qcoor, c, w, Nf,
  dNx, dNy, Jdet, B, Ke=Table[0, {10}, {10}]},
  If [Length[options]==2, {numer, p}=options, {numer}=options];
  If [p<1|p>4, Print["p out of range"]; Return[Null]];
  For [k=1, k<=p*p, k++,
    {qcoor, w}= QuadGaussRuleInfo[{p, numer}, k];
    {Nf, dNx, dNy, Jdet}=Quad5IsoPShapeFunDer[ncoor, qcoor];
    If [Length[th]>0, h=th.Nf]; c=w*Jdet*h;
    B={ Flatten[Table[{dNx[[i]], 0}, {i, 5}]],
        Flatten[Table[{0, dNy[[i]]}, {i, 5}]],
        Flatten[Table[{dNy[[i]], dNx[[i]]}, {i, 5}]]];
    Ke+=Simplify[c*Transpose[B].(Emat.B)];
  ]; Return[Ke];
];
```

FIGURE E23.1. Stiffness module for the 5-node “bilinear+bubble” iso-P quadrilateral.

```
Quad5IsoPShapeFunDer[ncoor_, qcoor_] := Module[
  {Nf, dNx, dNy, dNξ, dNη, Nb, dNbξ, dNbη, J11, J12, J21, J22, Jdet, ξ, η, x, y},
  {ξ, η}=qcoor; Nb=(1-ξ^2)*(1-η^2); (* Nb: node-5 "bubble" function *)
  dNbξ=2*ξ*(η^2-1); dNbη=2*η*(ξ^2-1);
  Nf= { ((1-ξ)*(1-η)-Nb)/4, ((1+ξ)*(1-η)-Nb)/4,
        ((1+ξ)*(1+η)-Nb)/4, ((1-ξ)*(1+η)-Nb)/4, Nb};
  dNξ={-(1-η-dNbξ)/4, (1-η-dNbξ)/4,
        (1+η-dNbξ)/4, -(1+η-dNbξ)/4, dNbξ};
  dNη={-(1-ξ+dNbη)/4, -(1+ξ+dNbη)/4,
        (1+ξ-dNbη)/4, (1-ξ-dNbη)/4, dNbη};
  x=Table[ncoor[[i, 1]], {i, 5}]; y=Table[ncoor[[i, 2]], {i, 5}];
  J11=dNξ.x; J12=dNξ.y; J21=dNη.x; J22=dNη.y;
  Jdet=Simplify[J11*J22-J12*J21];
  dNx= ( J22*dNξ-J12*dNη)/Jdet; dNx=Simplify[dNx];
  dNy= (-J21*dNξ+J11*dNη)/Jdet; dNy=Simplify[dNy];
  Return[{Nf, dNx, dNy, Jdet}]
];
```

FIGURE E23.2. The shape function module for the 5-node “bilinear+bubble” iso-P quadrilateral.

Test `Quad5IsoPMembraneStiffness` for the 2:1 rectangular element studied in §23.3.1, with node 5 placed at the element center. Use Gauss rules 1×1 , 2×2 and 3×3 . Take $E = 96 \times 30 = 2880$ in lieu of $E = 96$ to


```

Quad5IsoPMembraneCondStiffness[Ke5_] :=
Module[{i,j,k,n,c,Ke=Ke5,Kc=Table[0,{8},{8}]},
  For [n=10,n>=9,n--,
    For [i=1,i<=n-1,i++, c=Ke[[i,n]]/Ke[[n,n]],
      For [j=1,j<=i,j++, Ke[[j,i]]=Ke[[i,j]]=Ke[[i,j]]-c*Ke[[n,j]]];
    ]];
  For [i=1,i<=8,i++, For [j=1,j<=8,j++, Kc[[i,j]]=Ke[[i,j]]];
  Return[Kc]
];

```

FIGURE E23.3. A mystery module for Exercise 23.2.

get exact integer entries in \mathbf{K}^e for all Gauss rules while keeping $\nu = 1/3$ and $h = 1$. Report on which rules give rank sufficiency. Partial result: $K_{22} = 3380$ and 3588 for the 2×2 and 3×3 rules, respectively.

EXERCISE 23.2 [D:10] Module `Quad5IsoPMembraneCondStiffness` in Figure E23.3 is designed to receive, as only argument, the 10×10 stiffness \mathbf{K}_e computed by `Quad5IsoPMembraneStiffness`, and returns a smaller (8×8) stiffness matrix. State what the function of the module is but do not describe programming details.

EXERCISE 23.3 [C:20] Repeat Exercise 17.3 for the problem illustrated in Figure E17.4, but with the 5-node “bilinear+bubble” iso-P quadrilateral as the 2D element that models the plane beam. Skip item (a). Use the modules of Figures E23.1–3 to do the following. Form the 10×10 stiffness matrix \mathbf{K}_e using `Quad5IsoPMembraneStiffness` with $p = 2$ and `num=False`. Insert this \mathbf{K}_e into `Quad5IsoPMembraneCondStiffness`, which returns a 8×8 stiffness \mathbf{K}_e . Stick this \mathbf{K}_e into equations (E17.6) and (E17.7) to get U_{quad} . Show that the energy ratio is

$$r = \frac{U_{quad}}{U_{beam}} = \frac{\gamma^2(1+\nu)(2+\gamma^2(1-\nu))}{(1+\gamma^2)^2}. \quad (\text{E23.1})$$

Compare this to the energy ratio (E17.8) for $\gamma = 1/10$ and $\nu = 0$ to conclude that shear locking has not been eliminated, or even mitigated, by the injection of the bubble shape functions associated with the interior node.⁵

EXERCISE 23.4 [C:25] Implement the 9-node biquadratic quadrilateral element for plane stress to get its 18×18 stiffness matrix. Follow the style of Figures 23.3–4 or E23.1–2. (The Gauss quadrature module may be reused without change.) Test it for the 2:1 rectangular element studied in §23.3.1, with nodes 5–8 placed at the side midpoints, and node 9 at the element center. For the elastic modulus take $E = 96 \times 39 \times 11 \times 55 \times 7 = 15855840$ instead of $E = 96$, along with $\nu = 1/3$ and $h = 1$, so as to get exact integer entries in \mathbf{K}^e . Use both 2×2 and 3×3 Gauss integration rules and show that the 2×2 rule produces a rank deficiency of 3 in the stiffness. (If the computation with `num=False` takes too long on a slow PC, set `num=True` and `Rationalize` entries as in Figure 23.8.) Partial result: $K_{11} = 5395390$ and 6474468 for the 2×2 and 3×3 rules, respectively.

EXERCISE 23.5 [C:25] An element is said to be *distortion insensitive* when the discrete solution does not appreciably change when the mesh is geometrically distorted while keeping the same number of elements, nodes and degrees of freedom. It is *distortion sensitive* otherwise. A distortion sensitivity test often found in the FEM literature for plane stress quadrilateral elements is pictured in Figure E23.4.

⁵ Even the addition of an infinite number of bubble functions to the 4-node iso-P quadrilateral will not cure shear locking. This “bubble futility” has been known since the late 1960s. But memories are short. Bubbles have been recently revived by some FEM authors for other application contexts, such as multiscale modeling.

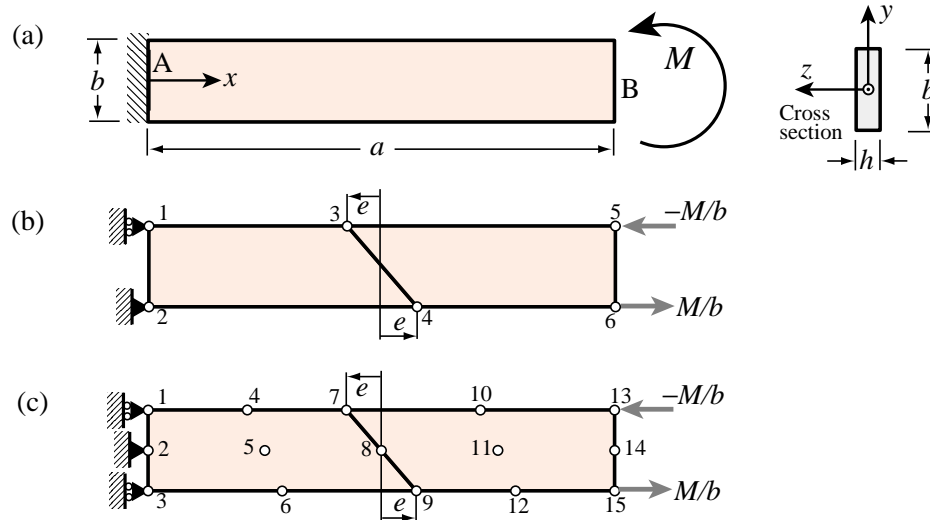


FIGURE E23.4. Two-element cantilever model for assessing distortion sensitivity, for Exercise E23.5.

The cantilever beam of span a , height b and narrow rectangular cross section of thickness h depicted in Figure E23.4 is under an applied end moment M . If the material is isotropic with elastic modulus E and Poisson ratio $\nu = 0$ the end vertical deflection given by the Bernoulli-Euler beam theory is $v_{beam} = Ma^2/(2EI_{zz})$, where $I_{zz} = hb^3/12$. This is also the exact solution given by elasticity theory if the surface tractions over the free end section match the beam stress distribution.⁶

The problem is discretized with two 4-node isoP bilinear quadrilaterals as illustrated in Figure E23.4(b), which show appropriate displacement and force boundary conditions. The mesh distortion is parametrized by the distance e , which defines the slope of interface 3-4. If $e = 0$ there is no distortion.

Obtain the finite element solution for $a = 10$, $b = 2$, $h = E = M = 1$ and $e = 0, 1, 2, 3, 5$ and record the end deflection v_{quad} as the average of the vertical displacements u_{y5} and u_{y6} . Define the ratio $r(e) = v_{quad}/v_{beam}$ and plot $g(e) = r(e)/r(0)$ as function of e . This function $g(e)$ characterizes the distortion sensitivity. Show that $g(e) < 1$ as e increases and thus the mesh stiffness further as a result of the distortion. Conclude that the 4-node bilinear element is distortion sensitive.

EXERCISE 23.6 [C:25] Repeat the steps of Exercise 23.5 for the mesh depicted in Figure E23.4(c). The cantilever beam is modeled with two 9-node biquadratic quadrilaterals integrated by the 3×3 Gauss product rule. It is important to keep the side nodes at the midpoints of the sides, and the center node at the crossing of the medians (that is, the 9-node elements are superparametric). Show that $r = 1$ for any $e < \frac{1}{2}a$. Thus not only this element is exact for the regular mesh, but it is also distortion insensitive.

⁶ This statement would not be true if $\nu \neq 0$ since the fixed-displacement BC at the cantilever root would preclude the lateral expansion or contraction of the cross section. However, the support condition shown in the models (b) and (c) allow such changes so the results are extendible to nonzero ν .

EXERCISE 23.7 [C:25] Implement the 8-node Serendipity quadrilateral element for plane stress to get its 16×16 stiffness matrix. Call the module `Quad8IsoPMembraneStiffness`; the subordinate shape-function and Gauss-quadrature-info modules should be called `Quad8IsoPShapeFunDer` and `QuadGaussRuleInfo`, respectively. Follow the style of Figures 23.3–4 for argument sequence and coding schematics. (The Gauss quadrature module may be reused without change.) Test it for the 2:1 rectangular element studied in §23.3.1, with nodes 5, 6, 7, and 8 placed at midpoints of sides 1–2, 2–3, 3–4, and 4–1, respectively. For elastic modulus take $E = 96 \times 39 \times 11 \times 55 \times 7 = 15855840$ along with $\nu = 1/3$ and $h = 1$. The elastic modulus matrix \mathbf{E} that is passed as second argument should be

$$\mathbf{E} = \begin{bmatrix} 17837820 & 5945940 & 0 \\ 5945940 & 17837820 & 0 \\ 0 & 0 & 5945940 \end{bmatrix}. \quad (\text{E23.2})$$

Use both 2×2 and 3×3 Gauss integration rules and show that the 2×2 rule produces a rank deficiency of 1 in the stiffness. (If the computation with `num=False` takes too long on a slow PC, set `num=True` and `Rationalize` entries as in Figure 23.8.) Partial results: $K_{11} = 11561550$ and 12024012 for the 2×2 and 3×3 rules, respectively, whereas $K_{13} = 4954950$ and 5021016 for the 2×2 and 3×3 rules, respectively.

24

Implementation of Iso-P Triangular Elements

TABLE OF CONTENTS

	Page
§24.1. Introduction	24-3
§24.2. Gauss Quadrature for Triangles	24-3
§24.2.1. Requirements for Gauss Rules	24-3
§24.2.2. Superparametric Triangles	24-4
§24.2.3. Arbitrary Iso-P Triangles	24-5
§24.2.4. Implementation	24-6
§24.3. Partial Derivative Computation	24-6
§24.3.1. Triangle Coordinate Partial	24-7
§24.3.2. Solving the Jacobian System	24-8
§24.4. The Quadratic Triangle	24-9
§24.4.1. Shape Function Module	24-9
§24.4.2. Stiffness Module	24-11
§24.4.3. Test on Straight-Sided Triangle	24-12
§24.4.4. Test on Highly Distorted Triangle	24-12
§24.5. *The Cubic Triangle	24-14
§24.5.1. *Shape Function Module	24-14
§24.5.2. *Stiffness Module	24-16
§24.5.3. *Test Element	24-17
§24. Notes and Bibliography	24-18
§24. References	24-19
§24. Exercises	24-20
§24. Solutions to Exercises	24-22

§24.1. Introduction

This Chapter continues with the computer implementation of two-dimensional finite elements. It covers the programming of isoparametric *triangular* elements for the plane stress problem. Triangular elements bring two *sui generis* implementation quirks with respect to quadrilateral elements:

- (1) The numerical integration rules for triangles are not product of one-dimensional Gauss rules, as in the case of quadrilaterals. They are instead specialized to the triangle geometry.
- (2) The computation of x - y partial derivatives and the element-of-area scaling by the Jacobian determinant must account for the fact that the triangular coordinates ζ_1 , ζ_2 and ζ_3 do not form an independent set.

We deal with these issues in the next two sections.

§24.2. Gauss Quadrature for Triangles

The numerical integration schemes for *quadrilaterals* introduced in §17.3 and implemented in §23.2 are built as “tensor products” of two one-dimensional Gauss formulas. On the other hand, Gauss rules for *triangles* are *not* derivable from one-dimensional rules, and must be constructed especially for the triangular geometry.

§24.2.1. Requirements for Gauss Rules

Gauss quadrature rules for triangles must possess *triangular symmetry* in the following sense:

If the sample point $(\zeta_1, \zeta_2, \zeta_3)$ is present in a Gauss integration rule with weight w , then all other points obtainable by permuting the three triangular coordinates arbitrarily must appear in that rule, and have the same weight.

(24.1)

This constraint guarantees that the result of the quadrature process will not depend on element node numbering.¹ If ζ_1 , ζ_2 , and ζ_3 are different, (24.1) forces six equal-weight sample points to be present in the rule, because $3! = 6$. If two triangular coordinates are equal, the six points coalesce to three, and (24.1) forces three equal-weight sample points to be present. Finally, if the three coordinates are equal (which can only happen for the centroid $\zeta_1 = \zeta_2 = \zeta_3 = 1/3$), the six points coalesce to one.²

Additional requirements for a Gauss rule to be numerically acceptable are:

All sample points must be inside the triangle (or on the triangle boundary) and all weights must be positive.

(24.2)

This is called a *positivity* condition. It insures that the element internal energy evaluated by numerical quadrature is nonnegative definite.

¹ It would be disconcerting to users, to say the least, to have the FEM solution depend on how nodes are numbered.

² It follows that the number of sample points in triangle Gauss quadrature rules that satisfy (24.1) must be of the form $6i + 3j + k$, where i and j are nonnegative integers and k is 0 or 1. Consequently there are no rules with 2, 5 or 8 points.

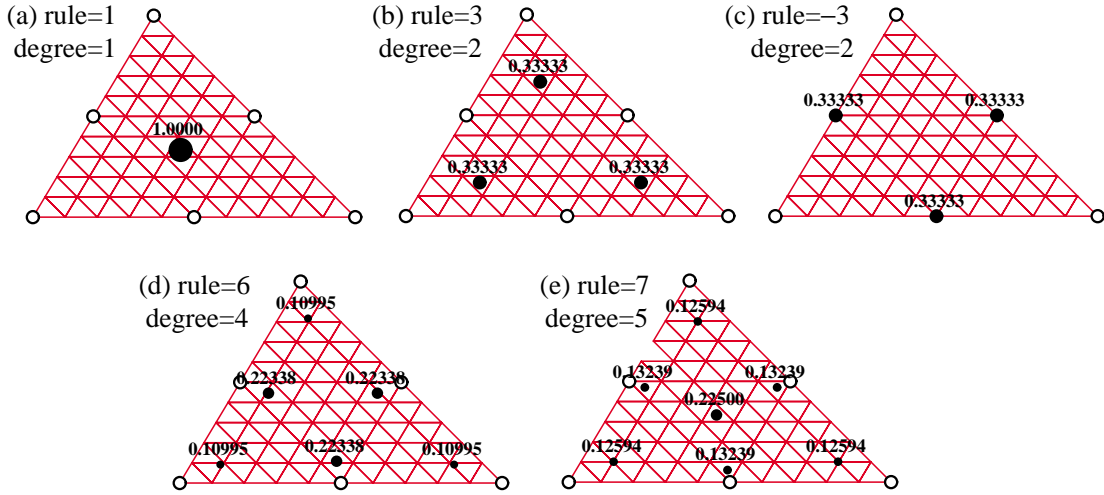


FIGURE 24.1. Location of sample points (dark circles) of five Gauss quadrature rules for straight sided (superparametric) 6-node triangles. Weight written to 5 places near each sample point; sample-point circle areas are proportional to weight.

A rule is said to be of degree n if it integrates exactly all polynomials in the triangular coordinates of order n or less when the Jacobian determinant is constant, and there is at least one polynomial of order $n + 1$ that is not exactly integrated by the rule.

Remark 24.1. The positivity requirement (24.2) is automatically satisfied in quadrilaterals by using Gauss product rules, since the points are always inside while weights are positive. Consequently it was not necessary to call attention to it. On the other hand, for triangles there are Gauss rules with as few as 4 points that violate positivity.

§24.2.2. Superparametric Triangles

We first consider superparametric straight-sided triangles geometry defined by the three corner nodes. Over such triangles the Jacobian determinant defined below is constant. The five simplest Gauss rules that satisfy the requirements (24.1) and (24.2) have 1, 3, 3, 6 and 7 points, respectively. The two rules with 3 points differ in the location of the sample points. The five rules are depicted in Figure 24.1 over 6-node straight-sided triangles; for such triangles to be superparametric the side nodes must be located at the midpoint of the sides.

One point rule. The simplest Gauss rule for a triangle has *one* sample point located at the centroid. For a straight sided triangle,

$$\frac{1}{A} \int_{\Omega^e} F(\zeta_1, \zeta_2, \zeta_3) d\Omega \approx F\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right), \quad (24.3)$$

where A is the triangle area

$$A = \int_{\Omega^e} d\Omega = \frac{1}{2} \det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = \frac{1}{2} [(x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) + (x_1 y_2 - x_2 y_1)]. \quad (24.4)$$

This rule is illustrated in Figure 24.1(a). It has degree 1, meaning that it integrates exactly up to linear polynomials in $\{\zeta_1, \zeta_2, \zeta_3\}$. For example, $F = 4 - \zeta_1 + 2\zeta_2 - \zeta_3$ is exactly integrated by (24.3).

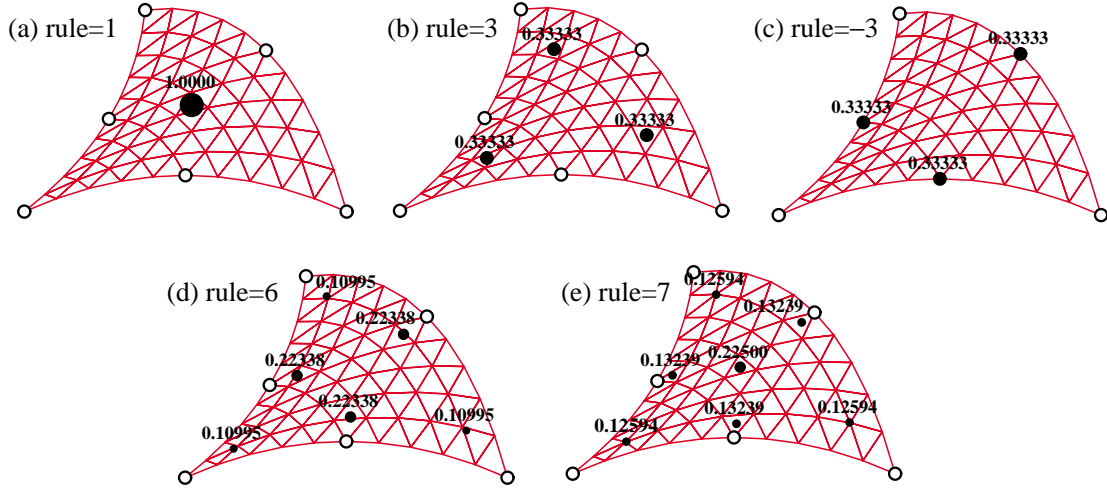


FIGURE 24.2. Location of sample points (dark circles) of five Gauss quadrature rules for curved sided 6-node triangles. Weight written to 5 places near each sample point; sample-point circle areas are proportional to weight.

Three Point Rules. The next two rules in order of simplicity contain *three* sample points:

$$\frac{1}{A} \int_{\Omega^e} F(\zeta_1, \zeta_2, \zeta_3) d\Omega \approx \frac{1}{3} F\left(\frac{2}{3}, \frac{1}{6}, \frac{1}{6}\right) + \frac{1}{3} F\left(\frac{1}{6}, \frac{2}{3}, \frac{1}{6}\right) + \frac{1}{3} F\left(\frac{1}{6}, \frac{1}{6}, \frac{2}{3}\right). \quad (24.5)$$

$$\frac{1}{A} \int_{\Omega^e} F(\zeta_1, \zeta_2, \zeta_3) d\Omega \approx \frac{1}{3} F\left(\frac{1}{2}, \frac{1}{2}, 0\right) + \frac{1}{3} F\left(0, \frac{1}{2}, \frac{1}{2}\right) + \frac{1}{3} F\left(\frac{1}{2}, 0, \frac{1}{2}\right). \quad (24.6)$$

These are depicted in Figures 24.1(b,c). Both rules are of degree 2; that is, they integrate exactly up to quadratic polynomials in the triangular coordinates. For example, the function $F = 6 + \zeta_1 + 3\zeta_3 + \zeta_2^2 - \zeta_3^2 + 3\zeta_1\zeta_3$ is integrated exactly by either rule. Formula (24.6) is called the *midpoint rule*.

Six and Seven Point Rules. There is a 4-point rule of degree 3, but it has a negative weight and so violates (24.2). There are no symmetric rules with 5 points. The next useful rules have six and seven points. There is a 6-point rule of degree 4 and a 7-point rule of degree 5, which integrate exactly up to quartic and quintic polynomials in $\{\zeta_1, \zeta_2, \zeta_3\}$, respectively. The 7-point rule includes the centroid as sample point. The abscissas and weights are expressible as rational combinations of square roots of integers and fractions. The expressions are listed in the *Mathematica* implementation discussed in §24.2.4. The sample point configurations are depicted in Figure 24.1(d,e).

§24.2.3. Arbitrary Iso-P Triangles

If the triangle has variable metric, as in the curved sided 6-node triangle geometries shown in Figure 24.2, the foregoing formulas need adjustment because the element of area $d\Omega$ becomes a function of position. Consider the more general case of an isoparametric element with n nodes and shape functions N_i . In §24.3 it is shown that the differential area element is given by

$$d\Omega = J d\zeta_1 d\zeta_2 d\zeta_3, \quad J = \frac{1}{2} \det \begin{bmatrix} \sum_{i=1}^n x_i \frac{\partial N_i}{\partial \zeta_1} & \sum_{i=1}^n x_i \frac{\partial N_i}{\partial \zeta_2} & \sum_{i=1}^n x_i \frac{\partial N_i}{\partial \zeta_3} \\ \sum_{i=1}^n y_i \frac{\partial N_i}{\partial \zeta_1} & \sum_{i=1}^n y_i \frac{\partial N_i}{\partial \zeta_2} & \sum_{i=1}^n y_i \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \quad (24.7)$$


```

TrigGaussRuleInfo[{rule_,number_},point_]:= Module[
{zeta,p=rule,i=point,g1,g2,info={{Null,Null,Null},0}},
If [p== 1, info={{1/3,1/3,1/3},1}];
If [p== 3, info={{1,1,1}/6,1/3}; info[[1,i]]=2/3];
If [p== -3, info={{1,1,1}/2,1/3}; info[[1,i]]=0];
If [p== 6, g1=(8-Sqrt[10]+Sqrt[38-44*Sqrt[2/5]])/18;
g2=(8-Sqrt[10]-Sqrt[38-44*Sqrt[2/5]])/18;
If [i<4, info={{g1,g1,g1},(620+Sqrt[213125-
53320*Sqrt[10]])/3720}; info[[1,i]]=1-2*g1];
If [i>3, info={{g2,g2,g2},(620-Sqrt[213125-
53320*Sqrt[10]])/3720}; info[[1,i-3]]=1-2*g2];
If [p== 7, g1=(6-Sqrt[15])/21; g2=(6+Sqrt[15])/21;
If [i<4, info={{g1,g1,g1},(155-Sqrt[15])/1200};
info[[1,i]]= 1-2*g1];
If [i>3&& i<7, info={{g2,g2,g2},(155+Sqrt[15])/1200};
info[[1,i-3]]=1-2*g2];
If [i==7, info={{1/3,1/3,1/3},9/40}];
If [number, Return[N[info]], Return[Simplify[info]]];
];

```

FIGURE 24.3. Module to get triangle Gauss quadrature rule information.

Here J is the Jacobian determinant, which plays the same role as J in the isoparametric quadrilaterals. If the metric is simply defined by the 3 corners, as in Figure 24.1, the geometry shape functions are $N_1 = \zeta_1$, $N_2 = \zeta_2$ and $N_3 = \zeta_3$. Then the foregoing determinant reduces to that of (24.4), and $J = A$ everywhere. But for general (curved) geometries $J = J(\zeta_1, \zeta_2, \zeta_3)$, and the triangle area A cannot be factored out of the integration rules. For example the one point rule becomes

$$\int_{\Omega^e} F(\zeta_1, \zeta_2, \zeta_3) d\Omega \approx J\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right) F\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right). \quad (24.8)$$

whereas the midpoint rule becomes

$$\int_{\Omega^e} F(\zeta_1, \zeta_2, \zeta_3) d\Omega \approx \frac{1}{3} J\left(\frac{1}{2}, \frac{1}{2}, 0\right) F\left(\frac{1}{2}, \frac{1}{2}, 0\right) + \frac{1}{3} J\left(0, \frac{1}{2}, \frac{1}{2}\right) F\left(0, \frac{1}{2}, \frac{1}{2}\right) + \frac{1}{3} J\left(\frac{1}{2}, 0, \frac{1}{2}\right) F\left(\frac{1}{2}, 0, \frac{1}{2}\right). \quad (24.9)$$

These can be expressed more compactly by saying that the Gauss integration rule is applied to JF .

§24.2.4. Implementation

The five rules pictured in Figures 24.1 and 24.2 are implemented in the module TrigGaussRuleInfo listed in Figure 24.3. The module is invoked as

$$\{\{zeta_1, zeta_2, zeta_3\}, w\} = \text{TrigGaussRuleInfo}[\{\text{rule}, \text{number}\}, \text{point}] \quad (24.10)$$

The module has three arguments: rule, number and i. The first two are grouped in a two-item list. Argument rule, which can be 1, 3, -3, 6 or 7, defines the integration formula as follows. Abs[rule] is the number of sample points. Of the two 3-point choices, if rule is -3 the midpoint rule is picked, else if +3 the 3-interior point rule is chosen. Logical flag number is set to True or False to request floating-point or exact information, respectively

Argument point is the index of the sample point, which may range from 1 through Abs[rule].

The module returns the list $\{\{\zeta_1, \zeta_2, \zeta_3\}, w\}$, where $\zeta_1, \zeta_2, \zeta_3$ are the triangular coordinates of the sample point, and w is the integration weight. For example, TrigGaussRuleInfo[{3,False},1] returns $\{\{2/3, 1/6, 1/6\}, 1/3\}$. If rule is not implemented, the module returns $\{\{Null, Null, Null\}, 0\}$.

§24.3. Partial Derivative Computation

The calculation of Cartesian partial derivatives is illustrated in this section for the 6-node triangle shown in Figure 24.4. The results are applicable, however, to iso-P triangles with any number of nodes.

The element geometry is defined by the corner coordinates $\{x_i, y_i\}$, with $i = 1, 2, \dots, 6$. Corners are numbered 1, 2, 3 in counterclockwise sense. Side nodes are numbered 4, 5, 6 opposite to corners 3, 1, 2, respectively. Side nodes may be arbitrarily located within positive Jacobian constraints as discussed in §19.4.2. The triangular coordinates are as usual denoted by ζ_1, ζ_2 and ζ_3 , which satisfy $\zeta_1 + \zeta_2 + \zeta_3 = 1$. The quadratic displacement field $\{u_x(\zeta_1, \zeta_2, \zeta_3), u_y(\zeta_1, \zeta_2, \zeta_3)\}$ is defined by the 12 node displacements $\{u_{xi}, u_{yi}\}$, $i = 1, 2, \dots, 6$, as per the iso-P quadratic interpolation formula (16.10–11). That formula is repeated here for convenience:

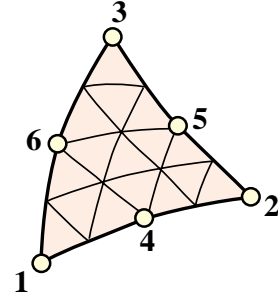


FIGURE 24.4. The 6-node iso-P triangle.

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \\ N_4^e \\ N_5^e \\ N_6^e \end{bmatrix} \quad (24.11)$$

in which the shape functions and their natural derivatives are

$$\mathbf{N}^T = \begin{bmatrix} N_1^e \\ N_2^e \\ N_3^e \\ N_4^e \\ N_5^e \\ N_6^e \end{bmatrix} = \begin{bmatrix} \zeta_1(2\zeta_1-1) \\ \zeta_2(2\zeta_2-1) \\ \zeta_3(2\zeta_3-1) \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial \zeta_1} = \begin{bmatrix} 4\zeta_1-1 \\ 0 \\ 0 \\ 4\zeta_2 \\ 0 \\ 4\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial \zeta_2} = \begin{bmatrix} 0 \\ 4\zeta_2-1 \\ 0 \\ 4\zeta_1 \\ 4\zeta_3 \\ 0 \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial \zeta_3} = \begin{bmatrix} 0 \\ 0 \\ 4\zeta_3-1 \\ 0 \\ 4\zeta_2 \\ 4\zeta_1 \end{bmatrix}. \quad (24.12)$$

§24.3.1. Triangle Coordinate Partials

In this and following sections the superscript e of shape functions will be omitted for brevity. The bulk of the shape function logic is concerned with the computation of the partial derivatives of the shape functions (24.12) with respect to x and y at any point in the element. For this purpose consider a *generic* scalar function $w(\zeta_1, \zeta_2, \zeta_3)$ that is quadratically interpolated over the triangle by

$$w = w_1 N_1 + w_2 N_2 + w_3 N_3 + w_4 N_4 + w_5 N_5 + w_6 N_6 = [w_1 \ w_2 \ w_3 \ w_4 \ w_5 \ w_6] \mathbf{N}^T. \quad (24.13)$$

Symbol w may stand for 1, x , y , u_x or u_y , which are interpolated in the iso-P representation (24.11), or other element-varying quantities such as thickness, temperature, etc. Taking partials of (24.13) with respect to x and y and applying the chain rule twice yields

$$\begin{aligned} \frac{\partial w}{\partial x} &= \sum w_i \frac{\partial N_i}{\partial x} = \sum w_i \left(\frac{\partial N_i}{\partial \zeta_1} \frac{\partial \zeta_1}{\partial x} + \frac{\partial N_i}{\partial \zeta_2} \frac{\partial \zeta_2}{\partial x} + \frac{\partial N_i}{\partial \zeta_3} \frac{\partial \zeta_3}{\partial x} \right), \\ \frac{\partial w}{\partial y} &= \sum w_i \frac{\partial N_i}{\partial y} = \sum w_i \left(\frac{\partial N_i}{\partial \zeta_1} \frac{\partial \zeta_1}{\partial y} + \frac{\partial N_i}{\partial \zeta_2} \frac{\partial \zeta_2}{\partial y} + \frac{\partial N_i}{\partial \zeta_3} \frac{\partial \zeta_3}{\partial y} \right), \end{aligned} \quad (24.14)$$

where all sums are understood to run over $i = 1, \dots, 6$. In matrix form:

$$\begin{bmatrix} \frac{\partial w}{\partial x} \\ \frac{\partial w}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_3}{\partial x} \\ \frac{\partial \zeta_1}{\partial y} & \frac{\partial \zeta_2}{\partial y} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} \begin{bmatrix} \sum w_i \frac{\partial N_i}{\partial \zeta_1} \\ \sum w_i \frac{\partial N_i}{\partial \zeta_2} \\ \sum w_i \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix}. \quad (24.15)$$

Transposing both sides of (24.15) while exchanging sides yields

$$\begin{bmatrix} \sum w_i \frac{\partial N_i}{\partial \zeta_1} & \sum w_i \frac{\partial N_i}{\partial \zeta_2} & \sum w_i \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} \end{bmatrix}. \quad (24.16)$$

Now make $w \equiv 1, x, y$ and stack the results row-wise:

$$\begin{bmatrix} \sum \frac{\partial N_i}{\partial \zeta_1} & \sum \frac{\partial N_i}{\partial \zeta_2} & \sum \frac{\partial N_i}{\partial \zeta_3} \\ \sum x_i \frac{\partial N_i}{\partial \zeta_1} & \sum x_i \frac{\partial N_i}{\partial \zeta_2} & \sum x_i \frac{\partial N_i}{\partial \zeta_3} \\ \sum y_i \frac{\partial N_i}{\partial \zeta_1} & \sum y_i \frac{\partial N_i}{\partial \zeta_2} & \sum y_i \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial 1}{\partial x} & \frac{\partial 1}{\partial y} \\ \frac{\partial x}{\partial x} & \frac{\partial x}{\partial y} \\ \frac{\partial y}{\partial x} & \frac{\partial y}{\partial y} \end{bmatrix}. \quad (24.17)$$

But $\partial x/\partial x = \partial y/\partial y = 1$ and $\partial 1/\partial x = \partial 1/\partial y = \partial x/\partial y = \partial y/\partial x = 0$ because x and y are independent coordinates. It is shown in Remark 24.2 below that, if $\sum N_i = 1$, the entries of the first row of the coefficient matrix are equal to a constant C . These entries can be scaled to unity because the first row of the right-hand side is null. Consequently we arrive at a system of linear equations of order 3 with two right-hand sides:

$$\begin{bmatrix} 1 & 1 & 1 \\ \sum x_i \frac{\partial N_i}{\partial \zeta_1} & \sum x_i \frac{\partial N_i}{\partial \zeta_2} & \sum x_i \frac{\partial N_i}{\partial \zeta_3} \\ \sum y_i \frac{\partial N_i}{\partial \zeta_1} & \sum y_i \frac{\partial N_i}{\partial \zeta_2} & \sum y_i \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (24.18)$$

§24.3.2. Solving the Jacobian System

By analogy with quadrilateral elements, the coefficient matrix of (24.18) will be called the *Jacobian matrix* and denoted by \mathbf{J} . Its determinant scaled by one half is equal to the Jacobian $J = \frac{1}{2} \det \mathbf{J}$ used in the expression of the area element introduced in §24.2.3. For compactness (24.18) is rewritten

$$\mathbf{J}\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 \\ J_{x1} & J_{x2} & J_{x3} \\ J_{y1} & J_{y2} & J_{y3} \end{bmatrix} \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \quad (24.19)$$

If $J \neq 0$, solving this system gives

$$\begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \frac{1}{2J} \begin{bmatrix} J_{y23} & J_{x32} \\ J_{y31} & J_{x13} \\ J_{y12} & J_{x21} \end{bmatrix} = \mathbf{P}, \quad (24.20)$$

in which $J_{xji} = J_{xj} - J_{xi}$, $J_{yji} = J_{yj} - J_{yi}$ and $J = \frac{1}{2} \det \mathbf{J} = \frac{1}{2}(J_{x21}J_{y31} - J_{y12}J_{x13})$. Substituting into (24.14) we arrive at

$$\begin{aligned}\frac{\partial w}{\partial x} &= \sum w_i \frac{\partial N_i}{\partial x} = \sum \frac{w_i}{2J} \left(\frac{\partial N_i}{\partial \zeta_1} J_{y23} + \frac{\partial N_i}{\partial \zeta_2} J_{y31} + \frac{\partial N_i}{\partial \zeta_3} J_{y12} \right), \\ \frac{\partial w}{\partial y} &= \sum w_i \frac{\partial N_i}{\partial y} = \sum \frac{w_i}{2J} \left(\frac{\partial N_i}{\partial \zeta_1} J_{x32} + \frac{\partial N_i}{\partial \zeta_2} J_{x13} + \frac{\partial N_i}{\partial \zeta_3} J_{x21} \right).\end{aligned}\quad (24.21)$$

In particular, the shape function derivatives are

$$\begin{aligned}\frac{\partial N_i}{\partial x} &= \frac{1}{2J} \left(\frac{\partial N_i}{\partial \zeta_1} J_{y23} + \frac{\partial N_i}{\partial \zeta_2} J_{y31} + \frac{\partial N_i}{\partial \zeta_3} J_{y12} \right), \\ \frac{\partial N_i}{\partial y} &= \frac{1}{2J} \left(\frac{\partial N_i}{\partial \zeta_1} J_{x32} + \frac{\partial N_i}{\partial \zeta_2} J_{x13} + \frac{\partial N_i}{\partial \zeta_3} J_{x21} \right).\end{aligned}\quad (24.22)$$

in which the natural derivatives $\partial N_i / \partial \zeta_j$ can be read off (24.12). Using the 3×2 \mathbf{P} matrix defined in (24.20) yields finally the compact form

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} & \frac{\partial N_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_i}{\partial \zeta_1} & \frac{\partial N_i}{\partial \zeta_2} & \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \mathbf{P}.\quad (24.23)$$

Remark 24.2. Here is the proof that each first row entry of the 3×3 matrix in (24.17) is a numerical constant, say C . Suppose the shape functions are polynomials of order n in the triangular coordinates, and let $Z = \zeta_1 + \zeta_2 + \zeta_3$. The completeness identity is

$$S = \sum N_i = 1 = c_1 Z + c_2 Z^2 + \dots c_n Z^n, \quad c_1 + c_2 + \dots + c_n = 1. \quad (24.24)$$

where the c_i are element dependent scalar coefficients. Differentiating S with respect to the ζ_i 's and setting $Z = 1$ yields

$$\begin{aligned}C = \sum \frac{\partial N_i}{\partial \zeta_1} &= \sum \frac{\partial N_i}{\partial \zeta_2} = \sum \frac{\partial N_i}{\partial \zeta_3} = c_1 + 2c_2 Z + c_3 Z^3 + \dots (n-1)Z^{n-1} = c_1 + 2c_2 + 3c_3 + \dots + c_{n-1} \\ &= 1 + c_2 + 2c_3 + \dots + (n-2)c_n,\end{aligned}\quad (24.25)$$

which proves the assertion. For the 3-node linear triangle, $S = Z$ and $C = 1$. For the 6-node quadratic triangle, $S = 2Z^2 - Z$ and $C = 3$. For the 10-node cubic triangle, $S = 9Z^3/2 - 9Z^2/2 + Z$ and $C = 11/2$. Because the first equation in (24.18) is homogeneous, the C 's can be scaled to unity.

§24.4. The Quadratic Triangle

§24.4.1. Shape Function Module

We specialize now the results of §24.3 to obtain the stiffness matrix of the 6-node quadratic triangle in plane stress. Taking the dot products of the natural-coordinate partials (24.12) with the node coordinates we obtain the entries of the Jacobian matrix of (24.19):

$$\begin{aligned}J_{x1} &= x_1(4\zeta_1 - 1) + 4(x_4\zeta_2 + x_6\zeta_3), & J_{x2} &= x_2(4\zeta_2 - 1) + 4(x_5\zeta_3 + x_4\zeta_1), & J_{x3} &= x_3(4\zeta_3 - 1) + 4(x_6\zeta_1 + x_5\zeta_2), \\ J_{y1} &= y_1(4\zeta_1 - 1) + 4(y_4\zeta_2 + y_6\zeta_3), & J_{y2} &= y_2(4\zeta_2 - 1) + 4(y_5\zeta_3 + y_4\zeta_1), & J_{y3} &= y_3(4\zeta_3 - 1) + 4(y_6\zeta_1 + y_5\zeta_2).\end{aligned}\quad (24.26)$$

```

Trig6IsoPShapeFunDer[ncoor_,tcoor_] := Module[
{ζ1,ζ2,ζ3,x1,x2,x3,x4,x5,x6,y1,y2,y3,y4,y5,y6,
dx4,dx5,dx6,dy4,dy5,dy6,Jx21,Jx32,Jx13,Jy12,Jy23,Jy31,
Nf,dNx,dNy,Jdet}, {ζ1,ζ2,ζ3}=tcoor;
{{x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5},{x6,y6}}=ncoor;
dx4=x4-(x1+x2)/2; dx5=x5-(x2+x3)/2; dx6=x6-(x3+x1)/2;
dy4=y4-(y1+y2)/2; dy5=y5-(y2+y3)/2; dy6=y6-(y3+y1)/2;
Nf={ζ1*(2*ζ1-1),ζ2*(2*ζ2-1),ζ3*(2*ζ3-1),4*ζ1*ζ2,4*ζ2*ζ3,4*ζ3*ζ1};
Jx21= x2-x1+4*(dx4*(ζ1-ζ2)+(dx5-dx6)*ζ3);
Jx32= x3-x2+4*(dx5*(ζ2-ζ3)+(dx6-dx4)*ζ1);
Jx13= x1-x3+4*(dx6*(ζ3-ζ1)+(dx4-dx5)*ζ2);
Jy12= y1-y2+4*(dy4*(ζ2-ζ1)+(dy6-dy5)*ζ3);
Jy23= y2-y3+4*(dy5*(ζ3-ζ2)+(dy4-dy6)*ζ1);
Jy31= y3-y1+4*(dy6*(ζ1-ζ3)+(dy5-dy4)*ζ2);
Jdet = Jx21*Jy31-Jy12*Jx13;
dNx= {(4*ζ1-1)*Jy23,(4*ζ2-1)*Jy31,(4*ζ3-1)*Jy12,4*(ζ2*Jy23+ζ1*Jy31),
4*(ζ3*Jy31+ζ2*Jy12),4*(ζ1*Jy12+ζ3*Jy23)}/Jdet;
dNy= {(4*ζ1-1)*Jx32,(4*ζ2-1)*Jx13,(4*ζ3-1)*Jx21,4*(ζ2*Jx32+ζ1*Jx13),
4*(ζ3*Jx13+ζ2*Jx21),4*(ζ1*Jx21+ζ3*Jx32)}/Jdet;
Return[Simplify[{Nf,dNx,dNy,Jdet}]]
];

```

FIGURE 24.5. Shape function module for 6-node quadratic triangle.

From these \mathbf{J} can be constructed, and shape function partials $\partial N_i/\partial x$ and $\partial N_i/\partial y$ explicitly obtained from (24.22). Somewhat simpler expressions, however, result by using the following “hierarchical” side node coordinates:

$$\begin{aligned} \Delta x_4 &= x_4 - \frac{1}{2}(x_1 + x_2), & \Delta x_5 &= x_5 - \frac{1}{2}(x_2 + x_3), & \Delta x_6 &= x_6 - \frac{1}{2}(x_3 + x_1), \\ \Delta y_4 &= y_4 - \frac{1}{2}(y_1 + y_2), & \Delta y_5 &= y_5 - \frac{1}{2}(y_2 + y_3), & \Delta y_6 &= y_6 - \frac{1}{2}(y_3 + y_1). \end{aligned} \quad (24.27)$$

Geometrically these represent the *deviations from midpoint positions*; thus for a superparametric element $\Delta x_4 = \Delta x_5 = \Delta x_6 = \Delta y_4 = \Delta y_5 = \Delta y_6 = 0$. The Jacobian coefficients become

$$\begin{aligned} J_{x21} &= x_{21} + 4(\Delta x_4(\zeta_1 - \zeta_2) + (\Delta x_5 - \Delta x_6)\zeta_3), & J_{x32} &= x_{32} + 4(\Delta x_5(\zeta_2 - \zeta_3) + (\Delta x_6 - \Delta x_4)\zeta_1), \\ J_{x13} &= x_{13} + 4(\Delta x_6(\zeta_3 - \zeta_1) + (\Delta x_4 - \Delta x_5)\zeta_2), & J_{y12} &= y_{12} + 4(\Delta y_4(\zeta_2 - \zeta_1) + (\Delta y_6 - \Delta y_5)\zeta_3), \\ J_{y23} &= y_{23} + 4(\Delta y_5(\zeta_3 - \zeta_2) + (\Delta y_4 - \Delta y_6)\zeta_1), & J_{y31} &= y_{31} + 4(\Delta y_6(\zeta_1 - \zeta_3) + (\Delta y_5 - \Delta y_4)\zeta_2). \end{aligned} \quad (24.28)$$

(Note that if all midpoint deviations vanish, $J_{xji} = x_{ji}$ and $J_{yji} = y_{ji}$.) From this one gets $J = \frac{1}{2} \det \mathbf{J} = \frac{1}{2} (J_{x21} J_{y31} - J_{y12} J_{x13})$ and

$$\mathbf{P} = \frac{1}{2J} \begin{bmatrix} y_{23} + 4(\Delta y_5(\zeta_3 - \zeta_2) + (\Delta y_4 - \Delta y_6)\zeta_1) & x_{32} + 4(\Delta x_5(\zeta_2 - \zeta_3) + (\Delta x_6 - \Delta x_4)\zeta_1) \\ y_{31} + 4(\Delta y_6(\zeta_1 - \zeta_3) + (\Delta y_5 - \Delta y_4)\zeta_2) & x_{13} + 4(\Delta x_6(\zeta_3 - \zeta_1) + (\Delta x_4 - \Delta x_5)\zeta_2) \\ y_{12} + 4(\Delta y_4(\zeta_2 - \zeta_1) + (\Delta y_6 - \Delta y_5)\zeta_3) & x_{21} + 4(\Delta x_4(\zeta_1 - \zeta_2) + (\Delta x_5 - \Delta x_6)\zeta_3) \end{bmatrix}. \quad (24.29)$$

and the Cartesian derivatives of the shape functions are

$$\frac{\partial \mathbf{N}^T}{\partial x} = \frac{1}{2J} \begin{bmatrix} (4\zeta_1 - 1)J_{y23} \\ (4\zeta_2 - 1)J_{y31} \\ (4\zeta_3 - 1)J_{y12} \\ 4(\zeta_2 J_{y23} + \zeta_1 J_{y31}) \\ 4(\zeta_3 J_{y31} + \zeta_2 J_{y12}) \\ 4(\zeta_1 J_{y12} + \zeta_3 J_{y23}) \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial y} = \frac{1}{2J} \begin{bmatrix} (4\zeta_1 - 1)J_{x32} \\ (4\zeta_2 - 1)J_{x13} \\ (4\zeta_3 - 1)J_{x21} \\ 4(\zeta_2 J_{x32} + \zeta_1 J_{x13}) \\ 4(\zeta_3 J_{x13} + \zeta_2 J_{x21}) \\ 4(\zeta_1 J_{x21} + \zeta_3 J_{x32}) \end{bmatrix}. \quad (24.30)$$

```

Trig6IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
Module[{i,k,p=3,numer=False,h=th,tcoor,w,c,
Nf,dNx,dNy,Jdet,Be,Ke=Table[0,{12},{12}]],
If [Length[options]>=1, numer=options[[1]]];
If [Length[options]>=2, p= options[[2]]];
If [!MemberQ[{1,-3,3,6,7},p], Print["Illegal p"]; Return[Null]];
For [k=1, k<=Abs[p], k++,
{tcoor,w}= TrigGaussRuleInfo[{p,numer},k];
{Nf,dNx,dNy,Jdet}= Trig6IsoPShapeFunDer[ncoor,tcoor];
If [numer, {Nf,dNx,dNy,Jdet}=N[{Nf,dNx,dNy,Jdet}]];
If [Length[th]==6, h=th.Nf]; c=w*Jdet*h/2;
Be= {Flatten[Table[{dNx[[i]],0},{i,6}]],
Flatten[Table[{0,dNy[[i]]},{i,6}]],
Flatten[Table[{dNy[[i]],dNx[[i]]},{i,6}]]};
Ke+=c*Transpose[Be].(Emat.Be);
]; If[!numer,Ke=Simplify[Ke]]; Return[Ke]
];

```

FIGURE 24.6. Stiffness matrix module for 6-node plane stress triangle.

A *Mathematica* shape function module Trig6IsoPShapeFunDer is shown in Figure 24.5. It receives two arguments: ncoor and tcoor. The first one is the list of $\{x_i, y_i\}$ coordinates of the six nodes. The second is the list of three triangular coordinates $\{\zeta_1, \zeta_2, \zeta_3\}$ of the location at which the shape functions and their Cartesian derivatives are to be computed.

The module returns $\{Nf, dNx, dNy, Jdet\}$ as module value. Here Nf collects the shape function values, dNx and dNy the x and y shape function derivatives, respectively, and Jdet is the determinant of matrix **J**, equal to $2J$ in the notation used here.

§24.4.2. Stiffness Module

The numerically integrated stiffness matrix is

$$\mathbf{K}^e = \int_{\Omega^e} h \mathbf{B}^T \mathbf{E} \mathbf{B} d\Omega \approx \sum_{i=1}^p w_i \mathbf{F}(\zeta_{1i}, \zeta_{2i}, \zeta_{3i}), \quad \text{where} \quad \mathbf{F}(\zeta_1, \zeta_2, \zeta_3) = h \mathbf{B}^T \mathbf{E} \mathbf{B} J. \quad (24.31)$$

Here p denotes the number of sample points of the Gauss rule being used, w_i is the integration weight for the i^{th} sample point, $\zeta_{1i}, \zeta_{2i}, \zeta_{3i}$ are the sample point triangular coordinates and $J = \frac{1}{2} \det \mathbf{J}$. This data is provided by TrigGaussRuleInfo. For the 6-node triangle (24.31) is implemented in module Trig6IsoPMembraneStiffness, listed in Figure 24.6. The module is invoked as

$$\mathbf{K}^e = \text{Trig6IsoPMembraneStiffness}[\text{ncoor}, \text{Emat}, \text{th}, \text{options}] \quad (24.32)$$

The arguments are:

- ncoor The list of node coordinates arranged as $\{\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_6, y_6\}\}$.
Emat The plane stress elasticity matrix

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \quad (24.33)$$

arranged as $\{\{E_{11}, E_{12}, E_{33}\}, \{E_{12}, E_{22}, E_{23}\}, \{E_{13}, E_{23}, E_{33}\}\}$.

- th Plate thickness specified as either a scalar h or a six-entry list: $\{h_1, h_2, h_3, h_4, h_5, h_6\}$.

The one-entry form specifies uniform thickness h . The six-entry form is used to specify an element of variable thickness, in which case the entries are the six node thicknesses and h is interpolated quadratically.

options Processing options list. May contain two items: { **numer**, **rule** } or one: { **numer** }.

numer is a logical flag. If True, the computations are forced to proceed in floating-point arithmetic. For symbolic or exact arithmetic work set **numer** to False.

rule specifies the triangle Gauss rule as described in §24.2.4; **rule** may be 1, 3, -3, 6 or 7. For the 6-node element the three point rules are sufficient to get the correct rank. If omitted **rule** = 3 is assumed.

The module returns \mathbf{K}^e as an 12×12 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2} \ u_{x3} \ u_{y3} \ u_{x4} \ u_{y4} \ u_{x5} \ u_{y5} \ u_{x6} \ u_{y6}]^T. \quad (24.34)$$

§24.4.3. Test on Straight-Sided Triangle

The stiffness module of Figure 24.6 is tested on two triangle geometries, one with straight sides and constant metric and one with curved sides and highly variable metric. The straight sided triangle geometry, shown in Figure 24.7, has the corner nodes placed at (0, 0), (6, 2) and (4, 4) with side nodes 4,5,6 at the midpoints of the sides. The element has unit plate thickness. The material is isotropic with $E = 288$ and $\nu = 1/3$.

The element is tested with the script shown in Figure 24.8. The stiffness matrix \mathbf{K}^e is computed by the three-interior-point Gauss rule, specified as $p=3$.

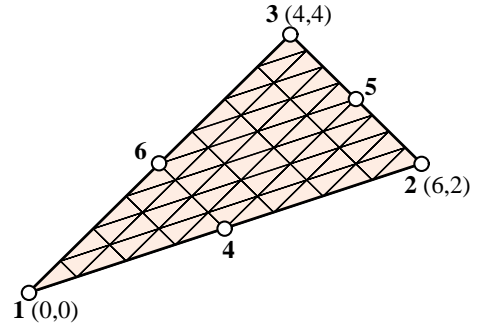


FIGURE 24.7. Straight sided 6-node triangle test element.

The computed \mathbf{K}^e for integration rules with 3 or more points are identical since those rules are exact for this element if the metric is constant, which is the case here. That stiffness is

$$\begin{bmatrix} 54 & 27 & 18 & 0 & 0 & 9 & -72 & 0 & 0 & 0 & 0 & -36 \\ 27 & 54 & 0 & -18 & 9 & 36 & 0 & 72 & 0 & 0 & -36 & -144 \\ 18 & 0 & 216 & -108 & 54 & -36 & -72 & 0 & -216 & 144 & 0 & 0 \\ 0 & -18 & -108 & 216 & -36 & 90 & 0 & 72 & 144 & -360 & 0 & 0 \\ 0 & 9 & 54 & -36 & 162 & -81 & 0 & 0 & -216 & 144 & 0 & -36 \\ 9 & 36 & -36 & 90 & -81 & 378 & 0 & 0 & 144 & -360 & -36 & -144 \\ -72 & 0 & -72 & 0 & 0 & 0 & 576 & -216 & 0 & -72 & -432 & 288 \\ 0 & 72 & 0 & 72 & 0 & 0 & -216 & 864 & -72 & -288 & 288 & -720 \\ 0 & 0 & -216 & 144 & -216 & 144 & 0 & -72 & 576 & -216 & -144 & 0 \\ 0 & 0 & 144 & -360 & 144 & -360 & -72 & -288 & -216 & 864 & 0 & 144 \\ 0 & -36 & 0 & 0 & 0 & -36 & -432 & 288 & -144 & 0 & 576 & -216 \\ -36 & -144 & 0 & 0 & -36 & -144 & 288 & -720 & 0 & 144 & -216 & 864 \end{bmatrix} \quad (24.35)$$

The eigenvalues are:

$$[1971.66 \ 1416.75 \ 694.82 \ 545.72 \ 367.7 \ 175.23 \ 157.68 \ 57.54 \ 12.899 \ 0 \ 0 \ 0] \quad (24.36)$$

The 3 zero eigenvalues pertain to the three independent rigid-body modes. The 9 other ones are positive. Consequently the computed \mathbf{K}^e has the correct rank of 9.

```

ClearAll[Em,v,a,b,e,h]; h=1; Em=288; v=1/3;
ncoor={{0,0},{6,2},{4,4},{3,1},{5,3},{2,2}};
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};
Print["Emat=",Emat//MatrixForm]
Ke=Trig6IsoPMembraneStiffness[ncoor,Emat,h,{False,3}];
Ke=Simplify[Ke]; Print[Chop[Ke]//MatrixForm];
Print["eigs of Ke=",Chop[Eigenvalues[N[Ke]]]];

```

FIGURE 24.8. Script to form \mathbf{K}^e of test triangle of Figure 24.7.

§24.4.4. Test on Highly Distorted Triangle

A highly distorted test geometry places the 3 corners at the vertices of an equilateral triangle: $\{-1/2, 0\}$, $\{1/2, 0\}$, $\{0, \sqrt{3}/2\}$ whereas the 3 side nodes are located at $\{0, -1/(2\sqrt{3})\}$, $\{1/2, 1/\sqrt{3}\}$ and $\{-1/2, 1/\sqrt{3}\}$. The result is that the six nodes lie on a circle of radius $1/\sqrt{3}$, as depicted in Figure 24.9. The element has unit thickness. The material is isotropic with $E = 504$ and $\nu = 0$. The stiffness \mathbf{K}^e is evaluated for four rank-sufficient rules: 3, -3, 6, 7, using the script of Figure 24.10.

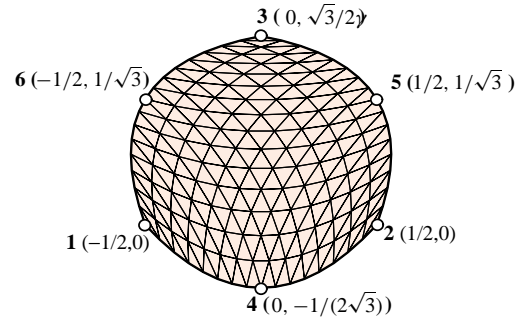


FIGURE 24.9. Test curved sided quadratic triangle, with the 6 nodes lying on a circle.

```

ClearAll[Em,v,h]; h=1; Em=7*72; v=0; h=1;
{x1,y1}={-1,0}/2; {x2,y2}={1,0}/2; {x3,y3}={0,Sqrt[3]}/2;
{x4,y4}={0,-1/Sqrt[3]}/2; {x5,y5}={1/2,1/Sqrt[3]};
{x6,y6}={-1/2,1/Sqrt[3]};
ncoor= {{x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5},{x6,y6}};
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};
For [i=2,i<=5,i++, p={1,-3,3,6,7}[i]];
  Ke=Trig6IsoPMembraneStiffness[ncoor,Emat,h,{True,p}];
  Ke=Chop[Simplify[Ke]];
  Print["Ke=",SetPrecision[Ke,4]//MatrixForm];
  Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]];
];

```

FIGURE 24.10. Script to form \mathbf{K}^e for the triangle of Figure 24.9, using four integration rules.

For rule=3 the result (printed to 4 places because of the SetPrecision statement in script) is

$$\begin{bmatrix}
 344.7 & 75.00 & -91.80 & 21.00 & -86.60 & -24.00 & -124.7 & -72.00 & -20.78 & -36.00 & -20.78 & 36.00 \\
 75.00 & 258.1 & -21.00 & -84.87 & 18.00 & -90.07 & 96.00 & 0 & -36.00 & 20.78 & -132.0 & -103.9 \\
 -91.80 & -21.00 & 344.7 & -75.00 & -86.60 & 24.00 & -124.7 & 72.00 & -20.78 & -36.00 & -20.78 & 36.00 \\
 21.00 & -84.87 & -75.00 & 258.1 & -18.00 & -90.07 & -96.00 & 0 & 132.0 & -103.9 & 36.00 & 20.78 \\
 -86.60 & 18.00 & -86.60 & -18.00 & 214.8 & 0 & 41.57 & 0 & -41.57 & 144.0 & -41.57 & -144.0 \\
 -24.00 & -90.07 & 24.00 & -90.07 & 0 & 388.0 & 0 & -41.57 & -24.00 & -83.14 & 24.00 & -83.14 \\
 -124.7 & 96.00 & -124.7 & -96.00 & 41.57 & 0 & 374.1 & 0 & -83.14 & -72.00 & -83.14 & 72.00 \\
 -72.00 & 0 & 72.00 & 0 & 0 & -41.57 & 0 & 374.1 & -72.00 & -166.3 & 72.00 & -166.3 \\
 -20.78 & -36.00 & -20.78 & 132.0 & -41.57 & -24.00 & -83.14 & -72.00 & 374.1 & 0 & -207.8 & 0 \\
 -36.00 & 20.78 & -36.00 & -103.9 & 144.0 & -83.14 & -72.00 & -166.3 & 0 & 374.1 & 0 & -41.57 \\
 -20.78 & -132.0 & -20.78 & 36.00 & -41.57 & 24.00 & -83.14 & 72.00 & -207.8 & 0 & 374.1 & 0 \\
 36.00 & -103.9 & 36.00 & 20.78 & -144.0 & -83.14 & 72.00 & -166.3 & 0 & -41.57 & 0 & 374.1
 \end{bmatrix}$$

(24.37)

For rule=-3:

$$\begin{bmatrix} 566.4 & 139.0 & 129.9 & 21.00 & 79.67 & 8.000 & -364.9 & -104.0 & -205.5 & -36.00 & -205.5 & -28.00 \\ 139.0 & 405.9 & -21.00 & 62.93 & 50.00 & 113.2 & 64.00 & -129.3 & -36.00 & -164.0 & -196.0 & -288.7 \\ 129.9 & -21.00 & 566.4 & -139.0 & 79.67 & -8.000 & -364.9 & 104.0 & -205.5 & 28.00 & -205.5 & 36.00 \\ 21.00 & 62.93 & -139.0 & 405.9 & -50.00 & 113.2 & -64.00 & -129.3 & 196.0 & -288.7 & 36.00 & -164.0 \\ 79.67 & 50.00 & 79.67 & -50.00 & 325.6 & 0 & -143.2 & 0 & -170.9 & 176.0 & -170.9 & -176.0 \\ 8.000 & 113.2 & -8.000 & 113.2 & 0 & 646.6 & 0 & -226.3 & 8.000 & -323.3 & -8.000 & -323.3 \\ -364.9 & 64.00 & -364.9 & -64.00 & -143.2 & 0 & 632.8 & 0 & 120.1 & -104.0 & 120.1 & 104.0 \\ -104.0 & -129.3 & 104.0 & -129.3 & 0 & -226.3 & 0 & 485.0 & -104.0 & 0 & 104.0 & 0 \\ -205.5 & -36.00 & -205.5 & 196.0 & -170.9 & 8.000 & 120.1 & -104.0 & 521.9 & -64.00 & -60.04 & 0 \\ -36.00 & -164.0 & 28.00 & -288.7 & 176.0 & -323.3 & -104.0 & 0 & -64.00 & 595.8 & 0 & 180.1 \\ -205.5 & -196.0 & -205.5 & 36.00 & -170.9 & -8.000 & 120.1 & 104.0 & -60.04 & 0 & 521.9 & 64.00 \\ -28.00 & -288.7 & 36.00 & -164.0 & -176.0 & -323.3 & 104.0 & 0 & 0 & 180.1 & 64.00 & 595.8 \end{bmatrix} \quad (24.38)$$

The stiffness for rule=6 is omitted to save space. For rule=7:

$$\begin{bmatrix} 661.9 & 158.5 & 141.7 & 21.00 & 92.53 & 7.407 & -432.1 & -117.2 & -190.2 & -29.10 & -273.8 & -40.61 \\ 158.5 & 478.8 & -21.00 & 76.13 & 49.41 & 125.3 & 50.79 & -182.7 & -29.10 & -156.6 & -208.6 & -341.0 \\ 141.7 & -21.00 & 661.9 & -158.5 & 92.53 & -7.407 & -432.1 & 117.2 & -273.8 & 40.61 & -190.2 & 29.10 \\ 21.00 & 76.13 & -158.5 & 478.8 & -49.41 & 125.3 & -50.79 & -182.7 & 208.6 & -341.0 & 29.10 & -156.6 \\ 92.53 & 49.41 & 92.53 & -49.41 & 387.3 & 0 & -139.8 & 0 & -216.3 & 175.4 & -216.3 & -175.4 \\ 7.407 & 125.3 & -7.407 & 125.3 & 0 & 753.4 & 0 & -207.0 & 7.407 & -398.5 & -7.407 & -398.5 \\ -432.1 & 50.79 & -432.1 & -50.79 & -139.8 & 0 & 723.6 & 0 & 140.2 & -117.2 & 140.2 & 117.2 \\ -117.2 & -182.7 & 117.2 & -182.7 & 0 & -207.0 & 0 & 562.6 & -117.2 & 4.884 & 117.2 & 4.884 \\ -190.2 & -29.10 & -273.8 & 208.6 & -216.3 & 7.407 & 140.2 & -117.2 & 602.8 & -69.71 & -62.78 & 0 \\ -29.10 & -156.6 & 40.61 & -341.0 & 175.4 & -398.5 & -117.2 & 4.884 & -69.71 & 683.3 & 0 & 207.9 \\ -273.8 & -208.6 & -190.2 & 29.10 & -216.3 & -7.407 & 140.2 & 117.2 & -62.78 & 0 & 602.8 & 69.71 \\ -40.61 & -341.0 & 29.10 & -156.6 & -175.4 & -398.5 & 117.2 & 4.884 & 0 & 207.9 & 69.71 & 683.3 \end{bmatrix} \quad (24.39)$$

The eigenvalues of these matrices are:

Rule	Eigenvalues of \mathbf{K}^e											
3	702.83	665.11	553.472	553.472	481.89	429.721	429.721	118.391	118.391	0	0	0
-3	1489.80	1489.80	702.833	665.108	523.866	523.866	481.890	196.429	196.429	0	0	0
6	1775.53	1775.53	896.833	768.948	533.970	533.970	495.570	321.181	321.181	0	0	0
7	1727.11	1727.11	880.958	760.719	532.750	532.750	494.987	312.123	312.123	0	0	0

(24.40)

Since the metric of this element is highly distorted near its boundary, the stiffness matrix entries and eigenvalues change significantly as the integration formulas are advanced from 3 to 6 and 7 points. However as can be seen the matrix remains rank-sufficient.

§24.5. *The Cubic Triangle

The 10-node cubic triangle, depicted in Figure 24.11, is rarely used as such in applications because of the difficulty of combining it with other elements. Nevertheless the derivation of the element modules is instructive as this triangle has other and more productive uses as a generator of more practical elements with “drilling” rotational degrees of freedom at corners for modeling shells. Such transformations are studied in advanced FEM courses.

The geometry of the triangle is defined by the coordinates of the ten nodes. A notational warning: the interior node is labeled as 0 instead of 10 (cf. Figure 24.8) to avoid confusion with notation such as $x_{12} = x_1 - x_2$ for coordinate differences.

§24.5.1. *Shape Function Module

The shape functions obtained in Exercise 18.1 are $N_1 = \frac{1}{2}\zeta_1(3\zeta_1 - 1)(3\zeta_1 - 2)$, $N_2 = \frac{1}{2}\zeta_2(3\zeta_2 - 1)(3\zeta_2 - 2)$, $N_3 = \frac{1}{2}\zeta_3(3\zeta_3 - 1)(3\zeta_3 - 2)$, $N_4 = \frac{9}{2}\zeta_1\zeta_2(3\zeta_1 - 1)$, $N_5 = \frac{9}{2}\zeta_1\zeta_2(3\zeta_2 - 1)$, $N_6 = \frac{9}{2}\zeta_2\zeta_3(3\zeta_2 - 1)$, $N_7 = \frac{9}{2}\zeta_2\zeta_3(3\zeta_3 - 1)$, $N_8 = \frac{9}{2}\zeta_3\zeta_1(3\zeta_3 - 1)$, $N_9 = \frac{9}{2}\zeta_3\zeta_1(3\zeta_1 - 1)$ and $N_0 = 27\zeta_1\zeta_2\zeta_3$. Their natural derivatives are

$$\frac{\partial \mathbf{N}^T}{\partial \zeta_1} = \frac{9}{2} \begin{bmatrix} \frac{2}{9} - 2\zeta_1 + 3\zeta_1^2 \\ 0 \\ 0 \\ \zeta_2(6\zeta_1 - 1) \\ \zeta_2(3\zeta_2 - 1) \\ 0 \\ 0 \\ \zeta_3(3\zeta_3 - 1) \\ \zeta_3(6\zeta_1 - 1) \\ 6\zeta_2\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial \zeta_2} = \frac{9}{2} \begin{bmatrix} 0 \\ \frac{2}{9} - 2\zeta_2 + 3\zeta_2^2 \\ 0 \\ \zeta_1(3\zeta_1 - 1) \\ \zeta_1(6\zeta_2 - 1) \\ \zeta_3(6\zeta_2 - 1) \\ \zeta_3(3\zeta_3 - 1) \\ 0 \\ 0 \\ 6\zeta_1\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial \zeta_3} = \frac{9}{2} \begin{bmatrix} 0 \\ 0 \\ \frac{2}{9} - 2\zeta_3 + 3\zeta_3^2 \\ \zeta_2(3\zeta_2 - 1) \\ \zeta_2(6\zeta_3 - 1) \\ \zeta_1(6\zeta_3 - 1) \\ \zeta_1(3\zeta_1 - 1) \\ 6\zeta_1\zeta_2 \end{bmatrix}. \quad (24.41)$$

As in the case of the 6-node triangle it is convenient to introduce the deviations from thirdpoints and centroid: $\Delta x_4 = x_4 - \frac{1}{3}(2x_1 + x_2)$, $\Delta x_5 = x_5 - \frac{1}{3}(x_1 + 2x_2)$, $\Delta x_6 = x_6 - \frac{1}{3}(2x_2 + x_3)$, $\Delta x_7 = x_7 - \frac{1}{3}(x_2 + 2x_3)$, $\Delta x_8 = x_8 - \frac{1}{3}(2x_3 + x_1)$, $\Delta x_9 = x_9 - \frac{1}{3}(x_3 + 2x_1)$, $\Delta x_0 = x_0 - \frac{1}{3}(x_1 + x_2 + x_3)$, $\Delta y_4 = y_4 - \frac{1}{3}(2y_1 + y_2)$, $\Delta y_5 = y_5 - \frac{1}{3}(y_1 + 2y_2)$, $\Delta y_6 = y_6 - \frac{1}{3}(2y_2 + y_3)$, $\Delta y_7 = y_7 - \frac{1}{3}(y_2 + 2y_3)$, $\Delta y_8 = y_8 - \frac{1}{3}(2y_3 + y_1)$, $\Delta y_9 = y_9 - \frac{1}{3}(y_3 + 2y_1)$, and $\Delta y_0 = y_0 - \frac{1}{3}(y_1 + y_2 + y_3)$.

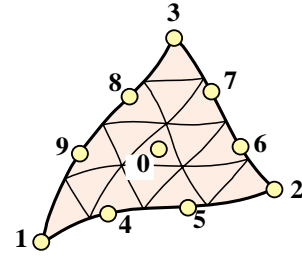


FIGURE 24.11. The 10-node cubic triangle.

Using (24.22) and (24.41), the shape function partial derivatives can be worked out to be

$$\frac{\partial \mathbf{N}^T}{\partial x} = \frac{9}{4J} \begin{bmatrix} J_{y23}(\frac{2}{9} - 2\zeta_1 + 3\zeta_1^2/2) \\ J_{y31}(\frac{2}{9} - 2\zeta_2 + 3\zeta_2^2/2) \\ J_{y12}(\frac{2}{9} - 2\zeta_3 + 3\zeta_3^2/2) \\ J_{y31}\zeta_1(3\zeta_1 - 1) + J_{y23}\zeta_2(6\zeta_1 - 1) \\ J_{y23}\zeta_2(3\zeta_2 - 1) + J_{y31}\zeta_1(6\zeta_2 - 1) \\ J_{y12}\zeta_2(3\zeta_2 - 1) + J_{y31}\zeta_3(6\zeta_2 - 1) \\ J_{y31}\zeta_3(3\zeta_3 - 1) + J_{y12}\zeta_2(6\zeta_3 - 1) \\ J_{y23}\zeta_3(3\zeta_3 - 1) + J_{y12}\zeta_1(6\zeta_3 - 1) \\ J_{y12}\zeta_1(3\zeta_1 - 1) + J_{y23}\zeta_3(6\zeta_1 - 1) \\ 6(J_{y12}\zeta_1\zeta_2 + J_{y31}\zeta_1\zeta_3 + J_{y23}\zeta_2\zeta_3) \end{bmatrix}, \quad \frac{\partial \mathbf{N}^T}{\partial y} = \frac{9}{4J} \begin{bmatrix} J_{x32}(\frac{2}{9} - 2\zeta_1 + 3\zeta_1^2/2) \\ J_{x13}(\frac{2}{9} - 2\zeta_2 + 3\zeta_2^2/2) \\ J_{x21}(\frac{2}{9} - 2\zeta_3 + 3\zeta_3^2/2) \\ J_{x13}\zeta_1(3\zeta_1 - 1) + J_{x32}\zeta_2(6\zeta_1 - 1) \\ J_{x32}\zeta_2(3\zeta_2 - 1) + J_{x13}\zeta_1(6\zeta_2 - 1) \\ J_{x21}\zeta_2(3\zeta_2 - 1) + J_{x13}\zeta_3(6\zeta_2 - 1) \\ J_{x13}\zeta_3(3\zeta_3 - 1) + J_{x21}\zeta_2(6\zeta_3 - 1) \\ J_{x32}\zeta_3(3\zeta_3 - 1) + J_{x21}\zeta_1(6\zeta_3 - 1) \\ J_{x21}\zeta_1(3\zeta_1 - 1) + J_{x32}\zeta_3(6\zeta_1 - 1) \\ 6(J_{x21}\zeta_1\zeta_2 + J_{x13}\zeta_1\zeta_3 + J_{x32}\zeta_2\zeta_3) \end{bmatrix}. \quad (24.42)$$

where the expressions of the Jacobian coefficients are

$$\begin{aligned} J_{x21} &= x_{21} + \frac{9}{2} [\Delta x_4(\zeta_1(3\zeta_1 - 6\zeta_2 - 1) + \zeta_2) + \Delta x_5(\zeta_2(1 - 3\zeta_2 + 6\zeta_1) - \zeta_1) + \Delta x_6\zeta_3(6\zeta_2 - 1) \\ &\quad + \Delta x_7\zeta_3(3\zeta_3 - 1) + \Delta x_8\zeta_3(1 - 3\zeta_3) + \Delta x_9\zeta_3(1 - 6\zeta_1) + 6\Delta x_0\zeta_3(\zeta_1 - \zeta_2)], \\ J_{x32} &= x_{32} + \frac{9}{2} [\Delta x_4\zeta_1(1 - 3\zeta_1) + \Delta x_5\zeta_1(1 - 6\zeta_2) + \Delta x_6(\zeta_2(3\zeta_2 - 6\zeta_3 - 1) + \zeta_3) \\ &\quad + \Delta x_7(\zeta_3(1 - 3\zeta_3 + 6\zeta_2) - \zeta_2) + \Delta x_8\zeta_1(6\zeta_3 - 1) + \Delta x_9\zeta_1(3\zeta_1 - 1) + 6\Delta x_0\zeta_1(\zeta_2 - \zeta_3)], \\ J_{x13} &= x_{13} + \frac{9}{2} [\Delta x_4(6\zeta_1 - 1)\zeta_2 + \Delta x_5\zeta_2(3\zeta_2 - 1) + \Delta x_6\zeta_2(1 - 3\zeta_2) + \Delta x_7\zeta_2(1 - 6\zeta_3) \\ &\quad + \Delta x_8(\zeta_3(3\zeta_3 - 6\zeta_1 - 1) + \zeta_1) + \Delta x_9(\zeta_1(1 - 3\zeta_1 + 6\zeta_3) - \zeta_3) + 6\Delta x_0\zeta_2(\zeta_3 - \zeta_1)], \end{aligned} \quad (24.43)$$

```

Trig10IsoPShapeFunDer[ncoor_,tcoor_]:= Module[
{ζ1,ζ2,ζ3,x1,x2,x3,x4,x5,x6,x7,x8,x9,x0,y1,y2,y3,y4,y5,y6,y7,y8,y9,y0,
dx4,dx5,dx6,dx7,dx8,dx9,dx0,dy4,dy5,dy6,dy7,dy8,dy9,dy0,
Jx21,Jx32,Jx13,Jy12,Jy23,Jy31,Nf,dNx,dNy,Jdet},
{{x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5},{x6,y6},{x7,y7},
{x8,y8},{x9,y9},{x0,y0}}=ncoor; {ζ1,ζ2,ζ3}=tcoor;
dx4=x4-(2*x1+x2)/3; dx5=x5-(x1+2*x2)/3; dx6=x6-(2*x2+x3)/3;
dx7=x7-(x2+2*x3)/3; dx8=x8-(2*x3+x1)/3; dx9=x9-(x3+2*x1)/3;
dy4=y4-(2*y1+y2)/3; dy5=y5-(y1+2*y2)/3; dy6=y6-(2*y2+y3)/3;
dy7=y7-(y2+2*y3)/3; dy8=y8-(2*y3+y1)/3; dy9=y9-(y3+2*y1)/3;
dx0=x0-(x1+x2+x3)/3; dy0=y0-(y1+y2+y3)/3;
Nf={ζ1*(3*ζ1-1)*(3*ζ1-2),ζ2*(3*ζ2-1)*(3*ζ2-2),ζ3*(3*ζ3-1)*(3*ζ3-2),
9*ζ1*ζ2*(3*ζ1-1),9*ζ1*ζ2*(3*ζ2-1),9*ζ2*ζ3*(3*ζ2-1),
9*ζ2*ζ3*(3*ζ3-1),9*ζ3*ζ1*(3*ζ3-1),9*ζ3*ζ1*(3*ζ1-1),54*ζ1*ζ2*ζ3}/2;
Jx21=x2-x1+(9/2)*(dx4*(ζ1*(3*ζ1-6*ζ2-1)+ζ2)+
dx5*(ζ2*(1-3*ζ2+6*ζ1)-ζ1)+dx6*ζ3*(6*ζ2-1)+dx7*ζ3*(3*ζ3-1)+
dx8*ζ3*(1-3*ζ3)+dx9*ζ3*(1-6*ζ1)+6*dx0*ζ3*(ζ1-ζ2));
Jx32=x3-x2+(9/2)*(dx4*ζ1*(1-3*ζ1)+dx5*ζ1*(1-6*ζ2)+
dx6*(ζ2*(3*ζ2-6*ζ3-1)+ζ3)+dx7*(ζ3*(1-3*ζ3+6*ζ2)-ζ2)+
dx8*ζ1*(6*ζ3-1)+dx9*ζ1*(3*ζ1-1)+6*dx0*ζ1*(ζ2-ζ3));
Jx13=x1-x3+(9/2)*(dx4*(6*ζ1-1)*ζ2+dx5*ζ2*(3*ζ2-1)+
dx6*ζ2*(1-3*ζ2)+dx7*ζ2*(1-6*ζ3)+dx8*(ζ3*(3*ζ3-6*ζ1-1)+ζ1)+
dx9*(ζ1*(1-3*ζ1+6*ζ3)-ζ3)+6*dx0*ζ2*(ζ3-ζ1));
Jy12=y1-y2-(9/2)*(dy4*(ζ1*(3*ζ1-6*ζ2-1)+ζ2)+
dy5*(ζ2*(1-3*ζ2+6*ζ1)-ζ1)+dy6*ζ3*(6*ζ2-1)+dy7*ζ3*(3*ζ3-1)+
dy8*ζ3*(1-3*ζ3)+dy9*ζ3*(1-6*ζ1)+6*dy0*ζ3*(ζ1-ζ2));
Jy23=y2-y3-(9/2)*(dy4*ζ1*(1-3*ζ1)+dy5*ζ1*(1-6*ζ2)+
dy6*(ζ2*(3*ζ2-6*ζ3-1)+ζ3)+dy7*(ζ3*(1-3*ζ3+6*ζ2)-ζ2)+
dy8*ζ1*(6*ζ3-1)+dy9*ζ1*(3*ζ1-1)+6*dy0*ζ1*(ζ2-ζ3));
Jy31=y3-y1-(9/2)*(dy4*(6*ζ1-1)*ζ2+dy5*ζ2*(3*ζ2-1)+
dy6*ζ2*(1-3*ζ2)+dy7*ζ2*(1-6*ζ3)+dy8*(ζ3*(3*ζ3-6*ζ1-1)+ζ1)+
dy9*(ζ1*(1-3*ζ1+6*ζ3)-ζ3)+6*dy0*ζ2*(ζ3-ζ1));
Jdet = Jx21*Jy31-Jy12*Jx13;
dNx={Jy23*(2/9-2*ζ1+3*ζ1^2),Jy31*(2/9-2*ζ2+3*ζ2^2),Jy12*(2/9-2*ζ3+3*ζ3^2),
Jy31*ζ1*(3*ζ1-1)+Jy23*ζ2*(6*ζ1-1),Jy23*ζ2*(3*ζ2-1)+Jy31*ζ1*(6*ζ2-1),
Jy12*ζ2*(3*ζ2-1)+Jy31*ζ3*(6*ζ2-1),Jy31*ζ3*(3*ζ3-1)+Jy12*ζ2*(6*ζ3-1),
Jy23*ζ3*(3*ζ3-1)+Jy12*ζ1*(6*ζ3-1),Jy12*ζ1*(3*ζ1-1)+Jy23*ζ3*(6*ζ1-1),
6*(Jy12*ζ1*ζ2+Jy31*ζ1*ζ3+Jy23*ζ2*ζ3)}/(2*Jdet/9);
dNy={Jx32*(2/9-2*ζ1+3*ζ1^2),Jx13*(2/9-2*ζ2+3*ζ2^2),Jx21*(2/9-2*ζ3+3*ζ3^2),
Jx13*ζ1*(3*ζ1-1)+Jx32*ζ2*(6*ζ1-1),Jx32*ζ2*(3*ζ2-1)+Jx13*ζ1*(6*ζ2-1),
Jx21*ζ2*(3*ζ2-1)+Jx13*ζ3*(6*ζ2-1),Jx13*ζ3*(3*ζ3-1)+Jx21*ζ2*(6*ζ3-1),
Jx32*ζ3*(3*ζ3-1)+Jx21*ζ1*(6*ζ3-1),Jx21*ζ1*(3*ζ1-1)+Jx32*ζ3*(6*ζ1-1),
6*(Jx21*ζ1*ζ2+Jx13*ζ1*ζ3+Jx32*ζ2*ζ3)}/(2*Jdet/9);
Return[Simplify[{Nf,dNx,dNy,Jdet}]]
];

```

FIGURE 24.12. Shape function module for 10-node cubic triangle.

and

$$\begin{aligned}
J_{y12} &= y_{12} - \frac{9}{2} \left[\Delta y_4 (\zeta_1 (3\zeta_1 - 6\zeta_2 - 1) + \zeta_2) + \Delta y_5 (\zeta_2 (1 - 3\zeta_2 + 6\zeta_1) - \zeta_1) + \Delta y_6 \zeta_3 (6\zeta_2 - 1) \right. \\
&\quad \left. + \Delta y_7 \zeta_3 (3\zeta_3 - 1) + \Delta y_8 \zeta_3 (1 - 3\zeta_3) + \Delta y_9 \zeta_3 (1 - 6\zeta_1) + 6\Delta y_0 \zeta_3 (\zeta_1 - \zeta_2) \right], \\
J_{y23} &= y_{23} - \frac{9}{2} \left[\Delta y_4 \zeta_1 (1 - 3\zeta_1) + \Delta y_5 \zeta_1 (1 - 6\zeta_2) + \Delta y_6 (\zeta_2 (3\zeta_2 - 6\zeta_3 - 1) + \zeta_3) \right. \\
&\quad \left. + \Delta y_7 (\zeta_3 (1 - 3\zeta_3 + 6\zeta_2) - \zeta_2) + \Delta y_8 \zeta_1 (6\zeta_3 - 1) + \Delta y_9 \zeta_1 (3\zeta_1 - 1) + 6\Delta y_0 \zeta_1 (\zeta_2 - \zeta_3) \right], \\
J_{y31} &= y_{31} - \frac{9}{2} \left[\Delta y_4 (6\zeta_1 - 1) \zeta_2 + \Delta y_5 \zeta_2 (3\zeta_2 - 1) + \Delta y_6 \zeta_2 (1 - 3\zeta_2) + \Delta y_7 \zeta_2 (1 - 6\zeta_3) \right. \\
&\quad \left. + \Delta y_8 (\zeta_3 (3\zeta_3 - 6\zeta_1 - 1) + \zeta_1) + \Delta y_9 (\zeta_1 (1 - 3\zeta_1 + 6\zeta_3) - \zeta_3) + 6\Delta y_0 \zeta_2 (\zeta_3 - \zeta_1) \right].
\end{aligned} \tag{24.44}$$

The Jacobian determinant is $J = \frac{1}{2}(J_{x21}J_{y31} - J_{y12}J_{x13})$. The shape function module Trig10IsoPShapeFunDer that implements these expressions is listed in Figure 24.12. This has the same arguments as Trig6IsoPShapeFunDer. As there Jdet denotes $2J$.

```

Trig10IsoPMembraneStiffness[ncoor_,Emat_,th_,options_]:=
Module[{i,k,l,p=6,numer=False,h=th,tcoor,w,c,
Nf,dNx,dNy,Jdet,Be,Ke=Table[0,{20},{20}]},
If [Length[options]>=1, numer=options[[1]]];
If [Length[options]>=2, p= options[[2]]];
If [!MemberQ[{1,3,-3,6,7},p], Print["Illegal p"]; Return[Null]];
For [k=1, k<=Abs[p], k++,
{tcoor,w}= TrigGaussRuleInfo[{p,numer},k];
{Nf,dNx,dNy,Jdet}= Trig10IsoPShapeFunDer[ncoor,tcoor];
If [numer, {Nf,dNx,dNy,Jdet}=N[{Nf,dNx,dNy,Jdet}]];
If [Length[th]==10, h=th.Nf]; c=w*Jdet*h/2;
Be= {Flatten[Table[{dNx[[i]],0},{i,10}]],
Flatten[Table[{0,dNy[[i]]},{i,10}]],
Flatten[Table[{dNy[[i]],dNx[[i]]},{i,10}]]};
Ke+=c*Transpose[Be].(Emat.Be);
]; If[!numer,Ke=Simplify[Ke]]; Return[Ke]
];

```

FIGURE 24.13. Stiffness module for 10-node cubic triangle.

§24.5.2. *Stiffness Module

Module Trig10IsoPMembraneStiffness, listed in Figure 24.13, implements the computation of the element stiffness matrix of the 10-node cubic plane stress triangle. The module is invoked as

$$Ke = \text{Trig10IsoPMembraneStiffness}[ncoor,Emat,th,options] \quad (24.45)$$

this has the same arguments of the quadratic triangle module invoked in (24.32), with the following changes. The node coordinate list $ncoor$ contains 10 entries: $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}, \dots, \{x_9, y_9\}, \{x_0, y_0\}\}$. If the plate thickness varies, th is a list of 10 entries: $\{h_1, h_2, h_3, h_4, \dots, h_9, h_0\}$. The triangle Gauss rule specified in options as $\{numer, rule\}$ should be 6 or 7 to produce a rank sufficient stiffness. If omitted $rule = 6$ is assumed. The module returns Ke as an 20×20 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2} \ u_{x3} \ u_{y3} \ u_{x4} \ u_{y4} \ \dots \ u_{x9} \ u_{y9} \ u_{x0} \ u_{y0}]^T. \quad (24.46)$$

Remark 24.3. For symbolic work with this element the 7-point rule should be preferred because the exact expressions of the abscissas and weights at sample points are simpler than for the 6-point rule, as can be observed in Figure 24.3. This speeds up algebraic simplification. For numerical work the 6-point rule is slightly faster.

§24.5.3. *Test Element

The stiffness module is tested on the superparametric triangle geometry shown in Figure 24.14, which has been already used for the quadratic triangle.

The corner nodes are placed at $(0, 0)$, $(6, 2)$ and $(4, 4)$. The 6 side nodes are located at the thirdpoints of the sides and the interior node at the centroid. The plate has unit thickness. The material is isotropic with $E = 1920$ and $\nu = 0$.

The script of Figure 24.15 computes \mathbf{K}^e using the 7-point Gauss rule and exact arithmetic.

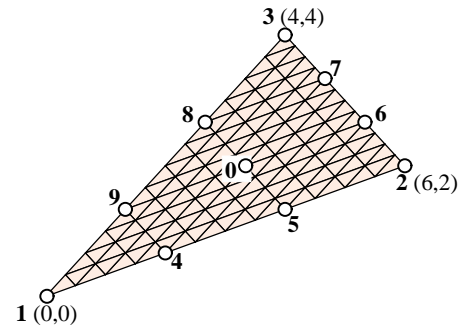


FIGURE 24.14. Test 10-node triangle.

The returned stiffness matrix using either 6- or 7-point integration is the same, since for a superparametric element the integrand is quartic in the triangular coordinates. For the given combination of inputs the entries are exact integers:

$$\mathbf{K}^e = \begin{bmatrix} 306 & 102 & -42 & -42 & -21 & 21 & -315 & -333 & 171 & 153 \\ 102 & 306 & 42 & 42 & -63 & -105 & 351 & 369 & -135 & -117 \\ -42 & 42 & 1224 & -408 & -210 & 42 & 252 & -180 & -234 & 306 \\ -42 & 42 & -408 & 1224 & 126 & -294 & 108 & -36 & -378 & 450 \\ -21 & -63 & -210 & 126 & 1122 & -306 & -99 & 27 & -99 & 27 \\ 21 & -105 & 42 & -294 & -306 & 1938 & 27 & -171 & 27 & -171 \\ -315 & 351 & 252 & 108 & -99 & 27 & 5265 & -1215 & -1539 & -243 \\ -333 & 369 & -180 & -36 & 27 & -171 & -1215 & 6885 & 729 & -891 \\ 171 & -135 & -234 & -378 & -99 & 27 & -1539 & 729 & 5265 & -1215 \\ 153 & -117 & 306 & 450 & 27 & -171 & -243 & -891 & -1215 & 6885 \\ -27 & -9 & -1602 & 990 & 819 & -171 & 81 & 81 & -405 & -405 \\ -9 & -27 & 306 & -2286 & -459 & 1179 & 81 & 405 & -405 & -2025 \\ -27 & -9 & 828 & -468 & -1611 & 315 & 81 & 81 & 81 & 81 \\ -9 & -27 & -180 & 1116 & 999 & -2223 & 81 & 405 & 81 & 405 \\ 99 & 225 & -108 & 36 & -72 & 144 & 810 & -324 & 810 & -324 \\ -63 & 387 & 36 & -108 & -540 & -684 & -324 & 1134 & -324 & 1134 \\ -144 & -504 & -108 & 36 & 171 & -99 & -4050 & 1620 & 810 & -324 \\ 180 & -828 & 36 & -108 & 189 & 531 & 1620 & -5670 & -324 & 1134 \\ 0 & 0 & 0 & 0 & 0 & 0 & -486 & -486 & -4860 & 1944 \\ 0 & 0 & 0 & 0 & 0 & 0 & -486 & -2430 & 1944 & -6804 \end{bmatrix}$$

$$\begin{bmatrix} -27 & -9 & -27 & -9 & 99 & -63 & -144 & 180 & 0 & 0 \\ -9 & -27 & -9 & -27 & 225 & 387 & -504 & -828 & 0 & 0 \\ -1602 & 306 & 828 & -180 & -108 & 36 & -108 & 36 & 0 & 0 \\ 990 & -2286 & -468 & 1116 & 36 & -108 & 36 & -108 & 0 & 0 \\ 819 & -459 & -1611 & 999 & -72 & -540 & 171 & 189 & 0 & 0 \\ -171 & 1179 & 315 & -2223 & 144 & -684 & -99 & 531 & 0 & 0 \\ 81 & 81 & 81 & 81 & 810 & -324 & -4050 & 1620 & -486 & -486 \\ 81 & 405 & 81 & 405 & -324 & 1134 & 1620 & -5670 & -486 & -2430 \\ -405 & -405 & 81 & 81 & 810 & -324 & 810 & -324 & -4860 & 1944 \\ -405 & -2025 & 81 & 405 & -324 & 1134 & -324 & 1134 & 1944 & -6804 \\ 5265 & -1215 & -3483 & 729 & 162 & 0 & 162 & 0 & -972 & 0 \\ -1215 & 6885 & 1701 & -4779 & 0 & -162 & 0 & -162 & 0 & 972 \\ -3483 & 1701 & 5265 & -1215 & -810 & 0 & 162 & 0 & -486 & -486 \\ 729 & -4779 & -1215 & 6885 & 0 & 810 & 0 & -162 & -486 & -2430 \\ 162 & 0 & -810 & 0 & 5265 & -1215 & -1296 & -486 & -4860 & 1944 \\ 0 & -162 & 0 & 810 & -1215 & 6885 & 486 & -2592 & 1944 & -6804 \\ 162 & 0 & 162 & 0 & -1296 & 486 & 5265 & -1215 & -972 & 0 \\ 0 & -162 & 0 & -162 & -486 & -2592 & -1215 & 6885 & 0 & 972 \\ -972 & 0 & -486 & -486 & -4860 & 1944 & -972 & 0 & 12636 & -2916 \\ 0 & 972 & -486 & -2430 & 1944 & -6804 & 0 & 972 & -2916 & 16524 \end{bmatrix} \quad (24.47)$$

The eigenvalues are

$$\begin{bmatrix} 26397. & 16597. & 14937. & 12285. & 8900.7 & 7626.2 & 5417.8 & 4088.8 & 3466.8 & 3046.4 \\ 1751.3 & 1721.4 & 797.70 & 551.82 & 313.22 & 254.00 & 28.019 & 0 & 0 & 0 \end{bmatrix} \quad (24.48)$$

The 3 zero eigenvalues pertain to the three independent rigid-body modes in two dimensions. The 17 other eigenvalues are positive. Consequently the computed \mathbf{K}^e has the correct rank of 17.

```

ClearAll[Em,v,a,b,e,h]; Em=1920; v=0; h=1;
ncoor={{0,0},{6,2},{4,4}};
x4=(2*x1+x2)/3; x5=(x1+2*x2)/3; y4=(2*y1+y2)/3; y5=(y1+2*y2)/3;
x6=(2*x2+x3)/3; x7=(x2+2*x3)/3; y6=(2*y2+y3)/3; y7=(y2+2*y3)/3;
x8=(2*x3+x1)/3; x9=(x3+2*x1)/3; y8=(2*y3+y1)/3; y9=(y3+2*y1)/3;
x0=(x1+x2+x3)/3; y0=(y1+y2+y3)/3;
ncoor={{x1,y1},{x2,y2},{x3,y3},{x4,y4},{x5,y5},{x6,y6},{x7,y7},
      {x8,y8},{x9,y9},{x0,y0}};
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};
Ke=Trig10IsoPMembraneStiffness[ncoor,Emat,h,{False,7}];
Ke=Simplify[Ke]; Print[Ke//MatrixForm]; ev=Chop[Eigenvalues[N[Ke]]];
Print["eigs of Ke=",ev];

```

FIGURE 24.15. Script for testing cubic triangle of Figure 24.14.

Notes and Bibliography

The 3-node, 6-node and 10-node plane stress triangular elements are generated by complete polynomials in two-dimensions. In order of historical appearance:

1. The three-node linear triangle, also known as *Constant Strain Triangle* (CST) and Turner triangle, was developed as “triangular skin panel” by Turner, Clough and Martin in 1951–53 [68,69] using interelement flux assumptions, and published in 1956 [392]. It is not clear when the assumed-displacement derivation, which yields the same stiffness matrix, was done first. The displacement derivation is mentioned in passing by Clough in [61] and worked out in the theses of Melosh [261] and Wilson [414].
2. The six-node quadratic triangle, also known as *Linear Strain Triangle* (LST) and Veubeke triangle, was developed by B. M. Fraeijs de Veubeke in 1962–63 [433]; published 1965 [149].
3. The ten-node cubic triangle, also known as *Quadratic Strain Triangle* (QST), was developed by the writer in 1965; published 1966 [100]. Shape functions for the cubic triangle were presented there but used for plate bending instead of plane stress.

A version of the cubic triangle with freedoms migrated to corners and recombined to produce a “drilling rotation” at corners, was used in static and dynamic shell analysis in Carr’s thesis under Ray Clough [57,65]. The drilling-freedom idea was independently exploited for rectangular and quadrilateral elements, respectively, in the theses of Abu-Ghazaleh [2] and Willam [413], both under Alex Scordelis. A variant of the Willam quadrilateral, developed by Bo Almroth at Lockheed, has survived in the nonlinear shell analysis code STAGS as element 410 [326].

Numerical integration came into FEM by the mid 1960s. Five triangle integration rules were tabulated in the writer’s thesis [100, pp. 38–39]. These were gathered from three sources: two papers by Hammer and Stroud [178,179] and the 1964 Handbook of Mathematical Functions [1, §25.4]. They were adapted to FEM by converting Cartesian abscissas to triangle coordinates. The table has been reproduced in Zienkiewicz’ book since the second edition [431, Table 8.2] and, with corrections and additions, in the monograph of Strang and Fix [360, p. 184].

The monograph by Stroud [363] contains a comprehensive collection of quadrature formulas for multiple integrals. That book gathers most of the formulas known by 1970, as well as references until that year. (Only a small fraction of Stroud’s tabulated rules, however, are suitable for FEM work.) The collection has been periodically kept up to date by Cools [73–75], who also maintains a dedicated web site: <http://www.cs.kuleuven.ac.be/~nines/research/ecf/ecf.html> This site provides rule information in 16- and 32-digit accuracy for many geometries and dimensionalities as well as a linked “index card” of references to source publications.

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 24

Implementation of Iso-P Triangular Elements

EXERCISE 24.1 [C:20] Write an element stiffness module and a shape function module for the 4-node “transition” iso-P triangular element with only one side node: 4, which is located between 1 and 2. See Figure E24.1. The shape functions are $N_1 = \zeta_1 - 2\zeta_1\zeta_2$, $N_2 = \zeta_2 - 2\zeta_1\zeta_2$, $N_3 = \zeta_3$ and $N_4 = 4\zeta_1\zeta_2$. Use the three interior point integration rule=3.

Test the element for the geometry of the triangle depicted in Figure 24.7, removing nodes 5 and 6, and with $E = 2880$, $\nu = 1/3$ and $h = 1$. Report results for the stiffness matrix \mathbf{K}^e and its 8 eigenvalues. Note: Notebook Trig6Stiffness.nb for the 6-node triangle, posted on the index of Chapter 24, may be used as “template” in support of this Exercise. Partial results: $K_{11} = 1980$, $K_{18} = 1440$.

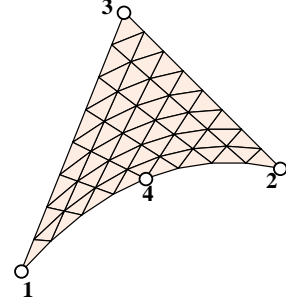


FIGURE E24.1. The 4-node transition triangle for Exercise 24.1.

EXERCISE 24.2 [A/C:5+20] As in the foregoing Exercise, but now write the module for a five-node triangular “transition” element that lacks midnode 6 opposite corner 2. Begin by deriving the five shape functions.

EXERCISE 24.3 [A+C:25] Consider the superparametric straight-sided 6-node triangle where side nodes are located at the midpoints. By setting $x_4 = \frac{1}{2}(x_1 + x_2)$, $x_5 = \frac{1}{2}(x_2 + x_3)$, $x_3 = \frac{1}{2}(x_3 + x_1)$, $y_4 = \frac{1}{2}(y_1 + y_2)$, $y_5 = \frac{1}{2}(y_2 + y_3)$, $y_3 = \frac{1}{2}(y_3 + y_1)$ in (24.26), deduce that

$$\mathbf{J} = \begin{bmatrix} 1 & 1 & 1 \\ 3x_1 & 2x_1 + x_2 & 2x_1 + x_3 \\ 3y_1 & 2y_1 + y_2 & 2y_1 + y_3 \end{bmatrix} \zeta_1 + \begin{bmatrix} 1 & 1 & 1 \\ x_1 + 2x_2 & 3x_2 & 2x_2 + x_3 \\ y_1 + 2y_2 & 3y_2 & 2y_2 + y_3 \end{bmatrix} \zeta_2 + \begin{bmatrix} 1 & 1 & 1 \\ x_1 + 2x_3 & x_2 + 2x_3 & 3x_3 \\ y_1 + 2y_3 & y_2 + 2y_3 & 3y_3 \end{bmatrix} \zeta_3. \quad (\text{E24.1})$$

This contradicts publications that, by mistakenly assuming that the results for the linear triangle (Chapter 15) can be extended by analogy, take

$$\mathbf{J} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \quad (\text{E24.2})$$

Show, however, that

$$2J = 2A = \det \mathbf{J} = x_3y_{12} + x_1y_{23} + x_2y_{31}, \quad \mathbf{P} = \frac{1}{2J} \begin{bmatrix} y_{23} & y_{32} \\ y_{31} & x_{13} \\ y_{12} & x_{21} \end{bmatrix} \quad (\text{E24.3})$$

are the same for both (E24.1) and (E24.2). Thus the mistake has no effect on the computation of derivatives.

EXERCISE 24.4 [A/C:15+15] Consider the superparametric straight-sided 6-node triangle where side nodes are at the midpoints of the sides and the thickness h is constant. Using the 3-midpoint quadrature rule show that the element stiffness can be expressed in a closed form obtained in 1966 [100]:

$$\mathbf{K}^e = \frac{1}{3}Ah (\mathbf{B}_1^T \mathbf{E} \mathbf{B}_1 + \mathbf{B}_2^T \mathbf{E} \mathbf{B}_2 + \mathbf{B}_3^T \mathbf{E} \mathbf{B}_3) \quad (\text{E24.4})$$

in which A is the triangle area and

$$\begin{aligned} \mathbf{B}_1 &= \frac{1}{2A} \begin{bmatrix} y_{32} & 0 & y_{31} & 0 & y_{12} & 0 & 2y_{23} & 0 & 2y_{32} & 0 & 2y_{23} & 0 \\ 0 & x_{23} & 0 & x_{13} & 0 & x_{21} & 0 & 2x_{32} & 0 & 2x_{23} & 0 & 2x_{32} \\ x_{23} & y_{32} & x_{13} & y_{31} & x_{21} & y_{12} & 2x_{32} & 2y_{23} & 2x_{23} & 2y_{32} & 2x_{32} & 2y_{23} \end{bmatrix} \\ \mathbf{B}_2 &= \frac{1}{2A} \begin{bmatrix} y_{23} & 0 & y_{13} & 0 & y_{12} & 0 & 2y_{31} & 0 & 2y_{31} & 0 & 2y_{13} & 0 \\ 0 & x_{32} & 0 & x_{31} & 0 & x_{21} & 0 & 2x_{13} & 0 & 2x_{13} & 0 & 2x_{31} \\ x_{32} & y_{23} & x_{31} & y_{13} & x_{21} & y_{12} & 2x_{13} & 2y_{31} & 2x_{13} & 2y_{31} & 2x_{31} & 2y_{13} \end{bmatrix} \\ \mathbf{B}_3 &= \frac{1}{2A} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{21} & 0 & 2y_{21} & 0 & 2y_{12} & 0 & 2y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{12} & 0 & 2x_{12} & 0 & 2x_{21} & 0 & 2x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{12} & y_{21} & 2x_{12} & 2y_{21} & 2x_{21} & 2y_{12} & 2x_{21} & 2y_{12} \end{bmatrix} \end{aligned} \quad (\text{E24.5})$$

With this form \mathbf{K}^e can be computed in approximately 1000 floating-point operations.

Using next the 3-interior-point quadrature rule, show that the element stiffness can be expressed again as (E24.4) but with

$$\begin{aligned}\mathbf{B}_1 &= \frac{1}{6A} \begin{bmatrix} 5y_{23} & 0 & y_{13} & 0 & y_{21} & 0 & 2y_{21} + 6y_{31} & 0 & 2y_{32} & 0 & 6y_{12} + 2y_{13} & 0 \\ 0 & 5x_{32} & 0 & x_{31} & 0 & x_{12} & 0 & 2x_{12} + 6x_{13} & 0 & 2x_{23} & 0 & 6x_{21} + 2x_{31} \\ 5x_{32} & 5y_{23} & x_{31} & y_{13} & x_{12} & y_{21} & 2x_{12} + 6x_{13} & 2y_{21} + 6y_{31} & 2x_{23} & 2y_{32} & 6x_{21} + 2x_{31} & 6y_{12} + 2y_{13} \end{bmatrix} \\ \mathbf{B}_2 &= \frac{1}{6A} \begin{bmatrix} y_{32} & 0 & 5y_{31} & 0 & y_{21} & 0 & 2y_{21} + 6y_{23} & 0 & 6y_{12} + 2y_{32} & 0 & 2y_{13} & 0 \\ 0 & x_{23} & 0 & 5x_{13} & 0 & x_{12} & 0 & 2x_{12} + 6x_{32} & 0 & 6x_{21} + 2x_{23} & 0 & 2x_{31} \\ x_{23} & y_{32} & 5x_{13} & 5y_{31} & x_{12} & y_{21} & 2x_{12} + 6x_{32} & 2y_{21} + 6y_{23} & 6x_{21} + 2x_{23} & 6y_{12} + 2y_{32} & 2x_{31} & 2y_{13} \end{bmatrix} \\ \mathbf{B}_3 &= \frac{1}{6A} \begin{bmatrix} y_{32} & 0 & y_{13} & 0 & 5y_{12} & 0 & 2y_{21} & 0 & 6y_{31} + 2y_{32} & 0 & 2y_{13} + 6y_{23} & 0 \\ 0 & x_{23} & 0 & x_{31} & 0 & 5x_{21} & 0 & 2x_{12} & 0 & 6x_{13} + 2x_{23} & 0 & 2x_{31} + 6x_{32} \\ x_{23} & y_{32} & x_{31} & y_{13} & 5x_{21} & 5y_{12} & 2x_{12} & 2y_{21} & 6x_{13} + 2x_{23} & 6y_{31} + 2y_{32} & 2x_{31} + 6x_{32} & 2y_{13} + 6y_{23} \end{bmatrix}\end{aligned}\quad (\text{E24.6})$$

The fact that two very different expressions yield the same \mathbf{K}^e explains why sometimes authors rediscover the same element derived with different methods.

Yet another set that produces the correct stiffness is

$$\begin{aligned}\mathbf{B}_1 &= \frac{1}{6A} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{21} & 0 & 2y_{21} & 0 & 2y_{12} & 0 & 2y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{12} & 0 & 2x_{12} & 0 & 2x_{21} & 0 & 2x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{12} & y_{21} & 2x_{12} & 2y_{21} & 2x_{21} & 2y_{12} & 2x_{21} & 2y_{12} \end{bmatrix} \\ \mathbf{B}_2 &= \frac{1}{6A} \begin{bmatrix} y_{32} & 0 & y_{31} & 0 & y_{12} & 0 & 2y_{23} & 0 & 2y_{32} & 0 & 2y_{23} & 0 \\ 0 & x_{23} & 0 & x_{13} & 0 & x_{21} & 0 & 2x_{32} & 0 & 2x_{23} & 0 & 2x_{32} \\ x_{23} & y_{32} & x_{13} & y_{31} & x_{21} & y_{12} & 2x_{32} & 2y_{23} & 2x_{23} & 2y_{32} & 2x_{32} & 2y_{23} \end{bmatrix} \\ \mathbf{B}_3 &= \frac{1}{6A} \begin{bmatrix} y_{23} & 0 & y_{13} & 0 & y_{12} & 0 & 2y_{31} & 0 & 2y_{31} & 0 & 2y_{13} & 0 \\ 0 & x_{32} & 0 & x_{31} & 0 & x_{21} & 0 & 2x_{13} & 0 & 2x_{13} & 0 & 2x_{31} \\ x_{32} & y_{23} & x_{31} & y_{13} & x_{21} & y_{12} & 2x_{13} & 2y_{31} & 2x_{13} & 2y_{31} & 2x_{31} & 2y_{13} \end{bmatrix}\end{aligned}\quad (\text{E24.7})$$

EXERCISE 24.5 [A/C:40] (Research paper level) Characterize the most general form of the \mathbf{B}_i ($i = 1, 2, 3$) matrices that produce the same stiffness matrix \mathbf{K}^e in (E24.4). (Entails solving an algebraic Riccati equation.)

EXERCISE 24.6 [A/C:25]. Derive the 4-node transition triangle by forming the 6-node stiffness and applying MFCs to eliminate 5 and 6. Prove that this technique only works if sides 1–3 and 2–3 are straight with nodes 5 and 6 initially at the midpoint of those sides.

25

The Assembly Process

TABLE OF CONTENTS

	Page
§25.1. Introduction	25-3
§25.2. Simplifications	25-3
§25.3. Simplified Assemblers	25-4
§25.3.1. A Plane Truss Example Structure	25-4
§25.3.2. Implementation	25-6
§25.4. MET Assemblers	25-7
§25.4.1. A Plane Stress Assembler	25-7
§25.4.2. Implementation	25-9
§25.5. MET-VFC Assemblers	25-10
§25.5.1. Node Freedom Arrangement	25-10
§25.5.2. Node Freedom Signature	25-11
§25.5.3. The Node Freedom Allocation Table	25-11
§25.5.4. The Node Freedom Map Table	25-12
§25.5.5. The Element Freedom Signature	25-12
§25.5.6. The Element Freedom Table	25-13
§25.5.7. A Plane Trussed Frame Structure	25-14
§25.5.8. Implementation	25-17
§25.6. *Handling MultiFreedom Constraints	25-19
§25. Notes and Bibliography	25-20
§25. Exercises	25-21

§25.1. Introduction

Chapters 20, 23 and 24 dealt with element level operations. Sandwiched between element processing and solution there is the *assembly* process that constructs the master stiffness equations. Assembler examples for special models were given as recipes in the complete programs of Chapters 21 and 22. In the present chapter assembly will be studied with more generality.

The position of the assembler in a DSM-based code for static analysis is sketched in Figure 25.1. In most codes the assembler is implemented as a *element library loop driver*. This means that instead of forming all elements first and then assembling, the assembler constructs one element at a time in a loop, and immediately merges it into the master equations. This is the *merge loop*.

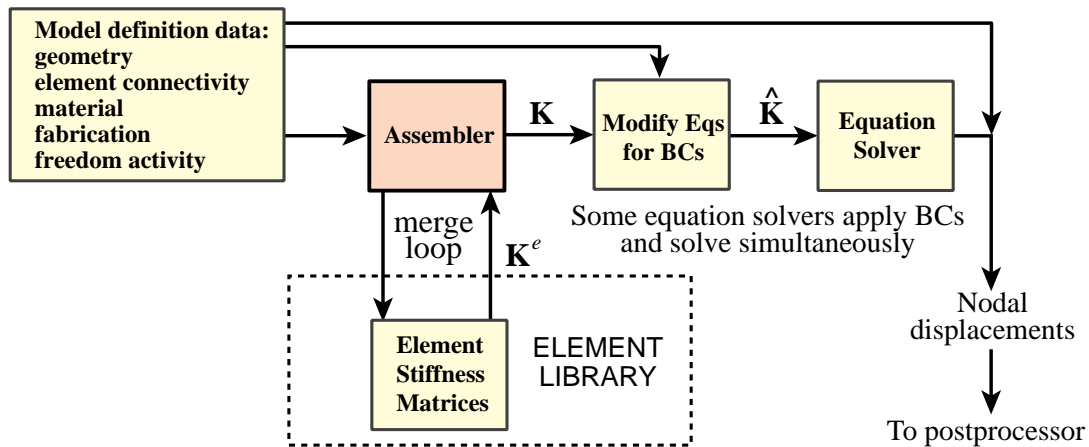


FIGURE 25.1. Role of assembler in FEM program.

Assembly is the most complicated stage of a production finite element program in terms of data flow. The complexity is not due to mathematical operations, which merely involve matrix addition, but interfacing with a large element library. Forming an element requires access to geometry, connectivity, material, and fabrication data. (In nonlinear analysis, to the state as well.) Merge requires access to freedom activity data, as well as knowledge of the sparse matrix format in which \mathbf{K} is stored. As illustrated in Figure 25.1, the assembler sits at the crossroads of this data exchange.

The coverage of the assembly process for a general FEM implementation is beyond the scope of an introductory course. Instead this Chapter takes advantage of assumptions that lead to coding simplifications. This allows the basic aspects to be covered without excessive delay.

§25.2. Simplifications

The assembly process is considerably simplified if the FEM implementation has these properties:

- All elements are of the same type. For example: all elements are 2-node plane bars.
- The number and configuration of degrees of freedom at each node is the same.
- There are no “gaps” in the node numbering sequence.
- There are no multifreedom constraints treated by master-slave or Lagrange multiplier methods.
- The master stiffness matrix is stored as a full symmetric matrix.

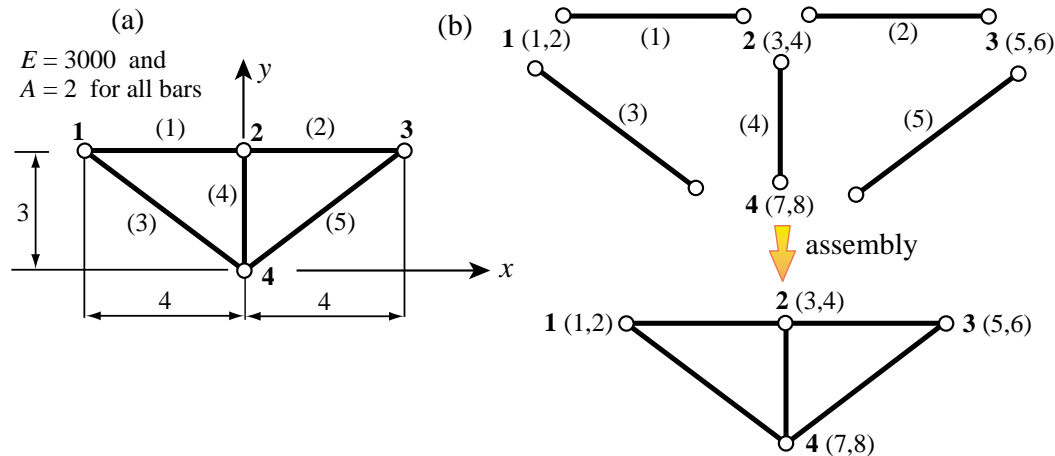


FIGURE 25.2. Left: Example plane truss structure. (a): model definition; (b) disconnection \rightarrow assembly process. Numbers in parentheses written after node numbers are global DOF numbers.

If the first four conditions are met the implementation is simpler because the element freedom table described below can be constructed “on the fly” from the element node numbers. The last condition simplifies the indexing to access entries of \mathbf{K} .

§25.3. Simplified Assemblers

In this section all simplifying assumptions listed in §25.2 are assumed to hold. This allows us to focus on *local-to-global DOF mapping*. The process is illustrated on a plane truss structure.

§25.3.1. A Plane Truss Example Structure

The plane truss shown in Figure 25.2(a) will be used to illustrate the details of the assembly process. The structure has 5 elements, 4 nodes and 8 degrees of freedom. The disconnection-to-assembly process is pictured in Figure 25.2(b).

Begin by clearing all entries of the 8×8 master stiffness matrix \mathbf{K} to zero, so that we effectively start with the null matrix:

$$\mathbf{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \end{matrix} \quad (25.1)$$

The numbers written after each row of \mathbf{K} are the *global DOF numbers*.

Element (1) joins nodes 1 and 2. The global DOFs of those nodes are $2 \times 1 - 1 = 1$, $2 \times 1 = 1$, $2 \times 2 - 1 = 3$ and $2 \times 2 = 4$. See Figure 25.2(b). Those four numbers are collected into an array called the *element freedom table*, or EFT for short. The element stiffness matrix $\mathbf{K}^{(1)}$ is listed on

the left below with the EFT entries annotated after the matrix rows. On the right is \mathbf{K} upon merging the element stiffness:

$$\begin{bmatrix} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \quad \begin{bmatrix} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 3 \\ 4 \end{matrix} \quad (25.2)$$

Element (2) joins nodes 2 and 3. The global DOFs of those nodes are $2 \times 2 - 1 = 3$, $2 \times 2 = 4$, $2 \times 3 - 1 = 5$ and $2 \times 3 = 6$; thus the EFT is $\{3, 4, 5, 6\}$. Matrices $\mathbf{K}^{(2)}$ and \mathbf{K} upon merge are

$$\begin{bmatrix} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 5 \\ 6 \end{matrix} \quad \begin{bmatrix} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 5 \\ 6 \\ 5 \\ 6 \end{matrix} \quad (25.3)$$

Element (3) joins nodes 1 and 4. Its EFT is $\{1, 2, 7, 8\}$. Matrices $\mathbf{K}^{(3)}$ and \mathbf{K} upon merge are

$$\begin{bmatrix} 768 & -576 & -768 & 576 \\ -576 & 432 & 576 & -432 \\ -768 & 576 & 768 & -576 \\ 576 & -432 & -576 & 432 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 7 \\ 8 \end{matrix} \quad \begin{bmatrix} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & 0 & 0 & 0 & -576 & 432 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 7 \\ 8 \\ 7 \\ 8 \end{matrix} \quad (25.4)$$

Element (4) joins nodes 2 and 4. Its EFT is $\{3, 4, 7, 8\}$. Matrices $\mathbf{K}^{(4)}$ and \mathbf{K} upon merge are

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 2000 & 0 & -2000 \\ 0 & 0 & 0 & 0 \\ 0 & -2000 & 0 & 2000 \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 7 \\ 8 \end{matrix} \quad \begin{bmatrix} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & -2000 & 0 & 0 & -576 & 2432 \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 3 \\ 4 \\ 7 \\ 8 \\ 7 \\ 8 \end{matrix} \quad (25.5)$$

Finally, element (5) joins nodes 3 and 4. Its EFT is { 5,6,7,8 }. Matrices $\mathbf{K}^{(5)}$ and \mathbf{K} upon merge are

$$\begin{bmatrix} 768 & 576 & -768 & -576 \\ 576 & 432 & -576 & -432 \\ -768 & -576 & 768 & 576 \\ -576 & -432 & 576 & 432 \end{bmatrix} \begin{matrix} 5 \\ 6 \\ 7 \\ 8 \end{matrix} \quad \begin{bmatrix} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 2268 & 576 & -768 & -576 \\ 0 & 0 & 0 & 0 & 576 & 432 & -576 & -432 \\ -768 & 576 & 0 & 0 & -768 & -576 & 1536 & 0 \\ 576 & -432 & 0 & -2000 & -576 & -432 & 0 & 2864 \end{bmatrix} \begin{matrix} 5 \\ 6 \\ 7 \\ 8 \end{matrix} \quad (25.6)$$

Since all elements have been processed, (25.6) is the master stiffness before application of boundary conditions.

```
PlaneTrussMasterStiffness[nodxyz_,elenod_,elemat_,elefab_,
  eleopt_]:=Module[{numele=Length[elenod],numnod=Length[nodxyz],
  e,ni,nj,eft,i,j,ii,jj,ncoor,Em,A,options,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];
  For[e=1,e<=numele,e++,{ni,nj}=elenod[[e]];
    eft={2*ni-1,2*ni,2*nj-1,2*nj};
    ncoor={nodxyz[[ni]],nodxyz[[nj]]};
    Em=elemat[[e]]; A=elefab[[e]]; options=eleopt;
    Ke=PlaneBar2Stiffness[ncoor,Em,A,options];
    For[i=1,i<=4,i++,ii=eft[[i]];
      For[j=i,j<=4,j++,jj=eft[[j]];
        K[[jj,ii]]=K[[ii,jj]]+Ke[[i,j]]];
    ];
  ]; Return[K]
];
```

FIGURE 25.3. Plane truss assembler module.

```
nodxyz={{-4,3},{0,3},{4,3},{0,0}};
elenod={{1,2},{2,3},{1,4},{2,4},{3,4}};
elemat=Table[3000,{5}]; elefab=Table[2,{5}]; eleopt={True};
K=PlaneTrussMasterStiffness[nodxyz,elenod,elemat,elefab,eleopt];
Print["Master Stiffness of Plane Truss of Fig 25.2:"];
K=Chop[K]; Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];
```

Master Stiffness of Plane Truss of Fig 25.2:

$$\begin{pmatrix} 2268. & -576. & -1500. & 0 & 0 & 0 & -768. & 576. \\ -576. & 432. & 0 & 0 & 0 & 0 & 576. & -432. \\ -1500. & 0 & 3000. & 0 & -1500. & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000. & 0 & 0 & 0 & -2000. \\ 0 & 0 & -1500. & 0 & 2268. & 576. & -768. & -576. \\ 0 & 0 & 0 & 0 & 576. & 432. & -576. & -432. \\ -768. & 576. & 0 & 0 & -768. & -576. & 1536. & 0 \\ 576. & -432. & 0 & -2000. & -576. & -432. & 0 & 2864. \end{pmatrix}$$

Eigs of K={5007.22, 4743.46, 2356.84, 2228.78, 463.703, 0, 0, 0}

FIGURE 25.4. Plane truss assembler tester and output results.

§25.3.2. Implementation

Figure 25.3 shows a *Mathematica* implementation of the assembly process just described. This assembler calls the element stiffness module `PlaneBar2Stiffness` of Figure 20.2. The assembler is invoked by

$$K = \text{SpaceTrussMasterStiffness}[\text{nodxyz}, \text{elenod}, \text{elemat}, \text{elefab}, \text{prcopt}] \quad (25.7)$$

The five arguments: `nodxyz`, `elenod`, `elemat`, `elefab`, and `prcopt` have the same function as those described in §21.1.3 for the `SpaceTrussMasterStiffness` assembler. In fact, comparing the code in Figure 25.3 to that of Figure 21.1, the similarities are obvious.

Running the script listed in the top of Figure 25.4 produces the output shown in the bottom of that figure. The master stiffness (25.6) is reproduced. The eigenvalue analysis verifies that the assembled stiffness has the correct rank of $8 - 3 = 5$.

§25.4. MET Assemblers

The next complication occurs when elements of different types are to be assembled, while the number and configuration of degrees of freedom at each node remains the same. This scenario is common in special-purpose programs for analysis of specific problems: plane stress, plate bending, solids. This will be called a *multiple-element-type assembler*, or MET assembler for short.

The only incremental change with respect to the simplest assemblers is that an array of element types, denoted here by `elotyp`, has to be provided.

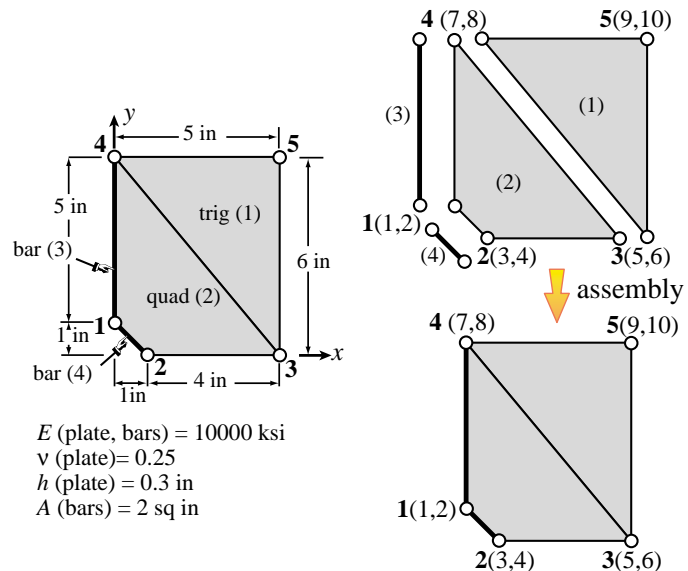


FIGURE 25.5. A plane stress structure used to test a MET assembler.

The assembler calls the appropriate element stiffness module according to type, and builds the local to global DOF mapping accordingly.

§25.4.1. A Plane Stress Assembler

This kind of assembler will be illustrated using the module `PlaneStressMasterStiffness`. As its name indicates, this is suitable for use in plane stress analysis programs. We make the following assumptions:

- 1 All nodes in the FEM mesh have 2 DOFs, namely the $\{u_x, u_y\}$ displacements. There are no node gaps or MFCs. The master stiffness is stored as a full matrix.
- 2 Element types can be: 2-node bars, 3-node linear triangles, or 4-node bilinear quadrilaterals. These elements can be freely intermixed. Element types are identified with character strings "Bar2", "Trig3" and "Quad4", respectively.

As noted above, the chief modification over the simplest assembler is that a different stiffness module is invoked as per type. Returned stiffness matrices are generally of different order; in our case 4×4 , 6×6 and 8×8 for the bar, triangle and quadrilateral, respectively. But the construction of the EFT is immediate, since node n maps to global freedoms $2n - 1$ and $2n$. The technique is illustrated with the 4-element, 5-node plane stress structure shown in Figure 25.5. It consists of one triangle, one quadrilateral, and two bars. The assembly process goes as follows.

Element (1) is a 3-node triangle with nodes 3, 5 and 4, whence the EFT is $\{5, 6, 9, 10, 7, 8\}$. The element stiffness is

$$\mathbf{K}^{(1)} = \begin{bmatrix} 500.0 & 0 & -500.0 & -600.0 & 0 & 600.0 \\ 0 & 1333.3 & -400.0 & -1333.3 & 400.0 & 0 \\ -500.0 & -400.0 & 2420.0 & 1000.0 & -1920.0 & -600.0 \\ -600.0 & -1333.3 & 1000.0 & 2053.3 & -400.0 & -720.0 \\ 0 & 400.0 & -1920.0 & -400.0 & 1920.0 & 0 \\ 600.0 & 0 & -600.0 & -720.0 & 0 & 720.0 \end{bmatrix} \begin{matrix} 5 \\ 6 \\ 9 \\ 10 \\ 7 \\ 8 \end{matrix} \quad (25.8)$$

Element (2) is a 4-node quad with nodes 2, 3, 4 and 1, whence the EFT is $\{3, 4, 5, 6, 7, 8, 1, 2\}$. The element stiffness computed with a 2×2 Gauss rule is

$$\mathbf{K}^{(2)} = \begin{bmatrix} 2076.4 & 395.55 & -735.28 & -679.11 & -331.78 & -136.71 & -1009.3 & 420.27 \\ 395.55 & 2703.1 & -479.11 & -660.61 & -136.71 & -1191.5 & 220.27 & -850.95 \\ -735.28 & -479.11 & 2067.1 & 215.82 & 386.36 & 427.34 & -1718.1 & -164.05 \\ -679.11 & -660.61 & 215.82 & 852.12 & 627.34 & 358.30 & -164.05 & -549.81 \\ -331.78 & -136.71 & 386.36 & 627.34 & 610.92 & 178.13 & -665.50 & -668.76 \\ -136.71 & -1191.5 & 427.34 & 358.30 & 178.13 & 1433.4 & -468.76 & -600.15 \\ -1009.3 & 220.27 & -1718.1 & -164.05 & -665.50 & -468.76 & 3393.0 & 412.54 \\ 420.27 & -850.95 & -164.05 & -549.81 & -668.76 & -600.15 & 412.54 & 2000.9 \end{bmatrix} \begin{matrix} 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 1 \\ 2 \end{matrix} \quad (25.9)$$

Element (3) is a 2-node bar with nodes 1 and 4, whence the EFT is $\{1, 2, 7, 8\}$. The element stiffness is

$$\mathbf{K}^{(3)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4000.0 & 0 & -4000.0 \\ 0 & 0 & 0 & 0 \\ 0 & -4000.0 & 0 & 4000.0 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 7 \\ 8 \end{matrix} \quad (25.10)$$

Element (4) is a 2-node bar with nodes 1 and 2, whence the EFT is $\{1, 2, 3, 4\}$. The element stiffness is

$$\mathbf{K}^{(4)} = \begin{bmatrix} 7071.1 & -7071.1 & -7071.1 & 7071.1 \\ -7071.1 & 7071.1 & 7071.1 & -7071.1 \\ -7071.1 & 7071.1 & 7071.1 & -7071.1 \\ 7071.1 & -7071.1 & -7071.1 & 7071.1 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \end{matrix} \quad (25.11)$$

Upon assembly, the master stiffness of the plane stress structure, printed with one digit after the


```

PlaneStressMasterStiffness[nodxyz_,eletyp_,elenod_,
  elemat_,elefab_,prcopt_]:=Module[{numele=Length[elenod],
  numnod=Length[nodxyz],ncoor,type,e,enl,neldof,
  i,n,ii,jj,eftab,Emat,th,numer,Ke,K},
K=Table[0,{2*numnod},{2*numnod}]; numer=prcopt[[1]];
For [e=1,e<=numele,e++, type=eletyp[[e]];
  If [!MemberQ[{"Bar2","Trig3","Quad4"},type], Print["Illegal type",
    " of element e=",e," Assembly interrupted"]; Return[K]];
  enl=elenod[[e]]; n=Length[enl];
  eftab=Flatten[Table[{2*enl[[i]]-1,2*enl[[i]]},{i,1,n}]];
  ncoor=Table[nodxyz[[enl[[i]]]],{i,n}];
  If [type=="Bar2", Em=elemat[[e]]; A=elefab[[e]];
    Ke=PlaneBar2Stiffness[ncoor,Em,A,{numer}] ];
  If [type=="Trig3", Emat=elemat[[e]]; th=elefab[[e]];
    Ke=Trig3IsoPMembraneStiffness[ncoor,Emat,th,{numer}] ];
  If [type=="Quad4", Emat=elemat[[e]]; th=elefab[[e]];
    Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,{numer,2}] ];
  neldof=Length[Ke];
  For [i=1,i<=neldof,i++, ii=eftab[[i]];
    For [j=i,j<=neldof,j++, jj=eftab[[j]];
      K[[jj,ii]]=K[[ii,jj]]+Ke[[i,j]] ];
  ];
]; Return[K];
];

```

FIGURE 25.6. Implementation of MET assembler for plane stress.

decimal point, is

$$\begin{bmatrix}
 10464.0 & -6658.5 & -8080.4 & 7291.3 & -1718.1 & -164.1 & -665.5 & -468.8 & 0 & 0 \\
 -6658.5 & 13072.0 & 7491.3 & -7922.0 & -164.1 & -549.8 & -668.8 & -4600.2 & 0 & 0 \\
 -8080.4 & 7491.3 & 9147.5 & -6675.5 & -735.3 & -679.1 & -331.8 & -136.7 & 0 & 0 \\
 7291.3 & -7922.0 & -6675.5 & 9774.1 & -479.1 & -660.6 & -136.7 & -1191.5 & 0 & 0 \\
 -1718.1 & -164.1 & -735.3 & -479.1 & 2567.1 & 215.8 & 386.4 & 1027.3 & -500.0 & -600.0 \\
 -164.1 & -549.8 & -679.1 & -660.6 & 215.8 & 2185.5 & 1027.3 & 358.3 & -400.0 & -1333.3 \\
 -665.5 & -668.8 & -331.8 & -136.7 & 386.4 & 1027.3 & 2530.9 & 178.1 & -1920.0 & -400.0 \\
 -468.8 & -4600.2 & -136.7 & -1191.5 & 1027.3 & 358.3 & 178.1 & 6153.4 & -600.0 & -720.0 \\
 0 & 0 & 0 & 0 & -500.0 & -400.0 & -1920.0 & -600.0 & 2420.0 & 1000.0 \\
 0 & 0 & 0 & 0 & -600.0 & -1333.3 & -400.0 & -720.0 & 1000.0 & 2053.3
 \end{bmatrix} \quad (25.12)$$

The eigenvalues of \mathbf{K} are

$$[32883. \quad 10517.7 \quad 5439.33 \quad 3914.7 \quad 3228.99 \quad 2253.06 \quad 2131.03 \quad 0 \quad 0 \quad 0] \quad (25.13)$$

which displays the correct rank.

§25.4.2. Implementation

An implementation of the MET plane stress assembler as a *Mathematica* module called `PlaneStressMasterStiffness` is shown in Figure 25.6. The assembler is invoked by

$$K = \text{PlaneStressMasterStiffness}[\text{nodxyz}, \text{eletyp}, \text{elenod}, \text{elemat}, \text{elefab}, \text{prcopt}] \quad (25.14)$$

The only additional argument is `eletyp`, which is a list of element types.

The script listed in the top of Figure 25.7 runs module `PlaneStressMasterStiffness` with the inputs appropriate to the problem defined in Figure 25.5. The output shown in the bottom of the figure reproduces the master stiffness matrix (25.12) and the eigenvalues (25.13). .

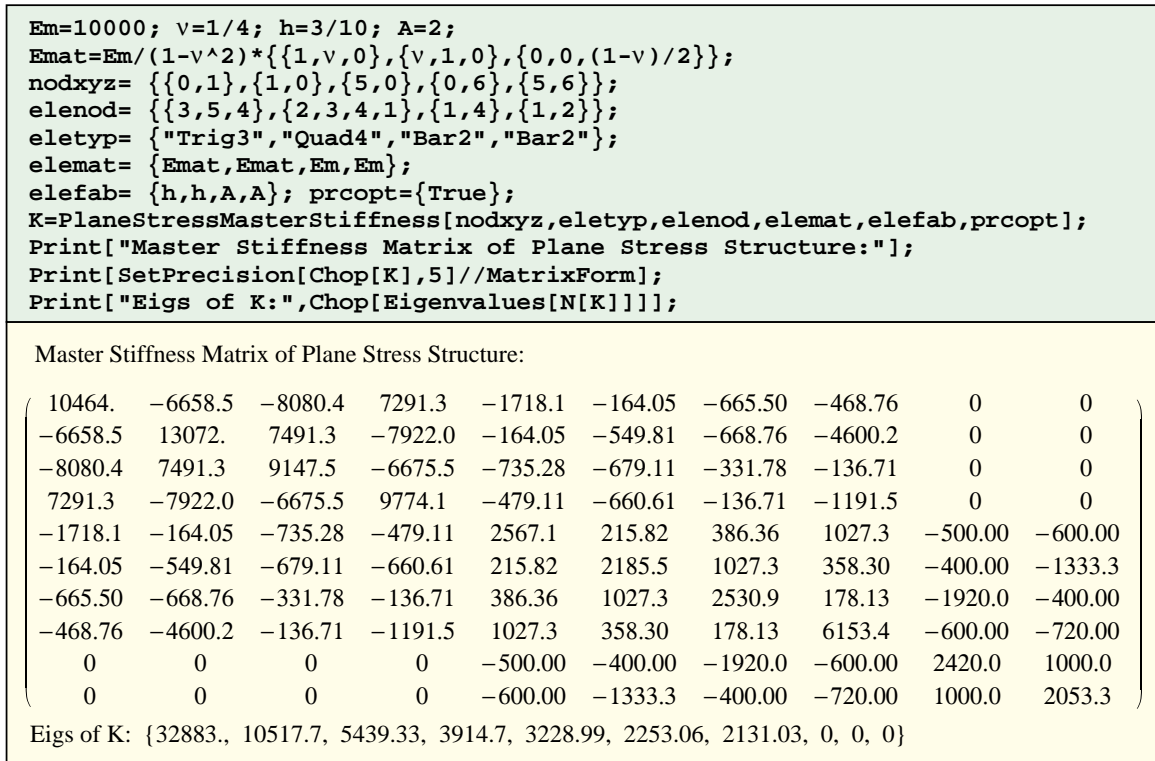


FIGURE 25.7. Script for assembling the plane stress model of Figure 25.5 and output results.

§25.5. MET-VFC Assemblers

The next complication beyond multiple element types is a major one: allowing nodes to have different freedom configurations. An assembler that handles this feature will be called a *MET-VFC assembler*, where VFC is an acronym for Variable Freedom Configuration.

A MET-VFC assembler gets closer to what is actually implemented in production FEM programs. A assembler of this kind can handle orientation nodes as well as node “gaps” automatically. Only two advanced attributes remain: allowing treatment of MFCs by Lagrange multipliers, and handling a sparse matrix storage scheme. The former can be done by an extension of the concept of element. The latter requires major programming modifications, however, and is not considered in this Chapter.

The implementation of a MET-VFC assembler requires the definition of additional data structures pertaining to freedoms, at both node and element levels. These are explained in the following subsections before giving a concrete application example.

§25.5.1. Node Freedom Arrangement

A key decision made by the implementor of a FEM program for structural analysis is: what freedoms will be allocated to nodes, and how are they arranged? While many answers are possible, we focus here on the most common arrangement for a linear three-dimensional FEM program.¹ At each node n three displacements $\{u_{xn}, u_{yn}, u_{zn}\}$ and three infinitesimal rotations $\{\theta_{xn}, \theta_{yn}, \theta_{zn}\}$ are chosen as freedoms and ordered as follows:

$$u_{xn}, u_{yn}, u_{zn}, \theta_{xn}, \theta_{yn}, \theta_{zn} \quad (25.15)$$

This is a *Node Freedom Arrangement* or NFA. Once the decision is made on a NFA, the *position of a freedom never changes*.² Thus if (25.15) is adopted, position #2 is forever associated to u_{yn} .

The length of the Node Freedom Arrangement is called the NFA length and often denoted (as a variable) by `nfaLen`. For the common choice (25.15) of 3D structural codes, `nfaLen` is 6.

§25.5.2. Node Freedom Signature

Having picked a NFA such as (25.15) does not mean that all such freedoms need to be present at a node. Take for example a model, such as a space truss, that only has translational degrees of freedom. Then only $\{u_{xn}, u_{yn}, u_{zn}\}$ will be used, and it becomes unnecessary to carry rotations in the master equations. There are also cases where allocations may vary from node to node.

To handle such scenarios in general terms a *Node Freedom Signature* or NFS, is introduced. The NFS is a sequence of zero-one integers. If the j^{th} entry is one, it means that the j^{th} freedom in the NFA is allocated; whereas if zero that freedom is not in use. For example, suppose that the underlying NFA is (25.15) and that node 5 uses the three freedoms $\{u_{x5}, u_{y5}, \theta_{z5}\}$. Then its NFS is

$$\{1, 1, 0, 0, 0, 1\} \quad (25.16)$$

If a node has no allocated freedoms (e.g, an orientation node), or has never been defined, its NFS contains only zeros.

It is convenient to decimally pack signatures into integers to save storage because allocating a full integer to hold just 0 or 1 is wasteful. The packed NFS (25.16) would be 110001. This technique requires utilities for packing and unpacking. *Mathematica* functions that provide those services are listed in Figure 25.8. All of them use built-in functions. A brief description follows.

<code>p = PackNodeFreedomSignature[s]</code>	Packs the NFS list <code>s</code> into integer <code>p</code> .
<code>s = UnpackNodeFreedomSignature[p,m]</code>	Unpacks the NFS <code>p</code> into a list <code>s</code> of length <code>m</code> , where <code>m</code> is the NFA length. (The second argument is necessary in case the NFS has leading zeros.)
<code>k = NodeFreedomCount[p]</code>	Returns count of ones in <code>p</code> .

¹ This is the scheme used by most commercial codes. In a geometrically nonlinear FEM code, however, finite rotation measures must be used. Consequently the infinitesimal node rotations $\{\theta_x, \theta_y, \theta_z\}$ must be replaced by something else.

² NFA positions #7 and beyond are available for additional freedom types, such as Lagrange multipliers, temperatures, pressures, fluxes, etc. It is also possible to reserve more NFA positions for structural freedoms.

```
PackNodeFreedomSignature[s_]:=FromDigits[s,10];
UnpackNodeFreedomSignature[p_,m_]:=IntegerDigits[p,10,m];
NodeFreedomCount[p_]:=DigitCount[p,10,1];
```

FIGURE 25.8. Node freedom manipulation utility functions.

Example 25.1. `PackNodeFreedomSignature[{1,1,0,0,0,1}]` returns 110001. The inverse is `UnpackNodeFreedomSignature[110001,6]`, which returns `{1,1,0,0,0,1}`. `NodeFreedomCount[110001]` returns 3.

§25.5.3. The Node Freedom Allocation Table

The Node Freedom Allocation Table, or NFAT, is a node by node list of packed node freedom signatures. In *Mathematica* the list containing this table is internally identified as `nodfat`. The configuration of the NFAT for the plane stress example structure of Figure 25.5 is

$$\text{nodfat} = \{110000, 110000, 110000, 110000, 110000\} = \text{Table}[110000, \{5\}]. \quad (25.17)$$

This says that only two freedoms: $\{u_x, u_y\}$, are used at each of the five nodes.

A zero entry in the NFAT flags a node with no allocated freedoms. This is either an orientation node, or an undefined one. The latter case happens when there is a node numbering gap, which is a common occurrence when certain mesh generators are used. In the case of a MET-VFC assembler such gaps are not considered errors.

§25.5.4. The Node Freedom Map Table

The Node Freedom Map Table, or NFMT, is an array with one entry per node. Suppose that node n has $k \geq 0$ allocated freedoms. These have global equation numbers $i + j$, $j = 1, \dots, k$. This i is called the *base equation index*: it represent the global equation number *before* the first equation to which node n contributes. Obviously $i = 0$ if $n = 1$. Base equation indices for all nodes are recorded in the NFMT.³

In *Mathematica* code the table is internally identified as `nodfmt`. Figure 25.9 lists a module that constructs the NFMT given the NFAT. It is invoked as

$$\text{nodfmt} = \text{NodeFreedomMapTable}[\text{nodfat}] \quad (25.18)$$

The only argument is the NFAT. The module builds the NFMT by simple incrementation, and returns it as function value.

Example 25.2. The NFAT for the plane truss-frame structure of Figure 25.11 is `nodfat = {110001, 110000, 110001, 000000, 110001}`. The node freedom counts are `{3, 2, 3, 0, 3}`. The call

$$\text{nodfmt} = \text{NodeFreedomMapTable}[\text{nodfat}] \quad (25.19)$$

returns `{0, 3, 5, 8, 8}` in `nodfmt`. This can be easily verified visually: $0+3=3$, $0+3+2=5$, etc.

Figure 25.9 also lists a function `TotalFreedomCount` that returns the total number of freedoms in the master equations, given the NFAT as argument.

³ The “map” qualifier comes from the use of this array in computing element-to-global equation mapping.

```

NodeFreedomMapTable[nodfat_]:=Module[{numnod=Length[nodfat],
i,nodfmt}, nodfmt=Table[0,{numnod}];
For [i=1,i<=numnod-1,i++, nodfmt[[i+1]]=nodfmt[[i]]+
DigitCount[nodfat[[i]],10,1] ];
Return[nodfmt];

TotalFreedomCount[nodfat_]:=Sum[DigitCount[nodfat[[i]],10,1],
{i,Length[nodfat]}];

```

FIGURE 25.9. Modules to construct the NFMT from the NFAT, and to compute the total number of freedoms.

```

ElementFreedomTable[enl_,efs_,nodfat_,nodfmt_,m_]:=Module[
{nelnod=Length[enl],eft,i,j,k=0,ix,n,s,es,sx},
eft=Table[0,{Sum[DigitCount[efs[[i]],10,1],{i,1,nelnod}]}];
For [i=1,i<=nelnod,i++, n=enl[[i]];
s=IntegerDigits[nodfat[[n]],10,m]; ix=0;
sx=Table[0,{m}]; Do [If[s[[j]]>0,sx[[j]]=++ix],{j,1,m}];
es=IntegerDigits[efs[[i]],10,m];
For [j=1,j<=m,j++, If [es[[j]]>0, k++;
If [s[[j]]>0, eft[[k]]=nodfmt[[n]]+sx[[j]] ]];
];
Return[eft];

```

FIGURE 25.10. Module to construct the Element Freedom Table for a MET-VFC assembler.

§25.5.5. The Element Freedom Signature

The Element Freedom Signature or EFS is a data structure that shares similarities with the NFAT, but provides freedom data at the element rather than global level. More specifically, given an element, it tells to which degrees of freedom it contributes.

The EFS is best described through an example. Suppose that in a general FEM program all defined structural nodes have the signature 111111, meaning the arrangement (25.15). The program includes 2-node space bar elements, which contribute only to translational degrees of freedom $\{u_x, u_y, u_z\}$. The EFS of any such element will be

$$\text{efs} = \{111000, 111000\} \quad (25.20)$$

For a 2-node space beam element, which contributes to all six nodal freedoms, the EFS will be

$$\text{efs} = \{111111, 111111\} \quad (25.21)$$

For a 3-node space beam element in which the third node is an orientation node:

$$\text{efs} = \{111111, 111111, 000000\} \quad (25.22)$$

This information, along with the NFAT and NFMT, is used to build Element Freedom Tables.

§25.5.6. The Element Freedom Table

The Element Freedom Table or EFT has been encountered in all previous assembler examples. It is a one dimensional list of length equal to the number of element DOF. If the i^{th} element DOF maps to the k^{th} global DOF, then the i^{th} entry of EFT contains k . This allows to write the merge loop compactly. In the simpler assemblers discussed in previous sections, the EFT can be built “on the fly” simply from element node numbers. For a MET-VFC assembler that is no longer the case. To take care of the VFC feature, the construction of the EFT is best done by a separate module. A *Mathematica* implementation is shown in Figure 25.10. Discussion of the logic, which is not trivial, is relegated to an Exercise, and we simply describe here the interface. The module is invoked as

$$\text{eft} = \text{ElementFreedomTable}[\text{enl}, \text{efs}, \text{nodfat}, \text{nodfmt}, \text{m}] \quad (25.23)$$

The arguments are

<code>enl</code>	The element node list.
<code>efs</code>	The Element Freedom Signature (EFS) list.
<code>nodfat</code>	The Node Freedom Allocation Table (NFAT) described in 25.5.3
<code>nodfmt</code>	The Node Freedom Map Table (NFMT) described in 25.5.4
<code>m</code>	The NFA length, often 6.

The module returns the Element Freedom Table as function value. Examples of use of this module are provided in the plane trussed frame example below.

Remark 25.1. The EFT constructed by `ElementFreedomTable` is guaranteed to contain only positive entries if every freedom declared in the EFS matches a globally allocated freedom in the NFAT. But it is possible for the returned EFT to contain zero entries. This can only happen if an element freedom does not match a globally allocated one. For example suppose that a 2-node space beam element with the EFS (25.21) is placed into a program that does not accept rotational freedoms and thus has a NFS of 111000 at all structural nodes. Then six of the EFT entries, namely those pertaining to the rotational freedoms, would be zero.

The occurrence of a zero entry in a EFT normally flags a logic or input data error. If detected, the assembler should print an appropriate error message and abort. The module of Figure 25.10 does not make that decision because it lacks certain information, such as the element number, that should be placed into the message.

§25.5.7. A Plane Trussed Frame Structure

To illustrate the workings of a MET-VFC assembler we will follow the assembly process of the structure pictured in Figure 25.11(a). The finite element discretization, element disconnection and assembly process are illustrated in Figure 25.11(b,c,d). Although the structure is chosen to be plane to facilitate visualization, it will be considered within the context of a general purpose 3D program with the freedom arrangement (25.15) at each node.

The structure is a trussed frame. It uses two element types: plane (Bernoulli-Euler) beam-columns and plane bars. Geometric, material and fabrication data are given in Figure 25.11(a). The FEM idealization has 4 nodes and 5 elements. Nodes numbers are 1, 2, 3 and 5 as shown in Figure 25.11(b). Node 4 is purposely left out to illustrate handling of numbering gaps. There are 11 DOFs, three

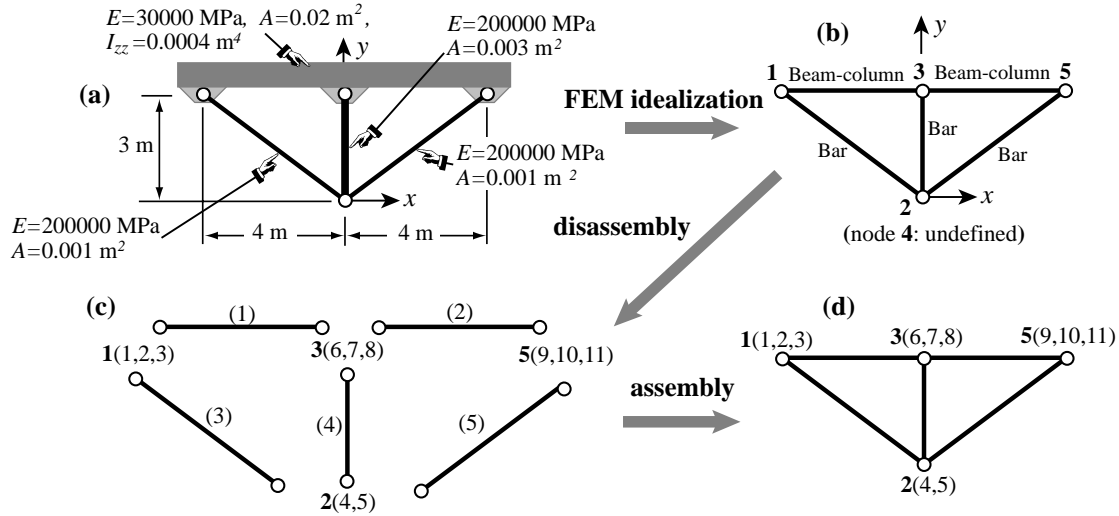


FIGURE 25.11. Trussed frame structure to illustrate a MET-VFC assembler: (a) original structure showing dimensions, material and fabrication properties; (b) finite element idealization with bars and beam column elements; (c) conceptual disassembly; (d) assembly. Numbers written in parentheses after a node number in (c,d) are the global freedom (equation) numbers allocated to that node.

at nodes 1, 3 and 5, and two at node 2. They are ordered as

Global DOF #:	1	2	3	4	5	6	7	8	9	10	11
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	u_{x3}	u_{y3}	θ_{z3}	u_{x5}	u_{y5}	θ_{z5}
Node #:	1	1	1	2	2	3	3	3	5	5	5

(25.24)

The NFAT and NFMT can be constructed by inspection of (25.24) to be

$$\begin{aligned} \text{nodfat} &= \{ 110001, 110000, 110001, 000000, 110001 \} \\ \text{nodfmt} &= \{ 0, 3, 5, 8, 8 \} \end{aligned} \quad (25.25)$$

The element freedom data structures can be also constructed by inspection:

Elem	Type	Nodes	EFS	EFT
(1)	Beam-column	{ 1, 3 }	{ 110001, 110001 }	{ 1, 2, 3, 6, 7, 8 }
(2)	Beam-column	{ 3, 5 }	{ 110001, 110001 }	{ 6, 7, 8, 9, 10, 11 }
(3)	Bar	{ 1, 2 }	{ 110000, 110000 }	{ 1, 2, 4, 5 }
(4)	Bar	{ 2, 3 }	{ 110000, 110000 }	{ 4, 5, 6, 7 }
(5)	Bar	{ 2, 5 }	{ 110000, 110000 }	{ 4, 5, 9, 10 }

(25.26)

Next we list the element stiffness matrices, computed with the modules discussed in Chapter 20. The EFT entries are annotated as usual.

Element (1): This is a plane beam-column element with stiffness

$$\mathbf{K}^{(1)} = \begin{bmatrix} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 6 \\ 7 \\ 8 \end{matrix} \quad (25.27)$$

Element (2): This is a plane beam-column with element stiffness identical to the previous one

$$\mathbf{K}^{(2)} = \begin{bmatrix} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{bmatrix} \begin{matrix} 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{matrix} \quad (25.28)$$

Element (3): This is a plane bar element with stiffness

$$\mathbf{K}^{(3)} = \begin{bmatrix} 25.6 & -19.2 & -25.6 & 19.2 \\ -19.2 & 14.4 & 19.2 & -14.4 \\ -25.6 & 19.2 & 25.6 & -19.2 \\ 19.2 & -14.4 & -19.2 & 14.4 \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 4 \\ 5 \end{matrix} \quad (25.29)$$

Element (4): This is a plane bar element with stiffness

$$\mathbf{K}^{(4)} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 200. & 0 & -200. \\ 0 & 0 & 0 & 0 \\ 0 & -200. & 0 & 200. \end{bmatrix} \begin{matrix} 4 \\ 5 \\ 6 \\ 7 \end{matrix} \quad (25.30)$$

Element (5): This is a plane bar element with stiffness

$$\mathbf{K}^{(5)} = \begin{bmatrix} 25.6 & 19.2 & -25.6 & -19.2 \\ 19.2 & 14.4 & -19.2 & -14.4 \\ -25.6 & -19.2 & 25.6 & 19.2 \\ -19.2 & -14.4 & 19.2 & 14.4 \end{bmatrix} \begin{matrix} 4 \\ 5 \\ 9 \\ 10 \end{matrix} \quad (25.31)$$

Upon merging the 5 elements the master stiffness matrix becomes

$$\mathbf{K} = \begin{bmatrix} 175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & 0 & 0 & 0 & 0 & 0 \\ -19.2 & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & 0 & 0 & 0 \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 \\ -25.6 & 19.2 & 0 & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 \\ 19.2 & -14.4 & 0 & 0 & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & 0 \\ -150. & 0 & 0 & 0 & 0 & 300. & 0 & 0 & -150. & 0 & 0 \\ 0 & -22.5 & -45. & 0 & -200. & 0 & 245. & 0 & 0 & -22.5 & 45. \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. \\ 0 & 0 & 0 & -25.6 & -19.2 & -150. & 0 & 0 & 175.6 & 19.2 & 0 \\ 0 & 0 & 0 & -19.2 & -14.4 & 0 & -22.5 & -45. & 19.2 & 36.9 & -45. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. \end{bmatrix} \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \end{matrix} \quad (25.32)$$

in which global freedom numbers are annotated for convenience. Since all elements have been processed, (25.32) is the master stiffness matrix. Its eigenvalues are

$$[460.456 \ 445.431 \ 306.321 \ 188.661 \ 146.761 \ 82.7415 \ 74.181 \ 25.4476 \ 0 \ 0 \ 0] \quad (25.33)$$

This has the correct rank of $11 - 3 = 8$.

Remark 25.2. For storage as a skyline matrix (Chapter 26) the template for (25.32) would look like

$$\mathbf{K} = \begin{bmatrix} 175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & & & & & \\ & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & & & \\ & & 120. & 0 & 0 & 0 & -45. & 60. & & & \\ & & & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & \\ & & & & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & \\ & & & & & 300. & 0 & 0 & -150. & 0 & \\ & & & & & & 245. & 0 & 0 & -22.5 & 45. \\ & & & & & & & 240. & 0 & -45. & 60. \\ & & & & & & & & 175.6 & 19.2 & 0 \\ & & & & & & & & & 36.9 & -45. \\ & & & & & & & & & & 120. \end{bmatrix} \quad (25.34)$$

symm

The diagonal location pointers of (25.34) are defined by the list

$$\text{DLT} = \{0, 1, 3, 6, 10, 15, 21, 27, 34, 40, 47, 52\} \quad (25.35)$$

Examination of the template (25.34) reveals that some additional zero entries could be removed from the template; for example K_{13} . The fact that those entries are zero is, however, fortuitous. It comes from the fact that some elements such as the beams are aligned along x , which decouples axial and bending stiffnesses.

§25.5.8. Implementation

Figure 25.12 lists the *Mathematica* implementation of a MET-VFC assembler capable of doing the trussed frame structure of Figure 25.11. The assembler module is invoked as

$$\mathbf{K} = \text{PlaneTrussedFrameMasterStiffness}[\text{nodxyz}, \text{eletyp}, \text{elenod}, \\ \text{elemat}, \text{elefab}, \text{nodfat}, \text{prcopt}] \quad (25.36)$$

The only new argument with respect to the MET assembler of §25.4.2 is

nodfat The Node Freedom Arrangement Table (NFAT).

The module returns the master stiffness matrix as function value.

`PlaneTrussedFrameMasterStiffness` uses five other modules: the element stiffness modules `PlaneBar2Stiffness` and `PlaneBeamColumn2Stiffness` of Chapter 20, the NFMT constructor `NodeFreedomMapTable` of Figure 25.9, the total-DOF-counter `TotalFreedomCount` of Figure 25.9, and the EFT constructor `ElementFreedomTable` of Figure 25.10.

The element fabrication list `elefab` has minor modifications. A plane beam-column element requires two properties: $\{A, I_{zz}\}$, which are the cross section area and the moment of inertia about the local z axis. A bar element requires only the cross section area: A . For the example trussed frame structure `elefab` is $\{\{0.02, 0.004\}, \{0.02, 0.004\}, 0.001, 0.003, 0.001\}$ in accordance with

```

PlaneTrussedFrameMasterStiffness[nodxyz_,eletyp_,
  elenod_,elemat_,elefab_,nodfat_,prcopt_]:=Module[
{numele=Length[elenod],numnod=Length[nodxyz],numdof,nodfmt,e,enl,
eftab,n,ni,nj,i,j,k,m,ncoor,Em,A,Izz,options,Ke,K},
nodfmt=NodeFreedomMapTable[nodfat];
numdof=TotalFreedomCount[nodfat]; K=Table[0,{numdof},{numdof}];
For [e=1, e<=numele, e++, enl=elenod[[e]];
  If [eletyp[[e]]=="Bar2", {ni,nj}=enl;
    ncoor={nodxyz[[ni]],nodxyz[[nj]]};
    Em=elemat[[e]]; A=elefab[[e]]; options=prcopt;
    eftab=ElementFreedomTable[enl,{110000,110000},nodfat,nodfmt,6];
    Ke=PlaneBar2Stiffness[ncoor,Em,A,options] ];
  If [eletyp[[e]]=="BeamCol2", {ni,nj}=enl;
    ncoor={nodxyz[[ni]],nodxyz[[nj]]};
    eftab=ElementFreedomTable[enl,{110001,110001},nodfat,nodfmt,6];
    Em=elemat[[e]]; {A,Izz}=elefab[[e]]; options=prcopt;
    Ke=PlaneBeamColumn2Stiffness[ncoor,Em,{A,Izz},options] ];
  If [MemberQ[eftab,0], Print["Zero entry in eftab for element ",e];
    Print["Assembly process aborted"]; Return[K]];
  neldof=Length[eftab];
  For [i=1, i<=neldof, i++, ii=eftab[[i]];
    For [j=i, j<=neldof, j++, jj=eftab[[j]];
      K[[jj,ii]]=K[[ii,jj]]+Ke[[i,j]] ];
    ];
]; Return[K];

```

FIGURE 25.12. A MET-VFC assembler written for the plane trussed frame structure.

```

nodxyz={{-4,3},{0,0},{0,3},0,{4,3}};
eletyp= {"BeamCol2","BeamCol2","Bar2","Bar2","Bar2"};
elenod= {{1,3},{3,5},{1,2},{2,3},{2,5}};
elemat= Join[Table[30000,{2}],Table[200000,{3}]];
elefab= {{0.02,0.004},{0.02,0.004},0.001,0.003,0.001};
prcopt= {True};
nodfat={110001,110000,110001,000000,110001};
K=PlaneTrussedFrameMasterStiffness[nodxyz,eletyp,
  elenod,elemat,elefab,nodfat,prcopt]; K=Chop[K];
Print["Master Stiffness of Example Trussed Frame:"];
Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];

```

Master Stiffness of Example Trussed Frame:

175.6	-19.6	0	-25.6	19.2	-150.	0	0	0	0	0
-19.2	36.9	45.	19.2	-14.4	0	-22.5	45.	0	0	0
0	45.	120.	0	0	0	-45.	60.	0	0	0
-25.6	19.2	0	51.2	0	0	0	0	-25.6	-19.2	0
19.2	-14.4	0	0	228.8	0	-200.	0	-19.2	-14.4	0
-150.	0	0	0	0	300.	0	0	-150.	0	0
0	-22.5	-45.	0	-200.	0	245.	0	0	-22.5	45.
0	45.	60.	0	0	0	0	240.	0	-45.	60.
0	0	0	-25.6	-19.2	-150.	0	0	175.6	19.2	0
0	0	0	19.2	-14.4	0	-22.5	-45.	19.2	36.9	-45.
0	0	0	0	0	0	45.	60.	0	-45.	120.

Eigs of K= {460.456, 445.431, 306.321, 188.661, 146.761, 82.7415, 74.181, 25.4476, 0, 0, 0}

FIGURE 25.13. Script to test the assembler of Figure 25.12 with the structure of Figure 25.11.

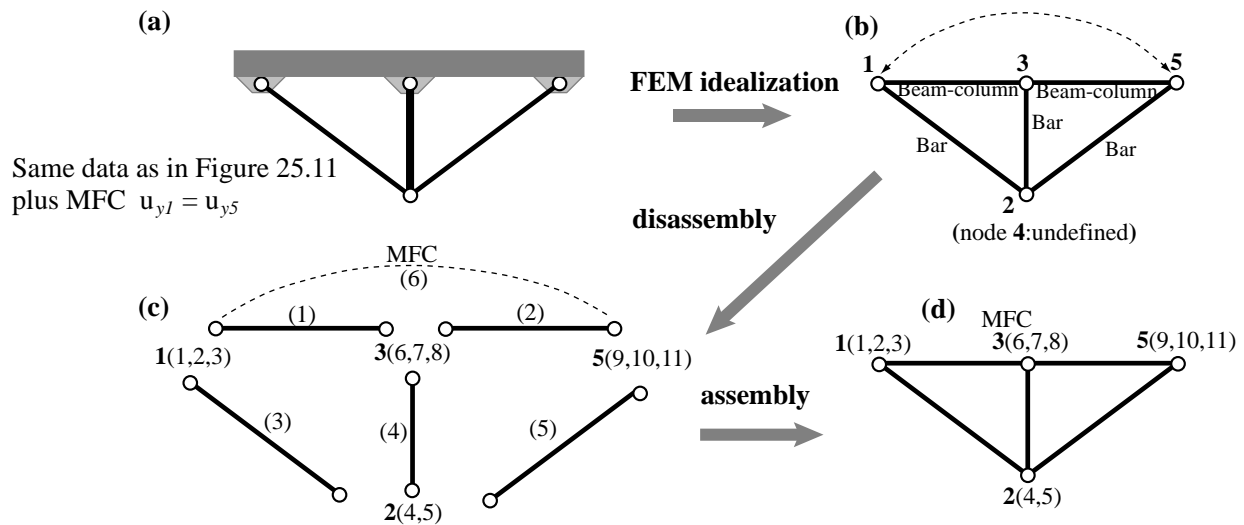


FIGURE 25.14. Finite element discretization, disassembly and assembly of trussed frame example structure with MFC $u_{y1} = u_{y5}$.

the data in Figure 25.11. The element material property list `elemat` is modified on the account that the elastic moduli for the beam columns (made of reinforced concrete) and bars made of (steel) are different.

Running the script listed in the top of Figure 25.13 produces the output shown in the bottom of that figure. As can be observed this agrees with (25.32) and (25.33).

§25.6. *Handling MultiFreedom Constraints

To see the effect of imposing an MFC through the Lagrange multiplier method on the configuration of the master stiffness matrix, suppose that the trussed frame example structure of the previous section is subjected to the constraint that nodes 1 and 5 must move vertically by the same amount. That is,

$$u_{y1} = u_{y5} \quad \text{or} \quad u_{y1} - u_{y5} = 0. \quad (25.37)$$

For assembly purposes (25.37) may be viewed as a fictitious element⁴ labeled as (6). See Figure 25.14.

The degrees of freedom of the assembled structure increase by one to 12. They are ordered as

Global DOF #:	1	2	3	4	5	6	7	8	9	10	11	12
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	u_{x3}	u_{y3}	θ_{z3}	u_{x5}	u_{y5}	θ_{z5}	$\lambda^{(6)}$
Node #:	1	1	1	2	2	3	3	3	5	5	5	none

(25.38)

The assembly of the first five elements proceeds as explained before, and produces the same master stiffness as (25.34), except for an extra zero row and column. Processing the MFC element (6) yields the 12×12

⁴ This device should not be confused with penalty elements, which have stiffness matrices. The effect of adding a “Lagrange multiplier element” is to append rows and columns to the master stiffness.

bordered stiffness

$$\mathbf{K} = \begin{bmatrix} 175.6 & -19.2 & 0 & -25.6 & 19.2 & -150. & 0 & 0 & 0 & 0 & 0 & 0 \\ -19.2 & 36.9 & 45. & 19.2 & -14.4 & 0 & -22.5 & 45. & 0 & 0 & 0 & 1. \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 & 0 \\ -25.6 & 19.2 & 0 & 51.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 & 0 \\ 19.2 & -14.4 & 0 & 0 & 228.8 & 0 & -200. & 0 & -19.2 & -14.4 & 0 & 0 \\ -150. & 0 & 0 & 0 & 0 & 300. & 0 & 0 & -150. & 0 & 0 & 0 \\ 0 & -22.5 & -45. & 0 & -200. & 0 & 245. & 0 & 0 & -22.5 & 45. & 0 \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. & 0 \\ 0 & 0 & 0 & -25.6 & -19.2 & -150. & 0 & 0 & 175.6 & 19.2 & 0 & 0 \\ 0 & 0 & 0 & -19.2 & -14.4 & 0 & -22.5 & -45. & 19.2 & 36.9 & -45. & -1. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. & 0 \\ 0 & 1. & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1. & 0 \end{bmatrix} \quad (25.39)$$

in which the coefficients 1 and -1 associated with the MFC (25.37) end up in the last row and column of \mathbf{K} .

There are several ways to incorporate multiplier adjunction into an automatic assembly procedure. The most elegant ones associate the fictitious element with a special freedom signature and a reserved position in the NFA. Being of advanced nature such schemes are beyond the scope of this Chapter.

Notes and Bibliography

The DSM assembly process for the simplest case of §25.3 is explained in any finite element text. Few texts cover, however, the complications that arise in more general scenarios. Especially when VFCs are allowed.

Assembler implementation flavors have fluctuated since the DSM became widely accepted as standard. Variants have been strongly influenced by limits on random access memory (RAM). Those limitations forced the use of “out-of-core” blocked equation solvers, meaning that heavy use was made of disk auxiliary storage to store and retrieve the master stiffness matrix in blocks. Only a limited number of blocks could reside on RAM. Thus a straightforward element-by-element assembly loop (as in the assemblers presented here) was likely to “miss the target” on the receiving end of the merge, forcing blocks to be read in, modified and saved. On large problems this swapping was likely to “trash” the system with heavy I/O, bringing processing to a crawl.

One solution favored in programs of the 1965–85 period was to process all elements without assembling, saving matrices on disk. The assembler then cycled over stiffness blocks and read in the contributing elements. With smart asynchronous buffering and tuned-up direct access I/O such schemes were able to achieved reasonable efficiency. However, system dependent logic quickly becomes a maintenance nightmare.

A second way out emerged by 1970: the frontal solvers referenced in Chapter 11. A frontal solver carries out assembly, BC application and solution concurrently. Element contributions are processed in a special order that traverses the FEM mesh as a “wavefront.” Application of displacement BCs and factorization can “trail the wave” once no more elements are detected as contributing to a given equation. As can be expected of trying to do too much at once, frontal solvers can be extremely sensitive to changes. One tiny alteration in the element library over which the solver sits, and the whole thing may crumble like a house of cards.

The availability of large amounts of RAM (even on PCs and laptops) since the mid 90s, has had a happy consequence: interweaved assembly and solver implementations, as well as convoluted matrix blocking, are no longer needed. The assembler can be modularly separated from the solver. As a result even the most complex assembler presented here fits in one page of text. Of course it is too late for the large scale FEM codes that got caught in the limited-RAM survival game decades ago. Changing their assemblers and solvers incrementally is virtually impossible. The gurus that wrote those thousands of lines of spaghetti Fortran are long gone. The only practical way out is rewrite the whole shebang from scratch. In a commercial environment such investment-busting decisions are unlikely.

Homework Exercises for Chapter 25

The Assembly Process

EXERCISE 25.1 [D:10] Suppose you want to add a six-node plane stress quadratic triangle to the MET assembler of §25.4. Sketch how you would modify the module of Figure 25.6 for this to happen.

EXERCISE 25.2 [C:10] Exercise the module `ElementFreedomTable` listed in Figure 25.10 on the trussed frame structure. Verify that it produces the last column of (25.26).

EXERCISE 25.3 [D:15] Describe the logic of module `ElementFreedomTable` listed in Figure 25.10. Can it return zero entries in the EFT? If yes, give a specific example of how this can happen. Hint: read the Chapter carefully.

EXERCISE 25.4 [A/C:30] The trussed frame structure of Figure 25.4 is reinforced with two triangular steel plates attached as shown in Figure EE25.1(a). The plate thickness is $1.6 \text{ mm} = 0.0016 \text{ m}$; the material is isotropic with $E = 240000 \text{ MPa}$ and $\nu = 1/3$. Each reinforcing plate is modeled with a single plane stress 3-node linear triangle. The triangles are numbered (6) and (7), as illustrated in Figure E25.1(c). Compute the master stiffness matrix \mathbf{K} of the structure.⁵

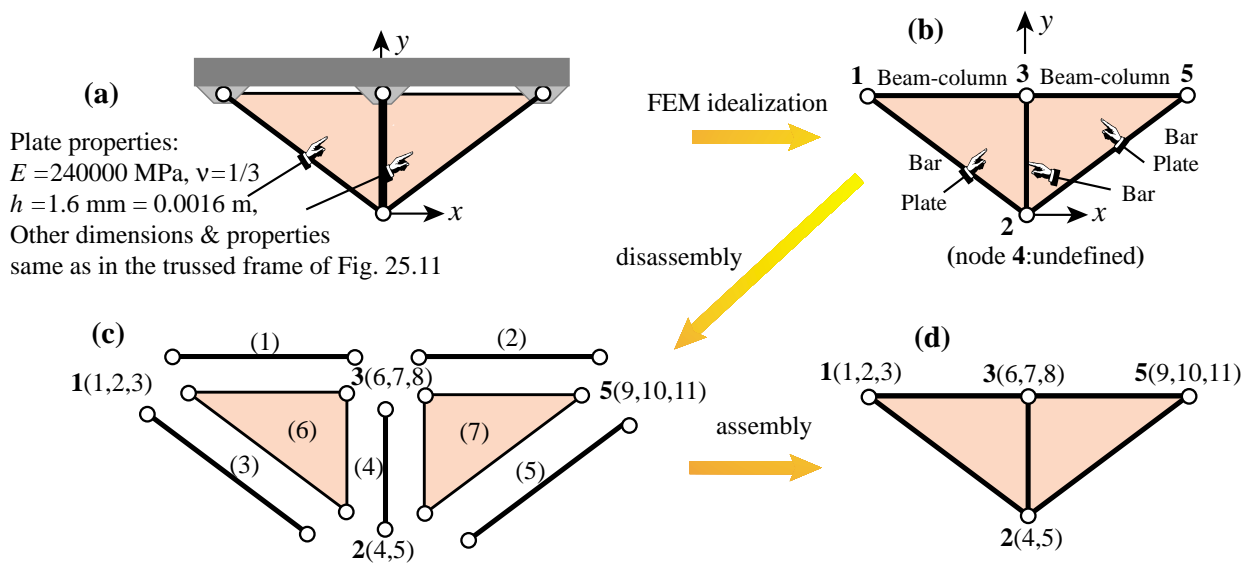


FIGURE E25.1. Plate-reinforced trussed frame structure for Exercise 25.4.

This exercise may be done through *Mathematica*. For this download Notebook `ExampleAssemblers.nb` from this Chapter index, and complete Cell 4 by writing the assembler.⁶ The plate element identifier is "Trig3". The driver script to run the assembler is also provided in Cell 4 (blue text).

⁵ This structure would violate the compatibility requirements stated in Chapter 19 unless the beams are allowed to deflect laterally independently of the plates. This is the fabrication actually sketched in Figure E25.1(a).

⁶ Cells 1–3 contain the assemblers presented in §25.3, §25.4 and §25.5, respectively, which may be used as guides. The stiffness modules for the three element types used in this structure are available in Cells 2 and 3 and may be reused for this Exercise.

When reusing the assembler of Cell 3 as a guide, please do not remove the internal Print commands that show element information (red text). Those come in handy for debugging. Please keep that printout in the returned homework to help the grader.

Another debugging hint: check that the master stiffness (25.32) is obtained if the plate thickness, called `hplate` in the driver script, is temporarily set to zero.

Target:

$$\mathbf{K} = \begin{bmatrix} 337.6 & -19.2 & 0 & -25.6 & 91.2 & -312. & -72. & 0 & 0 & 0 & 0 \\ -19.2 & 90.9 & 45. & 91.2 & -14.4 & -72. & -76.5 & 45. & 0 & 0 & 0 \\ 0 & 45. & 120. & 0 & 0 & 0 & -45. & 60. & 0 & 0 & 0 \\ -25.6 & 91.2 & 0 & 243.2 & 0 & -192. & 0 & 0 & -25.6 & -91.2 & 0 \\ 91.2 & -14.4 & 0 & 0 & 804.8 & 0 & -776. & 0 & -91.2 & -14.4 & 0 \\ -312. & -72. & 0 & -192. & 0 & 816. & 0 & 0 & -312. & 72. & 0 \\ -72. & -76.5 & -45. & 0 & -776. & 0 & 929. & 0 & 72. & -76.5 & 45. \\ 0 & 45. & 60. & 0 & 0 & 0 & 0 & 240. & 0 & -45. & 60. \\ 0 & 0 & 0 & -25.6 & -91.2 & -312. & 72. & 0 & 337.6 & 19.2 & 0 \\ 0 & 0 & 0 & -91.2 & -14.4 & 72. & -76.5 & -45. & 19.2 & 90.9 & -45. \\ 0 & 0 & 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. \end{bmatrix} \quad (\text{E25.1})$$

EXERCISE 25.5 [A:30] §25.5 does not explain how to construct the NFAT from the input data. (In the trussed frame example script, `nodfat` was set up by inspection, which is OK only for small problems.) Explain how this table could be constructed automatically if the EFS of each element in the model is known. Note: the logic is far from trivial.

26

Solving FEM Equations

TABLE OF CONTENTS

	Page
§26.1. Motivation for Sparse Solvers	26–3
§26.1.1. The Curse of Fullness	26–3
§26.1.2. The Advantages of Sparsity	26–4
§26.2. Sparse Solution of Stiffness Equations	26–5
§26.2.1. Skyline Storage Format	26–5
§26.2.2. Factorization	26–6
§26.2.3. Solution	26–6
§26.2.4. Treating MFCs with Lagrange Multipliers	26–6
§26.3. A SkySolver Implementation	26–7
§26.3.1. Skymatrix Representation	26–7
§26.3.2. *Skymatrix Factorization	26–8
§26.3.3. *Solving for One or Multiple RHS	26–11
§26.3.4. *Matrix-Vector Multiply	26–13
§26.3.5. *Printing and Mapping	26–14
§26.3.6. *Reconstruction of SkyMatrix from Factors	26–15
§26.3.7. *Miscellaneous Utilities	26–16
§26. Exercises	26–20

§26.1. Motivation for Sparse Solvers

In the Direct Stiffness Method (DSM) of finite element analysis, the element stiffness matrices and consistent nodal force vectors are immediately assembled to form the *master stiffness matrix* and *master force vector*, respectively, by the process called *merge*. The assembly process is described in Chapter 25. For simplicity the description that follows assumes that no MultiFreedom Constraints (MFCs) are present. The end result of the assembly process are the master stiffness equations

$$\mathbf{K}\mathbf{u} = \mathbf{f} \quad (26.1)$$

where \mathbf{K} is the master stiffness matrix, \mathbf{f} the vector of node forces and \mathbf{u} the vector of node displacements. Upon imposing the displacement boundary conditions, the system (26.1) is solved for the unknown node displacements. The solution concludes the main phase of DSM computations.

In practical applications the order of the stiffness system (26.1) can be quite large. Systems of order 1000 to 10000 are routinely solved in commercial software. Larger ones (say up to 100000 equations) are not uncommon and even millions of equations are being solved on supercomputers. Presently the record is about 50 million equations on parallel computers.¹

In *linear* FEM analysis the cost of solving this system of equations rapidly overwhelms other computational phases. Much attention has therefore been given to matrix processing techniques that economize storage and solution time by taking advantage of the special structure of the stiffness matrix.

The master force vector is stored as a conventional one-dimensional array of length equal to the number N of degrees of freedom. This storage arrangement presents no particular difficulties even for very large problem sizes.² Handling the master stiffness matrix, however, presents computational difficulties.

§26.1.1. The Curse of Fullness

If \mathbf{K} is stored and processed as if it were a *full* matrix, the storage and processing time resources rapidly become prohibitive as N increases. This is illustrated in Table 26.1, which summarizes the storage and factor-time requirements for orders $N = 10^4$, 10^5 and 10^6 .³

As regards memory needs, a full square matrix stored without taking advantage of symmetry, requires storage for N^2 entries. If each entry is an 8-byte, double precision floating-point number, the required storage is $8N^2$ bytes. Thus, a matrix of order $N = 10^4$ would require 8×10^8 bytes or 800 MegaBytes (MB) for storage.

For large N the solution of (26.1) is dominated by the factorization of \mathbf{K} , an operation discussed in §26.2. This operation requires approximately $N^3/6$ floating point operation units. [A floating-point operation unit is conventionally defined as a (multiply,add) pair plus associated indexing and data movement operations.] Now a fast workstation can typically do 10^7 of these operations per second,

¹ For fluid equations solved by fluid-volume methods the number is over 100 million.

² A force of displacement vector of, say, 1M equations uses 8MB for double precision storage. In these days of GB RAMs, that is a modest amount.

³ The factor times given in that table reflect 1998 computer technology. To update to 2003, divide times by 10 to 20.

Table 26.1 Storage & Solution Time for a Fully-Stored Stiffness Matrix

Matrix order N	Storage (double prec)	Factor op.units (FLOPS)	Factor time workstation (or fast PC)	Factor time supercomputer
10^4	800 MB	$10^{12}/6$	3 hrs	2 min
10^5	80 GB	$10^{15}/6$	4 mos	30 hrs
10^6	8 TB	$10^{18}/6$	300 yrs	3 yrs

Table 26.2 Storage & Solution Time for a Skyline Stored Stiffness Matrix
Assuming $B = \sqrt{N}$

Matrix order N	Storage (double prec)	Factor op.units (FLOPS)	Factor time workstation (or fast PC)	Factor time supercomputer
10^4	8 MB	$10^8/2$	5 sec	0.05 sec
10^5	240 MB	$10^{10}/2$	8 min	5 sec
10^6	8 GB	$10^{12}/2$	15 hrs	8 min

whereas a supercomputer may be able to sustain 10^9 or more. These times assume that the entire matrix is kept in RAM; for otherwise the elapsed time may increase by factors of 10 or more due to I/O transfer operations. The elapsed times estimated given in Table 26.1 illustrate that for present computer resources, orders above 10^4 would pose significant computational difficulties.

§26.1.2. The Advantages of Sparsity

Fortunately a very high percentage of the entries of the master stiffness matrix \mathbf{K} are zero. Such matrices are called *sparse*. There are clever programming techniques that take advantage of sparsity that fit certain patterns. Although a comprehensive coverage of such techniques is beyond the scope of this course, we shall concentrate on a particular form of sparse scheme that is widely used in FEM codes: *skyline* storage. This scheme is simple to understand, manage and implement, while cutting storage and processing times by orders of magnitude as the problems get larger.

The skyline storage format is a generalization of its widely used predecessor called the *band storage* scheme. A matrix stored in accordance with the skyline format will be called a *skymatrix* for short. Only symmetric skymatrices will be considered here, since the stiffness matrices in linear FEM are symmetric.

If a skymatrix of order N can be stored in S memory locations, the ratio $B = S/N$ is called the *mean bandwidth*. If the entries are, as usual, 8-byte double-precision floating-point numbers, the storage requirement is $8NB$ bytes. The factorization of a skymatrix requires approximately $\frac{1}{2}NB^2$ floating-point operation units. In two-dimensional problems B is of the order of \sqrt{N} . Under this assumption, storage requirements and estimated factorization times for $N = 10^4$, $N = 10^5$ and

$N = 10^6$ are reworked in Table 26.2. It is seen that by going from full to skyline storage significant reductions in computer resources have been achieved. For example, now $N = 10^4$ is easy on a workstation and trivial on a supercomputer. Even a million equations do not look far-fetched on a supercomputer as long as enough memory is available.⁴

In preparation for assembling \mathbf{K} as a skymatrix one has to set up several auxiliary arrays related to nodes and elements. These auxiliary arrays are described in the previous Chapter. Knowledge of that material is useful for understanding the following description.

§26.2. Sparse Solution of Stiffness Equations

§26.2.1. Skyline Storage Format

The skyline storage arrangement for \mathbf{K} is best illustrated through a simple example. Consider the 6×6 stiffness matrix

$$\mathbf{K} = \begin{bmatrix} K_{11} & 0 & K_{13} & 0 & 0 & K_{16} \\ & K_{22} & 0 & K_{24} & 0 & 0 \\ & & K_{33} & K_{34} & 0 & 0 \\ & & & K_{44} & 0 & K_{46} \\ & & & & K_{55} & K_{56} \\ \text{symm} & & & & & K_{66} \end{bmatrix} \quad (26.2)$$

Since the matrix is symmetric only one half, the upper triangle in the above display, need to be shown.

Next we define the *envelope* of \mathbf{K} as follows. From each diagonal entry move *up* the corresponding column until the last nonzero entry is found. The envelope separates that entry from the rest of the upper triangle. The remaining zero entries are conventionally removed:

$$\mathbf{K} = \begin{bmatrix} K_{11} & & K_{13} & & K_{16} \\ & K_{22} & 0 & K_{24} & 0 \\ & & K_{33} & K_{34} & 0 \\ & & & K_{44} & K_{46} \\ & & & & K_{55} & K_{56} \\ \text{symm} & & & & & K_{66} \end{bmatrix} \quad (26.3)$$

What is left constitute the *skyline profile* of *skyline template* of the matrix. A sparse matrix that can be profitably stored in this form is called a *skymatrix* for brevity. Notice that the skyline profile may include zero entries. During the factorization step discussed below these zero entries will in general become nonzero, a phenomenon that receives the name *fill-in*.

⁴ On a PC or laptop with a 32-bit processor, there is a RAM “hard barrier” because of hardware addressing limitations. That is typically 2GB. As PCs migrate to 64-bit processors over the next 10 years, the barrier disappears. For example the high-end Mac workstation can presently be expanded up to 16GB RAM. The new limits are primarily dictated by power consumption, circuitry and cooling considerations. A distributed (Beowulf) network is a practical solution to go beyond the individual machine limits.

The key observation is that only *the entries in the skyline template need to be stored*, because *fill-in in the factorization process will not occur outside the envelope*. To store these entries it is convenient to use a one-dimensional *skyline array*:

$$\mathbf{s} : [K_{11}, K_{22}, K_{13}, 0, K_{33}, K_{24}, K_{34}, K_{44}, K_{55}, K_{16}, 0, 0, K_{46}, K_{56}, K_{66}] \quad (26.4)$$

This array is complemented by a $(N + 1)$ integer array \mathbf{p} that contains addresses of *diagonal locations*. The array has $N + 1$ entries. The $(i + 1)^{th}$ entry of \mathbf{p} has the location of the i^{th} diagonal entry of \mathbf{K} in \mathbf{s} . For the example matrix:

$$\mathbf{p} : [0, 1, 2, 5, 8, 9, 15] \quad (26.5)$$

In the previous Chapter, this array was called the Global Skyline Diagonal Location Table, or GSDLT. Equations for which the displacement component is prescribed are identified by a *negative* diagonal location value. For example if u_3 and u_5 are prescribed displacement components in the test example, then

$$\mathbf{p} : [0, 1, 2, -5, 8, -9, 15] \quad (26.6)$$

Remark 26.1. In Fortran it is convenient to dimension the diagonal location array as $\mathbf{p}(0:n)$ so that indexing begins at zero. In C this is the standard indexing.

§26.2.2. Factorization

The stiffness equations (26.1) are solved by a direct method that involves two basic phases: *factorization* and *solution*.

In the first stage, the skyline-stored symmetric stiffness matrix is factored as

$$\mathbf{K} = \mathbf{LDU} = \mathbf{LDL}^T = \mathbf{U}^T \mathbf{D} \mathbf{U}, \quad (26.7)$$

where \mathbf{L} is a unit lower triangular matrix, \mathbf{D} is a nonsingular diagonal matrix, and \mathbf{U} and \mathbf{L} are the transpose of each other. The original matrix is overwritten by the entries of \mathbf{D}^{-1} and \mathbf{U} ; details may be followed in the program implementation. No pivoting is used in the factorization process. This factorization is carried out by *Mathematica* module `SymmSkyMatrixFactor`, which is described later in this Chapter.

§26.2.3. Solution

Once \mathbf{K} has been factored, the solution \mathbf{u} for a given right hand side \mathbf{f} is obtained by carrying out three stages:

$$\text{Forward reduction : } \mathbf{Lz} = \mathbf{f}, \quad (26.8)$$

$$\text{Diagonal scaling : } \mathbf{Dy} = \mathbf{z}, \quad (26.9)$$

$$\text{Back substitution : } \mathbf{Uu} = \mathbf{y}, \quad (26.10)$$

where \mathbf{y} and \mathbf{z} are intermediate vectors. These stages are carried out by *Mathematica* modules `SymmSkyMatrixVectorSolve`, which is described later.

§26.2.4. Treating MFCs with Lagrange Multipliers

In *Mathematica* implementations of FEM, MultiFreedom Constraints (MFCs) are treated with Lagrange multipliers. There is one multiplier for each constraint. The multipliers are placed at the end of the solution vector.

Specifically, let the $\text{nummul} > 0$ MFCs be represented in matrix form as $\mathbf{C}\mathbf{u} = \mathbf{g}$, where \mathbf{C} and \mathbf{g} are given, and let the nummul multipliers be collected in a vector $\boldsymbol{\lambda}$. The multiplier-augmented master stiffness equations are

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \quad (26.11)$$

or

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \quad (26.12)$$

where the symmetric matrix \mathbf{A} , called a stiffness-bordered matrix, is of order $\text{numdof} + \text{nummul}$.

The stiffness bordered matrix is also stored in skyline form, and the previous solution procedure applies, as long as the skyline array is properly constructed as described in the previous Chapter.

The main difference with respect to the no-MFC case is that, because of the configuration (26.11), \mathbf{A} can no longer be positive definite. In principle pivoting should be used during the factorization of \mathbf{A} to forestall possible numerical instabilities. Pivoting can have a detrimental effect on solution efficiency because entries can move outside of the skyline template. However, by placing the $\boldsymbol{\lambda}$ at the end such difficulties will not be encountered if \mathbf{K} is positive definite, and the constraints are linearly independent (that is, \mathbf{C} has full rank). Thus pivoting is not necessary.

§26.3. A SkySolver Implementation

The remaining sections of this revised Chapter describe a recent implementation of the skyline solver and related routines in *Mathematica*. This has been based on similar Fortran codes used since 1967.

§26.3.1. Skymatrix Representation

In what follows the computer representation in *Mathematica* of a symmetric skymatrix will be generally denoted by the symbol \mathbf{S} . Such a representation consists of a list of two numeric objects:

$$\mathbf{S} = \{ \mathbf{p}, \mathbf{s} \} \quad (26.13)$$

Here $\mathbf{p} = \text{GSDLT}$ is the Global Skyline Diagonal Location Table introduced in §11.6, and \mathbf{s} is the array of skymatrix entries, arranged as described in the previous section. This array usually consists of floating-point numbers, but it may also contain exact integers or fractions, or even symbolic entries.

For example, suppose that the numerical entries of the 6×6 skymatrix (26.10) are actually

$$\mathbf{K} = \begin{bmatrix} 11 & & 13 & & 16 \\ & 22 & 0 & 24 & 0 \\ & & 33 & 34 & 0 \\ & & & 44 & 46 \\ & & & & 55 & 56 \\ \text{symm} & & & & & 66 \end{bmatrix} \quad (26.14)$$

Cell 26.1 Factorization of a Symmetric SkyMatrix

```

SymmSkyMatrixFactor[S_,tol_]:= Module[
  {p,a, fail,i,j,k,l,m,n,ii,ij,jj,jk,jmj,d,s,row,v},
  row=SymmSkyMatrixRowLengths[S]; s=Max[row];
  {p,a}=S; n=Length[p]-1; v=Table[0,{n}]; fail=0;
  Do [jj=p[[j+1]]; If [jj<0|row[[j]]==0, Continue[]]; d=a[[jj]];
    jmj=Abs[p[[j]]]; jk=jj-jmj;
    Do [i=j-jk+k; v[[k]]=0; ii=p[[i+1]];
      If [ii<0, Continue[]]; m=Min[ii-Abs[p[[i]]],k]-1;
      ij=jmj+k; v[[k]]=a[[ij]];
      v[[k]]-=Take[a,{ii-m,ii-1}].Take[v,{k-m,k-1}];
      a[[ij]]=v[[k]]*a[[ii]],
      {k,1,jk-1}];
    d-=Take[a,{jmj+1,jmj+jk-1}].Take[v,{1,jk-1}];
    If [Abs[d]<tol*row[[j]], fail=j; a[[jj]]=Infinity; Break[] ];
    a[[jj]]=1/d,
  {j,1,n}];
  Return[{{p,a},fail}]
];

SymmSkyMatrixRowLengths[S_]:= Module[
  {p,a,i,j,n,ii,jj,m,d,row},
  {p,a}=S; n=Length[p]-1; row=Table[0,{n}];
  Do [ii=p[[i+1]]; If [ii<0, Continue[]]; m=ii-i; row[[i]]=a[[ii]]^2;
    Do [If [p[[j+1]]>0, d=a[[m+j]]^2; row[[i]]+=d; row[[j]]+=d,
      {j,Max[1,Abs[p[[i]]]-m+1],Min[n,i]-1}],
    {i,1,n}]; Return[Sqrt[row]];
];

```

Its *Mathematica* representation, using the symbols (26.13) is

$$\begin{aligned}
 p &= \{ 0,1,2,5,8,9,15 \}; \\
 s &= \{ 11,22,13,0,33,24,34,44,55,16,0,0,46,56,66 \}; \\
 S &= \{ p, s \};
 \end{aligned} \tag{26.15}$$

or more directly

$$S = \{ \{ 0,1,2,5,8,9,15 \}, \{ 11,22,13,0,33,24,34,44,55,16,0,0,46,56,66 \} \}; \tag{26.16}$$

[The remaining sections on details of skyline processing logic, marked with a *, will not be covered in class. They are intended for a more advanced course.]

Cell 26.2 Factorization Test Input

```

ClearAll[n]; n=5; SeedRandom[314159];
p=Table[0,{n+1}]; Do[p[[i+1]]=p[[i]]+
    Max[1,Min[i,Round[Random[]*i]]],{i,1,n}];
a=Table[1.,{i,1,p[[n+1]]}];
Print["Mean Band=",N[p[[n+1]]/n]];
S={p,a};
Sr=SymmSkyMatrixLDUReconstruct[S];
Print["Reconstructed SkyMatrix:"]; SymmSkyMatrixLowerTrianglePrint[Sr];
SymmSkyMatrixLowerTriangleMap[Sr];
Print["eigs=",Eigenvalues[SymmSkyMatrixConvertToFull[Sr]]];
x=Table[{N[i],3.,(-1)^i*N[n-i]},{i,1,n}];
Print["Assumed x=",x];
b=SymmSkyMatrixColBlockMultiply[Sr,x];
(*x=Transpose[x]; b=SymmSkyMatrixRowBlockMultiply[Sr,x];*)
Print["b=Ax=",b];
Print[Timing[{F,fail}=SymmSkyMatrixFactor[Sr,10.^(-12)]]];
If [fail!=0, Print["fail=",fail]; Abort[]];
Print["F=",F]; Print["fail=",fail];
Print["Factor:"];
SymmSkyMatrixLowerTrianglePrint[F];
x=SymmSkyMatrixColBlockSolve[F,b];
(*x=SymmSkyMatrixRowBlockSolve[F,b];*)
Print["Computed x=",x//InputForm];

```

§26.3.2. *Skymatrix Factorization

Module `SymmSkyMatrixFactor`, listed in Cell 26.1, factors a symmetric skymatrix into the product **LDU** where **L** is the transpose of **U**. No pivoting is used. The module is invoked as

$$\{ Sf, fail \} = \text{SymmSkyMatrixFactor}[S, tol]$$

The input arguments are

- | | |
|------------|---|
| S | The skymatrix to be factored, stored as the two-object list $\{ p, s \}$; see previous subsection. |
| tol | Tolerance for singularity test. The appropriate value of tol depends on the kind of skymatrix entries stored in s . |

If the skymatrix entries are floating-point numbers handled by default in double precision arithmetic, **tol** should be set to $8\times$ or $10\times$ the machine precision in that kind of arithmetic. The factorization aborts if, when processing the j -th row, $d_j \leq tol * r_j$, where d_j is the computed j^{th} diagonal entry of **D**, and r_j is the Euclidean norm of the j^{th} skymatrix row.

If the skymatrix entries are exact (integers, fractions or symbols), **tol** should be set to zero. In this case exact singularity is detectable, and the factorization aborts only on that condition.

The outputs are:

- | | |
|-----------|--|
| Sf | If fail is zero on exit, Sf is the computed factorization of S . It is a two-object list $\{ p, du$ |
|-----------|--|

Cell 26.3 Output from Program of Cells 26.1 and 26.2

```

Mean Band=1.6
Reconstructed SkyMatrix:

      Col 1    Col 2    Col 3    Col 4    Col 5
Row 1    1.0000
Row 2           1.0000
Row 3           1.0000    2.0000
Row 4                   1.0000
Row 5                   1.0000    3.0000

      1 2 3 4 5

1  +
2  +
3  + +
4  +
5  + + +
eigs={3.9563, 2.20906, 1., 0.661739, 0.172909}
Assumed x={{1., 3., -4.}, {2., 3., 3.}, {3., 3., -2.}, {4., 3., 1.},
           {5., 3., 0.}}
b=Ax={{1., 3., -4.}, {5., 6., 1.}, {13., 12., -1.}, {9., 6., 1.},
      {22., 15., -1.}}
{0.0666667 Second, {{0, 1, 2, 4, 5, 8},
                    {1., 1., 1., 1., 1., 1., 1., 1.}}, 0}}
F={{0, 1, 2, 4, 5, 8}, {1., 1., 1., 1., 1., 1., 1., 1.}}
fail=0
Factor:

      Col 1    Col 2    Col 3    Col 4    Col 5
Row 1    1.0000
Row 2           1.0000
Row 3           1.0000    1.0000
Row 4                   1.0000
Row 5                   1.0000    1.0000
Computed x={{1., 3., -4.}, {2., 3., 3.}, {3., 3., -2.}, {4., 3., 1.},
           {5., 3., 0.}}

```

}, where du stores the entries of \mathbf{D}^{-1} in the diagonal locations, and of \mathbf{U} in its strict upper triangle.

fail A singularity detection indicator. A zero value indicates that no singularity was detected. If **fail** returns $j > 0$, the factorization was aborted at the j -th row. In this case **Sf** returns the aborted factorization with ∞ stored in d_j .

Cell 26.4 Solving for a Single RHS

```

SymmSkyMatrixVectorSolve[S_,b_]:= Module[
  {p,a,n,i,j,k,m,ii,jj,bi,x},
  {p,a}=S; n=Length[p]-1; x=b;
  If [n!=Length[x], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixVectorSolve"]; Return[Null]];
  Do [ii=p[[i+1]];If [ii>=0, Continue[]]; ii=-ii; k=i-ii+Abs[p[[i]]]+1;
    bi=x[[i]]; If [bi==0, Continue[]];
    Do [jj=p[[j+1]], If [jj<0, Continue[]];
      m=j-i; If [m<0, x[[j]]-=a[[ii+m]]*bi; Break[]];
      ij=jj-m; If [ij>Abs[p[[j]]], x[[j]]-=a[[ij]]*bi],
      {j,k,n}],
  {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, x[[i]]=0; Continue[]];
    imi=Abs[p[[i]]]; m=ii-imi-1;
    x[[i]]-=Take[a,{imi+1,imi+m}].Take[x,{i-m,i-1}],
  {i,1,n}];
  Do [ii=Abs[p[[i+1]]]; x[[i]]*=a[[ii]], {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, x[[i]]=b[[i]]; Continue[]];
    m=ii-Abs[p[[i]]]-1;
    Do [ x[[i-j]]-=a[[ii-j]]*x[[i]], {j,1,m}],
  {i,n,1,-1}];
  Return[x]
];

```

A test of `SymmSkyMatrixFactor` on the matrix (26.14) is shown in Cells 26.2 and 26.3. The modules that print and produce skyline maps used in the test program are described later in this Chapter.

§26.3.3. *Solving for One or Multiple RHS

Module `SymmSkyMatrixVectorSolve`, listed in Cell 26.4, solves the linear system $\mathbf{Ax} = \mathbf{b}$ for \mathbf{x} , following the factorization of the symmetric skymatrix \mathbf{A} by `SymmSkyMatrixFactor`. The module is invoked as

$$\mathbf{x} = \text{SymmSkyMatrixVectorSolve}[\mathbf{Sf}, \mathbf{b}]$$

The input arguments are

- | | |
|-----------|--|
| Sf | The factored matrix returned by <code>SymmSkyMatrixFactor</code> . |
| b | The right-hand side vector to be solved for, stored as a single-level (one dimensional) list. If the i -th entry of \mathbf{x} is prescribed, the known value must be supplied in this vector. |

The outputs are:

- | | |
|----------|---|
| x | The computed solution vector, stored as a single-level (one-dimensional) list. Prescribed solution components return the value of the entry in \mathbf{b} . |
|----------|---|

Sometimes it is necessary to solve linear systems for multiple ($m > 1$) right hand sides. One way to do that is to call `SymmSkyMatrixVectorSolve` repeatedly. Alternatively, if m right hand sides are collected as columns of a rectangular matrix \mathbf{B} , module `SymmSkyMatrixColBlockSolve` may be invoked as

Cell 26.5 Solving for a Block of Righ Hand Sides

```

SymmSkyMatrixColBlockSolve[S_,b_]:= Module[
  {p,a,n,nrhs,i,j,k,m,r,ii,jj,bi,x},
  {p,a}=S; n=Length[p]-1; x=b;
  If [n!=Dimensions[x][[1]], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixBlockColSolve"]; Return[Null]]; nrhs = Dimensions[x][[2]];
  Do [ii=p[[i+1]];If [ii>=0, Continue[]]; ii=-ii; k=i-ii+Abs[p[[i]]]+1;
    Do [bi=x[[i,r]]; If [bi==0, Continue[]];
      Do [jj=p[[j+1]], If [jj<0, Continue[]];
        m=j-i; If [m<0,x[[j,r]]-=a[[ii+m]]*bi; Break[]];
        ij=jj-m; If [ij>Abs[p[[j]]], x[[j,r]]-=a[[ij]]*bi,
          {j,k,n}],
        {r,1,nrhs}],
    {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, Do[x[[i,r]]=0,{r,1,nrhs}];Continue[]];
    imi=Abs[p[[i]]]; m=ii-imi-1;
    Do [ Do [ x[[i,r]]-=a[[imi+j]]*x[[i-m+j-1,r]], {j,1,m}], {r,1,nrhs}],
    {i,1,n}];
  Do [ii=Abs[p[[i+1]]]; Do[x[[i,r]]*=a[[ii]], {r,1,nrhs}], {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, Do[x[[i,r]]=b[[i,r]],{r,1,nrhs}];Continue[]];
    m=ii-Abs[p[[i]]]-1;
    Do [ Do [ x[[i-j,r]]-=a[[ii-j]]*x[[i,r]], {j,1,m}], {r,1,nrhs}],
    {i,n,1,-1}];
  Return[x]
];

SymmSkyMatrixRowBlockSolve[S_,b_]:= Module[
  {p,a,n,nrhs,i,j,k,m,r,ii,jj,bi,x},
  {p,a}=S; n=Length[p]-1; x=b;
  If [n!=Dimensions[x][[2]], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixBlockRowSolve"]; Return[Null]]; nrhs = Dimensions[x][[1]];
  Do [ii=p[[i+1]];If [ii>=0, Continue[]]; ii=-ii; k=i-ii+Abs[p[[i]]]+1;
    Do [bi=x[[r,i]]; If [bi==0, Continue[]];
      Do [jj=p[[j+1]], If [jj<0, Continue[]];
        m=j-i; If [m<0,x[[j,r]]-=a[[ii+m]]*bi; Break[]];
        ij=jj-m; If [ij>Abs[p[[j]]], x[[r,j]]-=a[[ij]]*bi,
          {j,k,n}],
        {r,1,nrhs}],
    {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, Do[x[[r,i]]=0,{r,1,nrhs}];Continue[]];
    imi=Abs[p[[i]]]; m=ii-imi-1;
    Do [ Do [ x[[r,i]]-=a[[imi+j]]*x[[r,i-m+j-1]], {j,1,m}], {r,1,nrhs}],
    {i,1,n}];
  Do [ii=Abs[p[[i+1]]]; Do[x[[r,i]]*=a[[ii]], {r,1,nrhs}], {i,1,n}];
  Do [ii=p[[i+1]]; If [ii<0, Do[x[[r,i]]=b[[r,i]],{r,1,nrhs}];Continue[]];
    m=ii-Abs[p[[i]]]-1;
    Do [ Do [ x[[r,i-j]]-=a[[ii-j]]*x[[r,i]], {j,1,m}], {r,1,nrhs}],
    {i,n,1,-1}];
  Return[x]
];

```

Cell 26.6 Multiplying Skymatrix by Individual Vector

```

SymmSkyMatrixVectorMultiply[S_,x_]:= Module[
  {p,a,n,i,j,k,m,ii,b},
  {p,a}=S; n=Length[p]-1;
  If [n!=Length[x], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixVectorMultiply"]; Return[Null]];
  b=Table[a[[ Abs[p[[i+1]]] ]]*x[[i]], {i,1,n}];
  Do [ii=Abs[p[[i+1]]]; m=ii-Abs[p[[i]]]-1; If [m<=0,Continue[]];
    b[[i]]+=Take[a,{ii-m,ii-1}].Take[x,{i-m,i-1}];
    Do [b[[i-k]]+=a[[ii-k]]*x[[i]],{k,1,m}],
  {i,1,n}];
  Return[b]
];

(*
ClearAll[n]; n=10; SeedRandom[314159];
p=Table[0,{n+1}]; Do[p[[i+1]]=p[[i]]+
  Max[1,Min[i,Round[Random[]*i]]],{i,1,n}];
a=Table[1.,{i,1,p[[n+1]]}];
Print["Mean Band=",N[p[[n+1]]/n]];
S={p,a};
Sr=SymmSkyMatrixLDUReconstruct[S];
Print["Reconstructed SkyMatrix:"]; SymmSkyMatrixLowerTrianglePrint[Sr];
SymmSkyMatrixLowerTriangleMap[Sr];
x=Table[1.,{i,1,n}];
b=SymmSkyMatrixVectorMultiply[Sr,x];
Print["b=Ax=",b];*)

```

$$\mathbf{X} = \text{SymmSkyMatrixVectorSolve}[\mathbf{Sf}, \mathbf{B}]$$

to provide the solution \mathbf{X} of $\mathbf{SX} = \mathbf{B}$. This module is listed in Cell 26.5. The input arguments and function returns have the same function as those described for `SymmSkyMatrixVectorSolve`. The main difference is that \mathbf{B} and \mathbf{X} are matrices (two-dimensional lists) with the right-hand side and solution vectors as columns. There is a similar module `SymmSkyMatrixRowBlockSolve`, notlisted here, which solves for multiple right hand sides stored as rows of a matrix.

§26.3.4. *Matrix-Vector Multiply

For various applications it is necessary to form the matrix-vector product

$$\mathbf{b} = \mathbf{Sx} \quad (26.17)$$

where \mathbf{S} is a symmetric skymatrix and \mathbf{x} is given.

This is done by module `SymmSkyMatrixVectorMultiply`, which is listed in Cell 26.6. Its arguments are the skymatrix \mathbf{S} and the vector \mathbf{x} . The function returns \mathbf{Sx} in \mathbf{b} .

Module `SymmSkyMatrixColBlockMultiply` implements the multiplication by a block of vectors stored as columns of a rectangular matrix \mathbf{X} :

$$\mathbf{B} = \mathbf{SX} \quad (26.18)$$

Cell 26.7 Multiplying Skymatrix by Vector Block

```

SymmSkyMatrixColBlockMultiply[S_,x_]:= Module[
  {p,a,n,nrhs,i,j,k,m,r,ii,aij,b},
  {p,a}=S; n=Length[p]-1;
  If [n!=Dimensions[x][[1]], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixColBlockMultiply"]; Return[Null]];
  nrhs = Dimensions[x][[2]]; b=Table[0,{n},{nrhs}];
  Do [ii=Abs[p[[i+1]]]; m=ii-Abs[p[[i]]]-1;
    Do [b[[i,r]]=a[[ii]]*x[[i,r]], {r,1,nrhs}];
    Do [j=i-k; aij=a[[ii-k]]; If [aij==0, Continue[]];
      Do [b[[i,r]]+=aij*x[[j,r]]; b[[j,r]]+=aij*x[[i,r]], {r,1,nrhs}],
      {k,1,m}],
    {i,1,n}];
  Return[b]
];

SymmSkyMatrixRowBlockMultiply[S_,x_]:= Module[
  {p,a,n,nrhs,i,j,k,m,r,ii,aij,b},
  {p,a}=S; n=Length[p]-1;
  If [n!=Dimensions[x][[2]], Print["Inconsistent matrix dimensions in",
    " SymmSkyMatrixRowBlockMultiply"]; Return[Null]];
  nrhs = Dimensions[x][[1]]; b=Table[0,{nrhs},{n}];
  Do [ii=Abs[p[[i+1]]]; m=ii-Abs[p[[i]]]-1;
    Do [b[[r,i]]=a[[ii]]*x[[r,i]], {r,1,nrhs}];
    Do [j=i-k; aij=a[[ii-k]]; If [aij==0, Continue[]];
      Do [b[[r,i]]+=aij*x[[r,j]]; b[[r,j]]+=aij*x[[r,i]], {r,1,nrhs}],
      {k,1,m}],
    {i,1,n}];
  Return[b]
];

```

This module is listed in Cell 26.7. Its arguments are the skymatrix S and the rectangular matrix X . The function returns SX in B .

There is a similar module `SymmSkyMatrixRowBlockMultiply`, also listed in Cell 26.7, which postmultiplies a vector block stored as rows.

§26.3.5. *Printing and Mapping

Module `SymmSkyMatrixUpperTrianglePrint`, listed in Cell 26.8, prints a symmetric skymatrix in upper triangle form. It is invoked as

```
SymmSkyMatrixUpperTrianglePrint[S]
```

where S is the skymatrix to be printed. For an example of use see Cells 262-3.

The print format resembles the configuration depicted in Section 26.1. This kind of print is useful for program

development and debugging although of course it should not be attempted with a very large matrix.⁵

There is a similar module called `SymmSkyMatrixLowerTrianglePrint`, which displays the skymatrix entries in lower triangular form. This module is also listed in Cell 26.8.

Sometimes one is not interested in the actual values of the skymatrix entries but only on how the skyline template looks like. Such displays, called *maps*, can be done with just one symbol per entry. Module `SymmSkyMatrixUpperTriangleMap`, listed in Cell 26.9, produces a map of its argument. It is invoked as

$$\text{SymmSkyMatrixUpperTriangleMap}[S]$$

The entries within the skyline template are displayed by symbols +, – and 0, depending on whether the value is positive, negative or zero, respectively. Entries outside the skyline template are blank.

As in the case of the print module, there is module `SymmSkyMatrixLowerTriangleMap` which is also listed in Cell 26.9.

§26.3.6. *Reconstruction of SkyMatrix from Factors

In testing factorization and solving modules it is convenient to have modules that perform the “inverse process” of the factorization. More specifically, suppose that \mathbf{U} , \mathbf{D} (or \mathbf{D}^{-1}) are given, and the problem is to reconstruct the skymatrix that have them as factors:

$$\mathbf{S} = \mathbf{LDU}, \quad \text{or} \quad \mathbf{S} = \mathbf{LD}^{-1}\mathbf{U} \quad (26.19)$$

in which $\mathbf{L} = \mathbf{U}^T$. Modules `SymmSkyMatrixLDUReconstruction` and `SymmSkyMatrixLDinvUReconstruction` perform those operations. These modules are listed in Cell 26.10. Their argument is a factored form of \mathbf{S} : \mathbf{U} and \mathbf{D} in the first case, and \mathbf{U} and \mathbf{D}^{-1} in the second case.

⁵ Computer oriented readers may notice that the code for the printing routine is substantially more complex than those of the computational modules. This is primarily due to the inadequacies of *Mathematica* in handling tabular format output. The corresponding Fortran or C implementations would be simpler because those languages provide much better control over low-level display.

Cell 26.8 Skymatrix Printing

```

SymmSkyMatrixLowerTrianglePrint[S_]:= Module[
  {p,a,cycle,i,ii,ij,it,j,jj,j1,j2,jref,jbeg,jend,jt,kcmax,kc,kr,m,n,c,t},
  {p,a}=S; n=Dimensions[p][[1]]-1; kcmax=5; jref=0;
  Label[cycle]; Print[" "];
  jbeg=jref+1; jend=Min[jref+kcmax,n]; kc=jend-jref;
  t=Table[" ",{n-jref+1},{kc+1}];
  Do [If [p[[j+1]]>0,c=" ",c="*"];
    t[[1,j-jref+1]]=StringJoin[c,"Col",ToString[PaddedForm[j,3]]],
    {j,jbeg,jend}]; it=1;
  Do [ii=Abs[p[[i+1]]]; m=ii-Abs[p[[i]]]-1; j1=Max[i-m,jbeg]; j2=Min[i,jend];
    kr=j2-j1+1; If [kr<=0, Continue[]]; If [p[[i+1]]>0,c=" ",c="*"];
    it++; t[[it,1]]=StringJoin[c,"Row",ToString[PaddedForm[i,3]]];
    jt=j1-jbeg+2; ij=j1+ii-i;
    Do[t[[it,jt++]]=PaddedForm[a[[ij++]]]//FortranForm,{7,4}},{j,1,kr}},
    {i,jbeg,n}];
  Print[TableForm[Take[t,it],TableAlignments->{Right,Right},
    TableDirections->{Column,Row},TableSpacing->{0,2}]];
  jref=jend; If[jref<n,Goto[cycle]];
];

SymmSkyMatrixUpperTrianglePrint[S_]:= Module[
  {p,a,cycle,i,ij,it,j,j1,j2,jref,jbeg,jend,kcmax,kc,m,n,c,t},
  {p,a}=S; n=Dimensions[p][[1]]-1; kcmax=5; jref=0;
  Label[cycle]; Print[" "];
  jbeg=jref+1; jend=Min[jref+kcmax,n]; kc=jend-jref;
  t=Table[" ",{jend+1},{kc+1}];
  Do [If [p[[j+1]]>0,c=" ",c="*"];
    t[[1,j-jref+1]]=StringJoin[c,"Col",ToString[PaddedForm[j,3]]],
    {j,jbeg,jend}]; it=1;
  Do [it++; If [p[[i+1]]>0,c=" ",c="*"];
    t[[it,1]]=StringJoin[c,"Row",ToString[PaddedForm[i,3]]]; j=jref;
    Do [j++; If [j<i, Continue[]]; ij=Abs[p[[j+1]]]+i-j;
      If [ij<=Abs[p[[j]]], Continue[]];
      t[[it,k+1]]=PaddedForm[a[[ij]]]//FortranForm,{7,4},
      {k,1,kc}},
    {i,1,jend}];
  Print[TableForm[Take[t,it],TableAlignments->{Right,Right},
    TableDirections->{Column,Row},TableSpacing->{0,2}]];
  jref=jend; If[jref<n,Goto[cycle]];
];

Sr={{0, 1, 3, 6}, {1., 2., 7., 4., 23., 97.}};
SymmSkyMatrixLowerTrianglePrint[Sr];SymmSkyMatrixUpperTrianglePrint[Sr];

```

Cell 26.9 Skymatrix Mapping

```

SymmSkyMatrixLowerTriangleMap[S_]:=Module[
  {p,a,cycle,i,ii,ij,it,itop,j,jj,j1,j2,jref,jbeg,jend,jt,kcmax,kc,kr,m,n,c,t},
  {p,a}=S; n=Dimensions[p][[1]]-1; kcmax=40; jref=0;
  Label[cycle]; Print[" "];
  jbeg=jref+1; jend=Min[jref+kcmax,n]; kc=jend-jref;
  itop=2; If[jend>9,itop=3]; If[jend>99,itop=4]; If[jend>999,itop=5];
  t=Table["",{n-jref+itop},{kc+1}]; it=0;
  If [itop>=5, it++; Do [m=Floor[j/1000];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  If [itop>=4, it++; Do [m=Floor[j/100];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  If [itop>=3, it++; Do [m=Floor[j/10];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  it++; Do[t[[it,j-jref+1]]=ToString[Mod[j,10]],{j,jbeg,jend}];
  it++; Do[If[p[[j+1]]<0,t[[it,j-jref+1]]="*"],{j,jbeg,jend}];
  Do [ii=Abs[p[[i+1]]]; m=ii-Abs[p[[i]]]-1; j1=Max[i-m,jbeg]; j2=Min[i,jend];
    kr=j2-j1+1; If [kr<=0, Continue[]]; If [p[[i+1]]>0,c=" ",c="*"];
    it++; t[[it,1]]=StringJoin[ToString[PaddedForm[i,2]],c];
    jt=j1-jbeg+2; ij=j1+ii-i;
    Do [ c=" 0"; If[a[[ij]]>0,c=" +"]; If[a[[ij++]]<0,c=" -"];
      t[[it,jt++]]=c, {j,1,kr}],
    {i,jbeg,n}];
  Print[TableForm[Take[t,it],TableAlignments->{Right,Right},
    TableDirections->{Column,Row},TableSpacing->{0,0}]];
  jref=jend; If[jref<n,Goto[cycle]];
];

SymmSkyMatrixUpperTriangleMap[S_]:=Module[
  {p,a,cycle,i,ij,it,itop,j,j1,j2,jref,jbeg,jend,kcmax,kc,m,n,c,t},
  {p,a}=S; n=Dimensions[p][[1]]-1; kcmax=40; jref=0;
  Label[cycle]; Print[" "];
  jbeg=jref+1; jend=Min[jref+kcmax,n]; kc=jend-jref;
  itop=2; If[jend>9,itop=3]; If[jend>99,itop=4]; If[jend>999,itop=5];
  t=Table["",{jend+itop},{kc+1}]; it=0;
  If [itop>=5, it++; Do [m=Floor[j/1000];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  If [itop>=4, it++; Do [m=Floor[j/100];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  If [itop>=3, it++; Do [m=Floor[j/10];
    If [m>0,t[[it,j-jref+1]]=ToString[Mod[m,10]], {j,jbeg,jend}]];
  it++; Do[t[[it,j-jref+1]]=ToString[Mod[j,10]],{j,jbeg,jend}];
  it++; Do[If[p[[j+1]]<0,t[[it,j-jref+1]]="*"],{j,jbeg,jend}];
  Do [it++; If [p[[i+1]]>0,c=" ",c="*"];
    t[[it,1]]=StringJoin[ToString[PaddedForm[i,2]],c]; j=jref;
    Do [j++; If [j<i, Continue[]]; ij=Abs[p[[j+1]]]+i-j;
      If [ij<=Abs[p[[j]]], Continue[]]; c=" 0";
      If[a[[ij]]>0,c=" +"]; If[a[[ij++]]<0,c=" -"]; t[[it,k+1]]=c,
      {k,1,kc}],
    {i,1,jend}];
  Print[TableForm[Take[t,it],TableAlignments->{Right,Right},
    TableDirections->{Column,Row},TableSpacing->{0,0}]];
  jref=jend; If[jref<n,Goto[cycle]];
];

```

Cell 26.10 Skymatrix Reconstruction from Factors

```

SymmSkyMatrixLDUReconstruct[S_]:= Module[
  {p,ldu,a,v,n,i,ii,ij,j,jj,jk,jmj,k,m},
  {p,ldu}=S; a=ldu; n=Length[p]-1; v=Table[0,{n}];
  Do [jmj=Abs[p[[j]]]; jj=p[[j+1]]; If [jj<0, Continue[]];
    jk=jj-jmj; v[[jk]]=ldu[[jj]];
    Do [ij=jmj+k; i=j+ij-jj; ii=p[[i+1]]; If [ii<0, v[[k]]=0; Continue[]];
      If [i!=j, v[[k]]=ldu[[ij]]*ldu[[ii]]];
      m=Min[ii-Abs[p[[i]]],k]; a[[ij]]= v[[k]];
      a[[ij]]+=Take[ldu,{ii-m+1,ii-1}].Take[v,{k-m+1,k-1}],
      {k,1,jk}],
    {j,1,n}]; Return[{p,a}];
];

SymmSkyMatrixLDinvUReconstruct[S_]:= Module[
  {p,ldu,a,v,n,i,ii,ij,j,jj,jk,jmj,k,m},
  {p,ldu}=S; a=ldu; n=Length[p]-1; v=Table[0,{n}];
  Do [jmj=Abs[p[[j]]]; jj=p[[j+1]]; If [jj<0, Continue[]];
    jk=jj-jmj; v[[jk]]=1/ldu[[jj]];
    Do [ij=jmj+k; i=j+ij-jj; ii=p[[i+1]]; If [ii<0, v[[k]]=0; Continue[]];
      If [i!=j, v[[k]]=ldu[[ij]]/ldu[[ii]]];
      m=Min[ii-Abs[p[[i]]],k]; a[[ij]]= v[[k]];
      a[[ij]]+=Take[ldu,{ii-m+1,ii-1}].Take[v,{k-m+1,k-1}],
      {k,1,jk}],
    {j,1,n}]; Return[{p,a}];
];

p={0,1,2,5,8,9,15}; s={11,22,13,0,33,24,34,44,55,16,0,0,46,56,66};
S={p,s};
Sr=SymmSkyMatrixLDinvUReconstruct[S]; Print[Sr//InputForm];
Print[SymmSkyMatrixFactor[Sr,0]];

```

§26.3.7. *Miscellaneous Utilities

Finally, Cell 26.11 lists three miscellaneous modules. The most useful one is probably `SymmSkyMatrixConvertToFull`, which converts its skymatrix argument to a fully stored symmetric matrix. This is useful for things like a quick and dirty computation of eigenvalues:

```
Print[Eigenvalues[SymmSkyMatrixConvertToFull[S]]];
```

because *Mathematica* built-in eigensolvers require that the matrix be supplied in full storage form.

Cell 26.11 Miscellaneous Skymatrix Utilities

```

SymmSkyMatrixConvertToFull[S_]:= Module[
  {p,a,aa,n,j,jj,jmj,k},
  {p,a}=S; n=Length[p]-1; aa=Table[0,{n},{n}];
  Do [jmj=Abs[p[[j]]]; jj=Abs[p[[j+1]]]; aa[[j,j]]=a[[jj]];
    Do [aa[[j,j-k]]=aa[[j-k,j]]=a[[jj-k]],{k,1,jj-jmj-1}],
  {j,1,n}]; Return[aa];
];

SymmSkyMatrixConvertUnitUpperTriangleToFull[S_]:= Module[
  {p,ldu,aa,n,j,jj,jmj,k},
  {p,ldu}=S; n=Length[p]-1; aa=Table[0,{n},{n}];
  Do [jmj=Abs[p[[j]]]; jj=Abs[p[[j+1]]]; aa[[j,j]]=1;
    Do [aa[[j-k,j]]=ldu[[jj-k]],{k,1,jj-jmj-1}],
  {j,1,n}]; Return[aa];
];

SymmSkyMatrixConvertDiagonalToFull[S_]:= Module[
  {p,ldu,aa,n,i,j,jj,jmj,k},
  {p,ldu}=S; n=Length[p]-1; aa=Table[0,{n},{n}];
  Do [jj=Abs[p[[j+1]]]; aa[[j,j]]=ldu[[jj]],
  {j,1,n}]; Return[aa];
];

```

Homework Exercises for Chapter 26
Solving FEM Equations

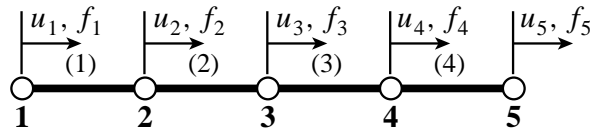


Figure 26.1. Structure for Exercise 26.1

EXERCISE 26.1 [A/C:10+10+15] Consider the 4-element assembly of bar elements shown in Figure 26.1. The only degree of freedom at each node is a translation along x . The element stiffness matrix of each element is

$$\mathbf{K}^{(e)} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (\text{E26.1})$$

- (a) Assemble the 5×5 master stiffness matrix \mathbf{K} showing it as a full symmetric matrix. Hint: the diagonal entries are 1, 2, 2, 2, 1.
- (b) Show \mathbf{K} stored as a skyline matrix using a representation like illustrated in (26.15). Hint $\mathbf{p} = \{ 0, 1, 3, 5, 7, 9 \}$.
- (c) Perform the symmetric factorization $\mathbf{K} = \mathbf{LDL}^T$ of (26.7), where \mathbf{K} is stored as a full matrix to simplify hand work.⁶ Show that the entries of \mathbf{D} are 1, 1, 1, 1 and 0, which mean that \mathbf{K} is singular. Why?

⁶ You can do this with *Mathematica* using the function `LUdecomposition`, but hand work is as quick.

27

A Complete Plane Stress FEM Program

TABLE OF CONTENTS

	Page
§27.1. Introduction	27-3
§27.2. Analysis Stages	27-3
§27.3. Model Definition	27-3
§27.3.1. Benchmark Problems	27-4
§27.3.2. Node Coordinates	27-4
§27.3.3. Element Type	27-6
§27.3.4. Element Connectivity	27-6
§27.3.5. Material Properties	27-8
§27.3.6. Fabrication Properties	27-8
§27.3.7. Node Freedom Tags	27-8
§27.3.8. Node Freedom Values	27-9
§27.3.9. Processing Options	27-9
§27.3.10. Model Display Utility	27-9
§27.3.11. Model Definition Print Utilities	27-10
§27.3.12. Model Definition Script Samples	27-10
§27.4. Processing	27-12
§27.4.1. Processing Tasks	27-12
§27.5. Postprocessing	27-14
§27.5.1. Result Print Utilities	27-14
§27.5.2. Displacement Field Contour Plots	27-15
§27.5.3. Stress Field Contour Plots	27-15
§27.5.4. Animation	27-16
§27.6. A Complete Problem Script Cell	27-16
§27. Notes and Bibliography	27-18

§27.1. Introduction

This Chapter describes a complete finite element program for analysis of plane stress problems. Unlike the previous chapters the description is top down, i.e. starts from the main driver down to more specific modules.

The program can support questions given in the take-home final exam, if they pertain to the analysis of given plane stress problems. Consequently this Chapter serves as an informal users manual.

§27.2. Analysis Stages

As in all DSM-based FEM programs, the analysis of plane stress problems involves three major stages: (I) preprocessing or model definition, (II) processing, and (III) postprocessing.

The preprocessing portion of the plane stress analysis is done by the first part of the *problem script*, driver program, already encountered in Chapters 21-22. The script directly sets the problem data structures. Preprocessing tasks include:

- I.1 Model definition by direct setting of the data structures.
- I.2 Plot of the FEM mesh for verification. At the minimum this involves producing a mesh picture that shows nodes and element labels.

The processing stage involves three tasks:

- II.1 Assembly of the master stiffness equations $\mathbf{Ku} = \mathbf{f}$. The plane stress assembler is of multiple element type (MET). This kind of assembler was discussed in §27.4. Element types include various plane stress Iso-P models as well as bars.
- II.2 Application of displacement BC by a modification method that produces $\hat{\mathbf{K}}\mathbf{u} = \hat{\mathbf{f}}$. The same modules described in §21.3.3 are used, since those are application problem independent.
- II.3 Solution of the modified equations for node displacements \mathbf{u} . The built-in *Mathematica* function `LinearSolve` is used for this task.

Upon executing the processing steps, the nodal displacement solution is available. The postprocessing stage embodies three tasks:

- III.1 Recovery of node forces including reactions through built-in matrix multiplication $\mathbf{f} = \mathbf{Ku}$.
- III.2 Recovery of plate stresses and bar forces (if bars are present). The former are subject to interelement averaging (Chapter 28) to get nodal stresses.
- III.3 Print and plotting of results.

§27.3. Model Definition

The model-definition data may be broken down into three sets, which are listed below by order of appearance:

Model definition	$\left\{ \begin{array}{l} \text{Geometry data: node coordinates} \\ \text{Element data: type, connectivity, material and fabrication} \\ \text{Degree of freedom (DOF) activity data: force and displacement BCs} \end{array} \right.$
------------------	--

(27.1)

The element data is broken down into four subsets: type, connectivity, material and fabrication, each of which has its own data structure. The degree of freedom data is broken into two subsets:

Table 27.1. Plane Stress Model Definition Data Structures

Long name	Short name	Dimensions	Description
NodeCoordinates	nodxyz	numnod x 2	Node coordinates in global system
ElemTypes	elenod	numele	Element type identifiers
ElemNodes	elenod	numele x <var>	Element node lists
ElemMaterials	elemat	numele x <var>	Element material properties
ElemFabrications	elefab	numele x <var>	Element fabrication properties
NodeDOFTags	nodtag	numnod x 2	Node freedom tags marking BC type
NodeDOFValues	nodval	numnod x 2	Node freedom specified values
ProcessOptions	prcopt	<var>	Processing specifications
Notation: numnod: number of nodes (no numbering gaps allowed, each node has two) displacement DOFs; numele: number of elements; <var> variable dimension. Long names are used in user-written problem scripts. Short names are used in programmed modules. Dimensions refer to the first two level Dimensions of the <i>Mathematica</i> list that implements the data structure. These give a guide as to implementation in a low level language such as C.			

tags and values. In addition there are miscellaneous process options, such as the symbolic versus numeric processing. These options are conveniently collected in a separate data set.

Accordingly, the model-definition input to the plane stress FEM program consists of eight data structures, which are called NodeCoordinates, ElemTypes, ElemNodes, ElemMaterials, ElemFabrications, NodeDOFTags, NodeDOFValues and ProcessOptions. These are summarized in Table 27.1.

The configuration of these data structures are described in the following subsections with reference to the benchmark problems and discretizations shown in Figures 27.1 and 27.2.

§27.3.1. Benchmark Problems

Figure 27.1(a) illustrates a rectangular steel plate in plane stress under uniform uniaxial loading in the y (vertical) direction. Note that the load q is specified as force per unit area (kips per square inch); thus q has not been integrated through the thickness h . The exact analytical solution of the problem is¹ $\sigma_{yy} = q$, $\sigma_{xx} = \sigma_{xy} = 0$, $u_y = qy/E$, $u_x = -\nu u_y = -\nu qx/E$. This problem should be solved exactly by *any* finite element mesh as long as the model is consistent and stable. In particular, the two one-quadrilateral-element models shown in Figure 27.1(b,c).

A similar but more complicated problem is shown in Figure 27.2: a rectangular steel plate dimensioned and loaded as that of Figure 27.1(a) but now with a central circular hole. This problem is defined in Figure 27.2(a). A FEM solution is to be obtained using the two quadrilateral element models (with 4-node and 9-nodes, respectively) depicted in Figure 27.2(b,c). The main result sought is the stress concentration factor on the hole boundary, and comparison of this computed factor with the exact analytical value.

¹ The displacement solution $u_y = qy/E$ and $u_x = -\nu u_y$ assumes that the plate centerlines do not translate or rotate, a condition enforced in the FEM discretizations shown in Figure 27.1(b,c).

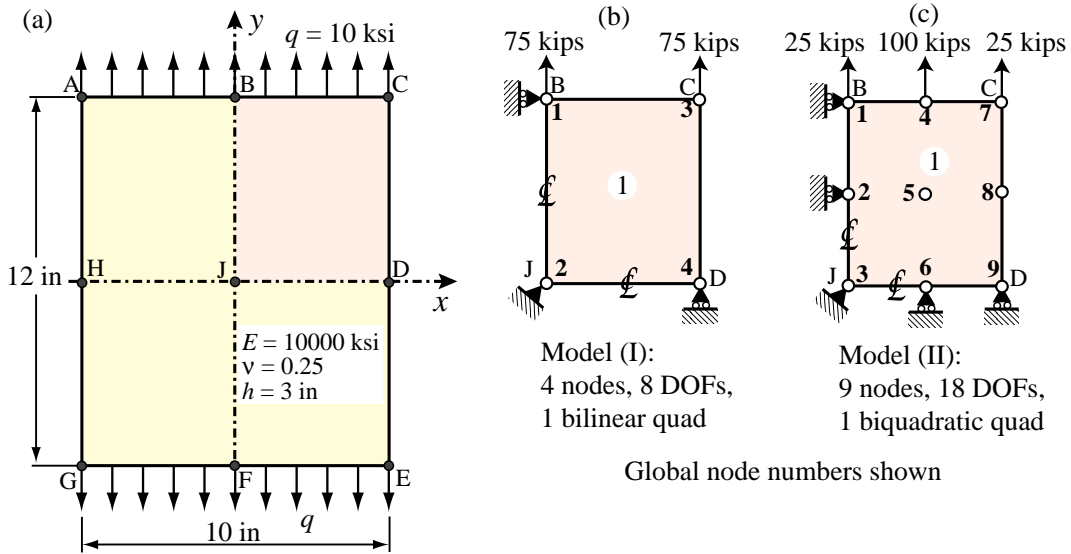


FIGURE 27.1. Rectangular plate under uniform uniaxial loading, and two one-element FEM discretizations of its upper right quadrant.

§27.3.2. Node Coordinates

The geometry data is specified through `NodeCoordinates`. This is a list of node coordinates configured as

$$\text{NodeCoordinates} = \{ \{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_N, y_N\} \} \quad (27.2)$$

where N is the number of nodes. Coordinate values should be normally be floating point numbers;² use the `N` function to insure that property if necessary. Nodes must be numbered consecutively and no gaps are permitted.

Example 27.1. For Model (I) of Figure 27.1(b):

```
NodeCoordinates=N[{ {0,6},{0,0},{5,6},{5,0} }];
```

Example 27.2. For Model (II) of Figure 27.1(c):

```
NodeCoordinates=N[{ {0,6},{0,3},{0,0},{5/2,6},{5/2,3},{5/2,0},{5,6},{5,3},{5,0} }];
```

Example 27.3. For Model (I) of Figure 27.2(a), using a bit of coordinate generation:

```
s={1,0.70,0.48,0.30,0.16,0.07,0.0};
xy1={0,6}; xy7={0,1}; xy8={2.5,6}; xy14={Cos[3*Pi/8],Sin[3*Pi/8]};
xy8={2.5,6}; xy21={Cos[Pi/4],Sin[Pi/4]}; xy15={5,6};
xy22={5,2}; xy28={Cos[Pi/8],Sin[Pi/8]}; xy29={5,0}; xy35={1,0};
NodeCoordinates=Table[{0,0},{35}];
Do[NodeCoordinates[[n]]=N[s[[n]]*xy1+(1-s[[n]])*xy7],{n,1,7}];
Do[NodeCoordinates[[n]]=N[s[[n-7]]*xy8+(1-s[[n-7]])*xy14],{n,8,14}];
Do[NodeCoordinates[[n]]=N[s[[n-14]]*xy15+(1-s[[n-14]])*xy21],{n,15,21}];
```

² Unless one is doing a symbolic or exact-arithmetic analysis. Those are rare at the level of a full FEM analysis.

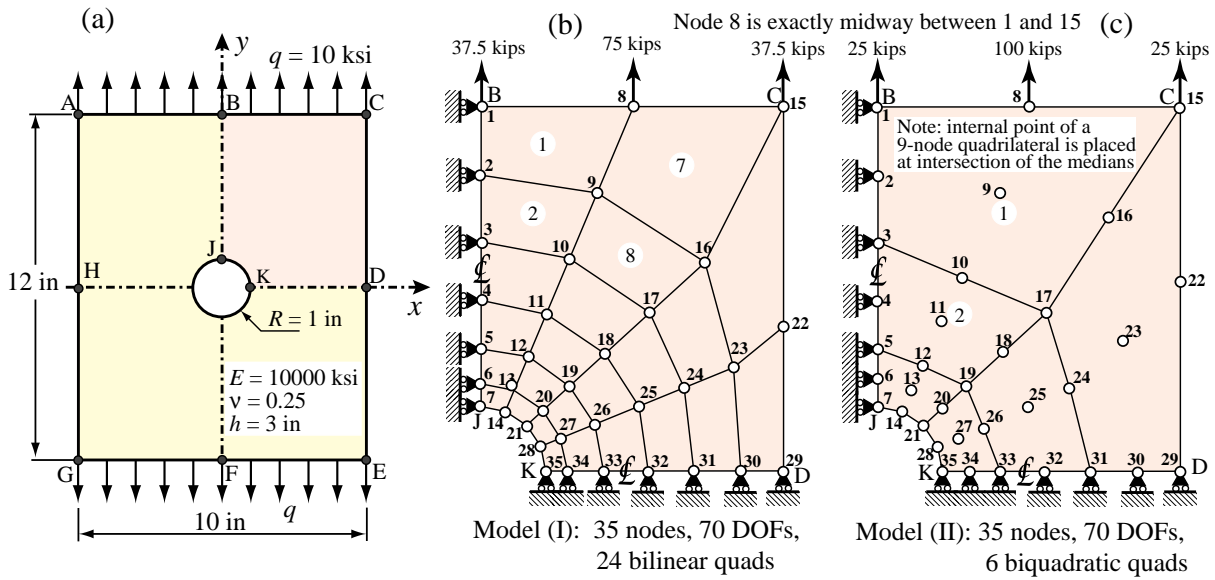


FIGURE 27.2. Plate with a circular hole and two FEM discretizations of its upper right quadrant. Only a few element numbers are shown to reduce clutter. Note: the meshes generated with the example scripts given later differ slightly from those shown above. Compare, for example, mesh in (b) above with that in Figure 27.3.

```
Do[NodeCoordinates[[n]] = N[s[[n-21]]*xy22 + (1-s[[n-21]])*xy28], {n, 22, 28}];
Do[NodeCoordinates[[n]] = N[s[[n-28]]*xy29 + (1-s[[n-28]])*xy35], {n, 29, 35}];
```

The result of this generation is that some of the interior nodes are not in the same positions as sketched in Figure 27.2(b), but this slight change hardly affects the results.

§27.3.3. Element Type

The element type is a label that specifies the type of model to be used. These labels are placed into an element type list:

$$\text{ElemTypes} = \{\text{type}^{(1)}, \text{type}^{(2)}, \dots, \text{type}^{N_e}\} \quad (27.3)$$

Here type^e is the type identifier of the e -th element specified as a character string. N_e is the number of elements; no element numbering gaps are allowed. Legal type identifiers are listed in Table 27.2.

```
ElemTypes=Table["Quad4",{numele}];
ElemTypes=Table["Quad9",{numele}];
```

Here `numele` is a variable that contains the number of elements. This can be extracted, for example, as `numele=Length[ElemNodes]`, where `ElemNodes` is defined below, if `ElemNodes` is defined first as is often the case.

§27.3.4. Element Connectivity

Element connectivity information specifies how the elements are connected.³ This information is

³ Some FEM programs call this the “topology” data.

Table 27.2. Element Type Identifiers Implemented in Plane Stress Program

Identifier	Nodes	Model for	Description
"Bar2"	2	bar	2-node bar element
"Bar3"	3	bar	3-node bar element; may be curved
"Trig3"	3	plate	3-node linear triangle
"Trig6"	6	plate	6-node quadratic triangle (3-interior point Gauss rule)
"Trig6.-3"	6	plate	6-node quadratic triangle (midpoint Gauss rule)
"Trig10"	10	plate	10-node cubic triangle (7 point Gauss rule)
"Quad4"	4	plate	RS 4-node iso-P bilinear quad (2×2 Gauss rule)
"Quad4.1"	4	plate	RD 4-node iso-P bilinear quad (1-point Gauss rule)
"Quad8"	8	plate	RS 8-node iso-P serendipity quad (3×3 Gauss rule)
"Quad8.2"	8	plate	RD 8-node iso-P serendipity quad (2×2 Gauss rule)
"Quad9"	9	plate	RS 9-node iso-P biquadratic quad (3×3 Gauss rule)
"Quad9.2"	9	plate	RD 9-node iso-P biquadratic quad (2×2 Gauss rule)
RS: rank sufficient, RD: Rank deficient.			

stored in ElemNodes, which is a list of element nodelists:

$$\text{ElemNodes} = \{ \{ \text{enl}^{(1)} \}, \{ \text{enl}^{(2)} \}, \dots \{ \text{enl}^{N_e} \} \} \quad (27.4)$$

Here enl^e denotes the lists of nodes of the element e (as given by global node numbers) and N_e is the total number of elements.

Element boundaries must be traversed counterclockwise (CCW) but you can start at any corner. Numbering elements with midnodes requires more care: begin listing corners CCW, followed by midpoints also CCW (first midpoint is the one that follows first corner when traversing CCW). If element has thirdpoints, as in the case of the 10-node triangle, begin listing corners CCW, followed by thirdpoints CCW (first thirdpoint is the one that follows first corner). When elements have an interior node, as in the 9-node quadrilateral or 10-node triangle, that node goes last.

Example 27.4. For Model (I) of which has only one 4-node quadrilateral:

```
ElemNodes={ { 1,2,4,3 } };
```

Example 27.5. For Model (II) of Figure 27.1(c), which has only one 9-node quadrilateral:

```
ElemNodes={ { 1,3,9,7,2,6,8,4,5 } };
```

numbering the elements from top to bottom and from left to right:

```
ElemNodes=Table[{0,0,0,0},{24}];
ElemNodes[[1]]={1,2,9,8};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,2,6}];
ElemNodes[[7]]=ElemNodes[[6]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,8,12}];
ElemNodes[[13]]=ElemNodes[[12]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,14,18}];
ElemNodes[[19]]=ElemNodes[[18]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,20,24}];
```

Example 27.6. For Model (II) of Figure 27.2(c), numbering the elements from top to bottom and from left to right:

```
ElemNodes=Table[{0,0,0,0,0,0,0,0,0},{6}];
ElemNodes[[1]]={1,3,17,15,2,10,16,8,9};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{2,2,2,2,2,2,2,2,2},{e,2,3}];
ElemNodes[[4]]=ElemNodes[[3]]+{10,10,10,10,10,10,10,10,10};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{2,2,2,2,2,2,2,2,2},{e,5,6}];
```

Since this particular mesh only has 6 elements, it would be indeed faster to write down the six nodelists.

§27.3.5. Material Properties

Data structure `ElemMaterials` is a list that provides the constitutive properties of the elements:

$$\text{ElemMaterials} = \{ \text{mprop}^{(1)}, \text{mprop}^{(2)}, \dots, \text{mprop}^{N_e} \} \quad (27.5)$$

For a plate element, `mprop` is the stress-strain matrix of elastic moduli (also known as elasticity matrix) arranged as $\{ \{ E_{11}, E_{12}, E_{33} \}, \{ E_{12}, E_{22}, E_{23} \}, \{ E_{13}, E_{23}, E_{33} \} \}$. Note that although this matrix is symmetric, it must be specified as a full 3×3 matrix. For a bar element, `mprop` is simply the longitudinal elastic modulus.

A common case in practice is that (i) all elements are plates, (ii) the plate material is uniform and isotropic. An isotropic elastic material is specified by the elastic modulus E and Poisson's ratio ν . Then this list can be generated by a single `Table` instruction upon building the elasticity matrix, as in the example below.

Example 27.7. For all FEM discretizations in Figures 27.1 and 27.2 all elements are plates of the same isotropic material. Suppose that values of the elastic modulus E and Poisson's ratio ν are stored in `Em` and `nu`, respectively, which are typically declared at the beginning of the problem script. Let `numele` give the number of elements. Then the material data is compactly declared by saying

```
Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
ElemMaterials=Table[Emat,{numele}];
```

§27.3.6. Fabrication Properties

Data structure `ElemFabrications` is a list that provides the fabrication properties of the elements:

$$\text{ElemFabrications} = \{ \text{fprop}^{(1)}, \text{fprop}^{(2)}, \dots, \text{fprop}^{N_e} \} \quad (27.6)$$

For a plate element, `fprop` is the thickness h of the plate, assumed constant.⁴ For a bar element, `fprop` is the cross section area.

If all elements are plates with the same thickness, this list can be easily generated by a `Table` instruction as in the example below.

⁴ It is possible also to specify a variable thickness by making `fprop` a list that contains the thicknesses at the nodes. Since the variable thickness case is comparatively rare, it will not be described here.

Example 27.8. For all FEM discretizations in Figures 27.1 and 27.2 all elements are plates with the same thickness h , which is stored in variable `th`. This is typically declared at the start of the problem script. As before, `numele` has the number of elements. Then the fabrication data is compactly declared by saying

```
ElemFabrications=Table[th,{ numele }]
```

§27.3.7. Node Freedom Tags

Data structure `NodeDOFTags` is a list that labels each node DOF as to whether the load or the displacement is specified. The configuration of this list is similar to that of `NodeCoordinates`:

$$\text{NodeDOFTags} = \{ \{ \text{tag}_{x1}, \text{tag}_{y1} \}, \{ \text{tag}_{x2}, \text{tag}_{y2} \}, \dots \{ \text{tag}_{xN}, \text{tag}_{yN} \} \} \quad (27.7)$$

The tag value is 0 if the force is specified and 1 if the displacement is specified.

When there are a lot of nodes, often the quickest way to specify this list is to create with a `Table` command that initializes it to all zeros. Then displacement BCs are inserted appropriately, as in the example below.

Example 27.9. For Model (I) in Figure 27.2(a):

```
numnod=Length[NodeCoordinates];
NodeDOFTags=Table[{0,0},{numnod}]; (* create and initialize to zero *)
Do[NodeDOFTags[[n]]={1,0},{n,1,7}]; (* vroller @ nodes 1 through 7 *)
Do[NodeDOFTags[[n]]={0,1},{n,29,35}]; (* hroller @ nodes 29 through 35 *)
```

This scheme works well because typically the number of supported nodes is small compared to the total number.

§27.3.8. Node Freedom Values

Data structure `NodeDOFValues` is a list with the same node by node configuration as `NodeDOFTags`:

$$\text{NodeDOFValues} = \{ \{ \text{value}_{x1}, \text{value}_{y1} \}, \{ \text{value}_{x2}, \text{value}_{y2} \}, \dots \{ \text{value}_{xN}, \text{value}_{yN} \} \} \quad (27.8)$$

Here `value` is the specified value of the applied node force component if the corresponding tag is zero, and of the prescribed displacement component if the tag is one.

Often most of the entries of (27.8) are zero. If so a quick way to build it is to create it with a `Table` command that initializes it to zero. Then nonzero values are inserted as in the example below.

Example 27.10. For the model (I) in Figure 27.2(a) only 3 values (for the y forces on nodes 1, 8 and 15) will be nonzero:

```
numnod=Length[NodeCoordinates];
NodeDOFValues=Table[{0,0},{numnod}]; (* create and initialize to zero *)
NodeDOFValues[[1]]=NodeDOFValues[[15]]={0,37.5};
NodeDOFValues[[8]]={0,75}; (* y nodal loads *)
```

§27.3.9. Processing Options

Array `ProcessOptions` is a list of general processing options that presently contains only the `numer` logical flag. This is normally be set to `True` to specify numeric computations:

```
ProcessOptions={ True };
```

§27.3.10. Model Display Utility

Only one graphic display utility is presently provided to show the mesh. Nodes and elements of Model (I) of Figure 27.2(a) may be plotted by saying

```
aspect=6/5;
Plot2DElementsAndNodes[NodeCoordinates,
ElemNodes,aspect,"Plate with circular
hole - 4-node quad model",True,True];
```

Here `aspect` is the plot frame aspect ratio (y dimension over x dimension). The 4th argument is a plot title textstring. The last two `True` argument values specify that node labels and element labels, respectively, be shown. The output of the mesh plot command is shown in Figure 27.3.

§27.3.11. Model Definition Print Utilities

Several print utilities are provided in the plane stress program to print out model definition data in tabular form. They are invoked as follows.

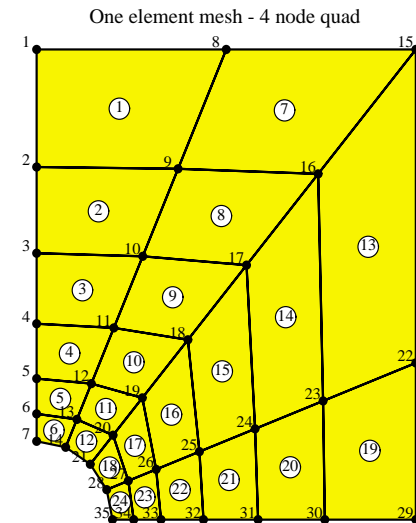


FIGURE 27.3. Mesh plot for Model (I) of Figure 27.2(a).

To print the node coordinates:

```
PrintPlaneStressNodeCoordinates[NodeCoordinates,title,digits];
```

To print the element types and nodes:

```
PrintPlaneStressElementTypeNodes[ElemTypes,ElemNodes,title,digits];
```

To print the element materials and fabrications:

```
PrintPlaneStressElementMatFab[ElemMaterials,ElemFabrications,title,digits];
```

To print freedom activity data:

```
PrintPlaneStressFreedomActivity[NodeDOFTags,NodeDOFValues,title,digits];
```

In all cases, `title` is an optional character string to be printed as a title before the table; for example "Node coordinates". To eliminate the title, specify "" (two quote marks together).

The last argument of the print modules: `digits`, is optional. If set to `{d,f}` it specifies that floating point numbers are to be printed with room for at least d digits, with f digits after the decimal point. If `digits` is specified as a void list: `{ }`, a preset default is used for d and f .

§27.3.12. Model Definition Script Samples

As capstone examples, Figures 27.4 and Figures 27.5 list the preprocessing (model definition) parts of the problem scripts for Model (I) and (II), respectively, of Figure 27.1(b,c).

```

ClearAll[Em,v,th];
Em=10000; v=.25; th=3; aspect=6/5; Nsub=4;
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};

(* Define FEM model *)

NodeCoordinates=N[{{0,6},{0,0},{5,6},{5,0}}];
PrintPlaneStressNodeCoordinates[NodeCoordinates,"",{6,4}];
ElemNodes= {{1,2,4,3}};
numnod=Length[NodeCoordinates]; numele=Length[ElemNodes];
ElemTypes= Table["Quad4",{numele}];
PrintPlaneStressElementTypeNodes[ElemTypes,ElemNodes,"",{1}];
ElemMaterials= Table[Emat, {numele}];
ElemFabrications=Table[th, {numele}];
PrintPlaneStressElementMatFab[ElemMaterials,ElemFabrications,"",{1}];
NodeDOFValues=NodeDOFTags=Table[{0,0},{numnod}];
NodeDOFValues[[1]]=NodeDOFValues[[3]]={0,75}; (* nodal loads *)
NodeDOFTags[[1]]={1,0}; (* vroller @ node 1 *)
NodeDOFTags[[2]]={1,1}; (* fixed node 2 *)
NodeDOFTags[[4]]={0,1}; (* hroller @ node 4 *)
PrintPlaneStressFreedomActivity[NodeDOFTags,NodeDOFValues,"",{1}];
ProcessOptions={True};
Plot2DElementsAndNodes[NodeCoordinates,ElemNodes,aspect,
    "One element mesh - 4-node quad",True,True];

```

FIGURE 27.4. Model definition part of Model (I) of Figure 27.1(b).

```

ClearAll[Em,v,th];
Em=10000; v=.25; th=3; aspect=6/5; Nsub=4;
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};

(* Define FEM model *)

NodeCoordinates=N[{{0,6},{0,3},{0,0},{5/2,6},{5/2,3},
    {5/2,0},{5,6},{5,3},{5,0}}];
PrintPlaneStressNodeCoordinates[NodeCoordinates,"",{6,4}];
ElemNodes= {{1,3,9,7,2,6,8,4,5}};
numnod=Length[NodeCoordinates]; numele=Length[ElemNodes];
ElemTypes= Table["Quad9",{numele}];
PrintPlaneStressElementTypeNodes[ElemTypes,ElemNodes,"",{1}];
ElemMaterials= Table[Emat, {numele}];
ElemFabrications=Table[th, {numele}];
PrintPlaneStressElementMatFab[ElemMaterials,ElemFabrications,"",{1}];
NodeDOFValues=NodeDOFTags=Table[{0,0},{numnod}];
NodeDOFValues[[1]]=NodeDOFValues[[7]]={0,25};
NodeDOFValues[[4]]={0,100}; (* nodal loads *)
NodeDOFTags[[1]]=NodeDOFTags[[2]]={1,0}; (* vroller @ nodes 1,2 *)
NodeDOFTags[[3]]={1,1}; (* fixed node 3 *)
NodeDOFTags[[6]]=NodeDOFTags[[9]]={0,1}; (* hroller @ nodes 6,9 *)
PrintPlaneStressFreedomActivity[NodeDOFTags,NodeDOFValues,"",{1}];
ProcessOptions={True};
Plot2DElementsAndNodes[NodeCoordinates,ElemNodes,aspect,
    "One element mesh - 9-node quad",True,True];

```

FIGURE 27.5. Model definition part of Model (II) of Figure 27.1(c).

```

PlaneStressSolution[nodxyz_,eletyp_,elenod_,elemat_,elefab_,
  nodtag_,nodval_,prcopt_] := Module[{K,Kmod,f,fmod,u,numer=True,
  noddis,nodfor,nodpnc,nodsig,barele,barfor},
  If [Length[prcopt]>=1, numer=prcopt[[1]]];
  K=PlaneStressMasterStiffness[nodxyz,eletyp,elenod,elemat,elefab,prcopt];
  If [K==Null,Return[Table[Null,{6}]]];
  Kmod=ModifiedMasterStiffness[nodtag,K];
  f=FlatNodePartVector[nodval];
  fmod=ModifiedNodeForces[nodtag,nodval,K,f];
  u=LinearSolve[Kmod,fmod]; If [numer,u=Chop[u]];
  f=K.u; If [numer,f=Chop[f]];
  nodfor=NodePartFlatVector[2,f]; noddis=NodePartFlatVector[2,u];
  {nodpnc,nodsig}=PlaneStressPlateStresses[nodxyz,eletyp,elenod,elemat,
    elefab,noddis,prcopt];
  {barele,barfor}=PlaneStressBarForces[nodxyz,eletyp,elenod,elemat,
    elefab,noddis,prcopt];
  ClearAll[K,Kmod];
  Return[{noddis,nodfor,nodpnc,nodsig,barele,barfor}];
];

```

FIGURE 27.6. Module to drive the analysis of the plane stress problem.

§27.4. Processing

Once the model definition is complete, the plane stress analysis is carried out by calling module `LinearSolutionOfPlaneStressModel` listed in Figure 27.6. This module is invoked from the problem driver as

```

{NodeDisplacements,NodeForces,NodePlateCounts,NodePlateStresses,
  ElemBarNumbers,ElemBarForces}= PlaneStressSolution[NodeCoordinates,
  ElemTypes,ElemNodes,ElemMaterials,ElemFabrications,
  NodeDOFTags,NodeDOFValues,ProcessOptions];

```

The module arguments: `NodeCoordinates`, `ElemTypes`, `ElemNodes`, `ElemMaterials`, `ElemFabrications`, `NodeDOFTags`, `NodeDOFValues` and `ProcessOptions` are the data structures described in the previous section.

The module returns the following:

- `NodeDisplacements` Computed node displacements, in node-partitioned form.
- `NodeForces` Recovered node forces including reactions, in node-partitioned form.
- `NodePlateCounts` For each node, number of plate elements attached to that node. A zero count means that no plate elements are attached to that node.
- `NodePlateStresses` Averaged stresses at plate nodes with a nonzero `NodePlateCounts`.
- `ElemBarNumbers` A list of bar elements if any specified, else an empty list.
- `ElemBarForces` A list of bar internal forces if any bar elements were specified, else an empty list.

§27.4.1. Processing Tasks

`PlaneStressSolution` carries out the following tasks. It assembles the free-free master stiffness matrix `K` by calling the MET assembler `PlaneStressMasterStiffness`. This module is listed in Figure 27.7. A study

```

PlaneStressMasterStiffness[nodxyz_,eletyp_,elenod_,elemat_,
  elefab_,prcopt_]:=Module[{numele=Length[elenod],
  numnod=Length[nodxyz],ncoor,type,e,enl,neldof,OKtyp,OKenl,
  i,j,n,ii,jj,eft,Emat,th,numer,Ke,K},
  OKtyp={"Bar2","Bar3","Quad4","Quad4.1","Quad8","Quad8.2","Quad9",
    "Quad9.2","Trig3","Trig6","Trig6.-3","Trig10","Trig10.6"};
  OKenl={2,3,4,4,8,8,9,9,3,6,6,10,10};
  K=Table[0,{2*numnod},{2*numnod}]; numer=prcopt[[1]];
  For [e=1,e<=numele,e++, type=eletyp[[e]];
    If [!MemberQ[OKtyp,type], Print["Illegal type:",type,
      " element e=",e," Assembly aborted"]; Return[Null] ];
    enl=elenod[[e]]; n=Length[enl]; {{j}}=Position[OKtyp,type];
    If [OKenl[[j]]!=n, Print ["Wrong node list length, element=",e,
      " Assembly aborted"]; Return[Null] ];
    eft=Flatten[Table[{2*enl[[i]]-1,2*enl[[i]]},{i,1,n}]];
    ncoor=Table[nodxyz[[enl[[i]]]],{i,1,n}];
    If [type=="Bar2", Em=elemat[[e]]; A=elefab[[e]];
      Ke=PlaneBar2Stiffness[ncoor,Em,A,{numer}] ];
    If [type=="Bar3", Em=elemat[[e]]; A=elefab[[e]];
      Ke=PlaneBar3Stiffness[ncoor,Em,A,{numer}] ];
    If [type=="Quad4", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,{numer,2}] ];
    If [type=="Quad4.1", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad4IsoPMembraneStiffness[ncoor,Emat,th,{numer,1}] ];
    If [type=="Quad8", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad8IsoPMembraneStiffness[ncoor,Emat,th,{numer,3}] ];
    If [type=="Quad8.2", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad8IsoPMembraneStiffness[ncoor,Emat,th,{numer,2}] ];
    If [type=="Quad9", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad9IsoPMembraneStiffness[ncoor,Emat,th,{numer,3}] ];
    If [type=="Quad9.2", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Quad9IsoPMembraneStiffness[ncoor,Emat,th,{numer,2}] ];
    If [type=="Trig3", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Trig3IsoPMembraneStiffness[ncoor,Emat,th,{numer}] ];
    If [type=="Trig6", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Trig6IsoPMembraneStiffness[ncoor,Emat,th,{numer,3}] ];
    If [type=="Trig6.-3", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Trig6IsoPMembraneStiffness[ncoor,Emat,th,{numer,-3}] ];
    If [type=="Trig10", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Trig10IsoPMembraneStiffness[ncoor,Emat,th,{numer,7}] ];
    If [type=="Trig10.6", Emat=elemat[[e]]; th=elefab[[e]];
      Ke=Trig10IsoPMembraneStiffness[ncoor,Emat,th,{numer,6}] ];
    neldof=Length[Ke];
    For [i=1,i<=neldof,i++, ii=eft[[i]];
      For [j=i,j<=neldof,j++, jj=eft[[j]];
        K[[jj,ii]]=K[[ii,jj]]+Ke[[i,j]] ];
    ];
  ]; Return[K];
];

```

FIGURE 27.7. Plane stress assembler module.

of its code reveals that it handle the multiple element types listed in Table 27.2. The modules that compute the element stiffness matrices have been studied in previous Chapters and need not be listed here.

The displacement BCs are applied by `ModifiedMasterStiffness` and `ModifiedNodeForces`. These are the same modules used in Chapter 21 for the space truss program, and thus need not be described further.

The unknown node displacements u are obtained through the built in `LinearSolve` function, as $u=\text{LinearSolve}[K_{\text{mod}},f_{\text{mod}}]$. This function is of course restricted to small systems, typically less than 200 equations (it gets extremely slow for something bigger) but it has the advantages of simplicity. The node

```

PlaneStressPlateStresses[nodxyz_,eletyp_,elenod_,elemat_,
  elefab_,noddis_,prcopt_]:=Module[{numele=Length[elenod],
  numnod=Length[nodxyz],ncoor,type,e,enl,i,k,n,Emat,th,
  numer=True,nodpnc,nodsig}, If [Length[prcopt]>0, numer=prcopt[[1]]];
  nodpnc=Table[0,{numnod}]; nodsig=Table[{0,0,0},{numnod}];
  If [Length[prcopt]>=1, numer=prcopt[[1]]];
  For [e=1,e<=numele,e++, type=eletyp[[e]];
    enl=elenod[[e]]; k=Length[enl];
    ncoor=Table[nodxyz[[enl[[i]]]],{i,1,k}];
    ue=Table[{noddis[[enl[[i]],1]],noddis[[enl[[i]],2]]},{i,1,k}];
    ue=Flatten[ue];
    If [StringTake[type,3]=="Bar", Continue[]];
    If [type=="Quad4", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad4IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Quad4.1", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad4IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Quad8", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad8IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Quad8.2", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad8IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Quad9", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad9IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Quad9.2", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Quad9IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Trig3", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Trig3IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Trig6", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Trig6IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Trig6.-3", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Trig6IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Trig10", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Trig10IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    If [type=="Trig10.6", Emat=elemat[[e]]; th=elefab[[e]];
      sige=Trig10IsoPMembraneStresses[ncoor,Emat,th,ue,{numer}] ];
    Do [n=enl[[i]]; nodpnc[[n]]++; nodsig[[n]]+=sige[[i]], {i,1,k}];
  ];
  Do [k=nodpnc[[n]]; If [k>1,nodsig[[n]]/=k], {n,1,numnod}];
  If [numer, nodsig=Chop[nodsig]];
  Return[{nodpnc,nodsig}];
];

```

FIGURE 27.8. Module that recovers averaged nodal stresses at plate nodes.

forces including reactions are recovered by the matrix multiplication $f = K \cdot u$, where K is the unmodified master stiffness.

Finally, plate stresses and bar internal forces are recovered by the modules `PlaneStressPlateStresses` and `PlaneStressBarForces`, respectively. The former is listed in Figure 27.8. This computation is actually part of the postprocessing stage. It is not described here since stress recovery is treated in more detail in a subsequent Chapter.

The bar force recovery module is not described here as it has not been used in assigned problems so far.

§27.5. Postprocessing

As noted above, module `PlaneStressSolution` carries out preprocessing tasks: recover node forces, plate stresses and bar forces. But those are not under control of the user. Here we describe result printing and plotting activities that can be specified in the problem script.

§27.5.1. Result Print Utilities

Several utilities are provided in the plane stress program to print solution data in tabular form. They are invoked as follows.

To print computed node displacements:

```
PrintPlaneStressNodeDisplacements [NodeDisplacements,title,digits];
```

To print recoverd node forces including reactions:

```
PrintPlaneStressNodeForces [NodeForces,title,digits];
```

To print plate node stresses:

```
PrintPlaneStressPlateNodeStresses [NodePlateCounts,NodePlateStresses,title,digits];
```

To print node displacements, node forces and node plate stresses in one table:

```
PrintPlaneStressSolution [NodeDisplacements,NodeForces,
                          NodePlateCounts,NodePlateStresses,title,digits];
```

No utility is provided to print bar forces, as problems involving plates and bars had not been assigned.

In all utilities listed above, *title* is an optional character string to be printed as a title before the table; for example "Node displacements for plate with a hole". To eliminate the title, specify "" (two quote marks together).

The last argument of the print modules: *digits*, is optional. If set to {*d*,*f*} it specifies that floating point numbers are to be printed with room for at least *d* digits, with *f* digits after the decimal point. If *digits* is specified as a void list: {}, a preset default is used for *d* and *f*.

§27.5.2. Displacement Field Contour Plots

Contour plots of displacement components u_x and u_y (interpolated over elements from the computed node displacements) may be produced. Displacement magnitudes are shown using a internally set color scheme based on "hue interpolation", in which white means zero. Plots can be obtained using a script typified by

```
ux=Table [NodeDisplacements [[n,1]],{n,numnod}];
uy=Table [NodeDisplacements [[n,2]],{n,numnod}];
{uxmax,uymax}=Abs [{Max [ux],Max [uy]}];
ContourPlotNodeFuncOver2DMesh [NodeCoordinates,ElemNodes,ux,
                               uxmax,Nsub,aspect,"Displacement component ux"];
ContourPlotNodeFuncOver2DMesh [NodeCoordinates,ElemNodes,uy,
                               uymax,Nsub,aspect,"Displacement component uy"];
```

The third argument of `ContourPlotNodeFuncOver2DMesh` is the function to be plotted, specified at nodes. A function maximum (in absolute value) is supplied as fourth argument. The reason for supplying this from the outside is that in many cases it is convenient to alter the actual maximum for zooming or animation purposes.

As for the other arguments, *Nsub* is the number of element subdivisions in each direction when breaking down the element area into plot polygons. Typically *Nsub* is set to 4 at the start of the script and is the same for all plots of a script. Plot smoothness in terms of color grading increases with *Nsub*, but plot time grows as *Nsub*-squared. So an appropriate tradeoff is to use a high *Nsub*, say 8, for coarse meshes containing few elements whereas for finer meshes a value of 4 or 2 may work fine. The *aspect* argument specifies the ratio between the *y* (vertical) and *x* (horizontal) dimensions of the plot frame, and is usually defined at the script start as a symbol so it is the same for all plots. The last argument is a plot title.



FIGURE 27.9. Stress component contour plots for rectangular plate with a circular central hole, Model (I) of Figure 27.2(b).

§27.5.3. Stress Field Contour Plots

Contour plots of stress components σ_{xx} , σ_{yy} and σ_{xy} (interpolated over elements from the computed node displacements) may be produced. Stress magnitudes are shown using a internally set color scheme based on “hue interpolation”, in which white means zero. Plots can be obtained using a script typified by

```
sxx=Table[NodePlateStresses[[n,1]],{n,numnod}];
syy=Table[NodePlateStresses[[n,2]],{n,numnod}];
sxy=Table[NodePlateStresses[[n,3]],{n,numnod}];
{sxxmax,syy,sxymax}=Abs[{Max[sxx],Max[syy],Max[sxy]}];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
    sxx,sxxmax,Nsub,aspect,"Nodal stress sig-xx"];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
    syy,syy,sxymax,Nsub,aspect,"Nodal stress sig-yy"];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
    sxy,sxymax,Nsub,aspect,"Nodal stress sig-xy"];
```

For a description of ContourPlotNodeFuncOver2DMesh arguments, see the previous subsection.

As an example, stress component contour plots for Model (I) of the rectangular plate with a circular central hole are shown in Figure 27.9.

Remark 27.1. Sometimes it is useful to show contour plots of principal stresses, Von Mises stresses, maximum shears, etc. To do that, the appropriate function should be constructed first from the Cartesian components available in NodePlateStresses, and then the plot utility called.

§27.5.4. Animation

Occasionally it is useful to animate results such as displacements or stress fields when a load changes as a function of a time-like parameter t .

This can be done by performing a sequence of analyses in a For loop. Such sequence produces a series of plot frames which may be animated by doubly clicking on one of them. The speed of the animation may be controlled by pressing on the rightmost buttons at the bottom of the *Mathematica* window. The second button from the left, if pressed, changes the display sequence to back and forth motion.

```

ClearAll[Em,v,th,aspect,Nsub];
Em=10000; v=.25; th=3; aspect=6/5; Nsub=4;
Emat=Em/(1-v^2)*{{1,v,0},{v,1,0},{0,0,(1-v)/2}};

(* Define FEM model *)

s={1,0.70,0.48,0.30,0.16,0.07,0.0};
xy1={0,6}; xy7={0,1}; xy8={2.5,6}; xy14={Cos[3*Pi/8],Sin[3*Pi/8]};
xy8={2.5,6}; xy21={Cos[Pi/4],Sin[Pi/4]}; xy15={5,6};
xy22={5,2}; xy28={Cos[Pi/8],Sin[Pi/8]}; xy29={5,0}; xy35={1,0};
NodeCoordinates=Table[{0,0},{35}];
Do[NodeCoordinates[[n]]=N[s[[n]] *xy1+(1-s[[n]]) *xy7],{n,1,7}];
Do[NodeCoordinates[[n]]=N[s[[n-7]] *xy8+(1-s[[n-7]]) *xy14],{n,8,14}];
Do[NodeCoordinates[[n]]=N[s[[n-14]]*xy15+(1-s[[n-14]])*xy21],{n,15,21}];
Do[NodeCoordinates[[n]]=N[s[[n-21]]*xy22+(1-s[[n-21]])*xy28],{n,22,28}];
Do[NodeCoordinates[[n]]=N[s[[n-28]]*xy29+(1-s[[n-28]])*xy35],{n,29,35}];
PrintPlaneStressNodeCoordinates[NodeCoordinates,"",{6,4}];
ElemNodes=Table[{0,0,0,0},{24}];
ElemNodes[[1]]=1,2,9,8;
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,2,6}];
ElemNodes[[7]]=ElemNodes[[6]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,8,12}];
ElemNodes[[13]]=ElemNodes[[12]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,14,18}];
ElemNodes[[19]]=ElemNodes[[18]]+{2,2,2,2};
Do [ElemNodes[[e]]=ElemNodes[[e-1]]+{1,1,1,1},{e,20,24}];
numnod=Length[NodeCoordinates]; numele=Length[ElemNodes];

ElemTypes=      Table["Quad4",{numele}];
PrintPlaneStressElementTypeNodes[ElemTypes,ElemNodes,"",{1}];
ElemMaterials=  Table[Emat, {numele}];
ElemFabrications=Table[th, {numele}];
(*PrintPlaneStressElementMatFab[ElemMaterials,ElemFabrications,"",{1}];*)
NodeDOFValues=NodeDOFTags=Table[{0,0},{numnod}];
NodeDOFValues[[1]]=NodeDOFValues[[15]]={0,37.5};
NodeDOFValues[[8]]={0,75}; (* nodal loads *)
Do[NodeDOFTags[[n]]={1,0},{n,1,7}]; (* vroller @ nodes 1-7 *)
Do[NodeDOFTags[[n]]={0,1},{n,29,35}]; (* hroller @ node 4 *)
PrintPlaneStressFreedomActivity[NodeDOFTags,NodeDOFValues,"",{1}];
ProcessOptions={True};
Plot2DElementsAndNodes[NodeCoordinates,ElemNodes,aspect,
  "One element mesh - 4-node quad",True,True];

```

FIGURE 27.10. Preprocessing (model definition) script for problem of Figure 27.2(b).

§27.6. A Complete Problem Script Cell

A complete problem script cell (which is Cell 13 in the plane stress Notebook placed in the web index of this Chapter) is listed in Figures 27.10 through 27.12. This driver cell does Model (I) of the plate with hole problem of Figure 27.2(b), which uses 4-node quadrilateral elements.

The script is divided into 3 parts for convenience. Figure 27.10 lists the model definition script followed by a mesh plot command. Figure 27.11 shows the call to `PlaneStressSolution` analysis driver and the call to get printout of node displacements, forces and stresses. Finally, Figure 27.12 shows commands to produce contour plots of displacements (skipped) and stresses.

Other driver cell examples may be studied in the `PlaneStress.nb` Notebook posted on the course web site. It can be observed that the processing and postprocessing scripts are largely the same.

```
(* Solve problem and print results *)

{NodeDisplacements,NodeForces,NodePlateCounts,NodePlateStresses,
 ElemBarNumbers,ElemBarForces}= PlaneStressSolution[
 NodeCoordinates,ElemTypes,ElemNodes,
 ElemMaterials,ElemFabrications,
 NodeDOFTags,NodeDOFValues,ProcessOptions];

PrintPlaneStressSolution[NodeDisplacements,NodeForces,NodePlateCounts,
 NodePlateStresses,"Computed Solution:",{}];
```

FIGURE 27.11. Processing script and solution print for problem of Figure 27.2(b).

```
(* Plot Displacement Components Distribution - skipped *)

(* ux=Table[NodeDisplacements[[n,1]],{n,numnod}];
 uy=Table[NodeDisplacements[[n,2]],{n,numnod}];
 {uxmax,uymax}=Abs[{Max[ux],Max[uy]}];
 ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,ux,
   uxmax,Nsub,aspect,"Displacement component ux"];
 ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,uy,
   uymax,Nsub,aspect,"Displacement component uy"]; *)

(* Plot Averaged Nodal Stresses Distribution *)

sxx=Table[NodePlateStresses[[n,1]],{n,numnod}];
syy=Table[NodePlateStresses[[n,2]],{n,numnod}];
sxy=Table[NodePlateStresses[[n,3]],{n,numnod}];
{sxxmax,syymax,sxymax}={Max[Abs[sxx]],Max[Abs[syy]],Max[Abs[sxy]]};
Print["sxxmax,syymax,sxymax=",{sxxmax,syymax,sxymax}];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
   sxx,sxxmax,Nsub,aspect,"Nodal stress sig-xx"];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
   syy,syymax,Nsub,aspect,"Nodal stress sig-yy"];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodes,
   sxy,sxymax,Nsub,aspect,"Nodal stress sig-xy"];
```

FIGURE 27.12. Result plotting script for problem of Figure 27.2(b). Note that displacement contour plots have been skipped by commenting out the code.

What changes is the model definition script portion, which often benefits from ad-hoc node and element generation constructs.

Notes and Bibliography

Few FEM books show a complete program. More common is the display of snippets of code. These are left dangling chapter after chapter, with no attempt at coherent interfacing. This historical problem comes from early reliance on low-level languages such as Fortran. Even the simplest program may run to several thousand code lines. At 50 lines/page that becomes difficult to display snugly in a textbook while providing a running commentary.

Another ages-old problem is plotting. Languages such as C or Fortran do not include plotting libraries for the simple reason that low-level universal plotting standards never existed. The situation changed when widely used high-level languages like *Matlab* or *Mathematica* appeared. The language engine provides the necessary fit to available computer hardware, concealing system dependent details. Thus plotting scripts become transportable.

28

Stress Recovery

TABLE OF CONTENTS

	Page
§28.1. Introduction	28-3
§28.2. Calculation of Element Strains and Stresses	28-3
§28.3. Direct Stress Evaluation at Nodes	28-4
§28.4. Extrapolation from Gauss Points	28-4
§28.5. Interelement Averaging	28-6

§28.1. Introduction

In this lecture we study the recovery of stress values for two-dimensional plane-stress elements.¹

This analysis step is sometimes called *postprocessing* because it happens after the main processing step — the calculation of nodal displacements — is completed. Stress calculations are of interest because in structural analysis and design the stresses are often more important to the engineer than displacements.

In the stiffness method of solution discussed in this course, the stresses are obtained from the computed displacements, and are thus *derived quantities*. The accuracy of derived quantities is generally lower than that of primary quantities (the displacements), an informal statement that may be mathematically justified in the theory of finite element methods. For example, if the accuracy level of displacements is 1% that of the stresses may be typically 10% to 20%, and even lower at boundaries.

It is therefore of interest to develop techniques that enhance the accuracy of the computed stresses. The goal is to “squeeze” as much accuracy from the computed displacements while keeping the computational effort reasonable. These procedures receive the generic name *stress recovery techniques* in the finite element literature. In the following sections we cover the simplest stress recovery techniques that have been found most useful in practice.

§28.2. Calculation of Element Strains and Stresses

In elastic materials, stresses σ are directly related to strains \mathbf{e} at each point through the elastic constitutive equations $\sigma = \mathbf{E}\mathbf{e}$. It follows that the stress computation procedure begins with strain computations, and that the accuracy of stresses depends on that of strains. Strains, however, are seldom saved or printed.

In the following we focus our attention on two-dimensional isoparametric elements, as the computation of strains, stresses and axial forces in bar elements is straightforward.

Suppose that we have solved the master stiffness equations

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \quad (28.1)$$

for the node displacements \mathbf{u} . To calculate strains and stresses we perform a loop over all defined elements. Let e be the element index of a specific two-dimensional isoparametric element encountered during this loop, and $\mathbf{u}^{(e)}$ the vector of computed element node displacements. Recall from §15.3 that the strains at any point in the element may be related to these displacements as

$$\mathbf{e} = \mathbf{B}\mathbf{u}^{(e)}, \quad (28.2)$$

where \mathbf{B} is the strain-displacement matrix (14.18) assembled with the x and y derivatives of the element shape functions evaluated at the point where we are calculating strains. The corresponding stresses are given by

$$\sigma = \mathbf{E}\mathbf{e} = \mathbf{E}\mathbf{B}\mathbf{u} \quad (28.3)$$

¹ This Chapter needs rewriting to show the use of Mathematica for stress computation. To be done in the future.

Table 28.1 Natural Coordinates of Bilinear Quadrilateral Nodes

Corner node	ξ	η	ξ'	η'	Gauss node	ξ	η	ξ'	η'
1	-1	-1	$-\sqrt{3}$	$-\sqrt{3}$	1'	$-1/\sqrt{3}$	$-1/\sqrt{3}$	-1	-1
2	+1	-1	$+\sqrt{3}$	$-\sqrt{3}$	2'	$+1/\sqrt{3}$	$-1/\sqrt{3}$	+1	-1
3	+1	+1	$+\sqrt{3}$	$+\sqrt{3}$	3'	$+1/\sqrt{3}$	$+1/\sqrt{3}$	+1	+1
4	-1	+1	$-\sqrt{3}$	$+\sqrt{3}$	4'	$-1/\sqrt{3}$	$+1/\sqrt{3}$	-1	+1
Gauss nodes, and coordinates ξ' and η' are defined in §28.4 and Fig. 28.1									

In the applications it is of interest to evaluate and report these stresses at the *element nodal points* located on the corners and possibly midpoints of the element. These are called *element nodal point stresses*.

It is important to realize that the stresses computed at the same nodal point from adjacent elements *will not generally be the same*, since stresses are not required to be continuous in displacement-assumed finite elements. This suggests some form of stress averaging can be used to improve the stress accuracy, and indeed this is part of the stress recovery technique further discussed in §28.5. The results from this averaging procedure are called *nodal point stresses*.

For the moment let us see how we can proceed to compute element nodal stresses. Two approaches are followed in practice:

1. Evaluate directly σ at the element node locations by substituting the natural coordinates of the nodal points as arguments to the shape function modules. These modules return \mathbf{q}_x and \mathbf{q}_y and direct application of (28.2)-(28.4) yields the strains and stresses at the nodes.
2. Evaluate σ at the Gauss integration points used in the element stiffness integration rule and then extrapolate to the element node points.

Empirical evidence indicates that the second approach generally delivers better stress values for *quadrilateral* elements whose geometry departs substantially from the rectangular shape. This is backed up by “superconvergence” results in finite element approximation theory. For rectangular elements there is no difference.

For isoparametric *triangles* both techniques deliver similar results (identical if the elements are straight sided with midside nodes at midpoints) and so the advantages of the second one are marginal. Both approaches are covered in the sequel.

§28.3. Direct Stress Evaluation at Nodes

This approach is straightforward and need not be discussed in detail.

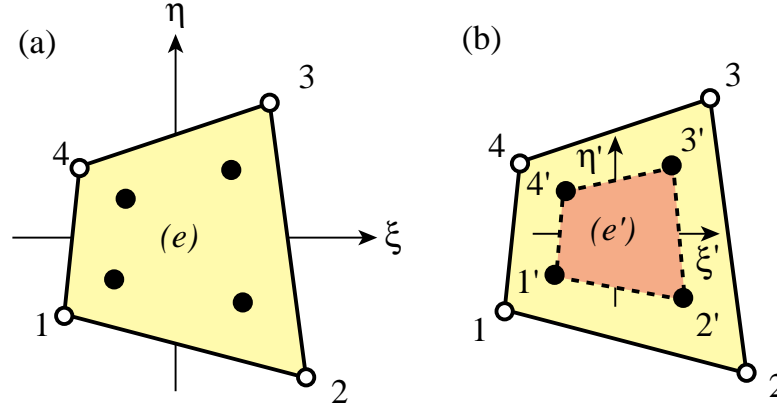


Figure 28.1. Extrapolation from 4-node quad Gauss points: (a) 2×2 rule, (b) Gauss element (e')

§28.4. Extrapolation from Gauss Points

This will again be explained for the four-node bilinear quadrilateral. The normal Gauss integration rule for element stiffness evaluation is 2×2 , as illustrated in Figure 28.1.

The stresses are calculated at the Gauss points, which are identified as $1'$, $2'$, $3'$ and $4'$ in Figure 28.1. Point i' is closest to node i so it is seen that Gauss point numbering essentially follows element node numbering in the counterclockwise sense. The natural coordinates of these points are listed in Table 28.1. The stresses are evaluated at these Gauss points by passing these natural coordinates to the shape function subroutine. Then each stress component is “carried” to the corner nodes 1 through 4 through a bilinear extrapolation based on the computed values at $1'$ through $4'$.

To understand the extrapolation procedure more clearly it is convenient to consider the region bounded by the Gauss points as an “internal element” or “Gauss element”. This interpretation is depicted in Figure 28.1(b). The Gauss element, denoted by (e'), is also a four-node quadrilateral. Its quadrilateral (natural) coordinates are denoted by ξ' and η' . These are linked to ξ and η by the simple relations

$$\xi = \xi'/\sqrt{3}, \quad \eta = \eta'/\sqrt{3}, \quad \xi' = \xi\sqrt{3}, \quad \eta' = \eta\sqrt{3}. \quad (28.4)$$

Any scalar quantity w whose values w'_i at the Gauss element corners are known can be interpolated through the usual bilinear shape functions now expressed in terms of ξ' and η' :

$$w(\xi', \eta') = [w'_1 \quad w'_2 \quad w'_3 \quad w'_4] \begin{bmatrix} N_1^{(e')} \\ N_2^{(e')} \\ N_3^{(e')} \\ N_4^{(e')} \end{bmatrix}, \quad (28.5)$$

where (cf. §15.6.2)

$$\begin{aligned} N_1^{(e')} &= \frac{1}{4}(1 - \xi')(1 - \eta'), \\ N_2^{(e')} &= \frac{1}{4}(1 + \xi')(1 - \eta'), \\ N_3^{(e')} &= \frac{1}{4}(1 + \xi')(1 + \eta'), \\ N_4^{(e')} &= \frac{1}{4}(1 - \xi')(1 + \eta'). \end{aligned} \quad (28.6)$$

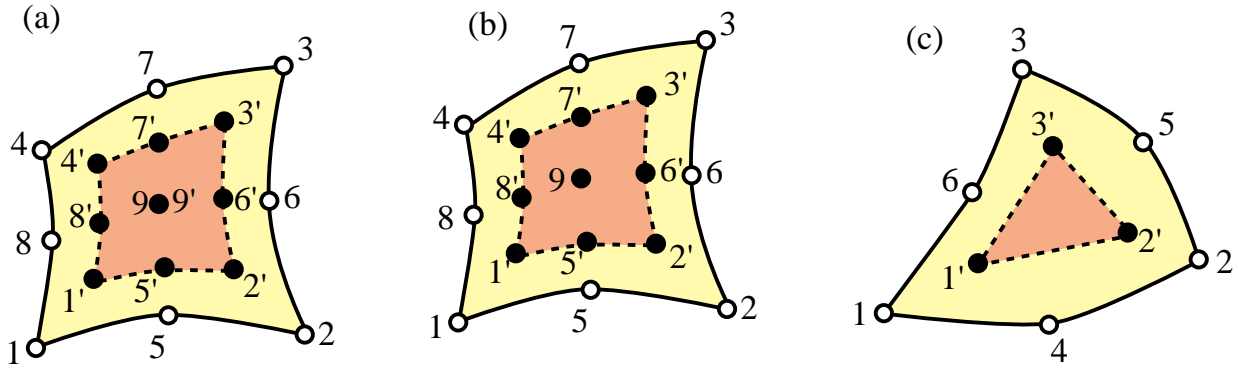


Figure 28.2. Gauss elements for higher order quadrilaterals and triangles:
 (a) 9-node element with 3×3 Gauss rule, (b) 8-node element with 3×3 Gauss rule, (c) 6-node element with 3-interior point rule.

To extrapolate w to corner 1, say, we replace its ξ' and η' coordinates, namely $\xi' = \eta' = -\sqrt{3}$, into the above formula. Doing that for the four corners we obtain

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} \\ 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} & -\frac{1}{2} \\ -\frac{1}{2} & 1 - \frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1 + \frac{1}{2}\sqrt{3} \end{bmatrix} \begin{bmatrix} w'_1 \\ w'_2 \\ w'_3 \\ w'_4 \end{bmatrix} \quad (28.7)$$

Note that the sum of the coefficients in each row is one, as it should be. For stresses we apply this formula taking w to be each of the three stress components, σ_{xx} , σ_{yy} and τ_{xy} , in turn.

Extrapolation in Higher Order Elements

For eight-node and nine-node isoparametric quadrilaterals the usual Gauss integration rule is 3×3 , and the Gauss elements are nine-noded quadrilaterals that look as in Figure 28.2(a) and (b) above. For six-node triangles the usual quadrature is the 3-point rule with internal sampling points, and the Gauss element is a three-node triangle as shown in Figure 28.2(c).

§28.5. Interelement Averaging

The stresses computed in element-by-element fashion as discussed above, whether by direct evaluation at the nodes or by extrapolation, will generally exhibit jumps between elements. For printing and plotting purposes it is usually convenient to “smooth out” those jumps by computing *averaged nodal stresses*. This averaging may be done in two ways:

- (I) Unweighted averaging: assign same weight to all elements that meet at a node;
- (II) Weighted averaging: the weight assigned to element contributions depends on the stress component and the element geometry and possibly the element type.

Several weighted average schemes have been proposed in the finite element literature, but they do require additional programming.

30

Thermomechanical Effects

TABLE OF CONTENTS

	Page
§30.1. Introduction	30–3
§30.2. Thermomechanical Behavior	30–3
§30.2.1. Thermomechanical Stiffness Relations	30–4
§30.2.2. Globalization	30–6
§30.2.3. Merge	30–6
§30.2.4. Solution	30–6
§30.2.5. Postprocessing	30–6
§30.2.6. Worked-Out Examples	30–7
§30. Notes and Bibliography	30–10
§30. References	30–10
§30. Exercises	30–11

§30.1. Introduction

The assumptions invoked in Chapters 2–3 for the example truss result in zero external forces under zero displacements. This is implicit in the linear-homogeneous expression of the master stiffness equation $\mathbf{f} = \mathbf{K}\mathbf{u}$. If \mathbf{u} vanishes, so does \mathbf{f} . This behavior does not apply, however, if there are *initial force effects*.¹ If those effects are present, there can be displacements without external forces, and internal forces without displacements.

A common source of initial force effects are temperature changes. Imagine that a plane truss structure is *unloaded* (that is, not subjected to external forces) and is held at a *uniform reference temperature*. External displacements are measured from this environment, which is technically called a *reference state*. Now suppose that the temperature of some members changes with respect to the reference temperature while the applied external forces remain zero. Because the length of members changes on account of thermal expansion or contraction, the joints will displace. If the structure is statically indeterminate those displacements will induce strains and stresses and thus internal forces. These are distinguished from mechanical effects by the qualifier “thermal.” For many structures, particularly in aerospace and mechanical engineering, such effects have to be considered in the analysis and design.

There are other physical sources of initial force effects, such as moisture (hygrosteric) effects,² member prestress, residual stresses, or lack of fit. For linear structural models *all* such sources may be algebraically treated in the same way as thermal effects. The treatment results in an *initial force* vector that has to be added to the applied mechanical forces. This subject is outlined in §30.3 from a general perspective. However, to describe the main features of the matrix analysis procedure it is sufficient to consider the case of temperature changes.

In this Section we go over the analysis of a plane truss structure whose members undergo temperature changes from a reference state. It is assumed that the disconnection and localization steps of the DSM have been carried out. Therefore we begin with the derivation of the matrix stiffness equations of a generic truss member.

§30.2. Thermomechanical Behavior

Consider the generic plane-truss member shown in Figure 30.1. The member is prismatic and uniform. The temperature T is also uniform. To reduce clutter the member identification subscript will be omitted in the following development until the globalization and assembly steps.

We introduce the concept of *reference temperature* T_{ref} . This is conventionally chosen to be the temperature throughout the structure at which the displacements, strains and stresses are zero if no mechanical forces are applied. In structures such as buildings and bridges T_{ref} is often taken to be the mean temperature during the construction period. Those zero displacements, strains and stresses, together with T_{ref} , define the *thermomechanical reference state* for the structure.

The member temperature variation from that reference state is $\Delta T = T - T_{ref}$. This may be positive or negative. If the member is *disassembled* or *disconnected*, under this variation the member length

¹ Called *initial stress* or *initial strain* effects by many authors. The different names reflect what is viewed as the physical source of initial force effects at the continuum mechanics level.

² These are important in composite materials and geomechanics.

is free to change from L to $L + d_T$. If the thermoelastic constitutive behavior is linear³ then d_T is proportional to L and ΔT :

$$d_T = \alpha L \Delta T. \quad (30.1)$$

Here α is the coefficient of thermal expansion, which has physical units of one over temperature. This coefficient will be assumed to be uniform over the generic member. It may, however, vary from member to member.

The *thermal strain* is defined as

$$e_T = d_T/L = \alpha \Delta T. \quad (30.2)$$

Now suppose that the member is also subject to mechanical forces, more precisely the applied axial force F shown in Figure 30.1. The member axial stress is $\sigma = F/A$. In response to this stress the length changes by d_M . The *mechanical strain* is $e_M = d_M/L$. The total strain $e = d/L = (d_M + d_T)/L$ is the sum of the mechanical and the thermal strains:

$$e = e_M + e_T = \frac{\sigma}{E} + \alpha \Delta T \quad (30.3)$$

This superposition of deformations is the basic assumption made in the thermomechanical analysis. It is physically obvious for an unconstrained member such as that depicted in Figure 30.1.

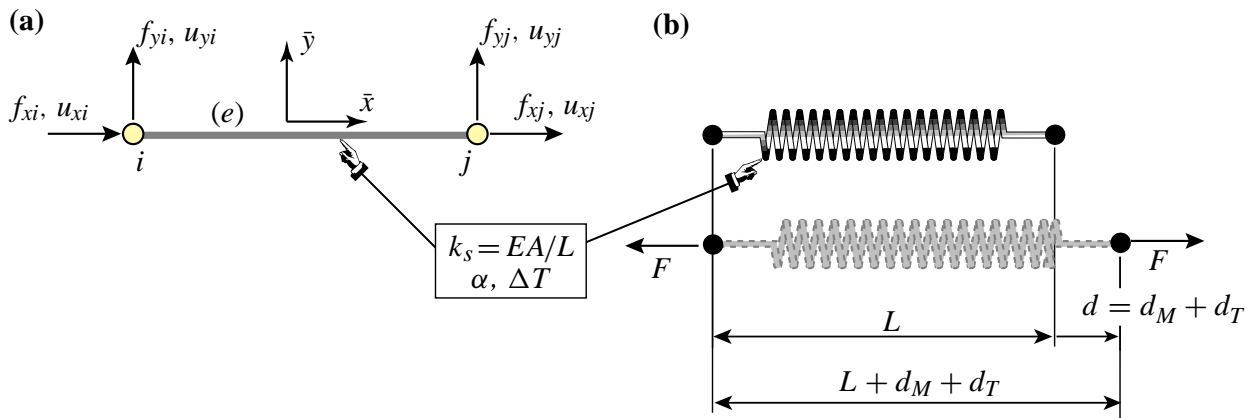


FIGURE 30.1. Generic truss member subjected to mechanical and thermal effects:
(a) idealization as bar, (b) idealization as equivalent linear spring.

At the other extreme, suppose that the member is completely blocked against axial elongation; that is, $d = 0$ but $\Delta T \neq 0$. Then $e = 0$ and $e_M = -e_T$. If $\alpha > 0$ and $\Delta T > 0$ the blocked member goes in *compression* because $\sigma = Ee_M = -Ee_T = -E\alpha \Delta T < 0$. This *thermal stress* is further discussed in Remark 30.2.

³ An assumption justified if the temperature changes are small enough so that α is approximately constant through the range of interest, and no material phase change effects occur.

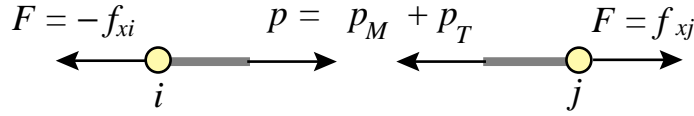


FIGURE 30.2. Equilibrium of truss member under thermomechanical forces.

§30.2.1. Thermomechanical Stiffness Relations

Because $e = d/L$ and $d = \bar{u}_{xj} - \bar{u}_{xi}$, (30.3) can be developed as

$$\frac{\bar{u}_{xj} - \bar{u}_{xi}}{L} = \frac{\sigma}{E} + \alpha \Delta T, \quad (30.4)$$

To pass to internal forces (30.4) is multiplied through by EA :

$$\frac{EA}{L}(\bar{u}_{xj} - \bar{u}_{xi}) = A\sigma + EA\alpha\Delta T = p_M + p_T = p = F. \quad (30.5)$$

Here $p_M = A\sigma$ denotes the mechanical axial force, and $p_T = EA\alpha\Delta T$, which has the dimension of a force, is called (not surprisingly) the *internal thermal force*. The sum $p = p_M + p_T$ is called the *effective internal force*. The last relation in (30.5), $F = p = p_M + p_T$ follows from free-body member equilibrium; see Figure 30.2. Passing to matrix form:

$$F = \frac{EA}{L} \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix}. \quad (30.6)$$

Noting that $F = \bar{f}_{xj} = -\bar{f}_{xi}$ while $\bar{f}_{yi} = \bar{f}_{yj} = 0$, we can relate joint forces to joint displacements as

$$\begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} -F \\ 0 \\ F \\ 0 \end{bmatrix} = \begin{bmatrix} \bar{f}_{Mxi} \\ \bar{f}_{Myi} \\ \bar{f}_{Mxj} \\ \bar{f}_{Myj} \end{bmatrix} + EA\alpha\Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix}. \quad (30.7)$$

In compact matrix form this is $\bar{\mathbf{f}} = \bar{\mathbf{f}}_M + \bar{\mathbf{f}}_T = \bar{\mathbf{K}}\bar{\mathbf{u}}$, or

$$\boxed{\bar{\mathbf{K}}\bar{\mathbf{u}} = \bar{\mathbf{f}}_M + \bar{\mathbf{f}}_T.} \quad (30.8)$$

Here $\bar{\mathbf{K}}$ is the same member stiffness matrix derived in Chapter 2. The new ingredient that appears is the vector

$$\bar{\mathbf{f}}_T = EA\alpha\Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad (30.9)$$

This is called the vector of *thermal joint forces* in local coordinates. It is an instance of an *initial force* vector at the element level.

Remark 30.1. A useful physical interpretation of (30.8) is as follows. Suppose that the member is precluded from joint (node) motions so that $\bar{\mathbf{u}} = \mathbf{0}$. Then $\bar{\mathbf{f}}_M + \bar{\mathbf{f}}_T = \mathbf{0}$ or $\bar{\mathbf{f}}_M = -\bar{\mathbf{f}}_T$. It follows that \mathbf{f}_T contains the negated joint forces (internal forces) that develop in a heated or cooled bar if joint motions are precluded. Because for most materials $\alpha > 0$, rising the temperature of a blocked bar: $\Delta T > 0$, produces an internal compressive thermal force $p_T = A\sigma_T = -EA\alpha T$, in accordance with the expected physics. The quantity $\sigma_T = -E\alpha \Delta T$ is the *thermal stress*. This stress can cause buckling or cracking in severely heated structural members that are not allowed to expand or contract. This motivates the use of expansion joints in pavements, buildings and rails, and roller supports in long bridges.

§30.2.2. Globalization

At this point we restore the member superscript so that the member stiffness equations (30.7) are rewritten as

$$\bar{\mathbf{K}}^e \bar{\mathbf{u}}^e = \bar{\mathbf{f}}_M^e + \bar{\mathbf{f}}_T^e. \quad (30.10)$$

Use of the transformation rules developed in Chapter 2 to change displacements and forces to the global system $\{x, y\}$ yields

$$\mathbf{K}^e \mathbf{u}^e = \mathbf{f}_M^e + \mathbf{f}_T^e, \quad (30.11)$$

where \mathbf{T}^e is the displacement transformation matrix (3.1), and the transformed quantities are

$$\mathbf{K}^e = (\mathbf{T}^e)^T \bar{\mathbf{K}}^e \mathbf{T}^e, \quad \mathbf{f}_M^e = (\mathbf{T}^e)^T \bar{\mathbf{f}}_M^e, \quad \mathbf{f}_T^e = (\mathbf{T}^e)^T \bar{\mathbf{f}}_T^e. \quad (30.12)$$

These globalized member equations are used to assemble the free-free master stiffness equations by a member merging process.

§30.2.3. Merge

The merge process is based on the same assembly rules stated in §3.1.3 with only one difference: thermal forces are added to the right hand side. The member by member merge is carried out much as described as in §3.1.4, the main difference being that the thermal force vectors \mathbf{f}_T^e are also merged into a master thermal force vector. Force merge can be done by augmentation-and-add (for hand work) or via freedom pointers (for computer work). Illustrative examples are provided below. Upon completion of the assembly process we arrive at the free-free master stiffness equations

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_T = \mathbf{f}.} \quad (30.13)$$

§30.2.4. Solution

The master system (30.13) has formally the same configuration as the master stiffness equations (2.3). The only difference is that the *effective* joint force vector \mathbf{f} contains a superposition of mechanical and thermal forces. Displacement boundary conditions can be applied by reduction or modification of these equations, simply by using effective joint forces in the descriptions of Chapters 2–3. Processing the reduced or modified system by a linear equation solver yields the displacement solution \mathbf{u} .

§30.2.5. Postprocessing

The postprocessing steps described in Chapter 3 require some modifications because the derived quantities of interest to the structural engineer are *mechanical* reaction forces and internal forces. Effective forces by themselves are of little use in design. Mechanical joint forces including reactions are recovered from

$$\mathbf{f}_M = \mathbf{K}\mathbf{u} - \mathbf{f}_T \quad (30.14)$$

To recover mechanical internal forces in member e , compute p^e by the procedure outlined in Chapter 3, and subtract the thermal component:

$$p_M^e = p^e - E^e A^e \alpha^e \Delta T^e. \quad (30.15)$$

This equation comes from solving (30.5) for p_M . The mechanical axial stress is $\sigma^e = p_M^e / A^e$.

§30.2.6. Worked-Out Examples

Example 30.1. The first worked out problem is defined in Figure 30.3. Two truss members are connected in series as shown and fixed at the ends. Properties $E = 1000$, $A = 5$ and $\alpha = 0.0005$ are common to both members. The member lengths are 4 and 6. A mechanical load $P = 90$ acts on the roller node. The temperature of member (1) increases by $\Delta T^{(1)} = 25^\circ$ while that of member (2) drops by $\Delta T^{(2)} = -10^\circ$. Find the stress in both members.

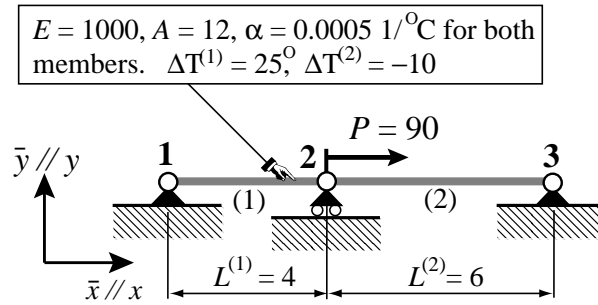


FIGURE 30.3. Structure for worked-out Example 1.

To reduce clutter note that all y motions are suppressed so only the x freedoms are kept: $u_{x1} = u_1$, $u_{x2} = u_2$ and $u_{x3} = u_3$. The corresponding node forces are denoted by $f_{x1} = f_1$, $f_{x2} = f_2$ and $f_{x3} = f_3$. The thermal force vectors, stripped to their $\bar{x} \equiv x$ components, are

$$\bar{\mathbf{f}}_T^{(1)} = \begin{bmatrix} \bar{f}_{T1}^{(1)} \\ \bar{f}_{T2}^{(1)} \end{bmatrix} = E^{(1)} A^{(1)} \alpha^{(1)} \Delta T^{(1)} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -150 \\ 150 \end{bmatrix}, \quad \bar{\mathbf{f}}_T^{(2)} = \begin{bmatrix} \bar{f}_{T2}^{(2)} \\ \bar{f}_{T3}^{(2)} \end{bmatrix} = E^{(2)} A^{(2)} \alpha^{(2)} \Delta T^{(2)} \begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 60 \\ -60 \end{bmatrix}. \quad (30.16)$$

The element stiffness equations are:

$$3000 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_1^{(1)} \\ \bar{u}_2^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{f}_{M1}^{(1)} \\ \bar{f}_{M2}^{(1)} \end{bmatrix} + \begin{bmatrix} -150 \\ 150 \end{bmatrix}, \quad 2000 \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_2^{(2)} \\ \bar{u}_3^{(2)} \end{bmatrix} = \begin{bmatrix} \bar{f}_{M2}^{(2)} \\ \bar{f}_{M3}^{(2)} \end{bmatrix} + \begin{bmatrix} 60 \\ -60 \end{bmatrix}, \quad (30.17)$$

No globalization is needed because the equations are already in the global system, and thus we can get rid of the localization marker symbols: $\bar{f} \rightarrow f$, $\bar{u} \rightarrow u$. Assembling by any method yields

$$1000 \begin{bmatrix} 3 & -3 & 0 \\ -3 & 5 & -2 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} f_{M1} \\ f_{M2} \\ f_{M3} \end{bmatrix} + \begin{bmatrix} -150 \\ 150 + 60 \\ -60 \end{bmatrix} = \begin{bmatrix} f_{M1} \\ f_{M2} \\ f_{M3} \end{bmatrix} + \begin{bmatrix} -150 \\ 210 \\ -60 \end{bmatrix}. \quad (30.18)$$

The displacement boundary conditions are $u_1 = u_3 = 0$. The mechanical force boundary condition is $f_{M2} = 90$. On removing the first and third equations, the reduced system is $5000 u_2 = f_{M2} + 210 = 90 + 210 = 300$, which yields $u_2 = 300/5000 = +0.06$. The mechanical internal forces in the members are recovered from

$$\begin{aligned} p_M^{(1)} &= \frac{E^{(1)} A^{(1)}}{L^{(1)}} (u_2 - u_1) - E^{(1)} A^{(1)} \alpha^{(1)} \Delta T^{(1)} = 3000 \times 0.06 - 12000 \times 0.0005 \times 25 = 60, \\ p_M^{(2)} &= \frac{E^{(2)} A^{(2)}}{L^{(2)}} (u_3 - u_2) - E^{(2)} A^{(2)} \alpha^{(2)} \Delta T^{(2)} = 2000 \times (-0.06) - 12000 \times 0.0005 \times (-10) = -72, \end{aligned} \quad (30.19)$$

whence the stresses are $\sigma^{(1)} = 60/12 = 5$ and $\sigma^{(2)} = -72/12 = -6$. Member (1) is in tension and member (2) in compression.

Example 30.2. The second example concerns the example truss of Chapters 2-3. The truss is mechanically *unloaded*, that is, $f_{Mx2} = f_{Mx3} = f_{My3} = 0$. However the temperature of members (1) (2) and (3) changes by ΔT , $-\Delta T$ and $3\Delta T$, respectively, with respect to T_{ref} . The thermal expansion coefficient of all three members is assumed to be α . We will perform the analysis keeping α and ΔT as variables.

The thermal forces for each member in global coordinates are obtained by using (30.10) and the third of (30.12):

$$\begin{aligned} \mathbf{f}_T^{(1)} &= E^{(1)} A^{(1)} \alpha^{(1)} \Delta T^{(1)} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 100\alpha \Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \\ \mathbf{f}_T^{(2)} &= E^{(2)} A^{(2)} \alpha^{(2)} \Delta T^{(2)} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 50\alpha \Delta T \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \\ \mathbf{f}_T^{(3)} &= E^{(3)} A^{(3)} \alpha^{(3)} \Delta T^{(3)} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 200\alpha \Delta T \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}. \end{aligned} \quad (30.20)$$

Merging the contribution of these 3 members gives the master thermal force vector

$$\mathbf{f}_T = \alpha \Delta T \begin{bmatrix} -100 + 0 - 200 \\ 0 + 0 - 200 \\ 100 + 0 + 0 \\ 0 - 50 + 0 \\ 0 + 0 + 200 \\ 0 + 50 + 200 \end{bmatrix} = \alpha \Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix} \quad (30.21)$$

The master stiffness matrix \mathbf{K} does not change. Consequently the master stiffness equations are

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} = 0 \\ u_{y1} = 0 \\ u_{x2} \\ u_{y2} = 0 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{Mx1} \\ f_{My1} \\ f_{Mx2} = 0 \\ f_{My2} \\ f_{Mx3} = 0 \\ f_{My3} = 0 \end{bmatrix} + \alpha \Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix} \quad (30.22)$$

in which f_{Mx1} , f_{My1} and f_{My2} are the unknown mechanical reaction forces, and the known forces and displacements have been marked. Since the prescribed displacements are zero, the reduced system is simply

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \alpha \Delta T \begin{bmatrix} 100 \\ 200 \\ 250 \end{bmatrix} = \alpha \Delta T \begin{bmatrix} 100 \\ 200 \\ 250 \end{bmatrix}. \quad (30.23)$$

Solving (30.23) gives $u_{x2} = u_{x3} = u_{y3} = 10\alpha \Delta T$. Completing \mathbf{u} with the prescribed zero displacements and premultiplying by \mathbf{K} gives the complete effective force vector:

$$\mathbf{f} = \mathbf{K}\mathbf{u} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 10 \\ 10 \end{bmatrix} \alpha \Delta T = \alpha \Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix}. \quad (30.24)$$

But this vector is exactly \mathbf{f}_T . Consequently

$$\mathbf{f}_M = \mathbf{K}\mathbf{u} - \mathbf{f}_T = \mathbf{0}. \quad (30.25)$$

All mechanical joint forces, including reactions, vanish, and so do the internal mechanical forces. This is a consequence of the example frame being statically determinate.⁴

Initial Force Effects

As previously noted, a wide spectrum of mechanical and non-mechanical effects can be accommodated under the umbrella of the *initial force* concept. The stiffness equations at the local (member) level are

$$\boxed{\tilde{\mathbf{K}}^e \tilde{\mathbf{u}}^e = \tilde{\mathbf{f}}_M^e + \tilde{\mathbf{f}}_I^e = \tilde{\mathbf{f}}^e}, \quad (30.26)$$

and at the global (assembled structure) level:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I = \mathbf{f}}. \quad (30.27)$$

In these equations subscripts M and I identify mechanical and initial node forces, respectively. The sum of the two: $\tilde{\mathbf{f}}$ at the local member level and \mathbf{f} at the global structure level, are called *effective* forces.

A physical interpretation of (30.26) can be obtained by considering that the structure is blocked against all motions: $\mathbf{u} = \mathbf{0}$. Then $\mathbf{f}_M = -\mathbf{f}_I$, and the undeformed structure experiences mechanical forces. These translate into internal forces and stresses. Engineers also call these *prestresses*. ‘prestress

Local effects that lead to initial forces at the member level are: temperature changes (studied in §4.2, in which $\mathbf{f}_I \equiv \mathbf{f}_T$), moisture diffusion, residual stresses, lack of fit in fabrication, and in-member prestressing. Global effects include prescribed nonzero joint displacements (studied in §4.1) and multimember prestressing (for example, by cable pretensioning of concrete structures).

As can be seen there is a wide variety of physical effects, whether natural or artificial, that lead to nonzero initial forces. The good news is that once the member equations (30.25) are formulated, the remaining DSM steps (globalization, merge and solution) are identical. This nice property extends to the general Finite Element Method.

⁴ For the definition of static determinacy, see any textbook on Mechanics of Materials. Such structures *do not develop thermal stresses under any combination of temperature changes*.

Pseudo Thermal Inputs

Some commercial FEM programs do not have a way to handle directly effects such as moisture, lack of fit, or prestress. But all of them can handle temperature variation inputs. Since in linear analysis all such effects can be treated as initial forces, it is possible (at least for bar elements) to model them as fictitious thermomechanical effects, by inputting phony temperature changes. The following example indicate that this is done for a bar element.

Suppose that a prestress force F_p is present in a bar. The total elongation is $d = d_M + d_p$ where $d_p = F_p L / (EA)$ is due to prestress. Equate to a thermal elongation: $d_T = \alpha \Delta T_p L$ and solve for $\Delta T_p = F_p / (EA\alpha)$. This is input to the program as a fictitious temperature change. If in addition there is a real temperature change ΔT one would of course specify $\Delta T + \Delta T_p$.

If this device is used, care should be exercised in interpreting results for internal forces and stresses given by the program. The trick is not necessary for personal or open-source codes over which you have full control.

Notes and Bibliography

The additional DSM topics treated in this Chapter are covered in virtually all books on Matrix Structural Analysis, such as the often quoted one by Przemieniecki [575]. Several recent FEM books ignore these topics as too elementary.

The physics of thermomechanics and the analysis of thermal stresses is covered adequately in textbooks such as Boley and Wiener [95], or manuals such as the widely used by Roark et. al. [620].

For the separate problems of heat conduction and heat transfer, the book by Özişik [516] provides a comprehensive classic treatment. There is a vast literature on prestressed structures; search under “prestress” in <http://www3.addall.com>.

The concepts of static determinacy and its counterpart: static indeterminacy, are important in skeletal structures such as trusses and frameworks. The pertinent design tradeoff is: insensitivity to initial force effects versus redundant safety. A discussion of this topic is beyond the scope of the book. Once going past skeletal structural systems, however, indeterminacy is the rule.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 30

Thermomechanical Effects

EXERCISE 30.1 [N:20] Use the same data of Exercise 3.7, except that $P = 0$ and consequently there are no applied mechanical forces. Both members have the same dilatation coefficient $\alpha = 10^{-6} \text{ 1/}^\circ\text{F}$. Find the crown displacements u_{x2} and u_{y2} and the member stresses $\sigma^{(1)}$ and $\sigma^{(2)}$ if the temperature of member (1) rises by $\Delta T = 120^\circ\text{F}$ above T_{ref} , whereas member (2) stays at T_{ref} .

Shortcut: the element stiffnesses and master stiffness matrix are the same as in Exercise 3.7, so if that Exercise has been previously assigned no stiffness recomputations are necessary.

EXERCISE 30.2 [A:15] Consider the generic truss member of §2.4, reproduced in Figure E30.1 for convenience.

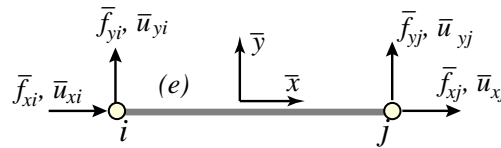


FIGURE E30.1. Generic truss member.

The disconnected member was supposed to have length L , but because of lack of quality control it was fabricated with length $L + \delta$, where δ is called the “lack of fit.” Determine the initial force vector $\bar{\mathbf{f}}_I$ to be used in (30.26). *Hint*: find the mechanical forces that would compensate for δ and restore the desired length.

EXERCISE 30.3 [A:10] Show that the lack of fit of the foregoing exercise can be viewed as equivalent to a prestress force of $-(EA/L)\delta$.

EXERCISE 30.4 [A:20] Show that prescribed nonzero displacements can, albeit somewhat artificially, be placed under the umbrella of initial force effects. Work this out for the example of §30.1.1. *Hint*: split node displacements into $\mathbf{u} = \mathbf{u}_H + \mathbf{u}_N$, where \mathbf{u}_N (the “nonhomogeneous” or “particular” part of the solution) carries the nonzero displacement values.

EXERCISE 30.5 [A:35]. (Research paper level). Prove that any statically determinate truss structure is free of thermal stresses.

31

Dynamics & Vibration Overview

TABLE OF CONTENTS

	Page
§31.1. Introduction	31-3
§31.2. Semidiscrete Equations of Motion	31-3
§31.2.1. Vibrations as Equilibrium Disturbance	31-4
§31.2.2. Undamped Free Vibrations	31-5
§31.2.3. The Vibration Eigenproblem	31-6
§31.2.4. Eigensystem Properties	31-6
§31.3. Solving The Vibration Eigenproblem	31-7
§31.3.1. Determinant Roots	31-7
§31.3.2. Reduction to the Standard Eigenproblem	31-7
§31.3.3. Unsymmetric Reduction	31-7
§31.3.4. Symmetry Preserving Reduction	31-8
§31. Notes and Bibliography	31-8
§31. References	31-9
§31. Exercises	31-10

§31.1. Introduction

The development in previous chapters pertain to static analysis, in which all quantities are independent of time. This kind of analysis applies also to *quasi-static* scenarios, where the state varies with time but does so slowly that inertial and damping effects can be ignored. For example one may imagine situations such as a roof progressively burdened by falling snow before collapse, the filling of a dam, or the construction of a tunnel. Or foundation settlements: think of the Pisa tower before leaning stopped. The quasi-static assumption is commonly used in design even for loads that vary in a faster time scale. For example, vehicles travelling over a bridge or wind effects on buildings.¹

By contrast *dynamic analysis* is appropriate when the variation of displacements with time is so rapid that inertial effects cannot be ignored. There are numerous practical examples: earthquakes, rocket launches, vehicle crashes, explosive forming, air blasts, underground explosions, rotating machinery, airplane flutter. The structural accelerations, which are second derivatives with respect to time, must be kept in the governing equations. Damping effects, which are associated with velocities (the first temporal derivatives of displacements), may be also included. However, *passive damping* effects are often neglected as they tend to take energy out of a system and thus reduce the response amplitude.

Dynamic analysis may be performed in the time domain or the frequency domain. The latter is restricted in scope in that it applies to *linear* structural models, or to linearized fluctuations about an equilibrium state. The frequency domain embodies naturally the analysis of free vibrations, which is the focus of the present Chapter.

Remark 31.1. Mathematically, a dynamical system consists of a phase space together with an evolution law. (J.C. Yoccoz). A major goal of the theory is to understand the long term behaviour of the system.

§31.2. Semidiscrete Equations of Motion

The essence of structural analysis is mastering forces. In the development of FEM, this was understood by the pioneers of the first generation, as narrated in §1.7.1. With the victory of the Direct Stiffness Method (DSM) by 1970, displacements came to the foreground as primary computational variables because they scale well into complicated systems.

To understand *dynamic* analysis, that dual role must be kept in mind. Displacements become even more important as computational variables. After all, velocities and accelerations are temporal derivatives of displacements. There is no easy way to do the job with forces only, since dynamics is about *motion*. On the other hand, the fundamental governing equations of structural dynamics are *force balance* statements. They are elaborate versions of Newtonian mechanics.

This Newtonian viewpoint is illustrated in Table 31.1 for several modeling scenarios that span statics, dynamics and vibrations. For notational simplicity it is assumed that the structure has been discretized in space, for example by the FEM. The right column shows the vector form of the governing equations as *force balance* statements. The table defines nomenclature.

¹ The quasi-static assumption can be done during design if dynamic effects can be accounted for through appropriate safety factors. For many types of structures (e.g., buildings, bridges, offshore towers) these are specified in building codes. This saves time when dynamic effects are inherently nondeterministic, as in traffic, winds or wave effects.

Table 31.1. Discrete Structural Mechanics Expressed as Force Balance Statements

<i>Case Problem type</i>	<i>Governing force balance equations</i>
I General nonlinear dynamics	$\underbrace{\mathbf{p}(\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, t)}_{\text{inertial}} = \underbrace{\mathbf{f}(\mathbf{u}, \dot{\mathbf{u}}, t)}_{\text{external}}$
II General nonlinear statics	$\underbrace{\mathbf{p}(\mathbf{u})}_{\text{inertial}} = \underbrace{\mathbf{f}(\mathbf{u})}_{\text{external}}$
III Flexible structure nonlinear dynamics	$\underbrace{\mathbf{p}_i(\mathbf{u}, \dot{\mathbf{u}}, \ddot{\mathbf{u}}, t)}_{\text{inertial}} + \underbrace{\mathbf{p}_d(\mathbf{u}, \dot{\mathbf{u}}, t)}_{\text{damping}} + \underbrace{\mathbf{p}_e(\mathbf{u}, t)}_{\text{elastic}} = \underbrace{\mathbf{f}(\mathbf{u}, t)}_{\text{external}}$
IV Flexible structure linear dynamics	$\underbrace{\mathbf{M}\ddot{\mathbf{u}}(t)}_{\text{inertial}} + \underbrace{\mathbf{C}\dot{\mathbf{u}}(t)}_{\text{damping}} + \underbrace{\mathbf{K}\mathbf{u}(t)}_{\text{elastic}} = \underbrace{\mathbf{f}(t)}_{\text{external}}$
V Linear elastostatics	$\underbrace{\mathbf{K}\mathbf{u}}_{\text{elastic}} = \underbrace{\mathbf{f}}_{\text{external}}$
VI Dynamic perturbations	$\underbrace{\mathbf{M}(\mathbf{u})\ddot{\mathbf{d}}(t)}_{\text{inertial}} + \underbrace{\mathbf{C}(\mathbf{u})\dot{\mathbf{d}}(t)}_{\text{damping}} + \underbrace{\mathbf{K}(\mathbf{u})\mathbf{d}(t)}_{\text{elastic}} + \underbrace{\mathbf{p}(\mathbf{u}) = \mathbf{f}(\mathbf{u})}_{\text{static equilibrium}}$
VII Damped forced vibrations	$\underbrace{\mathbf{M}\ddot{\mathbf{u}}(t)}_{\text{inertial}} + \underbrace{\mathbf{C}\dot{\mathbf{u}}(t)}_{\text{damping}} + \underbrace{\mathbf{K}\mathbf{u}(t)}_{\text{elastic}} = \underbrace{\mathbf{f}_p(t)}_{\text{periodic}}$
VIII Damped free vibrations	$\underbrace{\mathbf{M}\ddot{\mathbf{u}}(t)}_{\text{inertial}} + \underbrace{\mathbf{C}\dot{\mathbf{u}}(t)}_{\text{damping}} + \underbrace{\mathbf{K}\mathbf{u}(t)}_{\text{elastic}} = \mathbf{0}$
IX Undamped free vibrations	$\underbrace{\mathbf{M}\ddot{\mathbf{u}}(t)}_{\text{inertial}} + \underbrace{\mathbf{K}\mathbf{u}(t)}_{\text{elastic}} = \mathbf{0}$
<p>Symbol \mathbf{u} is array of total displacement DOFs; \mathbf{d} in case VI is a linearized perturbation of \mathbf{u}. Symbol t denotes time. Superposed dots abbreviate time derivatives: $\dot{\mathbf{u}} = d\mathbf{u}/dt$, $\ddot{\mathbf{u}} = d^2\mathbf{u}/dt^2$, etc. The history $\mathbf{u} = \mathbf{u}(t)$ is called the <i>response</i> of the system. This term is extendible to nonlinear statics. Initial force effects \mathbf{f}_i may be accommodated in forced cases by taking $\mathbf{f} = \mathbf{f}_i$ when $\mathbf{u} = \mathbf{0}$.</p>	

When the model is time dependent, the relations shown in the right column of Table 31.1 are called *semidiscrete equations of motion*. The qualifier “semidiscrete” says that the time dimension has not been discretized: t is still a continuous variable. This legalizes the use of time differentiation, abbreviated by superposed dots, to bring in velocities and accelerations.

This table may be scanned “top down” by starting with the most general case I: nonlinear structural dynamics, branching down to more restricted but specific forms. Along the way one finds in case V an old friend: the DSM master equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ for linear elastostatics, treated in previous Chapters. The last case IX: undamped free vibrations, is that treated in this and next two Chapters. Some brief comments are made as regards damped and forced vibrations.

§31.2.1. Vibrations as Equilibrium Disturbance

An elastic structure is placed in motion through some short-term disturbance, for example an impulse. Remove the disturbance. If wave propagation effects are ignored and the structure remains

elastic, it will keep on oscillating in a combination of time-periodic patterns called *vibration modes*. Associated with each vibration mode is a characteristic time called *vibration period*. The inverse of a period, normalized by appropriate scaling factors, is called a *vibration frequency*. The structure is said to be vibrating, or more precisely *undergoing free vibrations*. In the absence of damping mechanisms an elastic structure will vibrate forever. The presence of even minute amounts of viscous damping, however, will cause a gradual decrease in the amplitude of the oscillations. These will eventually cease.²

If the disturbances are sufficiently small to warrant linearization, this scenario fits case VI of Table 31.1, therein labeled “dynamic perturbations.” Its main application is the investigation of *dynamic stability* of equilibrium configurations. If the perturbation $\mathbf{d}(t)$ is unbounded under some initial conditions, that equilibrium configuration³ is said to be dynamically unstable.

The analysis of case VI does not belong to an introductory course because it requires advanced mathematical tools. Moreover it often involve nondeterministic (stochastic) effects. Cases VII through IX are more tractable in an introductory course. In these, fluctuations are linearized about an undeformed and unstressed state defined by $\mathbf{u} = \mathbf{0}$. Thus \mathbf{d} (the perturbed displacement) becomes simply \mathbf{u} (the total displacement). Matrices \mathbf{M} , \mathbf{C} and \mathbf{K} are called the mass, damping and stiffness matrices, respectively. These matrices are independent of \mathbf{u} since they are evaluated at the undeformed state $\mathbf{u} = \mathbf{0}$. Two scenarios are of interest in practice:

1. *Forced Vibrations*. The system is subjected to a time dependent force $\mathbf{f}(t)$. The response $\mathbf{u}(t)$ is determined from the linear dynamics equation: $\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{C}\dot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{f}(t)$ of case IV. Of particular interest in resonance studies is when $\mathbf{f}(t)$ is periodic in time, which is case VII.
2. *Free Vibrations*. The external force is zero for $t > 0$. The response $\mathbf{u}(t)$ is determined from initial conditions. If damping is viscous and light, the undamped model gives conservative answers and is much easier to handle numerically. Consequently the model of case IX is that generally adopted during design studies.

§31.2.2. Undamped Free Vibrations

From the foregoing discussion it follows that case IX: undamped free vibrations is of paramount importance in design. The governing equation is

$$\mathbf{M}\ddot{\mathbf{u}}(t) + \mathbf{K}\mathbf{u}(t) = \mathbf{0}. \quad (31.1)$$

This expresses a force balance⁴ in the following sense: in the absence of external loads the internal elastic forces $\mathbf{K}\mathbf{u}$ balance the negative of the inertial forces $\mathbf{M}\ddot{\mathbf{u}}$. The only ingredient beyond the by now familiar \mathbf{K} is the mass matrix \mathbf{M} . The size of these matrices will be denoted by n_f , the number of degrees of freedom upon application of support conditions.

² Mathematically a damped oscillation also goes on forever. Eventually, however, the motion amplitude reaches a molecular scale level at which a macroscopic idealization does not apply. At such point the oscillations in the physical structure can be considered to have ceased.

³ Usually obtained through a nonlinear static analysis. This kind of study, called dynamic stability analysis, is covered under Nonlinear Finite Element Methods.

⁴ Where is $f = ma$? To pass to internal forces change the sign of f : $f_{int} + ma = ku + ma = 0$. Replace by matrices and vectors and you have (31.1).

Equation (31.1) is linear and homogeneous. Its general solution is a linear combination of exponentials. Under matrix definiteness conditions discussed later the exponentials can be expressed as a combination of trigonometric functions: sines and cosines of argument ωt . A compact representation of such functions is obtained by using the exponential form $e^{j\omega t}$, where $j = \sqrt{-1}$:

$$\mathbf{u}(t) = \sum_i \mathbf{v}_i e^{j\omega_i t}. \quad (31.2)$$

Here ω_i is the i^{th} *circular frequency*, expressed in radians per second, and $\mathbf{v}_i \neq \mathbf{0}$ the corresponding vibration mode shape, which is independent of t .

§31.2.3. The Vibration Eigenproblem

Replacing $\mathbf{u}(t) = \mathbf{v} e^{j\omega t}$ in (31.1) segregates the time dependence to the exponential: $(-\omega^2 \mathbf{M} + \mathbf{K}) \mathbf{v} e^{j\omega t} = \mathbf{0}$. Since $e^{j\omega t}$ is not identically zero, it can be dropped leaving the algebraic condition:

$$(-\omega^2 \mathbf{M} + \mathbf{K}) \mathbf{v} = \mathbf{0}. \quad (31.3)$$

Because \mathbf{v} cannot be the null vector, this equation is an algebraic eigenvalue problem in ω^2 . The eigenvalues $\lambda_i = \omega_i^2$ are the roots of the characteristic polynomial be indexed by i :

$$\det(\mathbf{K} - \omega_i^2 \mathbf{M}) = 0. \quad (31.4)$$

Dropping the index i this eigenproblem is usually written as

$$\mathbf{K} \mathbf{v} = \omega^2 \mathbf{M} \mathbf{v}. \quad (31.5)$$

If \mathbf{M} and \mathbf{K} satisfy some mild conditions, solutions of (31.5) are denoted by ω_i and \mathbf{v}_i . These are called the *vibration frequencies* or *eigenfrequencies*, and the *vibration modes* or *eigenmodes*, respectively. The set of all ω_i is called the *frequency spectrum* or simply *spectrum*.

§31.2.4. Eigensystem Properties

Both stiffness \mathbf{K} and mass \mathbf{M} are symmetric matrices. In addition \mathbf{M} is nonnegative. Nothing more can be assumed in general. For example, if \mathbf{K} incorporates Lagrangian multipliers from the treatment of a MFC, as explained in Chapter 10, it will be indefinite.

If \mathbf{M} is positive definite, the following properties hold.

1. There are n_f squared vibration frequencies ω_i^2 , which are roots of the characteristic polynomial (31.4). These are not necessarily distinct. A root of (31.6) that appears m times is said to have multiplicity m .⁵
2. All roots ω_i^2 of (31.6) are real. The corresponding eigenmodes \mathbf{v}_i have real entries.
3. If \mathbf{K} is nonnegative, $\omega_i^2 \geq 0$ and the frequencies $\omega_i = +\sqrt{\omega_i^2}$ are also real and nonnegative. Furthermore, if \mathbf{K} is positive definite, all $\omega_i^2 > 0$ and consequently $+\sqrt{\omega_i^2} > 0$.

If \mathbf{M} is nonnegative, care must be exercised; this case is discussed in an Exercise. If \mathbf{M} is indefinite (which should never happen in structures) all of the foregoing properties are lost.

⁵ For example, a free-free (fully unsupported) structure has n_R zero frequencies, where n_R is the number of rigid body modes.

Example 31.1. This illustrates the weird things that can happen if \mathbf{M} is indefinite. Consider

$$\mathbf{K} = \begin{bmatrix} \alpha & 1 \\ 1 & 2 \end{bmatrix}, \quad \mathbf{M} = \begin{bmatrix} 0 & 1 \\ 1 & 1 + \beta \end{bmatrix}, \quad (31.6)$$

where α and β vary from 1 to -1 . Then

$$\mathbf{M}^{-1}\mathbf{K} = \frac{1}{2\alpha - 1} \begin{bmatrix} 1 & (1 - \beta) \\ \alpha & (-1 + \alpha + \alpha\beta) \end{bmatrix}. \quad (31.7)$$

The eigenvalues are

$$\omega_{1,2}^2 = \frac{-2 + \alpha + \alpha\beta \pm \sqrt{\alpha[4 - 4\beta + \alpha(1 + \beta)^2]}}{4\alpha - 2}. \quad (31.8)$$

These are complex if the radicand is negative. But that is not all. If $\alpha \rightarrow 0$ one eigenvalue goes to ∞ . If $\alpha = 0$, $\mathbf{A} = \mathbf{M}^{-1}\mathbf{K}$ is a 2×2 Jordan block and one eigenvector is lost.

§31.3. Solving The Vibration Eigenproblem

In what follows we often denote $\lambda_i = \omega_i^2$ to agree more closely with the conventional notation for the algebraic eigenproblem.

§31.3.1. Determinant Roots

Mathematically the ω_i^2 are the roots of the characteristic equation (31.4). The simple minded approach is to expand the determinant to get the characteristic polynomial $P(\omega_i^2)$ and get their roots:

$$\det(\mathbf{K} - \omega_i^2 \mathbf{M}) = P(\omega_i^2) = 0. \quad (31.9)$$

This approach is deprecated by numerical analysts. It seems as welcome as anthrax. Indeed for numerical floating point computations of large systems it risks numerical overflow; moreover the roots of the characteristic polynomial can be very ill-conditioned with respect to coefficients.

For small systems and using either exact or symbolic computation there is nothing wrong with this if the roots can be expressed exactly in terms of the coefficients, as in the above example.

§31.3.2. Reduction to the Standard Eigenproblem

The standard algebraic eigenproblem has the form

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}. \quad (31.10)$$

Most library routines included in packages such as *Matlab* and *Mathematica* are designed to solve this eigenproblem. If \mathbf{A} is symmetric the eigenvalues λ_i are real; moreover there exist a complete system of eigenvectors \mathbf{x}_i . If these are normalised to length one: $\|\mathbf{x}_i\|_2 = 1$ they satisfy the orthonormality conditions

$$\mathbf{x}_i^T \mathbf{x}_j = \delta_{ij} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}, \quad \mathbf{x}_i^T \mathbf{A} \mathbf{x}_j = \lambda_i, \quad (31.11)$$

where δ_{ij} is the Kronecker delta. If the \mathbf{x}_i are collected as columns of a matrix \mathbf{X} , the foregoing conditions can be expressed as $\mathbf{X}^T \mathbf{X} = \mathbf{I}$ and $\mathbf{X}^T \mathbf{K} \mathbf{X} = \mathbf{\Lambda} = \mathbf{diag} \lambda_i$.

§31.3.3. Unsymmetric Reduction

If \mathbf{M} is nonsingular, a simple way to reduce $\mathbf{K}\mathbf{v} = \omega^2\mathbf{M}\mathbf{v}$ to standard form is to premultiply both sides by \mathbf{M}^{-1} whence

$$\mathbf{M}^{-1}\mathbf{K}\mathbf{v} = \omega^2\mathbf{v} \Rightarrow \mathbf{A}\mathbf{x} = \lambda\mathbf{x}, \quad \text{with} \quad \mathbf{A} = \mathbf{M}^{-1}\mathbf{K}, \quad \lambda = \omega^2, \quad \mathbf{x} = \mathbf{v}. \quad (31.12)$$

The fastest way to form \mathbf{A} is by solving $\mathbf{M}\mathbf{A} = \mathbf{K}$ for \mathbf{A} . One nice feature of (31.12) is that the eigenvectors need not be backtransformed, as happens in symmetry-preserving methods.

As in the case of the characteristic polynomial, this is deprecated by numerical analysts, also not so vehemently. Their objection is that \mathbf{A} is not generally symmetric even if \mathbf{K} and \mathbf{M} are. So $\mathbf{A}\mathbf{x} = \lambda\mathbf{x}$ has to be submitted to an unsymmetric eigensolver. Thus risks contaminating the spectrum with complex numbers. Plus, it is slower.

The writer's experience is that (31.12) works perfectly fine for small systems. If tiny imaginary components appear, they are set to zero and life goes on.

§31.3.4. Symmetry Preserving Reduction

It is possible to retain symmetry by proceeding as follows. Decompose the mass matrix as

$$\mathbf{M} = \mathbf{L}\mathbf{L}^T \quad (31.13)$$

This is the Cholesky decomposition, which can be carried out to completion if \mathbf{M} is positive definite. Then

$$\mathbf{A} = \mathbf{L}^{-1}\mathbf{K}\mathbf{L}^{-T}. \quad (31.14)$$

The demonstraion is in one of the Exercises. The symmetric eigenproblem can be handled by standard library routines, which give back all the eigenvalues and eigenvectors. The square root of the eigenvalues give the vibration frequencies and the vibration modes are recovered from the relation $\mathbf{L}\mathbf{v}_i = \mathbf{x}_i$, which can be handled by standard library routines.

Notes and Bibliography

The literature on dynamics and vibrations of structures is quite large. It is sufficient to cite here titles that incorporate modern analysis methods: Clough and Penzien [45], Geradin and Rixen [112], Meirovich [174,175] and Wilson [272].

Several books in matrix methods and FEM books contain at least an introductory treatment of dynamics. Citable textbooks include Bathe [15], Cook, Malkus and Plesha [50], Hughes [142]. Despite their age, Przemieniecki [205] remains a useful source of mass matrices, and Pestel and Leckie [194] contains a catalog of transfer matrices (an early 1960 method suitable for small computers).

As regards books on linear algebra matrix theory and matrix calculus see the Bibliography cited in Appendix A. The most elegant coverage is that of Strang [229]. Two comprehensive references on matrix computations in general are Golub and VanLoan [115] and Stewart [225]. The former is more up to date as regard recent literature. Bellman [24] contains more advanced material. Stewart and Sun [226] cover the sensitivity analysis of standard and generalized eigenproblems.

There are comprehensive books that treat the algebraic eigenproblem. Wilkinson's masterpiece [265] is dated in several subjects, particularly the generalized eigenproblem and the treatment of large eigenproblems. But it is still unsurpassed as the "bible" of backward error analysis. More up to date in methods is Parlett [192], which is however restricted to the symmetric eigenproblem.

As regards source code for matrix computations, the Handbook compilation of Algol 60 procedures by Wilkinson and Reisch [266] is elegant and still useful as template for other languages. Half of the handbook deals with eigenvalue problems. By contrast, the description of Fortran EISPACK code [110] suffers from the inherent ugliness and unreadability of Fortran IV.

And of course there is Numerical Recipes in various flavors. To borrow from the immortal words of Winston Churchill, “never have so few wasted the time of so many.”

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 31 - Dynamics & Vibration Overview

EXERCISE 31.1 [A:15]. A 3-element model of a bar in 1D gives

$$\mathbf{M} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 1 & 4 & 1 & 0 \\ 0 & 1 & 4 & 1 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \quad \mathbf{K} = \begin{bmatrix} 1 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{bmatrix}. \quad (\text{E31.1})$$

Solve the vibration eigenproblem and show the natural frequencies and associated vibration modes. Normalize the latter so that $\mathbf{V}^T \mathbf{M} \mathbf{V} = \mathbf{I}$ (“mass normalized eigenvectors”).

EXERCISE 31.2 [A:25]. In (E31.1) replace the (4,4) mass entry by $2 - \alpha$ and the (4,4) stiffness entry by $1 - \alpha/2$. Using *Matlab* or *Mathematica*, solve the eigenproblem for α varying from 0 to 4 in 0.5 increments. Discuss what happens to the frequencies and vibration modes as α goes to 2 and beyond. Explain.

EXERCISE 31.3 [D:20]. Eigenvectors can be scaled by arbitrary nonzero factors. Discuss 4 ways in which the eigenvectors \mathbf{v}_i of $\mathbf{K}\mathbf{v}_i = \omega_i^2 \mathbf{M}\mathbf{v}_i$ can be normalized, and what assumptions are necessary in each case.

32

Lumped and Consistent Mass Matrices

TABLE OF CONTENTS

	Page
§32.1. Introduction	32-3
§32.2. Mass Matrix Construction	32-3
§32.2.1. Direct Mass Lumping	32-3
§32.2.2. Variational Mass Lumping	32-4
§32.2.3. Template Mass Lumping	32-4
§32.2.4. Mass Matrix Properties	32-5
§32.2.5. Rank and Numerical Integration	32-6
§32.3. Globalization	32-6
§32.4. Mass Matrix Examples: Bars and Beams	32-8
§32.4.1. The 3-Node Bar	32-8
§32.4.2. The Bernoulli-Euler Plane Beam	32-8
§32.4.3. The Plane Beam-Column	32-10
§32.4.4. *The Timoshenko Plane Beam	32-11
§32.4.5. *Spar and Shaft Elements	32-12
§32.5. Mass Matrix Examples: Plane Stress	32-12
§32.5.1. The Plane Stress Linear Triangle	32-12
§32.5.2. Four-Node Bilinear Quadrilateral	32-14
§32.6. Mass Diagonalization Methods	32-15
§32.6.1. HRZ Lumping	32-15
§32.6.2. Lobatto Lumping	32-15
§32. Notes and Bibliography	32-16
§32. References	32-18
§32. Exercises	32-19

§32.1. Introduction

To do dynamic and vibration finite element analysis, you need at least a mass matrix to pair with the stiffness matrix. This Chapter provides a quick introduction to standard methods for computing this matrix.

As a general rule, the construction of the master mass matrix \mathbf{M} largely parallels that of the master stiffness matrix \mathbf{K} . Mass matrices for individual elements are formed in local coordinates, transformed to global, and merged into the master mass matrix following exactly the same techniques used for \mathbf{K} . In practical terms, the assemblers for \mathbf{K} and \mathbf{M} can be made identical. This procedural uniformity is one of the great assets of the Direct Stiffness Method.

A notable difference with the stiffness matrix is the possibility of using a *diagonal* mass matrix based on direct lumping. A master diagonal mass matrix can be stored simply as a vector. If all entries are nonnegative, it is easily inverted, since the inverse of a diagonal matrix is also diagonal. Obviously a lumped mass matrix entails significant computational advantages for calculations that involve \mathbf{M}^{-1} . This is balanced by some negative aspects that are examined in some detail later.

§32.2. Mass Matrix Construction

The master mass matrix is built up from element contributions, and we start at that level. The construction of the mass matrix of individual elements can be carried out through several methods. These can be categorized into three groups: direct mass lumping, variational mass lumping, and template mass lumping. The last group is more general in that includes all others. Variants of the first two techniques are by now standard in the FEM literature, and implemented in all general purpose codes. Consequently this Chapter covers the most widely used methods, focusing on techniques that produce diagonally lumped and consistent mass matrices. The next Chapter covers the template approach to produce customized mass matrices.

§32.2.1. Direct Mass Lumping

The total mass of element e is directly apportioned to nodal freedoms, ignoring any cross coupling. The goal is to build a *diagonally lumped mass matrix* or DLMM, denoted here by \mathbf{M}_L^e .

As the simplest example, consider a 2-node prismatic bar element with length ℓ , cross section area A , and mass density ρ , which can only move in the axial direction x , as depicted in Figure 32.1. The total mass of the element is $M^e = \rho A \ell$. This is divided into two equal parts and assigned to each end node to produce

$$\mathbf{M}_L^e = \frac{1}{2} \rho A \ell \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{2} \rho A \ell \mathbf{I}_2, \quad (32.1)$$

in which \mathbf{I}_2 denotes the 2×2 identity matrix. As shown in the figure, we have replaced the bar with a “dumbbell.”

This process conserves the translational kinetic energy or, equivalently, the linear momentum. To show this for the bar example, take the constant x -velocity vector $\dot{\mathbf{u}}^e = v [1 \ 1]^T$. The kinetic

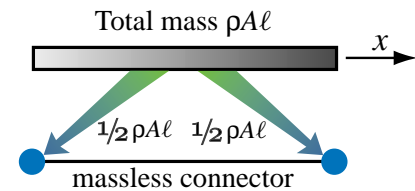


FIGURE 32.1. Direct mass lumping for 2-node prismatic bar element.

energy of the element is $T^e = \frac{1}{2}(\dot{\mathbf{u}}^e)^T \mathbf{M}_L^e \dot{\mathbf{u}}^e = \frac{1}{2}\rho A \ell v^2 = \frac{1}{2}M^e v^2$. Thus the linear momentum $p^e = \partial T^e / \partial v = M^e v$ is preserved. When applied to simple elements that can rotate, however, the direct lumping process may not necessarily preserve *angular* momentum.

A key motivation for direct lumping is that, as noted in §32.1, a diagonal mass matrix may offer computational and storage advantages in certain simulations, notably explicit time integration. Furthermore, direct lumping covers naturally the case where concentrated (point) masses are natural part of model building. For example, in aircraft engineering it is common to idealize nonstructural masses (fuel, cargo, engines, etc.) as concentrated at given locations.¹

§32.2.2. Variational Mass Lumping

A second class of mass matrix construction methods are based on a variational formulation. This is done by taking the *kinetic energy* as part of the governing functional. The kinetic energy of an element of mass density ρ that occupies the domain Ω^e and moves with velocity field $\vec{\mathbf{v}}^e$ is

$$T^e = \frac{1}{2} \int_{\Omega^e} \rho (\vec{\mathbf{v}}^e)^T \vec{\mathbf{v}}^e d\Omega^e. \quad (32.2)$$

Following the FEM philosophy, the element velocity field is interpolated by shape functions: $\vec{\mathbf{v}}^e = \mathbf{N}_v^e \dot{\mathbf{u}}^e$, where $\dot{\mathbf{u}}^e$ are node DOF velocities and \mathbf{N}_v^e a shape function matrix. Inserting into (32.2) and moving the node velocities out of the integral gives

$$T^e = \frac{1}{2}(\dot{\mathbf{u}}^e)^T \int_{\Omega^e} \rho (\mathbf{N}_v^e)^T \mathbf{N}_v^e d\Omega \dot{\mathbf{u}}^e \stackrel{\text{def}}{=} \frac{1}{2}(\dot{\mathbf{u}}^e)^T \mathbf{M}^e \dot{\mathbf{u}}^e, \quad (32.3)$$

whence the element mass matrix follows as the Hessian of T^e :

$$\boxed{\mathbf{M}^e = \frac{\partial^2 T^e}{\partial \dot{\mathbf{u}}^e \partial \dot{\mathbf{u}}^e} = \int_{\Omega^e} \rho (\mathbf{N}_v^e)^T \mathbf{N}_v^e d\Omega.} \quad (32.4)$$

If the same shape functions used in the derivation of the stiffness matrix are chosen, that is, $\mathbf{N}_v^e = \mathbf{N}^e$, (32.4) is called the *consistent mass matrix*² or CMM. It is denoted here by \mathbf{M}_C^e .

For the 2-node prismatic bar element moving along x , the stiffness shape functions of Chapter 12 are $N_i = 1 - (x - x_i)/\ell = 1 - \zeta$ and $N_j = (x - x_i)/\ell = \zeta$. With $dx = \ell d\zeta$, the consistent mass is easily obtained as

$$\mathbf{M}_C^e = \int_0^\ell \rho A (\mathbf{N}^e)^T \mathbf{N}^e dx = \rho A \int_0^1 \begin{bmatrix} 1 - \zeta \\ \zeta \end{bmatrix} [1 - \zeta \quad \zeta] \ell d\zeta = \frac{1}{6} \rho A \ell \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}. \quad (32.5)$$

It can be verified that this mass matrix preserves linear momentum along x .

¹ Such concentrated masses in general have rotational freedoms. Rotational inertia lumping is then part of the process.

² A better name would be stiffness-consistent. The shorter sobriquet has the unfortunately implication that other choices are “inconsistent,” which is far from the truth. In fact, the consistent mass is not necessarily the best one, a topic elaborated in the next Chapter. However the name is by now ingrained in the FEM literature.

§32.2.3. Template Mass Lumping

A generalization of the two foregoing methods consists of expressing the mass as a linear combination of k component mass matrices:

$$\mathbf{M}^e = \sum_{i=1}^k \mu_i \mathbf{M}_i^e. \quad (32.6)$$

Appropriate constraints on the free parameters μ_i are placed to enforce matrix properties discussed in §32.2.4. Variants result according to how the component matrices \mathbf{M}_i are chosen, and how the parameters μ_i are determined. The best known scheme of this nature results on taking a weighted average of the consistent and diagonally-lumped mass matrices:

$$\mathbf{M}_{LC}^e \stackrel{\text{def}}{=} (1 - \mu) \mathbf{M}_C^e + \mu \mathbf{M}_L^e, \quad (32.7)$$

in which μ is a free scalar parameter. This is called the LC (“lumped-consistent”) weighted mass matrix. If $\mu = 0$ and $\mu = 1$ this combination reduces to the consistent and lumped mass matrix, respectively. For the 2-node prismatic bar we get

$$\mathbf{M}_{LC}^e = (1 - \mu) \frac{1}{6} \rho A \ell \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \mu \frac{1}{2} \rho A \ell \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \frac{1}{6} \rho A \ell \begin{bmatrix} 2 + \mu & 1 - \mu \\ 1 - \mu & 2 + \mu \end{bmatrix}. \quad (32.8)$$

It is known (since the early 1970s) that the best choice with respect to minimizing low frequency dispersion is $\mu = 1/2$. This is proven in the next Chapter.

The most general method of this class uses *finite element templates* to fully parametrize the element mass matrix. For the prismatic 2-node bar element one would start with the 3-parameter template

$$\mathbf{M}^e = \rho A \ell \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{12} & \mu_{22} \end{bmatrix}, \quad (32.9)$$

which includes the symmetry constraint from the start. Invariance requires $\mu_{22} = \mu_{11}$, which cuts the free parameters to two. Conservation of linear momentum requires $\mu_{11} + \mu_{12} + \mu_{12} + \mu_{22} = 2\mu_{11} + 2\mu_{12} = 1$, or $\mu_{12} = 1/2 - \mu_{11}$. Taking $\mu = 6\mu_{11} - 2$ reduces (32.9) to (32.8). Consequently for the 2-node bar LC-weighting and templates are the same thing, because only one free parameter is left upon imposing essential constraints. This is not the case for more complicated elements.

§32.2.4. Mass Matrix Properties

Mass matrices must satisfy certain conditions that can be used for verification and debugging. They are: (1) matrix symmetry, (2) physical symmetries, (3) conservation and (4) positivity.

Matrix Symmetry. This means $(\mathbf{M}^e)^T = \mathbf{M}^e$, which is easy to check. For a variationally derived mass matrix this follows directly from the definition (32.4), while for a DLMM is automatic.

Physical Symmetries. Element symmetries must be reflected in the mass matrix. For example, the CMM or DLMM of a prismatic bar element must be symmetric about the antidiagonal: $M_{11} = M_{22}$. To see this, flip the end nodes: the element remains the same and so does the mass matrix.

Conservation. At a minimum, total element mass must be preserved.³ This is easily verified by applying a uniform translational velocity and checking that linear momentum is conserved. Higher order conditions, such as conservation of angular momentum, are optional and not always desirable.

Positivity. For any nonzero velocity field defined by the node values $\dot{\mathbf{u}}^e \neq \mathbf{0}$, $(\dot{\mathbf{u}}^e)^T \mathbf{M}^e \dot{\mathbf{u}}^e \geq 0$. That is, \mathbf{M}^e must be nonnegative. Unlike the previous three conditions, this constraint is nonlinear in the mass matrix entries. It can be checked in two ways: through the eigenvalues of \mathbf{M}^e , or the sequence of principal minors. The second technique is more practical if the entries of \mathbf{M}^e are symbolic.

Remark 32.1. A more demanding form of the positivity constraint is to require that \mathbf{M}^e be *positive definite*: $(\dot{\mathbf{u}}^e)^T \mathbf{M}^e \dot{\mathbf{u}}^e > 0$ for any $\dot{\mathbf{u}}^e \neq \mathbf{0}$. This is more physically reassuring because one half of that quadratic form is the kinetic energy associated with the velocity field defined by $\dot{\mathbf{u}}^e$. In a continuum T can vanish only for zero velocities. But allowing $T^e = 0$ for some nonzero $\dot{\mathbf{u}}^e$ makes life easier in some situations, particularly for elements with rotational or Lagrange multiplier freedoms.

The $\dot{\mathbf{u}}^e$ for which $T^e = 0$ form the *null space* of \mathbf{M}^e . Because of the conservation requirement, a rigid velocity field (the time derivative $\dot{\mathbf{u}}_R^e$ of a rigid body mode \mathbf{u}_R^e) cannot be in the mass matrix null space, since it would imply zero mass. This scenario is dual to that of the element stiffness matrix. For the latter, $\mathbf{K}^e \mathbf{u}_R^e = \mathbf{0}$, since a rigid body motion produces no strain energy. Thus \mathbf{u}_R^e must be in the null space of the stiffness matrix.

§32.2.5. Rank and Numerical Integration

Suppose the element has a total of n_F^e freedoms. A mass matrix \mathbf{M}^e is called *rank sufficient* or of *full rank* if its rank is $r_M^e = n_F^e$. Because of the positivity requirement, a rank-sufficient mass matrix must be positive definite. Such matrices are preferred from a numerical stability standpoint.

If \mathbf{M}^e has rank $r_M^e < n_F^e$ the mass is called rank deficient by $d_M^e = n_F^e - r_M^e$. Equivalently \mathbf{M}^e is d_M^e times singular. For a numerical matrix the rank is easily computed by taking its eigenvalues and looking at how many of them are zero. The null space can be extracted by functions such as `NullSpace` in *Mathematica* without the need of computing eigenvalues.

The computation of \mathbf{M}^e by the variational formulation (32.4) is often done using Gauss numerical quadrature. Each Gauss points adds n_D to the rank, where n_D is the row dimension of the shape function matrix \mathbf{N}^e , up to a maximum of n_F^e . For most elements n_D is the same as element spatial dimensionality; that is, $n_D = 1, 2$ and 3 for $1, 2$ and 3 dimensions, respectively. This property can be used to pick the minimum Gauss integration rule that makes \mathbf{M}^e positive definite.

§32.3. Globalization

Like their stiffness counterparts, mass matrices are often developed in a local or element frame. Should globalization be necessary before merge, a congruent transformation is applied:

$$\mathbf{M}^e = (\mathbf{T}^e)^T \bar{\mathbf{M}}^e \mathbf{T}^e \quad (32.10)$$

Here $\bar{\mathbf{M}}^e$ is the element mass referred to the local frame whereas \mathbf{T}^e is the local-to-global displacement transformation matrix. Matrix \mathbf{T}^e is in principle that used for the stiffness globalization. Some procedural differences, however, must be noted. For stiffness matrices \mathbf{T}^e is often *rectangular* if the local stiffness has lower dimensionality. For example, the bar and spar elements formulated of

³ We are taking about classical mechanics here; in relativistic mechanics mass and energy can be exchanged.

Chapter 6 have 2×2 local stiffnesses. Globalization to 2D and 3D involves application of 2×4 and 2×6 transformation matrices, respectively. This works fine because the local element has zero stiffness in some directions. If the associated freedoms are explicitly kept in the local stiffness, as in Chapters 2–3, those rows and columns are zero and have no effect on the global stiffness.

In contrast to stiffnesses, *translational masses never vanish*. One way to understand this is to think of an element moving in a translational rigid body motion u_R with acceleration \ddot{u}_R . According to Newton's second law, $f_R = M^e \ddot{u}_R$, where M^e is the translational mass. This cannot be zero.

The conclusion is: *all translational masses must be retained in the local mass matrix*. The 2-node prismatic bar moving in the $\{x, y\}$ plane furnishes a simple illustration. With the freedoms arranged as $\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2}]^T$, the local mass matrix constructed by consistent and diagonalized lumping are

$$\bar{\mathbf{M}}_C^e = \frac{1}{6} \rho A \ell \begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}, \quad \bar{\mathbf{M}}_L^e = \frac{1}{2} \rho A \ell \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \frac{1}{2} \rho A \ell \mathbf{I}_4, \quad (32.11)$$

respectively. Globalize via (32.10) using the transformation matrix (3.2). The result is $\mathbf{M}_C^e = \bar{\mathbf{M}}_C^e$ and $\mathbf{M}_L^e = \bar{\mathbf{M}}_L^e$. We say that these mass matrices *repeat*. Verification for the DLMM is easy because \mathbf{T}^e is orthogonal: $(\mathbf{T}^e)^T \bar{\mathbf{M}}_L^e \mathbf{T}^e = \frac{1}{2} \rho A \ell (\mathbf{T}^e)^T \mathbf{I}_4 \mathbf{T}^e = \frac{1}{2} \rho A \ell (\mathbf{T}^e)^T \mathbf{T}^e = \frac{1}{2} \rho A \ell \mathbf{I}_4$. For the CMM, however, repetition is not obvious. It is best shown by temporarily rearranging the element DOF so that instead of $[u_{x1} \ u_{y1} \ u_{x2} \ u_{y2}]$ the x and y components are grouped: $[u_{x1} \ u_{x2} \ u_{y1} \ u_{y2}]$. Rearranging $\bar{\mathbf{M}}_C^e$ and \mathbf{T}^e accordingly yields

$$\bar{\mathbf{M}}_C^e = \begin{bmatrix} \tilde{\mathbf{M}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{M}} \end{bmatrix}, \quad \mathbf{T}^e = \begin{bmatrix} c\mathbf{I}_2 & s\mathbf{I}_2 \\ -s\mathbf{I}_2 & c\mathbf{I}_2 \end{bmatrix}, \quad \text{with } \tilde{\mathbf{M}} = \frac{\rho A \ell}{6} \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad c = \cos \phi, \quad s = \sin \phi. \quad (32.12)$$

in which $\phi = \angle x, \bar{x}$. Carrying out the transformation in block form gives

$$\mathbf{M}_C^e = \begin{bmatrix} c\mathbf{I}_2 & -s\mathbf{I}_2 \\ s\mathbf{I}_2 & c\mathbf{I}_2 \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{M}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{M}} \end{bmatrix} \begin{bmatrix} c\mathbf{I}_2 & s\mathbf{I}_2 \\ -s\mathbf{I}_2 & c\mathbf{I}_2 \end{bmatrix} = \begin{bmatrix} (c^2 + s^2)\tilde{\mathbf{M}} & (cs - cs)\tilde{\mathbf{M}} \\ (cs - cs)\tilde{\mathbf{M}} & (c^2 + s^2)\tilde{\mathbf{M}} \end{bmatrix} = \begin{bmatrix} \tilde{\mathbf{M}} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{M}} \end{bmatrix} = \bar{\mathbf{M}}_C^e. \quad (32.13)$$

A matrix that can be put in a block diagonal form built up of identical submatrices, such as $\bar{\mathbf{M}}_C^e$ in (32.12) is called *repeating block diagonal* or RBD. Note that the contents and order of $\tilde{\mathbf{M}}$ are irrelevant to the result (32.13). Hence the following generalization follows. If upon rearranging the element DOF: (i) $\bar{\mathbf{M}}^e$ is two-block RBD, and (ii) \mathbf{T}^e takes the block form shown above, the local and global matrices will coalesce. For (ii) to hold, it is sufficient that all nodal DOF be translational and be referred to the same coordinate system. The same conclusion holds for 3D; this is the subject of Exercise 32.5. This *repetition rule* can be summarized as follows:

A RBD local mass matrix globalizes to the same matrix if all element DOFs are translational and all of them are referred to the same global system.

(32.14)

This property should be taken advantage of to skip superfluous local-to-global transformations. Frequently such operation costs more than forming the local mass matrix.

What if the (32.14) fails on actual computation? Then something (mass matrix or transformation) is wrong and must be fixed. As example, suppose that one tries to parrot the bar stiffness derivation process by starting with the 1D bar mass (32.1). A rectangular 2×4 transformation matrix \mathbf{T}^e is built by taking rows 1 and 3 of (3.2). Then the globalization (32.10) is carried out. The resulting \mathbf{M}_L^e is found to violate (32.14) because entries depend on the orientation angle $\varphi = \angle(\bar{x}, x)$. The reason for this mistake is that $\bar{\mathbf{M}}_L^e$ must account for the inertia in the y direction, as in the second of (32.11), to start with.

Remark 32.2. The repetition rule (32.14) can be expected to fail in the following scenarios:

1. The element has non-translational freedoms. (Occasionally the rule may work, but that is unlikely.)
2. The mass blocks are different in content and/or size. This occurs if different models are used in different directions. Examples are furnished by beam-column, element with curved sides or faces, and shell elements.
3. Nodes are referred to different coordinate frames in the global system. This can happen if certain nodes are referred to special frames to facilitate the application of boundary conditions.

§32.4. Mass Matrix Examples: Bars and Beams

The diagonally lumped and consistent mass matrices for the 2-node bar element were explicitly given in (32.1) and (32.5), and an optimal combination is investigated in the next Chapter. In this section the DLMM and CMM of other simple elements are worked out. More complicated ones are relegated to Exercises. The overbar of \mathbf{M}^e is omitted for brevity unless a distinction between local and global mass matrices is required.

§32.4.1. The 3-Node Bar

The element is prismatic with length ℓ , area A , and uniform mass density ρ . Midnode 3 is at the center. The DOFs are arranged $\mathbf{u}^e = [u_1 \ u_2 \ u_3]^T$. Using the shape function in the isoparametric coordinate ξ presented in Exercise 16.2 we get the CMM

$$\mathbf{M}_C^e = \rho A \int_{-1}^1 (\mathbf{N}^e)^T \mathbf{N}^e (1/2\ell) d\xi = \frac{\rho A \ell}{30} \begin{bmatrix} 4 & -1 & 2 \\ -1 & 4 & 2 \\ 2 & 2 & 16 \end{bmatrix}. \quad (32.15)$$

To produce a DLMM, the total mass of the element is divided into 3 parts: $\alpha\rho A\ell$, $\alpha\rho A\ell$, and $(1 - 2\alpha)\rho A\ell$, which are assigned to nodes 1, 2 and 3, respectively.

For reasons discussed later the best choice is $\alpha = 1/6$, as depicted in Figure 32.2. Consequently $2/3$ of the total mass goes to the midpoint, and what is left to the corners, giving

$$\mathbf{M}_L^e = \frac{1}{6}\rho A \ell \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 4 \end{bmatrix}. \quad (32.16)$$

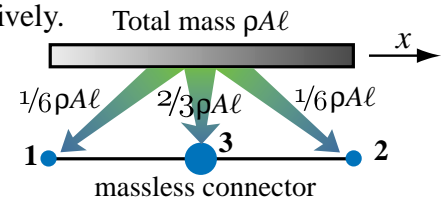


FIGURE 32.2. Direct mass lumping for 3-node bar element.

The $1/6:1/6:2/3$ allocation happens to be Simpson's rule for integration. This meshes in with the interpretation of diagonal mass lumping as a Lobatto integration rule, a topic discussed in §32.6.

Both (32.15) and (32.16) can be used as building blocks for expanding the element to 2D or 3D space. The repetition rule (32.14) holds.

```

Beam2BEConsMass[Le_, ρ_, A_, {numer_, p_}]:=
Module[{i, k, Ne, NeT, ξ, w, fac, Me=Table[0, {4}, {4}]},
Ne={2*(1-ξ)^2*(2+ξ), (1-ξ)^2*(1+ξ)*Le,
2*(1+ξ)^2*(2-ξ), -(1+ξ)^2*(1-ξ)*Le}/8;
NeT=Transpose[Ne]; fac=ρ*A*Le/2;
If [p==0, Me=fac*Integrate[NeT.Ne,{ξ,-1,1}]; Return[Me]];
For [k=1, k<=p, k++,
{xi,w}= LineGaussRuleInfo[{p,numer},k];
Me+= w*fac*(NeT/.ξ->xi).(Ne/.ξ->xi);
]; If[!numer,Me=Simplify[Me]]; Return[Me]
];
ClearAll[ρ,A,Le,ξ]; mfac=(ρ*A*Le)/420;
MeC=Simplify[Beam2BEConsMass[Le,ρ,A,{False,0}]/mfac];
For [p=1,p<=5,p++, Print["p=",p];
Me=Simplify[Beam2BEConsMass[Le,ρ,A,{False,p}]/mfac];
Print["Me=",mfac,"",Me//MatrixForm,
" vs exact=",mfac,"",MeC//MatrixForm];
Print["eigs scaled ME=",Chop[Eigenvalues[N[Me/.Le->1]]]];
uRot={-Le/2,1,Le/2,1}*θ; TRot=Simplify[(1/2)*uRot.MeC*uRot*mfac];
TRotex=Simplify[(Le/2)*Integrate[(1/2)*ρ*A*(θ*Le*ξ/2)^2,{ξ,-1,1}]];
Print["TRot=",TRot," should match ",TRotex];

```

FIGURE 32.3. Module to form the CMM of a prismatic 2-node Bernoulli-Euler plane beam element. Integration is done analytically if $p=0$, and numerically if $p > 0$ using a p -point Gauss rule.

§32.4.2. The Bernoulli-Euler Plane Beam

The stiffness of this element was derived in Chapter 13. The 2-node plane beam element has length ℓ , cross section area A and uniform mass density ρ . Only the translational inertia due to the lateral motion of the beam is considered in computing the kinetic energy $T = \frac{1}{2} \int_0^\ell \rho \dot{v}(\bar{x})^2 d\bar{x}$ of the element, whereas the rotational inertia is ignored. With the freedoms arranged as $\mathbf{u}^e = [v_1 \ \theta_1 \ v_2 \ \theta_2]^T$, use of the cubic shape functions (13.12) gives the CMM

$$\bar{\mathbf{M}}_C^e = \rho A \int_{-1}^1 (1/2\ell)(\mathbf{N}^e)^T \mathbf{N}^e d\xi = \frac{\rho A \ell}{420} \begin{bmatrix} 156 & 22\ell & 54 & -13\ell \\ 22\ell & 4\ell^2 & 13\ell & -3\ell^2 \\ 54 & 13\ell & 156 & -22\ell \\ -13\ell & -3\ell^2 & -22\ell & 4\ell^2 \end{bmatrix}. \quad (32.17)$$

in which $1/2\ell$ is the Jacobian $J = dx/d\xi$. This result may be verified using the *Mathematica* module *Beam2BEConsMatrix* listed in Figure 32.3. The arguments are self-explanatory except for p . The module computes the integral (32.17) analytically if $p=0$; else using a p -point 1D Gauss rule (extracted from *LineGaussRuleInfo*, described in Chapter 17) if $1 \leq p \leq 5$. *Beam2BEConsMatrix* is run for p varying from 0 through 5 by the statements following the module. The mass matrices obtained with integration rules of 1, 2 and 3 points are

$$c_1 \begin{bmatrix} 16 & 4\ell & 16 & -4\ell \\ 4\ell & \ell^2 & 4\ell & -\ell^2 \\ 16 & 4\ell & 16 & -4\ell \\ -4\ell & -\ell^2 & -4\ell & \ell^2 \end{bmatrix}, \quad c_2 \begin{bmatrix} 86 & 13\ell & 22 & -5\ell \\ 13\ell & 2\ell^2 & 5\ell & -\ell^2 \\ 22 & 5\ell & 86 & -13\ell \\ -5\ell & -\ell^2 & -13\ell & 2\ell^2 \end{bmatrix}, \quad c_3 \begin{bmatrix} 444 & 62\ell & 156 & -38\ell \\ 62\ell & 11\ell^2 & 38\ell & -9\ell^2 \\ 156 & 38\ell & 444 & -62\ell \\ -38\ell & -9\ell^2 & -62\ell & 11\ell^2 \end{bmatrix} \quad (32.18)$$

in which $c_1 = \rho A \ell / 64$, $c_2 = \rho A \ell / 216$ and $c_3 = \rho A \ell / 1200$. The eigenvalue analysis shows that all three are singular, with rank 1, 2 and 3, respectively. The result for 4 and 5 points agrees with

(32.17), which has full rank. The purpose of this example is to illustrate the rank property quoted in §32.2.5: each Gauss point adds one to the rank (up to 4) since the problem is one-dimensional. The matrix (32.17) conserves linear and angular momentum; the latter property being checked by the last 3 statements of Figure 32.3. So do the reduced-integration mass matrices if $p > 1$.

To get a diagonally lumped mass matrix is trickier. Obviously the translational nodal masses must be the same as that of a bar: $\frac{1}{2}\rho A\ell$. See Figure 32.4. But there is no consensus on rotational masses. To accommodate these variations, it is convenient to leave the latter parametrized as follows

$$\bar{\mathbf{M}}_L^e = \rho A \ell \begin{bmatrix} 1/2 & 0 & 0 & 0 \\ 0 & \alpha \ell^2 & 0 & 0 \\ 0 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & \alpha \ell^2 \end{bmatrix}, \quad \alpha \geq 0. \quad (32.19)$$

Here α is a nonnegative parameter, typically between 0 and $1/50$. The choice of α has been argued in the FEM literature over several decades, but the whole discussion is largely futile. Matching the angular momentum of the beam element gyrating about its midpoint gives $\alpha = -1/24$. This violates the positivity condition stated in 32.2.4. It follows that the *best possible* α — as opposed to possible best — is zero. This choice gives, however, a singular mass matrix, which is undesirable in scenarios where a mass-inverse appears.

Remark 32.3. This result can be readily understood physically. As shown in §32.3.2, the $\frac{1}{2}\rho A\ell$ translational end node masses grossly overestimate (by a factor of 3 in fact) the angular momentum of the element. Hence adding any rotational lumped mass only makes things worse.

§32.4.3. The Plane Beam-Column

To use the foregoing results for dynamics of a plane frame structure, such as a multistory building, the 2-node bar and plane beam elements must be combined to form a plane beam-column element with six degrees of freedom in the local system. The element is then rotated into its global position. The stiffness computation process was covered in Chapter 20. Figure 32.5, which is largely a reproduction of Figure 20.6, shows this element in its local and global configurations. In this case the global and local mass matrices are not identical because of the presence of rotational DOFs; furthermore the models in the longitudinal \bar{x} and lateral \bar{y} directions are different. Consequently the distinction between local and global masses must be carefully kept.

The local consistent mass matrix $\bar{\mathbf{M}}_C^e$ is easily obtained by augmenting (32.5) and (32.17) with zeros rows and columns to fill up missing DOFs, and adding. The local-to-global transformation matrix \mathbf{T}^e is that given in Chapter 20 and reproduced here for convenience. The two matrices are

$$\bar{\mathbf{M}}_C^e = \frac{\rho A \ell}{420} \begin{bmatrix} 140 & 0 & 0 & 70 & 0 & 0 \\ 0 & 156 & 22\ell & 0 & 54 & -13\ell \\ 0 & 22\ell & 4\ell^2 & 0 & 13\ell & -3\ell^2 \\ 70 & 0 & 0 & 140 & 0 & 0 \\ 0 & 54 & 13\ell & 0 & 156 & -22\ell \\ 0 & -13\ell & -3\ell^2 & 0 & -22\ell & 4\ell^2 \end{bmatrix}, \quad \mathbf{T}^e = \begin{bmatrix} c_\varphi & s_\varphi & 0 & 0 & 0 & 0 \\ -s_\varphi & c_\varphi & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c_\varphi & s_\varphi & 0 \\ 0 & 0 & 0 & -s_\varphi & c_\varphi & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (32.20)$$

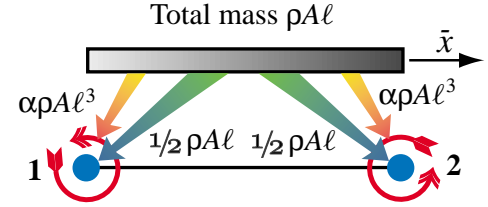


FIGURE 32.4. Direct mass lumping for 2-node Bernoulli-Euler plane beam element.

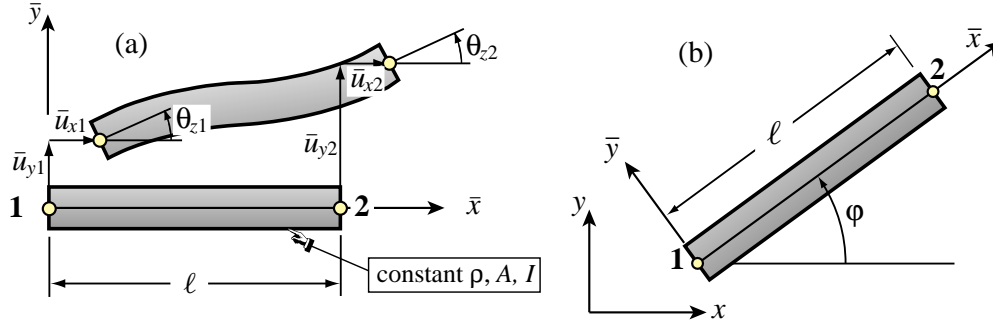


FIGURE 32.5. The 2-node plane beam-column element: (a) referred to its local system $\{\bar{x}, \bar{y}\}$; (b) referred to the global system $\{x, y\}$.

where $c_\varphi = \cos \varphi$, $s_\varphi = \sin \varphi$, and $\varphi = \angle(x, \bar{x})$, positive counterclockwise, see Figure 32.5(b). The globalized CMM is

$$\mathbf{M}_C^e = (\mathbf{T}^e)^T \bar{\mathbf{M}}_C^e \mathbf{T}^e = \frac{\rho A \ell}{420} \begin{bmatrix} 148-8c_{\varphi\varphi} & -8s_{\varphi\varphi} & -22\ell s_\varphi & 62+8c_{\varphi\varphi} & 8s_{\varphi\varphi} & 13\ell s_\varphi \\ -8s_{\varphi\varphi} & 148+8c_{\varphi\varphi} & 22\ell c_\varphi & 8s_{\varphi\varphi} & 62-8c_{\varphi\varphi} & -13\ell c_\varphi \\ -22\ell s_\varphi & 22\ell c_\varphi & 4\ell^2 & -13\ell s_\varphi & 13\ell c_\varphi & -3\ell^2 \\ 62+8c_{\varphi\varphi} & 8s_{\varphi\varphi} & -13\ell s_\varphi & 148-8c_{\varphi\varphi} & -8s_{\varphi\varphi} & 22\ell s_\varphi \\ 8s_{\varphi\varphi} & 62-8c_{\varphi\varphi} & 13\ell c_\varphi & -8s_{\varphi\varphi} & 148+8c_{\varphi\varphi} & -22\ell c_\varphi \\ 13\ell s_\varphi & -13\ell c_\varphi & -3\ell^2 & 22\ell s_\varphi & -22\ell c_\varphi & 4\ell^2 \end{bmatrix} \quad (32.21)$$

in which $c_{\varphi\varphi} = \cos 2\varphi$ and $s_{\varphi\varphi} = \sin 2\varphi$. The global and local matrices differ for arbitrary φ . It may be verified, however, that $\bar{\mathbf{M}}_C^e$ and \mathbf{M}_C^e have the same eigenvalues, since \mathbf{T}^e is orthogonal.

§32.4.4. *The Timoshenko Plane Beam

The Timoshenko plane beam model for static analysis was presented as advanced material in Chapter 13.

This model is more important for dynamics and vibration than Bernoulli-Euler, and indispensable for short transient and wave propagation analysis. (As remarked in **Notes and Bibliography** of Chapter 13, the Bernoulli-Euler beam has infinite phase velocity, because the equation of motion is parabolic, and thus useless for simulating wave propagation.) The Timoshenko beam incorporates two refinements over the Bernoulli-Euler model:

1. For both statics and dynamics: plane sections remain plane but not necessarily normal to the deflected midsurface. See Figure 32.6. This assumption allows the averaged shear distortion to be included in both strain and kinetic energies.
2. In dynamics: the rotary inertia is included in the kinetic energy.

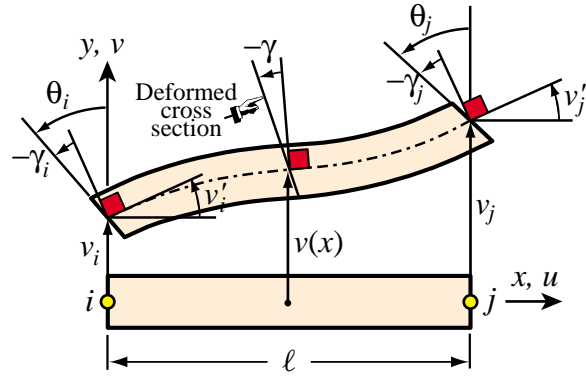


FIGURE 32.6. Kinematic assumptions of the Timoshenko plane beam element. (A reproduction of Figure 13.14 for the reader convenience.)

According to the second assumption, the kinetic energy of the Timoshenko beam element is given by

$$T = \frac{1}{2} \int_0^\ell (\rho A \dot{v}(x)^2 + \rho I_R \dot{\theta}(x)^2) dx. \quad (32.22)$$

Here I_R is the second moment of inertia to be used in the computation of the rotary inertia and $\theta = v' + \gamma$ is the cross-section rotation angle shown in Figure 32.6; $\gamma = V/(GA_s)$ being the section-averaged shear distortion. The element DOF are ordered $\mathbf{u}^e = [v_1 \ \theta_1 \ v_2 \ \theta_2]^T$. The lateral displacement interpolation is

$$v(\xi) = v_1 N_{v1}^e(\xi) + v_1' N_{v1'}^e(\xi) + v_2 N_{v2}^e(\xi) + v_2' N_{v2'}^e(\xi), \quad \xi = \frac{2x}{\ell} - 1, \quad (32.23)$$

in which the cubic interpolation functions (13.12) are used. A complication over Bernoulli-Euler is that the rotational freedoms are θ_1 and θ_2 but the interpolation (32.23) is in terms of the neutral surface end slopes: $v_1' = (dv/dx)_1 = \theta_1 - \gamma$ and $v_2' = (dv/dx)_2 = \theta_2 - \gamma$. From the analysis of §13.7 we can derive the relation

$$\begin{bmatrix} v_1' \\ v_2' \end{bmatrix} = \frac{1}{1 + \Phi} \begin{bmatrix} -\frac{\Phi}{\ell} & 1 + \frac{\Phi}{2} & \frac{\Phi}{\ell} & -\frac{\Phi}{2} \\ -\frac{\Phi}{\ell} & -\frac{\Phi}{2} & \frac{\Phi}{\ell} & 1 + \frac{\Phi}{2} \end{bmatrix} \begin{bmatrix} v_1 \\ \theta_1 \\ v_2 \\ \theta_2 \end{bmatrix}. \quad (32.24)$$

where as in (13.22) the dimensionless parameter $\Phi = 12EI/(GA_s \ell^2)$ characterizes the ratio of bending and shear rigidities. The end slopes of (32.24) are replaced into (32.23), the interpolation for θ obtained, and v and θ inserted into the kinetic energy (32.22). After lengthy algebra the CMM emerges as the sum of two contributions:

$$\mathbf{M}_C^e = \mathbf{M}_{CT}^e + \mathbf{M}_{CR}^e, \quad (32.25)$$

where \mathbf{M}_{CT} and \mathbf{M}_{CR} accounts for the translational and rotary inertia, respectively:

$$\mathbf{M}_{CT}^e = \frac{\rho A \ell}{(1 + \Phi)^2} \begin{bmatrix} \frac{13}{35} + \frac{7}{10} \Phi + \frac{1}{3} \Phi^2 & (\frac{11}{210} + \frac{11}{120} \Phi + \frac{1}{24} \Phi^2) \ell & \frac{9}{70} + \frac{3}{10} \Phi + \frac{1}{6} \Phi^2 & -(\frac{13}{420} + \frac{3}{40} \Phi + \frac{1}{24} \Phi^2) \ell \\ (\frac{1}{105} + \frac{1}{60} \Phi + \frac{1}{120} \Phi^2) \ell^2 & (\frac{13}{420} + \frac{3}{40} \Phi + \frac{1}{24} \Phi^2) \ell & -(\frac{1}{140} + \frac{1}{60} \Phi + \frac{1}{120} \Phi^2) \ell^2 & (\frac{11}{210} + \frac{11}{120} \Phi + \frac{1}{24} \Phi^2) \ell \\ \text{symmetric} & & & (\frac{1}{105} + \frac{1}{60} \Phi + \frac{1}{120} \Phi^2) \ell^2 \end{bmatrix}$$

$$\mathbf{M}_{CR}^e = \frac{\rho I_R}{(1 + \Phi)^2 \ell} \begin{bmatrix} \frac{6}{5} & (\frac{1}{10} - \frac{1}{2} \Phi) \ell & -\frac{6}{5} & (\frac{1}{10} - \frac{1}{2} \Phi) \ell \\ (\frac{2}{15} + \frac{1}{6} \Phi + \frac{1}{3} \Phi^2) \ell^2 & (-\frac{1}{10} + \frac{1}{2} \Phi) \ell & -(\frac{1}{30} + \frac{1}{6} \Phi - \frac{1}{6} \Phi^2) \ell^2 & (-\frac{1}{10} + \frac{1}{2} \Phi) \ell \\ \text{symmetric} & & & (\frac{2}{15} + \frac{1}{6} \Phi + \frac{1}{3} \Phi^2) \ell^2 \end{bmatrix} \quad (32.26)$$

Caveat: the I in $\Phi = 12EI/(GA_s \ell^2)$ is the second moment of inertia that enters in the elastic flexural elastic rigidity defined in (13.5). If the beam is homogeneous $I_R = I$, but that is not necessarily the case if, as sometimes happens, the beam has nonstructural attachments that contribute rotary inertia.

The factor of \mathbf{M}_{CR}^e can be further transformed to facilitate parametric studies by introducing $r_R^2 = I_R/A$ as cross-section gyration radius and $\Psi = r_R/\ell$ as element slenderness ratio. Then the factor $\rho I_R/((1 + \Phi)^2 \ell)$ becomes $\rho A \ell \Psi^2/(1 + \Phi)^2$. If $\Phi = 0$ and $\Psi = 0$, \mathbf{M}_{CR}^e vanishes and \mathbf{M}_{CT}^e in (32.26) reduces to (32.17).

Obtaining a diagonally lumped matrix can be done by the HRZ scheme explained in 32.6.1. The optimal lumped mass is derived in the next Chapter by the template method.

§32.4.5. *Spar and Shaft Elements

The mass matrices for these 2-node elements are very similar to those of the bar, since they can be derived from linear displacement interpolation for the CMM. The only thing that changes is the matrix factor and the end DOFs. The derivation of these elements is done as Exercises.

```

Trig3IsoPMembraneConsMass[ncoor_, ρ_, h_, {numer_, p_}] :=
Module[{i, k, x1, y1, x2, y2, x3, y3, A, Nfxy,
  tcoor, w, Me = Table[0, {6}, {6}]}],
For[k = 1, k <= Abs[p], k++,
  {{x1, y1}, {x2, y2}, {x3, y3}} = ncoor;
  A = Simplify[(x2 - x1)*(y3 - y1) - (x1 - x3)*(y1 - y2)]/2;
  {tcoor, w} = TrigGaussRuleInfo[{p, numer}, k];
  Nfxy = {Flatten[Table[{tcoor[[i]], 0}, {i, 3}]]},
    Flatten[Table[{0, tcoor[[i]]}, {i, 3}]]];
  Me += ρ*w*A*h*Transpose[Nfxy].Nfxy;
]; If[!numer, Me = Simplify[Me]]; Return[Me]
];

```

FIGURE 32.7. CMM module for 3-node linear triangle in plane stress.

§32.5. Mass Matrix Examples: Plane Stress

To illustrate the two-dimensional case, this section works out the mass matrices of two simple plane stress elements. More complicated cases are relegated to the Exercises.

§32.5.1. The Plane Stress Linear Triangle

The stiffness formulation of the 3-node triangle was discussed in Chapter 15. For the following derivations the plate is assumed to have constant mass density ρ , area A , uniform thickness h , and motion restricted to the $\{x, y\}$ plane. The six DOFs are arranged as $\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2} \ u_{x3} \ u_{y3}]^T$. The consistent mass matrix is obtained using the linear displacement interpolation (15.17). Expanding $(\mathbf{N}^e)^T \mathbf{N}^e$ gives a 6×6 matrix quadratic in the triangular coordinates. This can be integrated with the formulas (15.27) exemplified by $\int_{\Omega^e} \zeta_1^2 d\Omega = A/3$, $\int_{\Omega^e} \zeta_1 \zeta_2 d\Omega = A/6$, etc. The result is

$$\mathbf{M}_C^e = \rho h \int_{\Omega^e} \begin{bmatrix} \zeta_1 \zeta_1 & 0 & \zeta_1 \zeta_2 & 0 & \zeta_1 \zeta_3 & 0 \\ 0 & \zeta_1 \zeta_1 & 0 & \zeta_1 \zeta_2 & 0 & \zeta_1 \zeta_3 \\ \zeta_2 \zeta_1 & 0 & \zeta_2 \zeta_2 & 0 & \zeta_2 \zeta_3 & 0 \\ 0 & \zeta_2 \zeta_1 & 0 & \zeta_2 \zeta_2 & 0 & \zeta_2 \zeta_3 \\ \zeta_3 \zeta_1 & 0 & \zeta_3 \zeta_2 & 0 & \zeta_3 \zeta_3 & 0 \\ 0 & \zeta_3 \zeta_1 & 0 & \zeta_3 \zeta_2 & 0 & \zeta_3 \zeta_3 \end{bmatrix} d\Omega = \frac{\rho A h}{12} \begin{bmatrix} 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 & 0 & 1 \\ 1 & 0 & 2 & 0 & 1 & 0 \\ 0 & 1 & 0 & 2 & 0 & 1 \\ 1 & 0 & 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 1 & 0 & 2 \end{bmatrix} \quad (32.27)$$

This computation may be checked with the *Mathematica* module listed in Figure 32.7. The module is invoked as `Trig3IsoPMembraneConsMass[ncoor, ρ, h, {numer, p}]` and returns matrix `Me`. The arguments are: `ncoor` passes the node coordinate list $\{\{x1, y1\}, \{x2, y2\}, \{x3, y3\}\}$, ρ the mass density, h the plate thickness, `numer` is a logical flag set to `True` or `False` for numeric or symbolic computations, respectively, and `p` identifies the triangle integration rule as described in §24.2.1. Subordinate module `TrigGaussRuleInfo` is described in Chapter 24. Since the order of \mathbf{M}^e is 6, and each Gauss point adds two (the number of space dimensions) to the rank, a rule with 3 or more points is required to reach full rank, as can be verified by simple numerical experiments.

The lumped mass matrix is constructed by taking the total mass of the element, which is $\rho A h$, dividing it by 3 and assigning those to the corner nodes. See Figure 32.8. This process produces a

diagonal matrix:

$$\mathbf{M}_L^e = \frac{\rho Ah}{3} \mathbf{diag}[1 \ 1 \ 1 \ 1 \ 1 \ 1] = \frac{\rho Ah}{3} \mathbf{I}_6. \quad (32.28)$$

The same matrix is obtained by any other diagonalization method.

Remark 32.4. If this element is used in three dimensions (for example as membrane component of a shell element), it is necessary to insert the normal-to-the-plate z mass components in either (32.27) or (32.28). According to the invariance rule (32.14) the globalization process is trivial because \mathbf{M}_C^e or \mathbf{M}_L^e becomes RBD on grouping the element DOFs by component. Thus the local element mass matrix repeats in the global frame.

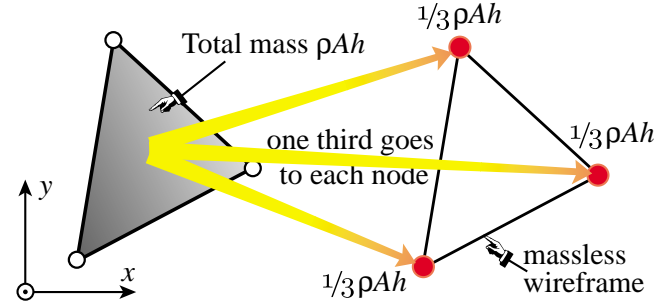


FIGURE 32.8. DLMM for 3-node triangular element.

```
Quad4IsoPMembraneCMass[ncoor_,rho_,h_,{numer_,p_}]:=
Module[{i,k,Nf,dNx,dNy,Jdet,Nfxy,qcoor,w,Me=Table[0,{8},{8}]},
For[k=1,k<=p*p,k++,
{qcoor,w}=QuadGaussRuleInfo[{p,numer},k];
{Nf,dNx,dNy,Jdet}=Quad4IsoPShapeFunDer[ncoor,qcoor];
Nfxy={Flatten[Table[{Nf[[i]],0},{i,4]}],
Flatten[Table[{0,Nf[[i]]},{i,4]}]};
Me+=(rho*w*Jdet*h/2)*Transpose[Nfxy].Nfxy;
];If[!numer,Me=Simplify[Me]];Return[Me]
];
```

FIGURE 32.9. CMM module for 4-node bilinear quad in plane stress.

§32.5.2. Four-Node Bilinear Quadrilateral

Module Quad4IsoPMembraneConsMass, listed in Figure 32.9, returns the CMM of a 4-node bilinear quadrilateral under plane stress, moving in the $\{x, y\}$ plane. The plate is homogeneous with density ρ and constant thickness h . The arguments are similar to those described for the linear triangle, except that the quadrature rule pertains to quadrilaterals, and is specified as described in Chapter 23. The subordinate modules QuadGaussRuleInfo (shape functions) and Quad4IsoPShapeFunDer (Gauss quadrature information) are described in that Chapter.

The integration is carried out numerically using a $p \times p$ Gauss product rule, with p specified as argument. Testing the module on a rectangular element of dimensions $\{a, b\}$ returns the following CMMs for the 1×1 and 2×2 Gauss rules:

$$\mathbf{M}_{C1 \times 1}^e = \frac{\rho abh}{32} \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{M}_{C2 \times 2}^e = \frac{\rho abh}{72} \begin{bmatrix} 4 & 0 & 2 & 0 & 1 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 & 0 & 1 & 0 & 2 \\ 2 & 0 & 4 & 0 & 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 4 & 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 & 4 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 & 0 & 4 & 0 & 2 \\ 2 & 0 & 1 & 0 & 2 & 0 & 4 & 0 \\ 0 & 2 & 0 & 1 & 0 & 2 & 0 & 4 \end{bmatrix}. \quad (32.29)$$

The mass given by 1-point integration has rank 2 and 6 zero eigenvalues, and thus rank-deficient by 6. The mass given by the 2×2 rule is rank-sufficient and positive definite. Either matrix repeats on globalization. Using $p > 2$ returns $\mathbf{M}_{C2 \times 2}^e$. The DLMM is obtained by assigning one fourth of the total element mass ρabh to each freedom.

§32.6. Mass Diagonalization Methods

The construction of consistent mass matrix (CMM) is fully defined by the choice of kinetic energy functional and shape functions. No procedural deviation is possible. On the other hand the construction of a diagonally lumped mass matrix (DLMM) is not a unique process, except for very simple elements in which the lumping is fully defined by conservation and symmetry considerations. A consequence of this ambiguity is that various methods have been proposed in the literature, ranging from heuristic through more scientific. This subsection gives a quick overview of the two more important methods.

§32.6.1. HRZ Lumping

This scheme is acronymed after the authors of [134]. It produces a DLMM given the CMM. Let M^e denote the total element mass. The procedure is as follows.

1. For each coordinate direction, select the DOFs that contribute to motion in that direction. From this set, separate translational DOF and rotational DOF subsets.
2. Add up the CMM diagonal entries pertaining to the translational DOF subset only. Call the sum S .
3. Apportion M^e to DLMM entries of both subsets on dividing the CMM diagonal entries by S .
4. Repeat for all coordinate directions.

Example 32.1. The see HRZ in action, consider the 3-node prismatic bar with CMM given by (32.15). Only one direction (x) is involved and all DOFs are translational. Excluding the factor $\rho A \ell / 30$, which does not affect the results, the diagonal entries are 4, 4 and 16, which add up to $S = 24$. Apportion the total element mass $\rho A \ell$ to nodes with weights $4/S = 1/6$, $4/S = 1/6$ and $16/S = 2/3$. The result is the DLMM (32.16).

Example 32.2. Next consider the 2-node Bernoulli-Euler plane beam element. Again only one direction (y) is involved but now there are translational and rotational freedoms. Excluding the factor $\rho A \ell / 420$, the diagonal entries of the CMM (32.17), are 156, $4\ell^2$, 156 and $4\ell^2$. Add the translational DOF entries: $S = 156 + 156 = 312$. Apportion the element mass $\rho A \ell$ to the four DOFs with weights $156/312 = 1/2$, $4\ell^2/312 = \ell^2/78$, $156/312 = 1/2$ and $4\ell^2/312 = \ell^2/78$. The result is the DLMM (32.19) with $\alpha = 1/78$.

The procedure is heuristic but widely used on account of three advantages: easy to explain and implement, applicable to any element as long as a CMM is available, and retaining nonnegativity. The last attribute is particularly important: it means that the DLMM is physically admissible, precluding numerical instability headaches. As a general assessment, it gives reasonable results if the element has only translational freedoms. If there are rotational freedoms the results can be poor compared to templates.

§32.6.2. Lobatto Lumping

A DLMM with n_F^e diagonal entries m_i is formally equivalent to a numerical integration formula with n_F^e points for the element kinetic energy:

$$T^e = \sum_{i=1}^{n_F^e} m_i T_i, \quad \text{where} \quad T_i = \frac{1}{2} \dot{u}_i^2 \quad (32.30)$$

Table 32.1. One-Dimensional Lobatto Integration Rules

<i>Points</i>	<i>Abscissas</i> $\xi_i \in [-1, 1]$	<i>Weights</i> w_i	<i>Comments</i>
2	$-\xi_1 = 1 = \xi_2$	$w_1 = w_2 = 1$	Trapezoidal rule
3	$-\xi_1 = 1 = \xi_3, \xi_2 = 0$	$w_1 = w_3 = \frac{1}{3}, w_2 = \frac{4}{3}$	Simpson's rule
4	$-\xi_1 = 1 = \xi_4, -\xi_2 = 1/\sqrt{5} = \xi_3$	$w_1 = w_4 = \frac{1}{6}, w_2 = w_3 = \frac{5}{6}$	Interior points at ± 0.447214
The 4-point Lobatto rule should not be confused with “Simpson's Three-Eighths rule,” a Newton-Coates quadrature that has equidistant abscissas: $-\xi_1 = \xi_2 = 1, -\xi_2 = \xi_3 = \frac{1}{3}, w_1 = w_2 = \frac{1}{4}, w_2 = w_3 = \frac{3}{4}$.			
For $4 < n \leq 10$, which is rarely important in FEM work, see Table 25.6 of [1].			

If the element is one-dimensional and has only translational DOF, (32.30) can be placed in correspondence with the so-called *Lobatto quadrature* in numerical analysis.⁴ A Lobatto rule is a 1D Gaussian quadrature formula in which the endpoints of the interval $\xi \in [-1, 1]$ are sample points. If the formula has $p \geq 2$ abscissas, only $p - 2$ of those are free. Abscissas are symmetric about the origin $\xi = 0$ and all weights are positive. The general form is

$$\int_{-1}^1 f(\xi) d\xi = w_1 f(-1) + w_p f(1) + \sum_{i=2}^{p-1} w_i f(\xi_i). \quad (32.31)$$

The rules for $p = 2, 3, 4$ are collected in Table 32.1. Comparing (32.30) with (32.31) clearly indicates that if the nodes of a 1D element are placed at the Lobatto abscissas, the diagonal masses m_i are simply the weights w_i . This fact was first noted by Fried and Malkus [106] and further explored in [168,169]. For the type of elements noted, the equivalence works well for $p = 2, 3$. For $p = 4$ a minor difficulty arises: the interior Lobatto points are not at the thirdpoints, as can be seen in Table 32.1. If instead the element has nodes at the thirdpoints $\xi = \pm \frac{1}{3}$, one must switch to the “Simpson three-eighths” rule also listed in that Table, and adjust the masses accordingly.

As a generalization to multiple dimensions, for conciseness we call “FEM Lobatto quadrature” one in which the *connected nodes* of an element are sample points of an integration rule. The equivalence with (32.30) still holds. But the method runs into some difficulties:

Zero or Negative Masses. If one insists in higher order accuracy, the weights of 2D and 3D Lobatto rules are not necessarily positive. See for example the case of the 6-node triangle in the Exercises. This shortcoming can be alleviated, however, by accepting lower accuracy, or by sticking to product rules in geometries that permit them. See Exercise 32.19.

Rotational Freedoms. If the element has rotational DOF, Lobatto rules do not exist. Any attempt to transform rules such as (32.31) to node rotations inevitably leads to coupling.

Varying Properties. If the element is nonhomogeneous or has varying properties (for instance, a tapered bar element) the construction of Lobatto rules runs into difficulties, since the problem effectively becomes the construction of a quadrature formula with non-unity kernel.

As a general assessment, the Lobatto integration technique seems advantageous only when the mass diagonalization problem happens to fit a rule already available in handbooks such as [231].

⁴ Also called Radau quadrature by some authors, e.g. [39]. However the handbook [1, p. 888] says that Lobatto and Radau rules are slightly different.

Notes and Bibliography

The first appearance of a mass matrix in a journal article occurs in two early-1930s papers⁵ by Duncan and Collar [62,63]. There it is called “inertia matrix” and denoted by $[m]$. The original example [62, p. 869] displays the 3×3 diagonal mass of a triple pendulum. In the book [105] the notation changes to A .

Diagonally lumped mass matrices (DLMM) dominate pre-1963 work. Computational simplicity was not the only reason. Direct lumping gives an obvious way to account for *nonstructural masses* in simple discrete models of the spring-dashpot-pointmass variety. For example, in a multistory building “stick model” wherein each floor is treated as one DOF in lateral sway under earthquake or wind action, it is natural to take the entire mass of the floor (including furniture, isolation, etc.) and assign it to that freedom. Nondiagonal masses pop-up occasionally in aircraft matrix analysis — e.g. wing oscillations in [105, §10.11] — as a result of measurements. As such they necessarily account for nonstructural masses due to fuel, control equipment, etc.

The formulation of the consistent mass matrix (CMM) by Archer [11,12] was a major advance. All CMMs displayed in §32.3 were first derived in those papers. The underlying idea is old. In fact it follows directly from the 18th-Century Lagrange dynamic equations [154], a proven technique to produce generalized masses. If T is the kinetic energy of a discrete system and $\dot{u}_i(x_i)$ the velocity field defined by the nodal velocities collected in $\dot{\mathbf{u}}$, the master (system-level) \mathbf{M} can be generally defined as the Hessian of T with respect to nodal velocities:

$$T = \frac{1}{2} \int_{\Omega} \rho \dot{u}_i \dot{u}_i d\Omega, \quad u_i = u_i(\dot{\mathbf{u}}), \quad \mathbf{M} = \frac{\partial^2 T}{\partial \dot{\mathbf{u}} \partial \dot{\mathbf{u}}}. \quad (32.32)$$

This matrix is constant if T is quadratic in $\dot{\mathbf{u}}$. Two key decisions remain before this could be used in FEM.

Localization: (32.32) is applied element by element, and the master \mathbf{M} assembled by the standard DSM steps.

Interpolation: the velocity field is defined by the same element shape functions as the displacement field.

These had to wait until three things became well established by the early 1960s: (i) the Direct Stiffness Method, (ii) the concept of shape functions, and (iii) the FEM connection to Rayleigh-Ritz. The critical ingredient (iii) was established in Melosh’s thesis [178] under Harold Martin. The link to dynamics was closed with Archer’s contributions, and CMM became a staple of FEM. But only a loose staple. Problems persisted:

- (a) Nonstructural masses are not naturally handled by CMM. In systems such as ships or aircraft, the structural mass is only a fraction (10 to 20%) of the total.
- (b) It is inefficient in some solution processes, notably explicit dynamics.⁶
- (c) It may not give the best results compared to other alternatives.⁷
- (d) For elements derived outside the assumed-displacement framework, the stiffness shape functions may be unknown or be altogether missing.

Problem (a) can be addressed by constructing “rigid mass elements” accounting for inertia (and possibly gravity or centrifugal forces) but no stiffness. Nodes of these elements must be linked to structural (elastic)

⁵ The brain behind these developments, which launched Matrix Structural Analysis as narrated in Appendix H, was Arthur Collar. But in the hierarchically rigid British system of the time his name, on account of less seniority, had to be last.

⁶ In explicit time integration, accelerations are computed at the global level as $\mathbf{M}^{-1}\mathbf{f}_d$, where \mathbf{f}_d is the effective dynamic force. This is efficient if \mathbf{M} is diagonal (and nonsingular). If \mathbf{M} is the CMM, a system of equations has to be solved; moreover the stable timestep generally decreases.

⁷ If \mathbf{K} results from a conforming displacement interpolation, pairing it with the CMM is a form of Rayleigh-Ritz, and thus guaranteed to provide upper bounds on natural frequencies. This is not necessarily a good thing. In practice it is observed that errors increase rapidly as one moves up the frequency spectrum. If the response is strongly influenced by intermediate and high frequencies, as in wave propagation dynamics, the CMM may give extremely bad results.

nodes by MFC constraints that enforce kinematic constraints. This is more of an implementation issue than a research topic, although numerical difficulties typical of rigid body dynamics may crop up.

Problems (b,c,d) can be attacked by parametrization. The father of NASTRAN, Dick MacNeal, was the first to observe [161,167] that averaging the DLMM and CMM of the 2-node bar element produced better results than using either alone. This idea was further studied by Belytschko and Mullen [22] using Fourier analysis. Krieg and Key [152] had emphasized that in transient analysis the introduction of a time discretization operator brings new compensation phenomena, and consequently the time integrator and the mass matrix should not be chosen separately.

A good discussion of mass diagonalization schemes can be found in the textbook by Cook et al. [50].

The template approach addresses the problems by allowing and encouraging full customization of the mass to the problem at hand. It was first described in [85,88] for a Bernoulli-Euler plane beam using Fourier methods. It is presented in more generality in the following Chapter, where it is applied to other elements. The general concept of template as parametrized form of FEM matrices is discussed in [84].

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 32

Lumped and Consistent Mass Matrices

EXERCISE 32.1 [A/C:15]. Derive the consistent mass for a 2-node tapered bar element of length ℓ and constant mass density ρ , moving along its axis x , if the cross section area varies as $A = \frac{1}{2}A_1(1-\xi) + \frac{1}{2}A_2(1+\xi)$. Obtain also the DLMM by the HRZ scheme. Show that

$$\mathbf{M}_C^e = \frac{\rho\ell}{12} \begin{bmatrix} 3A_1 + A_2 & A_1 + A_2 \\ A_1 + A_2 & A_1 + 3A_2 \end{bmatrix}, \quad \mathbf{M}_L^e = \frac{\rho\ell}{8(A_1 + A_2)} \begin{bmatrix} 3A_1 + A_2 & 0 \\ 0 & A_1 + 3A_2 \end{bmatrix}. \quad (\text{E32.1})$$

EXERCISE 32.2 [A:15]. Dr. I. M. Clueless proposed a new diagonally lumped mass scheme for the 2-node bar: placing one half of the element mass $\rho A\ell$ at each Gauss point $\xi = \pm 1/\sqrt{3}$ of a two-point rule. His argument is that this configuration conserves total mass and angular momentum, which is true. Explain in two sentences why the idea is worthless.

EXERCISE 32.3 [A:25]. Extend the result (32.13) to three dimensions. The element has n^e nodes and three translational DOF per node. Arrange the element DOFs by component: \mathbf{M}^e is RBD, formed by three $n^e \times n^e$ submatrices $\tilde{\mathbf{M}}$, which are left arbitrary. For \mathbf{T}^e , assume nine blocks $t_{ij}\mathbf{I}$ for $\{i, j\} = 1, 2, 3$, where t_{ij} are the direction cosines of the local system $\{\bar{x}, \bar{y}, \bar{z}\}$ with respect to the global system $\{x, y, z\}$, and \mathbf{I} is the $n^e \times n^e$ identity matrix. Hint: use the orthogonality properties of the t_{ij} : $t_{ij}t_{ik} = \delta_{jk}$.

EXERCISE 32.4 [A:20]. Show that the local and global mass matrices of elements with only translational DOFs repeat under these constraints: all local DOFs are referred to the same local frame, and all global DOF are referred to the same global frame. Hints: the kinetic energy must be the same in both frames, and the local-to-global transformation matrix is orthogonal.

EXERCISE 32.5 [A/C:25]. Derive the CMM and DLMM for the 2-node spar and shaft elements derived in Chapter 6. Assume that the elements are prismatic with constant properties along their length ℓ .

EXERCISE 32.6 [A/C:20]. Derive the consistent mass for a 2-node Bernoulli-Euler tapered plane beam element of length ℓ if the cross section area varies as $A = A_i(1-\xi)/2 + A_j(1+\xi)/2$ while the mass density ρ is constant. Show that

$$\mathbf{M}_C^e = \frac{\rho\ell}{840} \begin{bmatrix} 240A_i + 72A_j & 2(15A_i + 7A_j)\ell & 54(A_i + A_j) & -2(7A_i + 6A_j)\ell \\ 2(15A_i + 7A_j)\ell & (5A_i + 3A_j)\ell^2 & 2(6A_i + 7A_j)\ell & -3(A_i + A_j)\ell^2 \\ 54(A_i + A_j) & 2(6A_i + 7A_j)\ell & 72A_i + 240A_j & -2(7A_i + 15A_j)\ell \\ -2(7A_i + 6A_j)\ell & -3(A_i + A_j)\ell^2 & -2(7A_i + 15A_j)\ell & (3A_i + 5A_j)\ell^2 \end{bmatrix}. \quad (\text{E32.2})$$

EXERCISE 32.7 [A:15]. Derive the local CMM and DLMM for the 2-node spar element derived in Chapter 6. Assume that the element is prismatic with constant properties along their length ℓ , and that can move in 3D space.

EXERCISE 32.8 [A:15]. Derive the local CMM and DLMM for the 2-node spar shaft derived in Chapter 6. Assume that the element is prismatic with constant properties along their length ℓ , and that can move in 3D space.

EXERCISE 32.9 [C:25]. Write a *Mathematica* script to verify the result (32.25)–(32.26). (Don't try this by hand.)

EXERCISE 32.10 [A/C:20]. Derive the consistent mass for a plane stress 3-node linear triangle with constant mass density ρ and linearly varying thickness h defined by the corner values h_1 , h_2 and h_3 .

EXERCISE 32.11 [A/C:20]. Derive the consistent mass for a plane stress 6-node quadratic triangle with constant mass density ρ and uniform thickness h , moving in the $\{x, y\}$ plane. The triangle has straight sides and side nodes placed at the midpoints; consequently the metric (and thus the Jacobian) is constant. Show that with the element DOFs arranged $\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ \dots \ u_{y6}]^T$, the CMM is [68, p. 35]

$$\mathbf{M}_C^e = \frac{\rho h A}{180} \begin{bmatrix} 6 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 & 0 \\ 0 & 6 & 0 & -1 & 0 & -1 & 0 & 0 & 0 & -4 & 0 & 0 \\ -1 & 0 & 6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -4 & 0 \\ 0 & -1 & 0 & 6 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & -4 \\ -1 & 0 & -1 & 0 & 6 & 0 & -4 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & -1 & 0 & 6 & 0 & -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -4 & 0 & 32 & 0 & 16 & 0 & 16 & 0 \\ 0 & 0 & 0 & 0 & 0 & -4 & 0 & 32 & 0 & 16 & 0 & 16 \\ -4 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 32 & 0 & 16 & 0 \\ 0 & -4 & 0 & 0 & 0 & 0 & 0 & 16 & 0 & 32 & 0 & 16 \\ 0 & 0 & -4 & 0 & 0 & 0 & 16 & 0 & 16 & 0 & 32 & 0 \\ 0 & 0 & 0 & -4 & 0 & 0 & 0 & 16 & 0 & 16 & 0 & 32 \end{bmatrix} \quad (\text{E32.3})$$

and check that all eigenvalues are positive. Hint: use the shape functions of Chapter 24 and the 6 or 7-point triangle integration rule.

EXERCISE 32.12 [A:20=5+15]. Assuming the result (E32.3), use the HRZ and Lobatto integration methods to get two DLMMs for the six-node plane stress triangle. Show that HRZ gives

$$\mathbf{M}_L^e = \frac{\rho h A}{57} \mathbf{diag}[3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 16 \ 16 \ 16 \ 16 \ 16 \ 16], \quad (\text{E32.4})$$

whereas Lobatto integration, using the triangle midpoint rule (24.6), gives

$$\mathbf{M}_L^e = \frac{\rho h A}{3} \mathbf{diag}[0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1]. \quad (\text{E32.5})$$

EXERCISE 32.13 [A/C:30]. Derive the consistent and the HRZ-diagonalized lumped mass matrices for a plane stress 10-node cubic triangle with constant mass density ρ and uniform thickness h , moving in the $\{x, y\}$ plane. The triangle has straight sides, side nodes placed at the thirdpoints and node 0 at the centroid. Consequently the metric (and thus the Jacobian) is constant. Show that with the element DOFs arranged $\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ \dots \ u_{y0}]^T$, the CMM is [68, p. 35]

$$\mathbf{M}_C^e = \frac{\rho h A}{6720} \begin{bmatrix} 76 & 0 & 11 & 0 & 11 & 0 & 18 & 0 & 0 & 0 & 27 & 0 & 27 & 0 & 0 & 0 & 18 & 0 & 36 & 0 \\ 0 & 76 & 0 & 11 & 0 & 11 & 0 & 18 & 0 & 0 & 0 & 27 & 0 & 27 & 0 & 0 & 0 & 18 & 0 & 36 \\ 11 & 0 & 76 & 0 & 11 & 0 & 0 & 0 & 18 & 0 & 18 & 0 & 0 & 0 & 27 & 0 & 27 & 0 & 36 & 0 \\ 0 & 11 & 0 & 76 & 0 & 11 & 0 & 0 & 0 & 18 & 0 & 18 & 0 & 0 & 0 & 27 & 0 & 27 & 0 & 36 \\ 11 & 0 & 11 & 0 & 76 & 0 & 27 & 0 & 27 & 0 & 0 & 18 & 0 & 18 & 0 & 0 & 0 & 0 & 36 & 0 \\ 0 & 11 & 0 & 11 & 0 & 76 & 0 & 27 & 0 & 27 & 0 & 0 & 18 & 0 & 18 & 0 & 0 & 0 & 0 & 36 \\ 18 & 0 & 0 & 0 & 27 & 0 & 540 & 0 & -189 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 270 & 0 & 162 & 0 \\ 0 & 18 & 0 & 0 & 0 & 27 & 0 & 540 & 0 & -189 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 270 & 0 & 162 \\ 0 & 0 & 18 & 0 & 27 & 0 & -189 & 0 & 540 & 0 & 270 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 162 & 0 \\ 0 & 0 & 0 & 18 & 0 & 27 & 0 & -189 & 0 & 540 & 0 & 270 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 162 \\ 27 & 0 & 18 & 0 & 0 & 0 & -135 & 0 & 270 & 0 & 540 & 0 & -189 & 0 & -135 & 0 & -54 & 0 & 162 & 0 \\ 0 & 27 & 0 & 18 & 0 & 0 & 0 & -135 & 0 & 270 & 0 & 540 & 0 & -189 & 0 & -135 & 0 & -54 & 0 & 162 \\ 27 & 0 & 0 & 0 & 18 & 0 & -54 & 0 & -135 & 0 & -189 & 0 & 540 & 0 & 270 & 0 & -135 & 0 & 162 & 0 \\ 0 & 27 & 0 & 0 & 0 & 18 & 0 & -54 & 0 & -135 & 0 & -189 & 0 & 540 & 0 & 270 & 0 & -135 & 0 & 162 \\ 0 & 0 & 27 & 0 & 18 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 270 & 0 & 540 & 0 & -189 & 0 & 162 & 0 \\ 0 & 0 & 0 & 27 & 0 & 18 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & 270 & 0 & 540 & 0 & -189 & 0 & 162 \\ 18 & 0 & 27 & 0 & 0 & 0 & 270 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & -189 & 0 & 540 & 0 & 162 & 0 \\ 0 & 18 & 0 & 27 & 0 & 0 & 0 & 270 & 0 & -135 & 0 & -54 & 0 & -135 & 0 & -189 & 0 & 540 & 0 & 162 \\ 36 & 0 & 36 & 0 & 36 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 1944 & 0 \\ 0 & 36 & 0 & 36 & 0 & 36 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 162 & 0 & 1944 \end{bmatrix} \quad (\text{E32.6})$$

```

Quad8IsoPMembraneConsMass[ncoor_, ρ_, h_, {numer_, p_}]:=
Module[{i,k,Nf,dNx,dNy,Jdet,Nfxy,qcoor,w,Me=Table[0,{16},{16}]],
  For[k=1, k<=p*p, k++,
    {qcoor,w}= QuadGaussRuleInfo[{p,numer},k];
    {Nf,dNx,dNy,Jdet}= Quad8IsoPShapeFunDer[ncoor,qcoor];
    Nfxy={Flatten[Table[{Nf[[i]], 0},{i,8}]],
          Flatten[Table[{0,Nf[[i]]},{i,8}]]};
    Me+=(ρ*w*Jdet*h/2)*Transpose[Nfxy].Nfxy;
  ]; If[!numer,Me=Simplify[Me]]; Return[Me]
];

```

FIGURE E32.1. CMM module for 8-node serendipity quadrilateral in plane stress.

Check that all eigenvalues are positive. Show that the HRZ-diagonalized LMM is

$$\mathbf{M}_L^e = \frac{\rho h A}{1353} \mathbf{diag}[19 \ 19 \ 19 \ 19 \ 19 \ 19 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 135 \ 486 \ 486]. \quad (\text{E32.7})$$

EXERCISE 32.14 [A/C:20]. Derive the consistent mass for a plane stress 4-node bilinear rectangle with constant mass density ρ , uniform thickness h and side dimensions a and b .

EXERCISE 32.15 [A:15]. Explain why, for a plane stress element moving in the $\{x, y\}$ plane, each Gauss integration point adds 2 to the rank of the CMM. What would happen in three dimensions?

EXERCISE 32.16 [A:25]. For *any* geometry, the CMM of the 4-node bilinear quadrilateral in plane stress (assumed homogeneous and of constant thickness) repeats once the $p \times p$ Gauss integration rule verifies $p \geq 2$. Explain why.

EXERCISE 32.17 [C:15]. Using the script of Figure E32.1 derive the consistent and HRZ-diagonalized lumped mass matrices for a 8-node serendipity quadrilateral specialized to a rectangle of dimensions $\{a, b\}$ that moves in the $\{x, y\}$ plane. Assume constant mass density ρ and uniform thickness h . Show that

$$\mathbf{M}_C^e = \frac{\rho abh}{360} \begin{bmatrix} 6 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & -6 & 0 & -8 & 0 & -8 & 0 & -6 & 0 \\ 0 & 6 & 0 & 2 & 0 & 3 & 0 & 2 & 0 & -6 & 0 & -8 & 0 & -8 & 0 & -6 \\ 2 & 0 & 6 & 0 & 2 & 0 & 3 & 0 & -6 & 0 & -6 & 0 & -8 & 0 & -8 & 0 \\ 0 & 2 & 0 & 6 & 0 & 2 & 0 & 3 & 0 & -6 & 0 & -6 & 0 & -8 & 0 & -8 \\ 3 & 0 & 2 & 0 & 6 & 0 & 2 & 0 & -8 & 0 & -6 & 0 & -6 & 0 & -8 & 0 \\ 0 & 3 & 0 & 2 & 0 & 6 & 0 & 2 & 0 & -8 & 0 & -6 & 0 & -6 & 0 & -8 \\ 2 & 0 & 3 & 0 & 2 & 0 & 6 & 0 & -8 & 0 & -8 & 0 & -6 & 0 & -6 & 0 \\ 0 & 2 & 0 & 3 & 0 & 2 & 0 & 6 & 0 & -8 & 0 & -8 & 0 & -6 & 0 & -6 \\ -6 & 0 & -6 & 0 & -8 & 0 & -8 & 0 & 32 & 0 & 20 & 0 & 16 & 0 & 20 & 0 \\ 0 & -6 & 0 & -6 & 0 & -8 & 0 & -8 & 0 & 32 & 0 & 20 & 0 & 16 & 0 & 20 \\ -8 & 0 & -6 & 0 & -6 & 0 & -8 & 0 & 20 & 0 & 32 & 0 & 20 & 0 & 16 & 0 \\ 0 & -8 & 0 & -6 & 0 & -6 & 0 & -8 & 0 & 20 & 0 & 32 & 0 & 20 & 0 & 16 \\ -8 & 0 & -8 & 0 & -6 & 0 & -6 & 0 & 16 & 0 & 20 & 0 & 32 & 0 & 20 & 0 \\ 0 & -8 & 0 & -8 & 0 & -6 & 0 & -6 & 0 & 16 & 0 & 20 & 0 & 32 & 0 & 20 \\ -6 & 0 & -8 & 0 & -8 & 0 & -6 & 0 & 20 & 0 & 16 & 0 & 20 & 0 & 32 & 0 \\ 0 & -6 & 0 & -8 & 0 & -8 & 0 & -6 & 0 & 20 & 0 & 16 & 0 & 20 & 0 & 32 \end{bmatrix} \quad (\text{E32.8})$$

Which is the minimum integration rule required to get this matrix? Show that HRZ gives

$$\mathbf{M}_L^e = \frac{\rho abh}{76} \mathbf{diag}[3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 3 \ 16 \ 16 \ 16 \ 16 \ 16 \ 16 \ 16 \ 16]. \quad (\text{E32.9})$$

```

Quad9IsoPMembraneConsMass[ncoor_, ρ_, h_, {numer_, p_}] :=
Module[{i, k, Nf, dNx, dNy, Jdet, Nfxy, qcoor, w, Me = Table[0, {18}, {18}]},
  For [k=1, k<=p*p, k++,
    {qcoor, w} = QuadGaussRuleInfo[{p, numer}, k];
    {Nf, dNx, dNy, Jdet} = Quad9IsoPShapeFunDer[ncoor, qcoor];
    Nfxy = {Flatten[Table[{Nf[[i]], 0}, {i, 9}]],
            Flatten[Table[{0, Nf[[i]]}, {i, 9}]]};
    Me += (ρ*w*Jdet*h/2)*Transpose[Nfxy].Nfxy;
  ]; If[!numer, Me = Simplify[Me]]; Return[Me]
];

```

FIGURE E32.2. CMM module for 9-node biquadratic quadrilateral in plane stress.

EXERCISE 32.18 [C:15]. Using the script of Figure E32.2 derive the consistent and HRZ-diagonalized lumped mass matrices for a 9-node biquadratic quadrilateral specialized to a rectangle of dimensions $\{a, b\}$ that moves in the $\{x, y\}$ plane. Assume constant mass density ρ and uniform thickness h . Show that

$$\mathbf{M}_C^e = \frac{\rho abh}{1800} \begin{bmatrix} 16 & 0 & -4 & 0 & 1 & 0 & -4 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 4 & 0 \\ 0 & 16 & 0 & -4 & 0 & 1 & 0 & -4 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 4 \\ -4 & 0 & 16 & 0 & -4 & 0 & 1 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 4 & 0 \\ 0 & -4 & 0 & 16 & 0 & -4 & 0 & 1 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 4 \\ 1 & 0 & -4 & 0 & 16 & 0 & -4 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & 4 & 0 \\ 0 & 1 & 0 & -4 & 0 & 16 & 0 & -4 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & 4 \\ -4 & 0 & 1 & 0 & -4 & 0 & 16 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & 4 & 0 \\ 0 & -4 & 0 & 1 & 0 & -4 & 0 & 16 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & 4 \\ 8 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 64 & 0 & 4 & 0 & -16 & 0 & 4 & 0 & 32 & 0 \\ 0 & 8 & 0 & 8 & 0 & -2 & 0 & -2 & 0 & 64 & 0 & 4 & 0 & -16 & 0 & 4 & 0 & 32 \\ -2 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & 4 & 0 & 64 & 0 & 4 & 0 & -16 & 0 & 32 & 0 \\ 0 & -2 & 0 & 8 & 0 & 8 & 0 & -2 & 0 & 4 & 0 & 64 & 0 & 4 & 0 & -16 & 0 & 32 \\ -2 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & -16 & 0 & 4 & 0 & 64 & 0 & 4 & 0 & 32 & 0 \\ 0 & -2 & 0 & -2 & 0 & 8 & 0 & 8 & 0 & -16 & 0 & 4 & 0 & 64 & 0 & 4 & 0 & 32 \\ 8 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 4 & 0 & -16 & 0 & 4 & 0 & 64 & 0 & 32 & 0 \\ 0 & 8 & 0 & -2 & 0 & -2 & 0 & 8 & 0 & 4 & 0 & -16 & 0 & 4 & 0 & 64 & 0 & 32 \\ 4 & 0 & 4 & 0 & 4 & 0 & 4 & 0 & 32 & 0 & 32 & 0 & 32 & 0 & 32 & 0 & 256 & 0 \\ 0 & 4 & 0 & 4 & 0 & 4 & 0 & 4 & 0 & 32 & 0 & 32 & 0 & 32 & 0 & 32 & 0 & 256 \end{bmatrix} \quad (\text{E32.10})$$

Which is the minimum integration rule required to get this matrix? Show that HRZ gives

$$\mathbf{M}_L^e = \frac{\rho abh}{36} \mathbf{diag}[1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 4 \ 4 \ 4 \ 4 \ 4 \ 4 \ 4 \ 4 \ 16 \ 16]. \quad (\text{E32.11})$$

EXERCISE 32.19 A:25] Two Lobatto formulas that may be used for the 9-node biquadratic quadrilateral may be found listed in [231, p. 244–245]. Translated to a rectangle geometry of area A and FEM quadrilateral-coordinate notation, they are

$$\begin{aligned} \frac{1}{A} \int_{\Omega^e} f(\xi, \eta) d\Omega &= \frac{16}{36} f(0, 0) + \frac{4}{36} [f(0, -1) + f(1, 0) + f(0, 1) + f(-1, 0)] \\ &\quad + \frac{1}{36} [f(-1, -1) + f(1, -1) + f(1, 1) + f(-1, 1)], \end{aligned} \quad (\text{E32.12})$$

$$\begin{aligned} \frac{1}{A} \int_{\Omega^e} f(\xi, \eta) d\Omega &= \frac{20}{48} f(0, 0) + \frac{6}{48} [f(0, -1) + f(1, 0) + f(0, 1) + f(-1, 0)] \\ &\quad + \frac{1}{48} [f(-1, -1) + f(1, -1) + f(1, 1) + f(-1, 1)]. \end{aligned} \quad (\text{E32.13})$$

(E32.12) is a product Simpson formula, whereas (E32.13) was derived by Albrecht and Collatz in 1953. Explain (i) how these formulas can be used to set up a DLMM for a quadrilateral of arbitrary geometry, and (ii) whether (E32.12) coalesces with the HRZ result in the case of a rectangle.

33

Customized Mass Matrices of 1D Elements

TABLE OF CONTENTS

	Page
§33.1. Introduction	33–3
§33.2. Customization Scenarios	33–3
§33.3. Parametrization Techniques	33–4
§33.4. Two-Node Bar Element	33–5
§33.4.1. Best μ by Angular Momentum Preservation	33–6
§33.4.2. Best μ by Fourier Analysis	33–6
§33.4.3. *Best μ By Modified Equation	33–9
§33.5. Three-Node Bar Element	33–10
§33.5.1. Patch Equations	33–11
§33.5.2. Fourier Analysis	33–12
§33.5.3. Customization	33–14
§33.6. The Bernoulli-Euler Beam	33–16
§33.7. *Two-Node Timoshenko Beam Element	33–18
§33.7.1. *Continuum Analysis	33–18
§33.7.2. *Beam Element	33–21
§33.7.3. *Setting Up the Mass Template	33–21
§33.7.4. *Full Mass Parametrization	33–21
§33.7.5. *Block-Diagonal Mass Parametrization	33–23
§33.7.6. *Fourier Analysis	33–23
§33.7.7. *Template Instances	33–24
§33.7.8. *Vibration Analysis Example	33–26
§33. Notes and Bibliography	33–29
§33. References	33–29
§33. Exercises	33–30

§33.1. Introduction

Two standard procedures for building finite element mass matrices have been covered in the previous Chapter. Those lead to consistent and diagonally-lumped forms. These models are denoted by \mathbf{M}_C and \mathbf{M}_L , respectively, in the sequel. Abbreviations CMM and DLMM, respectively, will be also used. Collectively these take care of many engineering applications in structural dynamics. Occasionally, however, they fall short. The gap can be filled with a more general approach that relies on *templates*. These are algebraic forms that carry free parameters. This approach is covered in this paper using one-dimensional structural elements as expository examples.

The template approach has the virtue of generating a set of mass matrices that satisfy certain *a priori* constraints such as symmetry, nonnegativity, invariance and momentum conservation. In particular, the diagonally-lumped and consistent mass matrices can be obtained as instances. Thus those standard models are not excluded. Availability of free parameters, however, allows the mass matrix to be *customized* to special needs such as high precision in vibration analysis, or minimally dispersive wave propagation. This versatility will be evident from the examples. The set of parameters is called the *template signature*, and uniquely characterizes an element instance.

An attractive feature of templates for FEM programming is that each “custom mass matrix” need not be coded and tested individually. It is sufficient to implement the template as a single element-level module, with free parameters as arguments, and simply adjust the signature to the problem at hand. In particular the same module should be able to produce the conventional DLMM and CMM models, which can provide valuable crosschecks.

§33.2. Customization Scenarios

The ability to customize the mass matrix is not free of cost. The derivation is more complicated, even for 1D elements, than those based on standard procedures. In fact, hand computations rapidly become unfeasible. Help from a computer algebra system (CAS) is needed to complete the task. When is this additional work justified? Two scenarios can be mentioned.

The first is *high fidelity systems*. Dynamic analysis covers a wide range of applications. There is a subclass that calls for a level of simulation precision beyond that customary in engineering analysis. Examples are deployment of precision structures, resonance analysis of machinery or equipment, adaptive active control, ultrasonics imaging, signature detection, radiation loss in layered circuits, and molecular- and crystal-level simulations in micro- and nano-mechanics.

In structural static analysis an error of 20% or 30% in peak stresses is not cause for alarm — such discrepancies are usually covered adequately by safety factors. But a similar error in frequency analysis or impedance response of a high fidelity system may be disastrous. Achieving acceptable precision with a fine mesh, however, can be expensive. Model adaptivity comes to the rescue in statics; but this is less effective in dynamics on account of the time dimension. Customized elements may provide a practical solution: achieving adequate accuracy with a coarse regular mesh.

A second possibility is that the stiffness matrix comes from a method that *avoids displacement shape functions*. For example assumed-stress or assumed strain elements. [Or, it could simply be an array of numbers provided by a black-box program, with no documentation explaining its source.] If this happens the concept of “consistent mass matrix,” in which velocity shape functions are taken to coincide with displacement ones, loses its comfortable variational meaning. One way

out is to take the mass of an element with similar geometry and freedom configuration derived with shape functions, and to pair it with the given stiffness. But in certain cases, notably when the FEM model has rotational freedoms, this may not be easy or desirable.

§33.3. Parametrization Techniques

There are several ways to parametrize mass matrices. Three techniques found effective in practice are summarized below. All of them are illustrated in the worked out examples of Sections 4–6.

Matrix-Weighted Parametrization. A matrix-weighted mass template for element e is a linear combination of $(k + 1)$ component mass matrices, $k \geq 1$ of which are weighted by parameters:

$$\mathbf{M}^e \stackrel{\text{def}}{=} \mathbf{M}_0^e + \mu_1 \mathbf{M}_1^e + \dots \mu_k \mathbf{M}_k^e \quad (33.1)$$

Here \mathbf{M}_0^e is the *baseline mass matrix*. This should be an acceptable mass matrix on its own if $\mu_1 = \dots \mu_k = 0$. The simplest instance of (33.1) is a linear combination of the consistent and diagonally-lumped masses

$$\mathbf{M}^e \stackrel{\text{def}}{=} (1 - \mu) \mathbf{M}_C^e + \mu \mathbf{M}_L^e \quad (33.2)$$

This can be reformatted as (33.1) by writing $\mathbf{M}^e = \mathbf{M}_C^e + \mu(\mathbf{M}_L^e - \mathbf{M}_C^e)$. Here $k = 1$, the baseline is $\mathbf{M}_0^e \equiv \mathbf{M}_C^e$, $\mu \equiv \mu_1$ and \mathbf{M}_1^e is the “consistent mass deviator” $\mathbf{M}_L^e - \mathbf{M}_C^e$. Expression (33.2) is often abbreviated to “LC-weighted mass matrix” or simply LCM.

A matrix-weighted mass template represents a tradeoff. It cuts down on the number of free parameters. Such a reduction is essential for 2D and 3D elements. It makes it easier to satisfy conservation and nonnegativity conditions through appropriate choice of the \mathbf{M}_i^e . On the minus side it generally spans only a subspace of acceptable matrices.

Spectral Parametrization. This has the form

$$\mathbf{M}^e \stackrel{\text{def}}{=} \mathbf{H}^T \mathbf{D}_\mu \mathbf{H}, \quad \mathbf{D}_\mu = \mathbf{diag} [c_0 \mu_0 \ c_1 \mu_1 \ \dots \ c_k \mu_k]. \quad (33.3)$$

in which \mathbf{H} is a generally full matrix. Parameters $\mu_0 \dots \mu_k$ appear as entries of the diagonal matrix \mathbf{D}_μ . Scaling coefficients c_i may be introduced for convenience. Often $\mu_0 = 1$ or $\mu_0 = 0$ are preset from conservation conditions.

Configuration (33.3) occurs naturally when the mass matrix is constructed first in generalized coordinates, followed by transformation to physical coordinates via \mathbf{H} . If the generalized mass is derived using mass-orthogonal functions (for example, Legendre polynomials in 1D elements), the unparametrized generalized mass $\mathbf{D} = \mathbf{diag} [c_0 \ c_1 \ \dots \ c_k]$ is diagonal. Parametrization is effected by scaling entries of this matrix. Some entries may be left fixed, however, to satisfy *a priori* constraints.

Expanding (33.3) and collecting matrices that multiply μ_i leads to a matrix weighted combination form (33.1) in which each \mathbf{M}_i^e is a rank-one matrix. The analogy with the spectral representation theorem of symmetric matrices is obvious. But in practice it is usually better to work directly with the congruent representation (33.3).

Entry-Weighted Parametrization. An entry-weighted mass template applies parameters directly to every entry of the mass matrix, except for *a priori* constraints on symmetry, invariance and

conservation. This form is the most general one and can be expected to lead to best possible solutions. But it is restricted to simple (usually 1D) elements because the number of parameters grows quadratically in the matrix size, whereas for the other two schemes it grows linearly.

Combined Approach. A hierarchical combination of parametrization schemes can be used to advantage if the kinetic energy can be naturally decomposed from physics. For example the Timoshenko beam element covered in Section 6 uses a two-matrix-weighted template form similar to (33.2) as top level. The two components are constructed by spectral and entry-weighted parametrization, respectively.

§33.4. Two-Node Bar Element

The template concept is best grasped through an example that involves the simplest structural finite element: the two-node prismatic bar of density ρ , area A and length ℓ , moving along x . See Figure 33.1. The most general form of the 2×2 mass matrix form is the entry-weighted template

$$\mathbf{M}^e = \begin{bmatrix} M_{11}^e & M_{12}^e \\ M_{21}^e & M_{22}^e \end{bmatrix} = M^e \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix} = \rho A \ell \begin{bmatrix} \mu_{11} & \mu_{12} \\ \mu_{21} & \mu_{22} \end{bmatrix}. \quad (33.4)$$

The first form is merely a list of entries. To parametrize it, the total element mass $M^e = \rho A \ell$ is taken out as factor. The free parameters μ_{11} through μ_{22} are simply numbers. This illustrates a basic convenience rule: *free template parameters should be dimensionless*.

To cut down on the number of free parameters one looks at *mass property constraints*. The most common ones are

Matrix symmetry: $\mathbf{M}^e = (\mathbf{M}^e)^T$. For (33.4) this requires $\mu_{21} = \mu_{12}$.

Physical symmetry: For a prismatic bar \mathbf{M}^e must exhibit antidiagonal symmetry: $\mu_{22} = \mu_{11}$.

Conservation of total translational mass: same as conservation of linear momentum or of kinetic energy. Apply the uniform velocity field $\dot{\mathbf{u}} = \mathbf{v}$ to the bar. The associated nodal velocity vector is $\dot{\mathbf{u}}^e = \mathbf{v}^e = v [1 \ 1]^T$. The kinetic energy is $T^e = \frac{1}{2}(\mathbf{v}^e)^T \mathbf{M}^e \mathbf{v}^e = \frac{1}{2} M^e v^2 (\mu_{11} + \mu_{12} + \mu_{21} + \mu_{22})$. This must equal $\frac{1}{2} M^e v^2$, whence $\mu_{11} + \mu_{12} + \mu_{21} + \mu_{22} = 1$.

Nonnegativity: \mathbf{M}^e should not be indefinite. [This is not an absolute must, and it is actually relaxed in the Timoshenko beam element discussed in 33.6.] Whether checked by computing eigenvalues or principal minors, this constraint is nonlinear and of inequality type. Consequently it is not often applied *ab initio*, unless the element is quite simple, as in this case.

On applying the symmetry and conservation rules three parameters of (33.4) are eliminated. The remaining one, called μ , is taken for convenience to be $\mu_{11} = \mu_{22} = \frac{1}{6}(2 + \mu)$ and $\mu_{12} = \mu_{21} = \frac{1}{6}(1 - \mu)$, which gives

$$\begin{aligned} \mathbf{M}_\mu^e &= \frac{1}{6} \rho A \ell \begin{bmatrix} 2 + \mu & 1 - \mu \\ 1 - \mu & 2 + \mu \end{bmatrix} = (1 - \mu) \frac{1}{6} \rho A \ell \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} + \mu \frac{1}{6} \rho A \ell \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\ &= (1 - \mu) \mathbf{M}_C^e + \mu \mathbf{M}_L^e. \end{aligned} \quad (33.5)$$

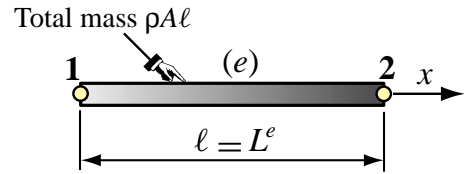


FIGURE 33.1. The two-node prismatic bar element.

Expression (33.5) shows that the one-parameter template can be presented as a linear combination of the well known consistent and diagonally-lumped mass matrices. So starting with the general entry-weighted form (33.4) we end up with a two-matrix-weighted form befitting (33.2). If $\mu = 0$ and $\mu = 1$, (33.5) reduces to \mathbf{M}_C^e and \mathbf{M}_L^e respectively. This illustrates another desirable property: the CMM and DLMM models ought to be instances of the template.

Finally we can apply the nonnegativity constraint. For the two principal minors of \mathbf{M}_μ^e to be nonnegative, $2 + \mu \geq 0$ and $(2 + \mu)^2 - (1 - \mu)^2 = 3 + 6\mu \geq 0$. Both are satisfied if $\mu \geq -1/2$. Unlike the others, this constraint is of inequality type, and only limits the range of μ .

The remaining task is to find μ . This is done by introducing an *optimality criterion that fits the problem at hand*. This is where customization comes in. Even for this simple case the answer is not unique. Thus the sentence “the best mass matrix for the two-node bar is so-and-so” has no unique meaning. Two specific optimization criteria are studied below.

§33.4.1. Best μ by Angular Momentum Preservation

Allow the bar to move in the $\{x, y\}$ plane by expanding its nodal DOF to $\mathbf{u}^e = [u_{x1} \ u_{y1} \ u_{x2} \ u_{y2}]^T$ so (33.5) becomes a 4×4 matrix

$$\mathbf{M}_\mu^e = \frac{1}{6}\rho A \ell \begin{bmatrix} 2 + \mu & 0 & 1 - \mu & 0 \\ 0 & 2 + \mu & 0 & 1 - \mu \\ 1 - \mu & 0 & 2 + \mu & 0 \\ 0 & 1 - \mu & 0 & 2 + \mu \end{bmatrix} \quad (33.6)$$

Apply a uniform angular velocity $\dot{\theta}$ about the midpoint. The associated node velocity vector at $\theta = 0$ is $\dot{\mathbf{u}}^e = \frac{1}{2}\ell\dot{\theta} [0 \ -1 \ 0 \ 1]^T$. The discrete and continuum energies are

$$T_\mu^e = \frac{1}{2}(\dot{\mathbf{u}}^e)^T \mathbf{M}_\mu^e \dot{\mathbf{u}}^e = \frac{1}{24}\rho A \ell^3 (1 + 2\mu), \quad T^e = \int_{-\ell/2}^{\ell/2} \rho A (\dot{\theta} x)^2 dx = \frac{1}{24}\rho A \ell^3. \quad (33.7)$$

Matching $T_\mu^e = T^e$ gives $\mu = 0$. So according to this criterion the optimal mass matrix is the consistent one (CMM). Note that if $\mu = 1$, $T_\mu^e = 3T^e$, whence the DLMM overestimates the rotational (rotary) inertia by a factor of 3.

§33.4.2. Best μ by Fourier Analysis

Another useful optimization criterion is the fidelity with which planes waves are propagated over a bar element lattice, when compared to the case of a continuum bar pictured in Figure 33.2.

Symbols used for propagation of harmonic waves are collected in Table 33.1 for the reader's convenience. (Several of these are reused in Sections 5 and 6.) The discrete counterpart of Figure 33.2 is shown in Figure 33.3.

This is a lattice of repeating two-node bar elements of length ℓ . Lattice wave propagation nomenclature is similar to that defined for the continuum case in Table 33.1, but without zero subscripts.

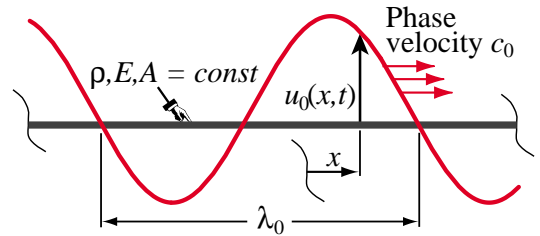


FIGURE 33.2. Propagation of a harmonic wave over an infinite, continuum prismatic bar. The wave-profile axial displacement $u(x, t)$ is plotted normal to the bar.

Table 33.1 Nomenclature for Harmonic Wave Propagation in a Continuum Bar

Quantity	Meaning (physical dimension in brackets)
ρ, E, A	Mass density, elastic modulus, and cross section area of bar
$\rho \ddot{u}_0 = E u_0''$	Bar wave equation. Alternate forms: $-\omega_0^2 u = c_0^2 u''$ and $u'' + k_0^2 u = 0$.
$u_0(x, t)$	Waveform $u_0 = B_0 \exp(i(k_0 x - \omega_0 t))$ [length], in which $i = \sqrt{-1}$
B_0	Wave amplitude [length]
λ_0	Wavelength [length]
k_0	Wavenumber $k_0 = 2\pi/\lambda_0$ [1/length]
ω_0	Circular (a.k.a. angular) frequency $\omega_0 = k_0 c_0 = 2\pi f_0 = 2\pi c_0/\lambda_0$ [radians/time]
f_0	Cyclic frequency $f_0 = \omega_0/(2\pi)$ [cycles/time: Hz if time in seconds]
T_0	Period $T_0 = 1/f_0 = 2\pi/\omega_0 = \lambda_0/c_0$ [time]
c_0	Phase velocity $c_0 = \omega_0/k_0 = \lambda_0/T_0 = \lambda_0 f_0 = \sqrt{E/\rho}$ [length/time]
κ_0	Dimensionless wavenumber $\kappa_0 = k_0 \lambda_0$ ($\kappa_0 = 2\pi$ in continuum)
Ω_0	Dimensionless circular frequency $\Omega_0 = \omega_0 T_0 = \omega_0 \lambda_0 / c_0$
Zero subscripted quantities, such as k_0 or c_0 , refer to the continuum bar. Unsubscripted counterparts, such as k or c , pertain to a discrete FEM lattice as in Figure 33.3.	

The lattice propagation process is governed by the semidiscrete equation of motion $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{0}$, which can be solved by Fourier methods. To study solutions it is sufficient to extract a two-element patch as illustrated in Figure 33.3(a). Within some constraints noted later the lattice can propagate travelling harmonic waves of wavelength λ and phase velocity c , as depicted in Figure 33.3(b). The wavenumber is $k = 2\pi/\lambda$ and the circular frequency $\omega = 2\pi/T = 2\pi c/\lambda = kc$. Figure 33.3(b) displays two characteristic lengths: λ and ℓ . The element-to-wavelength ratio is called $\chi = \ell/\lambda$. This ratio characterizes the fineness of the discretization with respect to wavelength. A harmonic wave of amplitude B is described by the function

$$u(x, t) = B \exp[i(kx - \omega t)] = B \exp[i(\kappa x - \Omega c_0 t)/\ell], \quad i = \sqrt{-1}. \quad (33.8)$$

Here the dimensionless wavenumber κ and circular frequency Ω are defined as $\kappa = k\ell = 2\pi\ell/\lambda = 2\pi\chi$ and $\Omega = \omega\ell/c_0$, respectively, in which $c_0 = \sqrt{E/\rho}$ is the continuum bar wavespeed. Using the well-known bar stiffness matrix and the mass template (33.5) gives the patch equations

$$\frac{\rho A \ell}{6} \begin{bmatrix} 2 + \mu & 1 - \mu & 0 \\ 1 - \mu & 4 + 2\mu & 1 - \mu \\ 0 & 1 - \mu & 2 + \mu \end{bmatrix} \begin{bmatrix} \ddot{u}_{j-1} \\ \ddot{u}_j \\ \ddot{u}_{j+1} \end{bmatrix} + \frac{EA}{\ell} \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} u_{j-1} \\ u_j \\ u_{j+1} \end{bmatrix} = 0. \quad (33.9)$$

From this one takes the middle (node j) equation, which repeats in the infinite lattice:

$$\frac{\rho A \ell}{6} \begin{bmatrix} 1 - \mu & 4 + 2\mu & 1 - \mu \end{bmatrix} \begin{bmatrix} \ddot{u}_{j-1} \\ \ddot{u}_j \\ \ddot{u}_{j+1} \end{bmatrix} + \frac{EA}{\ell} \begin{bmatrix} -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} u_{j-1} \\ u_j \\ u_{j+1} \end{bmatrix} = 0. \quad (33.10)$$

Evaluate (33.8) at $x = x_{j-1} = x_j - \ell$, $x = x_j$ and $x = x_{j+1} = x_j + \ell$ while keeping t continuous. Substitution into (33.10) gives the wave propagation condition

$$\frac{\rho A c_0^2}{3\ell} \left[6 - (2 + \mu)\Omega^2 - (6 - (1 - \mu)\Omega^2) \cos \kappa \right] \left(\cos \frac{\Omega c_0 t}{\ell} - i \sin \frac{\Omega c_0 t}{\ell} \right) B = 0. \quad (33.11)$$

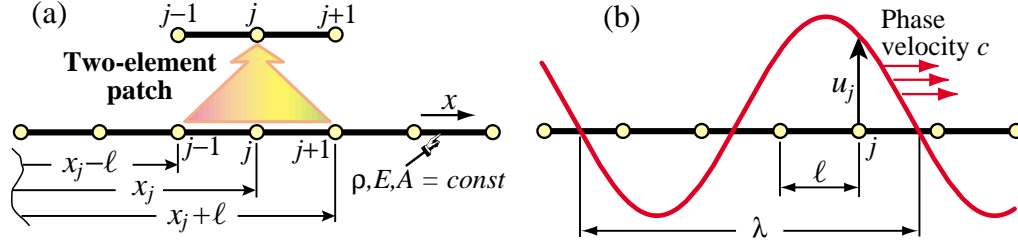


FIGURE 33.3. An infinite lattice of two-node prismatic bar elements: (a) 2-element patch extracted from lattice; (b) characteristic dimensions for a propagating harmonic wave.

If this is to be zero for any t and B , the expression in brackets must vanish. Solving gives the frequency-wavenumber relations

$$\begin{aligned}\Omega^2 &= \frac{6(1 - \cos \kappa)}{2 + \mu + (1 - \mu) \cos \kappa} = \kappa^2 + \frac{1 - 2\mu}{12} \kappa^4 + \frac{1 - 10\mu + 10\mu^2}{360} \kappa^6 + \dots, \\ \kappa &= \arccos \left[\frac{6 - (2 + \mu)\Omega^2}{6 + (1 - \mu)\Omega^2} \right] = \Omega - \frac{1 - 2\mu}{24} \Omega^3 + \frac{9 - 20\mu + 20\mu^2}{1920} \Omega^5 + \dots\end{aligned}\quad (33.12)$$

Returning to physical wavenumber $k = \kappa/\ell$ and frequency $\omega = \Omega c_0/\ell$:

$$\omega^2 = \left(\frac{6c_0^2}{\ell^2} \right) \frac{1 - \cos(k\ell)}{2 + \mu + (1 - \mu) \cos(k\ell)} = c_0^2 k^2 \left(1 + \frac{1 - 2\mu}{12} k^2 \ell^2 + \frac{1 - 10\mu + 10\mu^2}{360} k^4 \ell^4 + \dots \right) \quad (33.13)$$

An equation that links frequency and wavenumber: $\omega = \omega(k)$ as in (33.13), is a *dispersion relation*. An oscillatory dynamical system is *nondispersive* if ω is linear in k , in which case $c = \omega/k$ is constant and the wavespeed is the same for all frequencies. The dispersion relation for the continuum bar (within the limits of MoM assumptions) is $c_0 = \omega_0/k_0$: all waves propagate with the same speed c_0 . On the other hand the FEM model is *dispersive* for any μ , since from (33.12) we get

$$\frac{c}{c_0} = \frac{\omega}{k c_0} = \frac{1}{\kappa} \sqrt{\frac{6(1 - \cos \kappa)}{2 + \mu + (1 - \mu) \cos \kappa}} = 1 + \frac{1 - 2\mu}{24} \kappa^2 + \frac{1 - 20\mu + 20\mu^2}{1920} \kappa^4 + \dots \quad (33.14)$$

The best fit to the continuum for *small wavenumbers* $\kappa = k\ell \ll 1$ is obtained by taking $\mu = 1/2$. This makes the second term of the foregoing series vanish. So from this standpoint the best mass matrix for the bar is

$$\mathbf{M}_\mu^e|_{\mu=1/2} = \frac{1}{2} \mathbf{M}_C^e + \frac{1}{2} \mathbf{M}_L^e = \frac{\rho A \ell}{12} \begin{bmatrix} 5 & 1 \\ 1 & 5 \end{bmatrix}. \quad (33.15)$$

Figure 33.4(a) plots the dimensionless dispersion relation (33.12) for the consistent ($\mu = 0$), diagonally lumped ($\mu = 1$) and LC-averaged ($\mu = 1/2$) mass matrices, along with the continuum-bar relation $\Omega_0 = \kappa_0$. The lattice curves of Figure 33.4(a) have a 2π period: $\Omega(\kappa) = \Omega(\kappa + 2\pi n)$, n being an integer. Thus it is enough to plot $\Omega(\kappa)$ over $\kappa \in [0, 2\pi]$. The maximum lattice frequency, which occurs for $\kappa = k\ell = \pi$ or $\lambda = 2\ell$, is called the *Nyquist* or *folding frequency*.

If it is possible to pick μ as function of Ω or κ we can match the continuum over a certain range of

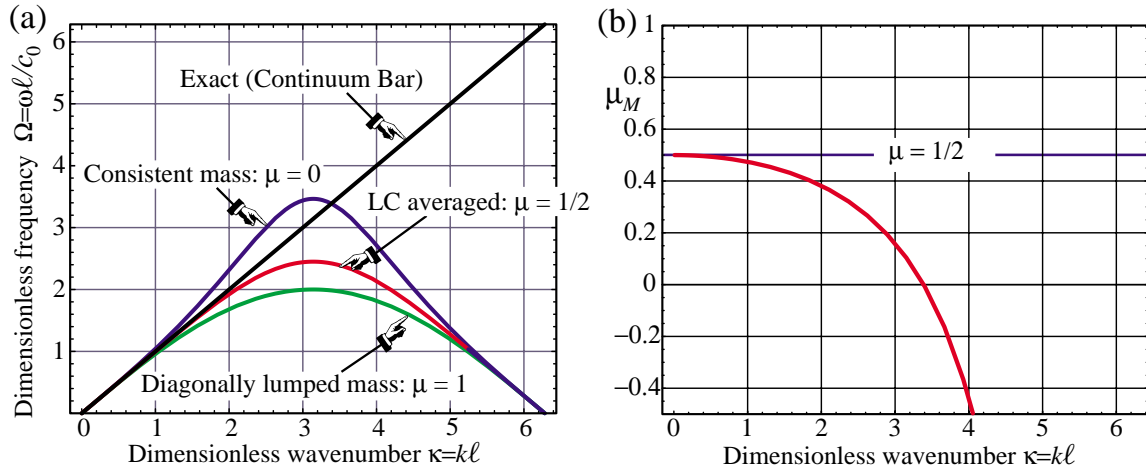


FIGURE 33.4. Results from Fourier analysis of two-node bar lattice: (a) dispersion curves for various choices of μ ; (b) wavenumber dependent μ_M that makes lattice match the continuum.

κ or Ω . This can be done by equating $\Omega = \kappa$ (or $c = c_0$) and solving for μ :

$$\begin{aligned} \mu_M &= 1 + \frac{6}{\kappa^2} - \frac{3}{1 - \cos \kappa} = \frac{1}{2} - \frac{\kappa^2}{40} - \frac{\kappa^4}{1008} - \frac{\kappa^6}{28800} - \dots = \frac{1}{2} - \frac{4\pi^2 \chi^2}{40} - \frac{16\pi^4 \chi^4}{1008} - \dots \\ &= 1 + \frac{6}{\Omega^2} - \frac{3}{1 - \cos \Omega} = \frac{1}{2} - \frac{\Omega^2}{40} - \frac{\Omega^4}{1008} - \frac{\Omega^6}{28800} - \dots \end{aligned} \quad (33.16)$$

in which $\chi = 2\pi \kappa$. The function $\mu_M(\kappa)$ is plotted in Figure 33.4(b). Interesting values are $\mu_M = 0$ if $\kappa = 3.38742306673364$ and $\mu_M = -1/2$ if $\kappa = \kappa_{lim} = 4.05751567622863$. If $\kappa > \kappa_{lim}$ the fitted \mathbf{M}^e becomes indefinite. So (33.16) is practically limited to the range $0 \leq k \leq \approx 4/\ell$ shown in the plot.

§33.4.3. *Best μ By Modified Equation

The gist of Fourier analysis is to find an exact solution, which separates space and time in the characteristic equation (33.11). The rest is routine mathematics. The method of modified differential equations or MoDE, introduced in §13.8, makes less initial assumptions but is not by any means routine. The objective is to find a MoDE that, if solved exactly, produces the FEM solution at nodes, and to compare it with the continuum wave equation given in Table 33.1. The method assumes only¹ that $\ddot{\mathbf{u}} = -\omega^2 \mathbf{u}$, where ω is a lattice frequency left to be determined. This takes care of the time variation. Applying this assumption to the patch equation (33.10) and passing to dimensionless variables we get

$$\begin{bmatrix} -1 - \frac{1}{6}(1 - \mu)\Omega^2 & 2 - \frac{1}{6}(4 + 2\mu)\Omega^2 & -1 - \frac{1}{6}(1 - \mu)\Omega^2 \end{bmatrix} \begin{bmatrix} u_{j-1} \\ u_j \\ u_{j+1} \end{bmatrix} = 0. \quad (33.17)$$

¹ This assumption is necessary because no discretization in the time domain has been specified. If a time integrator had been applied, we would face a partial differential equation in space and time. That is a tougher nut to crack in an introductory course.

in which $\Omega = \omega\ell/c_0$ is the dimensionless circular frequency used in the previous subsection. This difference equation is “continuified” by replacing $u_j \rightarrow u(t)$ and $u_{j\pm 1} \rightarrow u(t \pm \ell)$, and scaled through to produce the following difference-differential form or DDMoDE:

$$\begin{bmatrix} -\frac{1}{2} & \psi & -\frac{1}{2} \end{bmatrix} \begin{bmatrix} u(t-\ell) \\ u(t) \\ u(t+\ell) \end{bmatrix} = \psi u(t) - \frac{1}{2}(u(t-\ell) + u(t+\ell)) = 0, \quad \text{with } \psi = \frac{6 - (2 + \mu)\Omega^2}{6 + (1 - \mu)\Omega^2}. \quad (33.18)$$

As MoDE expansion coefficient we select the element-to-wavelength ratio $\chi = \ell/\lambda$. Accordingly, expanding the the end values in Taylor series about $u(t)$ gives $u(t - \ell) = u_j - \ell u'(t) + \ell^2 u''(t)/2! - \dots$, and $u(t + \ell) = u_j + \ell u'(t) + \ell^2 u''(t)/2! + \dots$. The foregoing semisum is $\frac{1}{2}(u(t - \ell) + u(t + \ell)) = u(t) + \ell^2 u''(t)/2! + \ell^4 u''''(t)/4! + \dots$, which contains only even u derivatives. Replacing into (33.18) and setting $\ell = \lambda\chi$, we obtain the infinite-order modified differential equation or IOMoDE:

$$(1 - \psi)u + \frac{1}{2!}\lambda^2\chi^2 u'' + \frac{\lambda^4\chi^4}{4!}u'''' + \dots = 0. \quad (33.19)$$

This form needs further work. To see why, consider a mesh refinement process that makes $\chi \rightarrow 0$ while keeping λ fixed. Then (33.19) approaches $(1 - \psi)u = 0$. Since $u \neq 0$, $\psi \rightarrow 1$ in the sense that $1 - \psi = O(\chi^2)$. To obtain the canonical MoDE form exhibited below, the transformation $1 - \psi \rightarrow \frac{1}{2}\gamma\chi^2$, where γ is another free coefficient, is specified, and solved for Ω^2 , giving $\Omega^2 = 6\gamma\chi^2/(6 - \gamma(1 - \mu)\chi^2)$. Replacing into (33.19) and dividing through by $\ell^2 = \lambda^2\chi^2$ yields

$$\frac{\gamma}{2\lambda^2}u + \frac{1}{2!}u'' + \frac{\lambda^2\chi^2}{4!}u'''' + \frac{\lambda^4\chi^4}{6!}u'''''' + \dots = 0. \quad (33.20)$$

This has the correct IOMoDE form: as $\chi \rightarrow 0$ two terms survive and a second-order ODE emerges. In fact, but for the sign of the first term this is a canonical IOMoDE form studied in [94] for a different application. The last and quite complicated MoDE step is elimination of derivatives of order higher than two. The technique of that reference leads to the finite order modified equation or FOMoDE:

$$u'' + \frac{4}{\lambda^2\chi^2} \left(\arcsin \frac{\chi\sqrt{\gamma}}{2} \right)^2 u = 0. \quad (33.21)$$

This is matched to the continuum bar equation $u'' + k_0^2 u = 0$ of Table 33.1, assuming that $\lambda = \lambda_0$. Equating coefficients gives $k_0\lambda_0\chi/2 = \pi\chi = \arcsin(\frac{1}{2}\chi\sqrt{\gamma})$ or $\sin(\pi\chi) = \frac{1}{2}\chi\sqrt{\gamma}$, whence $\gamma = 4\sin^2(\pi\chi)/\chi^2$ and

$$\Omega^2 = \frac{6\gamma\chi^2}{6 - \gamma(1 - \mu)\chi^2} = \frac{12\sin^2(\pi\chi)}{2 + \mu + (1 - \mu)\cos(2\pi\chi)} = 4\pi^2\chi^2. \quad (33.22)$$

The last equation comes from the definition $\Omega = \omega\ell/c_0 = 2\pi\chi$. Solving for μ gives the discrete-to-continuum matching value

$$\mu_M = 1 + \frac{3}{2\pi^2\chi^2} - \frac{3}{2\cos^2(\pi\chi)} = \frac{1}{2} - \frac{\pi^2\chi^2}{10} - \frac{\pi^4\chi^4}{63} - \frac{\pi^6\chi^6}{450} - \dots \quad (33.23)$$

Replacing $2\pi\chi$ by κ (or by $\Omega = \kappa$) gives $\mu_M = 1 + 6/\kappa^2 - 3/(1 - \cos\kappa)$, thus reproducing the result (33.16). So Fourier analysis and MoDE deliver the same result.

Remark 33.1. When Fourier can be used, as here, it is far simpler than MoDE. However the latter provides a side bonus: *a priori* error estimates. For example, suppose that one plans to put 10 elements within the shortest wavelength of interest. Thus $\chi = \ell/\lambda = 1/10$ and $\gamma = 4\sin^2(\pi\chi)/\chi^2 = 38.1966$. Replacing into (33.22) gives $\Omega/(2\pi\chi) = 1.016520, 0.999670$ and 0.983632 for $\mu = 0, \mu = 1/2$ and $\mu = 1$, respectively. The estimate frequency errors with respect to the continuum are 1.65%, -0.04% and -1.64% , respectively.

§33.5. Three-Node Bar Element

As pictured in Figure 33.5, this element is prismatic with length ℓ , cross section area A and mass density ρ . Midnode 3 is at the center. The element DOFs are arranged as $\mathbf{u}^e = [u_1 \ u_2 \ u_3]^T$. Its well known stiffness matrix is paired with a entry-weighted mass template:

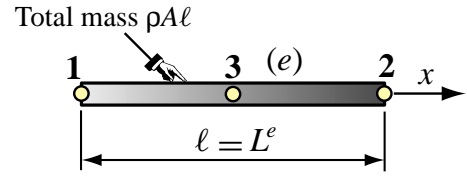


FIGURE 33.5. The three-node prismatic bar element.

$$\mathbf{K}^e = \frac{EA}{3\ell} \begin{bmatrix} 7 & 1 & -8 \\ 1 & 7 & -8 \\ -8 & -8 & 16 \end{bmatrix}, \quad \mathbf{M}_\mu^e = \frac{\rho A \ell}{90} \begin{bmatrix} 12 + \mu_1 & -3 + \mu_3 & 6 + \mu_4 \\ -3 + \mu_3 & 12 + \mu_1 & 6 + \mu_4 \\ 6 + \mu_4 & 6 + \mu_4 & 48 + \mu_2 \end{bmatrix} \quad (33.24)$$

The idea behind the assumed form of \mathbf{M}_μ^e in (33.24) is to define the mass template as a parametrized deviation from the consistent mass matrix. That is, setting $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 0$ makes $\mathbf{M}_\mu^e = \mathbf{M}_C^e$. Setting $\mu_1 = \mu_3 = 3$, $\mu_2 = 12$ and $\mu_4 = -6$ gives the well known diagonally lumped mass matrix (DLMM) generated by Simpson's integration rule: $\mathbf{M}_L^e = \rho A \ell \mathbf{diag} [1/6, 1/6, 2/3]$. Thus again the standard models are template instances. Notice that \mathbf{M}_μ^e in (33.24) incorporates matrix and physical symmetries *a priori* but not conservation conditions.

Linear and angular momentum conservation requires $2\mu_1 + \mu_2 + 2\mu_3 + 4\mu_4 = 0$ and $\mu_3 = \mu_1$, respectively. Eliminating μ_3 and μ_4 from those constraints reduces the template to two parameters:

$$\mathbf{M}_\mu^e = \frac{\rho A \ell}{360} \begin{bmatrix} 4(12 + \mu_1) & 4(-3 + \mu_1) & 24 - 4\mu_1 - \mu_2 \\ 4(-3 + \mu_1) & 4(12 + \mu_1) & 24 - 4\mu_1 - \mu_2 \\ 24 - 4\mu_1 - \mu_2 & 24 - 4\mu_1 - \mu_2 & 4(48 + \mu_2) \end{bmatrix} \quad (33.25)$$

For (33.25) to be nonnegative, $\mu_1 \geq -9/2$ and $15 + \mu_1 - 3\sqrt{5}\sqrt{9 + 2\mu_1} \leq \frac{1}{4}\mu_2 \leq 15 + \mu_1 + 3\sqrt{5}\sqrt{9 + 2\mu_1}$. These inequality constraints should be checked *a posteriori*.

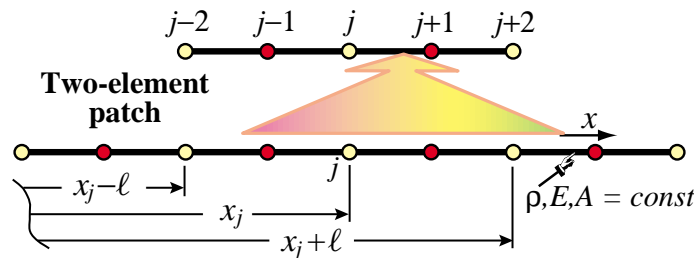


FIGURE 33.6. Lattice of three-node bar elements from which a 2-element patch is extracted. Yellow and red-filled circles flag endnodes and midnodes, respectively.

§33.5.1. Patch Equations

Unlike the two-node bar, two free parameters remain after the angular momentum conservation condition is enforced. Consequently we can ask for satisfactory wave propagation conditions in addition to conservation. To assess performance of mass-stiffness combinations we carry out the plane wave analysis of the infinite beam lattice shown in Figure 33.6.

From the lattice we extract a typical two node patch as illustrated. The patch has five nodes: three endpoints and two midpoints, which are assigned global numbers $j-2, j-1, \dots, j+2$. The unforced semidiscrete dynamical equations of the patch are $\mathbf{M}^P \ddot{\mathbf{u}}^P + \mathbf{K}^P \mathbf{u}^P = \mathbf{0}$, where

$$\mathbf{M}^P = \frac{\rho A \ell}{360} \begin{bmatrix} 4(12 + \mu_1) & 24 - 4\mu_1 - \mu_2 & 4(-3 + \mu_1) & 0 & 0 \\ 24 - 4\mu_1 - \mu_2 & 4(48 + \mu_2) & 24 - 4\mu_1 - \mu_2 & 0 & 0 \\ 4(-3 + \mu_1) & 24 - 4\mu_1 - \mu_2 & 8(12 + \mu_1) & 24 - 4\mu_1 - \mu_2 & 4(-3 + \mu_1) \\ 0 & 0 & 24 - 4\mu_1 - \mu_2 & 4(48 + \mu_2) & 24 - 4\mu_1 - \mu_2 \\ 0 & 0 & 4(-3 + \mu_1) & 24 - 4\mu_1 - \mu_2 & 4(12 + \mu_1) \end{bmatrix}$$

$$\mathbf{K}^P = \frac{EA}{3\ell} \begin{bmatrix} 7 & -8 & 1 & 0 & 0 \\ -8 & 16 & -8 & 0 & 0 \\ 1 & -8 & 14 & -8 & 1 \\ 0 & 0 & -8 & 16 & -8 \\ 0 & 0 & 1 & -8 & 7 \end{bmatrix}, \quad \mathbf{u}^P = [u_{j-2} \quad u_{j-1} \quad u_j \quad u_{j+1} \quad u_{j+2}]^T.$$
(33.26)

From the foregoing we keep the third and fourth equations, namely those for nodes j and $j+1$. This selection provides the equations for a typical corner point j and a typical midpoint $j+1$. The retained patch equations are

$$\mathbf{M}_{j,j+1}^P \ddot{\mathbf{u}}_P + \mathbf{K}_{j,j+1}^P \mathbf{u}_P = \mathbf{0}.$$
(33.27)

The 2×5 matrices $\mathbf{M}_{j,j+1}^P$ and $\mathbf{K}_{j,j+1}^P$ result on deleting rows 1,2,5 of \mathbf{M}^P and \mathbf{K}^P , respectively.

§33.5.2. Fourier Analysis

We study the propagation of harmonic plane waves of wavelength λ , wavenumber $k = 2\pi/\lambda$, and circular frequency ω over the lattice of Figure 33.6. For convenience they are separated into corner and midpoint waves:

$$u_c(x, t) = B_c e^{i(kx - \omega t)}, \quad u_m(x, t) = B_m e^{i(kx - \omega t)}.$$
(33.28)

Wave $u_c(x, t)$ propagates only over corners and vanishes at midpoints, whereas $u_m(x, t)$ propagates only over midpoints and vanishes at corners. Both have the same wavenumber and frequency but different amplitudes and phases. [Waves (33.28) can be combined to form a single waveform that propagates over all nodes. The combination has two components that propagate with the same speed but in opposite directions. This is useful when studying boundary conditions or transitions in finite lattices, but is not needed for a periodic infinite lattice.] As in the two-node bar case, we will work with the dimensionless frequency $\Omega = \omega \ell / c_0$ and dimensionless wavenumber $\kappa = k \ell$.

Inserting (33.28) into (33.26), passing to dimensionless variables and requiring that solutions exist for any t yields the characteristic equation

$$\frac{1}{180} \begin{bmatrix} 960 - 2(48 + \mu_2)\Omega^2 & -(960 + (24 - 4\mu_1 - \mu_2)\Omega^2) \cos \frac{1}{2}\kappa \\ \text{symm} & 4(210 - (12 + \mu_1)\Omega^2 + (30 + (3 - \mu_1)\Omega^2) \cos \kappa) \end{bmatrix} \begin{bmatrix} B_c \\ B_m \end{bmatrix} = \mathbf{0}. \quad (33.29)$$

For nontrivial solutions the determinant of the characteristic matrix must vanish. Solving for Ω^2 gives two frequencies for each wavenumber κ . They can be expressed as the dispersion relations

$$\Omega_a^2 = \frac{\phi_1 + \delta}{\phi_5 + \phi_6 \cos \kappa}, \quad \Omega_o^2 = \frac{\phi_1 - \delta}{\phi_5 + \phi_6 \cos \kappa}, \quad (33.30)$$

in which $\phi_1 = 720(-208 - \mu_2 + (\mu_2 - 32) \cos \kappa)$, $\phi_2 = 64(\mu_1 - 60)\mu_1 - 32\mu_1\mu_2 + 13\mu_2^2 + 384(474 + \mu_2)$, $\phi_3 = 12(-112 + \mu_2)(128 + \mu_2)$, $\phi_4 = 64(132 + (\mu_1 - 60)\mu_1) - 32(\mu_1 - 6)\mu_2 + \mu_2^2$, $\phi_5 = 16(-540 + (\mu_1 - 60)\mu_1) - 8(30 + \mu_1)\mu_2 + \mu_2^2$, $\phi_6 = 2880 + 16(\mu_1 - 60)\mu_1 - 8\mu_1\mu_2 + \mu_2^2$ and $\delta = 120\sqrt{6}\sqrt{\phi_2 - \phi_3 \cos \kappa - \phi_4 \cos 2\kappa}$. Frequencies Ω_a and Ω_o pertain to the so-called *acoustic* and *optical* branches, respectively. This nomenclature originated in crystal physics, in which both branches have physical meaning as modeling molecular oscillations. [In molecular crystallography, acoustic waves are long-wavelength, low-frequency mechanical waves caused by sonic-like disturbances, in which adjacent molecules move in the same direction. Optical waves are short-wavelength, high-frequency oscillations caused by interaction with light or electromagnetics, in which adjacent molecules move in opposite directions. **Notes and Bibliography** provides references.]

Figure 33.7 illustrates nomenclature used for a two-branched dispersion diagram such as that given by (33.30). The meaning of terms such as “stopping band” is defined below. In FEM discretization work only the acoustic branch has physical meaning because for small κ (that is, long wavelengths) it approaches the continuum bar relation $\Omega = \kappa$, as shown below in (33.31). On the other hand, the optical branch is physically spurious. It is caused by the discretization and pertains to high-frequency lattice oscillations, also known as “mesh modes.”

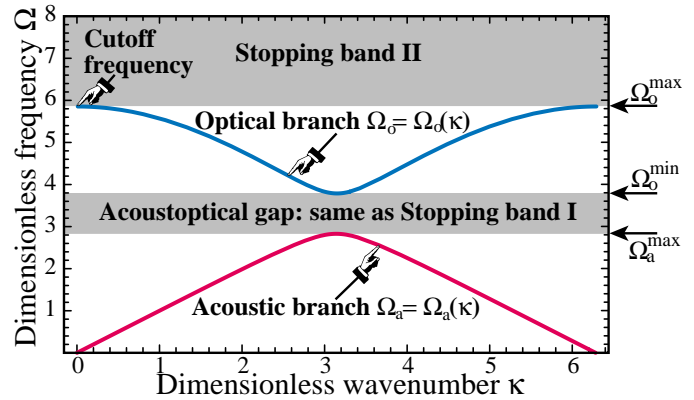


FIGURE 33.7. Notation pertaining to a typical two-branch dispersion diagram. The stopping band is the union of I and II.

The distinction between the two branches can be better grasped by examining the Taylor expansions of the frequencies (33.30) about $\kappa = 0$:

$$\Omega_a^2 = \kappa^2 + \frac{C_4\kappa^4}{4!} + \frac{C_6\kappa^6}{6!} + \frac{C_8\kappa^8}{8!} + \dots, \quad \Omega_o^2 = D_0 + \frac{D_2\kappa^2}{2!} + \dots, \quad (33.31)$$

in which

$$\begin{aligned}
5 \times 1440 C_4 &= 14400 - \psi_1^2 = (240 - 4\mu_1 + \mu_2)(4\mu_1 - \mu_2), \\
10 \times 1440^2 C_6 &= 41472000 + 7200\psi_1^2 + 180\psi_1^3 + \psi_1^4 - 720\psi_1^2\psi_2, \\
60 \times 1440^3 C_8 &= -2030469120000 + 348364800\psi_1^2 - 14515200\psi_1^3 - 342720\psi_1^4 - 2520\psi_1^5 \\
&\quad - 7\psi_1^6 + 58060800\psi_1^2\psi_2 + 1451520\psi_1^3\psi_2 + 10080\psi_1^4\psi_2 - 2903040\psi_1^2\psi_2^2, \\
D_0 &= \frac{1}{\psi_3}, \quad D_2 = -\frac{\psi_1^2(43200 + 360\psi_1 + \psi_1^2 - 1440\psi_2)}{\psi_3^2}.
\end{aligned} \tag{33.32}$$

Here $\psi_1 = 4\mu_1 - \mu_2 - 120$, $\psi_2 = \mu_1 - 2$ and $\psi_3 = 28800 + 360\psi_1 + \psi_1^2 - 1440\psi_2 = -2880 - 960\mu_1 + 16\mu_1^2 - 120\mu_2 - 8\mu_1\mu_2 + \mu_2^2$. Note that the expansion of Ω_a^2 approaches κ^2 as $\kappa \rightarrow 0$. Clearly the acoustic branch is the long-wavelength counterpart of the continuum bar, for which $\Omega = \kappa$. On the other hand, the optical branch has a nonzero frequency $\Omega_o^2 = 1/\psi_3$ at $\kappa = 0$, called the *cutoff frequency*, which cannot vanish although it may go to infinity if $\psi_3 = 0$. As illustrated in Figure 33.7, the lowest and highest values of Ω_o (taking the + square root of Ω_o^2) are called Ω_o^{max} and Ω_o^{min} , respectively, while the largest Ω_a is called Ω_a^{max} . Usually, but not always, Ω_o^{min} and Ω_a^{max} occur at $\kappa = \pi$.

If $\Omega_o^{min} > \Omega_a^{max}$, the range $\Omega_o^{min} > \Omega > \Omega_a^{max}$ is called the *acoustoptical frequency gap* or simply the AO gap. Frequencies in this gap are said to pertain to portion I of the *stopping band*, a term derived from filter technology. Frequencies $\Omega > \Omega_o^{max}$ pertain to portion II of the stopping band. A stopping band frequency cannot be propagated as harmonic plane wave over the lattice. This can be proven by showing that if Ω pertains to the stopping band, the characteristic equation (33.29) has complex roots with nonzero real parts. This causes exponential attenuation so any disturbance with that frequency will decay exponentially.

Table 33.2. Useful Mass Matrices for Three-Node Bar Element

Mass matrix	Template signature $\mu_1 \quad \mu_2 \quad \mu_3 \quad \mu_4$	Taylor expansion of Ω_a^2 (acoustic branch)	Taylor expansion of Ω_o^2 (optical branch)
CMM	0 0 0 0	$\kappa^2 + \frac{\kappa^6}{720} - \frac{11\kappa^8}{151200} + \frac{7\kappa^{10}}{129600} + O(\kappa^{12})$	$60 - 20\kappa^2 + O(\kappa^4)$
DLMM	3 12 3 -6	$\kappa^2 - \frac{\kappa^6}{1440} - \frac{\kappa^8}{48383} - \frac{\kappa^{10}}{4147200} + O(\kappa^{12})$	$24 - 2\kappa^2 + O(\kappa^4)$
BLC	2 8 2 -4	$\kappa^2 - \frac{\kappa^8}{37800} - \frac{\kappa^{10}}{864000} + O(\kappa^{12})$	$30 - \frac{15\kappa^2}{4} + O(\kappa^4)$
COF	8 32 8 -16	$\kappa^2 - \frac{\kappa^6}{240} - \frac{\kappa^8}{6048} + \frac{\kappa^{10}}{86400} + O(\kappa^{12})$	12

§33.5.3. Customization

Figure 33.8 shows dispersion curves for the four parameter settings tabulated in Table 33.2. The four associated mass matrices are positive definite. Dispersion curves for the consistent mass \mathbf{M}_C^e and the diagonally lumped mass \mathbf{M}_L^e are shown in (a,b). Both matrices have an acoustical branch that agrees with the continuum to order $O(\kappa^4)$, as shown by the series listed in Table 33.2.

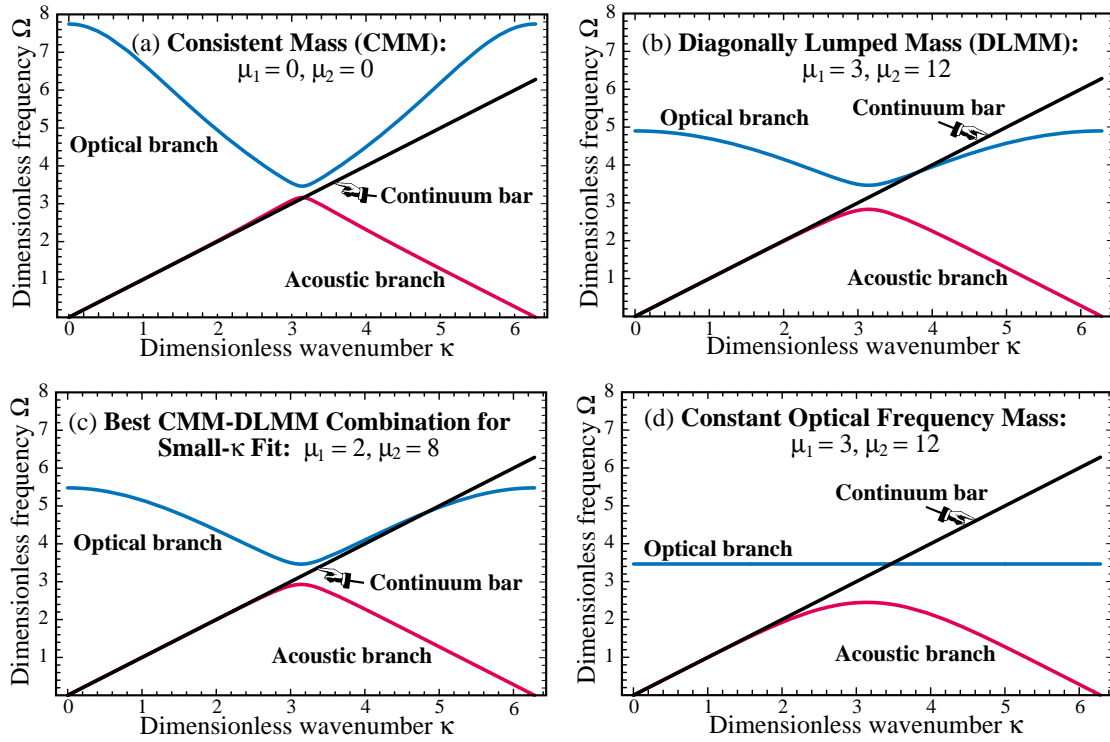


FIGURE 33.8. Dispersion curves of four mass matrices for the three-node prismatic bar, plotted for $\kappa \in [0, 2\pi]$. Acoustic branch in red, optical branch in blue, continuum bar line $\Omega = \kappa$ in black. Acoustic and optical branches repeat with period 2π ; note symmetry about $\kappa = \pi$.

For a mass matrix to produce fourth order accuracy in the acoustic branch, $C_4 = 0$ in the series (33.31). This has the two solutions $\mu_2 = 4\mu_1$ and $\mu_2 = 4\mu_1 - 240$. Both CM and DLM comply with the first solution. To get sixth order accuracy for small κ we impose $C_4 = C_6 = 0$. This has only two solutions: $\{\mu_1 = 2, \mu_2 = 8\}$ and $\{\mu_1 = 62, \mu_2 = 8\}$. Only the first solution is of interest, as the second one produces large positive-negative entries and exactly the same dispersion curves. The resulting mass matrix turns out to be a linear combination of CMM and DLMM. It is labeled BLC for “best lumped-consistent combination”:

$$\mathbf{M}_{\text{BLC}}^e = \frac{\rho A \ell}{90} \begin{bmatrix} 14 & -1 & 2 \\ -1 & 14 & 2 \\ 2 & 2 & 56 \end{bmatrix} = \frac{1}{3} \mathbf{M}_C^e + \frac{2}{3} \mathbf{M}_L^e. \quad (33.33)$$

As shown in Table 33.2, the acoustic branch of this matrix agrees up to $O(\kappa^6)$ with the continuum bar. The dispersion curves are shown in Figure 33.8(c).

A different kind of customization is advisable in dynamic simulations that involve propagation of high frequencies, such as shock and impact. The presence of the optical branch is undesirable, because it introduces spurious noise into the solutions. For such problems the two-node bar, which lacks an optical branch, should be used. If use of a three-node model is mandated for some reason, the harmful effects of the optical branch can be reduced by making it of constant frequency. Setting

$\{\mu_1 = 8, \mu_2 = 32\}$ produces the mass

$$\mathbf{M}_{\text{COF}}^e = \frac{\rho A \ell}{90} \begin{bmatrix} 10 & 5 & -10 \\ 5 & 10 & -10 \\ -10 & -10 & 80 \end{bmatrix}, \quad (33.34)$$

in which acronym COF stands for “Constant Optical Frequency.” Then $\Omega_o^2 = 12$ for all wavenumbers, as pictured in Figure 33.8(d). This configuration maximizes the stopping band and facilitates the implementation of a narrow band filter centered at that frequency. The acoustic branch accuracy is inferior to that of the other models, however, so this customization involves a tradeoff.

One final parameter choice is worth mentioning as a curiosity. Setting $\{\mu_1 = -2, \mu_2 = -8\}$ produces a dispersion diagram *with no stopping band*: the optical branch comes down from $+\infty$ at $\kappa = 0, 2\pi$ and merges with the acoustic branch at $\kappa = \pi$. The application of this mass matrix (which is singular) as a modeling tool is presently unclear and its dispersion diagram is omitted.

§33.6. The Bernoulli-Euler Beam

The Bernoulli-Euler beam model is a special case of the Timoshenko beam treated in the next section. It is nonetheless useful to do its mass template first, since results provide a valuable cross check with the more complicated Timoshenko beam. We take the consistent mass matrix derived in the previous chapter, and modify their entries to produce the following entry-weighted template:

$$\mathbf{M}_\mu^e = \rho A \ell \begin{bmatrix} \frac{13}{35} + \mu_{11} & (\frac{11}{210} + \mu_{12})\ell & \frac{9}{70} + \mu_{13} & -(\frac{13}{420} + \mu_{14})\ell \\ & (\frac{1}{105} + \mu_{22})\ell^2 & (\frac{13}{420} + \mu_{23})\ell & -(\frac{1}{140} + \mu_{24})\ell^2 \\ & & \frac{13}{35} + \mu_{11} & -(\frac{11}{210} + \mu_{12})\ell \\ \text{symm} & & & (\frac{1}{105} + \mu_{22})\ell^2 \end{bmatrix} \quad (33.35)$$

The parameters in (33.35) are μ_{ij} , where ij identifies the mass matrix entry. The template (33.35) accounts for matrix symmetry and some physical symmetries. Three more conditions can be imposed right away: $\mu_{14} = \mu_{23}$, $\mu_{13} = -\mu_{11}$ and $2\mu_{12} = \mu_{11} + 2\mu_{22} + 2\mu_{23} - 2\mu_{24}$. The first comes from beam symmetry and the others from conservation of total translational mass and angular momentum, respectively. This reduces the free parameters to four: $\{\mu_{11}, \mu_{22}, \mu_{23}, \mu_{24}\}$. The Fourier analysis procedure should be familiar by now to the reader. An infinite lattice of identical beam elements of length ℓ is set up. Plane waves of wavenumber k and frequency ω propagating over the lattice are represented by

$$v(x, t) = B_v \exp(i(kx - \omega t)), \quad \theta(x, t) = B_\theta \exp(i(kx - \omega t)) \quad (33.36)$$

At a typical lattice node j there are two freedoms: v_j and θ_j . Two patch equations are extracted, and converted to dimensionless form on defining $\kappa = k\ell$ and $\Omega = \omega c_0/\ell$, in which $c_0 = EI/(\rho A \ell^4)$ is a reference phase velocity. The condition for wave propagation gives the characteristic matrix equation

$$\det \begin{bmatrix} C_{vv} & C_{v\theta} \\ C_{\theta v} & C_{\theta\theta} \end{bmatrix} = C_{vv}C_{\theta\theta} - C_{v\theta}C_{\theta v} = 0, \quad (33.37)$$

where $C_{vv} = (840 - 2(13 + 35\mu_{11})\Omega^2 - (840 + (9 - 70\mu_{11})\Omega^2) \cos \kappa)/35$, $-C_{\theta v} = C_{v\theta} = i(2520 + (13 + 420\mu_{23})\Omega^2) \sin \kappa/210$, $C_{\theta\theta} = (1680 - 4(1 + 105\mu_{22})\Omega^2 + (840 + 3(1 + 140\mu_{24})\Omega^2) \cos \kappa)/210$. The condition (33.37) gives a quadratic equation in Ω^2 that provides two dispersion solutions: acoustical branch $\Omega_a^2(\kappa)$ and optical branch $\Omega_o^2(\kappa)$. These were already encountered in the analysis of the 3-node bar in §33.2. The acoustical branch represent genuine flexural modes, whereas the optical one is a spurious byproduct of the discretization. The small- κ (long wavelength) expansions of these roots are

$$\Omega_a^2 = \kappa^4 + C_6\kappa^6 + C_8\kappa^8 + C_{10}\kappa^{10} + C_{12}\kappa^{12} + \dots, \quad \Omega_o^2 = D_0 + D_2\kappa^2 + D_4\kappa^4 + \dots, \quad (33.38)$$

in which $C_6 = -\mu_{11} - 2\mu_{22} - 4\mu_{23} + 2\mu_{24}$, $C_8 = 1/720 + \mu_{11}^2 + 4\mu_{22}^2 + 2\mu_{23}/3 + 16\mu_{22}\mu_{23} + 16\mu_{23}^2 + \mu_{11}(1/12 + 4\mu_{22} + 8\mu_{23} - 4\mu_{24}) - \mu_{24} - 8\mu_{22}\mu_{24} - 16\mu_{23}\mu_{24} + 4\mu_{24}^2$, etc.; and $D_0 = 2520/(1 + 420\mu_{22} - 420\mu_{24})$, etc. *Mathematica* calculated these series up to C_{14} and D_4 .

The continuum dispersion curve is $\Omega^2 = \kappa^4$, which automatically matches Ω_a^2 as $\kappa \rightarrow 0$. Thus four free parameters offer the opportunity to match coefficients of four powers: $\{\kappa^6, \kappa^8, \kappa^{10}, \kappa^{12}\}$. But it will be seen that the last match is unfeasible if \mathbf{M}^e is to stay nonnegative. We settle for a scheme that agrees up to κ^{10} . Setting $C_6 = C_8 = C_{10} = 0$ while keeping μ_{22} free yields two sets of solutions, of which the useful one is

$$\begin{aligned} \mu_{11} &= 4\mu_{22} - 67/540 - (4/27)\sqrt{38/35 - 108\mu_{22}}, \\ \mu_{23} &= 43/1080 - 2\mu_{22} + \sqrt{95/14 - 675\mu_{22}}/54, \\ \mu_{24} &= 19/1080 - \mu_{22} + \sqrt{19/70 - 27\mu_{22}}/27. \end{aligned} \quad (33.39)$$

The positivity behavior of \mathbf{M}_μ^e as μ_{22} is varied is shown in Figure 33.9(a). $\mathbf{M}^{(e)}$ is indefinite for $\mu_{22} < \mu_{22}^{\min} = (27 - 4\sqrt{35})/5040 = 0.0006618414419844316$. At the other extreme the solutions of (33.39) become complex if $\mu_{22} > \mu_{22}^{\max} = 19/1890 = 0.010052910052910053$.

Figure 33.9(b) plots $C_{12}(\mu_{22}) = (-111545 - 3008\psi + 15120(525 + 4\psi)\mu_{22})/685843200$, with $\psi = \sqrt{70}\sqrt{19 - 1890\mu_{22}}$. This has one real root $\mu_{22}^z = -0.02830257472322391$, but that gives an indefinite mass matrix. For μ_{22} in the legal range $[\mu_{22}^{\min}, \mu_{22}^{\max}]$, C_{12} is minimized for $\mu_{22}^b = (25\sqrt{105} - 171)/30240 = 0.0028165928951385567$, which substituted gives the optimal mass matrix:

$$\mathbf{M}_B^e = \frac{\rho A \ell}{30240} \begin{bmatrix} a_{11} & 1788\ell & a_{13} & -732\ell \\ & a_{22}\ell^2 & 732\ell & a_{24}\ell^2 \\ & & a_{33} & 1788\ell \\ \text{symm} & & & a_{44}\ell^2 \end{bmatrix} = \rho A \ell \begin{bmatrix} 0.389589 & 0.059127\ell & 0.110410 & -0.024206\ell \\ & 0.012340\ell^2 & 0.024206\ell & -0.005548\ell^2 \\ & & 0.389589 & -0.059127\ell \\ & & & 0.012340\ell^2 \end{bmatrix} \quad (33.40)$$

in which $a_{11} = a_{33} = 12396 - 60\sqrt{105}$, $a_{13} = 2724 + 60\sqrt{105}$, $a_{22} = a_{44} = 117 + 25\sqrt{105}$ and $a_{24} = -219 + 5\sqrt{105}$. For this set, $C_{12} = (25\sqrt{105} - 441)/91445760 = -2.02 \cdot 10^{-6}$. Another interesting value is $\mu_{22} = 13/3150 = 0.004126984126984127$, which substituted in (33.39) yields rational values for the other parameters: $\mu_{11} = -\mu_{13} = 23/2100$, $\mu_{12} = -\mu_{14} = -\mu_{23} = 23/4200$, $\mu_{24} = 23/4200$ and $\mu_{24} = -17/12600$. Replacing into the template gives

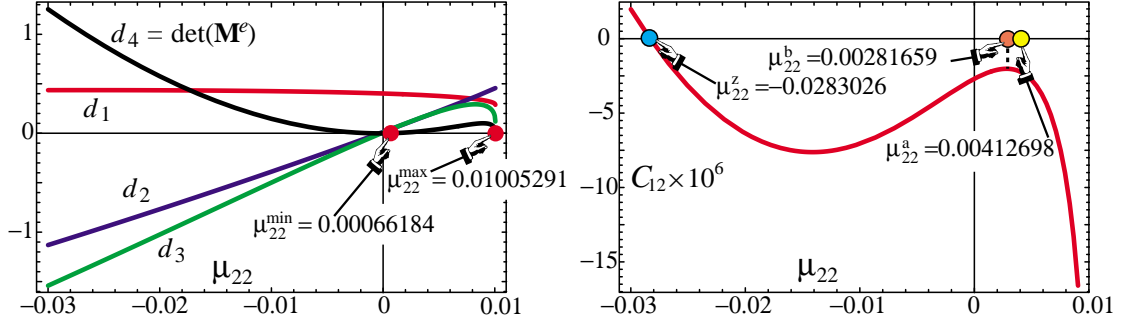


FIGURE 33.9. Behavior of \mathbf{M}_μ^e as function of μ_{22} when other μ parameters are picked from (33.39): (a) determinants d_k of principal minors of order k of \mathbf{M}_μ^e , showing legal positivity range $\{\mu_{22}^{\min}, \mu_{22}^{\max}\}$; (b) coefficient C_{12} of κ^{12} in Ω_a^2 series.

$$\mathbf{M}_Q^e = \frac{\rho A \ell}{12600} \begin{bmatrix} 4818 & 729\ell & 1482 & -321\ell \\ & 172\ell^2 & 321\ell & -73\ell^2 \\ & & 4818 & -729\ell \\ \text{symm} & & & 172\ell^2 \end{bmatrix} = \rho A \ell \begin{bmatrix} 0.382381 & 0.057857\ell & 0.117619 & -0.025476\ell \\ & 0.013651\ell^2 & 0.025476\ell & -0.005794\ell^2 \\ & & 0.382381 & -0.057857\ell \\ \text{symm} & & & 0.013651\ell^2 \end{bmatrix} \quad (33.41)$$

For this matrix, $C_{12} = -41/18144000 = -2.25 \cdot 10^{-6}$, which is only about 10% higher than for the optimal mass. Since the entries are simpler, (33.41) is adopted as custom mass matrix and used as a baseline for the Timoshenko beam.

§33.7. *Two-Node Timoshenko Beam Element

The last example is far more elaborate than the previous ones. The goal is to construct a mass template for the prismatic, plane-beam Timoshenko model. This includes the Bernoulli-Euler model as special case, and consequently results can be crosschecked with those of the previous section. The continuum Timoshenko model is first examined in some detail, since frequency expansion formulas applicable to template customization by characteristic root fitting are not easily found in the literature.

§33.7.1. *Continuum Analysis

Consider a structural beam member modeled as a shear-flexible Timoshenko plane beam, as illustrated in Figure 33.10. This figure provides the notation used below. Section properties $\{\rho, E, A, A_s, I, I_R\}$ are constant along x . The beam is transversally loaded by line load $q(x, t)$ (not shown in figure), with dimension of force per length. The primary kinematic variables are the transverse deflection $v(x, t)$ and the total cross-section rotation $\theta(x, t) = v'(x, t) + \gamma(x, t)$, where $\gamma = V/(GA_s)$ is the mean shear rotation. The kinetic and potential energies in terms of those variables are

$$T[v, \theta] = \frac{1}{2} \int_0^L (\rho A \dot{v}^2 + \rho I_R \dot{\theta}^2) dx, \quad \Pi[v, \theta] = \int_0^L \left(\frac{1}{2} EI (v'')^2 + \frac{1}{2} GA_s (\theta - v')^2 - qv \right) dx. \quad (33.42)$$

where superposed dots denote time derivatives. The equations of motion (EOM) follow on forming the Euler-Lagrange equations from the Lagrangian $L = T - \Pi$:

$$\frac{\delta L}{\delta v} = 0 \rightarrow GA_s (\theta' - v'') + \rho A \ddot{v} = q, \quad \frac{\delta L}{\delta \theta} = 0 \rightarrow EI \theta'' + GA_s (v' - \theta) - \rho I_R \ddot{\theta} = 0. \quad (33.43)$$

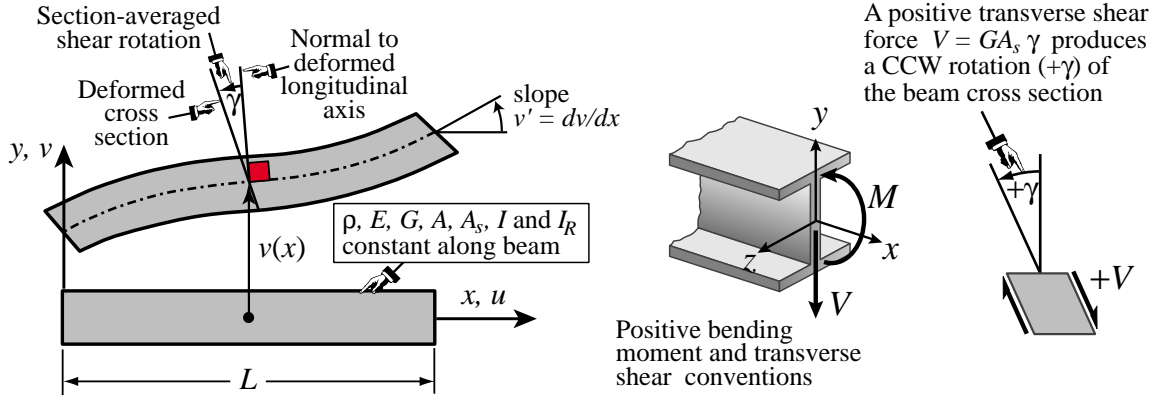


FIGURE 33.10. A plane beam member modeled as Timoshenko beam, illustrating notation followed in the continuum analysis. Transverse load $q(x)$ not shown to reduce clutter. Infinitesimal deflections and deformations grossly exaggerated for visibility.

An expedient way to eliminate θ is to rewrite the coupled equations (33.43) in transform space:

$$\begin{bmatrix} \rho A s^2 - G A_s p^2 & G A_s p \\ G A_s p & E I p^2 - G A_s - \rho I_R s^2 \end{bmatrix} \begin{bmatrix} \tilde{v} \\ \tilde{\theta} \end{bmatrix} = \begin{bmatrix} \tilde{q} \\ 0 \end{bmatrix}, \quad (33.44)$$

in which $\{p, s, \tilde{v}, \tilde{\theta}, \tilde{q}\}$ denote transforms of $\{d/dx, d/dt, v, \theta, q\}$, respectively (Fourier in x and Laplace in t). Eliminating $\tilde{\theta}$ and returning to the physical domain yields

$$E I v'''' + \rho A \ddot{v} - \left(\rho I_R + \frac{\rho A E I}{G A_s} \right) \ddot{v}'' + \frac{\rho^2 A I_R}{G A_s} \ddot{v} = q - \frac{E I}{G A_s} q'' + \frac{\rho I_R}{G A_s} \ddot{q}. \quad (33.45)$$

(Note that this derivation does not pre-assume $I \equiv I_R$, as usually done in textbooks.) For the unforced case $q = 0$, (33.45) has plane wave solutions $v = B \exp(i(k_0 x - \omega_0 t))$. The propagation condition yields a characteristic equation relating k_0 and ω_0 . To render it dimensionless, introduce a reference phase velocity $c_0^2 = E I / (\rho A L^4)$ so that $k_0 = \omega_0 / c_0 = 2\pi / \lambda_0$, a dimensionless frequency $\Omega = \omega_0 L / c_0$ and a dimensionless wavenumber $\kappa = k_0 L$. As dimensionless measures of relative bending-to-shear rigidities and rotary inertia take

$$\Phi_0 = 12 E I / (G A_s L^2), \quad r_R^2 = I_R / A, \quad \Psi_0 = r_R / L. \quad (33.46)$$

The resulting dimensionless characteristic equation is

$$\kappa^4 - \Omega^2 - \left(\frac{1}{12} \Phi_0 + \Psi_0^2 \right) \kappa^2 \Omega^2 + \frac{1}{12} \Phi_0 \Psi_0^2 \Omega^4 = 0. \quad (33.47)$$

This is quadratic in Ω^2 . Its solution yields two kinds of squared-frequencies, which will be denoted by Ω_f^2 and Ω_s^2 because they are associated with flexural and shear modes, respectively. Their expressions are listed below along with their small- κ (long wavelength) Taylor series:

$$\begin{aligned} \Omega_f^2 &= 6 \frac{P - \sqrt{Q}}{\Phi_0 \Psi_0^2} = \kappa^4 - \left(\frac{1}{12} \Phi_0 + \Psi_0^2 \right) \kappa^6 + \left(\frac{1}{144} \Phi_0^2 + \frac{1}{4} \Phi_0 \Psi_0^2 + \Psi_0^4 \right) \kappa^8 \\ &\quad - \left(\frac{1}{1728} \Phi_0^3 + \frac{1}{24} \Phi_0^2 \Psi_0^2 + \frac{1}{2} \Phi_0 \Psi_0^4 + \Psi_0^6 \right) \kappa^{10} + \dots = A_4 \kappa^4 + A_6 \kappa^6 + A_8 \kappa^8 + \dots \end{aligned} \quad (33.48)$$

$$\Omega_s^2 = 6 \frac{P + \sqrt{Q}}{\Phi_0 \Psi_0^2} = \frac{12}{\Phi_0 \Psi_0^2} + \left(\frac{12}{\Phi_0} + \frac{1}{\Psi_0^2} \right) \kappa^2 - \kappa^4 + \left(\frac{1}{12} \Phi_0 + \Psi_0^2 \right) \kappa^6 + \dots = B_0 + B_2 \kappa^2 + \dots \quad (33.49)$$

in which $P = 1 + \kappa^2 (\Psi_0^2 + \frac{1}{12} \Phi_0)$ and $Q = P^2 - \frac{1}{3} \kappa^4 \Phi_0 \Psi_0^2$. The dispersion relation $\Omega_f^2(\kappa)$ defines the *flexural frequency branch* whereas $\Omega_s^2(\kappa)$ defines the *shear frequency branch*. If $\Phi_0 \rightarrow 0$ and $\Psi_0 \rightarrow 0$, which

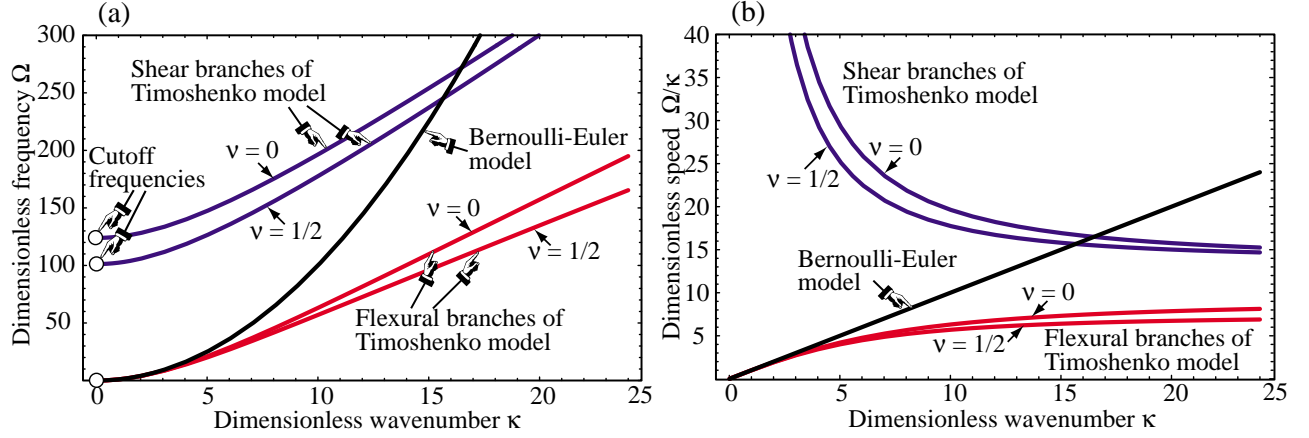


FIGURE 33.11. Spectral behavior of continuum Timoshenko beam model for a narrow $b \times h$ rectangular cross section. (a): dispersion curves $\Omega(\kappa)$ for $\Lambda = h/\ell = 1/4$ and two Poisson's ratios; Timoshenko flexural and shear branches in red and blue, respectively; Bernoulli-Euler curve $\Omega = \kappa^2$ in black. (b) Wavespeed Ω/κ .

reduces the Timoshenko model to the Bernoulli-Euler one, (33.47) collapses to $\Omega^2 = \kappa^4$ or (in principal value) $\Omega = \kappa^2$. This surviving branch pertains to flexural motions while the shear branch disappears — or more precisely, $\Omega_s^2(\kappa) \rightarrow +\infty$. It is easily shown that the radicand Q in the exact expressions is strictly positive for any $\{\Phi_0 > 0, \Psi_0 > 0, \kappa \geq 0\}$. Thus for any such triple, Ω_f^2 and Ω_s^2 are real, finite and distinct with $\Omega_f^2(\kappa) < \Omega_s^2(\kappa)$; furthermore $\{\Omega_f^2, \Omega_s^2\}$ increase indefinitely as $\kappa \rightarrow \infty$. Following the nomenclature introduced in Figure 33.7, the value Ω_s at $\kappa = 0$ is called the cutoff frequency.

To see how branches look like, consider a beam of narrow rectangular cross section of width b and height h , fabricated of isotropic material with Poisson's ratio ν . We have $E/G = 2(1 + \nu)$ and $A_s/A \approx 5/6$. [Actually a more refined A_s/A ratio would be $10(1 + \nu)/(12 + 11\nu)$, but that makes little difference in the results.] We have $A = bh$, $I = I_R = bh^3/12$, $r_R^2 = I_R/A = h^2/12$, $\Psi_0^2 = r_R^2/L^2 = \frac{1}{12}h^2/L^2$ and $\Phi_0 = 12EI/(GA_sL^2) = 12(1 + \nu)h^2/(5L^2)$. Since $\Phi_0/12 = 12(1 + \nu)\Psi_0^2/5$, the first-order effect of shear on Ω_f^2 , as measured by the κ^6 term in (33.48), is 2.4 to 3.6 times that from rotary inertia, depending on ν . Replacing into (33.48) and (33.49) yields

$$\left. \begin{aligned} \Omega_f^2 \\ \Omega_s^2 \end{aligned} \right\} &= \frac{60 + \kappa^2(17 + 12\nu)\Lambda^2 \mp \sqrt{(60 + \kappa^2(17 + 12\nu)\Lambda^2)^2 - 240\kappa^4(1 + \nu)\Lambda^4}}{2(1 + \nu)\Lambda^4} \\ &= \begin{cases} \kappa^4 - \frac{1}{60}(17 + 12\nu)\Lambda^2\kappa^6 + \frac{1}{3600}(349 + 468\nu + 144\nu^2)\Lambda^4\kappa^8 + \dots \\ \frac{60 + (17 + 12\nu)\Lambda^2\kappa^2 - (1 + \nu)\Lambda^4\kappa^4 + \dots}{(1 + \nu)\Lambda^4} \end{cases} \quad (33.50)$$

in which $\Lambda = h/L$. Dispersion curves $\Omega(\kappa)$ for $\Lambda = h/L = 1/4$ and $\nu = \{0, 1/2\}$ are plotted in Figure 33.11(a). Phase velocities Ω/κ are shown in Figure 33.11(b). The figure also shows the flexural branch of the Bernoulli-Euler model. The phase velocities of the Timoshenko model tend to finite values in the shortwave, high-frequency limit $\kappa \rightarrow \infty$, which is physically correct. The Bernoulli-Euler model is wrong in that limit because it predicts an infinite propagation speed.

§33.7.2. *Beam Element

The shear-flexible plane beam member of Figure 33.10 is discretized by two-node elements. An individual element of this type is shown in Figure 33.12, which illustrates its kinematics. The element has four nodal freedoms arranged as

$$\mathbf{u}^e = [v_1 \ \theta_1 \ v_2 \ \theta_2]^T \quad (33.51)$$

Here $\theta_1 = v_1 + \gamma_1$ and $\theta_2 = v_2 + \gamma_2$ are the total cross section rotations evaluated at the end nodes.

The dimensionless properties (33.46) that characterize relative shear rigidity and rotary inertia are redefined using the element length:

$$\Phi = 12EI/(GA_s \ell^2), \quad r_R^2 = I_R/A, \quad \Psi = r_R/\ell. \quad (33.52)$$

If the beam member is divided into N_e elements of equal length, $\ell = L/N_e$ whence $\Phi = \Phi_0 N_e^2$ and $\Psi = \Psi_0 N_e$. Thus even if Φ_0 and Ψ_0 are small with respect to one, they can grow without bound as the mesh is refined. For example if $\Phi_0 = 1/4$ and $\Psi_0^2 = 1/100$, which are typical values for a moderately thick beam, and we take $N_e = 32$, then $\Phi \approx 250$ and $\Psi^2 \approx 10$. Those are no longer small numbers, a fact that will impact performance as N_e increases. The stiffness matrix to be paired with the mass template is taken to be that of the equilibrium element:

$$\mathbf{K}^e = \frac{EI}{\ell^3(1 + \Phi)} \begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & \ell^2(4 + \Phi) & -6\ell & \ell^2(2 - \Phi) \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & \ell^2(2 - \Phi) & -6\ell & \ell^2(4 + \Phi) \end{bmatrix} \quad (33.53)$$

This is known to be nodally exact in static analysis for a prismatic beam member, and therefore an optimal choice in that sense.

§33.7.3. *Setting Up the Mass Template

FEM derivations usually split the 4×4 mass matrix of this element into $\mathbf{M}^e = \mathbf{M}_v^e + \mathbf{M}_\theta^e$, where \mathbf{M}_v^e and \mathbf{M}_θ^e come from the translational inertia and rotary inertia terms, respectively, of the kinetic energy functional $T[v, \theta]$ of (33.42). The most general mass template would result from applying an entry-weighted parametrization of those two matrices. This would require a set of 20 parameters (10 in each matrix), reducible to 9 through 11 on account of invariance and conservation conditions. Attacking the problem this way, however, leads to unwieldy algebraic equations even with the help of a computer algebra system, while concealing the underlying physics. A divide and conquer approach works better. This is briefly outlined next and covered in more detail in the next subsections.

(I) Express \mathbf{M}^e as the one-parameter matrix-weighted form $\mathbf{M}^e = (1 - \mu_0) \mathbf{M}_F^e + \mu_0 \mathbf{M}_D^e$. Here \mathbf{M}_F^e is full and includes the CMM as instance, whereas \mathbf{M}_D^e is 2×2 block diagonal and includes the DLMM as instance. This is plainly a generalization of the LC linear combination (33.2).

(II) Decompose the foregoing mass components as $\mathbf{M}_F^e = \mathbf{M}_{FT}^e + \mathbf{M}_{FR}^e$ and $\mathbf{M}_D^e = \mathbf{M}_{DT}^e + \mathbf{M}_{DR}^e$, where T and R subscripts identify their source in the kinetic energy functional: T if coming from the translational inertia term $\frac{1}{2} \rho A \dot{v}^2$ and R from the rotary inertia term $\frac{1}{2} \rho I_R \dot{\theta}^2$.

(III) Both components of \mathbf{M}_F^e are expressed as parametrized spectral forms, whereas those of \mathbf{M}_D^e are expressed as entry-weighted. The main reasons for choosing spectral forms for the full matrix are reduction of parameters and physical transparency. No such concerns apply to \mathbf{M}_D^e .

The analysis follows a “bottom up” sequence, in order (III)-(II)-(I). This has the advantage that if a satisfactory custom mass matrix for a target application emerges during (III), stages (II) and (I) need not be carried out, and that matrix directly used by setting the remaining parameters to zero.

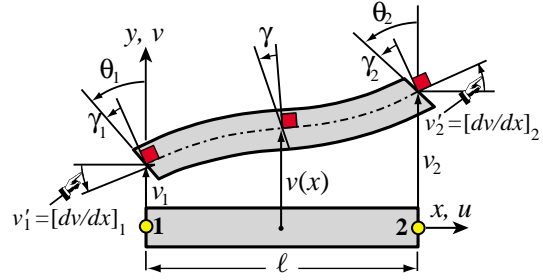


FIGURE 33.12. Two-node element for Timoshenko plane beam, illustrating kinematics.

§33.7.4. *Full Mass Parametrization

As noted above, one starts with full-matrix spectral forms. Let ξ denote the natural coordinate that varies from -1 at node 1 to $+1$ at node 2. Two element transverse displacement expansions in generalized coordinates are introduced:

$$\begin{aligned} v_T(\xi) &= L_1(\xi) c_{T1} + L_2(\xi) c_{T2} + L_3(\xi) c_{T3} + L_4(\xi) c_{T4} = \mathbf{L}_T \mathbf{c}_T, \\ v_R(\xi) &= L_1(\xi) c_{R1} + L_2(\xi) c_{R2} + L_3(\xi) c_{R3} + \widehat{L}_4(\xi) c_{R4} = \mathbf{L}_R \mathbf{c}_R, \\ L_1(\xi) &= 1, \quad L_2(\xi) = \xi, \quad L_3(\xi) = \frac{1}{2}(3\xi^2 - 1), \quad L_4(\xi) = \frac{1}{2}(5\xi^3 - 3\xi), \\ \widehat{L}_4(\xi) &= \frac{1}{2}(5\xi^3 - (5 + 10\Phi)\xi) = L_4(\xi) - (1 + 5\Phi)\xi. \end{aligned} \quad (33.54)$$

The v_T and v_R expansions are used for the translational and rotational parts of the kinetic energy, respectively. The interpolation function set $\{L_i\}$ used for v_T is formed by the first four Legendre polynomials over $\xi = [-1, 1]$. The set used for v_R is the same except that L_4 is adjusted to \widehat{L}_4 to produce a diagonal rotational mass matrix. All amplitudes c_{Ti} and c_{Ri} have dimension of length.

Unlike the usual Hermite cubic shape functions, the polynomials in (33.54) have a direct physical interpretation. L_1 : translational rigid mode; L_2 : rotational rigid mode; L_3 : pure-bending mode symmetric about $\xi = 0$; L_4 and \widehat{L}_4 : bending-with-shear modes antisymmetric about $\xi = 0$.

With the usual abbreviation $(\cdot)' \equiv d(\cdot)/dx = (2/\ell)d(\cdot)/d\xi$, the associated cross section rotations are

$$\theta_T = v'_T + \gamma_T = \mathbf{L}'_T \mathbf{c}_T + \gamma_T, \quad \theta_R = v'_R + \gamma_R = \mathbf{L}'_R \mathbf{c}_R + \gamma_R, \quad (33.55)$$

in which the mean shear distortions are constant over the element:

$$\gamma_T = \frac{\Phi \ell^2}{12} v'''_T = \frac{10\Phi}{\ell} c_{T4}, \quad \gamma_R = \frac{\Phi \ell^2}{12} v'''_R = \frac{10\Phi}{\ell} c_{R4}. \quad (33.56)$$

The kinetic energy of the element in generalized coordinates is

$$T^e = \frac{1}{2} \int_0^\ell (\rho A \dot{v}_T^2 + \rho I_R \dot{\theta}_R^2) dx = \frac{\ell}{4} \int_{-1}^1 (\rho A \dot{v}_T^2 + \rho I_R \dot{\theta}_R^2) d\xi = \frac{1}{2} \dot{\mathbf{c}}_T^T \mathbf{D}_T \dot{\mathbf{c}}_T + \frac{1}{2} \dot{\mathbf{c}}_R^T \mathbf{D}_R \dot{\mathbf{c}}_R, \quad (33.57)$$

in which both generalized mass matrices turn out to be diagonal as intended:

$$\mathbf{D}_T = \rho A \ell \mathbf{diag} \left[1 \quad \frac{1}{3} \quad \frac{1}{5} \quad \frac{1}{7} \right], \quad \mathbf{D}_R = 4\rho A \ell \Psi^2 \mathbf{diag} [0 \quad 1 \quad 3 \quad 5].$$

To convert \mathbf{D}_T and \mathbf{D}_R to physical coordinates (33.51), v_T , v_R , θ_T and θ_R are evaluated at the nodes by setting $\xi = \pm 1$. This establishes the transformations $\mathbf{u}^e = \mathbf{G}_T \mathbf{c}_T$ and $\mathbf{u}^e = \mathbf{G}_R \mathbf{c}_R$. Inverting: $\mathbf{c}_T = \mathbf{H}_T \mathbf{u}^e$ and $\mathbf{c}_R = \mathbf{H}_R \mathbf{u}^e$ with $\mathbf{H}_T = \mathbf{G}_T^{-1}$ and $\mathbf{H}_R = \mathbf{G}_R^{-1}$. A symbolic calculation yields

$$\begin{aligned} \mathbf{H}_T &= \frac{1}{60(1+\Phi)} \begin{bmatrix} 30(1+\Phi) & 5\ell(1+\Phi) & 30(1+\Phi) & -5\ell(1+\Phi) \\ -36-30\Phi & -3\ell & 36+30\Phi & -3\ell \\ 0 & -5\ell(1+\Phi) & 0 & 5\ell(1+\Phi) \\ 6 & 3\ell & -6 & 3\ell \end{bmatrix} \\ \mathbf{H}_R &= \frac{1}{60(1+\Phi)} \begin{bmatrix} 30(1+\Phi) & 5\ell(1+\Phi) & 30(1+\Phi) & -5\ell(1+\Phi) \\ -30 & 15\ell\Phi & 30 & 15\ell\Phi \\ 0 & -5\ell(1+\Phi) & 0 & 5\ell(1+\Phi) \\ 6 & 3\ell & -6 & 3\ell \end{bmatrix} \end{aligned} \quad (33.58)$$

Matrices \mathbf{H}_T and \mathbf{H}_R differ only in the second row. This comes from the adjustment of L_4 to \widehat{L}_4 in (33.54). To render this into a spectral template inject six free parameters in the generalized masses while moving $4\Psi^2$ inside $\mathbf{D}_{R\mu}$:

$$\mathbf{D}_{T\mu} = \rho A \ell \mathbf{diag} \left[1 \quad \frac{1}{3}\mu_{T1} \quad \frac{1}{5}\mu_{T2} \quad \frac{1}{7}\mu_{T3} \right], \quad \mathbf{D}_{R\mu} = \rho A \ell \mathbf{diag} [0 \quad \mu_{R1} \quad 3\mu_{R2} \quad 5\mu_{R3}]. \quad (33.59)$$

The transformation matrices (33.58) are reused without change to produce $\mathbf{M}_F^e = \mathbf{H}_T^T \mathbf{D}_{T\mu} \mathbf{H}_T + \mathbf{H}_R^T \mathbf{D}_{R\mu} \mathbf{H}_R$. If $\mu_{T1} = \mu_{T2} = \mu_{T3} = 1$ and $\mu_{R1} = \mu_{R2} = \mu_{R3} = 4\Psi^2$ one obtains the well known consistent mass matrix (CMM) of Archer, listed in [205, p. 296], as a valuable check. The configuration (33.59) already accounts for linear momentum conservation, which is why the upper diagonal entries are not parametrized. Imposing also angular momentum conservation requires $\mu_{T1} = 1$ and $\mu_{R1} = 4\Psi^2$, whence the template is reduced to four parameters:

$$\mathbf{M}_F^e = \rho A \ell \mathbf{H}_T^T \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{5}\mu_{T2} & 0 \\ 0 & 0 & 0 & \frac{1}{7}\mu_{T3} \end{bmatrix} \mathbf{H}_T + \rho A \ell \mathbf{H}_R^T \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 4\Psi^2 & 0 & 0 \\ 0 & 0 & 3\mu_{R2} & 0 \\ 0 & 0 & 0 & 5\mu_{R3} \end{bmatrix} \mathbf{H}_R. \quad (33.60)$$

Because both \mathbf{H}_T and \mathbf{H}_R are nonsingular, choosing all four parameters in (33.60) to be nonnegative guarantees that \mathbf{M}_F^e is nonnegative. This useful property eliminates lengthy *a posteriori* checks.

Setting $\mu_{T2} = \mu_{T3} = \mu_{R2} = \mu_{R3} = 0$ and $\Phi = 0$ yields the correct mass matrix for a rigid beam, including rotary inertia. This simple result highlights the physical transparency of spectral forms.

§33.7.5. *Block-Diagonal Mass Parametrization

Template (33.60) has a flaw: it does not include the DLMM. To remedy the omission, a block diagonal form, with four free parameters: $\{v_{T1}, v_{T2}, v_{R1}, v_{R2}\}$, is separately constructed:

$$\mathbf{M}_D^e = \mathbf{M}_{DT} + \mathbf{M}_{DR} = \rho A \ell \begin{bmatrix} 1/2 & v_{T1}\ell & 0 & 0 \\ v_{T1}\ell & v_{T2}\ell^2 & 0 & 0 \\ 0 & 0 & 1/2 & -v_{T1}\ell \\ 0 & 0 & -v_{T1}\ell & v_{T2}\ell^2 \end{bmatrix} + \rho A \ell \begin{bmatrix} 0 & v_{R1}\ell & 0 & 0 \\ v_{R1}\ell & v_{R2}\ell^2 & 0 & 0 \\ 0 & 0 & 0 & -v_{R1}\ell \\ 0 & 0 & -v_{R1}\ell & v_{R2}\ell^2 \end{bmatrix} \quad (33.61)$$

Four parameters can be merged into two by adding:

$$\mathbf{M}_D^e = \rho A \ell \begin{bmatrix} 1/2 & v_1\ell & 0 & 0 \\ v_1\ell & v_2\ell^2 & 0 & 0 \\ 0 & 0 & 1/2 & -v_1\ell \\ 0 & 0 & -v_1\ell & v_2\ell^2 \end{bmatrix}. \quad (33.62)$$

where $v_1 = v_{T1} + v_{R1}$ and $v_2 = v_{T2} + v_{R2}$. Sometimes it is convenient to use the split form (33.61), for example in lattices with varying beam properties or lengths, a topic not considered there. Otherwise (33.62) suffices. If $v_1 = 0$, \mathbf{M}_D^e is diagonal. However for computational purposes a block diagonal form is just as good and provides additional customization power. Terms in the (1,1) and (3,3) positions must be as shown to satisfy linear momentum conservation. If angular momentum conservation is imposed *a priori* it is necessary to set $v_2 = \frac{1}{2}\Psi^2$, and only one parameter remains.

The general template is obtained as a linear combination of \mathbf{M}_F^e and \mathbf{M}_D^e :

$$\mathbf{M}^e = (1 - \mu_0)\mathbf{M}_F^e + \mu_0\mathbf{M}_D^e \quad (33.63)$$

Summarizing there is a total of 7 parameters to play with: 4 in \mathbf{M}_F^e , 2 in \mathbf{M}_D^e , plus μ_0 . This is less than the 9-to-11 that would result from a full entry-weighted parametrization, so not all possible mass matrices are included by (33.63).

§33.7.6. *Fourier Analysis

An infinite lattice of identical beam elements of length ℓ is set up. Plane waves of wavenumber k and frequency ω propagating over the lattice are represented by

$$v(x, t) = B_v \exp(i(kx - \omega t)), \quad \theta(x, t) = B_\theta \exp(i(kx - \omega t)) \quad (33.64)$$

At each typical lattice node j there are two freedoms: v_j and θ_j . Two patch equations are extracted, and converted to dimensionless form on defining $\kappa = k\ell$ and $\Omega = \omega c/\ell$, in which $c = EI/(\rho A \ell^4)$ is a reference phase velocity. (Do not confuse with c_0). The condition for wave propagation gives the characteristic matrix equation

$$\det \begin{bmatrix} C_{vv} & C_{v\theta} \\ C_{\theta v} & C_{\theta\theta} \end{bmatrix} = C_{vv}C_{\theta\theta} - C_{v\theta}C_{\theta v} = 0, \quad (33.65)$$

where the coefficients are complicated functions not listed there. Solving the equation provides two equations: Ω_a^2 and Ω_o^2 , where a and o denote acoustic and optical branch, respectively. These are expanded in powers of κ for matching to the continuum. For the full mass matrix one obtains

$$\Omega_a^2 = \kappa^4 + C_6\kappa^6 + C_8\kappa^8 + C_{10}\kappa^{10} + \dots, \quad \Omega_o^2 = D_0 + D_2\kappa^2 + \dots \quad (33.66)$$

Coefficients up to κ^{12} were computed by *Mathematica*. Relevant ones for parameter selection are

$$\begin{aligned} C_6 &= -\Phi/12 - \Psi^2, \\ C_8 &= [2 - 15\mu_{R2} - \mu_{T2} + 5\Phi(1 + \Phi) + 60(1 + 3\Phi)\Psi^2 + 720\Psi^4]/720, \\ C_{10} &= [-44 + 35\mu_{T2} - 3\mu_{T3} - 282\Phi + 525\mu_{R2}(1 + \Phi) - 105\mu_{R3}(1 + \Phi) + \\ &\quad 1575\mu_{R2}\Phi(1 + \Phi) - \Phi(3\mu_{T3} - 35\mu_{T2}(4 + 3\Phi) + 35\Phi(17 + 5\Phi(3 + \Phi))) + \\ &\quad (-2940 + 12600\mu_{R2}(1 + \Phi) + 420(2\mu_{T2}(1 + \Phi) - 5\Phi(7 + 6\Phi(2 + \Phi))))\Psi^2 - \\ &\quad 25200(2 + \Phi(7 + 6\Phi))\Psi^4 - 302400(1 + \Phi)\Psi^6]/[302400(1 + \Phi)], \\ D_0 &= 25200(1 + \Phi)/[7 + 105\mu_{R2} + 3\mu_{T3} + 2100\Phi^2\Psi^2], \\ D_2 &= [2100(1 + \Phi)(-56 - 35\mu_{T2} + 3\mu_{T3} - 63\Phi + 3\mu_{T3}\Phi + 105\mu_{R3}(1 + \Phi) - \\ &\quad 525\mu_{R2}(1 + \Phi)^2 - 35\mu_{T2}\Phi(2 + \Phi)) + 2100(1 + \Phi)(3360\Phi + 6300\Phi^2 + \\ &\quad 2100\Phi^3)\Psi^2 + 52920000\Phi^2(1 + \Phi)\Psi^4]/[7 + 105\mu_{R2} + 3\mu_{T3} + 2100\Phi^2\Psi^2]^2 \end{aligned} \quad (33.67)$$

For the block-diagonal template (33.62):

$$\Omega_a^2 = \kappa^4 + F_6\kappa^6 + F_8\kappa^8 + F_{10}\kappa^{10} + \dots, \quad \Omega_o^2 = G_0 + G_2\kappa^2 + \dots \quad (33.68)$$

where

$$\begin{aligned} F_6 &= -24v_2 - \Phi, \quad F_8 = \frac{2880v_2 - 5\Phi + 360v_2\Phi - 1 - 5\Phi + 5\Phi^2}{720} \\ G_0 &= \frac{6}{v_2(1 + \Phi)}, \quad G_2 = \frac{24v_2 + \Phi - 2}{2v_2(1 + \Phi)} \end{aligned} \quad (33.69)$$

The expansions for the 7-parameter template (33.63) are considerably more complicated than the above ones, and are omitted to save space.

Table 33.3. Useful Template Instances for Timoshenko Beam Element

Instance name	Description	Comments
CMM	Consistent mass matrix of Archer. Matches flexural branch up to $O(\kappa^6)$.	A popular choice. Fairly inaccurate, however, as beam gets thicker. Grossly overestimates intermediate frequencies.
FBMS	Flexural-branch matched to $O(\kappa^{10})$ with spectral (Legendre) template (33.60).	Converges faster than CMM. Performance degrades as beam gets thicker, however, and element becomes inferior to CDLA.
SBM0	Shear branch matched to $O(\kappa^0)$ while flexure fitted to $O(\kappa^{10})$	Custom application: to roughly match shear branch and cutoff frequency as mesh is refined. Danger: indefinite for certain ranges of Φ and Ψ . Use with caution.
SBM2	Shear branch matched to $O(\kappa^2)$ while flexure fitted to $O(\kappa^8)$	Custom application: to finely match shear branch and cutoff frequency as mesh is refined. Danger: indefinite for wide ranges of Φ and Ψ . Use with extreme caution.
DLMM	Diagonally lumped mass matrix with rotational mass picked to match flexural branch to $O(\kappa^6)$.	Obvious choice for explicit dynamics. Accuracy degrades significantly, however, as beam gets thicker. Underestimates frequencies. Becomes singular in the Bernoulli-Euler limit.
CDLA	Average of CMM and DLMM. Matches flexure branch to $O(\kappa^8)$.	Robust all-around choice. Less accurate than FBMS and FBMG for thin beams, but becomes top performer as aspect ratio increases. Easily constructed if CMM and DLMM available in code.
FBMG	Flexural branch matched to $O(\kappa^{10})$ with 7-parameter template (33.63).	Known to be the globally optimal positive-definite choice for matching flexure in the Bernoulli-Euler limit. Accuracy, however, is only marginally better than FBMS. As in the case of the latter, performance degrades as beam gets thicker.

§33.7.7. *Template Instances

Seven useful instances of the foregoing templates are identified and described in Table 33.3. Table 33.4 lists the template signatures that generate those instances. These tables include two existing mass matrices (CMM and DLMM) re-expressed in the template context, and five new ones. The latter were primarily obtained by matching series such as (33.67) and (33.68) to the continuum ones (33.48) and (33.49), up to a certain number of terms as described in Table 33.3.

For the spectral template it is possible to match the flexure branch up to $O(\kappa^{10})$. Trying to match $O(\kappa^{12})$ leads to complex solutions. For the diagonal template the choice is more restrictive. It is only possible to match flexure up to $O(\kappa^6)$, which leads to instance DLMM. Trying to go further gives imaginary solutions. For

Table 33.4. Template Signatures for Mass Matrices of Table 33.3.

Instance name	Templ. form	Template signature							Fit to continuum freqs.	
		μ_{T2}	μ_{T3}	μ_{R2}	μ_{R3}	ν_1	ν_2	μ_0	Ω_f^2 (flexural)	Ω_s^2 (shear)
CMM	(33.60)	1	1	$4\Psi^2$	$4\Psi^2$				up to κ^6	none
FBMS	(33.60)	2	$26/3$	$4\Psi^2 + \Phi/3$	c_1				up to κ^{10}	none
SMB0	(33.60)	2	$-7/3$	$4\Psi^2 + \Phi/3$	$20\Phi\Psi^2$				up to κ^{10}	up to κ^0
SMB2	(33.60)	2	$-7/3$	c_2	$20\Phi\Psi^2$				up to κ^8	up to κ^2
DLMM	(33.62)					0	$\frac{1}{2}\Psi^2$		up to κ^6	none
CDLA	(33.63)	1	1	$4\Psi^2$	$4\Psi^2$	0	$\frac{1}{2}\Psi^2$	$1/2$	up to κ^8	none
FBMG	(33.63)	c_3	c_4	c_5	c_6	$1/12$	$\frac{1}{2}\Psi^2$	c_7	up to κ^{10}	none
$c_1 = (25\Phi^3 + 120\Psi^2 + \Phi^2(45 - 300\Psi^2) + 3\Phi(7 - 20\Psi^2 + 1200\Psi^4))/(15(1 + \Phi)),$ $c_2 = (-19 + 10\Phi^2(90\Psi^2 - 1) - 30\Phi(1 - 26\Psi^2 + 120\Psi^4))/(75(1 + \Phi)^2),$ $c_3 = (9 + \sqrt{105})/10, \quad c_4 = (61\sqrt{105} - 483)/18, \quad c_5 = (\sqrt{105} - 1)\Phi/30,$ $c_6 = (-48\Phi + 727\Phi^2 + 840\Phi^3 + 22128\Psi^2 + 19848\Phi\Psi^2 - 10080\Phi^2\Psi^2 - 113040\Psi^4$ $+ 120960\Phi\Psi^4 + 5\sqrt{105}(48\Phi + 87\Phi^2 + 40\Phi^3) - 24(6 + 21\Phi + 20\Phi^2)\Psi^2$ $+ 720(3 + 8\Phi)\Psi^4)/(60(21 + \sqrt{105})(1 + \Phi)), \quad c_7 = (3 - 5\sqrt{5/21})/8.$										

the 7-parameter template (33.63) it is again possible to match up to $O(\kappa^{10})$ but no further. The instance that exhibits least truncation error while retaining positivity is FBMG. This is globally optimal for the Bernoulli-Euler limit $\Phi = \Psi = 0$, but the results are only slightly better for the reasons discussed in 33.6.9. Matching both flexure and shear branches leads to instances SBM0 and SBM2, which have the disadvantages noted in Table 33.3.

The exact dispersion curves of these instances are shown in Figure 33.13 for $\Phi = 48/125$ and $\Psi^2 = 1/75$, which pertains to a thick beam. On examining Figure 33.13(c) it is obvious that trying to match the shear branch is difficult; the fit only works well over a tiny range near $\kappa = 0$.

§33.7.8. *Vibration Analysis Example

The performance of the seven instances of Tables 3–4 for vibration analysis is evaluated on a simply supported (SS) prismatic beam of length L divided into N_e equal elements. The cross section is rectangular with width b and height h . The material is isotropic with Poisson's ratio $\nu = 0$. Three different height-to-span ratios h/L , characterizing a thin, moderately thick and thick beam, respectively, are considered. Results for these configurations are collected in Figures 33.14, 33.15 and 33.16, respectively, for the first three vibration frequencies. All calculations are rendered dimensionless using the scaling techniques described previously.

Vibration accuracy is displayed as log-log plots of dimensionless natural frequency error versus N_e . The error is displayed as $d = \log_{10}(|\Omega_{comp} - \Omega_{exact}|)$, which gives at a glance the number of correct digits d , versus $\log_2 N_e$ for $N_e = 1$ to 32. Should the error be approximately controlled by a truncation term of the form $\propto \kappa^m$,

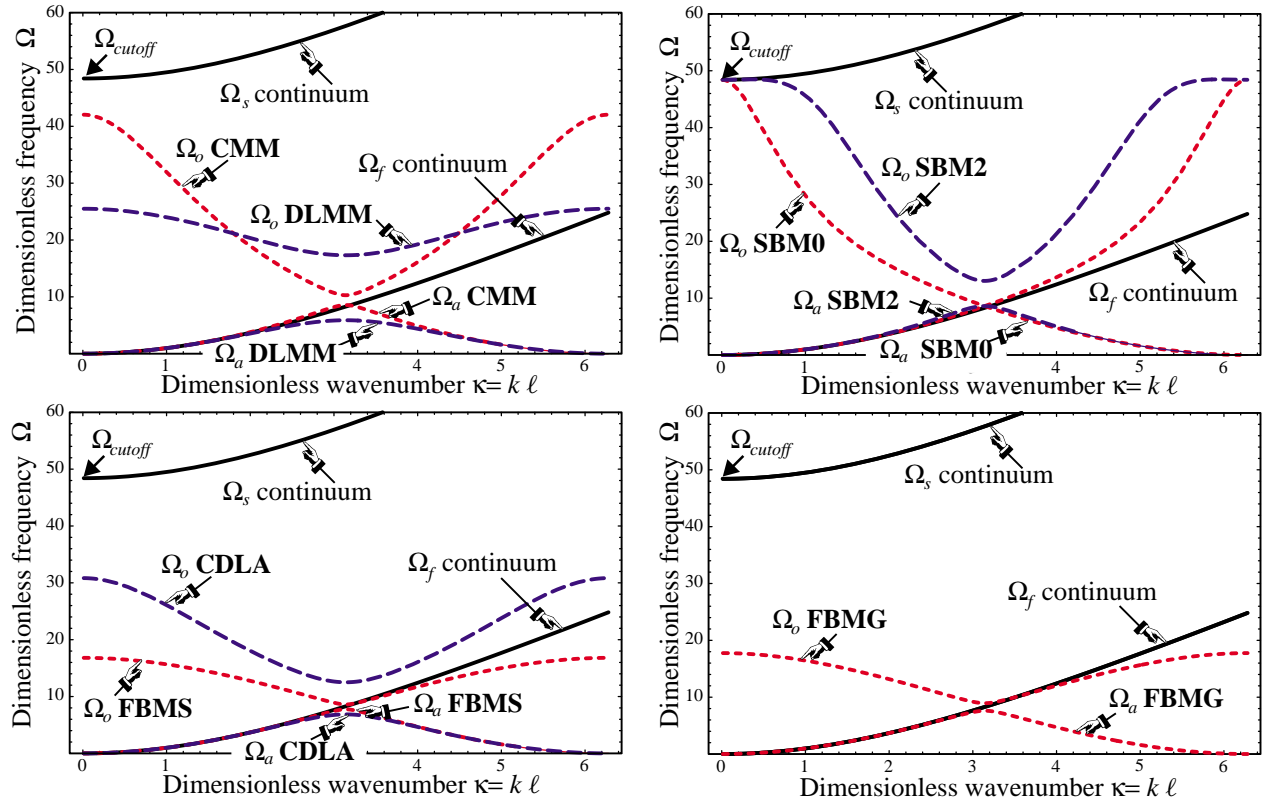


FIGURE 33.13. Dimensionless dispersion curves of Timoshenko mass matrices instances of Tables 33.3–33.4 for a thick beam with $\Phi_0 = 48/125 = 0.384$ and $\Psi_0^2 = 1/75 = 0.0133$. (a) Curves for standard consistent and diagonally-lumped matrices CMM and DLMM; (b) curves for the flexural-branch-matched FBMS and CDLA, (d) curves for the shear-branch-matched SBM0 and SBM2; (e) curve for flexure branch globally optimized FBMG.

the log-log plot should be roughly a straight line of slope $\propto m$, since $\kappa = k\ell = kL/N_e$.

The results for the Bernoulli-Euler model, shown in Figure 33.14, agree perfectly with the truncation error in the Ω_f^2 branch as listed in Table 33.4. For example, top performers FBMG and FBMS gain digits twice as fast as CMM, DLMM and SBM2, since the formers match Ω_f^2 to $O(\kappa^{10})$ whereas the latter do that only to $O(\kappa^6)$. Instances CDLA and SMB0, which agree through $O(\kappa^8)$, come in between. The highly complicated FBMG is only slightly better than the simpler FBMS. Their high accuracy case should be noted. For example, four FBMS elements give Ω_1 to six figures: 9.86960281... versus $\pi^2 = 9.86960440...$, whereas CMM gives less than three: 9.87216716.... The “accuracy ceiling” of about 11 digits for FBMS and FBMG observable for $N_e > 16$ is due to the eigensolver working in double precision (≈ 16 digits). Rerunning with higher (quad) floating point precision, the plots continues marching up as straight line before leveling at 25 digits.

On passing to the Timoshenko model, the well ordered Bernoulli-Euler world of Figure 33.14 unravels. The culprits are Φ and Ψ . These figure prominently in the branch series and grow without bound as N_e increases, as discussed in 33.6.2. Figure 33.15 collects results for a moderately thick beam with $h/L = 1/8$, which corresponds to $\Phi_0 = 3/80$ and $\Psi_0^2 = 1/768$. The Bernoulli-Euler top performers, FBMS and FBMG, gradually slow down and are caught by CDLA by $N_e = 32$. All other instances trail, with the standard ones: CMM and DLMM, becoming the worst performers. Note that for $N_e = 32$, CMM and DLMM provide only 1 digit of accuracy in Ω_3 although there are $32/1.5 \approx 21$ elements per wavelength.

Figure 33.16 collects results for a thick beam with $h/L = 2/5$, corresponding to $\Phi_0 = 24/625$ and $\Psi_0^2 = 1/75$. The foregoing trends are exacerbated, with FBMS and FBMG running out of steam by $N_e = 4$ and CDLA

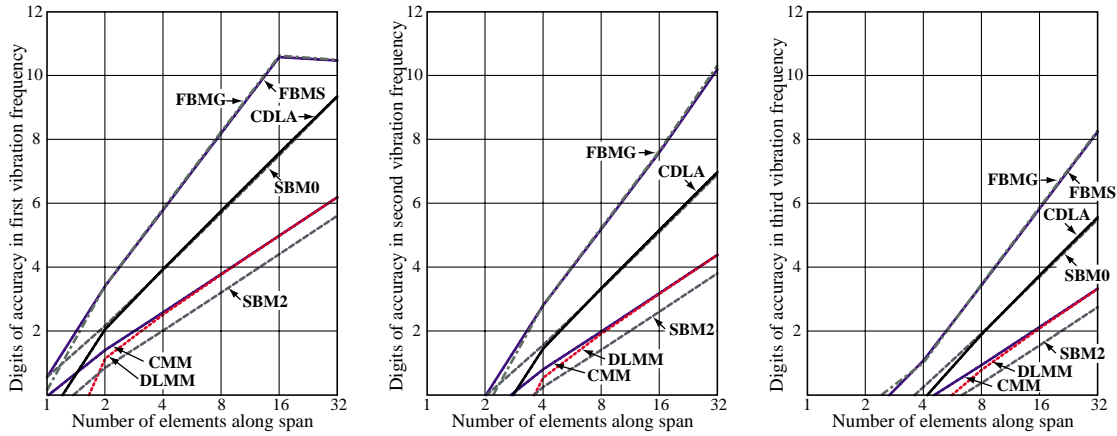


FIGURE 33.14. Accuracy of first 3 natural vibration frequencies of SS prismatic beam using mass matrices of Tables 33.3–33.4. Bernoulli-Euler model with $\Phi_0 = \Psi_0 = 0$. Exact (12-decimal) frequencies $\Omega_1 = \pi^2 = 9.869604401089$, $\Omega_2 = 4\pi^2 = 39.478417604357$ and $\Omega_3 = 9\pi^2 = 88.826439609804$. Cutoff frequency $+\infty$.

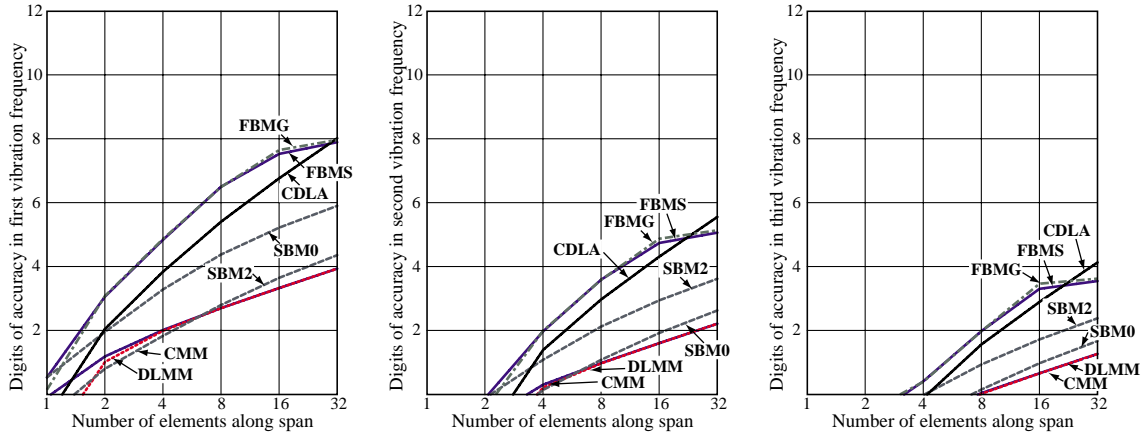


FIGURE 33.15. Accuracy of first 3 natural vibration frequencies of SS prismatic beam using mass matrices of Tables 33.3–33.4. Timoshenko model with $\Phi_0 = 3/80 = 0.0375$ and $\Psi_0^2 = 1/768 = 0.00130$, pertaining to a rectangular x-section with $h/L = 1/8$ and $\nu = 0$. Exact (12-decimal) frequencies $\Omega_1 = 9.662562122511$, $\Omega_2 = 36.507937703548$ and $\Omega_3 = 75.894968024537$. Cutoff frequency $\Omega_{cut} = 12/(\Phi_0 \Psi_0^2) = 495.741868314549$.

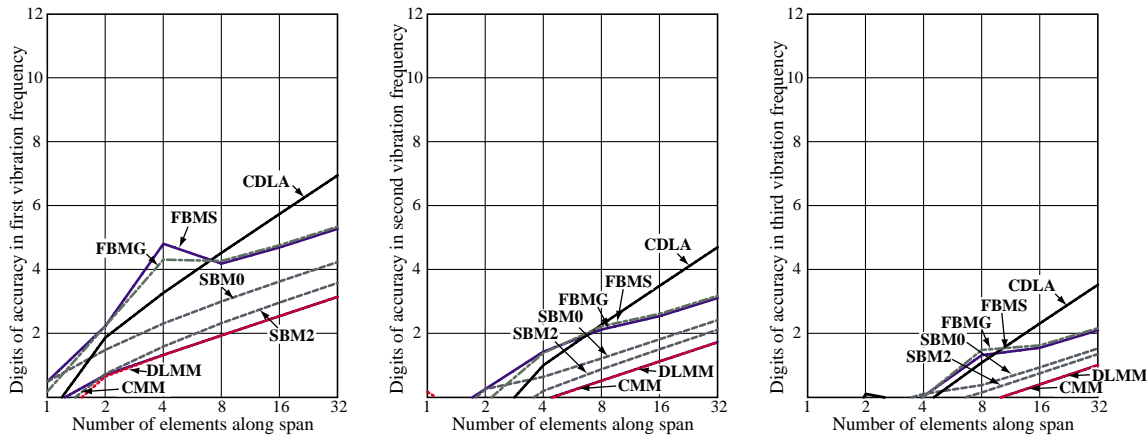


FIGURE 33.16. Accuracy of first 3 natural vibration frequencies of SS prismatic beam using mass matrices of Tables 33.3–33.4. Timoshenko model with $\Phi_0 = 24/625 = 0.384$ and $\Psi_0^2 = 1/75 = 0.0133$, pertaining to a rectangular x-section with $h/L = 2/5$ and $\nu = 0$. Exact (12-decimal) frequencies $\Omega_1 = 8.287891683498$, $\Omega_2 = 24.837128591729$ and $\Omega_3 = 43.182948411234$. Cutoff frequency $\Omega_{cut} = 12/(\Phi_0 \Psi_0^2) = 48.412291827593$.

emerging as best for $N_e \geq 8$. Again DCLM and CMM trail badly.

The reason for the performance degradation of FBMS and FBMG as the Timoshenko beam gets thicker is unclear. Eigensolver accuracy is not responsible since rerunning the cases of Figures 33.15 and 33.16 in quad precision did not change the plots. A numerical study of the Ω_f^2 truncation error shows that FBMS and FBMG fit the continuum branch better than CDLA even for very thick beams. Possible contamination of vibration mode shapes with the shear branch was not investigated.

Notes and Bibliography

The template approach addresses the deficiencies of the conventional mass models by using a parametrization approach that permits customization of that matrix to the problem and solution method at hand. The method was originally developed to construct high-performance stiffness matrices; a historical account and pertinent references are provided in a recent tutorial [92]. For stiffness-mass pairs it was used in [85,88] for a Bernoulli-Euler plane beam using Fourier analysis. One idea developed in those papers but not pursued here was to include the stiffness matrix template in the customization process. This provides more flexibility but has a negative side: highly optimized stiffness-mass pairs become sensitive to mesh distortion.

The symbolic derivation scheme used for the EOM (33.45) is due to Flaggs [95]; see also [189].

Making \mathbf{K} and \mathbf{M} frequency dependent was first proposed by Przemieniecki [205], who expanded both \mathbf{K}^e and \mathbf{M}^e as Taylor series in ω^2 . The idea was applied to eigenfrequency analysis of bars and beams, but not pursued further. The approach can be generalized to the template context by making free parameters frequency dependent, as illustrate in the two-node bar example. This may be of interest for problems dominated by a single driving frequency, as in some electronic and optical components. For more general use keeping the parameters frequency independent, as done in the last two examples, appears to be more practical.

Two powerful customization techniques used regularly for templates are Fourier methods and modified differential equations (MoDE). Fourier methods are limited to separable systems but can be straightforward to apply, requiring only undergraduate mathematics. (As tutorials for applied Fourier methods Hamming's textbooks [126,127], are recommended.) MoDE methods, first published in correct form in 1974 [262] are less restrictive but more demanding on two fronts: mathematical ability and support of a computer algebra system (CAS). Processing power limitations presently restrict MoDE to two-dimensional elements and regular meshes. The selection of template optimization criteria is not yet on firm ground. For example: is conservation of angular momentum useful in mass templates? The answer seems to depend on the element complexity.

Results for regular lattices of structural elements have direct counterparts in a very different area: molecular physics. More precisely, the wave mechanics of crystalline solids created in the XX Century by particle mechanicians; e.g., [35,282]. In crystal models, lattice nodes are occupied by molecules interacting with adjacent ones. Thus the "element dimension" ℓ acquires a physical meaning of molecular gap. In those applications masses are always *lumped* at molecule locations, and atoms vibrate as harmonic oscillators in the potential well of the force fields of their neighbors. Dispersion curves govern energy transmission. In a linear atomic chain, the wavenumber range $\kappa \in [-\pi, \pi]$ is called the first Brillouin zone [36,150]. Such a connection may be of interest as FEM and related discretization methods are extended into nano-mechanics.

References

Referenced items moved to Appendix R.

Homework Exercises for Chapter 33
Customized Mass Matrices of 1D Elements

EXERCISE 33.1 [A:10]. Express the Nyquist frequency for the 2-node bar lattice as function of μ .

EXERCISE 33.2 [A:30]. Investigate the behavior of an infinite 2-node bar lattice where element lengths alternate. Use either Fourier or MoDE.

EXERCISE 33.3 [A:25]. Investigate customized mass matrices for the 3-node bar without preimposing the angular momentum conservation condition $\mu_1 = \mu_3$. Use Fourier analysis on a lattice, starting from a 3-parameter mass template.

A

Linear Algebra: Vectors

TABLE OF CONTENTS

	Page
§A.1. Motivation	A–3
§A.2. Vectors	A–3
§A.2.1. Notational Conventions	A–4
§A.2.2. Visualization	A–5
§A.2.3. Special Vectors	A–5
§A.3. Vector Operations	A–5
§A.3.1. Transposition	A–6
§A.3.2. Equality	A–6
§A.3.3. Addition and Subtraction	A–6
§A.3.4. Multiplication and Division by Scalar	A–7
§A.3.5. Span	A–7
§A.3.6. Inner Product, Norm and Length	A–7
§A.3.7. Unit Vectors and Normalization	A–9
§A.3.8. Angles and Orthonormality	A–9
§A.3.9. Orthogonal Projection	A–10
§A.3.10. Orthogonal Bases and Subspaces	A–10
§A. Exercises	A–12
§A. Solutions to Exercises	A–13

§A.1. Motivation

Matrix notation was invented¹ primarily to express linear algebra relations in *compact form*. Compactness enhances visualization and understanding of essentials. To illustrate this point, consider the following set of m linear relations between one set of n quantities, x_1, x_2, \dots, x_n , and another set of m quantities, y_1, y_2, \dots, y_m :

$$\begin{array}{cccccccc}
 a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1j}x_j & + & \cdots & + & a_{1n}x_n & = & y_1 \\
 a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2j}x_j & + & \cdots & + & a_{2n}x_n & = & y_2 \\
 \cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots \\
 a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{ij}x_j & + & \cdots & + & a_{in}x_n & = & y_i \\
 \cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots & & \cdots \\
 a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mj}x_j & + & \cdots & + & a_{mn}x_n & = & y_m
 \end{array} \quad (\text{A.1})$$

The subscripted a , x and y quantities that appear in this set of relations may be formally arranged as follows:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_m \end{bmatrix}. \quad (\text{A.2})$$

Two kinds of mathematical objects can be distinguished in (A.2). The two-dimensional array expression enclosed in brackets is a *matrix*, which we call **A**. Matrices are defined and studied in Appendix B.

The one-dimensional array expressions in brackets are *column vectors* or simply *vectors*, which we call **x** and **y**, respectively. The use of **boldface** uppercase and lowercase letters to denote matrices and vectors, respectively, follows conventional matrix-notation rules in engineering applications. See §A.2.1 for conventions used in this book.

Replacing the expressions in (A.2) by these symbols we obtain the *matrix form* of (A.1):

$$\mathbf{A} \mathbf{x} = \mathbf{y}. \quad (\text{A.3})$$

Putting **A** next to **x** means “matrix product of **A** times **x**”, which is a generalization of the ordinary scalar multiplication, and follows the rules explained later.

Clearly (A.3) is a more compact, “short hand” form of (A.1).

Another key practical advantage of the matrix notation is that it translates directly to the computer implementation of linear algebra processes in languages that offer array data structures.

¹ Largely by Arthur Cayley (1821–1895) at Cambridge (UK) in 1858 [300] with substantial contributions from his colleague James Joseph Sylvester (1814–1897). See Appendix H for additional historical details related to the development of FEM. Curiously, vectors for mathematical physics (that is, living in 3D space) were independently created by J. Willard Gibbs (1839–1903) and Oliver Heaviside (1850–1925) in the late XIX century [168]. The overlap between matrices and vectors in n dimensions was not established until the XX century.

§A.2. Vectors

We begin by defining a *vector*, a set of n numbers which we shall write in the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}. \quad (\text{A.4})$$

This object is called a *column vector*. We shall see later that although a vector may be viewed as a special case of a matrix, it deserves treatment on its own. The symbol \mathbf{x} is the name of the vector.

If n numbers are arranged in a horizontal array, as in

$$\mathbf{z} = [z_1 \quad z_2 \quad \dots \quad z_n], \quad (\text{A.5})$$

then \mathbf{z} is called a *row vector*. If the term “vector” is used without a qualifier, it is understood to be a column vector such as (A.4).

§A.2.1. Notational Conventions

Typeset vectors will be designated by **bold** lowercase letters. For example:

$$\mathbf{a}, \quad \mathbf{b}, \quad \mathbf{c}, \quad \mathbf{x}, \quad \mathbf{y}, \quad \mathbf{z}. \quad (\text{A.6})$$

On the other hand, *handwritten or typewritten* vectors, which are those written on paper or on the blackboard, are identified by putting a wiggle or bar underneath the letter. For example:

$$\underline{a}. \quad (\text{A.7})$$

Subscripted quantities such as x_1 in (A.4) are called the *entries* or *components*² of \mathbf{x} , while n is called the *order* of the vector \mathbf{x} . Vectors of order one ($n = 1$) are called *scalars*. These are the usual quantities of analysis.

For compactness one sometimes abbreviates the phrase “vector of order n ” to just “ n -vector.” For example, \mathbf{z} in equation (A.8) below is a 4-vector.

If the components are real numbers, the vector is called a *real vector*. If the components are complex numbers we have a *complex vector*. Linear algebra embraces complex vectors as easily as it does real ones; however, we rarely need complex vectors for most of our exposition. Consequently real vectors will be assumed unless otherwise noted.

² The term *component* is primarily used in mathematical treatments whereas *entry* is used more in conjunction with the computer implementation. The term *element* is also used in the literature but this will be avoided here (as well as in companion books) as it may lead to confusion with finite elements.

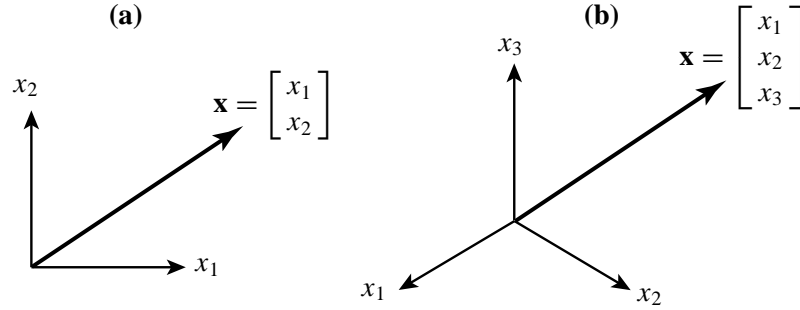


FIGURE A.1. Standard visualization of 2-vectors and 3-vectors as position vectors in 2-space and 3-space, respectively.

Example A.1.

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 3 \\ 0 \end{bmatrix} \quad (\text{A.8})$$

is real column vector of order 4, or, briefly, a 4-vector.

Example A.2.

$$\mathbf{q} = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1] \quad (\text{A.9})$$

is a real row vector of order 6.

Occasionally we shall use the short-hand “component notation”

$$\mathbf{x} = [x_i], \quad (\text{A.10})$$

for a generic vector. This comes handy when it is desirable to show the notational scheme obeyed by components.

§A.2.2. Visualization

To help visualization, a two-dimensional vector \mathbf{x} ($n = 2$) can be depicted as a line segment, or arrow, directed from the chosen origin to a point on the Cartesian plane of the paper with coordinates (x_1, x_2) . See Figure A.1(a). In mechanics this is called a *position vector* in 2-space.

One may resort to a similar geometrical interpretation in three-dimensional Cartesian space ($n = 3$). See Figure A.1(b). Drawing becomes a bit messier, however, although some knowledge of perspective and projective geometry helps.

The interpretation extends to Euclidean spaces of dimensions $n > 3$ but direct visualization is of course impaired.

§A.2.3. Special Vectors

The *null* vector, written $\mathbf{0}$, is the vector all of whose components are zero.

The *unit* vector, denoted by \mathbf{e}_i , is the vector all of whose components are zero, except the i^{th} component, which is one. After introducing matrices (Appendix B) a unit vector may be defined as the i^{th} column of the identity matrix.

The *unitary* vector, called \mathbf{e} , is the vector all of whose components are unity.

§A.3. Vector Operations

Operations on vectors in two-dimensional and three-dimensional space are extensively studied in courses on Mathematical Physics. Here we summarize operations on n -component vectors that are most useful from the standpoint of the development of finite elements.

§A.3.1. Transposition

The *transpose* of a column vector \mathbf{x} is the row vector that has the same components, and is denoted by \mathbf{x}^T :

$$\mathbf{x}^T = [x_1 \quad x_2 \quad \dots \quad x_n]. \quad (\text{A.11})$$

Similarly, the transpose of a row vector is the column vector that has the same components. Transposing a vector twice yields the original vector: $(\mathbf{x}^T)^T = \mathbf{x}$.

Example A.3. The transpose of

$$\mathbf{a} = \begin{bmatrix} 3 \\ -1 \\ 6 \end{bmatrix} \quad \text{is} \quad \mathbf{b} = \mathbf{a}^T = [3 \quad -1 \quad 6]$$

and transposing \mathbf{b} gives back \mathbf{a} .

§A.3.2. Equality

Two column vectors \mathbf{x} and \mathbf{y} of equal order n are said to be *equal* if and only if their components are equal, $x_i = y_i$, for all $i = 1, \dots, n$. We then write $\mathbf{x} = \mathbf{y}$. Similarly for row vectors.

Two vectors of different order cannot be compared for equality or inequality. A row vector cannot be directly compared to a column vector of the same order (unless their dimension is 1); one of the two has to be transposed before a comparison can be made.

§A.3.3. Addition and Subtraction

The simplest operation acting on two vectors is *addition*. The sum of two vectors of same order n , \mathbf{x} and \mathbf{y} , is written $\mathbf{x} + \mathbf{y}$ and defined to be the vector of order n

$$\mathbf{x} + \mathbf{y} \stackrel{\text{def}}{=} \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}. \quad (\text{A.12})$$

If \mathbf{x} and \mathbf{y} are not of the same order, the addition operation is undefined.

The operation is commutative: $\mathbf{x} + \mathbf{y} = \mathbf{y} + \mathbf{x}$, and associative: $\mathbf{x} + (\mathbf{y} + \mathbf{z}) = (\mathbf{x} + \mathbf{y}) + \mathbf{z}$.

Strictly speaking, the plus sign connecting \mathbf{x} and \mathbf{y} is not the same as the sign connecting x_i and y_i . However, since it enjoys the same analytical properties, there is no harm in using the same symbol in both cases. The geometric interpretation of the vector addition operator for two- and three-dimensional vectors ($n = 2, 3$) is the well known parallelogram law. See Figure A.2. For $n = 1$ the usual scalar addition results.

For vector subtraction, replace $+$ by $-$ in the foregoing expressions.

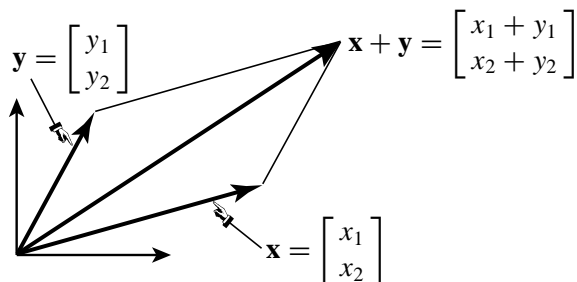


FIGURE A.2. In two and three-dimensional space, the vector addition operation is equivalent to the well known parallelogram composition law.

Example A.4. The sum of

$$\mathbf{a} = \begin{bmatrix} 3 \\ -1 \\ 6 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix} \quad \text{is} \quad \mathbf{a} + \mathbf{b} = \begin{bmatrix} 5 \\ 0 \\ 2 \end{bmatrix}$$

§A.3.4. Multiplication and Division by Scalar

Multiplication of a vector \mathbf{x} by a scalar c is defined by means of the relation

$$c \mathbf{x} \stackrel{\text{def}}{=} \begin{bmatrix} cx_1 \\ cx_2 \\ \vdots \\ cx_n \end{bmatrix} \quad (\text{A.13})$$

This operation is often called *scaling* of a vector. The geometrical interpretation of scaling is: the scaled vector points the same way, but its magnitude is multiplied by c .

If $c = 0$, the result is the null vector. If $c < 0$ the direction of the vector is reversed. In particular, if $c = -1$ the resulting operation $(-1)\mathbf{x} = -\mathbf{x}$ is called *reflexion about the origin* or simply *reflexion*.

Division of a vector by a scalar $c \neq 0$ is equivalent to multiplication by $1/c$. The operation is written

$$\frac{\mathbf{x}}{c} \equiv \mathbf{x}/c \stackrel{\text{def}}{=} (1/c)\mathbf{x} \quad (\text{A.14})$$

The operation is not defined if $c = 0$.

§A.3.5. Span

Sometimes it is not just \mathbf{x} which is of interest but the “line” determined by \mathbf{x} , namely the collection $c\mathbf{x}$ of all scalar multiples of \mathbf{x} , including $-\mathbf{x}$. We call this *Span*(\mathbf{x}). Note that the span always includes the null vector.

The span of two vectors, \mathbf{x} and \mathbf{y} with common origin, is the set of vectors $c_1 \mathbf{x} + c_2 \mathbf{y}$ for arbitrary scaling coefficients c_1 and c_2 . If the two vectors are not parallel, the geometric visualization of the span is that of the plane defined by the two vectors. Hence the statement: a plane is *spanned* by two noncoincident vectors, responds to this interpretation.

The span of an arbitrary number of vectors of common origin is studied in §A.3.10.

§A.3.6. Inner Product, Norm and Length

The *inner product* of two column vectors \mathbf{x} and \mathbf{y} , of same order n , is a scalar function denoted by (\mathbf{x}, \mathbf{y}) — as well as two other forms shown below — and is defined by

$$(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{y} = \mathbf{y}^T \mathbf{x} \stackrel{\text{def}}{=} \sum_{i=1}^n x_i y_i \stackrel{\text{sc}}{=} x_i y_i. \quad (\text{A.15})$$

This operation is also called *dot product* and *interior product*. If the two vectors are not of the same order, the inner product is undefined. The two other notations shown in (A.15), namely $\mathbf{x}^T \mathbf{y}$ and $\mathbf{y}^T \mathbf{x}$, exhibit the inner product as a special case of the *matrix product* discussed in Appendix B.

The last expression in (A.15) applies the so-called *Einstein's summation convention*, which implies sum on repeated indices (in this case, i) and allows the \sum operand to be dropped.

The inner product is commutative: $(\mathbf{x}, \mathbf{y}) = (\mathbf{y}, \mathbf{x})$ but not generally associative: $(\mathbf{x}, (\mathbf{y}, \mathbf{z})) \neq ((\mathbf{x}, \mathbf{y}), \mathbf{z})$. If $n = 1$ it reduces to the usual scalar product. The scalar product is of course associative.

Example A.5. The inner product of

$$\mathbf{a} = \begin{bmatrix} 3 \\ -1 \\ 6 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ -4 \end{bmatrix} \quad \text{is} \quad (\mathbf{a}, \mathbf{b}) = \mathbf{a}^T \mathbf{b} = 3 \times 2 + (-1) \times 1 + 6 \times (-4) = -19.$$

Remark A.1. The inner product is not the only way of “multiplying” two vectors. There are other vector products, such as the cross or outer product, which are important in many applications such as fluid mechanics and nonlinear dynamics. They are not treated here because they are not defined when going from vectors to matrices. Furthermore they are not easily generalized for vectors of order beyond 3.

The *Euclidean norm* or *2-norm* of a real vector \mathbf{x} is a scalar denoted by $\|\mathbf{x}\|$ that results by taking the inner product of the vector with itself:

$$\|\mathbf{x}\| \stackrel{\text{def}}{=} (\mathbf{x}, \mathbf{x}) = \sum_{i=1}^n x_i^2 \stackrel{\text{sc}}{=} x_i x_i. \quad (\text{A.16})$$

Because the norm is a sum of squares, it is zero only if \mathbf{x} is the null vector. It thus provides a “meter” (mathematically, a norm) on the vector magnitude.

The *Euclidean length* or simply *length* of a real vector, denoted by $|\mathbf{x}|$, is the positive square root of its Euclidean norm:

$$|\mathbf{x}| \stackrel{\text{def}}{=} +\sqrt{\|\mathbf{x}\|}. \quad (\text{A.17})$$

This definition agrees with the intuitive concept of vector magnitude in two- and three-dimensional space ($n = 2, 3$). For $n = 1$ observe that the length of a scalar x is its absolute value $|x|$, hence the notation $|\mathbf{x}|$.

The Euclidean norm is not the only vector norm used in practice, but it will be sufficient for the present book.

Two important inequalities satisfied by vector norms (and not just the Euclidean norm) are the *Cauchy-Schwarz inequality*:

$$|(\mathbf{x}, \mathbf{y})| \leq |\mathbf{x}| |\mathbf{y}|, \quad (\text{A.18})$$

and the *triangle inequality*:

$$|\mathbf{x} + \mathbf{y}| \leq |\mathbf{x}| + |\mathbf{y}|. \quad (\text{A.19})$$

Example A.6. The Euclidean norm of

$$\mathbf{x} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad (\text{A.20})$$

is $\|\mathbf{x}\| = 4^2 + 3^2 = 25$, and its length is $|\mathbf{x}| = \sqrt{25} = 5$.

§A.3.7. Unit Vectors and Normalization

A vector of length one is called a *unit vector*. Any non-null vector \mathbf{x} can be scaled to unit length by dividing all components by its original length:

$$\mathbf{x}/|\mathbf{x}| = \begin{bmatrix} x_1/|\mathbf{x}| \\ x_2/|\mathbf{x}| \\ \vdots \\ x_n/|\mathbf{x}| \end{bmatrix} \quad (\text{A.21})$$

This particular scaling is called *normalization to unit length*. If \mathbf{x} is the null vector, the normalization operation is undefined.

Example A.7. The unit-vector normalization of

$$\mathbf{x} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \quad \text{is} \quad \mathbf{x}/|\mathbf{x}| = \mathbf{x}/5 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} \quad (\text{A.22})$$

§A.3.8. Angles and Orthonormality

The *angle* in radians between two *unit* real vectors \mathbf{x} and \mathbf{y} , written $\angle(\mathbf{x}, \mathbf{y})$, is the real number θ satisfying $0 \leq \theta \leq \pi$ (or $0^\circ \leq \theta \leq 180^\circ$ if measured in degrees). The value is defined by the cosine formula:

$$\cos \theta = (\mathbf{x}, \mathbf{y}). \quad (\text{A.23})$$

If the vectors are not of unit length, they should be normalized to such, and so the general formula for two arbitrary vectors is

$$\cos \theta = \left(\frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\mathbf{y}}{|\mathbf{y}|} \right) = \frac{(\mathbf{x}, \mathbf{y})}{|\mathbf{x}| |\mathbf{y}|}. \quad (\text{A.24})$$

The angle θ formed by a non-null vector with itself is zero. This will always be the case for $n = 1$. If one of the vectors is null, the angle is undefined.

The definition (A.24) agrees with that of the angle formed by *oriented* lines in two- and three-dimensional Euclidean geometry ($n = 2, 3$). The generalization to n dimensions is a natural one.

For some applications the *acute* angle ϕ between the line on \mathbf{x} and the line on \mathbf{y} (i.e., between the vector spans) is more appropriate than θ . This angle between $\text{Span}(\mathbf{x})$ and $\text{Span}(\mathbf{y})$ is the real number ϕ satisfying $0 \leq \phi \leq \pi/2$ and

$$\cos \phi = \frac{|(\mathbf{x}, \mathbf{y})|}{|\mathbf{x}| |\mathbf{y}|} \quad (\text{A.25})$$

Two vectors \mathbf{x} and \mathbf{y} connected by the relation

$$(\mathbf{x}, \mathbf{y}) = 0, \quad (\text{A.26})$$

are said to be *orthogonal*. The acute angle formed by two orthogonal vectors is $\pi/2$ radians or 90° .

§A.3.9. Orthogonal Projection

The *orthogonal projection* of a vector \mathbf{y} onto a vector \mathbf{x} is the vector \mathbf{p} that has the direction of \mathbf{x} and is orthogonal to $\mathbf{y} - \mathbf{p}$. The condition can be stated as

$$(\mathbf{p}, \mathbf{y} - \mathbf{p}) = 0, \quad \text{in which} \quad \mathbf{p} = c \frac{\mathbf{x}}{|\mathbf{x}|}. \quad (\text{A.27})$$

It is easily shown that

$$c = (\mathbf{y}, \frac{\mathbf{x}}{|\mathbf{x}|}) = |\mathbf{y}| \cos \theta, \quad (\text{A.28})$$

where θ is the angle between \mathbf{x} and \mathbf{y} . If \mathbf{x} and \mathbf{y} are orthogonal, the projection of any of them onto the other vanishes.

§A.3.10. Orthogonal Bases and Subspaces

Let \mathbf{b}^k , $k = 1, \dots, m$ be a set of n -dimensional *unit* vectors which are *mutually orthogonal*. Then if a particular n -dimensional vector \mathbf{x} admits the representation

$$\mathbf{x} = \sum_{k=1}^m c_k \mathbf{b}^k \quad (\text{A.29})$$

the coefficients c_k are given by the inner products

$$c_k = (\mathbf{x}, \mathbf{b}^k) = \sum_{i=1}^n x_i b_i^k \stackrel{\text{sc}}{=} x_i b_i^k. \quad (\text{A.30})$$

We also have *Parseval's equality*

$$(\mathbf{x}, \mathbf{x}) = \sum_{k=1}^m c_k^2 \stackrel{\text{sc}}{=} c_k c_k. \quad (\text{A.31})$$

If the representation (A.29) holds, the set \mathbf{b}^k is called an *orthonormal basis* for the vector \mathbf{x} . The \mathbf{b}^k are called the *base vectors*.

The set of all vectors \mathbf{x} given by (A.30) forms a *subspace* of dimension m . The subspace is said to be *spanned* by the basis \mathbf{b}^k , and is called *Span* (\mathbf{b}^k).³ The numbers c_k are called the *coordinates* of \mathbf{x} with respect to that basis.

If $m = n$, the set \mathbf{b}^k forms a *complete orthonormal basis* for the n -dimensional space. (The qualifier “complete” means that all n -dimensional vectors are representable in terms of such a basis.)

The simplest complete orthonormal basis is of order n is the n -dimensional *Cartesian basis*

$$\mathbf{b}^1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad \mathbf{b}^2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \quad \dots \quad \mathbf{b}^n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}. \quad (\text{A.32})$$

In this case the *position coordinates* of \mathbf{x} are simply its components, that is, $c_k \equiv x_k$.

³ Note that if $m = 1$ we have the span of a single vector, which as noted in §A.3.5 is simply a line passing through the origin of coordinates.

Homework Exercises for Appendix A: Vectors**EXERCISE A.1** Given the four-dimensional vectors

$$\mathbf{x} = \begin{bmatrix} 2 \\ 4 \\ 4 \\ -8 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 1 \\ -5 \\ 7 \\ 5 \end{bmatrix} \quad (\text{EA.1})$$

- (a) compute the Euclidean norms and lengths of \mathbf{x} and \mathbf{y} ;
- (b) compute the inner product (\mathbf{x}, \mathbf{y}) ;
- (c) verify that the inequalities (A.18) (A.19) hold;
- (d) normalize \mathbf{x} and \mathbf{y} to unit length;
- (e) compute the angles $\theta = \angle(\mathbf{x}, \mathbf{y})$ and ϕ given by (?) and (?).
- (f) compute the projection of \mathbf{y} onto \mathbf{x} and verify that the orthogonality condition (A.28) holds.

EXERCISE A.2 Given the base vectors

$$\mathbf{b}^1 = \frac{1}{10} \begin{bmatrix} 1 \\ -7 \\ 1 \\ 7 \end{bmatrix}, \quad \mathbf{b}^2 = \frac{1}{10} \begin{bmatrix} 5 \\ 5 \\ -5 \\ 5 \end{bmatrix}. \quad (\text{EA.2})$$

- (a) Check that \mathbf{b}^1 and \mathbf{b}^2 are orthonormal, i.e., orthogonal and of unit length.
- (b) Compute the coefficients c_1 and c_2 in the following representation:

$$\mathbf{z} = \begin{bmatrix} 8 \\ -16 \\ -2 \\ 26 \end{bmatrix} = c_1 \mathbf{b}^1 + c_2 \mathbf{b}^2. \quad (\text{EA.3})$$

- (c) Using the values computed in (b), verify that the coordinate-expansion formulas and Parseval's equality are correct.

EXERCISE A.3 Prove that Parseval's equality holds in general.**EXERCISE A.4** What are the angles θ and ϕ formed by an arbitrary non-null vector \mathbf{x} and the opposite vector $-\mathbf{x}$?**EXERCISE A.5** Show that

$$(\alpha \mathbf{x}, \beta \mathbf{y}) = \alpha \beta (\mathbf{x}, \mathbf{y}) \quad (\text{EA.4})$$

where \mathbf{x} and \mathbf{y} are arbitrary vectors, and α and β are scalars.

Homework Exercises for Appendix A - Solutions

EXERCISE A.1

(a)

$$\|\mathbf{x}\| = 2^2 + 4^2 + 4^2 + (-8)^2 = 100, \quad |\mathbf{x}| = 10.$$

$$\|\mathbf{y}\| = 1^2 + (-5)^2 + 7^2 + 5^2 = 100, \quad |\mathbf{y}| = 10.$$

(b)

$$(\mathbf{x}, \mathbf{y}) = 2 - 20 + 28 - 40 = -30.$$

(c)

$$|(\mathbf{x}, \mathbf{y})| = 30 \leq |\mathbf{x}| |\mathbf{y}| = 100.$$

$$|\mathbf{x} + \mathbf{y}| = \sqrt{3^2 + (-1)^2 + (-11)^2 + (-3)^2} = \sqrt{140} \leq |\mathbf{x}| + |\mathbf{y}| = 10 + 10 = 20$$

(d)

$$\mathbf{x}'' = \frac{\mathbf{x}}{10} = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ -0.8 \end{bmatrix}, \quad \mathbf{y}'' = \frac{\mathbf{y}}{10} = \begin{bmatrix} 0.1 \\ -0.5 \\ 0.7 \\ 0.5 \end{bmatrix}$$

(e)

$$\cos \theta = (\mathbf{x}'', \mathbf{y}'') = -0.30, \quad \theta = 107.46^\circ.$$

$$\cos \phi = |(\mathbf{x}'', \mathbf{y}'')| = 0.30, \quad \phi = 72.54^\circ.$$

(f)

$$c = |\mathbf{y}| \cos \theta = -3.$$

$$\mathbf{p} = c \mathbf{x}'' = -3 \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ -0.8 \end{bmatrix} = \begin{bmatrix} -0.6 \\ -1.2 \\ -1.2 \\ 2.4 \end{bmatrix}.$$

$$\mathbf{y} - \mathbf{p} = \begin{bmatrix} 1.6 \\ -3.8 \\ 8.2 \\ 2.6 \end{bmatrix}.$$

$$(\mathbf{p}, \mathbf{y} - \mathbf{p}) = -0.96 + 4.56 - 9.84 + 6.24 = 0.$$

EXERCISE A.2

(a)

$$\|\mathbf{b}^1\| = 0.1^2 + (-0.7)^2 + 0.1^2 + 0.7^2 = 1, \quad |\mathbf{b}^1| = 1.$$

$$\|\mathbf{b}^2\| = 0.5^2 + 0.5^2 + (-0.5)^2 + 0.5^2 = 1, \quad |\mathbf{b}^2| = 1.$$

$$(\mathbf{b}^1, \mathbf{b}^2) = 0.05 - 0.35 - 0.05 + 0.35 = 0.$$

(b)

$$c_1 = 30, \quad c_2 = 10.$$

(by inspection, or solving a system of 2 linear equations)

(c)

$$c_1 = (\mathbf{z}, \mathbf{b}^1) = 0.8 + 11.2 - 0.2 + 18.2 = 30.$$

$$c_2 = (\mathbf{z}, \mathbf{b}^2) = 4.0 - 8.0 + 1.0 + 13.0 = 10.$$

$$c_1^2 + c_2^2 = 30^2 + 10^2 = 1000 = \|\mathbf{z}\|^2 = 8^2 + (-16)^2 + (-2)^2 + 26^2 = 1000.$$

Appendix A: LINEAR ALGEBRA: VECTORS

EXERCISE A.3 See any linear algebra book, e.g. [673].

EXERCISE A.4

$$\theta = 180^\circ, \quad \phi = 0^\circ.$$

EXERCISE A.5

$$(\alpha \mathbf{x}, \beta \mathbf{y}) = \alpha x_i \beta y_i = \alpha \beta x_i y_i = \alpha \beta (\mathbf{x}, \mathbf{y})$$

(summation convention used)

B

Linear Algebra: Matrices

TABLE OF CONTENTS

	Page
§B.1. Matrices	B–3
§B.1.1. Concept	B–3
§B.1.2. Real and Complex Matrices	B–3
§B.1.3. Square Matrices	B–3
§B.1.4. Symmetry and Antisymmetry	B–4
§B.1.5. Are Vectors a Special Case of Matrices?	B–4
§B.1.6. Where Do Matrices Come From?	B–5
§B.1.7. Special Matrices	B–5
§B.2. Elementary Matrix Operations	B–6
§B.2.1. Equality	B–6
§B.2.2. Transposition	B–6
§B.2.3. Addition and Subtraction	B–7
§B.2.4. Scalar Multiplication	B–7
§B.3. Matrix Products	B–7
§B.3.1. Matrix by Vector Product	B–7
§B.3.2. Matrix by Matrix Product	B–8
§B.3.3. Matrix Powers	B–9
§B.3.4. Matrix Product Properties	B–10
§B.4. Bilinear and Quadratic Forms	B–11
§B.5. Matrix Orthogonality	B–11
§B.5.1. Matrix Orthogonalization Via Projectors	B–11
§B.5.2. Orthogonal Projector Properties	B–12
§B. Exercises	B–14
§B. Solutions to Exercises	B–15

§B.1. Matrices

§B.1.1. Concept

Let us now introduce the concept of a *matrix*. Consider a set of scalar quantities arranged in a rectangular array containing m rows and n columns:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1j} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2j} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \cdots & a_{ij} & \cdots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mj} & \cdots & a_{mn} \end{bmatrix}. \quad (\text{B.1})$$

This array will be called a *rectangular matrix* of order m by n , or, briefly, an $m \times n$ matrix. Not every rectangular array is a matrix; to qualify as such it must obey the operational rules discussed below.

The quantities a_{ij} are called the *entries* or *components* of the matrix. Preference will be given to the latter unless one is talking about the computer implementation. As in the case of vectors, the term “matrix element” will be avoided to lessen the chance of confusion with finite elements. The two subscripts identify the row and column, respectively.

Matrices are conventionally identified by **bold uppercase** letters such as **A**, **B**, etc. The entries of matrix **A** may be denoted as A_{ij} or a_{ij} , according to the intended use. Occasionally we shall use the short-hand component notation

$$\mathbf{A} = [a_{ij}]. \quad (\text{B.2})$$

Example B.1. The following is a 2×3 numerical matrix:

$$\mathbf{B} = \begin{bmatrix} 2 & 6 & 3 \\ 4 & 9 & 1 \end{bmatrix} \quad (\text{B.3})$$

This matrix has 2 rows and 3 columns. The first row is (2, 6, 3), the second row is (4, 9, 1), the first column is (2, 4), and so on.

In some contexts it is convenient or useful to display the number of rows and columns. If this is so we will write them underneath the matrix symbol.¹ For the example matrix (B.3) we would show

$$\begin{matrix} \mathbf{B} \\ 2 \times 3 \end{matrix} \quad (\text{B.4})$$

Remark B.1. Matrices should not be confused with determinants.² A determinant is a number associated with square matrices ($m = n$), defined according to the rules stated in Appendix C.

¹ A convention introduced in Berkeley courses by Ray Clough. It is particularly useful in blackboard expositions.

² This confusion is apparent in the literature of the period 1860–1920.

§B.1.2. Real and Complex Matrices

As in the case of vectors, the components of a matrix may be real or complex. If they are real numbers, the matrix is called *real*, and *complex* otherwise. For the present exposition all matrices will be real.

§B.1.3. Square Matrices

The case $m = n$ is important in practical applications. Such matrices are called *square matrices* of order n . Matrices for which $m \neq n$ are called non-square (the term “rectangular” is also used in this context, but this is fuzzy because squares are special cases of rectangles).

Square matrices enjoy certain properties not shared by non-square matrices, such as the symmetry and antisymmetry conditions defined below. Furthermore many operations, such as taking determinants and computing eigenvalues, are only defined for square matrices.

Example B.2.

$$\mathbf{C} = \begin{bmatrix} 12 & 6 & 3 \\ 8 & 24 & 7 \\ 2 & 5 & 11 \end{bmatrix} \quad (\text{B.5})$$

is a square matrix of order 3.

Consider a square matrix $\mathbf{A} = [a_{ij}]$ of order $n \times n$. Its n components a_{ii} form the *main diagonal*, which runs from top left to bottom right. The *cross diagonal* runs from the bottom left to upper right. The main diagonal of the example matrix (B.5) is $\{12, 24, 11\}$ and the cross diagonal is $\{2, 24, 3\}$.

Entries that run parallel to and above (below) the main diagonal form superdiagonals (subdiagonals). For example, $\{6, 7\}$ is the first superdiagonal of the example matrix (B.5).

§B.1.4. Symmetry and Antisymmetry

Square matrices for which $a_{ij} = a_{ji}$ are called *symmetric about the main diagonal* or simply *symmetric*.

Square matrices for which $a_{ij} = -a_{ji}$ are called *antisymmetric* or *skew-symmetric*. The diagonal entries of an antisymmetric matrix must be zero.

Example B.3. The following is a symmetric matrix of order 3:

$$\mathbf{S} = \begin{bmatrix} 11 & 6 & 1 \\ 6 & 3 & -1 \\ 1 & -1 & -6 \end{bmatrix}. \quad (\text{B.6})$$

The following is an antisymmetric matrix of order 4:

$$\mathbf{W} = \begin{bmatrix} 0 & 3 & -1 & -5 \\ -3 & 0 & 7 & -2 \\ 1 & -7 & 0 & 0 \\ 5 & 2 & 0 & 0 \end{bmatrix}. \quad (\text{B.7})$$

§B.1.5. Are Vectors a Special Case of Matrices?

Consider the 3-vector \mathbf{x} and a 3×1 matrix \mathbf{X} with the same components:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{X} = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix}. \quad (\text{B.8})$$

in which $x_1 = x_{11}$, $x_2 = x_{21}$ and $x_3 = x_{31}$. Are \mathbf{x} and \mathbf{X} the same thing? If so we could treat column vectors as one-column matrices and dispense with the distinction.

Indeed in many contexts a column vector of order n may be treated as a matrix with a single column, i.e., as a matrix of order $n \times 1$. Similarly, a row vector of order m may be treated as a matrix with a single row, i.e., as a matrix of order $1 \times m$.

There are some operations, however, for which the analogy does not carry over, and one has to consider vectors as different from matrices. The dichotomy is reflected in the notational conventions of lower versus upper case. Another important distinction from a practical standpoint is discussed next.

§B.1.6. Where Do Matrices Come From?

Although we speak of “matrix algebra” as embodying vectors as special cases of matrices, in practice the quantities of primary interest to the structural engineer are vectors rather than matrices. For example, an engineer may be interested in displacement vectors, force vectors, vibration eigenvectors, buckling eigenvectors. In finite element analysis even stresses and strains are often arranged as vectors although they are really tensors.

On the other hand, matrices are rarely the quantities of primary interest: they work silently in the background where they are normally engaged in operating on vectors.

§B.1.7. Special Matrices

The *null* matrix, written $\mathbf{0}$, is the matrix all of whose components are zero.

Example B.4. The null matrix of order 2×3 is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (\text{B.9})$$

The *identity matrix*, written \mathbf{I} , is a square matrix all of which entries are zero except those on the main diagonal, which are ones.

Example B.5. The identity matrix of order 4 is

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (\text{B.10})$$

A *diagonal matrix* is a square matrix all of which entries are zero except for those on the main diagonal, which may be arbitrary.

Example B.6. The following matrix of order 4 is diagonal:

$$\mathbf{D} = \begin{bmatrix} 14 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}. \quad (\text{B.11})$$

A short hand notation which lists only the diagonal entries is sometimes used for diagonal matrices to save writing space. This notation is illustrated for the above matrix:

$$\mathbf{D} = \mathbf{diag} [14 \quad -6 \quad 0 \quad 3]. \quad (\text{B.12})$$

An *upper triangular* matrix is a square matrix in which all elements underneath the main diagonal vanish. A *lower triangular* matrix is a square matrix in which all entries above the main diagonal vanish.

Example B.7. Here are examples of each kind:

$$\mathbf{U} = \begin{bmatrix} 6 & 4 & 2 & 1 \\ 0 & 6 & 4 & 2 \\ 0 & 0 & 6 & 4 \\ 0 & 0 & 0 & 6 \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 10 & 4 & 0 & 0 \\ -3 & 21 & 6 & 0 \\ -15 & -2 & 18 & 7 \end{bmatrix}. \quad (\text{B.13})$$

§B.2. Elementary Matrix Operations

§B.2.1. Equality

Two matrices \mathbf{A} and \mathbf{B} of same order $m \times n$ are said to be *equal* if and only if all of their components are equal: $a_{ij} = b_{ij}$, for all $i = 1, \dots, m$, $j = 1, \dots, n$. We then write $\mathbf{A} = \mathbf{B}$. If the inequality test fails the matrices are said to be *unequal* and we write $\mathbf{A} \neq \mathbf{B}$.

Two matrices of different order cannot be compared for equality or inequality.

There is no simple test for greater-than or less-than.

§B.2.2. Transposition

The *transpose* of a matrix \mathbf{A} is another matrix denoted by \mathbf{A}^T that has n rows and m columns

$$\mathbf{A}^T = [a_{ji}]. \quad (\text{B.14})$$

The rows of \mathbf{A}^T are the columns of \mathbf{A} , and the rows of \mathbf{A} are the columns of \mathbf{A}^T .

Obviously the transpose of \mathbf{A}^T is again \mathbf{A} , that is, $(\mathbf{A}^T)^T = \mathbf{A}$.

Example B.8.

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 0 \\ 1 & 0 & 4 \end{bmatrix}, \quad \mathbf{A}^T = \begin{bmatrix} 5 & 1 \\ 7 & 0 \\ 0 & 4 \end{bmatrix}. \quad (\text{B.15})$$

The transpose of a square matrix is also a square matrix. The transpose of a symmetric matrix \mathbf{A} is equal to the original matrix, *i.e.*, $\mathbf{A} = \mathbf{A}^T$. The negated transpose of an antisymmetric matrix \mathbf{A} is equal to the original matrix, *i.e.* $\mathbf{A} = -\mathbf{A}^T$.

Example B.9.

$$\mathbf{A} = \begin{bmatrix} 4 & 7 & 0 \\ 7 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix} = \mathbf{A}^T, \quad \mathbf{W} = \begin{bmatrix} 0 & 7 & 0 \\ -7 & 0 & -2 \\ 0 & 2 & 0 \end{bmatrix} = -\mathbf{W}^T \quad (\text{B.16})$$

§B.2.3. Addition and Subtraction

The simplest operation acting on two matrices is *addition*. The sum of two matrices of the same order, \mathbf{A} and \mathbf{B} , is written $\mathbf{A} + \mathbf{B}$ and defined to be the matrix

$$\mathbf{A} + \mathbf{B} \stackrel{\text{def}}{=} [a_{ij} + b_{ij}]. \quad (\text{B.17})$$

Like vector addition, matrix addition is commutative: $\mathbf{A} + \mathbf{B} = \mathbf{B} + \mathbf{A}$, and associative: $\mathbf{A} + (\mathbf{B} + \mathbf{C}) = (\mathbf{A} + \mathbf{B}) + \mathbf{C}$. For $n = 1$ or $m = 1$ the operation reduces to the addition of two column or row vectors, respectively.

For matrix subtraction, replace $+$ by $-$ in the definition (?).

Example B.10. The sum of

$$\mathbf{A} = \begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 6 & 3 & -3 \\ 7 & -2 & 5 \end{bmatrix} \quad \text{is} \quad \mathbf{A} + \mathbf{B} = \begin{bmatrix} 7 & 0 & -3 \\ 11 & 0 & 4 \end{bmatrix}. \quad (\text{B.18})$$

§B.2.4. Scalar Multiplication

Multiplication of a matrix \mathbf{A} by a scalar c is defined by means of the relation

$$c \mathbf{A} \stackrel{\text{def}}{=} [ca_{ij}] \quad (\text{B.19})$$

That is, each entry of the matrix is multiplied by c . This operation is often called *scaling* of a matrix. If $c = 0$, the result is the null matrix. Division of a matrix by a nonzero scalar c is equivalent to multiplication by $(1/c)$.

Example B.11.

$$\text{If } \mathbf{A} = \begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix}, \quad 3\mathbf{A} = \begin{bmatrix} 3 & -9 & 0 \\ 12 & 6 & -3 \end{bmatrix}. \quad (\text{B.20})$$

§B.3. Matrix Products

§B.3.1. Matrix by Vector Product

Before describing the general matrix product of two matrices, let us treat the particular case in which the second matrix is a column vector. This so-called *matrix-vector product* merits special attention because it occurs very frequently in the applications. Let $\mathbf{A} = [a_{ij}]$ be an $m \times n$ matrix, $\mathbf{x} = \{x_j\}$ a column vector of order n , and $\mathbf{y} = \{y_i\}$ a column vector of order m . The matrix-vector product is symbolically written

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \quad (\text{B.21})$$

to mean the linear transformation

$$y_i \stackrel{\text{def}}{=} \sum_{j=1}^n a_{ij}x_j \stackrel{\text{sc}}{=} a_{ij}x_j, \quad i = 1, \dots, m. \quad (\text{B.22})$$

Example B.12. The product of a 2×3 matrix and a vector of order 3 is a vector of order 2:

$$\begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \end{bmatrix} \quad (\text{B.23})$$

This product definition is not arbitrary but emanates from the analytical and geometric properties of entities represented by matrices and vectors.

For the product definition to make sense, the column dimension of the matrix \mathbf{A} (called the pre-multiplicand) must equal the dimension of the vector \mathbf{x} (called the post-multiplicand). For example, the reverse product $\mathbf{x}\mathbf{A}$ does not make sense unless $m = n = 1$.

If the row dimension m of \mathbf{A} is one, the matrix formally reduces to a row vector, and the matrix-vector product reduces to the inner product defined by equation (A.15) of Appendix A. The result of this operation is a one-dimensional vector or scalar. We thus see that the present definition properly embodies previous cases.

The associative and commutative properties of the matrix-vector product fall under the rules of the more general matrix-matrix product discussed next.

§B.3.2. Matrix by Matrix Product

We now pass to the most general matrix-by-matrix product, and consider the operations involved in computing the product \mathbf{C} of two matrices \mathbf{A} and \mathbf{B} :

$$\mathbf{C} = \mathbf{A}\mathbf{B}. \quad (\text{B.24})$$

Here $\mathbf{A} = [a_{ij}]$ is a matrix of order $m \times n$, $\mathbf{B} = [b_{jk}]$ is a matrix of order $n \times p$, and $\mathbf{C} = [c_{ik}]$ is a matrix of order $m \times p$. The entries of the result matrix \mathbf{C} are defined by the formula

$$c_{ik} \stackrel{\text{def}}{=} \sum_{j=1}^n a_{ij}b_{jk} \stackrel{\text{sc}}{=} a_{ij}b_{jk}, \quad i = 1, \dots, m, \quad k = 1, \dots, p. \quad (\text{B.25})$$

We see that the $(i, k)^{th}$ entry of \mathbf{C} is computed by taking the *inner product* of the i^{th} row of \mathbf{A} with the k^{th} column of \mathbf{B} . For this definition to work and the product be possible, *the column dimension of \mathbf{A} must be the same as the row dimension of \mathbf{B}* . Matrices that satisfy this rule are said to be *product-conforming*, or *conforming* for short. If the two matrices do not conform, their product is undefined. The following mnemonic notation often helps in remembering this rule:

$$\underset{m \times p}{\mathbf{C}} = \underset{m \times n}{\mathbf{A}} \underset{n \times p}{\mathbf{B}} \quad (\text{B.26})$$

For the matrix-by-vector case treated in the preceding subsection, $p = 1$.

Matrix \mathbf{A} is called the pre-multiplicand and is said to *premultiply* \mathbf{B} . Matrix \mathbf{B} is called the post-multiplicand and is said to *postmultiply* \mathbf{A} . This careful distinction on which matrix comes first is a consequence of the absence of commutativity: even if \mathbf{BA} exists (it only does if $m = n$), it is not generally the same as \mathbf{AB} .

For *hand* computations, the matrix product is most conveniently organized by the so-called Falk's scheme:

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{i1} & \cdots & a_{in} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \rightarrow \begin{bmatrix} b_{11} & \cdots & b_{ik} & \cdots & b_{1p} \\ \vdots & \ddots & \downarrow & \ddots & \vdots \\ b_{n1} & \cdots & b_{nk} & \cdots & b_{np} \\ \vdots & & \vdots & & \vdots \\ \cdots & & c_{ik} & & \end{bmatrix}. \quad (\text{B.27})$$

Each entry in row i of \mathbf{A} is multiplied by the corresponding entry in column k of \mathbf{B} (note the arrows), and the products are summed and stored in the $(i, k)^{th}$ entry of \mathbf{C} .

Example B.13. To illustrate Falk's scheme, let us form the product $\mathbf{C} = \mathbf{AB}$ of the following matrices

$$\mathbf{A} = \begin{bmatrix} 3 & 0 & 2 \\ 4 & -1 & 5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & 1 & 0 & -5 \\ 4 & 3 & -1 & 0 \\ 0 & 1 & -7 & 4 \end{bmatrix} \quad (\text{B.28})$$

The matrices are conforming because the column dimension of \mathbf{A} and the row dimension of \mathbf{B} are the same (3). We arrange the computations as shown below:

$$\mathbf{A} = \begin{bmatrix} 3 & 0 & 2 \\ 4 & -1 & 5 \end{bmatrix} \begin{bmatrix} 2 & 1 & 0 & -5 \\ 4 & 3 & -1 & 0 \\ 0 & 1 & -7 & 4 \end{bmatrix} = \mathbf{B} \quad (\text{B.29})$$

$$\begin{bmatrix} 6 & 5 & -14 & -7 \\ 4 & 6 & -34 & 0 \end{bmatrix} = \mathbf{C} = \mathbf{AB}$$

Here $3 \times 2 + 0 \times 4 + 2 \times 0 = 6$ and so on.

§B.3.3. Matrix Powers

If $\mathbf{A} = \mathbf{B}$, the product $\mathbf{A}\mathbf{A}$ is called the *square* of \mathbf{A} and is denoted by \mathbf{A}^2 . Note that for this definition to make sense, \mathbf{A} must be a square matrix; else the factors would not be conforming.

Similarly, $\mathbf{A}^3 = \mathbf{A}\mathbf{A}\mathbf{A} = \mathbf{A}^2\mathbf{A} = \mathbf{A}\mathbf{A}^2$. Other positive-integer powers can be defined in an analogous manner.

This definition does not encompass negative powers. For example, \mathbf{A}^{-1} denotes the *inverse* of matrix \mathbf{A} , which is studied in Appendix C. The general power \mathbf{A}^m , where m can be a real or complex scalar, can be defined with the help of the matrix spectral form and requires the notion of eigensystem covered in Appendix D.

A square matrix \mathbf{A} that satisfies $\mathbf{A} = \mathbf{A}^2$ is called *idempotent*. We shall see later that this condition characterizes the so-called projector matrices.

A square matrix \mathbf{A} whose p^{th} power is the null matrix is called *p-nilpotent*.

§B.3.4. Matrix Product Properties

Associativity. The associative law is verified:

$$\mathbf{A}(\mathbf{B}\mathbf{C}) = (\mathbf{A}\mathbf{B})\mathbf{C}. \quad (\text{B.30})$$

Hence we may delete the parentheses and simply write $\mathbf{A}\mathbf{B}\mathbf{C}$.

Distributivity. The distributive law also holds: If \mathbf{B} and \mathbf{C} are matrices of the same order, then

$$\mathbf{A}(\mathbf{B} + \mathbf{C}) = \mathbf{A}\mathbf{B} + \mathbf{A}\mathbf{C}, \quad \text{and} \quad (\mathbf{B} + \mathbf{C})\mathbf{A} = \mathbf{B}\mathbf{A} + \mathbf{C}\mathbf{A}. \quad (\text{B.31})$$

Commutativity. The commutativity law of scalar multiplication does not generally hold. If \mathbf{A} and \mathbf{B} are square matrices of the same order, then the products $\mathbf{A}\mathbf{B}$ and $\mathbf{B}\mathbf{A}$ are both possible but in general $\mathbf{A}\mathbf{B} \neq \mathbf{B}\mathbf{A}$.

If $\mathbf{A}\mathbf{B} = \mathbf{B}\mathbf{A}$, the matrices \mathbf{A} and \mathbf{B} are said to *commute*. One important case is when \mathbf{A} and \mathbf{B} are diagonal. In general \mathbf{A} and \mathbf{B} commute if they share the same eigensystem.

Example B.14. Matrices

$$\mathbf{A} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} a - \beta & b \\ b & c - \beta \end{bmatrix}, \quad (\text{B.32})$$

commute for any a, b, c, β . More generally, \mathbf{A} and $\mathbf{B} = \mathbf{A} - \beta\mathbf{I}$ commute for any square matrix \mathbf{A} .

Transpose of a Product. The transpose of a matrix product is equal to the product of the transposes of the operands taken in reverse order:

$$(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T \mathbf{A}^T. \quad (\text{B.33})$$

The general transposition formula for an arbitrary product sequence is

$$(\mathbf{A}\mathbf{B}\mathbf{C} \dots \mathbf{M}\mathbf{N})^T = \mathbf{N}^T \mathbf{M}^T \dots \mathbf{C}^T \mathbf{B}^T \mathbf{A}^T. \quad (\text{B.34})$$

Congruential Transformation. If \mathbf{B} is a *symmetric* matrix of order m and \mathbf{A} is an arbitrary $m \times n$ matrix, then

$$\mathbf{S} = \mathbf{A}^T \mathbf{B} \mathbf{A}. \quad (\text{B.35})$$

is a symmetric matrix of order n . Such an operation is called a congruential transformation. It occurs very frequently in finite element analysis when changing coordinate bases because such a transformation preserves energy.

Loss of Symmetry. The product of two symmetric matrices is not generally symmetric.

Null Matrices may have Non-null Divisors. The matrix product \mathbf{AB} can be zero although $\mathbf{A} \neq \mathbf{0}$ and $\mathbf{B} \neq \mathbf{0}$. Likewise it is possible that $\mathbf{A} \neq \mathbf{0}$, $\mathbf{A}^2 \neq \mathbf{0}$, \dots , but $\mathbf{A}^p = \mathbf{0}$.

§B.4. Bilinear and Quadratic Forms

Let \mathbf{x} and \mathbf{y} be two column vectors of order n , and \mathbf{A} a real square $n \times n$ matrix. Then the following triple product produces a *scalar* result:

$$s = \underset{1 \times n}{\mathbf{y}^T} \underset{n \times n}{\mathbf{A}} \underset{n \times 1}{\mathbf{x}} \quad (\text{B.36})$$

This is called a *bilinear form*. Matrix \mathbf{A} is called the *kernel* of the form.

Transposing both sides of (B.36) and noting that the transpose of a scalar does not change, we obtain the result

$$s = \mathbf{x}^T \mathbf{A}^T \mathbf{y} = \mathbf{y}^T \mathbf{A} \mathbf{x}. \quad (\text{B.37})$$

If \mathbf{A} is symmetric and vectors \mathbf{x} and \mathbf{y} coalesce, *i.e.*

$$\mathbf{A}^T = \mathbf{A}, \quad \mathbf{x} = \mathbf{y}, \quad (\text{B.38})$$

the bilinear form becomes a *quadratic form*

$$s = \mathbf{x}^T \mathbf{A} \mathbf{x}. \quad (\text{B.39})$$

Transposing both sides of a quadratic form reproduces the same equation.

Example B.15. The kinetic energy of a dynamic system consisting of three point masses m_1, m_2, m_3 moving in one dimension with velocities v_1, v_2 and v_3 , respectively, is

$$T = \frac{1}{2}(m_1 v_1^2 + m_2 v_2^2 + m_3 v_3^2). \quad (\text{B.40})$$

This can be expressed as the quadratic form

$$T = \frac{1}{2} \mathbf{v}^T \mathbf{M} \mathbf{v}, \quad (\text{B.41})$$

in which

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}, \quad \mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}. \quad (\text{B.42})$$

Here \mathbf{M} denotes the system mass matrix whereas \mathbf{v} is the system velocity vector.

§B.5. Matrix Orthogonality

Let \mathbf{A} and \mathbf{B} be two product-conforming real matrices. For example, \mathbf{A} is $k \times m$ whereas \mathbf{B} is $m \times n$. If their product is the null matrix

$$\mathbf{C} = \mathbf{A} \mathbf{B} = \mathbf{0}, \quad (\text{B.43})$$

the matrices are said to be *orthogonal*. This is the generalization of the notions of vector orthogonality discussed in the previous Appendix.

§B.5.1. Matrix Orthogonalization Via Projectors

The *matrix orthogonalization* problem can be stated as follows. Product conforming matrices \mathbf{A} and \mathbf{B} are given but their product is not zero. How can \mathbf{A} be orthogonalized with respect to \mathbf{B} so that (B.43) is verified? Suppose that \mathbf{B} is $m \times n$ with $m \geq n$ and that $\mathbf{B}^T \mathbf{B}$ is nonsingular (equivalently, \mathbf{B} has full rank).³ Then form the $m \times m$ *orthogonal projector matrix*, or simply *projector*

$$\mathbf{P}_B = \mathbf{I} - \mathbf{B} (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T. \quad (\text{B.44})$$

in which \mathbf{I} is the $m \times m$ identity matrix. Since $\mathbf{P}_B = \mathbf{P}_B^T$, the projector is square symmetric.

Note that

$$\mathbf{P}_B \mathbf{B} = \mathbf{B} - \mathbf{B} (\mathbf{B}^T \mathbf{B})^{-1} (\mathbf{B}^T \mathbf{B}) = \mathbf{B} - \mathbf{B} = \mathbf{0}. \quad (\text{B.45})$$

It follows that \mathbf{P}_B projects \mathbf{B} onto its null space. Likewise $\mathbf{B}^T \mathbf{P}_B = \mathbf{0}$. Postmultiplying \mathbf{A} by \mathbf{P}_B yields

$$\tilde{\mathbf{A}} = \mathbf{A} \mathbf{P}_B = \mathbf{A} - \mathbf{A} \mathbf{B} (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T. \quad (\text{B.46})$$

Matrix $\tilde{\mathbf{A}}$ is called the *projection* of \mathbf{A} onto the null space of \mathbf{B} .⁴ It is easily verified that $\tilde{\mathbf{A}}$ and \mathbf{B} are orthogonal:

$$\tilde{\mathbf{A}} \mathbf{B} = \mathbf{A} \mathbf{B} - \mathbf{A} \mathbf{B} (\mathbf{B}^T \mathbf{B})^{-1} (\mathbf{B}^T \mathbf{B}) = \mathbf{A} \mathbf{B} - \mathbf{A} \mathbf{B} = \mathbf{0}. \quad (\text{B.47})$$

Consequently, forming $\tilde{\mathbf{A}}$ via (B.44) and (B.46) solves the orthogonalization problem.

If \mathbf{B} is square and nonsingular, $\tilde{\mathbf{A}} = \mathbf{0}$, as may be expected. If \mathbf{B} has more columns than rows, that is $m < n$, the projector (B.44) cannot be constructed since $\mathbf{B} \mathbf{B}^T$ is necessarily singular. A similar difficulty arises if $m \geq n$ but $\mathbf{B}^T \mathbf{B}$ is singular. Such cases require treatment using generalized inverses, which is a topic beyond the scope of this Appendix.⁵

In some applications, notably FEM, matrix \mathbf{A} is square symmetric and it is desirable to preserve symmetry in $\tilde{\mathbf{A}}$. That can be done by pre- and postmultiplying by the projector:

$$\tilde{\mathbf{A}} = \mathbf{P}_B \mathbf{A} \mathbf{P}_B. \quad (\text{B.48})$$

Since $\mathbf{P}_B^T = \mathbf{P}_B$, the operation (B.48) is a congruential transformation, which preserves symmetry.

³ If you are not sure what “singular”, “nonsingular” and “rank” mean or what $(\cdot)^{-1}$ stands for, please read §D.4.

⁴ In contexts such as control and signal processing, \mathbf{P}_B is called a *filter* and the operation (B.46) is called *filtering*.

⁵ See e.g., the textbooks [77,596].

§B.5.2. Orthogonal Projector Properties

The following properties of the projector (B.44) are useful when checking out computations. Forming its square as

$$\begin{aligned}\mathbf{P}_B^2 &= \mathbf{P}_B \mathbf{P}_B = \mathbf{I} - 2\mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T + \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \\ &= \mathbf{I} - 2\mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T + \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T = \mathbf{I} - \mathbf{B}(\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T = \mathbf{P}_B,\end{aligned}\tag{B.49}$$

shows that the projector matrix is idempotent. Repeating the process one sees that $\mathbf{P}_B^n = \mathbf{P}_B$, in which n is an arbitrary nonnegative integer.

If \mathbf{B} is $m \times n$ with $m \geq n$ and full rank n , \mathbf{P}_B has $m - n$ unit eigenvalues and n zero eigenvalues. This is shown in the paper [235], in which various applications of orthogonal projectors and orthogonalization to multilevel FEM computations are covered in detail.

Homework Exercises for Appendix B: Matrices**EXERCISE B.1** Given the three matrices

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 1 & 0 \\ -1 & 2 & 3 & 1 \\ 2 & 5 & -1 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & -2 \\ 1 & 0 \\ 4 & 1 \\ -3 & 2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & -3 & 2 \\ 2 & 0 & 2 \end{bmatrix} \quad (\text{EB.1})$$

compute the product $\mathbf{D} = \mathbf{ABC}$ by hand using Falk's scheme. *Hint:* do \mathbf{BC} first, then premultiply that by \mathbf{A} .**EXERCISE B.2** Given the square matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ -4 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & 0 \\ 1 & -2 \end{bmatrix} \quad (\text{EB.2})$$

verify by direct computation that $\mathbf{AB} \neq \mathbf{BA}$.**EXERCISE B.3** Given the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \\ 2 & 0 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & -1 & 4 \\ -1 & 2 & 0 \\ 4 & 0 & 0 \end{bmatrix} \quad (\text{EB.3})$$

(note that \mathbf{B} is symmetric) compute $\mathbf{S} = \mathbf{A}^T \mathbf{B} \mathbf{A}$, and verify that \mathbf{S} is symmetric.**EXERCISE B.4** Given the square matrices

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 2 \\ 1 & 0 & 3 \\ 3 & -2 & -5 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 3 & -6 & -3 \\ 7 & -14 & -7 \\ -1 & 2 & 1 \end{bmatrix} \quad (\text{EB.4})$$

verify that $\mathbf{AB} = \mathbf{0}$ although $\mathbf{A} \neq \mathbf{0}$ and $\mathbf{B} \neq \mathbf{0}$. Is \mathbf{BA} also null?**EXERCISE B.5** Given the square matrix

$$\mathbf{A} = \begin{bmatrix} 0 & a & b \\ 0 & 0 & c \\ 0 & 0 & 0 \end{bmatrix} \quad (\text{EB.5})$$

show by direct computation that $\mathbf{A}^2 \neq \mathbf{0}$ but $\mathbf{A}^3 = \mathbf{0}$.**EXERCISE B.6** Can a diagonal matrix be antisymmetric?**EXERCISE B.7** (Tougher) Prove the matrix product transposition rule (B.33). *Hint:* call $\mathbf{C} = (\mathbf{AB})^T$, $\mathbf{D} = \mathbf{B}^T \mathbf{A}^T$, and use the matrix product definition (B.25) to show that the generic entries of \mathbf{C} and \mathbf{D} agree.**EXERCISE B.8** If \mathbf{A} is an arbitrary $m \times n$ matrix, show: (a) both products $\mathbf{A}^T \mathbf{A}$ and $\mathbf{A} \mathbf{A}^T$ are possible, and (b) both products are square and symmetric. *Hint:* for (b) use the symmetry condition $\mathbf{S} = \mathbf{S}^T$ and (B.31).**EXERCISE B.9** Show that \mathbf{A}^2 only exists if and only if \mathbf{A} is square.**EXERCISE B.10** If \mathbf{A} is square and antisymmetric, show that \mathbf{A}^2 is symmetric. *Hint:* start from $\mathbf{A} = -\mathbf{A}^T$ and apply the results of Exercise B.8.

Homework Exercises for Appendix B - Solutions

EXERCISE B.1

$$\begin{aligned}
 \mathbf{B} &= \begin{bmatrix} 2 & -2 \\ 1 & 0 \\ 4 & 1 \\ -3 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & -3 & 2 \\ 2 & 0 & 2 \end{bmatrix} = \mathbf{C} \\
 & \quad \begin{bmatrix} -2 & -6 & 0 \\ 1 & -3 & 2 \\ 6 & -12 & 10 \\ 1 & 9 & -2 \end{bmatrix} = \mathbf{BC} \\
 \mathbf{A} &= \begin{bmatrix} 2 & 4 & 1 & 0 \\ -1 & 2 & 3 & 1 \\ 2 & 5 & -1 & 2 \end{bmatrix} \quad \begin{bmatrix} 6 & -36 & 18 \\ 23 & -27 & 32 \\ -3 & 3 & -4 \end{bmatrix} = \mathbf{ABC} = \mathbf{D}
 \end{aligned} \tag{EB.6}$$

EXERCISE B.2

$$\mathbf{AB} = \begin{bmatrix} 6 & -6 \\ -10 & -4 \end{bmatrix} \neq \mathbf{BA} = \begin{bmatrix} 3 & 9 \\ 9 & -1 \end{bmatrix} \tag{EB.7}$$

EXERCISE B.3

$$\mathbf{S} = \mathbf{A}^T \mathbf{BA} = \begin{bmatrix} 23 & -6 \\ -6 & 8 \end{bmatrix}, \tag{EB.8}$$

which is symmetric, like \mathbf{B} .

EXERCISE B.4

$$\begin{aligned}
 \mathbf{A} &= \begin{bmatrix} 3 & -1 & 2 \\ 1 & 0 & 3 \\ 3 & -2 & -5 \end{bmatrix} \quad \begin{bmatrix} 3 & -6 & -3 \\ 7 & -14 & -7 \\ -1 & 2 & 1 \end{bmatrix} = \mathbf{B} \\
 & \quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{AB} = \mathbf{0}
 \end{aligned} \tag{EB.9}$$

However,

$$\mathbf{BA} = \begin{bmatrix} -6 & 3 & 3 \\ -14 & 7 & 7 \\ 2 & -1 & -1 \end{bmatrix} \neq \mathbf{0}. \tag{EB.10}$$

EXERCISE B.5

$$\mathbf{A}^2 = \mathbf{AA} = \begin{bmatrix} 0 & 0 & ac \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}^3 = \mathbf{AAA} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{0} \tag{EB.11}$$

EXERCISE B.6 Only if it is the null matrix.

EXERCISE B.7 To avoid “indexing indigestion” let us carefully specify the dimensions of the given matrices and their transposes:

$$\mathbf{A}_{m \times n} = [a_{ij}], \quad \mathbf{A}_{n \times m}^T = [a_{ji}]. \tag{EB.12}$$

$$\mathbf{B}_{n \times p} = [b_{jk}], \quad \mathbf{B}_{p \times n}^T = [b_{kj}] \tag{EB.13}$$

Indices i, j and k run over $1 \dots m, 1 \dots n$ and $1 \dots p$, respectively. Now call

$$\mathbf{C}_{p \times m} = [c_{ki}] = (\mathbf{AB})^T. \tag{EB.14}$$

$$\mathbf{D}_{p \times m} = [d_{ki}] = \mathbf{B}^T \mathbf{A}^T. \quad (\text{EB.15})$$

From the definition of matrix product,

$$c_{ki} = \sum_{j=1}^n a_{ij} b_{jk}. \quad (\text{EB.16})$$

$$d_{ki} = \sum_{j=1}^n b_{jk} a_{ij} = \sum_{j=1}^n a_{ij} b_{jk} = c_{ki}. \quad (\text{EB.17})$$

Hence $\mathbf{C} = \mathbf{D}$ for any \mathbf{A} and \mathbf{B} , and the statement is proved.

EXERCISE B.8

(a) If \mathbf{A} is $m \times n$, \mathbf{A}^T is $n \times m$. Next we write the two products to be investigated:

$$\mathbf{A}^T_{n \times m} \mathbf{A}_{m \times n}, \quad \mathbf{A}_{m \times n} \mathbf{A}^T_{n \times m} \quad (\text{EB.18})$$

In both cases the column dimension of the premultiplicand is equal to the row dimension of the postmultiplicand. Therefore both products are possible.

(b) To verify symmetry we use three results. First, the symmetry test: transpose equals original; second, transposing twice gives back the original; and, finally, the transposed-product formula proved in Exercise B.7.

$$(\mathbf{A}^T \mathbf{A})^T = \mathbf{A}^T (\mathbf{A}^T)^T = \mathbf{A}^T \mathbf{A}. \quad (\text{EB.19})$$

$$(\mathbf{A} \mathbf{A}^T)^T = (\mathbf{A}^T)^T \mathbf{A}^T = \mathbf{A} \mathbf{A}^T. \quad (\text{EB.20})$$

Or, to do it more leisurely, call $\mathbf{B} = \mathbf{A}^T$, $\mathbf{B}^T = \mathbf{A}$, $\mathbf{C} = \mathbf{A} \mathbf{B}$, and let's go over the first one again:

$$\mathbf{C}^T = (\mathbf{A} \mathbf{B})^T = \mathbf{B}^T \mathbf{A}^T = \mathbf{A} \mathbf{A}^T = \mathbf{A} \mathbf{B} = \mathbf{C}. \quad (\text{EB.21})$$

Since $\mathbf{C} = \mathbf{C}^T$, $\mathbf{C} = \mathbf{A} \mathbf{A}^T$ is symmetric. Same mechanics for the second one.

EXERCISE B.9 Let \mathbf{A} be $m \times n$. For $\mathbf{A}^2 = \mathbf{A} \mathbf{A}$ to exist, the column dimension n of the premultiplicand \mathbf{A} must equal the row dimension m of the postmultiplicand \mathbf{A} . Hence $m = n$ and \mathbf{A} must be square.

EXERCISE B.10 Premultiply both sides of $\mathbf{A} = -\mathbf{A}^T$ by \mathbf{A} (which is always possible because \mathbf{A} is square):

$$\mathbf{A}^2 = \mathbf{A} \mathbf{A} = -\mathbf{A} \mathbf{A}^T. \quad (\text{EB.22})$$

But from Exercise B.8 we know that $\mathbf{A} \mathbf{A}^T$ is symmetric. Since the negated of a symmetric matrix is symmetric, so is \mathbf{A}^2 .

C

Continuum Mechanics Summary

TABLE OF CONTENTS

	Page
§C.1. Introduction	C–3
§C.2. The Strain-Displacement Equations	C–3
§C.2.1. The Linear Strain Tensor 	C–3
§C.2.2. The Engineering Notation 	C–4
§C.3. Compatibility Equations	C–4
§C.4. The Stress Vector	C–4
§C.5. The Stress Tensor	C–5
§C.6. Equilibrium Equations	C–6
§C.7. Constitutive Equations	C–6
§C.7.1. Elastic Solids 	C–7
§C.7.2. The Strain Energy 	C–7
§C.7.3. Principle of Virtual Work 	C–7
§C.7.4. Hyperelastic Constitutive Equations 	C–8

§C.1. Introduction

This Appendix summarizes the basic relations of three-dimensional continuum mechanics for linear elastic solids. These include strain-displacement, constitutive and equilibrium equations. Both indicial and full notations are used.

§C.2. The Strain-Displacement Equations

Let $u_i(x_j)$ denote the components of the displacement vector field $\mathbf{u}(x_j)$. Then the infinitesimal strains are given by

$$e_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \quad (\text{C.1})$$

These are the components of the strain tensor $[e_{ij}] = \underline{\mathbf{e}}$, which written in full is

$$\begin{aligned} e_{11} &= u_{1,1} = \frac{\partial u_1}{\partial x_1} \\ e_{22} &= u_{2,2} = \frac{\partial u_2}{\partial x_2} \\ e_{33} &= u_{3,3} = \frac{\partial u_3}{\partial x_3} \\ e_{12} &= \frac{1}{2}(u_{1,2} + u_{2,1}) = \frac{1}{2}\left(\frac{\partial u_1}{\partial x_2} + \frac{\partial u_2}{\partial x_1}\right) \\ e_{23} &= \frac{1}{2}(u_{2,3} + u_{3,2}) = \frac{1}{2}\left(\frac{\partial u_2}{\partial x_3} + \frac{\partial u_3}{\partial x_2}\right) \\ e_{31} &= \frac{1}{2}(u_{3,1} + u_{1,3}) = \frac{1}{2}\left(\frac{\partial u_3}{\partial x_1} + \frac{\partial u_1}{\partial x_3}\right) \end{aligned} \quad (\text{C.2})$$

§C.2.1. The Linear Strain Tensor

The infinitesimal or linear strain tensor in the x_i coordinate system is

$$\underline{\mathbf{e}} = [e_{ij}] = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ & e_{22} & e_{23} \\ \text{symm} & & e_{33} \end{bmatrix} \quad (\text{C.3})$$

In another coordinate system x'_j related to x_i by the transformation

$$x_i = a_{ij}x'_j \quad (\text{C.4})$$

the strain components become

$$e'_{ij} = a_{im}a_{jn}e_{mn} \quad (\text{C.5})$$

§C.2.2. The Engineering Notation

The standard engineering notation uses x, y, z for x_1, x_2, x_3 and u_x, u_y, u_z for u_1, u_2, u_3 , respectively. Then the engineering strains are related to the displacements by

$$\begin{aligned}
 e_{xx} &= e_{11} = \frac{\partial u_x}{\partial x} \\
 e_{yy} &= e_{22} = \frac{\partial u_y}{\partial y} \\
 e_{zz} &= e_{33} = \frac{\partial u_z}{\partial z} \\
 \gamma_{xy} &= 2e_{12} = \frac{\partial u_x}{\partial y} + \frac{\partial u_y}{\partial x} \\
 \gamma_{yz} &= 2e_{23} = \frac{\partial u_y}{\partial z} + \frac{\partial u_z}{\partial y} \\
 \gamma_{zx} &= 2e_{31} = \frac{\partial u_z}{\partial x} + \frac{\partial u_x}{\partial z}
 \end{aligned} \tag{C.6}$$

The linear strain tensor in terms of engineering strains is

$$[e] = \begin{bmatrix} e_{xx} & \frac{1}{2}\gamma_{xy} & \frac{1}{2}\gamma_{xz} \\ & e_{yy} & \frac{1}{2}\gamma_{yz} \\ \text{symm} & & e_{zz} \end{bmatrix} \tag{C.7}$$

§C.3. Compatibility Equations

The strain tensor \underline{e} has 6 independent components. The displacement field has 3 independent components. It follows that there must be 3 independent conditions between the e_{ij} . These expressions arise from the condition of compatibility of deformation. In the three-dimensional case these compatibility equations are

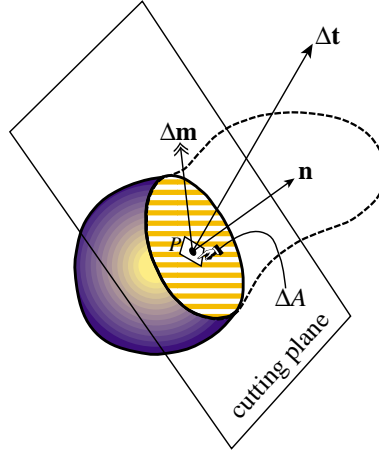
$$e_{ij,kl} + e_{kl,ij} = e_{ik,jl} + e_{jl,ik} \tag{C.8}$$

For the two dimensional case only one equation survives

$$e_{11,22} + e_{22,11} = 2e_{12,12} \tag{C.9}$$

which in standard notation is

$$\frac{\partial^2 e_{xx}}{\partial y^2} + \frac{\partial^2 e_{yy}}{\partial x^2} = \frac{\partial^2 \gamma_{xy}}{\partial_x \partial_y} \tag{C.10}$$

FIGURE C.1. Plane cut through a body for defining the interior force resultants at point P .

§C.4. The Stress Vector

Consider a continuum body and an interior point $P(x_i)$. Make a cut through P with a plane with exterior normal \mathbf{n} , as illustrated in Figure C.1.

The *stress vector* at P for direction \mathbf{n} is defined as

$$\mathbf{t}_n = \lim_{\Delta A \rightarrow 0} \frac{\Delta \mathbf{t}}{\Delta A}, \quad (\text{C.11})$$

where ΔA is a differential area surrounding P on the cutting plane (see Figure C.1).

The *couple stress vector* for direction \mathbf{n} is

$$\mathbf{m}_n = \lim_{\Delta A \rightarrow 0} \frac{\Delta \mathbf{m}}{\Delta A}. \quad (\text{C.12})$$

It is optional to include \mathbf{m}_n in the theory of stress. Doing so leads to the so-called *polar material* models. In classical continuum mechanics it is generally assumed that $\mathbf{m}_n = 0$, which corresponds to non-polar materials. Polar material models are generally considered only when continua are subjected to strong electromagnetic fields.

§C.5. The Stress Tensor

Consideration of the equilibrium of an elemental tetrahedron at P whose faces are normal to x_1 , x_2 , x_3 and \mathbf{n} leads to the expression

$$t_i = \sigma_{ij} n_j, \quad (\text{C.13})$$

where t_i is the component of \mathbf{t} in the x_i direction, and n_j are the components of \mathbf{n} . The nine values σ_{ij} are the components of the Cauchy stress tensor

$$\underline{\sigma} = [\sigma_{ij}] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \sigma_{13} \\ \sigma_{21} & \sigma_{22} & \sigma_{23} \\ \sigma_{31} & \sigma_{32} & \sigma_{33} \end{bmatrix}. \quad (\text{C.14})$$

For non-polar materials this tensor is *symmetric*. That is, $\sigma_{ij} = \sigma_{ji}$.

§C.6. Equilibrium Equations

Consider the equilibrium of a body of volume V and surface S subject to the following actions

- (a) Body force field \mathbf{f} of components b_i in V
- (b) Acceleration field $\mathbf{a} = d^2\mathbf{u}/dt^2 = \ddot{\mathbf{u}}$ (t = time) of components a_i in V
- (c) Stress vectors \mathbf{t} of components t_i on S

Dynamic equilibrium along any direction x_i requires

$$\int_S t_i dS + \int_V b_i dV = \int_V \rho a_i dV, \quad (\text{C.15})$$

where ρ is the body density. Substitute $t_i = \sigma_{ij}n_j$ in the surface integral:

$$\int_S \sigma_{ij}n_j dS + \int_V b_i dV = \int_V \rho a_i dV. \quad (\text{C.16})$$

To transform the surface integral to a volume integral we use Gauss' divergence theorem. For any vector field \mathbf{a} :

$$\int_S \mathbf{a} \cdot \mathbf{n} dS = \int_V \text{div} \cdot \mathbf{a} dV. \quad (\text{C.17})$$

or in component form

$$\int_S a_j n_j dS = \int_V \frac{\partial a_j}{\partial x_j} dV. \quad (\text{C.18})$$

Consequently the equilibrium integral (C.15) may be reduced to

$$\int_V [\sigma_{ij,j} + b_i - \rho a_i] dV = 0, \quad (\text{C.19})$$

for an arbitrary volume. Because the volume is arbitrary we must have

$$\sigma_{ij,j} + b_i - \rho a_i = 0. \quad (\text{C.20})$$

These are the three differential equations of dynamic equilibrium, which are obtained by setting the free index i to 1, 2 and 3. These are also called the internal equilibrium equations, or balance equations. If the medium is at rest or moving uniformly with respect to an inertial frame, the accelerations vanish and we obtain the equations of *static equilibrium*

$$\sigma_{ij,j} + b_i = 0. \quad (\text{C.21})$$

Example C.1. If $i = 1$ the first static equilibrium equation along axis x_1 is

$$\sigma_{1j,j} + b_1 = 0 \quad (\text{C.22})$$

or, written in full

$$\frac{\partial \sigma_{11}}{\partial x_1} + \frac{\partial \sigma_{12}}{\partial x_2} + \frac{\partial \sigma_{13}}{\partial x_3} + b_1 = 0. \quad (\text{C.23})$$

In conventional (engineering) notation:

$$\frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \tau_{xy}}{\partial y} + \frac{\partial \tau_{xz}}{\partial z} + b_x = 0. \quad (\text{C.24})$$

§C.7. Constitutive Equations

Constitutive equations characterize the behavior of the material of mechanical bodies. These equations are relations that constrain the space of deformations of a body (as defined by the strain tensor) and the state of internal forces (as defined by the stress tensor). The relations hold at each point of the body. They are generally partial differential equations, or even integrodifferential equations, in space and time.

The simplest type of constitutive equations are homogeneous linear algebraic relations that connect the stress and stress tensors at each point. This type of constitutive equation characterizes the *linear elastic solids*, which are the only ones we shall consider in this course.

§C.7.1. Elastic Solids

An elastic solid is a body characterized by two configurations:

- (1) The *natural state* or *undeformed state*, which is taken by the solid in the absence of applied forces
- (2) The *deformed state*, attainable by any *reversible* process.

The concepts just outlined are closely associated with the idea of *stored energy*, *strain energy* or *stress energy*. This expresses mathematically that the behavior of an elastic solid is independent of the preceding history of the material. In other words, the state of stress depends only on the state of strain, and not on the path followed to get to that strain.

§C.7.2. The Strain Energy

Let us define \mathcal{U} as the strain energy per unit of deformed volume, also called the *strain energy density*. This is generally a function of position (x_i) and of the deformation of the body at that position. Since in linear elasticity the deformation may be characterized by the strain tensor ϵ_{ij}

$$\mathcal{U} = \mathcal{U}(x_i, e_{ij}). \quad (\text{C.25})$$

Additional properties of this function are:

- (a) \mathcal{U} is a scalar invariant: it is unaffected by rigid body displacements and by the orientation of the global RCC system.
- (b) If \mathcal{U} is independent of x_i over a volume V , the material is said to be *homogeneous* in V .
- (c) The material is said to be *isotropic* if the stress-strain law is independent of directions in the material. If the material is isotropic, the strain energy density must be a function only of the invariants of the strain tensor.

Remark C.1. Isotropy must not be confused with invariance. All materials in classical mechanics satisfy the invariance principle (material properties do not depend on the observer), but not all materials are isotropic.

§C.7.3. Principle of Virtual Work

Consider a body in static equilibrium. The principle of virtual work (PVW) states that the virtual work done by all forces acting on that body during a virtual displacement* $\delta \mathbf{u}$ must be zero. Mathematically,

$$\delta W_i + \delta W = 0, \quad (\text{C.26})$$

where W_i and W are the virtual work done by the internal and external forces, respectively. The principle can be derived mathematically by taking the equilibrium equations and stress boundary conditions

$$\sigma_{ij,j} + f_i = 0, \quad \sigma_{ij} n_j = \hat{t}_i, \quad (\text{C.27})$$

multiply the first by δu_i , integrate over V and apply the divergence theorem which for symmetric σ_{ij} yields

$$\int_V (\sigma_{ij,j} + f_i) \delta u_i = \int_V [-\sigma_{ij} \delta \frac{1}{2}(u_{i,j} + u_{j,i}) + f_i \delta u_i] dV + \int_S \sigma_{ij} n_j \delta u_i dS = 0. \quad (\text{C.28})$$

Split the surface integral over $S_t \cup S_u$. The integral over the latter vanishes. The former can be transformed using the second of (F.3) weighted by δu_i and integrated over S_t to produce

$$\int_V \sigma_{ij} \delta e_{ij} dV = \int_V f_i \delta u_i + \int_{S_t} \hat{t}_i \delta u_i dS. \quad (\text{C.29})$$

But $\delta W_i = -\delta U$, where U is the total stored strain energy:

$$U = \int_V \mathcal{U} dV. \quad (\text{C.30})$$

Comparing with (C.5) we get

$$\delta U = \int_V \sigma_{ij} \delta e_{ij} dV, \quad \delta \mathcal{U} = \sigma_{ij} \delta e_{ij}. \quad (\text{C.31})$$

§C.7.4. Hyperelastic Constitutive Equations

For a homogeneous linear elastic body possessing a strain energy density (such material is called “hyperelastic” in the literature)

$$\mathcal{U} = \mathcal{U}(e_{ij}). \quad (\text{C.32})$$

$$\delta \mathcal{U} = \frac{\partial \mathcal{U}}{\partial e_{ij}} \delta e_{ij} = \frac{\partial \mathcal{U}}{\partial e_{11}} \delta e_{11} + \frac{\partial \mathcal{U}}{\partial e_{12}} \delta e_{12} + \cdots = \sigma_{ij} \delta e_{ij}. \quad (\text{C.33})$$

Since $\delta \mathcal{U} = \sigma_{ij} \delta e_{ij}$ we must have

$$\sigma_{ij} = \frac{\partial \mathcal{U}}{\partial e_{ij}}. \quad (\text{C.34})$$

* A virtual displacement is a change in the geometric configuration of the body compatible with all kinematic constraints, made while keeping all forces and stress acting on the body frozen.

These are the constitutive equations of a linear *hyperelastic* material.

For an isotropic linear elastic material, \mathcal{U} must be a quadratic form in e_{ij} that preserves the invariants of the strain tensor. It is shown in books on elasticity that this condition leads to the *generalized Hooke's law*

$$\sigma_{ij} = \lambda e_{ii} \delta_{ij} + 2\mu e_{ij}. \quad (\text{C.35})$$

Here λ and μ are called the Lamé coefficients; both of which have dimensions of stress. In engineering applications the material coefficients E , G , and ν (modulus of elasticity, shear modulus, and Poisson's ratio, respectively) are more commonly used. These are related to λ and μ by

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \quad \nu = \frac{\lambda}{2(\lambda + \mu)}, \quad G = \frac{E}{2(1 + \nu)}. \quad (\text{C.36})$$

Conversely,

$$\lambda = \frac{Ev}{(1 + \nu)(1 - 2\nu)}, \quad \mu = G = \frac{E}{2(1 + \nu)}. \quad (\text{C.37})$$

D

Linear Algebra: Determinants, Inverses, Rank

TABLE OF CONTENTS

	Page
§D.1. Introduction	D–3
§D.2. Determinants	D–3
§D.2.1. Some Properties of Determinants	D–3
§D.2.2. Cramer’s Rule	D–5
§D.2.3. Homogeneous Systems	D–6
§D.3. Singular Matrices, Rank	D–6
§D.3.1. Rank Deficiency	D–7
§D.3.2. Rank of Matrix Sums and Products	D–7
§D.3.3. Singular Systems: Particular and Homogeneous Solutions . . .	D–7
§D.3.4. Rank of Rectangular Matrices	D–8
§D.4. Matrix Inversion	D–8
§D.4.1. Explicit Computation of Inverses	D–9
§D.4.2. Some Properties of the Inverse	D–10
§D.5. The Inverse of a Sum of Matrices	D–11
§D.6. The Sherman-Morrison and Related Formulas	D–12
§D.6.1. The Sherman-Morrison Formula	D–12
§D.6.2. The Woodbury Formula	D–12
§D.6.3. Formulas for Modified Determinants	D–13
§D. Notes and Bibliography	D–13
§D. Exercises	D–14

§D.1. Introduction

This Chapter discusses more specialized properties of matrices, such as determinants, inverses and rank. These apply only to *square* matrices unless extension to rectangular matrices is explicitly stated.

§D.2. Determinants

The *determinant* of a *square* matrix $\mathbf{A} = [a_{ij}]$ is a number denoted by $|\mathbf{A}|$ or $\det(\mathbf{A})$, through which important properties such as singularity can be briefly characterized. This number is defined as the following function of the matrix elements:

$$|\mathbf{A}| = \det(\mathbf{A}) = \pm \prod a_{1j_1} a_{2j_2} \cdots a_{nj_n}, \quad (\text{D.1})$$

where the column indices j_1, j_2, \dots, j_n are taken from the set $\{1, 2, \dots, n\}$, with no repetitions allowed. The plus (minus) sign is taken if the permutation $(j_1 j_2 \dots j_n)$ is even (odd).

Example D.1. For a 2×2 matrix,

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \quad (\text{D.2})$$

Example D.2. For a 3×3 matrix,

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}. \quad (\text{D.3})$$

Remark D.1. The concept of determinant is not applicable to rectangular matrices or to vectors. Thus the notation $|\mathbf{x}|$ for a vector \mathbf{x} can be reserved for its magnitude (as in Appendix A) without risk of confusion.

Remark D.2. Inasmuch as the product (D.1) contains $n!$ terms, the calculation of $|\mathbf{A}|$ from the definition is impractical for general matrices whose order exceeds 3 or 4. For example, if $n = 10$, the product (D.1) contains $10! = 3,628,800$ terms, each involving 9 multiplications, so over 30 million floating-point operations would be required to evaluate $|\mathbf{A}|$ according to that definition. A more practical method based on matrix decomposition is described in Remark D.3.

§D.2.1. Some Properties of Determinants

Some useful rules associated with the calculus of determinants are listed next.

- I. Rows and columns can be interchanged without affecting the value of a determinant. Consequently

$$|\mathbf{A}| = |\mathbf{A}^T|. \quad (\text{D.4})$$

- II. If two rows, or two columns, are interchanged the sign of the determinant is reversed. For example:

$$\begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = - \begin{vmatrix} 1 & -2 \\ 3 & 4 \end{vmatrix}. \quad (\text{D.5})$$

- III. If a row (or column) is changed by adding to or subtracting from its elements the corresponding elements of any other row (or column) the determinant remains unaltered. For example:

$$\begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = \begin{vmatrix} 3+1 & 4-2 \\ 1 & -2 \end{vmatrix} = \begin{vmatrix} 4 & 2 \\ 1 & -2 \end{vmatrix} = -10. \quad (\text{D.6})$$

- IV. If the elements in any row (or column) have a common factor α then the determinant equals the determinant of the corresponding matrix in which $\alpha = 1$, multiplied by α . For example:

$$\begin{vmatrix} 6 & 8 \\ 1 & -2 \end{vmatrix} = 2 \begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = 2 \times (-10) = -20. \quad (\text{D.7})$$

- V. When at least one row (or column) of a matrix is a linear combination of the other rows (or columns) the determinant is zero. Conversely, if the determinant is zero, then at least one row and one column are linearly dependent on the other rows and columns, respectively. For example, consider

$$\begin{vmatrix} 3 & 2 & 1 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{vmatrix}. \quad (\text{D.8})$$

This determinant is zero because the first column is a linear combination of the second and third columns:

$$\text{column 1} = \text{column 2} + \text{column 3}. \quad (\text{D.9})$$

Similarly, there is a linear dependence between the rows which is given by the relation

$$\text{row 1} = \frac{7}{8} \text{row 2} + \frac{4}{5} \text{row 3}. \quad (\text{D.10})$$

- VI. The determinant of an upper triangular or lower triangular matrix is the product of the main diagonal entries. For example,

$$\begin{vmatrix} 3 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & 4 \end{vmatrix} = 3 \times 2 \times 4 = 24. \quad (\text{D.11})$$

This rule is easily verified from the definition (D.1) because all terms vanish except $j_1 = 1$, $j_2 = 2, \dots, j_n = n$, which is the product of the main diagonal entries. Diagonal matrices are a particular case of this rule.

- VII. The determinant of the product of two square matrices is the product of the individual determinants:

$$|\mathbf{AB}| = |\mathbf{A}| |\mathbf{B}|. \quad (\text{D.12})$$

The proof requires the concept of triangular decomposition, which is covered in the Remark below. This rule can be generalized to any number of factors. One immediate application is to matrix powers: $|\mathbf{A}^2| = |\mathbf{A}| |\mathbf{A}| = |\mathbf{A}|^2$, and more generally $|\mathbf{A}^n| = |\mathbf{A}|^n$ for integer n .

- VIII. The determinant of the transpose of a matrix is the same as that of the original matrix:

$$|\mathbf{A}^T| = |\mathbf{A}|. \quad (\text{D.13})$$

This rule can be directly verified from the definition of determinant, and also as direct consequence of Rule I.

Remark D.3. Rules VI and VII are the key to the practical evaluation of determinants. Any square nonsingular matrix \mathbf{A} (where the qualifier “nonsingular” is explained in §D.3) can be decomposed as the product of two triangular factors

$$\mathbf{A} = \mathbf{L}\mathbf{U}, \quad (\text{D.14})$$

in which \mathbf{L} is unit lower triangular and \mathbf{U} is upper triangular. This is called a LU triangularization, LU factorization or LU decomposition. It can be carried out in $O(n^3)$ floating point operations. According to rule VII:

$$|\mathbf{A}| = |\mathbf{L}| |\mathbf{U}|. \quad (\text{D.15})$$

According to rule VI, $|\mathbf{L}| = 1$ and $|\mathbf{U}| = u_{11}u_{22} \dots u_{nn}$. The last operation requires only $O(n)$ operations. Thus the evaluation of $|\mathbf{A}|$ is dominated by the effort involved in computing the factorization (D.14). For $n = 10$, that effort is approximately $10^3 = 1000$ floating-point operations, compared to approximately 3×10^7 from the naive application of the definition (D.1), as noted in Remark D.2. Thus the LU-based method is roughly 30,000 times faster for that modest matrix order, and the ratio increases exponentially for large n .

§D.2.2. Cramer's Rule

Cramer's rule provides a recipe for solving linear algebraic equations directly in terms of determinants. Let the simultaneous equations be as usual denoted as

$$\mathbf{A} \mathbf{x} = \mathbf{y}, \quad (\text{D.16})$$

in which \mathbf{A} is a given $n \times n$ matrix, \mathbf{y} is a given $n \times 1$ vector, and \mathbf{x} is the $n \times 1$ vector of unknowns. The explicit form of (D.16) is Equation (A.1) of Appendix A, with $n = m$.

The explicit solution for the components x_1, x_2, \dots, x_n of \mathbf{x} in terms of determinants is

$$x_1 = \frac{\begin{vmatrix} y_1 & a_{12} & a_{13} & \dots & a_{1n} \\ y_2 & a_{22} & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ y_n & a_{n2} & a_{n3} & \dots & a_{nn} \end{vmatrix}}{|\mathbf{A}|}, \quad x_2 = \frac{\begin{vmatrix} a_{11} & y_1 & a_{13} & \dots & a_{1n} \\ a_{21} & y_2 & a_{23} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n1} & y_n & a_{n3} & \dots & a_{nn} \end{vmatrix}}{|\mathbf{A}|}, \quad \dots \quad (\text{D.17})$$

The rule can be remembered as follows: in the numerator of the quotient for x_j , replace the j^{th} column of \mathbf{A} by the right-hand side \mathbf{y} .

This method of solving simultaneous equations is known as *Cramer's rule*. Because the explicit computation of determinants is impractical for $n > 3$ as explained in Remark C.3, direct use of the rule has practical value only for $n = 2$ and $n = 3$ (it is marginal for $n = 4$). But such small-order systems arise often in finite element calculations at the *Gauss point level*; consequently implementors should be aware of this rule for such applications.

Example D.3. Solve the 3×3 linear system

$$\begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \\ 3 \end{bmatrix}, \quad (\text{D.18})$$

by Cramer's rule:

$$x_1 = \frac{\begin{vmatrix} 8 & 2 & 1 \\ 5 & 2 & 0 \\ 3 & 0 & 2 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1, \quad x_2 = \frac{\begin{vmatrix} 5 & 8 & 1 \\ 3 & 5 & 0 \\ 1 & 3 & 2 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1, \quad x_3 = \frac{\begin{vmatrix} 5 & 2 & 8 \\ 3 & 2 & 5 \\ 1 & 0 & 3 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1. \quad (\text{D.19})$$

Example D.4. Solve the 2×2 linear algebraic system

$$\begin{bmatrix} 2 + \beta & -\beta \\ -\beta & 1 + \beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \quad (\text{D.20})$$

by Cramer's rule:

$$x_1 = \frac{\begin{vmatrix} 5 & -\beta \\ 0 & 1 + \beta \end{vmatrix}}{\begin{vmatrix} 2 + \beta & -\beta \\ -\beta & 1 + \beta \end{vmatrix}} = \frac{5 + 5\beta}{2 + 3\beta}, \quad x_2 = \frac{\begin{vmatrix} 2 + \beta & 5 \\ -\beta & 0 \end{vmatrix}}{\begin{vmatrix} 2 + \beta & -\beta \\ -\beta & 1 + \beta \end{vmatrix}} = \frac{5\beta}{2 + 3\beta}. \quad (\text{D.21})$$

Remark D.4. Cramer's rule importance has grown in *symbolic computations* carried out by computer algebra systems. This happens when the entries of \mathbf{A} and \mathbf{y} are algebraic expressions. For example the example system (D.20). In such cases Cramer's rule may be competitive with factorization methods for up to moderate matrix orders, for example $n \leq 20$. The reason is that determinantal products may be simplified on the fly.

§D.2.3. Homogeneous Systems

One immediate consequence of Cramer's rule is what happens if

$$y_1 = y_2 = \dots = y_n = 0. \quad (\text{D.22})$$

The linear equation systems with a null right hand side

$$\mathbf{Ax} = \mathbf{0}, \quad (\text{D.23})$$

is called a *homogeneous system*. From the rule (D.17) we see that if $|\mathbf{A}|$ is nonzero, all solution components are zero, and consequently the only possible solution is the trivial one $\mathbf{x} = \mathbf{0}$. The case in which $|\mathbf{A}|$ vanishes is discussed in the next section.

§D.3. Singular Matrices, Rank

If the determinant $|\mathbf{A}|$ of a $n \times n$ square matrix $\mathbf{A} \equiv \mathbf{A}_n$ is zero, then the matrix is said to be *singular*. This means that at least one row and one column are linearly dependent on the others. If this row and column are removed, we are left with another matrix, say \mathbf{A}_{n-1} , to which we can apply the same criterion. If the determinant $|\mathbf{A}_{n-1}|$ is zero, we can remove another row and column from it to get \mathbf{A}_{n-2} , and so on. Suppose that we eventually arrive at an $r \times r$ matrix \mathbf{A}_r whose determinant is nonzero. Then matrix \mathbf{A} is said to have *rank* r , and we write $\text{rank}(\mathbf{A}) = r$.

If the determinant of \mathbf{A} is nonzero, then \mathbf{A} is said to be *nonsingular*. The rank of a nonsingular $n \times n$ matrix is equal to n .

Obviously the rank of \mathbf{A}^T is the same as that of \mathbf{A} since it is only necessary to transpose “row” and “column” in the definition.

The notion of rank can be extended to rectangular matrices as outlined in section §C.2.4 below. That extension, however, is not important for the material covered here.

Example D.5. The 3×3 matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{bmatrix}, \quad (\text{D.24})$$

has rank $r = 3$ because $|\mathbf{A}| = -3 \neq 0$.

Example D.6. The matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{bmatrix}, \quad (\text{D.25})$$

already used as an example in §C.1.1 is singular because its first row and column may be expressed as linear combinations of the others through the relations (D.9) and (D.10). Removing the first row and column we are left with a 2×2 matrix whose determinant is $2 \times 3 - (-1) \times (-1) = 5 \neq 0$. Consequently (D.25) has rank $r = 2$.

§D.3.1. Rank Deficiency

If the square matrix \mathbf{A} is supposed to be of rank r but in fact has a smaller rank $\bar{r} < r$, the matrix is said to be *rank deficient*. The number $r - \bar{r} > 0$ is called the *rank deficiency*.

Example D.7. Suppose that the *unconstrained* master stiffness matrix \mathbf{K} of a finite element has order n , and that the element possesses b independent rigid body modes. Then the expected rank of \mathbf{K} is $r = n - b$. If the actual rank is less than r , the finite element model is said to be rank-deficient. This is usually undesirable.

Example D.8. An illustration of the foregoing rule, consider the two-node, 4-DOF, Bernoulli-Euler plane beam element stiffness derived in Chapter 12:

$$\mathbf{K} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ & 4L^2 & -6L & 2L^2 \\ & & 12 & -6L \\ \text{symm} & & & 4L^2 \end{bmatrix}, \quad (\text{D.26})$$

in which EI and L are nonzero scalars. It may be verified that this 4×4 matrix has rank 2. The number of rigid body modes is 2, and the expected rank is $r = 4 - 2 = 2$. Consequently this model is rank sufficient.

§D.3.2. Rank of Matrix Sums and Products

In finite element analysis matrices are often built through sum and product combinations of simpler matrices. Two important rules apply to “rank propagation” through those combinations.

The rank of the product of two square matrices \mathbf{A} and \mathbf{B} cannot exceed the smallest rank of the multiplicand matrices. That is, if the rank of \mathbf{A} is r_a and the rank of \mathbf{B} is r_b ,

$$\text{rank}(\mathbf{AB}) \leq \min(r_a, r_b). \quad (\text{D.27})$$

Regarding sums: the rank of a matrix sum cannot exceed the sum of ranks of the summand matrices. That is, if the rank of \mathbf{A} is r_a and the rank of \mathbf{B} is r_b ,

$$\text{rank}(\mathbf{A} + \mathbf{B}) \leq r_a + r_b. \quad (\text{D.28})$$

§D.3.3. Singular Systems: Particular and Homogeneous Solutions

Having introduced the notion of rank we can now discuss what happens to the linear system (D.16) when the determinant of \mathbf{A} vanishes, meaning that its rank is less than n . If so, (D.16) has either no solution or an infinite number of solutions. Cramer's rule is of limited or no help in this situation.

To discuss this case further we note that if $|\mathbf{A}| = 0$ and the rank of \mathbf{A} is $r = n - d$, where $d \geq 1$ is the *rank deficiency*, then there exist d nonzero independent vectors \mathbf{z}_i , $i = 1, \dots, d$ such that

$$\mathbf{A} \mathbf{z}_i = \mathbf{0}. \quad (\text{D.29})$$

These d vectors, suitably orthonormalized, are called *null eigenvectors* of \mathbf{A} , and form a basis for its *null space*.

Let \mathbf{Z} denote the $n \times d$ matrix obtained by collecting the \mathbf{z}_i as columns. If \mathbf{y} in (D.16) is in the *range* of \mathbf{A} , that is, there exists a nonzero \mathbf{x}_p such that $\mathbf{y} = \mathbf{A}\mathbf{x}_p$, its general solution is

$$\mathbf{x} = \mathbf{x}_p + \mathbf{x}_h = \mathbf{x}_p + \mathbf{Z}\mathbf{w}, \quad (\text{D.30})$$

where \mathbf{w} is an arbitrary $d \times 1$ weighting vector. This statement can be easily verified by substituting this solution into $\mathbf{A}\mathbf{x} = \mathbf{y}$ and noting that $\mathbf{A}\mathbf{Z}$ vanishes.

The components \mathbf{x}_p and \mathbf{x}_h are called the *particular* and *homogeneous* portions respectively, of the total solution \mathbf{x} . (The terminology: *homogeneous solution* and *particular solution*, are often used.) If $\mathbf{y} = \mathbf{0}$ only the homogeneous portion remains.

If \mathbf{y} is not in the range of \mathbf{A} , system (D.16) does not generally have a solution in the conventional sense, although least-square solutions can usually be constructed. The reader is referred to the many textbooks in linear algebra for further details.

§D.3.4. Rank of Rectangular Matrices

The notion of rank can be extended to rectangular matrices, real or complex, as follows. Let \mathbf{A} be $m \times n$. Its *column range space* $\mathcal{R}(\mathbf{A})$ is the subspace spanned by $\mathbf{A}\mathbf{x}$ where \mathbf{x} is the set of all complex n -vectors. Mathematically: $\mathcal{R}(\mathbf{A}) = \{\mathbf{A}\mathbf{x} : \mathbf{x} \in C^n\}$. The rank r of \mathbf{A} is the dimension of $\mathcal{R}(\mathbf{A})$.

The *null space* $\mathcal{N}(\mathbf{A})$ of \mathbf{A} is the set of n -vectors \mathbf{z} such that $\mathbf{A}\mathbf{z} = \mathbf{0}$. The dimension of $\mathcal{N}(\mathbf{A})$ is $n - r$.

Using these definitions, the product and sum rules (D.27) and (D.28) generalize to the case of rectangular (but conforming) \mathbf{A} and \mathbf{B} . So does the treatment of linear equation systems $\mathbf{A}\mathbf{x} = \mathbf{y}$ in which \mathbf{A} is rectangular. Such systems often arise in the fitting of observation and measurement data.

In finite element methods, rectangular matrices appear in change of basis through congruential transformations, and in the treatment of multifreedom constraints.

§D.4. Matrix Inversion

The *inverse* of a square nonsingular matrix \mathbf{A} is represented by the symbol \mathbf{A}^{-1} and is defined by the relation

$$\mathbf{A} \mathbf{A}^{-1} = \mathbf{I}. \quad (\text{D.31})$$

The most important application of the concept of inverse is the solution of linear systems. Suppose that, in the usual notation, we have

$$\mathbf{A} \mathbf{x} = \mathbf{y}. \quad (\text{D.32})$$

Premultiplying both sides by \mathbf{A}^{-1} we get the inverse relationship

$$\mathbf{x} = \mathbf{A}^{-1} \mathbf{y}. \quad (\text{D.33})$$

More generally, consider the matrix equation for multiple (m) right-hand sides:

$$\underset{n \times n}{\mathbf{A}} \underset{n \times m}{\mathbf{X}} = \underset{n \times m}{\mathbf{Y}}, \quad (\text{D.34})$$

which reduces to (D.32) for $m = 1$. The inverse relation that gives \mathbf{X} as function of \mathbf{Y} is

$$\mathbf{X} = \mathbf{A}^{-1} \mathbf{Y}. \quad (\text{D.35})$$

In particular, the solution of

$$\mathbf{A} \mathbf{X} = \mathbf{I}, \quad (\text{D.36})$$

is $\mathbf{X} = \mathbf{A}^{-1}$. Practical methods for computing inverses are based on directly solving this equation; see Remark D.4.

§D.4.1. Explicit Computation of Inverses

The explicit calculation of matrix inverses is seldom needed in large matrix computations. But occasionally the need arises for the explicit inverse of small matrices that appear in element level computations. For example, the inversion of Jacobian matrices at Gauss points, or of constitutive matrices.

A general formula for elements of the inverse can be obtained by specializing Cramer's rule to (?). Let $\mathbf{B} = [b_{ij}] = \mathbf{A}^{-1}$. Then

$$b_{ij} = \frac{A_{ji}}{|\mathbf{A}|}, \quad (\text{D.37})$$

in which A_{ji} denotes the so-called *adjoint* of entry a_{ij} of \mathbf{A} . The adjoint A_{ji} is defined as the determinant of the submatrix of order $(n - 1) \times (n - 1)$ obtained by deleting the j^{th} row and i^{th} column of \mathbf{A} , multiplied by $(-1)^{i+j}$.

This direct inversion procedure is useful only for small matrix orders, say 2 or 3. In the examples below the explicit inversion formulas for second and third order matrices are listed.

Example D.9. For order $n = 2$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \quad \mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}, \quad (\text{D.38})$$

in which $|\mathbf{A}|$ is given by (D.2).

Example D.10. For order $n = 3$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad \mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, \quad (\text{D.39})$$

where

$$\begin{aligned} b_{11} &= \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}, & b_{21} &= -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix}, & b_{31} &= \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}, \\ b_{12} &= -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}, & b_{22} &= \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}, & b_{32} &= -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix}, \\ b_{13} &= \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}, & b_{23} &= -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}, & b_{33} &= \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \end{aligned} \quad (\text{D.40})$$

in which $|\mathbf{A}|$ is given by (D.3).

Example D.11.

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 2 \\ 3 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \quad \mathbf{A}^{-1} = -\frac{1}{8} \begin{bmatrix} 1 & -4 & 2 \\ -2 & 0 & 4 \\ -1 & 4 & -10 \end{bmatrix}. \quad (\text{D.41})$$

If the order exceeds 3, the general inversion formula based on Cramer's rule becomes rapidly useless because it displays combinatorial complexity as noted in a previous Remark. For numerical work it is preferable to solve (D.36) after \mathbf{A} is factored. Those techniques are described in detail in linear algebra books; see also Remark C.4.

§D.4.2. Some Properties of the Inverse

- I. Assuming that \mathbf{A}^{-1} exists, the The inverse of its transpose is equal to the transpose of the inverse:

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T, \quad (\text{D.42})$$

because

$$(\mathbf{A}\mathbf{A}^{-1}) = (\mathbf{A}\mathbf{A}^{-1})^T = (\mathbf{A}^{-1})^T \mathbf{A}^T = \mathbf{I}. \quad (\text{D.43})$$

- II. The inverse of a symmetric matrix is also symmetric. Because of the previous rule, $(\mathbf{A}^T)^{-1} = \mathbf{A}^{-1} = (\mathbf{A}^{-1})^T$, hence \mathbf{A}^{-1} is also symmetric.

- III. The inverse of a matrix product is the reverse product of the inverses of the factors:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \quad (\text{D.44})$$

This is easily verified by substituting both sides of (D.39) into (D.31). This property generalizes to an arbitrary number of factors.

- IV. For a diagonal matrix \mathbf{D} in which all diagonal entries are nonzero, \mathbf{D}^{-1} is again a diagonal matrix with entries $1/d_{ii}$. The verification is straightforward.

V. If \mathbf{S} is a block diagonal matrix:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{22} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}_{nn} \end{bmatrix} = \text{diag} [\mathbf{S}_{ii}], \quad (\text{D.45})$$

then the inverse matrix is also block diagonal and is given by

$$\mathbf{S}^{-1} = \begin{bmatrix} \mathbf{S}_{11}^{-1} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{22}^{-1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33}^{-1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \cdots & \mathbf{S}_{nn}^{-1} \end{bmatrix} = \text{diag} [\mathbf{S}_{ii}^{-1}]. \quad (\text{D.46})$$

VI. The inverse of an upper triangular matrix is also an upper triangular matrix. The inverse of a lower triangular matrix is also a lower triangular matrix. Both inverses can be computed in $O(n^2)$ floating-point operations.

Remark D.5. The practical numerical calculation of inverses is based on triangular factorization. Given a nonsingular $n \times n$ matrix \mathbf{A} , calculate its LU factorization $\mathbf{A} = \mathbf{L}\mathbf{U}$, which can be obtained in $O(n^3)$ operations. Then solve the linear triangular systems:

$$\mathbf{U}\mathbf{Y} = \mathbf{I}, \quad \mathbf{L}\mathbf{X} = \mathbf{Y}, \quad (\text{D.47})$$

and the computed inverse \mathbf{A}^{-1} appears in \mathbf{X} . One can overwrite \mathbf{I} with \mathbf{Y} and \mathbf{Y} with \mathbf{X} . The whole process can be completed in $O(n^3)$ floating-point operations. For symmetric matrices the alternative decomposition $\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T$, where \mathbf{L} is unit lower triangular and \mathbf{D} is diagonal, is generally preferred to save computing time and storage.

§D.5. The Inverse of a Sum of Matrices

The formula for the inverse of a matrix product: $(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ is not too different from its scalar counterpart: $(ab)^{-1} = (1/a)(1/b) = (1/b)(1/a)$, except that factor order matters. On the other hand, formulas for matrix sum inverses in terms of the summands are considerably more involved, and there are many variants. We consider here the expression of $(\mathbf{A} + \mathbf{B})^{-1}$ where both \mathbf{A} and \mathbf{B} are square and \mathbf{A} is nonsingular. We begin from the identity introduced by Henderson and Searle in their review article [338]:

$$(\mathbf{I} + \mathbf{P})^{-1} = (\mathbf{I} + \mathbf{P})^{-1} (\mathbf{I} + \mathbf{P} - \mathbf{P}) = \mathbf{I} - (\mathbf{I} + \mathbf{P})^{-1} \mathbf{P}, \quad (\text{D.48})$$

in which \mathbf{P} is a square matrix. Using (D.48) we may develop $(\mathbf{A} + \mathbf{B})^{-1}$ as follows:

$$\begin{aligned} (\mathbf{A} + \mathbf{B})^{-1} &= (\mathbf{A}(\mathbf{I} + \mathbf{A}^{-1}\mathbf{B})^{-1})^{-1} = (\mathbf{I} + \mathbf{A}^{-1}\mathbf{B})^{-1} \mathbf{A}^{-1} \mathbf{B} \\ &= (\mathbf{I} - (\mathbf{I} + \mathbf{A}^{-1}\mathbf{B})^{-1} \mathbf{A}^{-1}) \mathbf{A}^{-1} = \mathbf{A}^{-1} - (\mathbf{I} + \mathbf{A}^{-1}\mathbf{B})^{-1} \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-1}. \end{aligned} \quad (\text{D.49})$$

Here \mathbf{B} may be singular. If $\mathbf{B} = \mathbf{0}$, it reduces to $\mathbf{A}^{-1} = \mathbf{A}^{-1}$. The check $\mathbf{B} = \beta \mathbf{A}$ also works. The last expression in (D.49) may be further transformed by matrix manipulations as

$$\begin{aligned}
 (\mathbf{A} + \mathbf{B})^{-1} &= \mathbf{A}^{-1} - (\mathbf{I} + \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-1} \\
 &= \mathbf{A}^{-1} - \mathbf{A}^{-1} (\mathbf{I} + \mathbf{B} \mathbf{A}^{-1})^{-1} \mathbf{B} \mathbf{A}^{-1} \\
 &= \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} (\mathbf{I} + \mathbf{A}^{-1} \mathbf{B})^{-1} \mathbf{A}^{-1} \\
 &= \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} \mathbf{A}^{-1} (\mathbf{I} + \mathbf{B} \mathbf{A}^{-1})^{-1}.
 \end{aligned} \tag{D.50}$$

In all of these forms \mathbf{B} may be singular (or null). If \mathbf{B} is also invertible, the third expression in (D.50) may be transformed to the a commonly used variant

$$(\mathbf{A} + \mathbf{B})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1} (\mathbf{A}^{-1} + \mathbf{B}^{-1})^{-1} \mathbf{A}^{-1}. \tag{D.51}$$

The case of singular \mathbf{A} may be handled using the notion of generalized inverses. This is a topic beyond the scope of this course, which may be studied, e.g., in the textbooks [77,119,596]. The special case of \mathbf{B} being of low rank merges with the Sherman-Morrison and Woodbury formulas, covered below.

§D.6. The Sherman-Morrison and Related Formulas

The Sherman-Morrison formula gives the inverse of a matrix modified by a rank-one matrix. The Woodbury formula extends the Sherman-Morrison formula to a modification of arbitrary rank. In structural analysis these formulas are of interest for problems of *structural modifications*, in which a finite-element (or, in general, a discrete model) is changed by an amount expressible as a low-rank correction to the original model.

§D.6.1. The Sherman-Morrison Formula

Let \mathbf{A} be a square $n \times n$ invertible matrix, whereas \mathbf{u} and \mathbf{v} are two n -vectors and β an arbitrary scalar. Assume that $\sigma = 1 + \beta \mathbf{v}^T \mathbf{A}^{-1} \mathbf{u} \neq 0$. Then

$$(\mathbf{A} + \beta \mathbf{u} \mathbf{v}^T)^{-1} = \mathbf{A}^{-1} - \frac{\beta}{\sigma} \mathbf{A}^{-1} \mathbf{u} \mathbf{v}^T \mathbf{A}^{-1}. \tag{D.52}$$

When $\beta = 1$ this is called the Sherman-Morrison formula after [644]. (For a history of this remarkable expression and its extensions, which are quite important in many applications such as statistics and probability, see the review paper by Henderson and Searle cited previously.) Since any rank-one correction to \mathbf{A} can be written as $\beta \mathbf{u} \mathbf{v}^T$, (D.52) gives the rank-one change to its inverse. The proof is by direct multiplication, as in Exercise D.5.

For practical computation of the change one solves the linear systems $\mathbf{A} \mathbf{a} = \mathbf{u}$ and $\mathbf{A} \mathbf{b} = \mathbf{v}$ for \mathbf{a} and \mathbf{b} , using the known \mathbf{A}^{-1} . Compute $\sigma = 1 + \beta \mathbf{v}^T \mathbf{a}$. If $\sigma \neq 0$, the change to \mathbf{A}^{-1} is the dyadic $-(\beta/\sigma) \mathbf{a} \mathbf{b}^T$.

§D.6.2. The Woodbury Formula

Let again \mathbf{A} be a square $n \times n$ invertible matrix, whereas \mathbf{U} and \mathbf{V} are two $n \times k$ matrices with $k \leq n$ and β an arbitrary scalar. Assume that the $k \times k$ matrix $\Sigma = \mathbf{I}_k + \beta \mathbf{V}^T \mathbf{A}^{-1} \mathbf{U}$, in which \mathbf{I}_k denotes the $k \times k$ identity matrix, is invertible. Then

$$(\mathbf{A} + \beta \mathbf{U} \mathbf{V}^T)^{-1} = \mathbf{A}^{-1} - \beta \mathbf{A}^{-1} \mathbf{U} \Sigma^{-1} \mathbf{V}^T \mathbf{A}^{-1}. \quad (\text{D.53})$$

This is called the Woodbury formula, after [774]. It reduces to (D.52) if $k = 1$, in which case $\Sigma \equiv \sigma$ is a scalar. The proof is by direct multiplication.

§D.6.3. Formulas for Modified Determinants

Let $\tilde{\mathbf{A}}$ denote the adjoint of \mathbf{A} . Taking the determinants from both sides of $\mathbf{A} + \beta \mathbf{u} \mathbf{v}^T$ one obtains

$$|\mathbf{A} + \beta \mathbf{u} \mathbf{v}^T| = |\mathbf{A}| + \beta \mathbf{v}^T \tilde{\mathbf{A}} \mathbf{u}. \quad (\text{D.54})$$

If \mathbf{A} is invertible, replacing $\tilde{\mathbf{A}} = |\mathbf{A}| \mathbf{A}^{-1}$ this becomes

$$|\mathbf{A} + \beta \mathbf{u} \mathbf{v}^T| = |\mathbf{A}| (1 + \beta \mathbf{v}^T \mathbf{A}^{-1} \mathbf{u}). \quad (\text{D.55})$$

Similarly, one can show that if \mathbf{A} is invertible, and \mathbf{U} and \mathbf{V} are $n \times k$ matrices,

$$|\mathbf{A} + \beta \mathbf{U} \mathbf{V}^T| = |\mathbf{A}| |\mathbf{I}_k + \beta \mathbf{V}^T \mathbf{A}^{-1} \mathbf{U}|. \quad (\text{D.56})$$

Notes and Bibliography

Much of the material summarized here is available in expanded form in linear algebra textbooks. For example, Bellman [67] and Strang [673].

For inverses of matrix sums, there are two SIAM Review articles: [314,338]. For an historical account of the topic and its close relation to the Schur complement, see the bibliography in Appendix P.

Exercises for Appendix D: Determinants, Inverses, Rank

EXERCISE D.1 If \mathbf{A} is a square matrix of order n and c a scalar, show that $\det(c\mathbf{A}) = c^n \det \mathbf{A}$.

EXERCISE D.2 Let \mathbf{u} and \mathbf{v} denote real n -vectors normalized to unit length, so that $\mathbf{u}^T \mathbf{u} = 1$ and $\mathbf{v}^T \mathbf{v} = 1$, and let \mathbf{I} denote the $n \times n$ identity matrix. Show that

$$\det(\mathbf{I} - \mathbf{u}\mathbf{v}^T) = 1 - \mathbf{v}^T \mathbf{u} \quad (\text{ED.1})$$

EXERCISE D.3 Let \mathbf{u} denote a real n -vector normalized to unit length, so that $\mathbf{u}^T \mathbf{u} = 1$ and \mathbf{I} denote the $n \times n$ identity matrix. Show that

$$\mathbf{H} = \mathbf{I} - 2\mathbf{u}\mathbf{u}^T \quad (\text{ED.2})$$

is orthogonal: $\mathbf{H}^T \mathbf{H} = \mathbf{I}$, and idempotent: $\mathbf{H}^2 = \mathbf{H}$. This matrix is called a *elementary Hermitian*, a *Householder matrix*, or a *reflector*. It is a fundamental ingredient of many linear algebra algorithms; for example the QR algorithm for finding eigenvalues.

EXERCISE D.4 The *trace* of a $n \times n$ square matrix \mathbf{A} , denoted $\text{trace}(\mathbf{A})$ is the sum $\sum_{i=1}^n a_{ii}$ of its diagonal coefficients. Show that if the entries of \mathbf{A} are real,

$$\text{trace}(\mathbf{A}^T \mathbf{A}) = \sum_{i=1}^n \sum_{j=1}^n a_{ij}^2 \quad (\text{ED.3})$$

EXERCISE D.5 Prove the Sherman-Morrison formula (D.53) by direct matrix multiplication.

EXERCISE D.6 Prove the Sherman-Morrison formula (D.53) for $\beta = 1$ by considering the following block bordered system

$$\begin{bmatrix} \mathbf{A} & \mathbf{U} \\ \mathbf{V}^T & \mathbf{I}_k \end{bmatrix} \begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{0} \end{bmatrix} \quad (\text{ED.4})$$

in which \mathbf{I}_k and \mathbf{I}_n denote the identity matrices of orders k and n , respectively. Solve (D.56) two ways: eliminating first \mathbf{B} and then \mathbf{C} , and eliminating first \mathbf{C} and then \mathbf{B} . Equate the results for \mathbf{B} .

EXERCISE D.7 Show that the eigenvalues of a real symmetric square matrix are real, and that the eigenvectors are real vectors.

EXERCISE D.8 Let the n real eigenvalues λ_i of a real $n \times n$ symmetric matrix \mathbf{A} be classified into two subsets: r eigenvalues are nonzero whereas $n - r$ are zero. Show that \mathbf{A} has rank r .

EXERCISE D.9 Show that if \mathbf{A} is p.d., $\mathbf{A}\mathbf{x} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$.

EXERCISE D.10 Show that for any real $m \times n$ matrix \mathbf{A} , $\mathbf{A}^T \mathbf{A}$ exists and is nonnegative.

EXERCISE D.11 Show that a triangular matrix is normal if and only if it is diagonal.

EXERCISE D.12 Let \mathbf{A} be a real orthogonal matrix. Show that all of its eigenvalues λ_i , which are generally complex, have unit modulus.

EXERCISE D.13 Let \mathbf{A} and \mathbf{T} be real $n \times n$ matrices, with \mathbf{T} nonsingular. Show that $\mathbf{T}^{-1} \mathbf{A} \mathbf{T}$ and \mathbf{A} have the same eigenvalues. (This is called a similarity transformation in linear algebra).

EXERCISE D.14 (Tough) Let \mathbf{A} be $m \times n$ and \mathbf{B} be $n \times m$. Show that the nonzero eigenvalues of $\mathbf{A}\mathbf{B}$ are the same as those of $\mathbf{B}\mathbf{A}$ (Kahan).

EXERCISE D.15 Let \mathbf{A} be real skew-symmetric, that is, $\mathbf{A} = -\mathbf{A}^T$. Show that all eigenvalues of \mathbf{A} are purely imaginary or zero.

EXERCISE D.16 Let \mathbf{A} be real skew-symmetric, that is, $\mathbf{A} = -\mathbf{A}^T$. Show that $\mathbf{U} = (\mathbf{I} + \mathbf{A})^{-1}(\mathbf{I} - \mathbf{A})$, called a Cayley transformation, is orthogonal.

EXERCISE D.17 Let \mathbf{P} be a real square matrix that satisfies

$$\mathbf{P}^2 = \mathbf{P}. \quad (\text{ED.5})$$

Such matrices are called *idempotent*, and also *orthogonal projectors*. Show that all eigenvalues of \mathbf{P} are either zero or one.

EXERCISE D.18 The necessary and sufficient condition for two square matrices to commute is that they have the same eigenvectors.

EXERCISE D.19 A matrix whose elements are equal on any line parallel to the main diagonal is called a Toeplitz matrix. (They arise in finite difference or finite element discretizations of regular one-dimensional grids.) Show that if \mathbf{T}_1 and \mathbf{T}_2 are any two Toeplitz matrices, they commute: $\mathbf{T}_1\mathbf{T}_2 = \mathbf{T}_2\mathbf{T}_1$. Hint: do a Fourier transform to show that the eigenvectors of any Toeplitz matrix are of the form $\{e^{i\omega nh}\}$; then apply the previous Exercise.

E

Linear Algebra: Eigenproblems

TABLE OF CONTENTS

	Page
§E.1. Introduction	E-3
§E.2. The Standard Algebraic Eigenproblem	E-3
§E.2.1. Characteristic Equation	E-3
§E.2.2. Eigenvectors	E-4
§E.2.3. Eigenpairs of Matrix Powers	E-4
§E.2.4. Multiplicities	E-5
§E.2.5. Similarity Transformation	E-5
§E.2.6. Diagonalization	E-5
§E.2.7. Other Reduced Forms	E-6
§E.3. Real Symmetric Real Matrices	E-7
§E.3.1. Spectral Properties	E-7
§E.3.2. Positivity	E-8
§E.4. Normal and Orthogonal Matrices	E-9
§E.5. Spectral Decomposition and Matrix Functions	E-9
§E.6. Matrix Deflation	E-10
§E.6.1. Hotelling Deflation	E-11
§E.6.2. Wielandt Deflation	E-11
§E.6.3. Matrix Order Reduction	E-12
§E.6.4. Nullspace Reduction	E-13
§E. Notes and Bibliography.	E-15
§E. Exercises	E-16

§E.1. Introduction

This Chapter discusses the conventional algebraic eigenvalue problem. It summarizes fundamental material of use in the main body of the book. It does not dwell on eigensolution methods for two reasons. First, those require specialized treatment beyond our intended scope. Second, efficient methods are implemented as black boxes in high level programming languages such as *Matlab*. Most properties and results will be stated without proof. Suggested textbooks and monographs devoted to the topic are listed under **Notes and Bibliography**.

§E.2. The Standard Algebraic Eigenproblem

Consider the linear equation system (D.16), namely $\mathbf{A}\mathbf{x} = \mathbf{y}$, in which \mathbf{A} is a square $n \times n$ matrix, while \mathbf{x} and \mathbf{y} are n -vectors. (Entries of \mathbf{A} , \mathbf{x} and \mathbf{y} may be real or complex numbers.) Suppose that the right-hand side vector \mathbf{y} is required to be a multiple λ of the solution vector \mathbf{x} :

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}, \quad (\text{E.1})$$

or, written in full,

$$\begin{array}{cccccccl} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & \lambda x_1 \\ a_{21}x_1 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & \lambda x_2 \\ \cdots & & \cdots & & \cdots & & \cdots & & \cdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & \lambda x_n. \end{array} \quad (\text{E.2})$$

These systems are trivially verified for $\mathbf{x} = \mathbf{0}$. The interesting solutions, called *nontrivial*, are those for which $\mathbf{x} \neq \mathbf{0}$. Pairing a nontrivial solution with a corresponding λ that satisfies (E.1) or (E.2) gives an *eigensolution pair*, or *eigenpair* for short.

Equations (E.1) or (E.2) are called an *eigensystem*. The determination of nontrivial eigenpairs $\{\lambda, \mathbf{x}\}$ is known as the standard (classical, conventional) *algebraic eigenproblem*. Properties linked to the eigenproblem are collectively called *spectral properties* of \mathbf{A} , a term that comes from original applications to optics. The set of eigenvalues of \mathbf{A} is called the *spectrum* of that matrix.

§E.2.1. Characteristic Equation

The eigensystem (E.1) can be rearranged into the homogeneous form

$$(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = \mathbf{0}. \quad (\text{E.3})$$

A nontrivial solution $\mathbf{x} \neq \mathbf{0}$ of (E.3) is possible if and only if the coefficient matrix $\mathbf{A} - \lambda\mathbf{I}$ is singular. Such a condition can be expressed as the vanishing of the determinant

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} a_{11} - \lambda & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} - \lambda \end{vmatrix} = 0. \quad (\text{E.4})$$

When this determinant is expanded, we obtain an algebraic polynomial equation in λ of degree n :

$$P(\lambda) = \lambda^n + \alpha_1\lambda^{n-1} + \cdots + \alpha_n = 0. \quad (\text{E.5})$$

This is known as the *characteristic equation* of the matrix \mathbf{A} . The left-hand side is called the *characteristic polynomial*. From the Fundamental Theorem of Algebra we know that a n^{th} degree polynomial has n (possibly complex) roots $\lambda_1, \lambda_2, \dots, \lambda_n$. Those n solutions, generically denoted by λ_i in which $i = 1, \dots, n$, are called the *eigenvalues*, *eigenroots* or *characteristic values*¹ of matrix \mathbf{A} . The characteristic polynomial of the transposed matrix is $\det(\mathbf{A}^T - \lambda \mathbf{I}) = \det(\mathbf{A} - \lambda \mathbf{I}) = P(\lambda)$. Consequently the eigenvalues of \mathbf{A} and \mathbf{A}^T are the same.

The coefficients of the characteristic polynomial of a real \mathbf{A} are real. However, its roots (the eigenvalues) may be complex. For real \mathbf{A} these will occur in complex conjugate pairs. If the matrix is symmetric real, however, the occurrence of real roots is guaranteed, as discussed in §E.3.

We will occasionally denote by $\lambda_1, \lambda_2, \dots, \lambda_p$, $p \leq n$, all the *distinct* eigenvalues of \mathbf{A} .

Eigenvalues may be ordered according to different conventions, which are stated in conjunction with specific examples. For future use, the diagonal matrix of eigenvalues will be denoted by $\mathbf{\Lambda} = \text{diag}[\lambda_i]$. This will be called the *eigenvalue matrix*.

§E.2.2. Eigenvectors

Associated to each eigenvalue λ_i there is a nonzero vector \mathbf{x}_i that satisfies (E.1) instantiated for $\lambda \rightarrow \lambda_i$:

$$\mathbf{A} \mathbf{x}_i = \lambda_i \mathbf{x}_i, \quad \mathbf{x}_i \neq 0. \quad (\text{E.6})$$

This \mathbf{x}_i is called a *right eigenvector* or *right characteristic vector*. This is often abbreviated to just *eigenvector*, in which case the “right” qualifier is tacitly understood. If \mathbf{A} is real and so is λ_i , \mathbf{x}_i has real entries. But if λ_i is complex, entries of \mathbf{x}_i will generally be complex.

The *left eigenvectors* of \mathbf{A} are the right eigenvectors of its transpose, and are denoted by \mathbf{y}_i :

$$\mathbf{A}^T \mathbf{y}_i = \lambda_i \mathbf{y}_i, \quad \mathbf{y}_i \neq 0. \quad (\text{E.7})$$

Note that the same index i can be used to pair \mathbf{x}_i and \mathbf{y}_i since the eigenvalues of \mathbf{A} and \mathbf{A}^T coalesce. Transposing both sides of (E.7) the definition may be restated as $\mathbf{y}_i^T \mathbf{A} = \lambda_i \mathbf{y}_i^T$.

There is an inherent magnitude indeterminacy in (E.6) since $\beta \mathbf{x}_i$, in which β is an arbitrary nonzero factor, is also a right eigenvector. Likewise for \mathbf{y}_i . Therefore eigenvectors are often *normalized* according to some convention. A popular one is unit dot product of left and right eigenvectors:

$$\mathbf{y}_i^T \mathbf{x}_i = \mathbf{x}_i^T \mathbf{y}_i = 1. \quad (\text{E.8})$$

A set of left and right eigenvectors $\{\mathbf{y}_i\}$ and $\{\mathbf{x}_j\}$, where i, j range over $1 \leq k \leq n$ integers, is said to be *biorthonormal* with respect to \mathbf{A} if they verify, with δ_{ij} denoting the Kronecker delta:

$$\mathbf{y}_i^T \mathbf{x}_j = \delta_{ij}, \quad \mathbf{y}_i^T \mathbf{A} \mathbf{x}_i = \lambda_i \delta_{ij}. \quad (\text{E.9})$$

¹ The terminology *characteristic value* is that found in publications before 1950; it derives from the works of Cauchy in French. *Eigenvalue* is a German-English hybrid. In German (and Dutch) “eigen” is used in the sense of special, inherent, or peculiar. It has become a popular qualifier attached (as prefix) to several items pertaining to the topic. For example: eigenvector, eigenproblem, eigensystem, eigenroot, eigenspace, eigenfunction, eigenfrequency, etc. *Characteristic equation* and *characteristic polynomial* are exceptions. “Eigen” hits the ear better than the long-winded “characteristic.”

§E.2.3. Eigenpairs of Matrix Powers

Premultiplying (E.1) by \mathbf{A} yields

$$\mathbf{A}^2 \mathbf{x}_i = \lambda_i \mathbf{A} \mathbf{x}_i = \lambda_i^2 \mathbf{x}_i. \quad (\text{E.10})$$

Thus the eigenvalues of \mathbf{A}^2 are λ_i^2 , whereas the eigenvectors do not change. Continuing the process, it is easily shown that the eigenvalues of \mathbf{A}^k , where k is a positive integer, are λ_i^k .

To investigate negative exponents, assume that \mathbf{A} is nonsingular so that \mathbf{A}^{-1} exists. Premultiplying (E.1) by \mathbf{A}^{-1} yields

$$\mathbf{I} \mathbf{x}_i = \lambda_i \mathbf{A}^{-1} \mathbf{x}_i, \quad \text{or} \quad \mathbf{A}^{-1} \mathbf{x}_i = \frac{1}{\lambda_i} \mathbf{x}_i. \quad (\text{E.11})$$

Consequently the eigenvalues of the inverse are obtained by taking the reciprocals of the original eigenvalues, whereas eigenvectors remain unchanged. The generalization to arbitrary negative integer exponents is immediate: the eigenvalues of \mathbf{A}^{-k} , where k is a positive integer, are λ_i^{-k} .

Extension of these results to non-integer powers (for example, the matrix square root) as well as to general matrix functions, requires the use of the spectral decomposition studied in §E.5.

§E.2.4. Multiplicities

The following terminology is introduced to succinctly state conditions for matrix reduction to diagonal form. They are largely taken from [629].

An eigenvalue λ is said to have *algebraic multiplicity* m_a if it is a root of multiplicity m_a of the characteristic polynomial. If $m_a = 1$, the eigenvalue is said to be *simple*; in which case it corresponds to an isolated root of $P(\lambda) = 0$. If $m_a > 1$ the eigenvalue is said to be *nonsimple* or *multiple*. Since $P(\lambda) = 0$ has n roots, we can state that the sum of algebraic multiplicities of all *distinct* eigenvalues must equal n . For example, the $n \times n$ identity matrix \mathbf{I} has one eigenvalue of value 1 that has algebraic multiplicity $m_a = n$.

An eigenvalue λ is said to have *geometric multiplicity* m_g if the maximum number of linearly independent eigenvectors associated with it is m_g . This is equal to the kernel dimension of $\mathbf{A} - \lambda \mathbf{I}$. Note that $1 \leq m_g \leq m_a$. If an eigenvalue is simple, that is $m_a = 1$, obviously $m_g = 1$.

A matrix is *derogatory* if the geometric multiplicity of at least one of its eigenvalues is greater than one. Obviously a matrix with only simple eigenvalues cannot be derogatory.

A multiple eigenvalue is *semi-simple* (a confusing name) if its algebraic and geometric multiplicities agree: $m_a = m_g$. A multiple eigenvalue that is not semi-simple is called *defective*.

§E.2.5. Similarity Transformation

Two square matrices, \mathbf{A} and \mathbf{B} , are said to be *similar* if there is a nonsingular matrix \mathbf{T} such that

$$\mathbf{B} = \mathbf{T}^{-1} \mathbf{A} \mathbf{T}. \quad (\text{E.12})$$

The mapping $\mathbf{A} \rightarrow \mathbf{B}$ is called a *similarity transformation*. The characteristic polynomials of the two matrices are identical. Consequently the two matrices have the same eigenvalues and each eigenvalue has the same algebraic multiplicity. An eigenvector \mathbf{x}_A of \mathbf{A} is transformed into the eigenvector $\mathbf{x}_B = \mathbf{T} \mathbf{x}_A$ of \mathbf{B} . It can be shown that geometric multiplicities are also preserved.

A key objective behind solving the eigenproblem of \mathbf{A} is to transform \mathbf{A} to a simpler form via (E.12). This process is called *reduction*. The simplest reduced form is the diagonal matrix for eigenvalues $\mathbf{A} = \mathbf{diag}(\lambda_i)$. This reduction, called *diagonalization*, is desirable but not always possible.

§E.2.6. Diagonalization

Diagonalization via (E.12) can be done if \mathbf{A} falls into one of two cases: (1) all eigenvalues are single, and (2) some eigenvalues are multiple but their geometric and algebraic multiplicities agree (i.e., they are not defective). Such matrices are called *diagonalizable*.

A diagonalizable matrix \mathbf{A} has a *complete* set of n linearly independent right eigenvectors $\{\mathbf{x}_i\}$ as well as left eigenvectors \mathbf{y}_i . Either set spans the Euclidean n -space and can be biorthogonalized so they verify (E.9) for $k = n$. To express this in compact matrix form, it is convenient to collect those sets into two $n \times n$ *eigenvector matrices* built by stacking eigenvectors as columns:

$$\mathbf{Y}^T = [\mathbf{y}_1^T \quad \mathbf{y}_2^T \quad \dots \quad \mathbf{y}_n^T], \quad \mathbf{X} = [\mathbf{x}_1 \quad \mathbf{x}_2 \quad \dots \quad \mathbf{x}_n]. \quad (\text{E.13})$$

These matrices verify

$$\mathbf{Y}^T \mathbf{A} \mathbf{X} = \mathbf{X}^T \mathbf{A}^T \mathbf{Y} = \mathbf{\Lambda} = \mathbf{diag}[\lambda_i], \quad \mathbf{Y}^T \mathbf{X} = \mathbf{X}^T \mathbf{Y} = \mathbf{I}. \quad (\text{E.14})$$

From the last relation, $\mathbf{X}^{-1} = \mathbf{Y}^T$ and $\mathbf{Y}^{-1} = \mathbf{X}^T$ whence

$$\mathbf{X}^{-1} \mathbf{A} \mathbf{X} = \mathbf{Y}^T \mathbf{A}^T \mathbf{Y}^{-T} = \mathbf{\Lambda} = \mathbf{diag}[\lambda_i]. \quad (\text{E.15})$$

The similarity transformations shown above are a particular case of (E.12). The spectral decompositions studied in §E.5 follow immediately from these relations.

On premultiplying the first and last terms of (E.15) by \mathbf{X} , the original eigenproblem can be recast as

$$\mathbf{A} \mathbf{X} = \mathbf{X} \mathbf{\Lambda}, \quad (\text{E.16})$$

This all-matrix form is often found in the literature.

§E.2.7. Other Reduced Forms

If the matrix \mathbf{A} is derogatory, diagonalization is not possible. Other reduced forms are used in such a case. They have the common goal of trying to simplify the original eigenproblem. Some possible choices are:

Jordan Block Form. An upper bidiagonal matrix with eigenvalues in its diagonal and ones or zeros in its superdiagonal. This reduction is always possible but may be numerically unstable.

Upper Triangular Form. This reduction is always possible and can be made stable. Eigenvalues occupy the diagonal of this form.

Companion Form. Closely related to the characteristic polynomial coefficients. Very compact but hardly used in numerical computation as it is prone to disastrous instability.

Intermediate reduced forms between the original matrix and the final reduced form include Hessenberg and tridiagonal matrices.

The sequence of operations to pass from the original to intermediate and final reduced forms may be studied in the specialized literature. See **Notes and Bibliography**.

Example E.1.

Consider the real unsymmetric matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 3 & 2 \\ -1 & 2 & 3 \end{bmatrix}. \quad (\text{E.17})$$

The characteristic polynomial is $P(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I}) = 8 - 14\lambda + 7\lambda^2 - \lambda^3$. This has the roots $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 4$, which are sorted in ascending order. The associated (unnormalized) right eigenvectors are $\mathbf{x}_1 = [-2 \ -2 \ 1]^T$, $\mathbf{x}_2 = [-2 \ 1 \ 1]^T$ and $\mathbf{x}_3 = [1 \ 1 \ 1]^T$. The associated (unnormalized) left eigenvectors are $\mathbf{y}_1 = [0 \ -1 \ 1]^T$, $\mathbf{y}_2 = [-1 \ 1 \ 0]^T$ and $\mathbf{y}_3 = [-3 \ 5 \ 4]^T$. These results can be used to construct the (diagonal) eigenvalue matrix and the (unnormalized) eigenvector matrices:

$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 4 \end{bmatrix}, \quad \mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3] = \begin{bmatrix} -2 & 3 & -1 \\ -2 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad \mathbf{Y} = [\mathbf{y}_1 \ \mathbf{y}_2 \ \mathbf{y}_3] = \begin{bmatrix} 0 & -1 & -3 \\ -1 & 1 & 5 \\ 1 & 0 & 4 \end{bmatrix}. \quad (\text{E.18})$$

We can now form the matrix products in (E.15):

$$\mathbf{Y}^T \mathbf{X} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -2 & 0 \\ 0 & 0 & 6 \end{bmatrix}, \quad \mathbf{Y}^T \mathbf{A} \mathbf{X} = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & 24 \end{bmatrix}. \quad (\text{E.19})$$

These products are not yet \mathbf{I} and $\mathbf{\Lambda}$ because the eigenvectors need to be appropriately normalized. To accomplish that, divide \mathbf{x}_1 , \mathbf{x}_2 , \mathbf{x}_3 , \mathbf{y}_1 , \mathbf{y}_2 , and \mathbf{y}_3 by $+\sqrt{3}$, $+\sqrt{2}$, $+\sqrt{6}$, $+\sqrt{3}$, $-\sqrt{2}$, and $+\sqrt{6}$, respectively. Note that the scale factors for \mathbf{x}_2 and \mathbf{y}_2 must have opposite signs to make the normalization work; that is, one of those vectors must be “flipped.” Then

$$\mathbf{X} = \begin{bmatrix} -2/\sqrt{3} & 3/\sqrt{2} & 1/\sqrt{6} \\ -2/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \\ 1/\sqrt{3} & 1/\sqrt{2} & 1/\sqrt{6} \end{bmatrix}, \quad \mathbf{Y} = \begin{bmatrix} 0 & 1/\sqrt{2} & -3/\sqrt{6} \\ -1/\sqrt{3} & -1/\sqrt{2} & 5/\sqrt{6} \\ 1/\sqrt{3} & 0 & 4/\sqrt{6} \end{bmatrix}. \quad (\text{E.20})$$

These do verify $\mathbf{X}^T \mathbf{X} = \mathbf{I}$ and $\mathbf{X}^T \mathbf{A} \mathbf{X} = \mathbf{\Lambda}$. It is easy to check that the biorthonormalized matrices (E.20) verify $\mathbf{X}^{-1} = \mathbf{Y}^T$ and $\mathbf{Y}^{-1} = \mathbf{X}^T$.

§E.3. Real Symmetric Real Matrices

Real symmetric matrices satisfying $\mathbf{A}^T = \mathbf{A}$ are of special importance in the finite element method, and thus deserve VIM treatment² in this Appendix.

§E.3.1. Spectral Properties

In linear algebra books dealing with the algebraic eigenproblem it is shown that all eigenvalues of a real symmetric matrix are real and semi-simple. Consequently symmetric matrices are diagonalizable and thus possess a complete set of linearly independent eigenvectors. Right and left eigenvectors coalesce: $\mathbf{y}_i = \mathbf{x}_i$, and such qualifiers may be omitted.

The eigenvectors \mathbf{x}_i can be *orthonormalized* so that

$$\mathbf{x}_i^T \mathbf{x}_j = \delta_{ij}, \quad \mathbf{x}_i^T \mathbf{A} \mathbf{x}_j = \lambda_i \delta_{ij}. \quad (\text{E.21})$$

² VIM stands for Very Important Matrix.

This is (E.9) specialized to $\mathbf{y}_i \rightarrow \mathbf{x}_i$. If eigenvectors are stacked as columns of a matrix \mathbf{X} , the foregoing orthonormality conditions can be compactly stated as

$$\mathbf{X}^T \mathbf{X} = \mathbf{I}, \quad \mathbf{X}^T \mathbf{A} \mathbf{X} = \mathbf{\Lambda} = \mathbf{diag}[\lambda_i]. \quad (\text{E.22})$$

Properties (a–c) also hold for the more general class of *Hermitian matrices*.³ A Hermitian matrix is equal to its own conjugate transpose. This extension has limited uses for structural analysis, but is important in other application areas.

§E.3.2. Positivity

Let \mathbf{A} be an $n \times n$ real symmetric matrix and \mathbf{v} an arbitrary real n -vector. \mathbf{A} is said to be *positive definite* (p.d.) if

$$\mathbf{v}^T \mathbf{A} \mathbf{v} > 0, \quad \forall \quad \mathbf{v} \neq \mathbf{0}. \quad (\text{E.23})$$

A positive definite matrix has rank n . This property can be quickly checked by computing the n eigenvalues λ_i of \mathbf{A} . If all $\lambda_i > 0$, \mathbf{A} is p.d. (This property follows on setting \mathbf{x} to the orthonormalized eigenvectors of \mathbf{A} , and using $\mathbf{v} \rightarrow \mathbf{x}_i$ to conclude that $\lambda_i = \mathbf{x}_i^T \mathbf{A} \mathbf{x}_i > 0$.)

\mathbf{A} is said to be *nonnegative*⁴ if zero equality is allowed in (E.23)

$$\mathbf{v}^T \mathbf{A} \mathbf{v} \geq 0, \quad \forall \quad \mathbf{v} \neq \mathbf{0}. \quad (\text{E.24})$$

A p.d. matrix is also nonnegative but the converse is not necessarily true. This property can be quickly checked by computing the n eigenvalues λ_i of \mathbf{A} . If $n - r$ eigenvalues are zero and the rest are positive, \mathbf{A} is nonnegative with rank r .

An $n \times n$ symmetric real matrix \mathbf{A} that has at least one negative eigenvalue is called *indefinite*.

The extension of these definitions to general matrices is discussed in specialized texts.

For the concept of *inertia* of Hermitian matrices, which include symmetric matrices as special case, see Appendix P. This concept is important in dynamics.

Example E.2.

Consider the real symmetric matrix

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 1 \\ -1 & 1 & 4 \end{bmatrix}. \quad (\text{E.25})$$

The characteristic polynomial is $P(\lambda) = \det(\mathbf{A} - \lambda \mathbf{I}) = 10 - 17\lambda + 8\lambda^2 - \lambda^3$. This has the roots $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 5$, which are sorted in ascending order. The associated (unnormalized) eigenvectors are $\mathbf{x}_1 = [1 \ 1 \ 0]^T$, $\mathbf{x}_2 = [1 \ -1 \ 1]^T$ and $\mathbf{x}_3 = [-1 \ 1 \ 2]^T$. These eigenpairs can be used to construct the (diagonal) eigenvalue matrix and the (unnormalized) eigenvector matrix:

$$\mathbf{\Lambda} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix}, \quad \mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3] = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix}. \quad (\text{E.26})$$

³ Also called *self-adjoint matrices*.

⁴ A property called *positive semidefinite* by some authors.

We can now form the matrix combinations in (E.22):

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 6 \end{bmatrix}, \quad \mathbf{X}^T \mathbf{A} \mathbf{X} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6 & 0 \\ 0 & 0 & 30 \end{bmatrix}. \quad (\text{E.27})$$

These are not yet \mathbf{I} and $\mathbf{\Lambda}$ because eigenvectors need to be normalized to unit length. To accomplish that, divide \mathbf{x}_1 , \mathbf{x}_2 and \mathbf{x}_3 by $+\sqrt{2}$, $+\sqrt{3}$ and $+\sqrt{6}$, respectively. Then

$$\mathbf{X} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{3} & -1/\sqrt{6} \\ 1/\sqrt{2} & -1/\sqrt{3} & 1/\sqrt{6} \\ 0 & 1/\sqrt{3} & 2/\sqrt{6} \end{bmatrix}, \quad \mathbf{X}^T \mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \mathbf{I}, \quad \mathbf{X}^T \mathbf{A} \mathbf{X} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 5 \end{bmatrix} = \mathbf{\Lambda}. \quad (\text{E.28})$$

§E.4. Normal and Orthogonal Matrices

Let \mathbf{A} be an $n \times n$ real square matrix. This matrix is called *normal* if

$$\mathbf{A}^T \mathbf{A} = \mathbf{A} \mathbf{A}^T \quad (\text{E.29})$$

A normal matrix is called *orthogonal* if

$$\mathbf{A}^T \mathbf{A} = \mathbf{A} \mathbf{A}^T = \mathbf{I} \quad \text{or} \quad \mathbf{A}^T = \mathbf{A}^{-1} \quad (\text{E.30})$$

All eigenvalues of an orthogonal matrix have modulus one, and the matrix has rank n .

The generalization of the orthogonality property to complex matrices, for which transposition is replaced by conjugation, leads to *unitary* matrices. These are not required, however, for the material covered in the text.

§E.5. Spectral Decomposition and Matrix Functions

Let \mathbf{A} be a $n \times n$ symmetric real matrix. As noted in §E.3, it possesses a complete set of real eigenvalues λ_i and corresponding eigenvectors \mathbf{x}_i , for $i = 1, \dots, n$. The latter are assumed to be orthonormalized so that the conditions (E.21) hold. The *spectral decomposition* of \mathbf{A} is the following expansion in terms of rank-one matrices:

$$\mathbf{A} = \sum_i^n \lambda_i \mathbf{x}_i \mathbf{x}_i^T. \quad (\text{E.31})$$

This can be easily proven by postmultiplying both sides of (E.31) by \mathbf{x}_j and using (E.21):

$$\mathbf{A} \mathbf{x}_j = \sum_i^n \lambda_i \mathbf{x}_i \mathbf{x}_i^T \mathbf{x}_j = \sum_i^n \lambda_i \mathbf{x}_i \delta_{ij} = \lambda_j \mathbf{x}_j. \quad (\text{E.32})$$

Assume next that \mathbf{A} is nonsingular so all $\lambda_i \neq 0$. Since the eigenvalues of the inverse are the reciprocal of the original eigenvalues as per (E.11), the spectral decomposition of the inverse is

$$\mathbf{A}^{-1} = \sum_i^n \frac{1}{\lambda_i} \mathbf{x}_i \mathbf{x}_i^T. \quad (\text{E.33})$$

This is valid for more general matrix powers:

$$\mathbf{A}^m \stackrel{\text{def}}{=} \sum_i^n \lambda_i^m \mathbf{x}_i \mathbf{x}_i^T. \quad (\text{E.34})$$

in which the exponent m may be arbitrary as long as all λ_i^m exist. This represents a generalization of the integer powers studied in §E.2.3. The expression (E.34) can be in fact extended to other matrix functions as a definition. For example, the matrix logarithm may be defined as

$$\log(\mathbf{A}) \stackrel{\text{def}}{=} \sum_i^n \log(\lambda_i) \mathbf{x}_i \mathbf{x}_i^T, \quad \text{iff } \lambda_i \neq 0 \quad \forall i, \quad (\text{E.35})$$

whence $\log(\mathbf{A})$ is also symmetric real if all $\lambda_i > 0$, that is, \mathbf{A} is p.d. More generally, if $f(\cdot)$ is a scalar function,

$$f(\mathbf{A}) \stackrel{\text{def}}{=} \sum_i^n f(\lambda_i) \mathbf{x}_i \mathbf{x}_i^T, \quad (\text{E.36})$$

in which function applications to each λ_i must make sense. For the matrix exponential function $\exp(\mathbf{A})$, no restrictions apply since the exponential of any real number is well defined.

The concept can be extended to a diagonalizable $n \times n$ matrix \mathbf{A} that has eigenvalues λ_i , with corresponding right eigenvectors \mathbf{x}_i and left eigenvectors \mathbf{y}_i . The latter are assumed to be biorthonormalized as per (E.9). For notational simplicity all eigenvalues and eigenvectors are assumed to be real. The spectral decomposition is

$$\mathbf{A} = \sum_i^n \lambda_i \mathbf{x}_i \mathbf{y}_i^T. \quad (\text{E.37})$$

The proof is identical to that of (E.32). If \mathbf{A} and/or its eigenvectors contain complex entries, the transpose operator is replaced by conjugate transpose.

These decompositions play a fundamental role in the investigation of spectral properties, as well as the definition and study of arbitrary matrix functions.

Example E.3.

Consider the real symmetric matrix (E.27), with eigenvalues (in ascending order) $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 5$, and the orthonormalized eigenvectors listed in (E.28). Its spectral decomposition is $\lambda_1 \mathbf{x}_1 \mathbf{x}_1^T + \lambda_2 \mathbf{x}_2 \mathbf{x}_2^T + \lambda_3 \mathbf{x}_3 \mathbf{x}_3^T$, which numerically works out to be

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 1 \\ -1 & 1 & 4 \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} 1 & -1 & 1 \\ -1 & 1 & -1 \\ 1 & -1 & 1 \end{bmatrix} + \frac{1}{6} \begin{bmatrix} 1 & -1 & -2 \\ -1 & 1 & 2 \\ -2 & 2 & 4 \end{bmatrix}. \quad (\text{E.38})$$

(The number in front of the matrices is just a scale factor, not the eigenvalue.)

Example E.4.

Consider the real unsymmetric matrix (E.17), with eigenvalues (in ascending order) $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 4$, and the orthonormalized eigenvectors listed in (E.20). Its spectral decomposition is $\lambda_1 \mathbf{x}_1 \mathbf{y}_1^T + \lambda_2 \mathbf{x}_2 \mathbf{y}_2^T + \lambda_3 \mathbf{x}_3 \mathbf{y}_3^T$, which numerically works out to be

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 2 \\ -1 & 3 & 2 \\ -1 & 2 & 3 \end{bmatrix} = \frac{1}{3} \begin{bmatrix} 0 & 2 & -2 \\ 0 & 2 & -2 \\ 0 & -1 & 1 \end{bmatrix} + \begin{bmatrix} 3 & -3 & 0 \\ 1 & -1 & 0 \\ 1 & -1 & 0 \end{bmatrix} + \frac{2}{3} \begin{bmatrix} -3 & 5 & 4 \\ -3 & 5 & 4 \\ -3 & 5 & 4 \end{bmatrix}. \quad (\text{E.39})$$

§E.6. Matrix Deflation

In eigenproblem solution processes it often happens that one or more eigenpairs are either known in advance, or are obtained before the others.⁵ A procedure by which those eigenpairs are “deactivated” in some manner is called *matrix deflation* or simply *deflation*. Two deflation schemes are commonly used in practice. They are associated with the names Hotelling and Wielandt, respectively.

§E.6.1. Hotelling Deflation

Let \mathbf{A} be a diagonalizable $n \times n$ real matrix with the spectral decomposition (E.37), in which the eigenvectors satisfy the biorthonormality condition (E.9). Suppose that eigenvalue λ_i , with associated right eigenvector \mathbf{x}_i and left eigenvector \mathbf{y}_i , are known. They are assumed to be real for notational simplicity. A *deflated matrix* $\tilde{\mathbf{A}}_i$ is obtained by subtracting a rank one correction:

$$\tilde{\mathbf{A}}_i = \mathbf{A} - \lambda_i \mathbf{x}_i \mathbf{y}_i^T. \quad (\text{E.40})$$

Obviously this has a spectral decomposition identical to (E.37), except that λ_i is replaced by zero. For example if $i = 1$, the deflated matrix $\tilde{\mathbf{A}}_1$ has eigenvalues $0, \lambda_2, \dots, \lambda_n$, and the same right and left eigenvectors. If \mathbf{A} is symmetric, $\mathbf{y}_i = \mathbf{x}_i$ and $\tilde{\mathbf{A}}_i$ is also symmetric. This deflation scheme produces nothing new if λ_i is zero.

If \mathbf{A} and/or its eigenvectors contain complex entries, the transpose operator is replaced by the conjugate transpose.

Example E.5.

Consider the real symmetric matrix (E.27), with eigenvalues (in ascending order) $\lambda_1 = 1, \lambda_2 = 2$, and $\lambda_3 = 5$, and the orthonormalized eigenvectors listed in (E.28). Deflating λ_1, λ_2 and λ_3 in turn through (E.40) gives

$$\tilde{\mathbf{A}}_1 = \frac{1}{6} \begin{bmatrix} 7 & -1 & 4 \\ -1 & 7 & -4 \\ 4 & -4 & 4 \end{bmatrix}, \quad \tilde{\mathbf{A}}_2 = \frac{1}{3} \begin{bmatrix} 4 & -1 & -5 \\ -1 & 4 & 5 \\ -5 & 5 & 10 \end{bmatrix}, \quad \tilde{\mathbf{A}}_3 = \frac{1}{2} \begin{bmatrix} 3 & -3 & -2 \\ -3 & 3 & 2 \\ -2 & 2 & 8 \end{bmatrix}. \quad (\text{E.41})$$

The eigenvalues of these matrices are $\{0, 1, 2\}$, $\{0, 1, 5\}$ and $\{0, 2, 5\}$, respectively. The eigenvectors remain the same.

Example E.6.

Consider the real unsymmetric matrix (E.17), with eigenvalues (in ascending order) $\lambda_1 = 1, \lambda_2 = 2$, and $\lambda_3 = 4$, and the biorthonormalized eigenvectors listed in (E.20). Deflating λ_1, λ_2 and λ_3 in turn through (E.40) gives

$$\tilde{\mathbf{A}}_1 = \frac{1}{3} \begin{bmatrix} 3 & 1 & 8 \\ -3 & 7 & 8 \\ -3 & 7 & 8 \end{bmatrix}, \quad \tilde{\mathbf{A}}_2 = \begin{bmatrix} -2 & 4 & 2 \\ -2 & 4 & 2 \\ -2 & 3 & 3 \end{bmatrix}, \quad \tilde{\mathbf{A}}_3 = \frac{1}{2} \begin{bmatrix} 9 & -7 & -2 \\ 3 & -1 & -2 \\ 3 & -4 & 1 \end{bmatrix}. \quad (\text{E.42})$$

The eigenvalues of these matrices are $\{0, 2, 4\}$, $\{0, 1, 4\}$ and $\{0, 1, 2\}$, respectively. The right and left eigenvectors remain the same.

⁵ The second scenario is the case in the classical power iteration method, which was important for “large” eigenproblems before 1960. The largest eigenvalue in modulus, also known as the *dominant* one, is obtained along with the associated eigenvector(s). The matrix is then deflated so that the next-in-modulus (subdominant) eigenvalue becomes the dominant one.

§E.6.2. Wielandt Deflation

This deflation scheme is more general than the previous one, which is included as special case. Let \mathbf{A} be a diagonalizable $n \times n$ real matrix, with eigenvectors that satisfy the biorthonormality condition (E.9). Again suppose that eigenvalue λ_i , with associated right eigenvector \mathbf{x}_i and left eigenvector \mathbf{y}_i , are known. All are assumed to be real for notational simplicity. Let \mathbf{w}_i be a column n -vector satisfying $\mathbf{w}_i^T \mathbf{x}_i = 1$, and σ a real scalar called the *spectral shift* or simply *shift*. The Wielandt-deflated matrix is

$$\tilde{\mathbf{A}}_i = \mathbf{A} - \sigma \mathbf{x}_i \mathbf{w}_i^T. \quad (\text{E.43})$$

Premultiplying both sides by \mathbf{y}_j^T yields

$$\mathbf{y}_j^T \tilde{\mathbf{A}}_i = \mathbf{y}_j^T \mathbf{A} - \sigma \delta_{ij} \mathbf{w}_i^T \quad \text{or} \quad \tilde{\mathbf{A}}_i^T \mathbf{y}_j = \mathbf{A}^T \mathbf{y}_j = \lambda_j \mathbf{y}_j, \quad \text{if } j \neq i. \quad (\text{E.44})$$

Consequently the eigenvalues λ_j for $j \neq i$, as well as the corresponding left eigenvectors \mathbf{y}_j , are preserved by the deflation process. What happens for $i = j$? Postmultiplying by \mathbf{x}_i yields

$$\tilde{\mathbf{A}}_i \mathbf{x}_i = \mathbf{A} \mathbf{x}_i - \sigma \mathbf{x}_i, \quad \text{or} \quad \tilde{\mathbf{A}}_i \mathbf{x}_i - (\lambda_i - \sigma) \mathbf{x}_i = 0. \quad (\text{E.45})$$

This shows that eigenvector \mathbf{x}_i is preserved whereas eigenvalue λ_i is shifted by $-\sigma$. The right eigenvectors \mathbf{x}_j for $i \neq j$ are not generally preserved; they become linear combinations of \mathbf{x}_i and \mathbf{x}_j . Following [629, p. 118], assume $\tilde{\mathbf{x}}_j = \mathbf{x}_j - \gamma_j \mathbf{x}_i$. Then

$$\tilde{\mathbf{A}}_i \tilde{\mathbf{x}}_j = \lambda_j \mathbf{x}_j - (\gamma_j \lambda_i + \sigma \mathbf{w}_i^T \mathbf{x}_j - \sigma \gamma_j) \mathbf{x}_i. \quad (\text{E.46})$$

This shows that $\tilde{\mathbf{x}}_j$ is an eigenvector of $\tilde{\mathbf{A}}_i$ if

$$\gamma_i = \frac{\mathbf{w}_i^T \mathbf{x}_j}{1 - (\lambda_i - \lambda_j)/\sigma}, \quad \text{for } j \neq i. \quad (\text{E.47})$$

This is undefined if the denominator vanishes: $\lambda_i - \lambda_j = \sigma$, which happens if $\lambda_i - \sigma$ becomes a multiple eigenvalue of the deflated matrix.

The process shown so far only requires knowledge of the right eigenvector \mathbf{x}_i . This *need not be normalized*, since that is taken care of by the unit dot product with \mathbf{w}_i . If $\sigma = \lambda_i$ and $\mathbf{w}_i = c \mathbf{y}_i$, in which c is determined by $\mathbf{w}_i^T \mathbf{x}_i = 1$, the Hotelling deflation (E.40) is recovered. This choice preserves *all* eigenvectors while shifting λ_i to zero, and retains symmetry if \mathbf{A} is real symmetric.

§E.6.3. Matrix Order Reduction

The freedom to choose \mathbf{w} in the Wielandt deflation (E.43) can be exploited to combine deflation with matrix order reduction. As in §E.6.2, assume that eigenvalue λ_i as well as the right eigenvector \mathbf{x}_i are known, and are both assumed real. Denote the j^{th} entry of \mathbf{x}_i by x_{ij} and assume $x_{ij} \neq 0$. Pick $\mathbf{w}_i = \mathbf{e}_j^T \mathbf{A}$, in which \mathbf{e}_j is the n -vector with all zero entries except the j^{th} one, which is one. Consequently the chosen \mathbf{w}_i is \mathbf{a}_j , the j^{th} row of \mathbf{A} . To identify that row, the deflated matrix will be denoted by $\tilde{\mathbf{A}}_{ij}$:

$$\tilde{\mathbf{A}}_{ij} = \mathbf{A} - S \mathbf{x}_i \mathbf{a}_j = (\mathbf{I} - S \mathbf{x}_i \mathbf{e}_j^T) \mathbf{A}, \quad (\text{E.48})$$

in which $S = 1/x_{ij}$. By inspection the j^{th} row of $\tilde{\mathbf{A}}_{ij}$ is zero. All eigenvectors $\tilde{\mathbf{x}}_j$ of $\tilde{\mathbf{A}}_{ij}$ with $i \neq j$ will have their j^{th} entry equal to zero. Consequently that row and the corresponding column can be deleted, which reduces its order by one. The order-reduced $(n-1) \times (n-1)$ matrix is denoted by $\hat{\mathbf{A}}_{ij}$. The operation is called *deflate-and-order-reduce*. It is better explained by an example.

Example E.7.

Consider the real symmetric matrix (E.27), with eigenvalues (in ascending order) $\lambda_1 = 1$, $\lambda_2 = 2$, and $\lambda_3 = 5$. The matrix and (unnormalized) eigenvector matrix are reproduced for convenience:

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & 1 \\ -1 & 1 & 4 \end{bmatrix}, \quad \mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3] = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix} = \mathbf{Y}. \quad (\text{E.49})$$

To deflate-and-reduce λ_1 , we pick eigenvector \mathbf{x}_1 , the first entry of which is $x_{11} = 1$, and the first row of \mathbf{A} : $\mathbf{a}_1 = [1 \ -1 \ -1]$, to get

$$\tilde{\mathbf{A}}_{11} = \mathbf{A} - \frac{1}{x_{11}} \mathbf{x}_1 \mathbf{a}_1 = \begin{bmatrix} 0 & 0 & 0 \\ -3 & 3 & 2 \\ -1 & 1 & 4 \end{bmatrix} \Rightarrow \hat{\mathbf{A}}_{11} = \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}. \quad (\text{E.50})$$

The eigenvalues of $\tilde{\mathbf{A}}_{11}$ and $\hat{\mathbf{A}}_{11}$ are $\{0, 2, 5\}$ and $\{2, 5\}$, respectively. The (unnormalized) eigenvector matrices are

$$\tilde{\mathbf{X}}_{11} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & -1 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \quad \tilde{\mathbf{Y}}_{11} = \begin{bmatrix} 1 & 1 & -1 \\ 1 & -1 & 1 \\ 0 & 1 & 2 \end{bmatrix}, \quad \hat{\mathbf{X}}_{11} = \begin{bmatrix} -2 & 1 \\ 1 & 1 \end{bmatrix}, \quad \hat{\mathbf{Y}}_{11} = \begin{bmatrix} -1 & 1 \\ 1 & 2 \end{bmatrix}. \quad (\text{E.51})$$

Since the deflated matrices (E.50) are unsymmetric, the left and right eigenvalues no longer coalesce. Note that left eigenvectors except \mathbf{y}_1 are preserved, whereas right eigenvectors except \mathbf{x}_1 change.

What happens if λ_1 is deflated via the second row? We get

$$\tilde{\mathbf{A}}_{12} = \mathbf{A} - \frac{1}{x_{12}} \mathbf{x}_1 \mathbf{a}_2 = \begin{bmatrix} 3 & -3 & -2 \\ 0 & 0 & 0 \\ -1 & 1 & 4 \end{bmatrix} \Rightarrow \hat{\mathbf{A}}_{12} = \begin{bmatrix} 3 & -2 \\ -1 & 4 \end{bmatrix}. \quad (\text{E.52})$$

Deflated and reduced matrices are different from those in (E.50) although they have the same eigenvalues.

Deflation via the third row is not possible since the third entry of \mathbf{x}_1 is zero.

The deflate-and-order-reduce process can be obviously continued by renaming $\hat{\mathbf{A}}_{11}$ or $\hat{\mathbf{A}}_{12}$ to \mathbf{A} , and shifting one of its eigenvalues, say $\lambda_1 = 2$, to zero. The reduced matrix will be 1×1 with eigenvalue 5, so the resulting $\hat{\mathbf{A}}_{1j}$ ($j = 1$ or 2) will be $[5]$.

§E.6.4. Nullspace Reduction

If several eigenpairs are known, they can be simultaneously deactivated instead of one by one. The procedure is known as *block deflation*.

The Hotelling deflation (E.40) is trivially extended to the block case: just subtract several rank-one terms. Eigenvectors are not altered. On the other hand, extending the general Wielandt deflation (E.43) is not trivial because each single eigenpair deflation changes the right eigenvectors, and generally no particular computational advantage accrues from doing blocks. But there is a special case where it deserves attention: deflate-and-order-reduce a null space.

Suppose the $n \times n$ diagonalizable real matrix \mathbf{A} is $m \geq 1$ times singular. That is, its kernel has dimension m . Reordering the characteristic roots as necessary, we can state that $\lambda_1 = \lambda_2 = \dots = \lambda_m = 0$ is a null eigenvalue of algebraic multiplicity m . The associated eigenspace matrix of right eigenvectors is denoted by

$$\mathbf{X}_m = [\mathbf{x}_1 \ \mathbf{x}_2 \ \dots \ \mathbf{x}_m]. \quad (\text{E.53})$$

(These eigenvectors are not necessarily orthonormal, but must be linearly independent.) For brevity \mathbf{X}_m will be referred to as the *nullspace basis*, or simply *nullspace*, of \mathbf{A} . From a computational standpoint it is important to realize that \mathbf{X}_m can be obtained without solving the eigenproblem (E.1). Gaussian row reduction is sufficient⁶ For example, the built-in function `NullSpace` of *Mathematica* returns that matrix.

Suppose that we want to get rid of both multiple singularity and nullspace by block deflate-and-reduce to end up with a $(n-m) \times (n-m)$ matrix with the same nonzero eigenvalues. To do this, pick m rows to zero out, and stack the corresponding rows of \mathbf{A} to form the $m \times n$ matrix \mathbf{A}_m . Then

$$\tilde{\mathbf{A}}_m = \mathbf{A} - \mathbf{X}_m \mathbf{S} \mathbf{U} \mathbf{A}_m. \quad (\text{E.54})$$

Here \mathbf{S} is a $m \times m$ diagonal scaling matrix that contains the reciprocals of the eigenvector entries corresponding to the rows to be cleared, while \mathbf{U} is a $m \times m$ unit upper triangular matrix, the unknown entries of which are chosen so that targetted rows of the deflated matrix vanish.⁷ It is of course possible to combine \mathbf{S} and \mathbf{U} into a single upper triangular matrix $\mathbf{U}_s = \mathbf{S} \mathbf{U}$. This contains eigenvector entry reciprocals in its diagonal and thus it is not necessarily unit upper triangular.

If $m = 1$, \mathbf{S} and \mathbf{U} collapse to 1×1 matrices containing $1/x_{ij}$ and 1, respectively, and (E.54) reduces to the single vector case (E.48).

The procedure is illustrated by a symbolic example.

Example E.8. The following 3×3 symbolic symmetric matrix of rank 1 was discovered through *Mathematica*. It depends on three real scalar parameters: b , c , and d , which must be nonzero:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} \frac{bc}{d} & b & c \\ b & \frac{bd}{c} & d \\ c & d & \frac{cd}{b} \end{bmatrix}, \quad (\text{E.55})$$

in which \mathbf{a}_i denotes the i^{th} row of \mathbf{A} . The eigenvalues and (unnormalized) eigenvectors are

$$\lambda_1 = \lambda_2 = 0, \quad \lambda_3 = \frac{bc}{d} + \frac{bd}{c} + \frac{cd}{b}, \quad \mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ \mathbf{x}_3] = \begin{bmatrix} -d/b & -d/c & b/d \\ 0 & 1 & b/c \\ 1 & 0 & 1 \end{bmatrix}. \quad (\text{E.56})$$

The nullspace is spanned by $[\mathbf{x}_1 \ \mathbf{x}_2]$. To block deflate it so that the first two rows of the deflated matrix vanish we apply (E.54):

$$\begin{aligned} \tilde{\mathbf{A}}_2 &= \mathbf{A} - [\mathbf{x}_1 \ \mathbf{x}_2] \mathbf{S} \mathbf{U} [\mathbf{a}_1 \ \mathbf{a}_2] \\ &= \mathbf{A} - \begin{bmatrix} -d/b & -d/c \\ 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1/1 & 0 \\ 0 & 1/1 \end{bmatrix} \begin{bmatrix} 1 & u_{12} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \frac{bc}{d} & b & c \\ b & \frac{bd}{c} & d \end{bmatrix} \\ &= \begin{bmatrix} c + \frac{bc}{d} + \frac{bd}{c} + d u_{12} & b + d + \frac{bd^2}{c^2} + \frac{d^2 u_{12}}{c} & c + \frac{cd}{b} + \frac{d^2}{c} + \frac{d^2 u_{12}}{b} \\ 0 & 0 & 0 \\ c - \frac{bc}{d} - b u_{12} & d - \frac{b(c + d u_{12})}{c} & c(\frac{d}{b} - 1) - d u_{12} \end{bmatrix} \end{aligned} \quad (\text{E.57})$$

⁶ At least in exact arithmetic; in floating-point work the detection of rank is an ill-posed problem.

⁷ The effect of \mathbf{U} is essentially to appropriately scale and combine the original eigenvectors stacked in \mathbf{X}_m so they become equivalent to the modified right eigenvectors in the one-vector-at-a-time deflation.

A simple calculation shows that taking $u_{12} = -c/d - b c/d^2 - b/(c d)$ makes the first row of $\tilde{\mathbf{A}}_2$ null, reducing it to

$$\tilde{\mathbf{A}}_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ c + b^2 \left(\frac{1}{c} + \frac{c}{d^2} \right) & d + b^2 \left(\frac{1}{d} + \frac{d}{c^2} \right) & \frac{bc}{d} + \frac{bd}{c} + \frac{cd}{b} \end{bmatrix} \quad (\text{E.58})$$

Removing the first two rows and columns reduces this to the 1×1 matrix $[bc/d + bd/c + cd/b] = [\lambda_3]$.

Notes and Bibliography

The best overall coverage of the algebraic eigenproblem is still the monograph of Wilkinson [761], although some of the computational methods described therein are dated; especially for sparse matrices. For symmetric matrices the most elegant exposition is Parlett's textbook [523], reprinted by SIAM.

Hotelling deflation was proposed in 1933 while its generalization: Wielandt deflation, appeared in 1944; see [359,761] for historical references. Block Wielandt deflation was introduced by Brauer [102] in 1952. Strangely, the nullspace deflation procedure described in §E.6.4 is not found in the literature despite its usefulness.

In FEM models the solution of very large eigensystems (thousands or millions of DOF) occurs frequently in applications such as structural analysis for vibration and stability. The matrices involved: mass, material stiffness, and geometric stiffness, are typically very sparse, meaning that the proportion of nonzero entries is very small. Specialized methods for efficient handling of such problems have been steadily developed since the 1960s. A comprehensive coverage from the standpoint of underlying theory is provided in the textbook by Saad [629], which has been republished by SIAM in a revised and updated 2011 edition. The book by Stewart [669] focuses more on theory, especially perturbation results.

The most exhaustive study of matrix functions, combining theory and computer implementation, is the textbook by Higham [348], which has a comprehensive reference list until 2007.

Exercises for Appendix E
Linear Algebra: Eigenproblems

EXERCISE E.1 What are the eigenvalues and eigenvectors of a diagonal matrix?

EXERCISE E.2 Matrix \mathbf{A} is changed to $\mathbf{A} - \sigma \mathbf{I}$, in which σ is a scalar called the *spectral shift* and \mathbf{I} is the identity matrix. Explain what happens to the eigenvalues and eigenvectors of \mathbf{A} .

EXERCISE E.3 Show that the eigenvalues of a real symmetric square matrix are real, and that all eigenvector entries are real.

EXERCISE E.4 Let the n real eigenvalues λ_i of a real $n \times n$ symmetric matrix \mathbf{A} be classified into two subsets: r eigenvalues are nonzero whereas $n - r$ are zero. Show that \mathbf{A} has rank r .

EXERCISE E.5 Show that the characteristic polynomials of \mathbf{A} and $\mathbf{T}^{-1} \mathbf{A} \mathbf{T}$, in which \mathbf{T} is a nonsingular transformation matrix, are identical. Conclude that the similarity transformation (E.12) preserves eigenvalues and their algebraic multiplicities.

EXERCISE E.6 Let \mathbf{Q} be a real orthogonal matrix: $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$. Show that all of its eigenvalues λ_i , which are generally complex, have unit modulus.

EXERCISE E.7 Let \mathbf{A} be real skew-symmetric, that is, $\mathbf{A} = -\mathbf{A}^T$. Show that all eigenvalues of \mathbf{A} are purely imaginary or zero.

EXERCISE E.8 Let \mathbf{P} be a real square matrix that satisfies

$$\mathbf{P}^2 = \mathbf{P}. \quad (\text{EE.1})$$

Such matrices are called *idempotent*, and also *orthogonal projectors*. Show that all eigenvalues of \mathbf{P} are either zero or one. *Hint:* (E.10).

EXERCISE E.9 Two conforming diagonalizable square matrices \mathbf{A} and \mathbf{B} are said to commute if $\mathbf{AB} = \mathbf{BA}$. Show that necessary and sufficient conditions for this to happen is that \mathbf{A} and \mathbf{B} have identical eigenvectors. *Hint:* for sufficiency use $\mathbf{A} = \mathbf{X} \mathbf{\Lambda}_A \mathbf{X}^{-1}$ and $\mathbf{B} = \mathbf{X} \mathbf{\Lambda}_B \mathbf{X}^{-1}$.

EXERCISE E.10

(Advanced) A matrix whose elements are equal on any line parallel to the main diagonal is called a Toeplitz matrix. (They arise in finite difference or finite element discretizations of regular one-dimensional grids.) Show that if \mathbf{T}_1 and \mathbf{T}_2 are any two Toeplitz matrices, they commute: $\mathbf{T}_1 \mathbf{T}_2 = \mathbf{T}_2 \mathbf{T}_1$. *Hint:* do a Fourier transform to show that the eigenvectors of any Toeplitz matrix are of the form $\{e^{i\omega n h}\}$; then apply the results of the previous Exercise.

EXERCISE E.11 Let \mathbf{A} and \mathbf{B} be two diagonalizable square matrices of equal order that commute. Show that the eigenvalues of \mathbf{AB} are the product of the eigenvalues of \mathbf{A} and \mathbf{B} (Frobenius).

F

Matrix Calculus

TABLE OF CONTENTS

		Page
§F.1.	Introduction	F-3
§F.2.	The Derivatives of Vector Functions	F-3
§F.2.1.	Derivative of Vector with Respect to Vector	F-3
§F.2.2.	Derivative of a Scalar with Respect to Vector	F-3
§F.2.3.	Derivative of Vector with Respect to Scalar	F-3
§F.2.4.	Jacobian of a Variable Transformation	F-4
§F.3.	The Chain Rule for Vector Functions	F-5
§F.4.	The Derivative of Scalar Functions of a Matrix	F-6
§F.4.1.	Functions of a Matrix Determinant	F-7
§F.5.	The Matrix Differential	F-8

§F.1. Introduction

In this Appendix we collect some useful formulas of matrix calculus that often appear in finite element derivations.

§F.2. The Derivatives of Vector Functions

Let \mathbf{x} and \mathbf{y} be vectors of orders n and m respectively:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \quad (\text{F.1})$$

where each component y_i may be a function of all the x_j , a fact represented by saying that \mathbf{y} is a function of \mathbf{x} , or

$$\mathbf{y} = \mathbf{y}(\mathbf{x}). \quad (\text{F.2})$$

If $n = 1$, \mathbf{x} reduces to a scalar, which we call x . If $m = 1$, \mathbf{y} reduces to a scalar, which we call y . Various applications are studied in the following subsections.

§F.2.1. Derivative of Vector with Respect to Vector

The derivative of the vector \mathbf{y} with respect to vector \mathbf{x} is the $n \times m$ matrix

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \quad (\text{F.3})$$

§F.2.2. Derivative of a Scalar with Respect to Vector

If y is a scalar,

$$\frac{\partial y}{\partial \mathbf{x}} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}. \quad (\text{F.4})$$

§F.2.3. Derivative of Vector with Respect to Scalar

If x is a scalar,

$$\frac{\partial \mathbf{y}}{\partial x} \stackrel{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} & \cdots & \frac{\partial y_m}{\partial x} \end{bmatrix} \quad (\text{F.5})$$

Remark F.1. Many authors, notably in statistics and economics, define the derivatives as the transposes of those given above.¹ This has the advantage of better agreement of matrix products with composition schemes such as the chain rule. Evidently the notation is not yet stable.

Example F.1. Given

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad (\text{F.6})$$

and

$$\begin{aligned} y_1 &= x_1^2 - x_2 \\ y_2 &= x_3^2 + 3x_2 \end{aligned} \quad (\text{F.7})$$

the partial derivative matrix $\partial \mathbf{y} / \partial \mathbf{x}$ is computed as follows:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_1}{\partial x_3} & \frac{\partial y_2}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{bmatrix} \quad (\text{F.8})$$

§F.2.4. Jacobian of a Variable Transformation

In multivariate analysis, if \mathbf{x} and \mathbf{y} are of the same order, the determinant of the square matrix $\partial \mathbf{x} / \partial \mathbf{y}$, that is

$$J = \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right| \quad (\text{F.9})$$

is called the *Jacobian* of the transformation determined by $\mathbf{y} = \mathbf{y}(\mathbf{x})$. The inverse determinant is

$$J^{-1} = \left| \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right|. \quad (\text{F.10})$$

Example F.2. The transformation from spherical to Cartesian coordinates is defined by

$$x = r \sin \theta \cos \psi, \quad y = r \sin \theta \sin \psi, \quad z = r \cos \theta \quad (\text{F.11})$$

where $r > 0$, $0 < \theta < \pi$ and $0 \leq \psi < 2\pi$. To obtain the Jacobian of the transformation, let

$$\begin{aligned} x &\equiv x_1, & y &\equiv x_2, & z &\equiv x_3 \\ r &\equiv y_1, & \theta &\equiv y_2, & \psi &\equiv y_3 \end{aligned} \quad (\text{F.12})$$

Then

$$\begin{aligned} J = \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right| &= \begin{vmatrix} \sin y_2 \cos y_3 & \sin y_2 \sin y_3 & \cos y_2 \\ y_1 \cos y_2 \cos y_3 & y_1 \cos y_2 \sin y_3 & -y_1 \sin y_2 \\ -y_1 \sin y_2 \sin y_3 & y_1 \sin y_2 \cos y_3 & 0 \end{vmatrix} \\ &= y_1^2 \sin y_2 = r^2 \sin \theta. \end{aligned} \quad (\text{F.13})$$

The foregoing definitions can be used to obtain derivatives to many frequently used expressions, including quadratic and bilinear forms.

¹ One author puts it this way: “When one does matrix calculus, one quickly finds that there are two kinds of people in this world: those who think the gradient is a row vector, and those who think it is a column vector.”

Example F.3. Consider the quadratic form

$$y = \mathbf{x}^T \mathbf{A} \mathbf{x} \quad (\text{F.14})$$

where \mathbf{A} is a square matrix of order n . Using the definition (D.3) one obtains

$$\frac{\partial y}{\partial \mathbf{x}} = \mathbf{A} \mathbf{x} + \mathbf{A}^T \mathbf{x} \quad (\text{F.15})$$

and if \mathbf{A} is symmetric,

$$\frac{\partial y}{\partial \mathbf{x}} = 2\mathbf{A} \mathbf{x}. \quad (\text{F.16})$$

We can of course continue the differentiation process:

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = \frac{\partial}{\partial \mathbf{x}} \left(\frac{\partial y}{\partial \mathbf{x}} \right) = \mathbf{A} + \mathbf{A}^T, \quad (\text{F.17})$$

and if \mathbf{A} is symmetric,

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = 2\mathbf{A}. \quad (\text{F.18})$$

The following table collects several useful vector derivative formulas.

\mathbf{y}	$\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{A} \mathbf{x}$	\mathbf{A}^T
$\mathbf{x}^T \mathbf{A}$	\mathbf{A}
$\mathbf{x}^T \mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T \mathbf{A} \mathbf{x}$	$\mathbf{A} \mathbf{x} + \mathbf{A}^T \mathbf{x}$

§F.3. The Chain Rule for Vector Functions

Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \quad \text{and} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \quad (\text{F.19})$$

where \mathbf{z} is a function of \mathbf{y} , which is in turn a function of \mathbf{x} . Using the definition (D.2), we can write

$$\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right)^T = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \cdots & \frac{\partial z_1}{\partial x_n} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_m}{\partial x_1} & \frac{\partial z_m}{\partial x_2} & \cdots & \frac{\partial z_m}{\partial x_n} \end{bmatrix} \quad (\text{F.20})$$

Each entry of this matrix may be expanded as

$$\frac{\partial z_i}{\partial x_j} = \sum_{q=1}^r \frac{\partial z_i}{\partial y_q} \frac{\partial y_q}{\partial x_j} \quad \begin{cases} i = 1, 2, \dots, m \\ j = 1, 2, \dots, n. \end{cases} \quad (\text{F.21})$$

Then

$$\begin{aligned}
 \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}} \right)^T &= \begin{bmatrix} \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \cdots & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_n} \end{bmatrix} \\
 &= \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \cdots & \frac{\partial z_1}{\partial y_r} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \cdots & \frac{\partial z_2}{\partial y_r} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial z_m}{\partial y_1} & \frac{\partial z_m}{\partial y_2} & \cdots & \frac{\partial z_m}{\partial y_r} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial y_r}{\partial x_1} & \frac{\partial y_r}{\partial x_2} & \cdots & \frac{\partial y_r}{\partial x_n} \end{bmatrix} \\
 &= \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}} \right)^T \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right)^T = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \right)^T. \tag{F.22}
 \end{aligned}$$

On transposing both sides, we finally obtain

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}}, \tag{F.23}$$

which is the *chain rule* for vectors. If all vectors reduce to scalars,

$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} \frac{\partial z}{\partial y} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}, \tag{F.24}$$

which is the conventional chain rule of calculus. Note, however, that when we are dealing with vectors, the chain of matrices builds “toward the left.” For example, if \mathbf{w} is a function of \mathbf{z} , which is a function of \mathbf{y} , which is a function of \mathbf{x} ,

$$\frac{\partial \mathbf{w}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{w}}{\partial \mathbf{z}}. \tag{F.25}$$

On the other hand, in the ordinary chain rule one can indistinctly build the product to the right or to the left because scalar multiplication is commutative.

§F.4. The Derivative of Scalar Functions of a Matrix

Let $\mathbf{X} = (x_{ij})$ be a matrix of order $(m \times n)$ and let

$$y = f(\mathbf{X}), \tag{F.26}$$

be a scalar function of \mathbf{X} . The derivative of y with respect to \mathbf{X} , denoted by

$$\frac{\partial y}{\partial \mathbf{X}}, \tag{F.27}$$

is defined as the following matrix of order $(m \times n)$:

$$\mathbf{G} = \frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1n}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2n}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y}{\partial x_{m1}} & \frac{\partial y}{\partial x_{m2}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix} = \left[\frac{\partial y}{\partial x_{ij}} \right] = \sum_{i,j} \mathbf{E}_{ij} \frac{\partial y}{\partial x_{ij}}, \quad (\text{F.28})$$

where \mathbf{E}_{ij} denotes the elementary matrix* of order $(m \times n)$. This matrix \mathbf{G} is also known as a *gradient matrix*.

Example F.4. Find the gradient matrix if y is the trace of a square matrix \mathbf{X} of order n , that is

$$y = \text{tr}(\mathbf{X}) = \sum_{i=1}^n x_{ii}. \quad (\text{F.29})$$

Obviously all non-diagonal partials vanish whereas the diagonal partials equal one, thus

$$\mathbf{G} = \frac{\partial y}{\partial \mathbf{X}} = \mathbf{I}, \quad (\text{F.30})$$

where \mathbf{I} denotes the identity matrix of order n .

§F.4.1. Functions of a Matrix Determinant

An important family of derivatives with respect to a matrix involves functions of the determinant of a matrix, for example $y = |\mathbf{X}|$ or $y = |\mathbf{A}\mathbf{X}|$. Suppose that we have a matrix $\mathbf{Y} = [y_{ij}]$ whose components are functions of a matrix $\mathbf{X} = [x_{rs}]$, that is $y_{ij} = f_{ij}(x_{rs})$, and set out to build the matrix

$$\frac{\partial |\mathbf{Y}|}{\partial \mathbf{X}}. \quad (\text{F.31})$$

Using the chain rule we can write

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \sum_i \sum_j \mathbf{Y}_{ij} \frac{\partial |\mathbf{Y}|}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial x_{rs}}. \quad (\text{F.32})$$

But

$$|\mathbf{Y}| = \sum_j y_{ij} \mathbf{Y}_{ij}, \quad (\text{F.33})$$

where \mathbf{Y}_{ij} is the *cofactor* of the element y_{ij} in $|\mathbf{Y}|$. Since the cofactors $\mathbf{Y}_{i1}, \mathbf{Y}_{i2}, \dots$ are independent of the element y_{ij} , we have

$$\frac{\partial |\mathbf{Y}|}{\partial y_{ij}} = \mathbf{Y}_{ij}. \quad (\text{F.34})$$

It follows that

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \sum_i \sum_j \mathbf{Y}_{ij} \frac{\partial y_{ij}}{\partial x_{rs}}. \quad (\text{F.35})$$

* The elementary matrix \mathbf{E}_{ij} of order $m \times n$ has all zero entries except for the (i, j) entry, which is one.

There is an alternative form of this result which is occasionally useful. Define

$$a_{ij} = \mathbf{Y}_{ij}, \quad \mathbf{A} = [a_{ij}], \quad b_{ij} = \frac{\partial y_{ij}}{\partial x_{rs}}, \quad \mathbf{B} = [b_{ij}]. \quad (\text{F.36})$$

Then it can be shown that

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \text{tr}(\mathbf{A}\mathbf{B}^T) = \text{tr}(\mathbf{B}^T \mathbf{A}). \quad (\text{F.37})$$

Example F.5. If \mathbf{X} is a nonsingular square matrix and $\mathbf{Z} = |\mathbf{X}|\mathbf{X}^{-1}$ its cofactor matrix,

$$\mathbf{G} = \frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = \mathbf{Z}^T. \quad (\text{F.38})$$

If \mathbf{X} is also symmetric,

$$\mathbf{G} = \frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = 2\mathbf{Z}^T - \text{diag}(\mathbf{Z}^T). \quad (\text{F.39})$$

§F.5. The Matrix Differential

For a scalar function $f(\mathbf{x})$, where \mathbf{x} is an n -vector, the ordinary differential of multivariate calculus is defined as

$$df = \sum_{i=1}^n \frac{\partial f}{\partial x_i} dx_i. \quad (\text{F.40})$$

In harmony with this formula, we define the differential of an $m \times n$ matrix $\mathbf{X} = [x_{ij}]$ to be

$$d\mathbf{X} \stackrel{\text{def}}{=} \begin{bmatrix} dx_{11} & dx_{12} & \dots & dx_{1n} \\ dx_{21} & dx_{22} & \dots & dx_{2n} \\ \vdots & \vdots & & \vdots \\ dx_{m1} & dx_{m2} & \dots & dx_{mn} \end{bmatrix}. \quad (\text{F.41})$$

This definition complies with the multiplicative and associative rules

$$d(\alpha \mathbf{X}) = \alpha d\mathbf{X}, \quad d(\mathbf{X} + \mathbf{Y}) = d\mathbf{X} + d\mathbf{Y}. \quad (\text{F.42})$$

If \mathbf{X} and \mathbf{Y} are product-conforming matrices, it can be verified that the differential of their product is

$$d(\mathbf{XY}) = (d\mathbf{X})\mathbf{Y} + \mathbf{X}(d\mathbf{Y}). \quad (\text{F.43})$$

which is an extension of the well known rule $d(xy) = y dx + x dy$ for scalar functions.

Example F.6. If $\mathbf{X} = [x_{ij}]$ is a square nonsingular matrix of order n , and denote $\mathbf{Z} = |\mathbf{X}|\mathbf{X}^{-1}$. Find the differential of the determinant of \mathbf{X} :

$$d|\mathbf{X}| = \sum_{i,j} \frac{\partial |\mathbf{X}|}{\partial x_{ij}} dx_{ij} = \sum_{i,j} \mathbf{X}_{ij} dx_{ij} = \text{tr}(|\mathbf{X}|\mathbf{X}^{-1})^T d\mathbf{X} = \text{tr}(\mathbf{Z}^T d\mathbf{X}), \quad (\text{F.44})$$

where \mathbf{X}_{ij} denotes the cofactor of x_{ij} in \mathbf{X} .

Example F.7. With the same assumptions as above, find $d(\mathbf{X}^{-1})$. The quickest derivation follows by differentiating both sides of the identity $\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$:

$$d(\mathbf{X}^{-1})\mathbf{X} + \mathbf{X}^{-1}d\mathbf{X} = \mathbf{0}, \quad (\text{F.45})$$

from which

$$d(\mathbf{X}^{-1}) = -\mathbf{X}^{-1}d\mathbf{X}\mathbf{X}^{-1}. \quad (\text{F.46})$$

If \mathbf{X} reduces to the scalar x we have

$$d\left(\frac{1}{x}\right) = -\frac{dx}{x^2}. \quad (\text{F.47})$$

G

Graphic Utilities

TABLE OF CONTENTS

		Page
§G.1.	Introduction	G-3
§G.1.1.	Finite Element Plot Types	G-3
§G.1.2.	Contour Plotting Terminology	G-3
§G.2.	Contour Band Plotter: Mini Version	G-4
§G.2.1.	Plot Driver: CBPOver2DMeshMini	G-4
§G.2.2.	Band-Contour 3-Node Triangle: CBPOverTrig3	G-7
§G.2.3.	Contour-Line-Segment 3-Node Triangle: CBPTrig3Segment	G-9
§G.2.4.	Band-Contour 4-Node Quadrilateral: CBPOverQuad4	G-9
§G.2.5.	Build Plot Legend: CBPLegend	G-10
§G.2.6.	Map Value to Color: CBPColorMap	G-13
§G.2.7.	Testing the Contour Band Plotter	G-15
§G.3.	Genesis and Evolution of Plotting Utilities	G-17

§G.1. Introduction

This Appendix collects a set of application-independent graphic utilities organized as *Mathematica* modules. By “application independent” is meant that these modules are reusable across different FEM application programs that do not necessarily simulate structures.

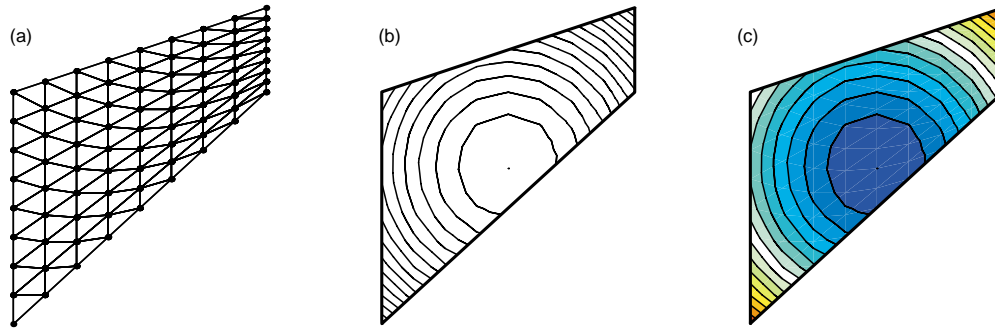


FIGURE G.1. Two-dimensional FEM plot types: (a) mesh plot; (b) contour line plot (CLP); (c) contour band plot (CBP); (d) contour polygonal plot (CPP), to be eventually added.

§G.1.1. Finite Element Plot Types

Figure G.1 illustrates several types of graphic displays pertaining to a two-dimensional (2D) finite element model. The mesh plot in Figure G.1(a) shows elements and nodes; in this case a mesh of 3-node triangles. Contour plots, such as those shown in Figure G.1(b)-(d) are a commonly used graphical technique to display a function of two variables defined over a 2D mesh. Distinguishing among these types requires some terminology, which is introduced in the next subsection.

§G.1.2. Contour Plotting Terminology

A *contour line*, also called *isoline*, of a function of two variables is a curve along which the function has a constant value. In cartography, a contour line, sometimes just called a *contour*, joins points of equal elevation (height) above a reference level, such as mean sea level. A *contour map*, also called *topographical map*, is a map illustrated with contour lines. The *contour interval* of a contour map is the difference in values between successive contour lines. The generalization of a contour line to functions of more than two variables is called a *level set*.

Now we are in a position to describe the three contour plot types shown in Figure G.1.

1. *Contour line plot*, or CLP. As shown in Figure G.1(b) this is just that: a set of contour lines, typically drawn at equal contour intervals. No color appears between lines. This kind of display is primarily used to identify general features at a glance; for example, steep gradient regions.
2. *Contour band plot*, or CBP. This is produced by coloring the contour bands (regions comprised between successive contour line) as pictured in Figure G.1(c). The color is picked as per a one-parameter value-to-color mapping scheme, in which “value” means the averaged value of the function at the band. This form is more frequently used than CLP, since it conveys more information at a glance. In particular, regions of maximum and minimum values are easily identified by color.

3. *Contour polygonal plot*, or CPP. Finite element domains are subdivided into polygons, each each of which is filled with a color again defined by a value-to-color mapping, in which “value” means the average function value over the polygon. This type is better suited to display of discontinuous functions that may jump between elements, but it has the disadvantage of rapid (superlinear) growth in cost as the number subdivisions over each element increases.

Both CLP and CBP are not restricted to finite element meshes but can be used for other applications. For example CLP is often used in cartography to plot elevations over a reference height. On the other hand, CPP is (by nature) restricted to FEM since it relies on subdividing element regions.

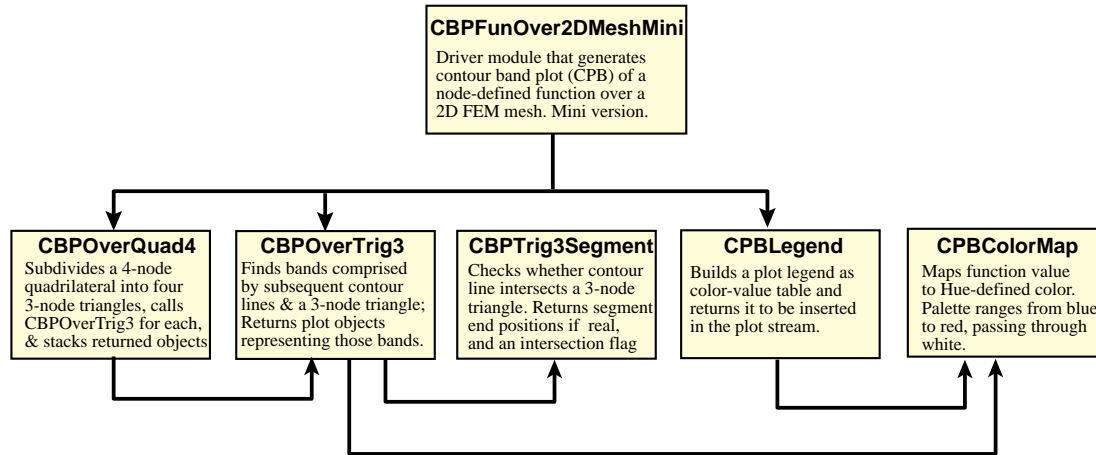


FIGURE G.2. Configuration of CBP plotter software with CBPOver2DMeshMini as driver.

§G.2. Contour Band Plotter: Mini Version

This section describes a “mini” CBP version with very basic contour plotting capabilities. Specific restrictions are:

1. Only 3-node triangles and 4-node quadrilaterals are accepted
2. Contour line interval is fixed
3. The plot object cannot be returned for downstream processing
4. The set of graphic options is limited.

The plotter software comprises six *Mathematica* modules, which are configured as shown in the diagram of Figure G.3. A full version is described in another Section. The full version allows more options, and accepts a richer set of element configurations. Both versions allow display of legends.

Both versions: mini and full, have an important limitation:

Function to be plotted is assumed C^0 continuous over the finite element mesh.

The main reason behind this restriction is that function values are given at nodes, and the value is assumed to be the same among all 2D elements connected to that node. If the function exhibits interelement jump discontinuities (for example, at material interfaces) a contour polygonal plotter (CPP) that allows functions to be specified element-by-element is more appropriate. One such module is described in another Section. (Eventually.)

```

CBPOver2DMeshMini[nodxyz_,elenod_,eletyp_,fn_,frain_,plotwhat_,
plotspecs_]:=Module[{e,i,n,nc,xyz,fc,enl,c,pb,pl,fmin,fmax,ncint,feps,
fspecs,colors,mesh,nodes,bacg,lines,legspecs,aspect,label,asprat,
prebacg,pbacg,preband,pband,preline,pline,premesh,pmesh,prenode,pnode,
preleg,pleg,empty,numele=Length[elenod]}, empty=Table[{},{numele}];
{colors,mesh,nodes,bacg,lines}=plotwhat;
pband=pmesh=pnode=pline=pbacg=preleg=pleg={};
{legspecs,aspect,label}=plotspecs; {fmin,fmax,ncint}=frain;
feps=10.^(-8)*Max[Abs[fmax],Abs[fmin]]; fspecs={fmin,fmax,ncint,feps};
If [colors,pband=empty]; If [mesh, pmesh=empty]; If [bacg,pbacg=empty];
If [lines, pline=empty]; If [nodes,pnode=empty];
prebacg={Graphics[RGBColor[.7,.7,.7]]}; preband={};
preline={Graphics[RGBColor[0,0,0]],Graphics[AbsoluteThickness[1]]};
premesh={Graphics[RGBColor[1,0,0]],Graphics[AbsoluteThickness[2]],
Graphics[AbsoluteDashing[{1,10}]]};
prenode={Graphics[RGBColor[0,0,0]],Graphics[AbsoluteDashing[{}]],
Graphics[AbsolutePointSize[4]]};
For [e=1,e<=numele,e++,
enl=elenod[[e]]; nc=Length[enl];
If [!MemberQ[{3,4},nc], Print["CBP mini:",
"Elem ",e," with ",nc," nodes skipped"]; Continue[]];
fc=xyz=Table[0,{nc}];
Do [n=enl[[i]]; xyz[[i]]=nodxyz[[n]];fc[[i]]=fn[[n]],{i,1,nc}];
If [nc==3, {pb,pl}=CBPOverTrig3[xyz,fc,fspecs,lines];
c=Table[xyz[[{1,2,3,1}][[i]]],{i,4}];
If [nc==4, {pb,pl}=CBPOverQuad4[xyz,fc,fspecs,lines];
c=Table[xyz[[{1,2,3,4,1}][[i]]],{i,5}];
If [colors, pband[[e]]=pb]; If [lines, pline[[e]]=pl];
If [mesh, pmesh[[e]]=Graphics[Line[c]]];
If [bacg, pbacg[[e]]=Graphics[Polygon[c]]];
If [nodes, pnode[[e]]=Table[Graphics[Point[ xyz[[i]] ]],
{i,nc}]];
];
asprat=aspect; If [aspect=={}, asprat=Automatic];
If [!colors,pband={}]; If [!mesh, pmesh={}]; If [!nodes, pnode={}];
If [!lines, pline={}]; If [!bacg, pbacg={}]; pleg={};
If [Length[legspecs]==0||!colors, Show[prebacg,pbacg,
preband,pband,preline,pline,premesh,pmesh,prenode,
pnode,AspectRatio->asprat,PlotLabel->label,
DisplayFunction->DisplayChannel ]];
If [Length[legspecs]>0&&colors,
{preleg,pleg}=CBPLegend[{fmin,fmax},legspecs,1];
Show[prebacg,pbacg,preband,pband,preline,pline,
premesh,pmesh,prenode,pnode,preleg,pleg,
TextStyle->{FontFamily->"Courier",
FontSlant->"Plain", FontSize->10},
AspectRatio->asprat,PlotLabel->label,
DisplayFunction->DisplayChannel ]];
ClearAll[pband,pline,pmesh,pnode,pbacg,pleg] ];

```

FIGURE G.3. Module CBPOver2DMeshMini to produce a contour band plot (CBP) of a function over a 2D finite element version. Mini version.

§G.2.1. Plot Driver: CBPOver2DMeshMini

The driver program is called CBPOver2DMeshMini. The code is listed in Figure G.3. It is invoked by

CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,plotwhat,plotspecs] (G.1)

The arguments are:

Appendix G: GRAPHIC UTILITIES

<code>nodxyz</code>	$\{x, y\}$ Node coordinates stored as $\{\{x_1, y_1\}, \{x_2, y_2\}, \dots, \{x_N, y_N\}\}$. in which N is number of nodes. The z coordinate is assumed zero and need not be listed,
<code>elenod</code>	Element node lists, stored as $\{enl1\}, \{enl2\}, \dots, \{enlNe\}$, in which Ne is number of elements. For a 3-node triangle, $enl=\{n1, n2, n3\}$. For a 4-node quadrilateral, $enl=\{n1, n2, n3, n4\}$. Element node lists that do not contain 3 or 4 nodes are skipped, and a warning message given.
<code>eletyp</code>	A dummy argument. In the full CBP version, it contains element type identifiers. ¹
<code>fn</code>	Function values at nodes: $\{f1, f2, \dots, fN\}$. The value is assumed to be the same for all elements connected to node.
<code>frain</code>	A 3-entry list $\{fmin, fmax, ncint\}$ specifying function range and contour interval: $fmin, fmax$: minimum and maximum function values to appear in plot $ncint$: number of contour intervals in the range $fmin$ through $fmax$. The contour interval will be $finc=(fmax-fmin)/ncint$. If $fmax \leq fmin$ or $ncint \leq 0$, contour lines and bands are not plotted. Regions where the function value is outside the range will appear black.
<code>plotwhat</code>	A list of five logical flags: $\{colors, mesh, nodes, bacg, lines\}$. These flags specify what is to be plotted: $colors$: If True, show band colors; if False omit. $mesh$: If True, show background mesh in dashed lines; if False omit. $nodes$: If True, show mesh nodes as black dots, if False omit. $bacg$: If True, show boundaries; if False omit. $lines$: If True, show contour lines that separate bands; if False omit. Note: this argument is more elaborate in the full CBP version.
<code>plotspecs</code>	A set of specifications: $\{legspec, aspect, label\}$ that controls appearance: $legspec$: Plot legend specifications: $\{\{xleg, yleg\}, mv\}$, in which $\{xleg, yleg\}$ are coordinates of a location where the legend frame is to appear, and mv is number of value intervals to shown in legend. To omit legend specify an empty list: $\{\}$. For legend placement and configuration details see §G.2.5. $aspect$: Aspect ratio of plot frame. If zero or negative, the default setting <code>AspectRatio->Automatic</code> is used. $label$: A textstring to be placed as plot label. To omit specify an empty list: $\{\}$.

The module does not return a value. To skip the plot display and have the CBP object returned for downstream processing, the full version should be used.

The major plotting operations performed by the driver module are carried out inside a For loop over the elements (see Figure G.3 for coding details; loop runs on element index e). The only elements accepted by the mini version are 3-node triangles and 4-node quadrilaterals. The latter are split into four 3-node triangles forming a Union Jack pattern, as described in §G.2.4. So eventually it

¹ The full version has exactly the same arguments, but some of them are more complicated.

```

CBPOverTrig3[xyc_,fc_,fspecs_,lines_] := Module[{fmin,fmax,ncint,
  feps,nv,iv,xc1,yc1,xc2,yc2,xc3,yc3,fc1,fc2,fc3,xs,ys,fs,
  x1,x2,x3,y1,y2,y3,f1,f2,f3,firsthit,finc,Pc1,Pc2,Pc3,
  ftop,fbot,favg,Pb,Pt,tb,tt,P1,P2,P3,P4,Q1,Q2,Q3,Q4,ptab,
  flast,frange,Plast,tlast,kb,kl,p,pb,pl},
{fmin,fmax,ncint,feps}=fspecs; nv=Min[ncint,1000];
If [fmax<fmin||ncint<=0, Return[{{}},{{}}]];
pb=Table[{{}},{nv}]; pl=Table[{{}},{nv+1}]; frange={fmin,fmax};
{{xc1,yc1},{xc2,yc2},{xc3,yc3}}=N[xyc]; {fc1,fc2,fc3}=N[fc];
{{x1,y1,f1},{x2,y2,f2},{x3,y3,f3}}=Sort[{{xc1,yc1,fc1},
  {xc2,yc2,fc2},{xc3,yc3,fc3}},#1[[3]]<=#2[[3]]&];
xs={x1,x2,x3}; ys={y1,y2,y3}; fs={f1,f2,f3};
Pc1={x1,y1}; Pc2={x2,y2}; Pc3={x3,y3};
ptab={{{{}},{Pc1,Pc2,Pc3},{Pc1,Q4,Q3},{Pc1,Pc2,Q3,Q4}},
  {{Null}},{{}},{{Null}},{{Null}}},
  {{Null}},{{Pc3,Pc2,Q1,Q2},{Q1,Q2,Q4,Q3},{Pc2,Q1,Q2,Q4,Q3}},
  {{Null}},{{Pc3,Q1,Q2},{Null}},{{Q1,Q2,Q4,Q3}}}};
finc=(fmax-fmin)/nv; flast=fmin-1000; firsthit=True; kb=kl=0;
For [iv=1,iv<=nv,iv++,
  fbot=N[fmin+(iv-1)*finc]; ftop=N[fbot+finc];
  If [ftop<=f1 || fbot>=f3, Continue[]]; favg=(fbot+ftop)/2;
  If [fbot==flast, {fbot,Pb,tb}={flast,Plast,tlast},
    {Pb,tb}=CBPTrig3Segment[xs,ys,fs,fbot,feps];
    {Pt,tt}=CBPTrig3Segment[xs,ys,fs,ftop,feps];
    {P1,P2}=Pb; {P3,P4}=Pt; {flast,Plast,tlast}={ftop,Pt,tt};
  If [lines,
    If [firsthit&&(tb>0),
      pl[[++kl]]=Graphics[Line[Pb]]; firsthit=False];
    If [tt>0, pl[[++kl]]=Graphics[Line[Pt]]];
  ];
  p=ptab[[tb+2,tt+2]]/.{Q1->P1,Q2->P2,Q3->P3,Q4->P4};
  pb[[++kb]]= {CBPColorMap[favg,frange,1],Graphics[Polygon[p]]};
  ];
If [kb>0,pb=Take[pb,kb],{{}},{{}}]; If [kl>0,pl=Take[pl,kl],{{}},{{}}];
ClearAll[ptab]; Return[{pb,pl}] ];

```

FIGURE G.4. Module CBPOverTrig3 that contour-bands a function over a 3-node triangle.

all boils down to contour-banding over 3-node triangles; this is done as described in §G.2.2. The reason for this splitting is that the function $f(x, y)$ can be then linearly interpolated from the 3 corner values $\{f_1, f_2, f_3\}$.

§G.2.2. Band-Contour 3-Node Triangle: CBPOverTrig3

Module CBPOverTrig3 searches over the plot range for all contour lines that cross a specific 3-node triangle, and builds the corresponding contour band polygons.² The module source code is listed in Figure G.4. The module is invoked as

$$\{\text{preleg}, \text{plab}\} = \text{CBPOverTrig3}[\text{xyc}, \text{fc}, \text{fspecs}, \text{lines}] \quad (\text{G.2})$$

The arguments are:

- | | |
|--------|--|
| xyc | The $\{x, y\}$ corner coordinates arranged $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$. |
| fc | Corner function values stored as $\{f_1, f_2, f_3\}$. Corners are internally renumbered within the module so that $f_1 \leq f_2 \leq f_3$. |
| fspecs | A 4-item list of function plotting specifications: $\{fmin, fmax, ncint, feps\}$. |

² This operation is called *polygon clipping* in the computer science literature.

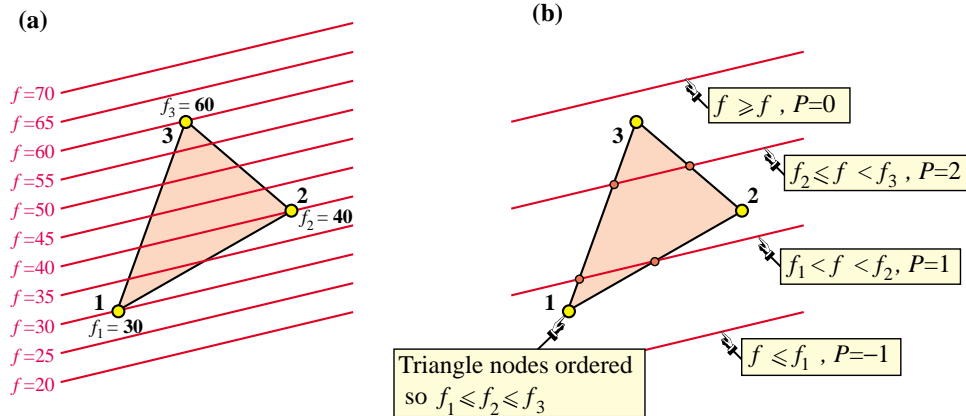


FIGURE G.5. Contour lines of a plane passing through triangle corners with values (sorted by increasing value) are $f_1=30$, $f_2=40$ and $f_3=60$. (a) Contour lines $f=20$ through $f=70$ are straight lines on account of linear interpolation from corner values (lines shown outside triangle for visualization convenience); (b) 4 intersection cases identified by an integer flag P .

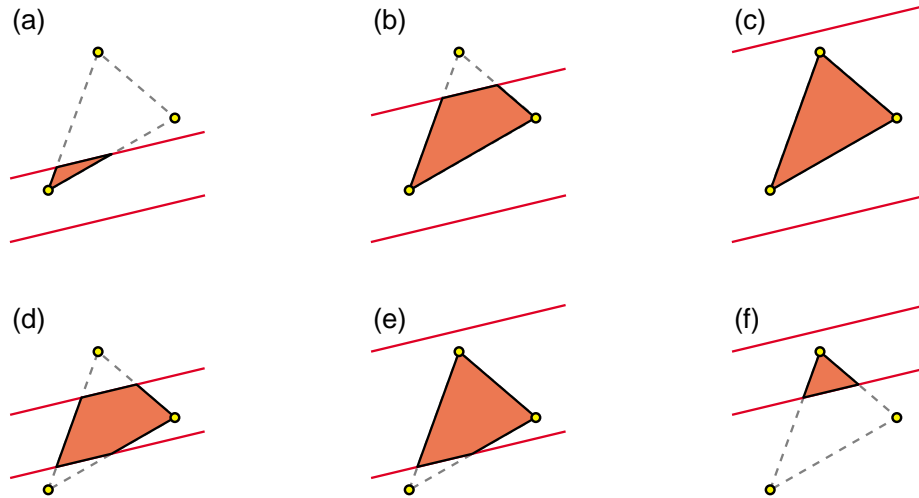


FIGURE G.6. Contour band polygons over a 3-node triangle. Shown are 6 cases in which one or more contour line intersections occur. Polygons have 3, 4, 3, 5, 4 and 3 sides for cases (a) through (f), respectively.

fmin,fmax,ncint: Same as items in argument **frain** of the driver module.

feps: A function value tolerance set by the driver module **CBP0ver2DMeshMini** to $\text{eps} * \text{Max}[\text{Abs}[f_{\text{min}}], \text{Abs}[f_{\text{max}}]]$, in which $\text{eps} \ll 1$ is typically 10^{-8} . Used to avoid ill conditioned computations in **CBPTrig3Segment**.

lines The logical flag supplied in argument **plotwhat** of driver module **CBP0ver2DMeshMini**. If **True**, contour lines separating bands are drawn.

The module returns

{ pb,pl } Graphic objects that define contour band polygons if any found. The first object: **pb** has information about the polygon geometry and color fill, whereas the second object: **texttpl**, return polygon boundary lines if requested. If no contour band polygons are found, the objects are empty.

```

CBPTrig3Segment[xs_,ys_,fs_,fv_,feps_]:=Module[{x1,x2,x3,
y1,y2,y3,f1,f2,f3,x21,y21,x32,y32,x13,y13,f21,f32,f31,
ζ211,ζ212,ζ311,ζ313,ζ322,ζ323,P1,P3,P21,P32,P13},
{xs,x2,x3}=xs; {y1,y2,y3}=ys; {f1,f2,f3}=fs; P1={x1,y1}; P3={x3,y3};
If [fv<f1, Return[{{P1,P1},-1}]]; If [fv>f3, Return[{{P3,P3},0}]];
f21=f2-f1; If [f21<feps,f1=f1-feps,f21=feps];
f32=f3-f2; If [f32<feps,f3=f2+feps,f32=feps]; f31=f3-f1;
ζ212=(fv-f1)/f21; ζ212=Max[Min[ζ212,1],0]; ζ211=1-ζ212;
ζ313=(fv-f1)/f31; ζ313=Max[Min[ζ313,1],0]; ζ311=1-ζ313;
P21={x2*ζ212+x1*ζ211,y2*ζ212+y1*ζ211};
P13={x1*ζ311+x3*ζ313,y1*ζ311+y3*ζ313};
If [fv<f2,Return[{{P21,P13},1}]];
ζ323=(fv-f2)/f32; ζ323=Max[Min[ζ323,1],0]; ζ322=1-ζ323;
P32={x3*ζ323+x2*ζ322,y3*ζ323+y2*ζ322};
Return[{{P32,P13},2}]];

```

FIGURE G.7. Module CBPTrig3Segment that determines whether a contour line intersects a 3-node triangle, and returns intersection segment information if that is the case.

The contour-band capturing logic³ can be better explained through examination of Figures G.5 and G.6. Module CBPOverTrig3 cycles over the set of contour line values and calls module CBPTrig3Segment (described in the next Subsection) to determine whether one or more intersections occur. If this the case, it uses a table-driven scheme to construct the appropriate polygon from the six cases shown in G.6.

§G.2.3. Contour-Line-Segment 3-Node Triangle: CBPTrig3Segment

Module CBPTrig3Segment is invoked by CBPOverTrig3. It receives the corner coordinates of a specific 3-node triangle, the corner function values sorted in ascending order, and a contour line value. It finds whether that line intersects the triangle, and returns appropriate information to that effect. The module source code is listed in Figure G.7. The module is invoked as

$$\{xint, flag\} = \text{CBPTrig3Segment}[xs, ys, fs, fv, feps] \quad (\text{G.3})$$

The arguments are:

- | | |
|------|---|
| xs | The sorted x corner coordinates $\{x1, x2, x3\}$. |
| ys | The sorted y corner coordinates $\{y1, y2, y3\}$. |
| fs | The sorted corner function values $\{f1, f2, f3\}$. |
| fv | The countour line value. |
| feps | A function tolerance value used to avoid ill-conditioned computations. See feps argument in CBPOverTrig3. |

The module returns

- $\{\{P1, P2\}, P\}$ If an intersection found, $P1, P2$ contain the $\{x, y\}$ coordinates of the end points of the segment. If no intersection is found, both $P1$ and $P2$ are set to the coordinates of the corner closest to the contour line. Item P is an “intersection tag”: an integer with values -1 through 2 that identifies the 4 cases pictured in Figure G.5(b).

The CBPTrig3Segment logic relies heavily on the use of the triangular coordinates $\{\zeta_1, \zeta_2, \zeta_3\}$ introduced in Chapter 15. This is done to help numerical accuracy and stability because such coordinates are dimensionless and well scaled, as they valued in the range $[0, 1]$. On the other hand, the Cartesian coordinates might be poorly scaled in some unit systems.

³ The table-driven logic in CBPOverTrig3 is admittedly hairy but can be understood after some study.

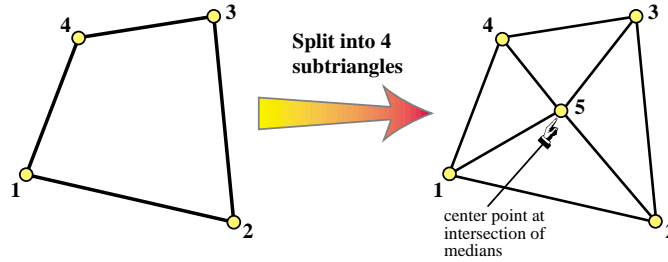


FIGURE G.8. Splitting a 4-node quadrilateral into four 3-node triangles.

```
CBPOverQuad4[xyc_,fc_,fspecs_,lines_]:=Module[
  {x1,x2,x3,x4,x5,y1,y2,y3,y4,y5,f1,f2,f3,f4,f5,
  e,xytab,ftab,xyt,ft, pbq,plq,pbt,plt},
  pbq=plq=Table[{},{4}];
  {{x1,y1},{x2,y2},{x3,y3},{x4,y4}}=xyc; {f1,f2,f3,f4}=fc;
  x5=(x1+x2+x3+x4)/4; y5=(y1+y2+y3+y4)/4; f5=(f1+f2+f3+f4)/4;
  xytab={{x1,y1},{x2,y2},{x5,y5}},{{x2,y2},{x3,y3},{x5,y5}},
  {{x3,y3},{x4,y4},{x5,y5}},{{x4,y4},{x1,y1},{x5,y5}}};
  ftab={{f1,f2,f5},{f2,f3,f5},{f3,f4,f5},{f4,f1,f5}};
  For [e=1,e<=4,e++, xyt=xytab[[e]]; ft=ftab[[e]];
    {pbq[[e]],plq[[e]]}=CBPOverTrig3[xyt,ft,fspecs,lines];
  ];
  Return[{pbq,plq}] ];
```

FIGURE G.9. Module CBPOverQuad4 that contour-bands a function over a 4-node quadrilateral.

§G.2.4. Band-Contour 4-Node Quadrilateral: CBPOverQuad4

Module CBPOverQuad4 processes a 4-node quadrilateral element (Quad4) by subdividing it into four subtriangles that meet at the element center, as illustrated in Figure G.8. The average of the corner function values is assigned at the center node. It then calls CBPTrig3 four times, and stacks the plot object information. The module source code is listed in Figure G.9.

The module is invoked as

$$\{plq,pbq\}=CBPOverQuad4[xyc,fc,fspecs,lines] \quad (G.4)$$

The arguments are:

- xyc The $\{x, y\}$ corner coordinates arranged $\{\{x1,y1\},\{x2,y2\},\{x3,y3\},\{x4,y4\}\}$.
- fc Corner function values stored as $\{f1,f2,f3,f4\}$.
- fspecs Same as in CBPOverTrig3.
- lines Same as in CBPOverTrig3.

The module returns

- $\{pbq,plq\}$ Graphic objects that define contour band polygons if any found. The first object: pbq has information about the polygon geometry and color fill, whereas the second object: texttplq, returns polygon boundary lines if requested. If no contour band polygons are found, the objects are empty.

```

CBPLegend[frange_, legspecs_, smax_] := Module[
  {fmin, fmax, f, fs, finc, fa, fb, xylinL, xylinR, xytext,
   black=Graphics[RGBColor[0,0,0]], preleg, plab={},
   white=Graphics[RGBColor[1,1,1]], xy1, xy2, xy3, xy4,
   baselineskip=10, dy, xf1, xf2, xf3, xf4, iv, xyleg, nvleg,
   thin=Graphics[AbsoluteThickness[1]], color, padOK},
  {xyleg, nvleg} = legspecs; nvleg = Min[Max[nvleg, 1], 20];
  {fmin, fmax} = frange; {fa, fb} = {Abs[fmin], Abs[fmax]};
  xf1 = Offset[{-30, 10}, xyleg]; xf2 = Offset[{85, 10}, xyleg];
  xf3 = Offset[{85, -10-baselineskip*(nvleg+1)}, xyleg];
  xf4 = Offset[{-30, -10-baselineskip*(nvleg+1)}, xyleg];
  preleg = {thin, white,
    Graphics[Polygon[{xf1, xf2, xf3, xf4}]], black,
    Graphics[Line[{xf1, xf2, xf3, xf4, xf1}]],
    Graphics[Text["LEGEND", Offset[{5, 0}, xyleg], {-1, 0}]]];
  plab = Table[{}, {nvleg+1}]; If [nvleg <= 0, Return[{preleg, plab}]];
  finc = (fmax - fmin) / nvleg; dy = baselineskip / 2;
  padOK = (Max[fa, fb] <= 1000) && (Min[fa, fb] >= 1 / 100);
  For [iv = 1, iv <= nvleg + 1, iv++, f = N[fmin + (iv - 1) * finc];
    xy1 = Offset[{-22, -baselineskip*iv + dy}, xyleg];
    xy2 = Offset[{-22, -baselineskip*iv - dy}, xyleg];
    xy3 = Offset[{8, -baselineskip*iv - dy}, xyleg];
    xy4 = Offset[{8, -baselineskip*iv + dy}, xyleg];
    xytext = Offset[{80, -baselineskip*iv}, xyleg];
    If [padOK, fs = PaddedForm[f, 4, NumberSigns -> {"-", "+"}],
      fs = ScientificForm[f, 4, NumberSigns -> {"-", "+"}]];
    color = CBPCOLORMAP[f, {fmin, fmax}, smax];
    plab[[iv]] = {color, Graphics[Polygon[{xy1, xy2, xy3, xy4}]],
      black, Graphics[Text[fs, xytext, {1, 0}]]];
  Return[{preleg, plab}]];

```

FIGURE G.10. Module CBPLegend that builds a plot legend and returns it to the caller as a plot object.

```

dfun = $DisplayFunction; If [$VersionNumber >= 6.0, dfun = Print];
For [ip = 1, ip <= 1, ip++, imgsiz = 300;
  xy1 = {0, 0}; xy2 = {imgsiz, 0}; xy3 = {imgsiz, imgsiz}; xy4 = {0, imgsiz};
  a = b = imgsiz / 2; xyleg = {a + 50, b + 80}; If [ip == 4, xyleg = {a + 50, b + 125}];
  If [ip == 6, xyleg = {a - 115, b - 15}];
  fmax = 100; If [ip == 5, fmax = 10000]; If [ip == 6, fmax = 1 / 3000];
  fmin = -fmax; smax = 1; If [ip == 2, smax = 1 / 2];
  mv = 10; If [ip == 3, mv = 2]; If [ip == 4, mv = 10];
  Print["fmin, fmax, xyleg, mv, smax = ", {fmin, fmax, xyleg, mv, smax}];
  {preleg, plab} = CBPLegend[{fmin, fmax}, {xyleg, mv}, smax];
  Show[Graphics[GrayLevel[0.9]], Graphics[Polygon[{xy1, xy2, xy3, xy4}]],
    Graphics[RGBColor[1, 0, 0]], Graphics[AbsoluteThickness[2]],
    Graphics[Line[{xy1, xy2, xy3, xy4, xy1}]], preleg, plab,
    AspectRatio -> Automatic, DisplayFunction -> dfun];
];

```

FIGURE G.11. Test script for exercising CBPLegend, which produces the six plots of Figure G.12.

§G.2.5. Build Plot Legend: CBPLegend

Module CBPLegend generates a *plot legend*, which is a tabular visualization of the value-to-color mapping. The module source code is listed in Figure G.10. The module is invoked as

$$\{\text{preleg}, \text{plab}\} = \text{CBPLegend}[\text{frange}, \text{xyleg}, \text{mv}, \text{smax}] \quad (\text{G.5})$$

The arguments are

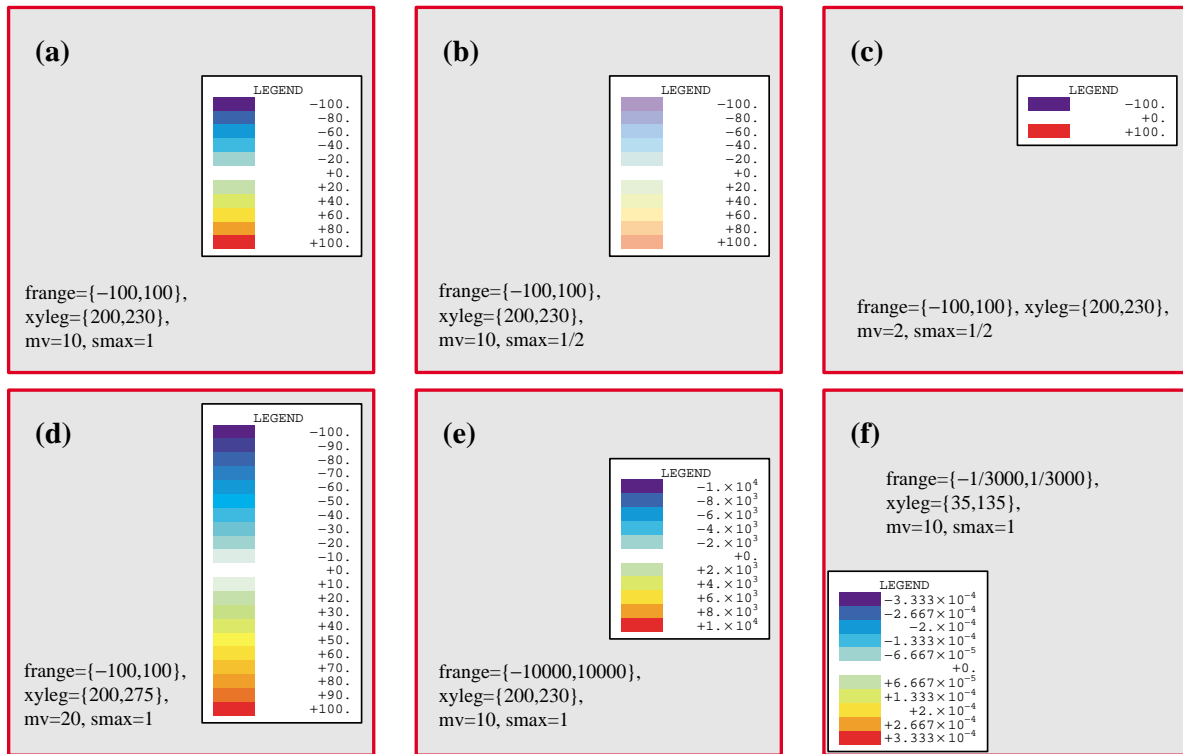


FIGURE G.12. Legend plots produced by the test script in Figure G.11

- frange** A list { *fmin*, *fmax* } of minimum and maximum function values, respectively, which defines the plot range.
- xyleg** The {*x*, *y*} coordinates of a location that collocates the legend frame in the plot. See the source code for details. (The positioning logic is likely to be changed in the future, as it is currently confusing.)
- mv** A positive integer that specifies the contour interval. values as $(f_{\max} - f_{\min}) / mv$. For example, if *mv*=2 three values will appear in the legend table: *fmin*, $(f_{\min} + f_{\max}) / 2$, and *fmax*. Note that *mv* is the same as the *ncint* input to the driver `CBPOver2DMeshMini` if `texttt.ncint>0`
- smax** Maximum saturation value in color specification returned by `CBPColorMap`; see §G.2.6 for details. Normally 1.

The function returns { *preleg*, *plab* }, in which

- preleg** Graphic commands to draw the legend table frame and title.
- plab** Graphic commands to show colors alongside function values.

These graphic objects are inserted in the command plot stream by the driver program.

The legend module may be exercised by the plot script listed in Figure G.11. Running the script produces the plots shown in Figure G.12. These can be used to visualize various effects and options.

```

CBPColorMap[f_,frange_,smax_]:= Module[{fmin,fmax,fs,
h,s,b=1,cs=25}, {fmin,fmax}=frange;
If [Abs[fmax]<=0 || fmin>=fmax, (* White if void range *)
Return[Graphics[Hue[0,0,1]]]];
If [f>fmax || f<fmin, (* Black if f outside range *)
Return[Graphics[Hue[0,0,0]]]];
fs=(fmax-f)/(fmax-fmin); h=0.64*fs;
s=smax*(1+1/cs)*(1-1/(1+cs*(1-2*fs)^2));
Return[Graphics[Hue[h,s,b]]]];

```

FIGURE G.13. Module CBPColorMap that maps a function value to a color defined through the built-in Hue graphics function.

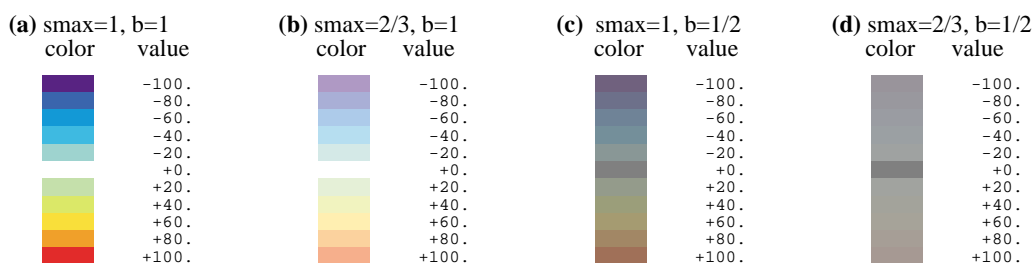


FIGURE G.14. Color palettes produced by calling CBPColorMap over the range $f_{\min}=-100$ through $f_{\max}=+100$ at $f_{\text{inc}}=+20$ intervals. Brightness and saturation values as indicated in the Figure.

§G.2.6. Map Value to Color: CBPColorMap

Module CBPColorMap, listed in Figure G.13, maps a function value associated with a band to a color through the mapping scheme described below. The module is invoked by

$$\text{color} = \text{CBPColorMap}[f, \text{frange}, \text{smax}] \quad (\text{G.6})$$

The arguments are:

- f Function value corresponding to color.
- frange A list $\{f_{\min}, f_{\max}\}$ that defines the plot range.
- smax Maximum color saturation; normally 1 (see description below)

The module returns the color specification in terms of the Hue built-in function:

$\text{Graphics}[\text{Hue}[h, s, b]]$ The triple h , s and b specify the hue, saturation and brightness, respectively, of the returned color.

On entry CBPColorMap checks for void function range: $|f_{\max}| \leq 0$ or $f_{\max} \leq f_{\min}$; if so it returns $\text{Graphics}[\text{Hue}[0, 0, 0]]$, which is white. If f is outside the range $[f_{\min}, f_{\max}]$ it returns $[\text{Hue}[0, 0, 1]]$, which is black. If both checks are passed, it computes the nondimensional function value $f_s = (f_{\max} - f) / (f_{\max} - f_{\min})$, which varies from 0 for $f = f_{\max}$ to 1 for $f = f_{\min}$. Hue and saturation are respectively set to $h = 0.64 \cdot f_s$, and $s = \text{smax} \cdot (1 + 1/\text{cs}) \cdot (1 - 1/(1 + \text{cs} \cdot (1 - 2 \cdot f_s)^2))$ with $\text{cs} = 25$. Brightness is internally preset to $b = 1$.

The effect of these choices is illustrated in the color palettes shown in Figure G.14. If $f = f_{\min}$ so that $f_s = 0$, $h = 0.64$, which with $s = 1$ gives deep blue, whereas if f reaches f_{\max} so that $f_s = 0$, $h = 0$, which along with $s = 1$ gives deep red. These extremes can be observed in the palette of Figure G.14(a).

For the average function value $f=(f_{\min}+f_{\max})/2$, whence $f_s=1/2$, one would like to get white. But white is not produced for any h as long as the saturation is nonzero. The solution is to force the saturation to be zero if $f_s=1/2$, which is done by the s function stated above. For $f_s=0$ and $f_s=1$ it yields $s=s_{\max}$ as expected. The rational interpolation form and its coefficient c_s were obtained by trial and error to get nice color variations in the vicinity of $f=0$, as can be observed in Figure G.14(a).

Setting the maximum saturation s_{\max} to less than 1 diffuses colors. This is evident in the palette of Figure G.14(b), produced with $s_{\max}=2/3$. Since this diffusivity has some uses, the value of s_{\max} can be set to the contour plotter from outside, overriding a default. On the other hand, moving brightness away from 1 is not a good idea since it mangles colors, as can be seen in the palettes of Figure G.14(c,d). For this reason $b=1$ is internally set once and for all in the local variable list.

```

GenNodeCoordOverTrapezoid[xycorn_,nx_,ny_]:= Module[
  {k=0,i,j,dx,dy,numnod,N1,N2,N3,N4,xi,eta,
  xc1,xc2,xc3,xc4,yc1,yc2,yc3,yc4,x,y,elenod},
  numnod=(nx+1)*(ny+1); nodxyz=Table[0,{numnod}];
  {{xc1,yc1},{xc2,yc2},{xc3,yc3},{xc4,yc4}}=xycorn;
  For[i=1,i<=nx+1,i++, For [j=1,j<=ny+1,j++,
    xi=2*(i-1)/nx-1; eta=2*(j-1)/ny-1;
    N1=(1-xi)*(1-eta)/4; N2=(1+xi)*(1-eta)/4;
    N3=(1+xi)*(1+eta)/4; N4=(1-xi)*(1+eta)/4;
    x=xc1*N1+xc2*N2+xc3*N3+xc4*N4;
    y=yc1*N1+yc2*N2+yc3*N3+yc4*N4;
    nodxyz[[++k]]={x,y};
  ]];
  Return[nodxyz]];

GenTrig3NodeNumbersOverTrapezoid[nx_,ny_,pat_]:= Module[{i,j,k,
  c1,c2,numele,enl},numele={2,2,4,2}[[pat]]*nx*ny;
  enl=Table[{0,0,0},{numele}]; k=0;
  Do [Do [ c1=(ny+1)*(i-1)+j; c2=c1+ny+1;
    If [pat==1, enl[[++k]]={c1,c2,c1+1};
      enl[[++k]]={c2+1,c1+1,c2}];
    If [pat==2, enl[[++k]]={c1,c2,c2+1};
      enl[[++k]]={c2+1,c1+1,c1}];
    If [pat==3, enl[[++k]]={c1,c2,c1+1};
      enl[[++k]]={c2+1,c1+1,c2};
      enl[[++k]]={c1,c2,c2+1};
      enl[[++k]]={c2+1,c1+1,c1}];
    If [pat==4, If [j<=ny/2,
      enl[[++k]]={c1,c2,c2+1};
      enl[[++k]]={c2+1,c1+1,c1},
      enl[[++k]]={c1,c2,c1+1};
      enl[[++k]]={c2+1,c1+1,c2}]]],
  {j,1,ny}},{i,1,nx}]; Return[enl]];

GenQuad4NodeNumbersOverTrapezoid[nx_,ny_]:= Module[
  {k,i,j,c1,c2,numele,elenod},numele=nx*ny;
  elenod=Table[{0,0,0,0},{numele}]; k=0;
  For [i=1,i<=nx,i++, For [j=1,j<=ny,j++,
    c1=(ny+1)*(i-1)+j; c2=c1+ny+1;
    elenod[[++k]]={c1,c2,c2+1,c1+1}
  ]]; Return[elenod]];

```

FIGURE G.15. Mesh generation modules used by the CBPOver2DMeshMini tester listed in Figure G.16.

Remark G.1. Sometimes it is desirable to associate white with the zero value of the function, instead of the average. To do that, specify the function range as $-f_{\max}, f_{\max}$, in which f_{\max} is the maximum absolute value. Obviously the average is zero. For example, suppose that the node function values span the range -85 to 26 . Then set $f_{\min}=-85$ and $f_{\max}=+85$ in the inputs to the driver module.

```

nx=ny=8; scale=1; nodxyz=GenNodeCoordOverTrapezoid[{{0,0},{48,44},
{48,60},{0,44}},nx,ny]; aspect=60/48;
elenod=GenTrig3NodeNumbersOverTrapezoid[nx,ny,1];
fun[x_,y_]:=N[(x-24)^2+(y-30)^2-200]*scale;
numele=Length[elenod]; numnod=Length[nodxyz]; fn=Table[0,{numnod}];
For[n=1,n<=numnod,n++, {xn,yn}=nodxyz[[n]]; fn[[n]]=fun[xn,yn]];
fmin=Min[fn]; fmax=Max[fn]; ncint=15; frain={fmin,fmax,ncint};
plotspecsleg={{28,18},6},aspect,"Bandplot over Cook's Wing";
plotspecsnoleg={{},aspect,"Bandplot over Cook's Wing";
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{True,False,False,True,True},plotspecsleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{True,False,False,True,False},plotspecsleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{False,False,False,True,True},plotspecsnoleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{False,True,True,True,False},plotspecsnoleg]]
elenod=GenQuad4NodeNumbersOverTrapezoid[nx,ny];
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{True,False,False,True,True},plotspecsleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{True,False,False,True,False},plotspecsleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{False,False,False,True,True},plotspecsnoleg]]
Timing[CBPOver2DMeshMini[nodxyz,elenod,eletyp,fn,frain,
{False,True,True,True,False},plotspecsnoleg]]

```

FIGURE G.16. Test script for CBPOver2DMeshMini.

§G.2.7. Testing the Contour Band Plotter

The CBP plotter is exercised by a script that generates regular meshes of triangles and quadrilaterals over a planform known as “Cook’s wing” in the FEM literature. It is a flat trapezoid with the corner x-y coordinates

$$\{\{0,0\}, \{48,44\}, \{48,60\}, \{0,44\}\}$$

Regular meshes with n_x elements in the x direction and n_y in the y direction are constructed by the auxiliary modules listed in Figure G.15. Module `GenNodeCoordOverTrapezoid` generates the node coordinates using an isoparametric mapping scheme, and returns that array as function value. Meshes of 3-node triangles and 4-node quadrilaterals are generated by modules `GenTrig3NodeNumbersOverTrapezoid` and `GenQuad4NodeNumbersOverTrapezoid`, respectively. These return the element nodelist array as function values.

The CBP tester script is listed in G.16. The node coordinates and element nodelists are placed in `nodxyz` and `elenod`, respectively. The function to be plotted is the quadratic polynomial

$$f(x, y) = (x - 24)^2 + (y - 30)^2 - 200.$$

whose $f = C$ lines are circles centered at $x = 24$ and $y = 30$. Meshes are generated with $n_x=n_y=8$. The function is evaluated at the nodes of the generated mesh and placed into the `fn` array. The

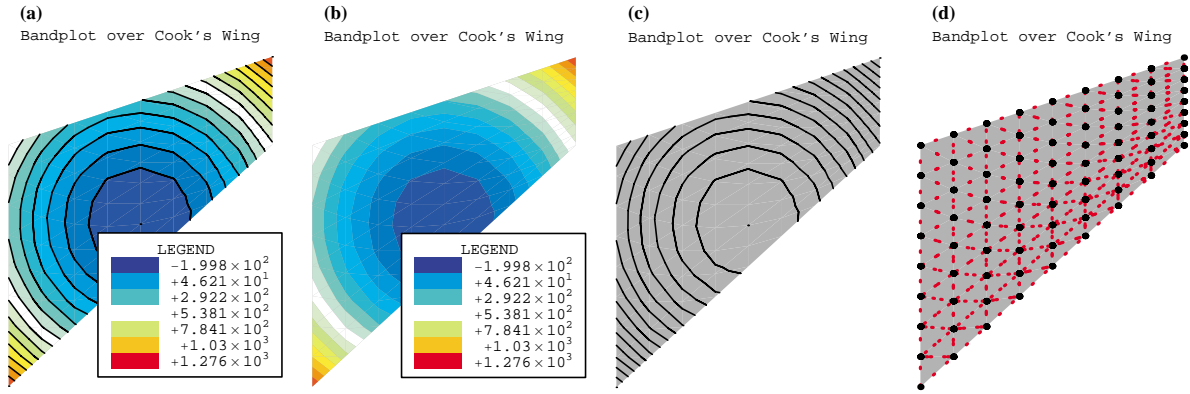


FIGURE G.17. Plots produced by the tester script of Figure G.16, over an 8×8 mesh of 3-node triangles.

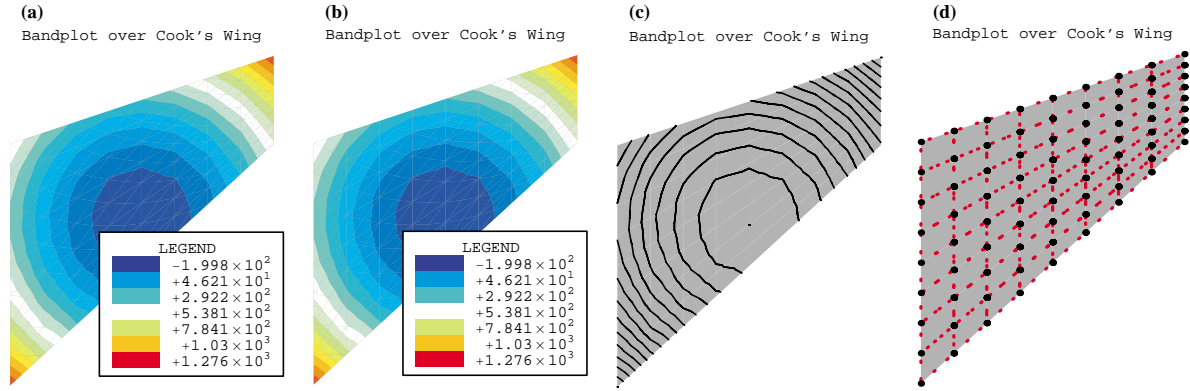


FIGURE G.18. Plots produced by the tester script of Figure G.16, over an 8×8 mesh of 4-node quadrilaterals.

minimum and maximum values are evaluated and placed into $\{f_{\min}, f_{\max}\}$, respectively, to define the plot range. The number of contour intervals is $ncint=15$. Four plots are generated for each mesh. These are shown in G.17 and G.18 for the 3-node triangle and the 4-node quadrilaterals, respectively. The plots are:

- (a) A CBP with contour lines drawn, plus legend.
- (b) A CBP without contour lines, plus legend.
- (c) A CLP without colors (grey background), and no legend.
- (d) A mesh plot showing node locations and element sides (dashed lines), and no legend.

One unexpected feature is the appearance of the legend. Its frame looks much bigger in the figures than on the screen. (The figures were created by saving *Mathematica* plot cells to EPS files and postprocessed through Illustrator.) This unresolved anomaly may require some modification in the plot object creation logic.

§G.3. Genesis and Evolution of Plotting Utilities

Graphic utilities presented in this Appendix had origins in Fortran code written over five decades ago, not long after FEM exploded beyond aerospace applications. The author developed in 1965 a FEM contour plotter called PSI6PL for thesis work at Berkeley [202] under Ray Clough. That thesis dealt with 3- and 6-node triangles for membrane and plate bending. PSI6PL was written in Fortran IV. It was a classical ink-jet contour plotter, meaning that contour lines were produced as segments drawn with a ink pen on a Calcomp plotter.⁴ Plot files were generated by a mainframe computer (at the time, an IBM 7094 that served the entire Berkeley campus) and stored on magnetic tape. The tape was later processed as an offline job, usually overnight.

PSI6PL was largely based on a similar program, called PSI3PL, written in Fortran II by Ed Wilson for his 1963 thesis [793], also under Ray Clough. (Wilson's FEM thesis program was called PSI, for Plane Stress Iteration, because the DSM solution process was iterative in nature.) The main extension for PSI6PL was to allow 6-node triangles. PSI3PL was likely the first FEM graphics tool developed in an academic environment. (Those programs seem to have been written at aerospace companies in the late 1950s, but there is no clear historical trace).

Over the following decades PSI6PL underwent drastic changes. An alphanumeric version called CONTPP in which the plot was done on a line printer (a "printer plot") was developed while at Boeing [738]. This one allowed triangular and quadrilateral elements with or without midside nodes. A version called MESHPP that plotted a FEM mesh was published in 1972 [204]. During the 1970s that kind of machine independent plotters gained popularity because their output could be directed to a alphanumeric terminal such as DEC's VT100 series, popularized with the advent of minicomputers. Loss of plot quality was compensated by human time saved. Once satisfactory results were obtained, higher quality offline plots could be requested.

Although vector-display, interactive graphic terminals were available by the late 1960s, high prices and limited software kept them out of the mainstream. The turning point was reached in the early 1980s, when the emergence of pixel-display screens, personal computers, digital fonts, and laser printers brought high-quality graphics to a wider audience. The appearance in the 1990s of high level languages such as *Matlab* and *Mathematica*, which included color plotting as intrinsic feature, finally allowed FEM graphics to become platform independent.⁵

Successors of the original Fortran codes were written in *Mathematica* and used in FEM course development since the mid 1990s. They are described in various Sections throughout this Appendix.

⁴ The Calcomp 565 drum plotter, introduced in 1959, was one of the first computer graphic output devices sold. The computer could control the rotation of an 11 inch wide drum within 0.01-in increments, as well as the horizontal movement of a pen holder over the drum. Paper rolls were 120-foot long. A pen was pressed by a spring against a paper scrolling over the drum. A solenoid could lift the pen off the paper. The arrangement allowed line drawings to be made under computer control. A metal bar above the take-up reel allowed the paper to be torn off and removed. The standard pen was a ball point, but liquid ink pens were also available for higher quality drawings. IBM sold the Calcomp 565 as the IBM 1627 for use in with its low-end scientific computers. Graphic output produced by batch jobs at high-end computers, such as the IBM 7094, was offloaded to tape and processed by the low-end one. Pictures of the Calcomp 565 (now in computer museums) may be seen in Google Images.

⁵ Fortran, like all low-level languages, never had platform independent graphics. It had to call machine-dependent libraries typically written in assembly language.

H

An Outline of MSA History

TABLE OF CONTENTS

	Page
§H.1. INTRODUCTION	H-3
§H.2. Background and Terminology	H-4
§H.3. Prolog - Victorian Artifacts: 1858-1930	H-6
§H.4. Act I - Gestation and Birth: 1930-1938	H-6
§H.4.1. The Source Papers	H-6
§H.4.2. The MSA Source Book	H-7
§H.5. Interlude I - WWII Blackout: 1938-1947	H-8
§H.6. Act II - The Matrix Forest: 1947-1956	H-8
§H.6.1. Computers Become Machines	H-8
§H.6.2. The Matrix CFM Takes Center Stage	H-8
§H.6.3. The Delta Wing Challenge	H-9
§H.6.4. Reduction Fosters Complexity	H-10
§H.6.5. Two Paths Through the Forest	H-11
§H.6.6. Dubious Duality	H-12
§H.7. Interlude II - Questions: 1956-1959	H-13
§H.8. Act III - Answers: 1959-1970	H-13
§H.8.1. A Path Outside the Forest	H-13
§H.8.2. The Fire Spreads	H-14
§H.8.3. The Final Test	H-15
§H.9. Epilogue - Revisiting the Past: 1970-date	H-16
§H.10. Concluding Remarks	H-16

A Historical Outline of Matrix Structural Analysis: A Play in Three Acts

C. A. FELIPPA

*Department of Aerospace Engineering Sciences
and Center for Aerospace Structures
University of Colorado, Boulder, CO 80309-0429, USA*

Report CU-CAS-00-14, June 2000; submitted for publication in *Computers & Structures*

Abstract

The evolution of Matrix Structural Analysis (MSA) from 1930 through 1970 is outlined. Highlighted are major contributions by Collar and Duncan, Argyris, and Turner, which shaped this evolution. To enliven the narrative the outline is configured as a three-act play. Act I describes the pre-WWII formative period. Act II spans a period of confusion during which matrix methods assumed bewildering complexity in response to conflicting demands and restrictions. Act III outlines the cleanup and consolidation driven by the appearance of the Direct Stiffness Method, through which MSA completed morphing into the present implementation of the Finite Element Method.

Keywords: matrix structural analysis; finite elements; history; displacement method; force method; direct stiffness method; duality

§H.1. INTRODUCTION

Who first wrote down a stiffness or flexibility matrix?

The question was posed in a 1995 paper [1]. The educated guess was “somebody working in the aircraft industry of Britain or Germany, in the late 1920s or early 1930s.” Since then the writer has examined reports and publications of that time. These trace the origins of Matrix Structural Analysis to the aeroelasticity group of the National Physics Laboratory (NPL) at Teddington, a town that has now become a suburb of greater London.

The present paper is an expansion of the historical vignettes in Section 4 of [1]. It outlines the major steps in the evolution of MSA by highlighting the fundamental contributions of four individuals: Collar, Duncan, Argyris and Turner. These contributions are lumped into three milestones:

Creation. Beginning in 1930 Collar and Duncan formulated discrete aeroelasticity in matrix form. The first two journal papers on the topic appeared in 1934-35 [2,3] and the first book, couthored with Frazer, in 1938 [4]. The representation and terminology for discrete dynamical systems is essentially that used today.

Unification. In a series of journal articles appearing in 1954 and 1955 [5] Argyris presented a formal unification of Force and Displacement Methods using dual energy theorems. Although practical applications of the duality proved ephemeral, this work systematized the concept of assembly of structural system equations from elemental components.

FEMinization. In 1959 Turner proposed [6] the Direct Stiffness Method (DSM) as an efficient and general computer implementation of the then embryonic, and as yet unnamed, Finite Element

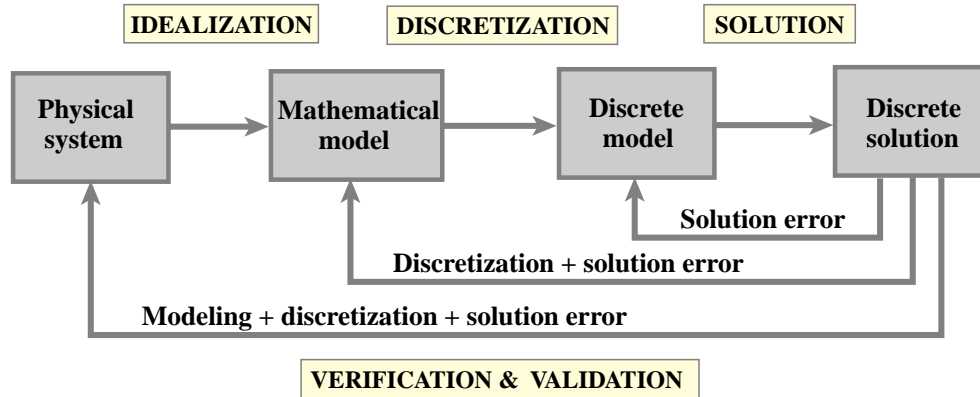


Figure 1. Flowchart of model-based simulation (MBS) by computer.

Method. This technique, fully explained in a follow-up article [7], naturally encompassed structural and continuum models, as well as nonlinear, stability and dynamic simulations. By 1970 DSM had brought about the demise of the Classical Force Method (CFM), and become the dominant implementation in production-level FEM programs.

These milestones help dividing MSA history into three periods. To enliven and focus the exposition these will be organized as three acts of a play, properly supplemented with a “matrix overture” prologue, two interludes and a closing epilogue. Here is the program:

Prologue - *Victorian Artifacts*: 1858–1930.

Act I - *Gestation and Birth*: 1930–1938.

Interlude I - *WWII Blackout*: 1938–1947.

Act II - *The Matrix Forest*: 1947–1956.

Interlude II - *Questions*: 1956–1959.

Act III - *Answers*: 1959–1970.

Epilogue - *Revisiting the Past*: 1970–date.

Act I, as well as most of the Prologue, takes place in the U.K. The following events feature a more international cast.

§H.2. Background and Terminology

Before departing for the theater, this Section offers some general background and explains historical terminology. Readers familiar with the subject should skip to Section 3.

The overall schematics of model-based simulation (MBS) by computer is flowcharted in Figure 1. For mechanical systems such as structures the Finite Element Method (FEM) is the most widely used discretization and solution technique. Historically the ancestor of FEM is MSA, as illustrated in Figure 2. The morphing of the MSA from the pre-computer era — as described for example in the first book [4] — into the first programmable computers took place, in wobbly gyrations, during the transition period herein called Act II. Following a confusing interlude, the young FEM begin to settle, during the early 1960s, into the configuration shown on the right of Figure 2. Its basic components have not changed since 1970.

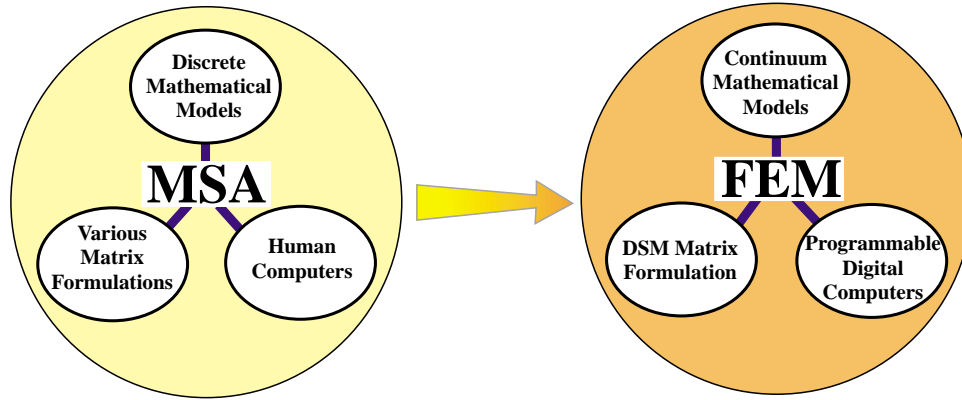


Figure 2. Morphing of the pre-computer MSA (before 1950) into the present FEM. On the left “human computer” means computations under direct human control, possibly with the help of analog devices (slide rule) or digital devices (desk calculator). The FEM configuration shown on the right settled by the mid 1960s.

MSA and FEM stand on three legs: mathematical models, matrix formulation of the discrete equations, and computing tools to do the numerical work. Of the three legs the latter is the one that has undergone the most dramatic changes. The “human computers” of the 1930s and 1940s morphed by stages into programmable computers of analog and digital type. The matrix formulation moved like a pendulum. It began as a simple displacement method in Act I, reached bewildering complexity in Act II and went back to conceptual simplicity in Act III.

Unidimensional structural models have changed little: a 1930 beam is still the same beam. The most noticeable advance is that pre-1955 MSA, following classical Lagrangian mechanics, tended to use spatially discrete energy forms from the start. The use of space-continuum forms as basis for multidimensional element derivation was pioneered by Argyris [5], successfully applied to triangular geometries by Turner, Clough, Martin and Topp [8], and finalized by Melosh [9] and Irons [10,11] with the precise statement of compatibility and completeness requirements for FEM.

Matrix formulations for MSA and FEM have been traditionally classified by the choice of *primary unknowns*. These are those solved for by the human or digital computer to determine the system state. In the Displacement Method (DM) these are physical or generalized displacements. In the Classical Force Method (CFM) these are amplitudes of redundant force (or stress) patterns. (The qualifier “classical” is important because there are other versions of the Force Method, which select for example stress function values or Lagrange multipliers as unknowns.) There are additional methods that involve combinations of displacements, forces and/or deformations as primary unknowns, but these have no practical importance in the pre-1970 period covered here.

Appropriate mathematical names for the DM are *range-space method* or *primal method*. This means that the primary unknowns are the same type as the primary variables of the governing functional. Appropriate names for the CFM are *null-space method*, *adjoint method*, or *dual method*. This means that the primary unknowns are of the same type of the adjoint variables of the governing functional, which in structural mechanics are forces. These names are not used in the historical outline, but are useful in placing more recent developments, as well as nonstructural FEM applications, within a general framework.

The terms Stiffness Method and Flexibility Method are more diffuse names for the Displacement and

Force Methods, respectively. Generally speaking these apply when stiffness and flexibility matrices, respectively, are important part of the modeling and solution process.

§H.3. Prolog - Victorian Artifacts: 1858-1930

Matrices — or “determinants” as they were initially called — were invented in 1858 by Cayley at Cambridge, although Gibbs (the co-inventor, along with Heaviside, of vector calculus) claimed priority for the German mathematician Grassmann. Matrix algebra and matrix calculus were developed primarily in the U.K. and Germany. Its original use was to provide a compact language to support investigations in mathematical topics such as the theory of invariants and the solution of algebraic and differential equations. For a history of these early developments the monograph by Muir [12] is unsurpassed. Several comprehensive treatises in matrix algebra appeared in the late 1920s and early 1930s [13–15].

Compared to vector and tensor calculus, matrices had relatively few applications in science and technology before 1930. Heisenberg’s 1925 matrix version of quantum mechanics was a notable exception, although technically it involved infinite matrices. The situation began to change with the advent of electronic desk calculators, because matrix notation provided a convenient way to organize complex calculation sequences. Aeroelasticity was a natural application because the stability analysis is naturally posed in terms of determinants of matrices that depend on a speed parameter.

The non-matrix formulation of Discrete Structural Mechanics can be traced back to the 1860s. By the early 1900s the essential developments were complete. A readable historical account is given by Timoshenko [16]. Interestingly enough, the term “matrix” never appears in this book.

§H.4. Act I - Gestation and Birth: 1930-1938

In the decade of World War I aircraft technology begin moving toward monoplanes. Biplanes disappeared by 1930. This evolution meant lower drag and faster speeds but also increased disposition to flutter. In the 1920s aeroelastic research began in an international scale. Pertinent developments at the National Physical Laboratory (NPL) are well chronicled in a 1978 historical review article by Collar [17], from which the following summary is extracted.

§H.4.1. The Source Papers

The aeroelastic work at the Aerodynamics Division of NPL was initiated in 1925 by R. A. Frazer. He was joined in the following year by W. J. Duncan. Two years later, in August 1928, they published a monograph on flutter [18], which came to be known as “The Flutter Bible” because of its completeness. It laid out the principles on which flutter investigations have been based since. In January 1930 A. R. Collar joined Frazer and Duncan to provide more help with theoretical investigations. Aeroelastic equations were tedious and error prone to work out in long hand. Here are Collar’s own words [17, page 17] on the motivation for introducing matrices:

“Frazer had studied matrices as a branch of applied mathematics under Grace at Cambridge; and he recognized that the statement of, for example, a ternary flutter problem in terms of matrices was neat and compendious. He was, however, more concerned with formal manipulation and transformation to other coordinates than with numerical results. On the other hand, Duncan and I were in search of numerical results for the vibration characteristics of airscrew blades; and we recognized that we could only advance by breaking the blade into, say, 10 segments and treating it as having 10 degrees

of freedom. This approach also was more conveniently formulated in matrix terms, and readily expressed numerically. Then we found that if we put an approximate mode into one side of the equation, we calculated a better approximation on the other; and the matrix iteration procedure was born. We published our method in two papers in *Phil. Mag.* [2,3]; the first, dealing with conservative systems, in 1934 and the second, treating damped systems, in 1935. By the time this had appeared, Duncan had gone to his Chair at Hull.”

The aforementioned papers appear to be the earliest *journal publications* of MSA. These are amazing documents: clean and to the point. They do not feel outdated. Familiar names appear: mass, flexibility, stiffness, and dynamical matrices. The matrix symbols used are $[m]$, $[f]$, $[c]$ and $[D] = [c]^{-1}[m] = [f][m]$, respectively, instead of the \mathbf{M} , \mathbf{F} , \mathbf{K} and \mathbf{D} in common use today. A general inertia matrix is called $[a]$. As befit the focus on dynamics, the displacement method is used. Point-mass displacement degrees of freedom are collected in a vector $\{x\}$ and corresponding forces in vector $\{P\}$. These are called $[q]$ and $[Q]$, respectively, when translated to generalized coordinates.

The notation was changed in the book [4] discussed below. In particular matrices are identified in [4] by capital letters without surrounding brackets, in more agreement with the modern style; for example mass, damping and stiffness are usually denoted by A , B and C , respectively.

§H.4.2. The MSA Source Book

Several papers on matrices followed, but apparently the traditional publication vehicles were not viewed as suitable for description of the new methods. At that stage Collar notes [17, page 18] that

“Southwell [Sir Richard Southwell, the “father” of relaxation methods] suggested that the authors of the various papers should be asked to incorporate them into a book, and this was agreed. The result was the appearance in November 1938 of “Elementary Matrices” published by Cambridge University Press [4]; it was the first book to treat matrices as a branch of applied mathematics. It has been reprinted many times, and translated into several languages, and even now after nearly 40 years, stills sells in hundreds of copies a year — mostly paperback. The interesting thing is that the authors did not regard it as particularly good; it was the book we were instructed to write, rather than the one we would have liked to write.”

The writer has copies of the 1938 and 1963 printings. No changes other than minor fixes are apparent. Unlike the source papers [2,3] the book feels dated. The first 245 pages are spent on linear algebra and ODE-solution methods that are now standard part of engineering and science curricula. The numerical methods, oriented to desk calculators, are obsolete. That leaves the modeling and application examples, which are not coherently interweaved. No wonder that the authors were not happy about the book. They had followed Southwell’s “merge” suggestion too literally. Despite these flaws its direct and indirect influence during the next two decades was significant. Being first excuses imperfections.

The book focuses on dynamics of a complete airplane and integrated components such as wings, rudders or ailerons. The concept of *structural element* is primitive: take a shaft or a cantilever and divide it into segments. The assembled mass, stiffness or flexibility is given directly. The source of damping is usually aerodynamic. There is no static stress analysis; pre-WWII aircraft were overdesigned for strength and typically failed by aerodynamic or propulsion effects.

Readers are reminded that in aeroelastic analysis stiffness matrices are generally unsymmetric, being the sum of a symmetric elastic stiffness and an unsymmetric aerodynamic stiffness. This clean decomposition does not hold for flexibility matrices because the inverse of a sum is not the sum

of inverses. The treatment of [4] includes the now called load-dependent stiffness terms, which represent another first.

On reading the survey articles by Collar [17,19] one cannot help being impressed by the lack of pretension. With Duncan he had created a tool for future generations of engineers to expand and improve upon. Yet he appears almost apologetic: “I will complete the matrix story as briefly as possible” [17, page 17]. The NPL team members shared a common interest: to troubleshoot problems by understanding the physics, and viewed numerical methods simply as helpers.

§H.5. Interlude I - WWII Blackout: 1938-1947

Interlude I is a “silent period” taken to extend from the book [4] to the first journal publication on the matrix Force Method for aircraft [20]. Aeroelastic research continued. New demands posed by high strength materials, higher speeds, combat maneuvers, and structural damage survival increased interest in stress analysis. For the beam-like skeletal configurations of the time, the traditional flexibility-based methods such as CFM were appropriate. Flexibilities were often measured experimentally by static load tests, and fitted into the calculations. Punched-card computers and relay-calculators were increasingly used, and analog devices relied upon to solve ODEs in guidance and ballistics. Precise accounts of MSA work in aerospace are difficult to trace because of publication restrictions. The blackout was followed by a 2-3 year hiatus until those restrictions were gradually lifted, R&D groups restaffed, and journal pipelines refilled.

§H.6. Act II - The Matrix Forest: 1947-1956

As Act II starts MSA work is still mainly confined to the aerospace community. But the focus has shifted from dynamics to statics, and especially stress, buckling, fracture and fatigue analysis. Turbines, supersonic flight and rocket propulsion brought forth thermomechanical effects. The Comet disasters forced attention on stress concentration and crack propagation effects due to cyclic cabin pressurization. Failsafe design gained importance. In response to these multiple demands aircraft companies staffed specialized groups: stress, aerodynamics, aeroelasticity, propulsion, avionics, and so on. A multilevel management structure with well defined territories emerged.

The transition illustrated in Figure 2 starts, driven by two of the legs supporting MSA: new computing resources and new mathematical models. The matrix formulation merely reacts.

§H.6.1. Computers Become Machines

The first electronic commercial computer: Univac I, manufactured by a division of Remington-Rand, appeared during summer 1951. The six initial machines were delivered to US government agencies [21]. It was joined in 1952 by the Univac 1103, a scientific-computation oriented machine built by ERA, a R-R acquisition. This was the first computer with a drum memory. T. J. Watson Sr., founder of IBM, had been once quoted as saying that six electronic computers would satisfy the needs of the planet. Turning around from that prediction, IBM launched the competing 701 model in 1953.

Big aircraft companies began purchasing or leasing these expensive wonders by 1954. But this did not mean immediate access for everybody. The behemoths had to be programmed in machine or assembly code by specialists, who soon formed computer centers allocating and prioritizing cycles. By 1956 structural engineers were still likely to be using their slides rules, Marchants and punched card equipment. Only after the 1957 appearance of the first high level language (Fortran I, offered on the IBM 704) were engineers and scientists able (and allowed) to write their own programs.

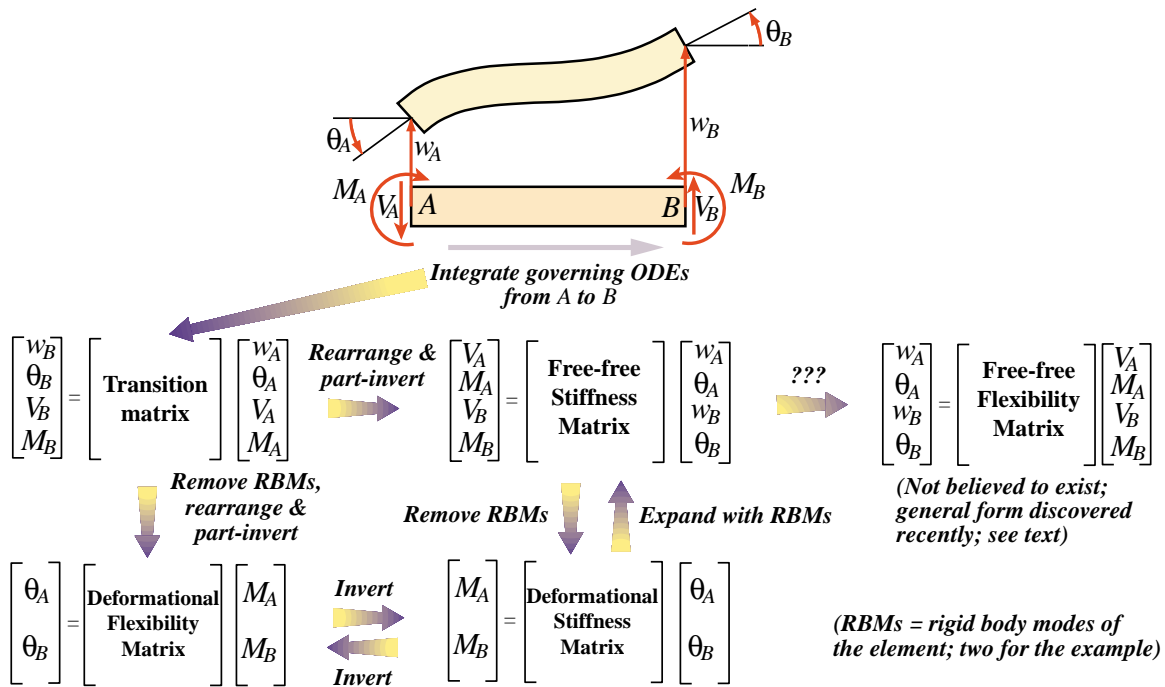


Figure 3. Transition, flexibility and stiffness matrices for unidimensional linear structural elements, such as the plane beam depicted here, can be obtained by integrating the governing differential equations, analytically or numerically, over the member to relate end forces and displacements. Clever things were done with this “method of lines” approach, such as including intermediate supports or elastic foundations.

§H.6.2. The Matrix CFM Takes Center Stage

In static analysis the non-matrix version of the Classical Force Method (CFM) had enjoyed a distinguished reputation since the source contributions by Maxwell, Mohr and Castigliano. The method provides directly the internal forces, which are of paramount interest in stress-driven design. It offers considerable scope of ingenuity to experienced structural engineers through clever selection of redundant force systems. It was routinely taught to Aerospace, Civil and Mechanical Engineering students.

Success in hand-computation dynamics depends on “a few good modes.” Likewise, the success of CFM depends crucially on the selection of good redundant force patterns. The structures of pre-1950 aircraft were a fairly regular lattice of ribs, spars and panels, forming beam-like configurations. If the panels are ignored, the selection of appropriate redundants was well understood. Panels were modeled conservatively as inplane shear-force carriers, circumventing the difficulties of two-dimensional elasticity. With some adjustments and experimental validations, sweptback wings of high aspect ratio were eventually fitted into these models.

A matrix framework was found convenient to organize the calculations. The first journal article on the matrix CFM, which focused on sweptback wing analysis, is by Levy [20], followed by publications of Rand [22], Langefors [23], Wehle and Lansing [24] and Denke[25]. The development culminates in the article series of Argyris [5] discussed in Section 6.5.

§H.6.3. The Delta Wing Challenge

The Displacement Method (DM) continued to be used for vibration and aeroelastic analysis, although as noted above this was often done by groups separated from stress and buckling analysis. A new modeling challenge entered in the early 1950s: delta wing structures. This rekindled interest in stiffness methods.

The traditional approach to obtain flexibility and stiffness matrices of unidimensional structural members such as bars and shafts is illustrated in Figure 3. The governing differential equations are integrated, analytically or numerically, from one end to the other. The end quantities, grouping forces and displacements, are thereby connected by a transition matrix. Using simple algebraic manipulations three more matrices shown in Figure 3 can be obtained: deformational flexibility, deformational stiffness and free-free stiffness. This well known technique has the virtue of reducing the number of unknowns since the integration process can absorb structural details that are handled in the present FEM with multiple elements.

Notably absent from the scheme of Figure 3 is the free-free flexibility. This was not believed to exist since it is the inverse of the free-free stiffness, which is singular. A general closed-form expression for this matrix as a Moore-Penrose generalized stiffness inverse was not found until recently [26,27].

Modeling delta wing configurations required two-dimensional panel elements of arbitrary geometry, of which the triangular shape, illustrated in Figure 4, is the simplest and most versatile. Efforts to follow the ODE-integration approach lead to failure. (One particularly bizarre proposal, for solving exactly the wrong problem, is mentioned for fun in the label of Figure 4.) This motivated efforts to construct the stiffness matrix of the panel directly. The first attempt in this direction is by Levy [28]; this was only partly successful but was able to illuminate the advantages of the stiffness approach.

The article series by Argyris [5] contains the derivation of the 8×8 free-free stiffness of a flat rectangular panel using bilinear displacement interpolation in Cartesian coordinates. But that geometry was obviously inadequate to model delta wings. The landmark contribution of Turner, Clough, Martin and Topp [8] finally succeeded in directly deriving the stiffness of a triangular panel. Clough [29] observes that this paper represents the delayed publication of 1952-53 work at Boeing. It is recognized as one of the two sources of present FEM implementations, the second being the DSM discussed later. Because of the larger number of unknowns compared to CFM, competitive use of the DM in stress analysis had necessarily to wait until computers become sufficiently powerful to handle hundreds of simultaneous equations.

§H.6.4. Reduction Fosters Complexity

For efficient digital computation on present computers, data organization (in terms of fast access as well as exploitation of sparseness, vectorization and parallelism) is of primary concern whereas raw problem size, up to certain computer-dependent bounds, is secondary. But for hand calculations minimal problem size is a key aspect. Most humans cannot comfortably solve by hand linear systems of more than 5 or 6 unknowns by direct elimination methods, and 5–10 times that through problem-oriented “relaxation” methods. The first-generation digital computers improved speed and reliability, but were memory strapped. For example the Univac I had 1000 45-bit words and the IBM 701, 2048 36-bit words. Clearly solving a full system of 100 equations was still a major challenge.

It should come as no surprise that problem reduction techniques were paramount throughout this period, and exerted noticeable influence until the early 1970s. In static analysis reduction was

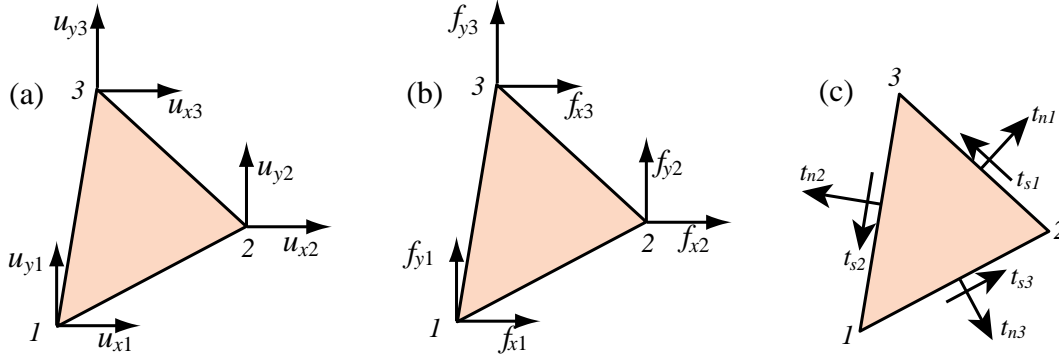


Figure 4. Modeling delta wing configurations required panel elements of arbitrary geometry such as the triangles depicted here. The traditional ODE-based approach of Figure 3 was tried by some researchers who (seriously) proposed finding the corner displacements in (a) produced by the concentrated corner forces in (b) on a supported triangle from the elasticity equations solved by numerical integration! Bad news: those displacements are infinite. Interior fields assumptions were inevitable, but problems persisted. A linear inplane displacement field is naturally specified by corner displacements, whereas a constant membrane force field is naturally defined by edge tractions (c). Those quantities “live” on different places. The puzzle was first solved in [8] by lumping edge tractions to node forces on the way to the free-free stiffness matrix.

achieved by elaborated *functional groupings* of static and kinematic variables. Most schemes of the time can be understood in terms of the following classification:

$$\begin{aligned}
 &\text{generalized forces} \quad \left\{ \begin{array}{l} \text{primary} \quad \left\{ \begin{array}{l} \text{applied forces } \mathbf{f}_a \\ \text{redundant forces } \mathbf{y} \end{array} \right. \\ \text{secondary} \quad \left\{ \begin{array}{l} \text{condensable forces } \mathbf{f}_c = \mathbf{0} \\ \text{support reactions } \mathbf{f}_s \end{array} \right. \end{array} \right. \\
 &\text{generalized displacements} \quad \left\{ \begin{array}{l} \text{primary} \quad \left\{ \begin{array}{l} \text{applied displacements } \mathbf{u}_a \\ \text{redundant displacements } \mathbf{z} \end{array} \right. \\ \text{secondary} \quad \left\{ \begin{array}{l} \text{condensable displacements } \mathbf{u}_c \\ \text{support conditions } \mathbf{u}_s = \mathbf{0} \end{array} \right. \end{array} \right.
 \end{aligned} \tag{H.1}$$

Here *applied forces* are those acting with nonzero values, that is, the ones visibly drawn as arrows by an engineer or instructor. In reduction-oriented thinking zero forces on unloaded degrees of freedom are classified as *condensable* because they can be removed through static condensation techniques. Similarly, nonzero *applied displacements* were clearly differentiated from zero-displacements arising from support conditions because the latter can be thrown out while the former must be retained. Redundant displacements, which are the counterpart of redundant forces, have been given many names, among them “kinematically indeterminate displacements” and “kinematic deficiencies.”

Matrix formulation evolved so that the unknowns were the force redundants \mathbf{y} in the CFM and the displacement redundants \mathbf{z} in the DM. Partitioning matrices in accordance to (H.1) fostered exuberant growth culminating in the *matrix forest* that characterizes works of this period.

To a present day FEM programmer familiar with the DSM, the complexity of the matrix forest would strike as madness. The DSM master equations can be assembled without functional labels. Boundary conditions are applied on the fly by the solver. But the computing limitations of the time must be kept in mind to see the method in the madness.

§H.6.5. Two Paths Through the Forest

A series of articles published by J. H. Argyris in four issues of *Aircraft Engrg.* during 1954 and 1955 collectively represents the second major milestone in MSA. In 1960 the articles were collected in a book, entitled “Energy Theorems and Structural Analysis” [5]. Part I, sub-entitled General Theory, reprints the four articles, whereas Part II, which covers additional material on thermal analysis and torsion, is co-authored by Argyris and Kelsey. Both authors are listed as affiliated with the Aerospace Department of the Imperial College at London.

The dual objectives of the work, stated in the Preface, are “to generalize, extend and unify the fundamental energy principles of elastic structures” and “to describe in detail practical methods of analysis of complex structures — in particular for aeronautical applications.” The first objective succeeds well, and represents a key contribution toward the development of continuum-based models. Part I carefully merges classical contributions in energy and work methods with matrix methods of discrete structural systems. The coverage is methodical, with numerous illustrative examples. The exposition of the Force Method for wing structures reaches a level of detail unequalled for the time.

The Displacement Method is then introduced by duality — called “analogy” in this work:

“The analogy between the developments for the flexibilities and stiffnesses ... shows clearly that parallel to the analysis of structures with forces as unknowns there must be a corresponding theory with deformations as unknowns.”

This section credits Ostenfeld [30] with being the first to draw attention to the parallel development. The duality is exhibited in a striking Form in Table II, in which both methods are presented side by side with simply an exchange of symbols and appropriate rewording. The steps are based on the following decomposition of internal deformation states \mathbf{g} and force patterns \mathbf{p} :

$$\mathbf{p} = \mathbf{B}_0 \mathbf{f}_a + \mathbf{B}_1 \mathbf{y}, \quad \mathbf{g} = \mathbf{A}_0 \mathbf{u}_a + \mathbf{A}_1 \mathbf{z}, \quad (\text{H.2})$$

Here the \mathbf{B}_i and \mathbf{A}_i denote system equilibrium and compatibility matrices, respectively. The vector symbols on the right reflect a particular choice of the force-displacement decomposition (H.1), with kinematic deficiencies taken to be the condensable displacements: $\mathbf{z} \equiv \mathbf{u}_c$.

This unification exerted significant influence over the next decade, particularly on the European community. An excellent textbook exposition is that of Pestel and Leckie [31]. This book covers both paths, following Argyris’ framework, in Chapters 9 and 10, using 83 pages and about 200 equations. These chapters are highly recommended to understand the organization of numeric and symbolic hand computations in vogue at that time, but it is out of print. Still in print (by Dover) is the book by Przemieniecki [32], which describes the DM and CFM paths in two Chapters: 6 and 8. The DM coverage is strongly influenced, however, by the DSM; thus duality is only superficially used.

§H.6.6. Dubious Duality

A key application of the duality in [5] was to introduce the DM by analogy to the then better known CFM. Although done with good intentions this approach did not anticipate the direct development of continuum-based finite elements through stiffness methods. These can be formulated from the total potential energy principle via shape functions, a technique not fully developed until the mid 1960s.

The side by side presentation of Table II of [5] tried to show that CFM and DM were going through exactly the same sequence of steps. Some engineers, eventually able to write Fortran programs,

concluded that the methods had similar capabilities and selecting one or the other was a matter of taste. (Most structures groups, upholding tradition, opted for the CFM.) But the few engineers who tried implementing both noticed a big difference. And that was before the DSM, which has no dual counterpart under the decomposition (H.2), appeared.

The paradox is explained in Section 4 of [1]. It is also noted there that (H.2) is not a particularly useful state decomposition. A better choice is studied in Section 2 of that paper; that one permits all known methods of Classical MSA, including the DSM, to be derived for skeletal structures as well as for a subset of continuum models.

§H.7. Interlude II - Questions: 1956-1959

Interlude I was a silent period dominated by the war blackout. Interlude II is more vocal: a time of questions. An array of methods, models, tools and applications is now on the table, and growing. Solid-state computers, Fortran, ICBMs, the first satellites. So many options. Stiffness or flexibility? Forces or displacements? Do transition matrix methods have a future? Is the CFM-DM duality a precursor to general-purpose programs that will simulate everything? Will engineers be allowed to write those programs?

As convenient milestone this outline takes 1959, the year of the first DSM paper, as the beginning of Act III. Arguments and counter-arguments raised by the foregoing questions will linger, however, for two more decades into diminishing circles of the aerospace community.

§H.8. Act III - Answers: 1959-1970

The curtain of Act III lifts in Aachen, Germany. On 6 November 1959, M. J. Turner, head of the Structural Dynamics Unit at Boeing and an expert in aeroelasticity, presented the first paper on the Direct Stiffness Method to an AGARD Structures and Materials Panel meeting [6]. (AGARD is NATO's Advisory Group for Aeronautical Research and Development, which had sponsored workshops and lectureships since 1952. Bound proceedings or reports are called AGARDographs.)

§H.8.1. A Path Outside the Forest

No written record of [6] seem to exist. Nonetheless it must have produced a strong impression since published contributions to the next (1962) panel meeting kept referring to it. By 1960 the method had been applied to nonlinear problems [33] using incremental techniques. In July 1962 Turner, Martin and Weikel presented an expanded version of the 1959 paper, which appeared in an AGARDograph volume published by Pergamon in 1964 [7]. Characteristic of Turner's style, the Introduction goes directly to the point:

"In a paper presented at the 1959 meeting of the AGARD Structures and Material Panel in Aachen, the essential features of a system for numerical analysis of structures, termed the direct-stiffness method, were described. The characteristic feature of this particular version of the displacement method is the assembly procedure, whereby the stiffness matrix for a composite structure is generated by direct addition of matrices associated with the elements of the structure."

The DSM is explained in six text lines and three equations:

"For an individual element e the generalized nodal force increments $\{\Delta X^e\}$ required to maintain a set of nodal displacement increments $\{\Delta u\}$ are given by a matrix equation

$$\{\Delta X^e\} = K^e \{\Delta u\} \quad (\text{H.3})$$

in which K^e denotes the stiffness matrix of the individual element. Resultant nodal force increments acting on the complete structure are

$$\{\Delta X\} = \sum \{\Delta X^e\} = K \{\Delta u\} \quad (\text{H.4})$$

wherein K , the stiffness of the complete structure, is given by the summation

$$K = \sum K^e \quad (\text{H.5})$$

which provides the basis for the matrix assembly procedure noted earlier.”

Knowledgeable readers will note a notational glitch. For (H.3)-(H.5) to be correct matrix equations, K^e must be an element stiffness fully expanded to global (in that paper: “basic reference”) coordinates, a step that is computationally unnecessary. A more suggestive notation used in present DSM expositions is $K = \sum (L^e)^T K^e L^e$, in which L^e are Boolean localization matrices. Note also the use of Δ in front of u and X and their identification as “increments.” This simplifies the extension to nonlinear analysis, as outlined in the next paragraph:

“For the solution of linear problems involving small deflections of a structure at constant uniform temperature which is initially stress-free in the absence of external loads, the matrices K^e are defined in terms of initial geometry and elastic properties of the materials comprising the elements; they remain unchanged throughout the analysis. Problems involving nonuniform heating of redundant structures and/or large deflections are solved in a sequence of linearized steps. Stiffness matrices are revised at the beginning of each step to account for changes in internal loads, temperatures and geometric configurations.”

Next are given some computer implementation details, including the first ever mention of user-defined elements:

“Stiffness matrices are generally derived in local reference systems associated with the elements (as prescribed by a set of subroutines) and then transformed to the basic reference system. It is essential that the basic program be able to accommodate arbitrary additions to the collection of subroutines as new elements are encountered. Associated with these are a set of subroutines for generation of stress matrices S^e relating matrices of stress components σ^e in the local reference system of nodal displacements:

$$\{\sigma^e\} = S^e \{\bar{u}\} \quad (\text{H.6})$$

The vector $\{\bar{u}\}$ denotes the resultant displacements relative to a local reference system which is attached to the element. ... Provision should also be made for the introduction of numerical stiffness matrices directly into the program. This permits the utilization and evaluation of new element representations which have not yet been programmed. It also provides a convenient mechanism for introducing local structural modifications into the analysis.”

The assembly rule (H.3)-(H.5) is insensitive to element type. It work the same way for a 2-node bar, or a 64-node hexahedron. To do dynamics and vibration one adds mass and damping terms. To do buckling one adds a geometric stiffness and solves the stability eigenproblem, a technique first explained in [33]. To do nonlinear analysis one modifies the stiffness in each incremental step. To apply multipoint constraints the paper [7] advocates a master-slave reduction method.

Some computational aspects are missing from this paper, notably the treatment of simple displacement boundary conditions, and the use of sparse matrix assembly and solution techniques. The latter were first addressed in Wilson’s thesis work [34,35].

§H.8.2. The Fire Spreads

DSM is a paragon of elegance and simplicity. The writer is able to teach the essentials of the method in three lectures to graduate and undergraduate students alike. Through this path the old MSA and the young FEM achieved smooth confluence. The matrix formulation returned to the crispness of the source papers [2,3]. A widely referenced MSA correlation study by Gallagher [36] helped dissemination. Computers of the early 1960s were finally able to solve hundreds of equations. In an ideal world, structural engineers should have quickly razed the forest and embraced DSM.

It did not happen that way. The world of aerospace structures split. DSM advanced first by word of mouth. Among the aerospace companies, only Boeing and Bell (influenced by Turner and Gallagher, respectively) had made major investments in DSM by 1965. Among academia the Civil Engineering Department at Berkeley became a DSM evangelist through Clough, who made his students — including the writer — use DSM in their thesis work. These codes were freely disseminated into the non-aerospace world since 1963. Martin established similar traditions at Washington University, and Zienkiewicz, influenced by Clough, at Swansea. The first textbook on FEM [37], which appeared in 1967, makes no mention of force methods. By then the application to non-structural field problems (thermal, fluids, electromagnetics, ...) had begun, and again the DSM scaled well into the brave new world.

§H.8.3. The Final Test

Legacy CFM codes continued, however, to be used at many aerospace companies. The split reminds one of Einstein's answer when he was asked about the reaction of the old-guard school to the new physics: "we did not convince them; we outlived them." Structural engineers hired in the 1940s and 1950s were often in managerial positions in the 1960s. They were set in their ways. How can duality fail? All that is needed are algorithms for having the computer select good redundants automatically. Substantial effort was spent in those "structural cutters" during the 1960s [32,38].

That tenacity was eventually put to a severe test. The 1965 NASA request-for-proposal to build the NASTRAN finite element system called for the simultaneous development of Displacement and Force versions [39]. Each version was supposed to have identical modeling and solution capabilities, including dynamics and buckling. Two separate contracts, to MSC and Martin, were awarded accordingly. Eventually the development of the Force version was cancelled in 1969. The following year may be taken as closing the transition depicted in Figure 2, and as marking the end of the Force Method as a serious contender for general-purpose FEM programs.

§H.9. Epilogue - Revisiting the Past: 1970-date

Has MSA, now under the wider umbrella of FEM, attained a final form? This seems the case for general-purpose FEM programs, which by now are truly “1960 heritage” codes.

Resurrection of the CFM for special uses, such as optimization, was the subject of a speculative technical note by the writer [40]. This was motivated by efforts of numerical analysts to develop sparse null-space methods [41–45]. That research appears to have been abandoned by 1990. Section 2 of [26] elaborates on why, barring unexpected breakthroughs, a resurrection of the CFM is unlikely.

A more modest revival involves the use of non-CFM *flexibility methods* for multilevel analysis. The structure is partitioned into subdomains or substructures, each of which is processed by DSM; but the subdomains are connected by Lagrange multipliers that physically represent node forces. A key driving application is massively parallel processing in which subdomains are mapped on distributed-memory processors and the force-based interface subproblem solved iteratively by FETI methods [46]. Another set of applications include inverse problems such as system identification and damage detection. Pertinent references and a historical sketch may be found in a recent article [47] that presents a hybrid variational formulation for this combined approach.

The true duality for structural mechanics is now known to involve displacements and stress functions, rather than displacements and forces. This was discovered by Fraeijs de Veubeke in the 1970s [48]. Although extendible beyond structures, the potential of this idea remains largely unexplored.

§H.10. Concluding Remarks

The patient reader who has reached this final section may have noticed that this essay is a critical overview of MSA history, rather than a recital of events. It reflects personal interpretations and opinions. There is no attempt at completeness. Only what are regarded as major milestones are covered in some detail. Furthermore there is only spotty coverage of the history of FEM itself as well as its computer implementation; this is the topic of an article under preparation for Applied Mechanics Reviews.

This outline can be hopefully instructive in two respects. First, matrix methods now in disfavor may come back in response to new circumstances. An example is the resurgence of flexibility methods in massively parallel processing. A general awareness of the older literature helps. Second, the sweeping victory of DSM over the befuddling complexity of the “matrix forest” period illustrates the virtue of Occam’s proscription against multiplying entities: when in doubt chose simplicity. This dictum is relevant to the present confused state of computational mechanics.

Acknowledgements

The present work has been supported by the National Science Foundation under award ECS-9725504. Thanks are due to the librarians of the Royal Aeronautical Society at London for facilitating access to archival copies of pre-WWII reports and papers. Feedback suggestions from early draft reviewers will be acknowledged in the final version.

References

- [1] C. A. Felippa, Parametrized unification of matrix structural analysis: classical formulation and d -connected elements, *Finite Elem. Anal. Des.*, **21**, pp. 45–74, 1995.

- [2] W. J. Duncan and A. R. Collar, A method for the solution of oscillations problems by matrices, *Phil. Mag.*, Series 7, **17**, pp. 865, 1934.
- [3] W. J. Duncan and A. R. Collar, Matrices applied to the motions of damped systems, *Phil. Mag.*, Series 7, **19**, pp. 197, 1935.
- [4] R. A. Frazer, W. J. Duncan and A. R. Collar, *Elementary Matrices, and some Applications to Dynamics and Differential Equations*, Cambridge Univ. Press, 1st ed. 1938, 7th (paperback) printing 1963.
- [5] J. H. Argyris and S. Kelsey, *Energy Theorems and Structural Analysis*, Butterworths, London, 1960; Part I reprinted from *Aircraft Engrg.* **26**, Oct-Nov 1954 and **27**, April-May 1955.
- [6] M. J. Turner, The direct stiffness method of structural analysis, Structural and Materials Panel Paper, AGARD Meeting, Aachen, Germany, 1959.
- [7] M. J. Turner, H. C. Martin and R. C. Weikel, Further development and applications of the stiffness method, AGARD Structures and Materials Panel, Paris, France, July 1962, in *AGARDograph 72: Matrix Methods of Structural Analysis*, ed. by B. M. Fraeijs de Veubeke, Pergamon Press, Oxford, pp. 203–266, 1964.
- [8] M. J. Turner, R. W. Clough, H. C. Martin, and L. J. Topp, Stiffness and deflection analysis of complex structures, *J. Aero. Sci.*, **23**, pp. 805–824, 1956.
- [9] R. J. Melosh, Bases for the derivation of matrices for the direct stiffness method, *AIAA J.*, **1**, pp. 1631–1637, 1963.
- [10] B. M. Irons, Comments on ‘Matrices for the direct stiffness method’ by R. J. Melosh, *AIAA J.*, **2**, p. 403, 1964.
- [11] B. M. Irons, Engineering application of numerical integration in stiffness methods, *AIAA J.*, **4**, pp. 2035–2037, 1966.
- [12] T. Muir, *The History of Determinants in the Historical Order of Development*, Vols I-IV, MacMillan, London, 1906–1923.
- [13] H. W. Turnbull, *The Theory of Determinants, Matrices and Invariants*, Blackie & Sons Ltd., London, 1929; reprinted by Dover Pubs., 1960.
- [14] C. C. MacDuffee, *The Theory of Matrices*, Springer, Berlin, 1933; Chelsea Pub. Co., New York, 1946.
- [15] T. Muir and W. J. Metzler, *A Treatise on the Theory of Determinants*, Longmans, Greens & Co., London and New York, 1933.
- [16] S. P. Timoshenko, *History of Strength of Materials*, McGraw-Hill, New York, 1953 (Dover edition 1983).
- [17] A. R. Collar, The first fifty years of aeroelasticity, *Aerospace*, pp. 12–20, February 1978.
- [18] R. A. Frazer and W. J. Duncan, *The Flutter of Airplane Wings*, Reports & Memoranda 1155, Aeronautical Research Committee, London, 1928.
- [19] A. R. Collar, Aeroelasticity, retrospect and prospects, *J. Royal Aeronautical Society*, **63**, No. 577, pp. 1–17, January 1959.
- [20] S. Levy, Computation of influence coefficients for aircraft structures with discontinuities and sweepback, *J. Aero. Sci.*, **14**, pp. 547–560, 1947.
- [21] P. E. Ceruzzi, *A History of Modern Computing*, The MIT Press, Cambridge, MA, 1998.
- [22] T. Rand, An approximate method for computation of stresses in sweptback wings, *J. Aero. Sci.*, **18**, pp. 61–63, 1951.
- [23] B. Langefors, Analysis of elastic structures by matrix coefficients, with special regard to semimonocoque structures, *J. Aero. Sci.*, **19**, pp. 451–458, 1952.
- [24] L. B. Wehle and W. Lansing, A method for reducing the analysis of complex redundant structures to a routine procedure, *J. Aero. Sci.*, **19**, pp. 677–684, 1952.

- [25] P. H. Denke, A matrix method of structural analysis, *Proc. 2nd U.S. Natl. Cong. Appl. Mech*, ASCE, pp. 445-457, 1954.
- [26] C. A. Felippa and K. C. Park, A direct flexibility method, *Comp. Meths. Appl. Mech. Engrg.*, **149**, 319–337, 1997.
- [27] C. A. Felippa, K. C. Park and M. R. Justino F., The construction of free-free flexibility matrices as generalized stiffness inverses, *Computers & Structures*, **68**, pp. 411–418, 1998.
- [28] S. Levy, Structural analysis and influence coefficients for delta wings, *J. Aero. Sci.*, **20**, pp. 677–684, 1953.
- [29] R. W. Clough, The finite element method – a personal view of its original formulation, in *From Finite Elements to the Troll Platform - the Ivar Holand 70th Anniversary Volume*, ed. by K. Bell, Tapir, Trondheim, Norway, pp. 89–100, 1994.
- [30] A. Ostenfeld, *Die Deformationmethode*, Springer, Berlin, 1926.
- [31] E. C. Pestel and F. A. Leckie, *Matrix Methods in Elastomechanics*, McGraw-Hill, New York, 1963.
- [32] J. S. Przemieniecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, 1968 (Dover edition 1986).
- [33] M. J. Turner, E. H. Dill, H. C. Martin and R. J. Melosh, Large deflection analysis of complex structures subjected to heating and external loads, *J. Aero. Sci.*, **27**, pp. 97-107, 1960.
- [34] E. L. Wilson, Finite element analysis of two-dimensional structures, *Ph. D. Dissertation*, Department of Civil Engineering, University of California at Berkeley, 1963.
- [35] E. L. Wilson, Automation of the finite element method — a historical view, *Finite Elem. Anal. Des.*, **13**, pp. 91–104, 1993.
- [36] R. H. Gallaguer, *A Correlation Study of Methods of Matrix Structural Analysis*, Pergamon, Oxford, 1964.
- [37] O. C. Zienkiewicz and Y. K. Cheung, *The Finite Element Method in Structural and Soil Mechanics*, McGraw Hill, London, 1967.
- [38] J. Robinson, *Structural Matrix Analysis for the Engineer*, Wiley, New York, 1966.
- [39] R. H. MacNeal, *The MacNeal Schwendler Corporation: The First Twenty Years*, Gardner Litograph, Buena Park, CA, 1988.
- [40] C. A. Felippa, Will the force method come back?, *J. Appl. Mech.*, **54**, pp. 728–729, 1987.
- [41] M. W. Berry, M. T. Heath, I. Kaneko, M. Lawo, R. J. Plemmons and R. C. Ward, An algorithm to compute a sparse basis of the null space, *Numer. Math.*, **47**, pp. 483–504, 1985.
- [42] I. Kaneko and R. J. Plemmons, Minimum norm solutions to linear elastic analysis problems, *Int. J. Numer. Meth. Engrg.*, **20**, pp. 983–998, 1984.
- [43] J. R. Gilbert and M. T. Heath, Computing a sparse basis for the null space, *SIAM J. Alg. Disc. Meth.*, **8**, pp. 446–459, 1987.
- [44] T. F. Coleman and A. Pothén, The null space problem: II. Algorithms, *SIAM J. Alg. Disc. Meth.*, **8**, pp. 544–563, 1987.
- [45] R. J. Plemmons and R. E. White, Substructuring methods for computing the nullspace of equilibrium matrices, *SIAM J. Matrix Anal. Appl.*, **1**, pp. 1–22, 1990.
- [46] C. Farhat and F. X. Roux, Implicit Parallel Processing in Structural Mechanics, *Computational Mechanics Advances*, **2**, No. 1, pp. 1–124, 1994.
- [47] K. C. Park and C. A. Felippa, A variational principle for the formulation of partitioned structural systems, *Int. J. Numer. Meth. Engrg.*, **47**, 395–418, 2000.
- [48] B. M. Fraeijs de Veubeke, Stress function approach, *Proc. World Congr. on Finite Element Methods*, October 1975, Woodlands, England; reprinted in *B. M. Fraeijs de Veubeke Memorial Volume of Selected Papers*, ed. by M. Geradin, Sithoff & Noordhoff, Alphen aan den Rijn, The Netherlands, pp. 663–715, 1980.

M

Converting IOMoDE to FOMoDE

TABLE OF CONTENTS

	Page
§M.1. Introduction	M–3
§M.2. A Fourier First	M–3
§M.3. Example 2: IOMoDE with Even Derivatives	M–3
§M.3.1. *A More Advanced Derivation	M–4
§M.4. Example 3: IOMoDE with Even and Odd Derivatives	M–5
§M.4.1. Example 3: IOMoDE with All Derivatives	M–6

§M.1. Introduction

This Appendix summarizes transformations that find application in the method of Modified Differential Equations or MoDE. technically the most difficult operation is passing from an Infinite Order MoDE (IOMoDE) to a Finite Order MoDE (FOMoDE). There is no general method because involves the process involves identification of series. Case by case is the rule. Nonetheless there are some forms which recur in application problems and which may be processed in closed form. Such forms are collected here as a hub for references from other chapters.

§M.2. A Fourier First

The inhomogeneous, even-derivative, infinite-order ODE

$$f(x) = u(x) - \frac{1}{n^2}u''(x) + \frac{1}{n^4}u''''(x) + \dots, \quad (\text{M.1})$$

provides a simple example of infinite-to-finite reduction. Here as usual $(.)'$ is an abbreviation for $d(.) / dx$ while n is a nonzero constant. It is assumed that $u(x) \in C^\infty$ for any finite x whereas $f(x) \in C^2$. The reduction is immediate. By inspection (M.1) satisfies

$$u(x) = f(x) + \frac{1}{n^2}f''(x). \quad (\text{M.2})$$

so the FOMoDE (M.2) is of zero order. This easy result is not relevant to modification methods, but allows the introduction of a historical curiosity.

Joseph Fourier used the transformation (M.1)→(M.2), in which n is an integer, on the way to finding a formula for the coefficients of what are now called Fourier series [262, pp. 187ff]. If $u(x)$ is given, the general solution of (M.2) is

$$f(x) = C \cos nx + D \sin nx + n \sin nx \int_0^x u(t) \cos nt \, dt - n \cos nx \int_0^x u(t) \sin nt \, dt. \quad (\text{M.3})$$

From this Fourier's famous formula emerges after a few more gyrations. Euler had effortlessly found the same result 70 years earlier through term-by-term integration. But in taking the hard way Fourier became apparently the first person to use an ODE of infinite order.

Hardy [328, §2.10], a gentle spirit, benignly observes that Fourier's laborious derivations rely on divergent series *in passim* but that his final result turned out to be correct. Truesdell, a stickler for mathematical orthodoxy, is not so kind: "Fourier proved it [the fundamental theorem for expansion in trigonometric series] through a mass of divergent gobbledegook which every competent mathematician of his own day rejected." [722, p. 77].¹

¹ Of course Fourier could retort that his name is well known in sciences and mathematics after two centuries. The ratio of Google hits for Fourier to those for Truesdell is 65:1 on June 4, 2012.

§M.3. Example 2: IOMoDE with Even Derivatives

As first nontrivial example consider the homogeneous, even-derivative, infinite-order ODE:

$$-\frac{\phi}{2a^2}u(x) + \frac{1}{2!}u''(x) + \frac{a^2\chi^2}{4!}u''''(x) + \frac{a^4\chi^4}{6!}u''''''(x) + \dots = 0, \quad a > 0, \quad \phi \neq 0, \quad 0 < \chi \leq 1. \quad (\text{M.4})$$

Here ϕ and χ are dimensionless real parameters whereas a , which is a characteristic problem dimension, has dimension of length. The reduction to finite order can be obtained by a variant of Warming and Hyett's [748] derivative elimination procedure, Differentiate (M.4) $2(n-1)$ times ($n = 1, 2, \dots$) with respect to x while discarding all odd derivatives. Truncate to the same level in χ , and set up a linear system in the even derivatives u'', u'''' , \dots . The configuration of the elimination system is illustrated for $n = 4$:

$$\begin{bmatrix} 1/2! & a^2\chi^2/4! & a^4\chi^4/6! & a^6\chi^6/8! \\ -\frac{1}{2}\phi a^{-2} & 1/2! & a^2\chi^2/4! & a^4\chi^4/6! \\ 0 & -\frac{1}{2}\phi a^{-2} & 1/2! & a^2\chi^2/4! \\ 0 & 0 & -\frac{1}{2}\phi a^{-2} & 1/2! \end{bmatrix} \begin{bmatrix} u'' \\ u'''' \\ u'''''' \\ u'''''''' \end{bmatrix} = \begin{bmatrix} \frac{1}{2}a^{-2}\phi u \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (\text{M.5})$$

The coefficient matrix of this system is Toeplitz and Hessenberg but not Hermitian. This can be solved for u'' to yield a truncated FOMoDE. Solving (M.5) and expanding in Taylor series gives

$$u'' = \frac{56\phi(360 + 60\lambda + \lambda^2)u}{20160 + 5040\lambda + 252\lambda^2 + \lambda^3} = \phi \left(1 - \frac{1}{12}\lambda + \frac{1}{90}\lambda^2 + \dots \right) u. \quad (\text{M.6})$$

where $\lambda = a^2\chi^2\phi$. Increasing n , the coefficients of the power series in λ are found to be generated by the recursion $c_1 = 1$, $c_{n+1} = -\frac{1}{2}n^2c_n/[(n+1)(2n+1)]$, $n \geq 1$, which produces the sequence $\{1, -1/12, 1/90, -1/560, 1/3150, -1/16632, \dots\}$. The generating function [?] can be found by *Mathematica*'s package `RSolve` by entering `<<DiscreteMath`RSolve`;`
`g=GeneratingFunction[a[n+1]==-n*n/(2*(n+1)*(2*n+1))*a[n], a[1]==1, a[n], n, λ];`
`Print[g].` To verify the answer do `Print[Series[g,{λ,0,8}]]`. The result is

$$\frac{4}{\lambda} \left(\operatorname{arcsinh} \frac{\sqrt{\lambda}}{2} \right)^2 = 1 - \frac{\lambda}{12} + \frac{\lambda^2}{90} - \frac{\lambda^3}{560} + \frac{\lambda^4}{3150} - \frac{\lambda^5}{16632} + \frac{\lambda^6}{84084} - \frac{\lambda^7}{411840} + \dots \quad (\text{M.7})$$

This yields the second-order FOMoDE

$$u'' = \frac{4}{a^2\chi^2} \left(\operatorname{arcsinh} \frac{\sqrt{\lambda}}{2} \right)^2 u = \frac{4}{a^2\chi^2} \left(\operatorname{arcsinh} \frac{\chi\sqrt{\phi}}{2} \right)^2 u. \quad (\text{M.8})$$

To give an example of matching suppose that the original ODE from which (M.4) comes is $u'' = (w/a^2)u$, where $w > 0$ is constant. For nodal exactness, $w = (4/\chi^2)(\operatorname{arcsinh}(\frac{1}{2}\chi\sqrt{\phi}))^2$. If ϕ is the free parameter, solving for it gives

$$\phi = \frac{4}{\chi^2} \left(\sinh \frac{\chi\sqrt{w}}{2} \right)^2 = \frac{2(\cosh(\chi\sqrt{w}) - 1)}{\chi^2}. \quad (\text{M.9})$$

In this analysis no term of model equation is assumed to be small. The procedure for handling a forcing term $f(x)$ follows the same technique.

§M.3.1. *A More Advanced Derivation

The foregoing construction of (M.8) has a heuristic flavor: it relies on recognizing a series. A more direct derivation, which however requires more advanced mathematical tools, is presented here. The method relies on the following determinant theorem [484, p. 704]. Given the formal series expansion

$$\frac{1}{g(x)} = \frac{1}{a_0 + a_1x + a_2x^2 + a_3x^3 + \dots} = A_0 - A_1x + A_2x^2 - A_3x^3 + \dots, \quad a_0 \neq 0, \quad (\text{M.10})$$

then the Toeplitz determinants formed with the a_i coefficients satisfy

$$A_1 = a_0^{-1} |a_1|, \quad A_2 = a_0^{-2} \begin{vmatrix} a_1 & a_2 \\ a_0 & a_1 \end{vmatrix}, \quad A_3 = a_0^{-3} \begin{vmatrix} a_1 & a_2 & a_3 \\ a_0 & a_1 & a_2 \\ 0 & a_0 & a_1 \end{vmatrix}, \quad A_4 = a_0^{-4} \begin{vmatrix} a_1 & a_2 & a_3 & a_4 \\ a_0 & a_1 & a_2 & a_3 \\ 0 & a_0 & a_1 & a_2 \\ 0 & 0 & a_0 & a_1 \end{vmatrix}, \dots \quad (\text{M.11})$$

with $A_0 = 1/a_0$. Now the determinants that appear in the foregoing FOMoDE derivation have the form

$$A_1 = |1/2!|, \quad A_2 = \begin{vmatrix} 1/2! & a^2\chi^2/4! \\ -\frac{1}{2}\phi a^{-2} & 1/2! \end{vmatrix}, \quad A_3 = \begin{vmatrix} 1/2! & a^2\chi^2/4! & a^4\chi^4/6! \\ -\frac{1}{2}\phi a^{-2} & 1/2! & a^2\chi^2/4! \\ 0 & -\frac{1}{2}\phi a^{-2} & 1/2! \end{vmatrix}, \quad (\text{M.12})$$

Identifying to (M.10) and (M.11) one obtains by inspection

$$a^2\chi^2 g(x) = \cosh(a\chi\sqrt{x}) - (1 + \frac{1}{2}\phi\chi^2) \quad (\text{M.13})$$

The n^{th} approximation to the FOMoDE ($n > 1$) is $u'' = C_n u$, with $C_n = A_{n-1}/A_n$. If the series (M.10) has radius of convergence R , then $C_n \rightarrow 1/R$ as $n \rightarrow \infty$. But the radius of convergence of $1/g(x)$ is the distance from $x = 0$ to the closest pole, or what is the the same, the closest zero of $g(x)$. This is obtained by solving $g(R) = 0$ or $\cosh(a\chi\sqrt{R}) = 1 + \frac{1}{2}\phi\chi^2$, whence $a\chi\sqrt{R} = \text{arccosh}(1 + \frac{1}{2}\phi\chi^2)$, which for $\phi > 0$ is equivalent to $R = 4a^{-2}\chi^{-2}(\text{arcsinh}(\frac{1}{2}\sqrt{\phi}\chi))^2$. This leads to the same solution: $u'' = Ru = 4a^{-2}\chi^{-2}(\text{arcsinh}(\frac{1}{2}\sqrt{\phi}\chi))^2$ found before.

Note that this method bypasses determinant expansions and series identification, but it is restricted to Toeplitz matrices.

§M.4. Example 3: IOMoDE with Even and Odd Derivatives

As second example consider the homogeneous infinite-order ODE:

$$-\frac{\phi}{2a}u(x) + u'(x) + \frac{a}{2!}u''(x) + \frac{a^2\chi^2}{3!}u'''(x) + \frac{a^3\chi^3}{4!}u''''(x) + \dots = 0, \quad a > 0, \quad \phi \neq 0, \quad 0 < \chi \leq 1. \quad (\text{M.14})$$

Here ϕ and χ are dimensionless real parameters whereas a , which is a characteristic problem dimension, has dimension of length. Proceeding as above one forms the elimination system,

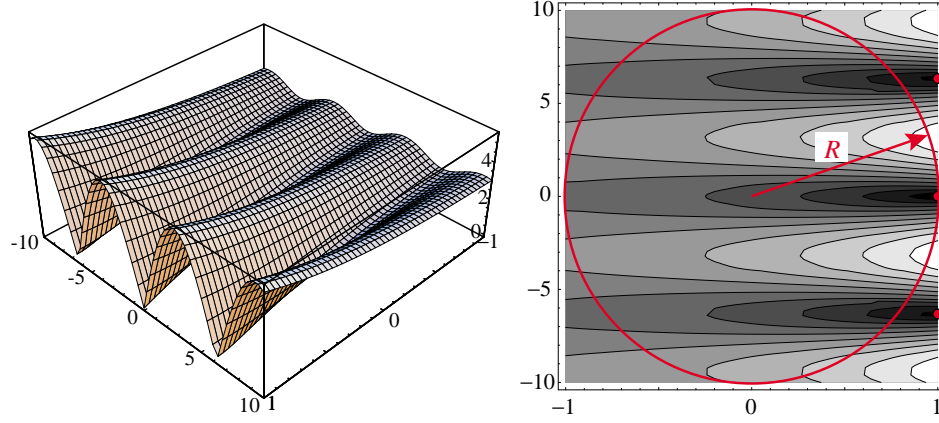


FIGURE M.1. Plot of the modulus $\sqrt{g_r^2 + g_i^2}$ of generating function (M.16) for $\{\phi = e - 1, a = \chi = 1\}$, showing zeros on line $x = x_R = 1$ and convergence radius.

illustrated for $n = 4$:

$$\begin{bmatrix} 1 & a\chi/2! & a^2\chi^2/3! & a^3\chi^3/4! \\ -\frac{1}{2}\phi/a & 1 & a\chi/2! & a^2\chi^2/3! \\ 0 & -\frac{1}{2}\phi/a & 1 & a\chi/2! \\ 0 & 0 & -\frac{1}{2}\phi/a & 1 \end{bmatrix} \begin{bmatrix} u' \\ u'' \\ u''' \\ u'''' \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\phi u/a \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (\text{M.15})$$

This is a Toeplitz matrix, so the quickest procedure is the determinant theorem. By inspection

$$a\chi g(x) = \exp(a\chi x) - (1 + \phi\chi). \quad (\text{M.16})$$

The radius of convergence of the $1/g(z)$ series is obtained by solving $g(R) = 0$ for a complex $g(z)$, $z = x + yi$. The real and imaginary parts of $g(z)$ are $a\chi g_r = \exp(ax\chi) \cos(ay\chi) - (1 + \phi\chi)$, $a\chi g_i = \exp(ax\chi) \sin(ay\chi)$. Solving $g_r = g_i = 0$ gives the solutions $x_R = \log(1 + \phi\chi)/(a\chi)$, $y_R = \pi j/(a\chi)$, where j is an arbitrary integer. Thus all zeros lie on the $x = x_R$ line. This is pictured in Figure M.1, drawn for $\phi = e - 1$, $a = \chi = 1$ so $x_R = 1$. The zero closest to the origin corresponds to $y_R = 0$ whence the radius of convergence is $R = x_R = \log(1 + \phi\chi)/(a\chi)$, and the FOMoDE is

$$u' = \frac{\log(1 + \phi\chi)}{a\chi} u. \quad (\text{M.17})$$

In this analysis no term of model equation is assumed to be small. The procedure for handling a forcing term $f(x)$ follows the same technique.

§M.4.1. Example 3: IOMoDE with All Derivatives

$$u'(t) + \frac{h}{2!}u''(t) + \frac{h^2}{3!}u'''(t) + \dots = \lambda u(t). \quad (\text{M.18})$$

To derive the FOMoDE, differentiate repeatedly with respect to t truncating after the same order in h , taking care to obtain a square coefficient matrix. For example, differentaiting 3 times gives

$$\begin{bmatrix} 1 & h/2! & h^2/3! & h^3/4! \\ -\lambda & 1 & h/2! & h^2/3! \\ 0 & -\lambda & 1 & h/2! \\ 0 & 0 & -\lambda & 1 \end{bmatrix} \begin{bmatrix} u' \\ u'' \\ u''' \\ u'''' \end{bmatrix} = \begin{bmatrix} \lambda u \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (\text{M.19})$$

Solving produces

$$u' = \frac{4\lambda(6 + 6h\lambda + h^2\lambda^2)}{24 + 36h\lambda + 14h^2\lambda^2 + h^3\lambda^3} = \left(\lambda - \frac{h\lambda^2}{2} + \frac{h^2\lambda^3}{3} - \frac{h^3\lambda^4}{4} + \frac{29h^4\lambda^5}{144} - \dots \right) u \quad (\text{M.20})$$

The series on the right looks like that of $\log(1 + h\lambda)$, which can be confirmed by increasing n . Therefore the FOMoDE has the explicit form

$$u' = \frac{\log(1 + h\lambda)}{h} u \quad (\text{M.21})$$

The explicit solution of this equation under the initial condition $u(0) = u_0$ is of course $u(t/h) = u_0(1 + h\lambda)^{(t/h)}$. As $h \rightarrow 0$ this approaches $u_0 e^{\lambda t}$, which confirms the consistency of Forward Euler.

O

The Origins of the Finite Element Method

TABLE OF CONTENTS

	Page
§O.1. Introduction	O–3
§O.1.1. Who Invented Finite Elements?	O–3
§O.1.2. G1: The Pioneers	O–3
§O.1.3. G2: The Golden Age	O–4
§O.1.4. G3: Consolidation	O–4
§O.1.5. G4: Back to Basics	O–5
§O.1.6. Precursors	O–5

§O.1. Introduction

This Appendix summarizes the history of structural finite elements since 1950 to date. It functions as a hub for chapter-dispersed historical references.

For exposition convenience, structural “finitelementology” may be divided into four generations that span 10 to 15 years each. There are no sharp intergenerational breaks, but noticeable change of emphasis. The following summary does not cover the conjoint evolution of Matrix Structural Analysis into the Direct Stiffness Method from 1934 through 1970. This was the subject of a separate essay [233], which is also given in Appendix H.

§O.1.1. Who Invented Finite Elements?

Not just one individual, as this historical sketch will make clear. But if the question is tweaked to: who created the FEM in everyday use? there is no question in the writer’s mind: M. J. (Jon) Turner at Boeing over the period 1950–1962. He generalized and perfected the Direct Stiffness Method, and forcefully got Boeing to commit resources to it while other aerospace companies were mired in the Force Method. During 1952–53 he oversaw the development of the first continuum based finite elements. In addition to Turner, major contributors to current practice include: B. M. Irons, inventor of isoparametric models, shape functions, the patch test and frontal solvers; R. J. Melosh, who recognized the Rayleigh-Ritz link and systematized the variational derivation of stiffness elements; and E. L. Wilson, who developed the first open source (and widely imitated and distributed) FEM software.

All of these pioneers were in the aerospace industry at least during part of their careers. That is not coincidence. FEM is the confluence of three ingredients, one of which is digital computation. And only large industrial companies (as well as some government agencies) were able to afford mainframe computers during the 1950s.

Who were the popularizers? Four academicians: J. H. Argyris, R. W. Clough, H. C. Martin, and O. C. Zienkiewicz are largely responsible for the “technology transfer” from the aerospace industry to a wider range of engineering applications during the 1950s and 1960s. The first three learned the method from Turner directly or indirectly. As a consultant to Boeing in the early 1950s, Argyris, a Force Method expert then at Imperial College, received reports from Turner’s group, and weaved the material into his influential 1954 serial [22]. To Argyris goes the credit of being the first in constructing a displacement-assumed continuum element [22, p. 62].

Clough and Martin, then junior professors at U.C. Berkeley and U. Washington, respectively, spent “faculty internship” summers at Turner’s group during 1952 and 1953. The result of this seminal collaboration was a celebrated paper [727], widely considered the start of the present FEM. Clough baptized the method in 1960 [134] and went on to form at Berkeley the first research group to propel the idea into Civil Engineering applications. Olek Zienkiewicz, originally an expert in finite difference methods who learned the trade from Southwell, was convinced in 1964 by Clough to try FEM. He went on to write the first textbook on the subject [785] and to organize another important Civil Engineering research group in the University of Wales at Swansea.

§O.1.2. G1: The Pioneers

The 1956 paper by Turner, Clough, Martin and Topp [727], henceforth abbreviated to TCMT, is recognized as the start of the current FEM, as used in the overwhelming majority of commercial

codes. Along with Argyris' serial [22] they prototype the first generation, which spans 1950 through 1962. A panoramic picture of this period is available in two textbooks [553,575]. Przemieniecki's text is still reprinted by Dover. The survey by Gallagher [280] was influential at the time but is now difficult to access outside libraries.

The pioneers were structural engineers, schooled in classical mechanics. They followed a century of tradition in regarding structural elements as a device to transmit forces. This "element as force transducer" was the standard view in pre-computer structural analysis. It explains the use of flux assumptions to derive stiffness equations in TCMT. Element developers worked in, or interacted closely with, the aircraft industry. (As noted above, only large aerospace companies were then able to afford mainframe computers.) Accordingly they focused on thin structures built up with bars, ribs, spars, stiffeners and panels. Although the Classical Force Method dominated stress analysis during the 1950s [233], stiffness methods were kept alive by use in dynamics and vibration. It is not coincidence that Turner was a world-class expert in aeroelasticity.

§O.1.3. G2: The Golden Age

The next period spans the golden age of FEM: 1962–1972. This is the "variational generation." Melosh showed [468] that conforming displacement models are a form of Rayleigh-Ritz based on the minimum potential energy principle. This influential paper marks the confluence of three lines of research: Argyris' dual formulation of energy methods [22], the Direct Stiffness Method (DSM) of Turner [728,730], and early ideas of interelement compatibility as basis for error bounding and convergence [266,467]. G1 workers thought of finite elements as idealizations of structural components. From 1962 onward a two-step interpretation emerges: discrete elements approximate continuum models, which in turn approximate real structures.

By the early 1960s FEM begins to expand into Civil Engineering through Clough's Boeing-Berkeley connection [142,143] and had been baptized [134,136]. Reading Fraeijs de Veubeke's famous article [267] side by side with TCMT [727] one can sense the ongoing change in perspective opened up by the variational framework. The first book devoted to FEM appears in 1967 [785]. Applications to nonstructural problems had started in 1965 [784], and were treated in some depth by Martin and Carey [453].

From 1962 onwards the displacement formulation dominates. This was given a big boost by the invention of the isoparametric formulation and related tools (numerical integration, fitted natural coordinates, shape functions, patch test) by Irons and coworkers [381,384]. Low order displacement models often exhibit disappointing performance. Thus there was a frenzy to develop higher order elements. Other variational formulations, notably hybrids [554,559], mixed [342,695] and equilibrium models [267] emerged. G2 can be viewed as closed by the monograph of Strang and Fix [672], the first book to focus on the mathematical foundations.

§O.1.4. G3: Consolidation

The post-Vietnam economic doldrums are mirrored during this post-1972 period. Gone is the youthful exuberance of the golden age. This is consolidation time. Substantial effort is put into improving the stock of G2 displacement elements by tools initially labeled "variational crimes" [671], but later justified. Textbooks by Hughes [374] and Bathe [53] reflect the technology of this period. Hybrid and mixed formulations record steady progress [39]. Assumed strain formulations

appear [438]. A booming activity in error estimation and mesh adaptivity is fostered by better understanding of the mathematical foundations [688].

Commercial FEM codes gradually gain importance. They provide a reality check on what works in the real world and what doesn't. By the mid-1980s there was gathering evidence that complex and high order elements were commercial flops. Exotic gadgetry interweaved amidst millions of lines of code easily breaks down in new releases. Complexity is particularly dangerous in nonlinear and dynamic analyses conducted by novice users. A trend back toward simplicity starts [440,444].

§O.1.5. G4: Back to Basics

The fourth generation begins by the early 1980s. More approaches come on the scene, notably the Free Formulation [81,85], orthogonal hourglass control [256], Assumed Natural Strain methods [56,665], stress hybrid models in natural coordinates [557,578], as well as variants and derivatives of those approaches: ANDES [219,475], EAS [648,649] and others. Although technically diverse the G4 approaches share two common objectives:

- (i) Elements must fit into DSM-based programs since that includes the vast majority of production codes, commercial or otherwise.
- (ii) Elements are kept simple but should provide answers of engineering accuracy with relatively coarse meshes. These were collectively labeled “high performance elements” in 1989 [213].

Two more recent trends can be noted: increased abstraction on the mathematical side,¹ and canned recipes for running commercial software on the physical side.

“Things are always at their best in the beginning,” said Pascal. Indeed. By now FEM looks like an aggregate of largely disconnected methods and recipes. The blame should not be placed on the method itself, but on the community split noted in the book Preface.

§O.1.6. Precursors

As used today, FEM represents the confluence of three ingredients: Matrix Structural Analysis (MSA), variational approximation theory, and the digital computer. These came together in the early 1950. The reader should not think, however, that they simultaneously appeared on the table through some alchemy. MSA came on the scene in the mid 1930s when desk calculators became popular, as narrated in Appendix H. And variational approximation schemes akin to those of modern FEM were proposed before digital computers. Three examples:

- The historical sketch of [453] says that “Archimedes used finite elements in determining the volume of solids.” The alleged linkage is tenuous. Indeed he calculated areas, lengths and volumes of geometrical objects by dividing them into simpler ones and adding their contributions, passing to the limit as necessary. Where does “variational approximation” come in? Well, one may argue that the volume (area, length) measure of an object is a scalar functional of its geometry. Transmute “measure” into “energy” and “simpler objects” into “elements” and you capture one of the FEM tenets: the energy of the system is the sum of element energies. But for Archimedes to reach modern FEM “long is the way, and hard,” since physical energy calculations require derivatives and Calculus would not be invented for 20 centuries.

¹ “If you go too far up, abstraction-wise, you run out of oxygen.” (Joel Spolsky).

- In his studies leading to the creation of variational calculus, Euler divided the interval of definition of a one-dimensional functional into finite intervals and assumed a linear variation over each, defined by end values [419, p. 53]. Passing to the limit he obtained what is now called the Euler-Lagrange differential equation of variational calculus. Thus Euler deserves credit for being the first to use a piecewise linear function with discontinuous derivatives at nodes to produce, out of the hat, an ODE with second derivatives. He did not use those functions, however, to obtain an approximate value of the functional.²
- In the early 1940s Courant wrote an expository article [151] advocating the variational treatment of partial differential equations. The Appendix of this article contains the first FEM-style calculations on a triangular net for determining the torsional stiffness of a hollow shaft. He used piecewise linear interpolation over each triangle as Rayleigh-Ritz trial functions, and called his idea “generalized finite differences.”
- A direct variational approach similar to Courant’s was continued by Synge and Prager in the context of functional analysis [571] and exposed in Synge’s book [686] as the “hypercircle” method.³
- The seminal paper by Turner et al [727] cites two immediate DSM precursors, both dated 1953, by Levy [427] and Schuerch [634]. (Only the former is available as a journal article; both have “delta wings” in the title.) From [727], p. 806: “In a recent paper Levy has presented a method of analysis for highly redundant structures that is particularly suited to the use of high-speed digital computing machines. . . . The stiffness matrix for the entire structure is computed by simple summation of of the stiffness matrices of the elements of the structure.”

Precursors prior to 1950 had no influence on the rapid developments of Generation 1 outlined in §O.7.2. Two crucial pieces were missing. First, and most important, was the programmable digital computer. Without computers FEM would be a curiosity, worth perhaps a footnote in an arcane book. Also missing was a *driving application* that could get the long-term attention of scientists and engineers as well as industrial resources to fund R&D work. Aerospace structural mechanics provided the driver because the necessary implementation apparatus of MSA was available since the late 1930s [274].

Matrix procedures had to be moved from desk calculators and punched-tape accounting machines to digital computers, which affluent aerospace companies were able to afford amidst Cold War paranoia. Can you imagine defense funds pouring into hypercircles or Courant’s triangles? Once all pieces were in place, synergy transformed the method into a *product*, and FEM took off.

² That would have preceded the invention of direct variational methods (Rayleigh-Ritz) for over one century, while representing also the first FEM-style calculation. A near miss indeed.

³ Curiously this book does not mention, even in passing, the use of digital computers that had already been commercially available for several years. The few numerical examples, all in 2D, are done by hand via relaxation methods.

P

Partitioned Matrices and the Schur Complement

TABLE OF CONTENTS

	Page
§P.1. Partitioned Matrix	P–3
§P.2. Schur Complements	P–3
§P.3. Block Diagonalization	P–3
§P.4. Determinant Formulas	P–3
§P.5. Partitioned Inversion	P–4
§P.6. Solution of Partitioned Linear System	P–4
§P.7. Rank Additivity	P–5
§P.8. Inertia Additivity	P–5
§P.9. The Quotient Property	P–6
§P.10. Generalized Schur Complements	P–6
§P. Notes and Bibliography	P–6

Partitioned matrices often appear in the exposition of Finite Element Methods. This Appendix collects some basic material on the subject. Emphasis is placed on results involving the *Schur complement*. This is the name given in the linear algebra literature to matrix objects obtained through the condensation (partial elimination) process discussed in Chapter 10.

§P.1. Partitioned Matrix

Suppose that the square matrix \mathbf{M} dimensioned $(n+m) \times (n+m)$, is partitioned into four submatrix blocks as

$$\mathbf{M}_{(n+m) \times (n+m)} = \begin{bmatrix} \mathbf{A}_{n \times n} & \mathbf{B}_{n \times m} \\ \mathbf{C}_{m \times n} & \mathbf{D}_{m \times m} \end{bmatrix} \quad (\text{P.1})$$

The dimensions of \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{D} are as shown in the block display. \mathbf{A} and \mathbf{D} are square matrices, but \mathbf{B} and \mathbf{C} are not square unless $n = m$. Entries are generally assumed to be complex.

§P.2. Schur Complements

If \mathbf{A} is nonsingular, the Schur complement of \mathbf{M} with respect to \mathbf{A} is defined as

$$\mathbf{M}/\mathbf{A} \stackrel{\text{def}}{=} \mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}. \quad (\text{P.2})$$

If \mathbf{D} is nonsingular, the Schur complement of \mathbf{M} with respect to \mathbf{D} is defined as

$$\mathbf{M}/\mathbf{D} \stackrel{\text{def}}{=} \mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C}. \quad (\text{P.3})$$

Matrices (P.2) and (P.3) are also called the *Schur complement of \mathbf{A} in \mathbf{M}* and the *Schur complement of \mathbf{D} in \mathbf{M}* , respectively. These equivalent statements are sometimes preferable. If both diagonal blocks are singular, see §P.10. See **Notes and Bibliography** for other notations in common use.

§P.3. Block Diagonalization

The following *block diagonalization forms*, due to Aitken, clearly display the Schur complement role. If \mathbf{A} is nonsingular,

$$\begin{aligned} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} &= \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}/\mathbf{A} \end{bmatrix}, \\ \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}/\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}. \end{aligned} \quad (\text{P.4})$$

whereas if \mathbf{D} is nonsingular

$$\begin{aligned} \begin{bmatrix} \mathbf{I} & -\mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{D}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix} &= \begin{bmatrix} \mathbf{M}/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}, \\ \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} &= \begin{bmatrix} \mathbf{I} & \mathbf{B}\mathbf{D}^{-1} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{M}/\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{D}^{-1}\mathbf{C} & \mathbf{I} \end{bmatrix}. \end{aligned} \quad (\text{P.5})$$

All of these can be directly verified by matrix multiplication.

§P.4. Determinant Formulas

Taking determinants on both sides of the second of (P.4) yields

$$\det(\mathbf{M}) = \begin{vmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{vmatrix} \begin{vmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}/\mathbf{A} \end{vmatrix} \begin{vmatrix} \mathbf{I} & \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{vmatrix} = \begin{vmatrix} \mathbf{A} & \mathbf{0} \\ \mathbf{0} & \mathbf{M}/\mathbf{A} \end{vmatrix} = \det(\mathbf{A}) \cdot \det(\mathbf{M}/\mathbf{A}). \quad (\text{P.6})$$

since the determinant of unit triangular matrices is one. Similarly, from the second of (P.4) one gets

$$\det(\mathbf{M}) = \det(\mathbf{D}) \cdot \det(\mathbf{M}/\mathbf{D}). \quad (\text{P.7})$$

It follows that the determinant is multiplicative in the Schur complement. Extensions to the case in which \mathbf{A} or \mathbf{D} are singular, and its historical connection to Schur's paper [635] are discussed in [781].

§P.5. Partitioned Inversion

Suppose that the upper diagonal block \mathbf{A} in (P.1) is nonsingular. Then the inverse of \mathbf{M} in partitioned form may be expressed in several ways that involve (P.2):

$$\begin{aligned} \mathbf{M}^{-1} &= \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & (\mathbf{M}/\mathbf{A})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{I} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1} \\ \mathbf{0} & (\mathbf{M}/\mathbf{A})^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ -\mathbf{C}\mathbf{A}^{-1} & \mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{A}^{-1}\mathbf{B} \\ \mathbf{I} \end{bmatrix} (\mathbf{M}/\mathbf{A})^{-1} [-\mathbf{C}\mathbf{A}^{-1} \quad \mathbf{I}] \\ &= \begin{bmatrix} \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & -\mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1} \\ -(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1} & (\mathbf{M}/\mathbf{A})^{-1} \end{bmatrix} \end{aligned} \quad (\text{P.8})$$

Suppose next that the lower diagonal block \mathbf{D} in (P.1) is nonsingular. Then the form corresponding to the last one above reads

$$\mathbf{M}^{-1} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix}^{-1} = \begin{bmatrix} (\mathbf{M}/\mathbf{D})^{-1} & -(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{A}^{-1} \\ -\mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1} & \mathbf{D}^{-1} + \mathbf{D}^{-1}\mathbf{C}(\mathbf{M}/\mathbf{D})^{-1}\mathbf{B}\mathbf{D}^{-1} \end{bmatrix} \quad (\text{P.9})$$

If both \mathbf{A} and \mathbf{D} are nonsingular, equating the top left hand block of (P.9) to that of the last form of (P.8) gives Duncan's inversion formula

$$(\mathbf{M}/\mathbf{D})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{M}/\mathbf{A})^{-1}\mathbf{C}\mathbf{A}^{-1}. \quad (\text{P.10})$$

or explicitly in terms of the original blocks,

$$(\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1} = \mathbf{A}^{-1} + \mathbf{A}^{-1}\mathbf{B}(\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}\mathbf{C}\mathbf{A}^{-1}. \quad (\text{P.11})$$

When $n = m$ so that \mathbf{A} , \mathbf{B} , \mathbf{C} and \mathbf{D} are all square, and all of them are nonsingular, one obtains the elegant formula due to Aitken:

$$\mathbf{M}^{-1} = \begin{bmatrix} (\mathbf{M}/\mathbf{D})^{-1} & (\mathbf{M}/\mathbf{B})^{-1} \\ (\mathbf{M}/\mathbf{C})^{-1} & (\mathbf{M}/\mathbf{A})^{-1} \end{bmatrix} \quad (\text{P.12})$$

Replacing $-\mathbf{D}$ by the inverse of a matrix, say, \mathbf{T}^{-1} , leads to the Woodbury formula presented in Appendix D. If \mathbf{B} and \mathbf{C} are column and row vectors, respectively, vectors and \mathbf{D} a scalar, one obtains the Sherman-Morrison inverse-update formula also presented in that Appendix.

§P.6. Solution of Partitioned Linear System

An important use of Schur complements is the partitioned solution of linear systems. In fact this is the most important application in FEM, in connection with superelement analysis. Suppose that we have the linear system $\mathbf{M}\mathbf{z} = \mathbf{r}$, in which the given coefficient matrix \mathbf{M} is partitioned as in (P.1). The RHS $\mathbf{r} = [\mathbf{p} \ \mathbf{q}]^T$ is given, while $\mathbf{z} = [\mathbf{x} \ \mathbf{y}]^T$ is unknown. The system is equivalent to the matrix equation pair

$$\begin{aligned} \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{y} &= \mathbf{p}, \\ \mathbf{C}\mathbf{x} + \mathbf{D}\mathbf{y} &= \mathbf{q}. \end{aligned} \quad (\text{P.13})$$

If \mathbf{D} is nonsingular, eliminating \mathbf{y} and solving for \mathbf{x} yields

$$\mathbf{x} = (\mathbf{A} - \mathbf{B}\mathbf{D}^{-1}\mathbf{C})^{-1}(\mathbf{p} - \mathbf{B}\mathbf{D}^{-1}\mathbf{q}) = (\mathbf{M}/\mathbf{D})^{-1}(\mathbf{p} - \mathbf{B}\mathbf{D}^{-1}\mathbf{q}). \quad (\text{P.14})$$

Similarly if \mathbf{A} is nonsingular, eliminating \mathbf{x} and solving for \mathbf{y} yields

$$\mathbf{y} = (\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B})^{-1}(\mathbf{q} - \mathbf{C}\mathbf{A}^{-1}\mathbf{p}) = (\mathbf{M}/\mathbf{A})^{-1}(\mathbf{q} - \mathbf{C}\mathbf{A}^{-1}\mathbf{p}). \quad (\text{P.15})$$

Readers familiar with the process of static condensation in FEM will notice that (P.14) is precisely the process of elimination of internal degrees of freedom of a superelement, as discussed in Chapter 10. In this case \mathbf{M} becomes the superelement stiffness matrix \mathbf{K} , which is symmetric. In the notation of that Chapter, the appropriate block mapping is $\mathbf{A} \rightarrow \mathbf{K}_{bb}$, $\mathbf{B} \rightarrow \mathbf{K}_{bi}$, $\mathbf{C} \rightarrow \mathbf{K}_{ib} = \mathbf{K}_{bi}^T$, $\mathbf{D} \rightarrow \mathbf{K}_{ii}$, $\mathbf{p} \rightarrow \mathbf{f}_b$, $\mathbf{q} \rightarrow \mathbf{f}_i$, $\mathbf{x} \rightarrow \mathbf{u}_b$, and $\mathbf{y} \rightarrow \mathbf{u}_i$. The Schur complement $\mathbf{M}/\mathbf{D} \rightarrow \mathbf{K}/\mathbf{K}_{ii}$ is the condensed stiffness matrix $\tilde{\mathbf{K}}_{bb} = \mathbf{K}_{bb} - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib}$, while the RHS of (P.14) is the condensed force vector $\tilde{\mathbf{f}}_b = \mathbf{f}_b - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{f}_i$.

§P.7. Rank Additivity

From (P.4) or (P.5) we immediately obtain the *rank additivity* formulas

$$\text{rank}(\mathbf{M}) = \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{M}/\mathbf{A}) = \text{rank}(\mathbf{D}) + \text{rank}(\mathbf{M}/\mathbf{D}), \quad (\text{P.16})$$

or, explicitly in term of the original blocks

$$\text{rank} \left(\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} \right) = \text{rank}(\mathbf{A}) + \text{rank}(\mathbf{D} - \mathbf{B}\mathbf{A}^{-1}\mathbf{C}) = \text{rank}(\mathbf{D}) + \text{rank}(\mathbf{A} - \mathbf{C}\mathbf{D}^{-1}\mathbf{B}), \quad (\text{P.17})$$

assuming that the indicated inverses exist. Consequently the rank is additive in the Schur complement, and so it the inertia of a Hermitian matrix, as considered next.

§P.8. Inertia Additivity

Consider the (generally complex) partitioned Hermitian matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{12}^* & \mathbf{A}_{22} \end{bmatrix}. \quad (\text{P.18})$$

Here \mathbf{A}_{11} is assumed nonsingular, and \mathbf{A}_{12}^* denotes the conjugate transpose of \mathbf{A}_{12} .

The *inertia* of \mathbf{A} is a list of three nonnegative integers

$$\text{In}(\mathbf{A}) = \{n_+, n_-, n_0\}, \quad (\text{P.19})$$

in which $n_+ = n_+(\mathbf{A})$, $n_- = n_-(\mathbf{A})$, and $n_0 = n_0(\mathbf{A})$ give the number of positive, negative, and zero eigenvalues of \mathbf{A} , respectively. (Recall that all eigenvalues of a Hermitian matrix are real.) Since \mathbf{A} is Hermitian, its rank is $n_+(\mathbf{A}) + n_-(\mathbf{A})$, and its nullity $n_0(\mathbf{A})$. (Note that n_0 is the dimension of the kernel of \mathbf{A} and also its corank.) The *inertia additivity formula* in terms of the Schur complement of \mathbf{A}_{11} in \mathbf{A} , is

$$\text{In}(\mathbf{A}) = \text{In}(\mathbf{A}_{11}) + \text{In}(\mathbf{A}/\mathbf{A}_{11}). \quad (\text{P.20})$$

It follows that if \mathbf{A}_{11} is positive definite, and $\mathbf{A}/\mathbf{A}_{11}$ is positive (nonnegative) definite, then \mathbf{A} is positive (nonnegative) definite. Also \mathbf{A} and $\mathbf{A}/\mathbf{A}_{11}$ have the same nullity.

§P.9. The Quotient Property

Consider the partitioned matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{D} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & \vdots & \mathbf{B}_1 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \vdots & \mathbf{B}_2 \\ \dots & \dots & \dots & \dots \\ \mathbf{C}_1 & \mathbf{C}_2 & \vdots & \mathbf{D} \end{bmatrix}, \quad (\text{P.21})$$

in which both \mathbf{A} and \mathbf{A}_{11} are nonsingular. Matrix \mathbf{M} need not be square. The *quotient property* of nested Schur complements is

$$\mathbf{M}/\mathbf{A} = (\mathbf{M}/\mathbf{A}_{11})/(\mathbf{A}/\mathbf{A}_{11}). \quad (\text{P.22})$$

On the RHS of (P.22) we see that the “denominator” matrix \mathbf{A}_{11} “cancels out” as if the expressions were scalar fractions. If \mathbf{M} is square and nonsingular, then taking determinants and applying the determinant formula we obtain

$$\det(\mathbf{M}/\mathbf{A}) = \det(\mathbf{M}/\mathbf{A}_{11})/\det(\mathbf{A}/\mathbf{A}_{11}). \quad (\text{P.23})$$

§P.10. Generalized Schur Complements

If both diagonal blocks \mathbf{A} and \mathbf{D} of \mathbf{M} in (P.1) are singular, the notion of Schur complement may be generalized by replacing the conventional inverse with the Moore-Penrose generalized inverse. Further developments of this fairly advanced topic may be found in Chapters 1 and 6 of [781].

Notes and Bibliography

Most of the following historical remarks are extracted from the **Historical Introduction** chapter of [781]. This edited book gives a comprehensive and up-to-date coverage of the Schur's complement and its use as a basic tool in linear algebra.

The name “Schur complement” for the matrices defined in (P.2) and (P.3) was introduced by Haynsworth in 1968 [334,335]. The name was motivated by the seminal “determinant lemma” by Schur published in 1917 in his studies of stability polynomials [635]. Earlier implicit manifestations of this idea starting with Laplace in 1812 are described in the **Historical Introduction** chapter of [781].

Although (P.6) and (P.7) are now commonly referred to as *Schur determinant formulas* in the literature, he actually proved only a special case, in which $n = m$ so all block matrices in (P.1) are square; in addition **B** and **C** were required to commute. The effect of these assumptions is that the formula can be extended to singular diagonal blocks. See §0.3 of [781] for details.

The Schur complement appears naturally in the inversion, linear-equation solving, rank and inertia determinations that involve block partitioned matrices. Among the most seminal formulas are:

- The last of the partitioned inversion formulas (P.8) was first published in 1937 by Banachiewicz [44]. It appeared one year later (independently derived) in the classical monograph by Frazer, Duncan and Collar [274], cited in Appendix H as one of the early sources for Matrix Structural Analysis.
- The Duncan inversion formula (P.10) appeared in 1944 [187]. It is a follow up to the inversion methods of [274].
- The Aitken block diagonalization formulas (P.4) and (P.5) appear in another classic book from the same period [7].
- The inertia additivity formula (P.20) as well as the quotient property (P.22) are due to Haynsworth and coworkers [334,335]; see also [154,336,514,515,515].
- From the Duncan inversion formula follows directly the Guttman rank additivity formula (P.16) first published in 1946 [311], as well as the already cited inertia additivity formula for Hermitian matrices (P.20), which appeared much later [334,335].

The notation \mathbf{M}/\mathbf{A} for $\mathbf{D} - \mathbf{C}\mathbf{A}^{-1}\mathbf{B}$ is now commonly used in linear algebra. Others are (\mathbf{M}/\mathbf{A}) , $\mathbf{M}|\mathbf{A}$, $(\mathbf{M}|\mathbf{A})$, and \mathbf{M}_A . The latter has visualization problems if matrix blocks are subscripted; for example $\mathbf{M}_{A_{22}}$ is clumsier to read than $\mathbf{M}/\mathbf{A}_{22}$ or $\mathbf{M}|\mathbf{A}_{22}$. A drawback of the slash (and vertical bar) notation is that enclosing parentheses are necessary if one takes a transpose or inverse; e.g., compare $(\mathbf{M}/\mathbf{A}_{22})^{-1}$ versus the more compact $\mathbf{M}_{A_{22}}^{-1}$.

Q

Miscellaneous FEM Formulation Topics

TABLE OF CONTENTS

	Page
§Q.1. *Natural Strains and Stresses	Q-3
§Q.1.1. *Straight Line Elements	Q-3
§Q.1.2. *Plane Stress Quadrilaterals	Q-4
§Q.1.3. *Three Dimensional Bricks	Q-4
§Q.2. *Hierarchical Shape Functions	Q-5
§Q.2.1. *The Six Node Triangle, Revisited	Q-5
§Q.2.2. *Hierarchical Interpolation	Q-6
§Q.2.3. *Hierarchical Stiffness Matrix and Load Vector	Q-8
§Q.2.4. *Equation Nesting and Node Dropping	Q-9
§Q.2.5. *Internal Node Injection	Q-10
§Q.3. *A 3-Node Curved Bar Element	Q-12
§Q.3.1. *Element Description	Q-13
§Q.3.2. The Stiffness Matrix	Q-16
§Q.3.3. Geometric Properties of Plane Curves	Q-19
§Q.3.4. *Why Is the Stiffness Matrix Rank Deficient?	Q-20
§Q. Exercises	Q-23

This Chapter concludes Part II with miscellaneous topics in finite element formulation. This material is not intended to be covered in an introductory course. It provide sources for term projects and take-home exams as well as serving as a bridge to more advanced courses. The three topics are: natural strains and stresses, hierarchical shape functions, and curved bar elements.

§Q.1. *Natural Strains and Stresses

In the element formulations of Chapters 12–19, shape functions have been developed in natural coordinates. For example $\{\zeta_1, \zeta_2, \zeta_3\}$ in triangles and $\{\xi, \eta\}$ in quadrilaterals.

Strains and stresses are, however, expressed in terms of Cartesian coordinate components. This is no accident. It connects seamlessly with the elasticity formulations engineers are familiar with. Furthermore, the constitutive properties are often expressed with reference to Cartesian coordinates.

In advanced FEM formulations it is often convenient to proceed further, and express also strains and stresses in terms of natural coordinate components. For brevity these are called *natural strains* and *natural stresses*, natural strains natural stresses respectively. The advantages of doing so become apparent when one goes beyond the pure displacement formulations and assumes also strains and/or stress patterns. Since this book does not go that far, natural strains and stresses are introduced as objects deserving study on their own.

There is an exposition problem, however, related to “lack of uniqueness” in the definition of natural strains. First, the definitions vary according to element geometry: line, triangle, quadrilateral, tetrahedron, brick, and so on. This is to be expected since the natural coordinates vary with the geometry. Second, definitions are often author dependent because the topic is unsettled. The following subsections present natural strains and stresses only for two configurations in which there is reasonable agreement in the literature.

§Q.1.1. *Straight Line Elements

Suppose that a straight one-dimensional bar element is defined in terms of the local axis x and axial displacement u in the isoparametric form

$$x = x(\xi), \quad u = u(\xi) \quad (\text{Q.1})$$

where ξ is a dimensionless natural coordinate that varies from -1 to $+1$. Derivatives with respect to ξ will be denoted by a subscript; for example $dx/d\xi = x_\xi$ and $u_\xi = du/d\xi$. The Cartesian strain is $e = du/dx = (du/d\xi)(d\xi/dx) = J^{-1}u_\xi$, where $J = dx/d\xi = x_\xi$ is the 1D Jacobian. The *natural strain* $e_{\xi\xi}$ is defined by

$$e_{\xi\xi} = \frac{dx}{d\xi} \frac{du}{d\xi} = Ju_\xi \quad (\text{Q.2})$$

Because ξ is dimensionless, $e_{\xi\xi}$ has dimension of length squared, that is, area. Obviously this does not lead to a simple physical interpretation, as is the case with Cartesian strains.

What is the relation between e and $e_{\xi\xi}$? This is easily obtained by performing some Jacobian manipulations:

$$e = \frac{du}{dx} = J^{-1}J \frac{du}{dx} = J^{-1} \frac{dx}{d\xi} \frac{du}{d\xi} \frac{d\xi}{dx} = J^{-1}e_{\xi\xi}J^{-1} = J^{-2}e_{\xi\xi} \quad (\text{Q.3})$$

Since J has dimension of length, J^{-2} restores the expected non-dimensionality of e .

The *natural stress* $\sigma_{\xi\xi}$ may be defined in several ways. The most straightforward definition uses the invariance of strain energy density: $\sigma e = \sigma_{\xi\xi} e_{\xi\xi}$. Hence $\sigma_{\xi\xi} = (e/e_{\xi\xi})\sigma = J^{-2}\sigma$ and $\sigma_{\xi\xi} = J^2\sigma$. Note that $\sigma_{\xi\xi}$ has dimensions of force and not of force per unit area.

If $\sigma = E e$, the last step is to define a natural constitutive relation $\sigma_{\xi\xi} = E_{\xi\xi} e_{\xi\xi}$. Evidently $E_{\xi\xi} = \sigma_{\xi\xi}/e_{\xi\xi} = (J^2\sigma)/(J^{-2}e) = J^4 E$. This “natural modulus” has dimensions of force times area.

In the Assumed Natural Strain (ANS) formulation of finite element, one assumes directly a form for $e_{\xi\xi}$ as a *simplification* of the expression (Q.2). One common simplification is to take an average Jacobian J_0 instead of the variable Jacobian $J = x_\xi$; for example the value at the element midpoint $\xi = 0$. The implications are explored in Exercise Q.1.

§Q.1.2. *Plane Stress Quadrilaterals

The foregoing derivation for line elements looks like empty formalism. And indeed it does not help much. The power of the ANS method comes in two and three dimensions. In this section we restrict the exposition to plane quadrilaterals in plane stress referred to a Cartesian system $\{x, y\}$. Collect coordinates $\{x, y\}$ in a 2-vector \vec{x} and the inplane displacement field into a 2-vector \vec{u} . Then the natural strains are defined as

$$e_{\xi\xi} = \frac{\partial \vec{x}}{\partial \xi} \cdot \frac{\partial \vec{u}}{\partial \xi}, \quad e_{\eta\eta} = \frac{\partial \vec{x}}{\partial \eta} \cdot \frac{\partial \vec{u}}{\partial \eta}, \quad e_{\xi\eta} = \frac{\partial \vec{x}}{\partial \xi} \cdot \frac{\partial \vec{u}}{\partial \eta}, \quad e_{\eta\xi} = \frac{\partial \vec{x}}{\partial \eta} \cdot \frac{\partial \vec{u}}{\partial \xi}, \quad \gamma_{\xi\eta} = e_{\xi\eta} + e_{\eta\xi}. \quad (\text{Q.4})$$

where \cdot denotes the vector dot product. Note that, unlike Cartesian strains, the shear strains $e_{\xi\eta}$ and $e_{\eta\xi}$ are generally different; consequently $\gamma_{\xi\eta} \neq 2e_{\xi\eta}$ and $\gamma_{\xi\eta} \neq 2e_{\eta\xi}$. To effect a transformation to Cartesian strains, it is convenient to go back to a tensor-like arrangement

$$\mathbf{e}_{\xi\eta} = \begin{bmatrix} e_{\xi\xi} & e_{\xi\eta} \\ e_{\eta\xi} & e_{\eta\eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial u_x}{\partial \xi} & \frac{\partial u_x}{\partial \eta} \\ \frac{\partial u_y}{\partial \xi} & \frac{\partial u_y}{\partial \eta} \end{bmatrix} = \mathbf{J} \mathbf{G}, \quad (\text{Q.5})$$

where $\mathbf{e}_{\xi\eta}$, \mathbf{J} and \mathbf{G} denote the indicated matrices (\mathbf{J} is the Jacobian matrix introduced in Chapter 17). It is easily verified that

$$\mathbf{J}^{-1} \mathbf{e}_{\xi\eta} \mathbf{J}^{-T} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} \end{bmatrix} = \begin{bmatrix} e_{xx} & e_{yx} \\ e_{xy} & e_{yy} \end{bmatrix}. \quad (\text{Q.6})$$

from which the Cartesian components may be extracted and rearranged as a 3-vector. Exercise Q.4 works out how to express the transformations as a matrix-vector product. The natural stresses are defined as the energy conjugates of the natural strains. The natural constitutive equation follows.

§Q.1.3. *Three Dimensional Bricks

The extension to three dimensions for brick elements is straightforward. We list here only the pertinent definitions and results. The natural coordinates are $\{\xi, \eta, \zeta\}$ and the Cartesian coordinates

$\{x, y, z\}$. Going directly to matrix-tensor form, the natural strains are defined as

$$\mathbf{e}_{\xi\eta\zeta} = \begin{bmatrix} e_{\xi\xi} & e_{\xi\eta} & e_{\xi\zeta} \\ e_{\eta\xi} & e_{\eta\eta} & e_{\eta\zeta} \\ e_{\zeta\xi} & e_{\zeta\eta} & e_{\zeta\zeta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial u_x}{\partial \xi} & \frac{\partial u_x}{\partial \eta} & \frac{\partial u_x}{\partial \zeta} \\ \frac{\partial u_y}{\partial \xi} & \frac{\partial u_y}{\partial \eta} & \frac{\partial u_y}{\partial \zeta} \\ \frac{\partial u_z}{\partial \xi} & \frac{\partial u_z}{\partial \eta} & \frac{\partial u_z}{\partial \zeta} \end{bmatrix} = \mathbf{J} \mathbf{G}, \quad (\text{Q.7})$$

where \mathbf{J} is the Jacobian matrix for bricks. To pass to Cartesian strains use

$$\mathbf{J}^{-1} \mathbf{e}_{\xi\eta\zeta} \mathbf{J}^{-T} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix} = \begin{bmatrix} e_{xx} & e_{yx} & e_{zx} \\ e_{xy} & e_{yy} & e_{zy} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix}. \quad (\text{Q.8})$$

from which the 6-vector of Cartesian strains is easily extracted.

§Q.2. *Hierarchical Shape Functions

Conventional shape functions, as covered in Chapters 15–19, express total displacements from a reference state. Hierarchical shape functions express *differences* from a set of simpler shape functions. Elements derived with hierarchical shape functions are called *hierarchical elements*. This concept finds applications in the following areas: hierarchical shape functions hierarchical elements

- (1) *Element Specialization* Hierarchical formulations simplify the derivation of special elements generated from a “parent” element by removing or transforming freedoms. In particular transition elements are easily produced.
- (2) *Implementation of p -convergence.* This adaptive-discretization technique, which relies on systematic use of higher polynomial orders, relies on hierarchical elements implemented in a multilevel manner.
- (3) *Advanced Element Formulations* Some advanced formulations rely on hierarchical “splitting” of the element response. Although the split may not necessarily be done with shape functions, learning the topic helps.

In what follows we shall emphasize only the first application.

§Q.2.1. *The Six Node Triangle, Revisited

Consider (again) the six-node quadratic triangle, which is pictured in Figure Q.1. The isoparametric definition of this element is repeated here for convenience:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \end{bmatrix}. \quad (\text{Q.9})$$

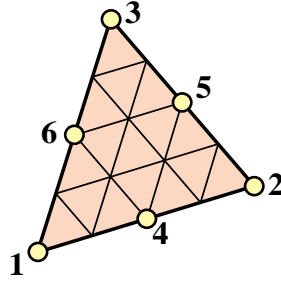


Figure Q.1. The six-node quadratic triangle (once more)

with the shape functions

$$\begin{aligned} N_1^{(e)} &= \zeta_1(2\zeta_1 - 1), & N_4^{(e)} &= 4\zeta_1\zeta_2 \\ N_2^{(e)} &= \zeta_2(2\zeta_2 - 1), & N_5^{(e)} &= 4\zeta_2\zeta_3 \\ N_3^{(e)} &= \zeta_3(2\zeta_3 - 1), & N_6^{(e)} &= 4\zeta_3\zeta_1 \end{aligned} \quad (\text{Q.10})$$

For use below consider a generic scalar function, w , interpolated with the shape functions (Q.10):

$$w = [w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6] \begin{bmatrix} N_1 \\ N_2 \\ \vdots \\ N_6 \end{bmatrix}. \quad (\text{Q.11})$$

Symbol w may represent x , y , u_x or u_y in the isoparametric representation (Q.10), or other element-varying quantities such as thickness, temperature, etc. See Figure Q.2(a).

§Q.2.2. *Hierarchical Interpolation

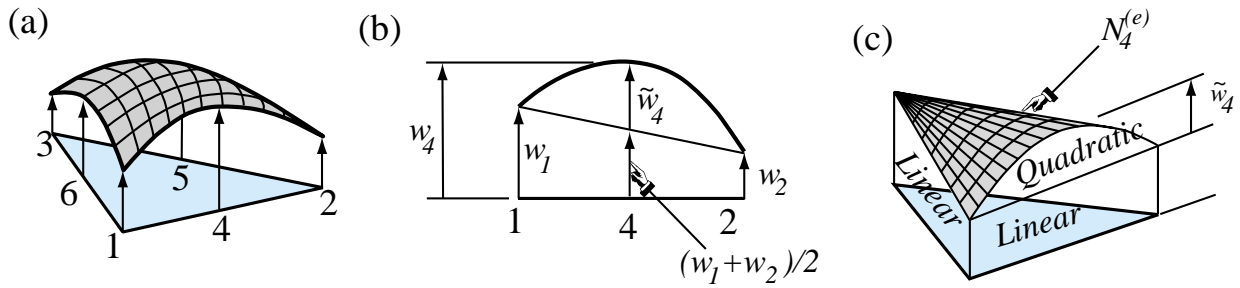
To “go hierarchical” the key step is to express the values of w at the midnodes 4, 5 and 6 as *deviations from the linear interpolation*:

$$w_4 = \frac{1}{2}(w_1 + w_2) + \tilde{w}_4, \quad w_5 = \frac{1}{2}(w_2 + w_3) + \tilde{w}_5, \quad w_6 = \frac{1}{2}(w_3 + w_1) + \tilde{w}_6. \quad (\text{Q.12})$$

These “deviations from linearity” \tilde{w}_4 , \tilde{w}_5 and \tilde{w}_6 are called *hierarchical values*. They have a straightforward geometric interpretation if w is plotted normal to the plane of the triangle. For example, Figure Q.2(b) illustrates the meaning of the hierarchical value \tilde{w}_4 at midnode 4.

If we insert (Q.12) into (Q.11) we get the *hierarchical interpolation* formula

$$w = [w_1 \quad w_2 \quad w_3 \quad \tilde{w}_4 \quad \tilde{w}_5 \quad \tilde{w}_6] \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}. \quad (\text{Q.13})$$


 Figure Q.2. Interpretation of hierarchical midnode value \tilde{w}_4 .

On comparing (Q.13) with the conventional interpolation two points become evident:

1. The shape functions for the three corner nodes (1, 2 and 3) have become the shape functions of the *linear triangle*.
2. The shape functions for the three midnodes (4, 5 and 6) stay the same.

These results are not surprising, for we have expressed the new (hierarchical) midnodes values as *corrections* from the expansion of the linear triangle. The midnode hierarchical shape functions have the same form, but are measured from the linear shape functions, as illustrated in Figure Q.2(b,c) for $\tilde{N}_4^{(e)} = 4\zeta_1\zeta_2$.

Insert now the hierarchical interpolation formula (Q.5) into rows 2 through 4 of (Q.1) by making $w \equiv x, y, u_x$ and u_y in turn. As for the first row, its last three entries vanish because the hierarchical deviation from a constant (the number 1) is zero. We thus arrive at the *hierarchical isoparametric representation* of the six noded triangle:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ x_1 & x_2 & x_3 & \tilde{x}_4 & \tilde{x}_5 & \tilde{x}_6 \\ y_1 & y_2 & y_3 & \tilde{y}_4 & \tilde{y}_5 & \tilde{y}_6 \\ u_{x1} & u_{x2} & u_{x3} & \tilde{u}_{x4} & \tilde{u}_{x5} & \tilde{u}_{x6} \\ u_{y1} & u_{y2} & u_{y3} & \tilde{u}_{y4} & \tilde{u}_{y5} & \tilde{u}_{y6} \end{bmatrix} \begin{bmatrix} \tilde{N}_1^{(e)} \\ \tilde{N}_2^{(e)} \\ \tilde{N}_3^{(e)} \\ \tilde{N}_4^{(e)} \\ \tilde{N}_5^{(e)} \\ \tilde{N}_6^{(e)} \end{bmatrix}, \quad (\text{Q.14})$$

where only the first three shape functions are different:

$$\begin{aligned} \tilde{N}_1^{(e)} &= \zeta_1, & \tilde{N}_4^{(e)} &= N_4^{(e)} = 4\zeta_1\zeta_2, \\ \tilde{N}_2^{(e)} &= \zeta_2, & \tilde{N}_5^{(e)} &= N_5^{(e)} = 4\zeta_2\zeta_3, \\ \tilde{N}_3^{(e)} &= \zeta_3, & \tilde{N}_6^{(e)} &= N_6^{(e)} = 4\zeta_3\zeta_1. \end{aligned} \quad (\text{Q.15})$$

The linear and quadratic parts of the element are now neatly separated:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} & \vdots & \\ & \vdots & \\ \text{Linear} & \vdots & \text{Quadratic} \\ & \vdots & \\ & \vdots & \end{bmatrix} \begin{bmatrix} \text{Linear} \\ \dots\dots \\ \text{Quadratic} \end{bmatrix}. \quad (\text{Q.16})$$

§Q.2.3. *Hierarchical Stiffness Matrix and Load Vector

The element stiffness equations for the conventional (non-hierarchical) shape function form are

$$\mathbf{K}^{(e)} \mathbf{u}^{(e)} = \mathbf{f}^{(e)}. \quad (\text{Q.17})$$

We would like to transform these equations to the hierarchical form

$$\tilde{\mathbf{K}}^{(e)} \tilde{\mathbf{u}}^{(e)} = \tilde{\mathbf{f}}^{(e)}. \quad (\text{Q.18})$$

The nodal displacement vector may be partitioned as

$$\tilde{\mathbf{u}}^{(e)} = \begin{bmatrix} \tilde{u}_x^{(e)} \\ \tilde{u}_y^{(e)} \end{bmatrix}, \quad (\text{Q.19})$$

where $\tilde{\mathbf{u}}_x$ and $\tilde{\mathbf{u}}_y$ denote the 6×1 vectors

$$\tilde{\mathbf{u}}_x^{(e)} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \tilde{u}_{x4} \\ \tilde{u}_{x5} \\ \tilde{u}_{x6} \end{bmatrix}, \quad \tilde{\mathbf{u}}_y^{(e)} = \begin{bmatrix} \tilde{u}_{y1} \\ \tilde{u}_{y2} \\ \tilde{u}_{y3} \\ \tilde{u}_{y4} \\ \tilde{u}_{y5} \\ \tilde{u}_{y6} \end{bmatrix}. \quad (\text{Q.20})$$

Let $\tilde{\mathbf{T}}_x$ and $\tilde{\mathbf{T}}_y$ denote the 6×6 transformation matrices that relate the non-hierarchical to the hierarchical displacement components along x and y , respectively:

$$\mathbf{u}_x^{(e)} = \tilde{\mathbf{T}}_x \tilde{\mathbf{u}}_x^{(e)}, \quad \mathbf{u}_y^{(e)} = \tilde{\mathbf{T}}_y \tilde{\mathbf{u}}_y^{(e)}. \quad (\text{Q.21})$$

To construct these transformation matrices, observe that

$$\begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x3} \\ u_{x4} \\ u_{x5} \\ u_{x6} \end{bmatrix} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \frac{1}{2}(u_{x1} + u_{x2}) + \tilde{u}_{x4} \\ \frac{1}{2}(u_{x2} + u_{x3}) + \tilde{u}_{x5} \\ \frac{1}{2}(u_{x3} + u_{x1}) + \tilde{u}_{x6} \end{bmatrix} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \frac{1}{2}(\tilde{u}_{x1} + \tilde{u}_{x2}) + \tilde{u}_{x4} \\ \frac{1}{2}(\tilde{u}_{x2} + \tilde{u}_{x3}) + \tilde{u}_{x5} \\ \frac{1}{2}(\tilde{u}_{x3} + \tilde{u}_{x1}) + \tilde{u}_{x6} \end{bmatrix}, \quad (\text{Q.22})$$

with exactly the same relation holding for u_y . We can present (Q.22) in matrix form with

$$\tilde{\mathbf{T}}_x = \tilde{\mathbf{T}}_y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 1 \end{bmatrix}. \quad (\text{Q.23})$$

Combining the two matrix equations in one:

$$\mathbf{u}^{(e)} = \tilde{\mathbf{T}} \tilde{\mathbf{u}}^{(e)} \quad (\text{Q.24})$$

where $\tilde{\mathbf{T}}$ is the 12×12 transformation matrix

$$\tilde{\mathbf{T}} = \begin{bmatrix} \tilde{\mathbf{T}}_x & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{T}}_y \end{bmatrix}. \quad (\text{Q.25})$$

Inserting (Q.17) into (Q.9) and premultiplying by $\tilde{\mathbf{T}}^T$ we obtain (Q.10), in which

$$\begin{aligned} \tilde{\mathbf{K}}^{(e)} &= \tilde{\mathbf{T}}^T \mathbf{K}^{(e)} \tilde{\mathbf{T}} \\ \tilde{\mathbf{f}}^{(e)} &= \tilde{\mathbf{T}}^T \mathbf{f}^{(e)} \end{aligned} \quad (\text{Q.26})$$

In practice the entries of $\tilde{\mathbf{K}}^{(e)}$ and $\tilde{\mathbf{f}}^{(e)}$ are *not* computed by forming the conventional stiffness matrices and force vectors and applying the preceding transformation equations. They are formed directly instead. However, the transformation equations are useful for checking element derivations and computer programs.

§Q.2.4. *Equation Nesting and Node Dropping

To take full advantage of the “node dropping” feature described in §Q.6, it is convenient to order the hierarchical element equations so that all *corner degrees of freedom come first*. That is, we pass from the ordering

$$\begin{aligned} & [\tilde{u}_{x1} \quad \tilde{u}_{x2} \quad \tilde{u}_{x3} \quad \tilde{u}_{x4} \quad \tilde{u}_{x5} \quad \tilde{u}_{x6} \quad \tilde{u}_{y1} \quad \tilde{u}_{y2} \quad \tilde{u}_{y3} \quad \tilde{u}_{y4} \quad \tilde{u}_{y5} \quad \tilde{u}_{y6}] \\ \text{to} & [\tilde{u}_{x1} \quad \tilde{u}_{y1} \quad \tilde{u}_{x2} \quad \tilde{u}_{y2} \quad \tilde{u}_{x3} \quad \tilde{u}_{y3} \quad \tilde{u}_{x4} \quad \tilde{u}_{y4} \quad \tilde{u}_{x5} \quad \tilde{u}_{y5} \quad \tilde{u}_{x6} \quad \tilde{u}_{y6}]. \end{aligned} \quad (\text{Q.27})$$

With this rearrangement of nodal freedoms, the hierarchical stiffness equations of the six-node triangle take the *nested form*

$$\begin{bmatrix} \text{Linear} \\ \text{Quadratic} \end{bmatrix} \begin{bmatrix} \text{Linear} \\ \text{Quadratic} \end{bmatrix} = \begin{bmatrix} \text{Linear} \\ \text{Quadratic} \end{bmatrix}, \quad (\text{Q.28})$$

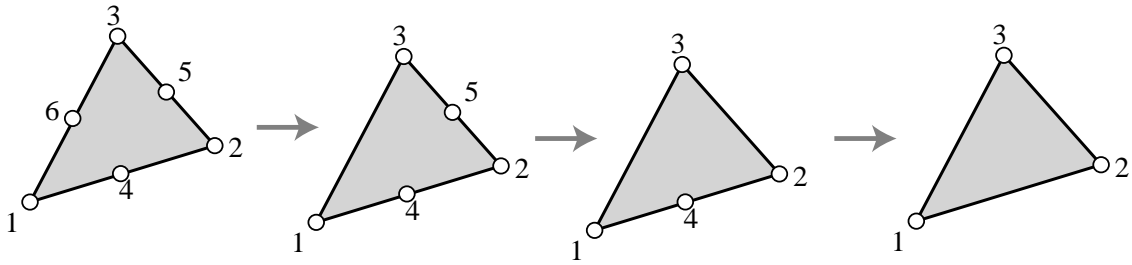


Figure Q.3. Transition triangular elements

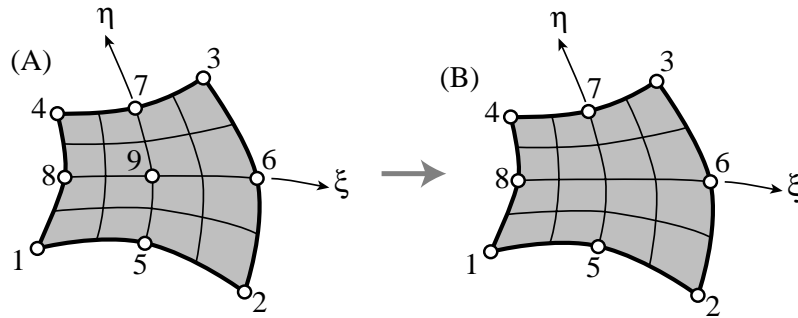


Figure Q.4. Conventional eight-node quadrilateral (A) and nine-node quadrilateral (B) with hierarchical internal node.

in which labels “Linear” and “Quadratic” refer to the association of the equations and degrees of freedom with the linear triangle and quadratic corrections thereto, respectively.

We would like to form the sequence of *transition triangles* illustrated in Figure Q.3 by simply “dropping” or “deleting” nodes in a simple and systematic way.

With the hierarchical stiffness equations in nested form, the task is simple. To get rid of node 6, we simply say that

$$\tilde{u}_{x6} = \tilde{u}_{y6} = 0, \quad (\text{Q.29})$$

and treat this boundary condition as a zero-displacement constraint, dropping equations 11 and 12 from the nested-form of the element stiffness equations. We are left with 10 equations, and have effectively formed the five-node transition element ($n = 5$) shown in Figure Q.4.

Dropping node 5 through a similar procedure we get the four-node transition element ($n = 4$) shown in Figure Q.4. Finally, dropping node 4 we reduce the element to the three-node linear triangle.

§Q.2.5. *Internal Node Injection

The hierarchical representation can also be applied to *internal* nodes. The algebra is more elaborate because *all* external nodes participate in the definition of hierarchical value. The technique used for handling internal nodes is illustrated here on the two elements shown in Figure Q.4: a conventional eight-node quadrilateral (A), and a nine-node quadrilateral (B) with hierarchical internal node 9.

It should be emphasized that the midnodes of the two example elements are *not* hierarchical; in practice they would be, but such “hierarchical nesting” would obscure the presentation that follows.

The procedure followed in this example illustrates the *node injection* technique: from a simpler element we build a more complex one by inserting one or more nodes in a hierarchical manner. This is roughly the inverse of node dropping procedure used in §Q.6 to build transition elements.

The isoparametric representation of (A) is

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \end{bmatrix}, \quad (\text{Q.30})$$

with the shape functions

$$\begin{aligned} N_1^{(e)} &= -\frac{1}{4}(1 - \xi)(1 - \eta)(1 + \xi + \eta), & N_5^{(e)} &= \frac{1}{2}(1 - \xi^2)(1 - \eta) \\ N_2^{(e)} &= -\frac{1}{4}(1 + \xi)(1 - \eta)(1 - \xi + \eta), & N_6^{(e)} &= \frac{1}{2}(1 - \eta^2)(1 + \xi) \\ N_3^{(e)} &= -\frac{1}{4}(1 + \xi)(1 + \eta)(1 - \xi - \eta), & N_7^{(e)} &= \frac{1}{2}(1 - \xi^2)(1 + \eta) \\ N_4^{(e)} &= -\frac{1}{4}(1 - \xi)(1 + \eta)(1 + \xi - \eta), & N_8^{(e)} &= \frac{1}{2}(1 - \eta^2)(1 - \xi) \end{aligned} \quad (\text{Q.31})$$

Express a generic quantity w over element (A) as

$$w^A = [w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8] \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \end{bmatrix}. \quad (\text{Q.32})$$

To construct (B) with 9 as a hierarchical node at the quadrilateral center ($\xi = \eta = 0$), we express the value at 9 as the sum of the interpolated center value in (A), plus a correction:

$$w_9 = w_9^A(0, 0) + \tilde{w}_9 \quad (\text{Q.33})$$

The center value is obtained by evaluating (Q.24) at $\xi = \eta = 0$, which gives

$$w_9^A(0, 0) = -\frac{1}{4}(w_1 + w_2 + w_3 + w_4) + \frac{1}{2}(w_5 + w_6 + w_7 + w_8), \quad (\text{Q.34})$$

so that the full form of (Q.25) is

$$w_9 = -\frac{1}{4}(w_1 + w_2 + w_3 + w_4) + \frac{1}{2}(w_5 + w_6 + w_7 + w_8) + \tilde{w}_9. \quad (\text{Q.35})$$

The transformation between non-hierarchical and hierarchical values may be expressed in matrix form:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 \\ \tilde{w}_4 \\ \tilde{w}_5 \\ \tilde{w}_6 \\ \tilde{w}_7 \\ \tilde{w}_8 \\ \tilde{w}_9 \end{bmatrix}. \quad (\text{Q.36})$$

The shape function associated with the internal node is the bubble function

$$N_9^{(e)} = (1 - \xi^2)(1 - \eta^2) \quad (\text{Q.37})$$

With these results the generic interpolation formula for (B) becomes

$$w^B = [w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad \tilde{w}_9] \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \\ N_9^{(e)} \end{bmatrix}, \quad (\text{Q.38})$$

where the shape functions are (Q.36) and (Q.37). [No tildes on the N 's are necessary, because none of these functions changes in going from (A) to (B).] Finally, on specializing this relation to x , y , etc, we obtain the isoparametric representation of element (B):

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & \tilde{x}_9 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & \tilde{y}_9 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} & \tilde{u}_{x9} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} & \tilde{u}_{y9} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \\ N_9^{(e)} \end{bmatrix}. \quad (\text{Q.39})$$

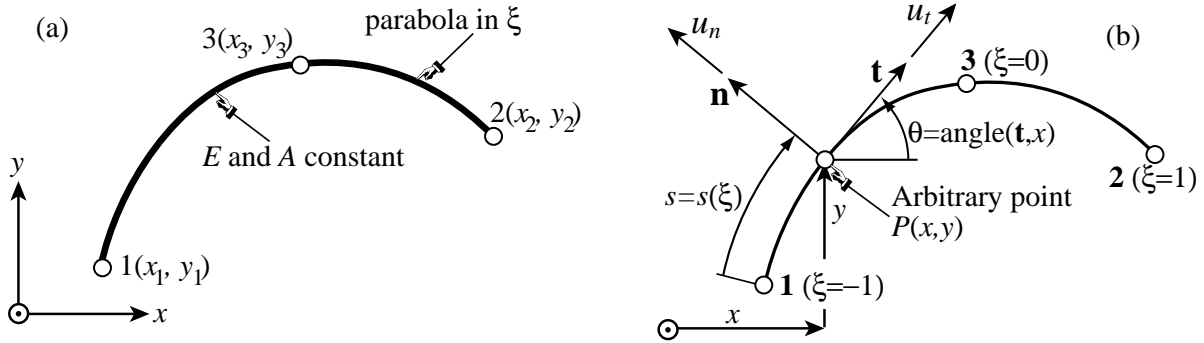


Figure Q.6. Figure (a) shows a 3-node curved bar element with constant modulus E and area A . This element has six degrees of freedom and can only transmit an axial force $N = EAe$. Figure Q.6(b) depicts geometric details covered in the text.

§Q.3. *A 3-Node Curved Bar Element

This section formulates a *curved* curved bar element 3-node bar isoparametric element in which node 3 is allowed to be off alignment from 1 and 2, as depicted in Figure Q.6(a). The derivation provides a first glimpse of techniques used in advanced FEM expositions to develop curved beam and shell elements.

§Q.3.1. *Element Description

The element moves in the $\{x, y\}$ plane. It has six degrees of freedom, namely, the displacements u_{xn} and u_{yn} at nodes $n = 1, 2, 3$. It has constant elastic modulus E and cross section area A , and can transmit only an axial force $N = EAe$, where e is the axial strain.

Additional geometric details are given in Figure Q.6(b). At an arbitrary point $P(x, y)$ of the element one defines the tangential and normal directions by the *unit* vectors \mathbf{t} and \mathbf{n} , respectively.¹ The positive sense of \mathbf{t} corresponds to traversing along from 1 to 2. The normal vector \mathbf{n} is at 90° CCW from \mathbf{t} . The angle formed by \mathbf{t} and x is θ , positive CCW. The arclength coordinate is s , where $ds^2 = dx^2 + dy^2$.

The Cartesian displacement vector at generic point P is $\mathbf{u} = [u_x \ u_y]^T$. The displacement components in the directions \mathbf{t} and \mathbf{n} are the tangential displacement $u_t = \mathbf{u}^T \mathbf{t} = \mathbf{t}^T \mathbf{u}$ and the normal displacement $u_n = \mathbf{u}^T \mathbf{n} = \mathbf{n}^T \mathbf{u}$. The axial strain is defined as

$$e = \frac{du_t}{ds} - \kappa u_n \quad (\text{Q.40})$$

where κ is the inplane bar curvature, the expression of which is derived in §Q.3.3. The corrective term κu_n is known as the *hoop strain* in the theory of curved bars and rods.²

¹ “Unit vectors” means they have unit length: $\mathbf{t}^T \mathbf{t} = 1$ and $\mathbf{n}^T \mathbf{n} = 1$.

² If the bar is straight, $\kappa = 0$ and $e = du_t/ds$. If then x is taken along the bar axis, $s \equiv x$ and $e = du_x/dx$, which is the well known axial strain used in Chapter 12.

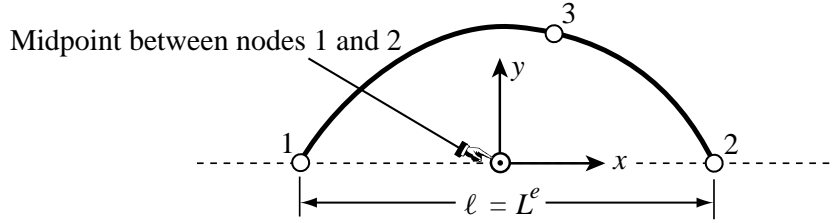


Figure Q.7. Choice of Cartesian local axes to simplify derivations.

The isoparametric definition of this element is³

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \quad (\text{Q.41})$$

Here

$$N_1 = -\frac{1}{2}\xi(1 - \xi), \quad N_2 = \frac{1}{2}\xi(1 + \xi), \quad N_3 = 1 - \xi^2 \quad (\text{Q.42})$$

are the shape functions defined in terms of an isoparametric coordinate ξ that takes the value -1 , 1 and 0 at nodes 1 , 2 and 3 , respectively. Derivatives taken with respect to coordinate ξ will be denoted by a prime; thus $x' = dx/d\xi$, etc. The strain energy taken up by the element is

$$U^{(e)} = \frac{1}{2} \int_{\text{arclength}} EA e^2 ds = \frac{1}{2} \int_{-1}^1 EA e^2 J d\xi = \frac{1}{2} EA \int_{-1}^1 e^2 J d\xi, \quad J = ds/d\xi = s'. \quad (\text{Q.43})$$

To simplify the analytical derivations that follow, we orient $\{x, y\}$ as shown in Figure Q.7, with origin at the midpoint of nodes 1 and 2 . Take $x_3 = \alpha\ell$ and $y_3 = \beta\ell$ where α and β are dimensionless parameters that characterize the deviation of node 3 from the 1 - 2 midpoint, and ℓ is the 1 - 2 distance.⁴

The axial strain e , from the curved rod theory outlined in §Q.3.3, is

$$\begin{aligned} e &= \frac{du_n}{ds} - \kappa u_n = \frac{d(\mathbf{u}^T \mathbf{t})}{ds} - \kappa u_n = \left(\frac{d\mathbf{u}}{ds} \right)^T \mathbf{t} + \mathbf{u}^T \frac{d\mathbf{t}}{ds} - \kappa u_n \\ &= \left(\frac{d\mathbf{u}}{d\xi} \right)^T \frac{d\xi}{ds} \mathbf{t} + \mathbf{u}^T (\kappa \mathbf{n}) - \kappa u_n = J^{-1} \mathbf{t}^T \mathbf{u}' + \kappa u_n - \kappa u_n = J^{-1} \mathbf{t}^T \mathbf{u}'. \end{aligned} \quad (\text{Q.44})$$

in which $J = s' = ds/d\xi$ is the curved-bar Jacobian, and the replacement of $d\mathbf{t}/ds$ by $\kappa \mathbf{n}$ comes from the first Frénet formula given in §Q.3.3. The values of \mathbf{u} and \mathbf{u}' can be calculated from the last two rows of the isoparametric definition:

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \quad (\text{Q.45})$$

³ Superscript (e) is omitted for brevity.

⁴ Axes $\{x, y\}$ in Figure Q.7 are actually the local axes called \bar{x} and \bar{y} in Chapters 2-3. The overbars are omitted to avoid cluttering; they are reintroduced in Figure Q.8. The role of the deviation parameters α and β is similar to the device used in Exercises 16.4 and 16.5 to study the effect of moving node 3 away from the midpoint.

Noting that $N'_1 = dN_1/d\xi = \xi - \frac{1}{2}$, $N'_2 = dN_2/d\xi = \xi + \frac{1}{2}$, $N'_3 = dN_3/d\xi = -2\xi$, we get

$$\mathbf{u}' = \begin{bmatrix} u'_x \\ u'_y \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N'_1 \\ N'_2 \\ N'_3 \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} \xi - \frac{1}{2} \\ \xi + \frac{1}{2} \\ -2\xi \end{bmatrix}. \quad (\text{Q.46})$$

Collecting terms and replacing the expression of \mathbf{t} given in §Q.3.3:

$$e = \mathbf{B} \mathbf{u}^{(e)}, \quad (\text{Q.47})$$

$$\mathbf{B} = J^{-2} [x'N'_1 \quad y'N'_1 \quad x'N'_2 \quad y'N'_2 \quad x'N'_3 \quad y'N'_3], \quad (\text{Q.48})$$

The derivatives $x' = dx/d\xi$ and $y' = dy/d\xi$ are obtained from rows 2-3 of the isoparametric element definition:

$$x' = x_1N'_1 + x_2N'_2 + x_3N'_3 = (\frac{1}{2} - 2\alpha\xi)\ell, \quad y' = y_1N'_1 + y_2N'_2 + y_3N'_3 = -2\beta\xi\ell, \quad (\text{Q.49})$$

whence

$$J = \sqrt{(x')^2 + (y')^2} = \ell \sqrt{(\frac{1}{2} - 2\alpha\xi)^2 + 4\beta^2\xi^2}, \quad (\text{Q.50})$$

Finally,

$$\mathbf{B} = \frac{\ell}{J^2} \begin{bmatrix} (\frac{1}{2} - 2\alpha\xi)(\xi - \frac{1}{2}) & 2\beta\xi(\xi - \frac{1}{2}) & (\frac{1}{2} - 2\alpha\xi)(\xi + \frac{1}{2}) & -2\beta\xi(\xi + \frac{1}{2}) & -2(\frac{1}{2} - 2\alpha\xi)\xi & 2\beta\xi^2 \end{bmatrix}. \quad (\text{Q.51})$$

If the element is straight ($\beta = 0$), $s \equiv x$, $J^2 = \ell^2(\frac{1}{2} - 2\alpha\xi)^2$, $J = \ell(\frac{1}{2} - 2\alpha\xi)$, and \mathbf{B} reduces to

$$\mathbf{B} = \frac{1}{\ell(\frac{1}{2} - 2\alpha\xi)} \begin{bmatrix} \xi - \frac{1}{2} & 0 & \xi + \frac{1}{2} & 0 & -2\xi & 0 \end{bmatrix}. \quad (\text{Q.52})$$

The Jacobian coincides with the expression derived in Chapter 19 for the straight 3-bar element.

Item (b). Here is an implementation of the foregoing \mathbf{B} in *Mathematica* 2.2, as a function that returns a 1×6 matrix:

```
B [xi_,alpha_,beta_,ell_] := {{(1/2-2*alpha*xi)*(xi-1/2),
-2*beta*xi*(xi-1/2), (1/2-2*alpha*xi)*(xi+1/2),
-2*beta*xi*(xi+1/2), -(1/2-2*alpha*xi)*(2*xi),
4*beta*xi^2}}*ell/JJ[xi,alpha,beta,ell];
J [xi_,alpha_,beta_,ell_] := ell*Sqrt[(1/2-2*alpha*xi)^2+4*beta^2*xi^2];
JJ[xi_,alpha_,beta_,ell_] := ell^2* ((1/2-2*alpha*xi)^2+4*beta^2*xi^2);
```

The function JJ above returns the squared Jacobian J^2 ; this separate definition is important for symbolic work, because it bypasses the hard-to-simplify square root. Function J, which returns the Jacobian J , is not needed here but it will be used in the stiffness matrix formation in Question 2.

Following is the verification that \mathbf{B} predicts zero strains under the three two-dimensional rigid body modes (RBMs) for arbitrary α , β and ℓ . It is follow by a uniform-strain test on a straight bar: $\alpha = 0$, $\beta = 0$ but arbitrary ℓ . (The uniform strain test on a curved bar is far trickier and is not required in the test.)

```

ClearAll[alpha,beta,ell,xi];
rm1={1,0,1,0,1,0}; rm2={0,1,0,1,0,1}; (* translations along x,y *)
rm3={0,ell/2,0,-ell/2,beta*ell,-alpha*ell}; (* rotation about z *)
Print["Check zero strain for x-RBM=",Simplify[B[xi,alpha,beta,ell].rm1]];
Print["Check zero strain for y-RBM=",Simplify[B[xi,alpha,beta,ell].rm2]];
Print["Check zero strain for z-RBM=",Simplify[B[xi,alpha,beta,ell].rm3]];
ue={-1/2,0,1/2,0,0,0};
Print["Check unif strain for straight bar=",Simplify[B[xi,0,0,ell].ue]];

```

Running this gives

```

Check zero strain for x-RBM={0}
Check zero strain for y-RBM={0}
Check zero strain for z-RBM={0}

Check unif strain for straight bar={---}
ell

```

These checks can also be done by hand, but there is some error-prone algebra involved.

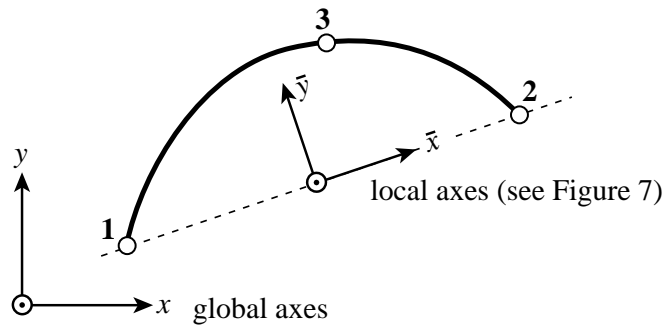


Figure Q.8. Global and local axes for the transformations used in forming the global stiffness matrix.

§Q.3.2. The Stiffness Matrix

Here is an implementation of the element stiffness matrix as module `Stiffness3NodePlaneCurvedBar` that returns a 6×6 matrix:

```

Stiffness3NodePlaneCurvedBar[ncoor_,mprop_,fprop_,opt_]:=
Module[{x1,y1,x2,y2,x3,y3,x21,y21,xm,ym,alpha,beta,ell2,ell,
Em,A,num,B,J,JJ,Kebar,T,Ke},
B [xi_,alpha_,beta_,ell_]:={{(1/2-2*alpha*xi)*(xi-1/2),
-2*beta*xi*(xi-1/2), (1/2-2*alpha*xi)*(xi+1/2),
-2*beta*xi*(xi+1/2), -(1/2-2*alpha*xi)*(2*xi),
4*beta*xi^2}}*ell/JJ[xi,alpha,beta,ell];
J[xi_,alpha_,beta_,ell_]:=ell*Sqrt[(1/2-2*alpha*xi)^2+4*beta^2*xi^2];
JJ[xi_,alpha_,beta_,ell_]:=ell^2*((1/2-2*alpha*xi)^2+4*beta^2*xi^2);
{{x1,y1},{x2,y2},{x3,y3}}=ncoor; {Em}=mprop; A=fprop; num=opt;
{x21,y21}={x2-x1,y2-y1}; {xm,ym}={x1+x2,y1+y2}/2;
ell2=x21^2+y21^2; ell=PowerExpand[Sqrt[ell2]];
alpha=Simplify[ ( (x3-xm)*x21+(y3-ym)*y21)/ell2];

```

```

beta= Simplify[ (-(x3-xm)*y21+(y3-ym)*x21)/ell2];
Print["alpha,beta in Stiffness3NodePlaneCurvedBar=",{alpha,beta}];
B1=B[-Sqrt[3]/3,alpha,beta,ell]; J1=J[-Sqrt[3]/3,alpha,beta,ell];
B2=B[ Sqrt[3]/3,alpha,beta,ell]; J2=J[ Sqrt[3]/3,alpha,beta,ell];
If [num, {B1,J1,B2,J2}=N[{B1,J1,B2,J2}]];
Kebar=Em*A*(J1*Transpose[B1].B1+J2*Transpose[B2].B2);
T={{x21, y21,0,0,0,0},{-y21,x21,0,0,0,0},{0,0, x21,y21,0,0},
    {0,0,-y21,x21,0,0},{0,0,0,0,x21,y21}, {0,0,0,0,-y21,x21}};
Ke=(Transpose[T].Kebar.T)/ell2; (* avoids taking Sqrt[ell2] *)
Return[Ke] ];

```

Arguments. The module has four arguments:

- ncoor Global node coordinates arranged as $\{\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}\}$.
- mprop Material properties supplied as $\{E_m\}$, which is the elastic modulus E .
- fprop Fabrication properties supplied as the cross-section area A .
- opt Processing option: contains logical flag numer; if True it forces floating-point computation.

Internal Functions. Functions B, J and JJ discussed in Question 1 are incorporated inside the body of `Stiffness3NodePlaneCurvedBar`, and their names made local to the module. This is just a precaution against name clashing when the module is incorporated in a larger FEM code.

Local Geometry Analysis. The node coordinates supplied to `Stiffness3NodePlaneCurvedBar` in `ncoor` are *global*, that is, referred to the global axes $\{x, y\}$. The local system $\{\bar{x}, \bar{y}\}$ is defined through the scheme depicted in Figure Q.8. To connect these two systems begin by computing $x_{21} = x_2 - x_1$, $y_{21} = y_2 - y_1$, $\ell^2 = x_{21}^2 + y_{21}^2$, $\ell = +\sqrt{\ell^2}$, $c = \cos \phi = x_{21}/\ell$ and $s = \sin \phi = y_{21}/\ell$. Local and global coordinates of arbitrary points $P(x, y) \equiv P(\bar{x}, \bar{y})$ are related by the transformations

$$\begin{aligned}\bar{x} &= (x - x_m) c + (y - y_m) s, & \bar{y} &= -(x - x_m) s + (y - y_m) c \\ x &= \bar{x} c - \bar{y} s + x_m, & y &= \bar{x} s + \bar{y} c + y_m,\end{aligned}\tag{Q.53}$$

in which $x_m = \frac{1}{2}(x_1 + x_2)$ and $y_m = \frac{1}{2}(y_1 + y_2)$ are the global coordinates of the midpoint between 1 and 2, which is taken as origin of the local system as shown in Figure Q.8. Whence

$$\begin{aligned}\alpha &= \bar{x}_3/\ell = \frac{(x_3 - x_m) c + (y_3 - y_m) s}{\ell} = \frac{(x_3 - x_m) x_{21} + (y_3 - y_m) y_{21}}{\ell^2}, \\ \beta &= \bar{y}_3/\ell = \frac{-(x_3 - x_m) s + (y_3 - y_m) c}{\ell} = \frac{-(x_3 - x_m) y_{21} + (y_3 - y_m) x_{21}}{\ell^2}.\end{aligned}\tag{Q.54}$$

The last expressions for α and β avoid square roots and are those implemented in the module. The local stiffness matrix $\bar{\mathbf{K}}^{(e)}$ is $\text{Kebar} = E_m A * (J_1 * \text{Transpose}[B_1] . B_1 + J_2 * \text{Transpose}[B_2] . B_2)$, where B_1, J_1 , etc., are function evaluations at the Gauss points of the 2-point rule. From inspection, the global stiffness matrix is given by

$$\mathbf{K}^{(e)} = \mathbf{T}^T \bar{\mathbf{K}}^{(e)} \mathbf{T}, \quad \text{where} \quad \mathbf{T} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & c & s & 0 & 0 \\ 0 & 0 & -s & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & s \\ 0 & 0 & 0 & 0 & -s & c \end{bmatrix}\tag{Q.55}$$

Since $c = x_{21}/\ell$ and $s = y_{21}/\ell$, taking square roots of $\ell^2 = x_{21}^2 + y_{21}^2$ can be avoided by slightly rearranging the foregoing transformation as follows:

$$\mathbf{K}^{(e)} = \frac{1}{\ell^2} \hat{\mathbf{T}}^T \bar{\mathbf{K}}^{(e)} \hat{\mathbf{T}}, \quad \text{where} \quad \hat{\mathbf{T}} = \begin{bmatrix} x_{21} & y_{21} & 0 & 0 & 0 & 0 \\ -y_{21} & x_{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{21} & y_{21} & 0 & 0 \\ 0 & 0 & -y_{21} & x_{21} & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{21} & y_{21} \\ 0 & 0 & 0 & 0 & -y_{21} & x_{21} \end{bmatrix}, \quad (\text{Q.56})$$

This is the transformation implemented in `Stiffness3NodePlaneCurvedBar`, in which $\hat{\mathbf{T}}$ is called \mathbf{T} .

Verification. Several tests on the element stiffness are performed now. The following statements form $\mathbf{K}^{(e)}$ for the straight bar with node 3 at the midpoint ($\alpha = \beta = 0$) while keeping E , A and ℓ symbolic:

```
ClearAll[Em,A,alpha,beta,ell,xi];
ncoor={{-ell/2,0},{ell/2,0},{0,0}};
Ke=Stiffness3NodePlaneCurvedBar[ncoor,{Em},A,False]; Ke=Simplify[Ke];
Print["Check Ke for straight bar with 3 at midpoint:"]
Print[Ke//MatrixForm];
```

The result is the stiffness matrix listed in Exercise 16.1. Next, a curved bar stiffness is formed numerically with $E = A = \ell = 1$, $\alpha = 1/5$, $\beta = 1/5$, and tested for rank and rigid body modes (RBMs):

```
Em=A=1; ell=1; alpha=1/5; beta=1/3;
ncoor={{-ell/2,0},{ell/2,0},{alpha,beta}*ell}};
Ke=Stiffness3NodePlaneCurvedBar[ncoor, {Em},A,True];
Print["Ke for Em=A=ell=1, alpha=1/5 and beta=1/5=",Ke//MatrixForm];
rm1={1,0,1,0,1,0}; rm2={0,1,0,1,0,1};
rm3={0,ell/2,0,-ell/2,beta*ell,-alpha*ell};
Print["eigs of Ke=", Chop[Eigenvalues[N[Ke]]]];
Print["Ke.rm1=",Chop[Ke.rm1]]; Print["Ke.rm2=",Chop[Ke.rm2]];
Print["Ke.rm3=",Chop[Ke.rm3]];
```

The results are:

```

                                1  1
alpha,beta in Stiffness3NodePlaneCurvedBar={-, -}
                                5  3

Ke for Em=A=ell=1, alpha=1/5 and beta=1/5=
  1.1042      0.573266      0.137226      -0.0417379      -1.24142      -0.531528
  0.573266      0.31358      -0.0417379      0.141101      -0.531528      -0.454681
  0.137226      -0.0417379      0.816961      -1.1576      -0.954187      1.19933
 -0.0417379      0.141101      -1.1576      1.66183      1.19933      -1.80293
 -1.24142      -0.531528      -0.954187      1.19933      2.19561      -0.667806
 -0.531528      -0.454681      1.19933      -1.80293      -0.667806      2.25761
eigs of Ke={5.36231, 2.98748, 0, 0, 0, 0}
Ke.rm1={0, 0, 0, 0, 0, 0}
Ke.rm2={0, 0, 0, 0, 0, 0}
```

$\mathbf{K}_{e.rm3} = \{0, 0, 0, 0, 0, 0\}$

The eigenvalue distribution show a rank deficiency of 1 (4 zero eigenvalues, one more than $6 - 3 = 3$). This property is explained in §Q.3.4. The RBM checks work fine, as can be expected.

Finally, a local-to-global invariance test is performed by rotating this element by 30° about z and displacing it by 6 and -4 along x and y , respectively. The global coordinates are recomputed and the new $\mathbf{K}^{(e)}$ formed:

```
{xbar1,ybar1},{xbar2,ybar2},{xbar3,ybar3}}=ncoor;
phi=Pi/6; c=Cos[phi]; s=Sin[phi]; xm=6; ym=-4;
x1=xbar1*c-ybar1*s+xm; y1=xbar1*s+ybar1*c+ym;
x2=xbar2*c-ybar2*s+xm; y2=xbar2*s+ybar2*c+ym;
x3=xbar3*c-ybar3*s+xm; y3=xbar3*s+ybar3*c+ym;
ncoor={{x1,y1},{x2,y2},{x3,y3}};
Ke=Stiffness3NodePlaneCurvedBar[ncoor, {Em},A,True]; Ke=N[Ke];
Print["Ke for 30-deg rotated & translated bar:",Ke//MatrixForm];
Print["eigs of Ke=", Chop[Eigenvalues[N[Ke]]]];
```

Running the test gives:

```
alpha,beta in Stiffness3NodePlaneCurvedBar={-, -}
1 1
5 3

Ke for 30-deg rotated & translated bar:
0.41008      0.62898      0.17434      -0.0225469    -0.58442      -0.606433
0.62898      1.0077       -0.0225469    0.103986     -0.606433     -1.11168
0.17434      -0.0225469    2.03069      -0.944636     -2.20503      0.967183
-0.0225469    0.103986     -0.944636     0.448104      0.967183     -0.552089
-0.58442     -0.606433     -2.20503      0.967183      2.78945      -0.36075
-0.606433    -1.11168      0.967183     -0.552089     -0.36075      1.66377
eigs of Ke={5.36231, 2.98748, 0, 0, 0, 0}
```

Note that $\mathbf{K}^{(e)}$ has changed completely. However, the eigenvalues are not changed because \mathbf{T} is orthogonal. Furthermore α and β are also preserved because they are intrinsic geometric properties.

§Q.3.3. Geometric Properties of Plane Curves

The following geometric properties of plane curves are collected to help in the analytical derivations of §Q.3.1. They can be found in any book on differential geometry, differential geometry such as the well known textbook by Struik.⁵

Consider a smooth plane curve given in parametric form

$$x = x(\xi), \quad y = y(\xi). \quad (\text{Q.57})$$

The arclength s is also a function $s = s(\xi)$ of the coordinate ξ ; cf. Figure Q.6.

First we need the Jacobian $J = s' = ds/d\xi$. The quickest way to get it is to differentiate both sides of the identity $ds^2 = dx^2 + dy^2$ with respect to ξ : $2ds s' = 2dx x' + 2dy y'$, whence

$$J = s' = x' \frac{dx}{ds} + y' \frac{dy}{ds} = x' \cos \theta + y' \sin \theta = \sqrt{(x')^2 + (y')^2}, \quad (\text{Q.58})$$

⁵ D. J. Struik, *Lectures on Classical Differential Geometry*, Addison-Wexley, New York, 2nd ed., 1961.

where θ is the angle formed by the tangent vector \mathbf{t} (directed along increasing s) with the x axis; see Figure 1(b).

The tangent \mathbf{t} and normal \mathbf{n} at a point are *unit length* vectors defined by

$$\mathbf{t} = \begin{bmatrix} dx/ds \\ dy/ds \end{bmatrix} = \begin{bmatrix} \cos \theta \\ \sin \theta \end{bmatrix} = J^{-1} \begin{bmatrix} x' \\ y' \end{bmatrix} \quad (\text{Q.59})$$

$$\mathbf{n} = \begin{bmatrix} -dy/ds \\ dx/ds \end{bmatrix} = \begin{bmatrix} -\sin \theta \\ \cos \theta \end{bmatrix} = J^{-1} \begin{bmatrix} -y' \\ x' \end{bmatrix} \quad (\text{Q.60})$$

Here \mathbf{n} has been taken to be at $+90^\circ = +\pi/2$ radians, from \mathbf{t} . The curvature is given by

$$\kappa = \frac{x'y'' - x''y'}{J^3}. \quad (\text{Q.61})$$

Expression (Q.61) is not needed for a bar element. It would be required, however, for a curved beam element as in Exercise Q.8.

The derivative of \mathbf{t} with respect to the arclength s is given by the first Frénet formula

$$\frac{d\mathbf{t}}{ds} = \kappa \mathbf{n}, \quad (\text{Q.62})$$

§Q.3.4. *Why Is the Stiffness Matrix Rank Deficient?

The rank deficiency is due to the presence of an *inextensional zero energy mode* or IZEM.⁶ The IZEM is a bending-like motion of the element that is not a rigid body mode (RBM) but produces no axial stretching or contraction, hence the qualifier “inextensional.” The IZEM produces a rank deficiency of one, no matter how exact the integration rule is, for Gauss rules of 2 or more points. To get an idea of what the IZEM looks like, the bar element with $E = A = \ell = 1$, $\alpha = 1/5$ and $\beta = 1/5$ previously treated in §Q.3.2 is used. The eigenvector analysis of its stiffness $\mathbf{K}^{(e)}$ would not show the IZEM because the zero eigenvalue has multiplicity 4. But three of the eigenvectors of the associated invariant subspace are known: the RBMs. Using a “spectral inflation” technique the three RBMs are separated by raising their eigenvalues to nonzero values. This leaves one zero eigenvalue, which is the IZEM. This mode is now easily captured by an eigenvector analysis. In the following, Ke, rm1, rm2, rm3 were generated by the *Mathematica* statements shown previously.

```
Ke=Ke+Transpose[{rm1}].{rm1}+Transpose[{rm2}].{rm2}+
Transpose[{rm3}].{rm3}; (* Spectrally separate RBMs to isolate 0-energy
mode *)
Print["zero energy mode=",Chop[Eigenvectors[Ke][[6]] ]];
```

Running this gives the IZEM eigenvector:

```
zero energy mode={0.531352,-0.165054,-0.70251,-0.194949,0.171158,0.360003}
```

These six numbers can be physically interpreted by using a graphic display. Here is a plotting module and the driving program:

⁶ These are also called “kinematic mechanisms” in the FEM literature.

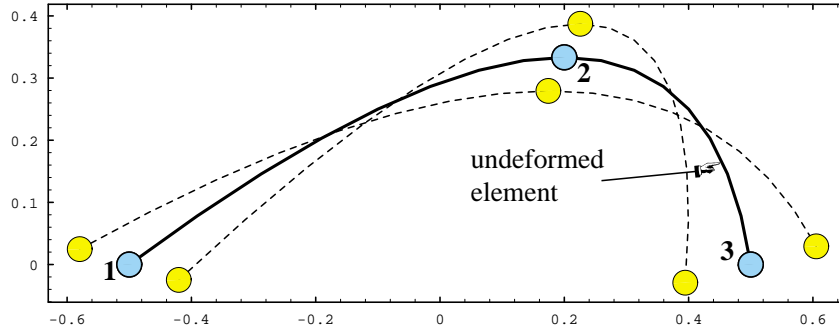


Figure Q.9. The inextensional zero-energy mode (IZEM) of a 3-node curved bar. Depicted for an element with $\ell = 1$, $\alpha = 1/5$ and $\beta = 1/3$.

```

PlotBar3Shape[ncoor_,ue_,amp_,nsub_,th_]:=Module[
  {x1,y1,x2,y2,x3,y3,x21,y21,ux1,uy1,ux2,uy2,ux3,uy3,
   k,xi,N1,N2,N3,m,a,atab,ttab,x,y,xold,yold,p={}},
  {{x1,y1},{x2,y2},{x3,y3}}=N[ncoor];
  {ux1,uy1,ux2,uy2,ux3,uy3}=N[ue];
  N1[xi_]:=-(1-xi)*xi/2; N2[xi_]:= (1+xi)*xi/2; N3[xi_]:=1-xi^2;
  {xold,yold}={x1+amp*ux1,y1+amp*uy1}; xi=-1;
  If [Length[amp]==0,atab={amp},atab=amp];
  If [Length[th]==0,ttab={th}, ttab=th ];
  For [m=1,m<=Length[atab],m++, a=atab[[m]];
    AppendTo[p,Graphics[AbsoluteThickness[ttab[[m]] ]]];
    {xold,yold}={x1+a*ux1,y1+a*uy1}; xi=-1;
    For [k=1, k<=nsub, k++, xi=N[xi+2/nsub];
      x=(x1+a*ux1)*N1[xi]+(x2+a*ux2)*N2[xi]+(x3+a*ux3)*N3[xi];
      y=(y1+a*uy1)*N1[xi]+(y2+a*uy2)*N2[xi]+(y3+a*uy3)*N3[xi];
      AppendTo[p,Graphics[Line[{{xold,yold},{x,y}}]]];
      xold=x; yold=y];
    AppendTo[p,Graphics[RGBColor[1,1,0]]];
    AppendTo[p,Graphics[Disk[{x1+a*ux1,y1+a*uy1},0.02]]];
    AppendTo[p,Graphics[Disk[{x2+a*ux2,y2+a*uy2},0.02]]];
    AppendTo[p,Graphics[Disk[{x3+a*ux3,y3+a*uy3},0.02]]];
    AppendTo[p,Graphics[RGBColor[0,0,0]]];
    AppendTo[p,Graphics[Circle[{x1+a*ux1,y1+a*uy1},0.02]]];
    AppendTo[p,Graphics[Circle[{x2+a*ux2,y2+a*uy2},0.02]]];
    AppendTo[p,Graphics[Circle[{x3+a*ux3,y3+a*uy3},0.02]]];
  ];
  Return[p];
];
ell=1; alpha=1/5; beta=1/3; ncoor={{-ell/2,0},{ell/2,0},{alpha,beta}*ell};
ue={0.531352, -0.165054, -0.70251, -0.194949, 0.171158, 0.360003};
p=PlotBar3Shape[ncoor,ue,{0,-0.15,.15},16,{2,1,1}];
Show[p,Frame->True,FrameTicks->Automatic,AspectRatio->Automatic];

```

The plot produced by this code, after some reformatting and labeling, is shown in Figure Q.9. The dashed curves therein depict the deformed element moving in the IZEM (being an eigenvector, it has arbitrary amplitude; two amplitudes of opposite signs are shown.) Note that the midline does

Appendix Q: MISCELLANEOUS FEM FORMULATION TOPICS

not change length (in the linear approximation), and thus takes no axial strain energy. If bending energy is taken into account, however, the element becomes rank-sufficient.

Homework Exercises for Chapter Q

Miscellaneous FEM Formulation Topics

EXERCISE Q.1 [A:30] An assumed-strain 1D bar element is developed by making a simplifying assumption on $e_{\xi\xi}$: the jacobian J is taken constant and equal to an average value J_0 . Discuss the implications of this assumption as regards compatibility and completeness requirements. Explain how you would construct the strain-displacement matrix (this is not a trivial problem).

EXERCISE Q.2 [A:35] As above, but for a two-dimensional isoparametric element.

EXERCISE Q.3 [A:30] Develop a theory of natural strains and stresses for the 3-node curved bar element formulated in §Q.3.

EXERCISE Q.4 [A:20] Prove the transformation (Q.6). Then express it as a matrix vector product $\mathbf{e} = \mathbf{T}_e \mathbf{e}_{nat}$ where $\mathbf{e} = [e_{xx} \ e_{yy} \ 2e_{xy}]^T$ and $\mathbf{e}_{nat} = [e_{\xi\xi} \ e_{\eta\eta} \ \gamma_{\xi\eta}]^T$.

EXERCISE Q.5 [A:25] As in the previous Exercise, but for the 3D case discussed in §Q.1.3.

EXERCISE Q.6 [A:25] Construct the hierarchical version of the 10-node triangular element of Exercise 17.1 proceeding in two stages:

First stage. Add the six node points 4 through 9 as cubic deviations from the linear shape functions of the three-node triangle. Compare those hierarchical shape functions for these nodes with those found in Exercise 17.1. The results for this stage should be a formula such as (Q.23) with 9 matrix columns plus a list of the shape functions for nodes 1–9. Verify that rigid body motions and constant strain states are preserved

Second stage. Inject the interior node point 10 as a hierarchical function. The results for this stage should be again a formula such as (Q.23) but with 10 matrix columns, and a 10×10 transformation matrix. Verify that rigid body motions and constant strain states are preserved.

EXERCISE Q.7 [A:25] Construct the hierarchical version of the 9-node biquadratic quadrilateral. Proceed in two stages as outlined in the previous exercise. Verify that rigid body motions and constant strain states are preserved.

EXERCISE Q.8 [A:35] Develop a 3-node, curved, plane bar-beam element with parabolic midline geometry defined by three nodes. Element has 8 degrees of freedom: three at the end nodes 1-2 (add the rotation θ_z to the freedoms of the curved bar element) and two at the midnode 3 (same as in the bar element). Assumed uniform flexural rigidity EI decoupled from the bar-axial constitutive relation. Investigate the rank of the numerically integrated stiffness matrix.

R

References (in progress)

TABLE OF CONTENTS

	Page
§R.1. Foreword	R-3
§R.2. Reference Database	R-3

§R.1. Foreword

Collected references for most Chapters (except those in progress) for books

Advanced Finite Element Methods; master-elective and doctoral level, abbrev. AFEM
Advanced Variational Methods in Mechanics; master-elective and doctoral level, abbrev. AVMM
Fluid Structure Interaction; doctoral level, abbrev. FSI
Introduction to Aerospace Structures; junior undergraduate level, abbrev. IAST
Matrix Finite Element Methods in Statics; senior-elective and master level, abbrev. MFEMS
Matrix Finite Element Methods in Dynamics; senior-elective and master level, abbrev. MFEMD
Introduction to Finite Element Methods; senior-elective and master level, abbrev. IFEM
Nonlinear Finite Element Methods; master-elective and doctoral level, abbrev. NFEM

Margin letters are to facilitate sort; will be removed on completion.

Note: Many books listed below are out of print. The advent of the Internet has meant that it is easier to surf for used books across the world without moving from your desk. There is a fast search “metaengine” for comparing prices at URL <http://www.addall.com>: click on the “search for used books” link. Amazon.com has also a search engine, which is poorly organized, confusing and full of unnecessary hype, but does link to online reviews. [Since about 2008, old scanned books posted online on Google are an additional potential source; free of charge if the useful pages happen to be displayed. Such files cannot be downloaded or printed.]

§R.2. Reference Database

A

- [1] Abbott, J. P., An efficient algorithm for the determination of certain bifurcation points, *J. Comput. Appl. Math.*, **4**, pp. 19–27, 1981.
- [2] Abramowitz, M. and Stegun, L. A. (eds.), *Handbook of Mathematical Functions with Formulas, Graphs and Mathematical Tables*, Applied Mathematics Series 55, Natl. Bur. Standards, U.S. Department of Commerce, Washington, D.C., 1964; reprinted by Wiley, 1993.
- [3] Abu-Gazaleh, B. N., Analysis of plate-type prismatic structures, *Ph. D. Dissertation*, Dept. of Civil Engineering, Univ. of California, Berkeley, CA, 1965.
- [4] Ackoff, R. L., Towards a system of systems concept, *Management Sciences*, **17**, 661–671, 1971.
- [5] Adini, A., Analysis of shell structures by the finite element method, *Ph. D. Dissertation*, Dept. of Civil Engineering, University of California, Berkeley, CA., 1961.
- [6] Ahmad, S., Irons, B. M., and Zienkiewicz, O. C., Analysis of thick and thin shell structures by curved finite elements, *Int. J. Numer. Meth. Engrg.*, **2**, 419–451, 1970.
- [7] Aitken, A. C., *Determinants and Matrices*, Oliver and Boyd, Edinburgh and London, 1939 (2nd-9th editions, 1942–56, 9th edition, reset and reprinted, 1967, Greenwood Press, Westport CN, 1983.)
- [8] Allgower, E. L., Simplicial and continuation methods for approximating fixed points and solutions to systems of equations, *SIAM Review*, **22**, 28–85, 1980.
- [9] Allgower, E. L., A survey of homotopy methods for smooth mappings, in *Numerical Solution of Nonlinear Equations*, ed. by E. L. Allgower, K. Glashoff and H.-O. Peitgen, Lecture Notes in Mathematics 878, Springer-Verlag, Berlin, 1981.
- [10] Allgower, E. L., Predictor-corrector and simplicial methods for approximating fixed points and zero points of nonlinear mappings, in *Mathematical Programming: The State of the Art*, ed. by A. Bachem, M. Grötschel and B. Korte, Springer-Verlag, Berlin 15–55, 1983.

Appendix R: REFERENCES (IN PROGRESS)

- [11] Allgower, E. L. and Georg, K., *Numerical Continuation Methods: An Introduction*, Springer-Verlag, Berlin, 1990.
- [12] Allman, D. J., Triangular finite elements for plate bending with constant and linearly varying bending moments, *Proc. IUTAM Conf. on High Speed Computing of Elastic Structures*, Liège, Belgium, 105–136, 1970.
- [13] Allman, D. J., Evaluation of the constant strain triangle with drilling rotations, *Int. J. Numer. Meth. Engrg.*, **26**, 2645–2655, 1988.
- [14] Almroth, B. O. and Brogan, F. A., Bifurcation buckling as an approximation to the collapse load for general shells, *AIAA J.*, **10**, 121–140, 1972.
- [15] Almroth, B. O. and Felippa, C. A., Structural stability, in *Structural Analysis Computer Programs: Surveys, Assessments and Availability*, ed. by W. Pilkey, K. Saczalski and H. Schaeffer, University Press of Virginia, Charlottesville, Va. 499–539, 1974.
- [16] Almroth, B. O., Stern, P., and Brogan, F. A., Automated choice of global shape functions in structural analysis, *AIAA J.*, **16**, 525–528, 1978.
- [17] Alvin, K., de la Fuente, H. M., Haugen, B., and Felippa, C.A., Membrane triangles with corner drilling freedoms: I. The EFF element, *Finite Elem. Anal. Des.*, **12**, 163–187, 1992.
- [18] Ambartsumyan, S., *Theory of Anisotropic Plates: Strength, Stability, and Vibrations*, Russian translation, Hemisphere Pubs. Corp., 1991.
- [19] Ames, W. F., *Nonlinear Partial Differential Equations in Engineering*, Academic Press, New York, 1965.
- [20] Anonymous, The NASTRAN Theoretical Manual, NASA SP-221, 1970; The NASTRAN User's Manual, NASA SP-222, 1970; The NASTRAN Programmer's Manual, NASA SP-223, 1970; The NASTRAN Demonstration Problem Manual, NASA SP-223, 1970.
- [21] Anselone, P. M. and Moore, R. H., An extension of the Newton-Kantorovich method for solving nonlinear equations with an application to elasticity, *J. Math. Anal. Appl.*, **13**, 476–501, 1966.
- [22] Argyris, J. H. and Kelsey, S., *Energy Theorems and Structural Analysis*, Butterworth, London, 1960; Part I reprinted from *Aircr. Engrg.*, **26**, Oct-Nov 1954 and **27**, April-May 1955.
- [23] Argyris, J. H., Kelsey, H., and Kamel, H., Matrix Methods of Structural Analysis: A Précis of Recent Developments, in *Matrix Methods of Structural Analysis*, ed. by B. M. Fraeijs de Veubeke, AGARDograph 72, Pergamon Press, Oxford, 1–164, 1964.
- [24] Argyris, J. H., Elasto-plastic matrix displacement analysis of three-dimensional continua, *J. Royal Aero. Soc.*, **69**, 633–636, 1965.
- [25] Argyris, J. H., Triangular elements with linearly varying strain for the Matrix Displacement Method, *J. Royal Aero. Soc.*, **69**, 711–713, 1965.
- [26] Argyris, J. H., Continua and discontinua, in *Proceedings Conference on Matrix Methods in Structural Engineering*, AFFDL-TR-66-80, Wright-Patterson AFB, Dayton, Ohio, 11–189, 1966.
- [27] Argyris, J. H., Continua and discontinua, in *Proceedings 1st Conference on Matrix Methods in Structural Mechanics*, ed. by R. Bader et. al., AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 10–170, 1966.
- [28] Argyris, J. H., Matrix analysis of three-dimensional elastic media: small and large displacements, *AIAA J.*, **3**, 45–51, 1965.
- [29] Argyris, J. H. and Bronlund, O. E., The natural factor formulation of the stiffness matrix displacement method, *Comp. Meths. Appl. Mech. Engrg.*, **5**, 97–119, 1975.

- [30] Argyris, J. H., Dunne, P. C., Malejannakis, G. A., and Schelkle, E., A simple triangular facet shell element with applications to linear and nonlinear equilibrium and elastic stability problems, *Comp. Meths. Appl. Mech. Engrg.*, **11**, 215–247, 1977.
- [31] Argyris, J. H., An excursion into large rotations, *Comp. Meths. Appl. Mech. Engrg.*, **32**, 85–155, 1982.
- [32] Argyris, J. H. *et. al.* (eds), *FENOMECH 1984, 3rd Int. Conf. on Finite Elements in Nonlinear Mechanics*, Stuttgart, 1984.
- [33] Argyris, J. H. and Mlejnek, H.-P., *Die Methode der Finiten Elemente, Vol. I–III*, Vieweg, Braunschweig, 1986, 1987 and 1988.
- [34] Archer, J. S., Consistent mass matrix for distributed mass systems, *J. ASCE Struct. Div.*, **89**, 161–178, 1963.
- [35] Archer, J. S., Consistent mass matrix formulation for structural analysis using finite element techniques, *AIAA J.*, **3**, 1910–1918, 1965.
- [36] Armen, H., Assumptions, models, and computational methods for plasticity, *Computers & Structures*, **10**, 161–174, 1979.
- [37] Ashwell, G. H. and R. H. Gallagher, R. H., (eds.), *Finite Elements for Thin Shells and Curved Members*, Wiley, New York, 1976.
- [38] Atluri, S. N., On “hybrid” finite-element models in solid mechanics, In: *Advances in Computer Methods for Partial Differential Equations*. Ed. by R. Vichnevetsky, AICA, Rutgers University, 1975, 346–356.
- [39] Atluri, S. N., Gallagher, R. N., and Zienkiewicz, O. C., (eds.), *Hybrid and Mixed Finite Element Methods*, Wiley, New York, 1983.
- [40] Atluri, S. N., and Reissner, E., On the formulation of variational theorems involving volume constraints, *Comput. Mech.*, **5**, 337–344, 1989.
- [41] Avila, J. H., The feasibility of continuation methods for nonlinear equations, *SIAM J. Numer. Anal.*, **11**, 102–120, 1974.

B

- [42] Ballarini, R., The Da Vinci-Bernoulli-Euler beam theory?, *Mech. Engrg. Magazine Online*, 2003. Available from <http://memagazine.org/contents/current/webonly/webex418.html>
- [43] Bakhvalov, N. S., *Numerical Methods*, Mir Publishers, Moscow, 1975.
- [44] Banachiewicz, T., Zur Berechnung der Determinanten, wie auch der Inversen, und zur darauf basierten Auflösung der Systeme linearer Gleichungen, *Acta Astronomica Series C*, **3**, 41–97, 1937.
- [45] Barlow, J., Optimal stress locations in finite element models, *Int. J. Numer. Meth. Engrg.*, **10**, 243–251, 1976.
- [46] Barlow, J., More on stress-points reduced integration, element distortions and error estimation, *Int. J. Numer. Meth. Engrg.*, **28**, 1487–1504, 1989.
- [47] Bartlett, M. S., An inverse matrix adjustment arising in discriminant analysis, *Ann. Math. Stat.*, **22**, 107–111, 1951.
- [48] Basu, A. K., New light on the Nayak alpha technique, *Int. J. Numer. Meth. Engrg.*, **6**, 152–153, 1973.
- [49] Bathe, K.-J., Ramm E., and Wilson, E. L., Finite element formulations for large deformation dynamic analysis, *Int. J. Numer. Meth. Engrg.*, **7**, 255–271, 1973.

Appendix R: REFERENCES (IN PROGRESS)

- [50] Bathe, K.-J. and Wilson, E. L., *Numerical Methods for Finite Element Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1976.
- [51] Bathe, K. J., Oden, J. T., and W. Wunderlich (eds.), *Formulation and Computational Algorithms in Finite Element Analysis*, MIT Press, Cambridge, 1977.
- [52] Bathe, K. J. and A. Cimento, Some practical procedures for the solution of nonlinear finite element equations, *Comp. Meths. Appl. Mech. Engrg.*, **22**, 59–85, 1980.
- [53] Bathe, K.-J., *Finite Element Procedures in Engineering Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1982.
- [54] Bathe, K. J. and Dvorkin, E., On the automatic solution of nonlinear finite element equations, *Computers & Structures*, **17**, 871–879, 1983.
- [55] Bathe, K. J. and Chaudhary, A., A solution method for planar and axisymmetric contact problems, *Int. J. Numer. Meth. Engrg.*, **21**, 65–88, 1985.
- [56] Bathe, K.-J. and Dvorkin, E. N., A four-node plate bending element based on Mindlin-Reissner plate theory and a mixed interpolation, *Int. J. Numer. Meth. Engrg.*, **21**, 367–383, 1985.
- [57] Batoz, J. L. and Dhatt, G., Incremental displacement algorithms for nonlinear problems, *Int. J. Numer. Meth. Engrg.*, **14**, 1262–1267, 1979.
- [58] Batoz, J. L., Bathe, K.-J., and Ho, L.-W., A study of three-node triangular plate bending elements, *Int. J. Numer. Meth. Engrg.*, **15**, 1771–1812, 1980.
- [59] Batoz, J. L., An explicit formulation for an efficient triangular plate-bending element. *Int. J. Numer. Meth. Engrg.*, **18**, 1077–1089, 1982.
- [60] Battini, J.-M. and Pacoste, C., Co-rotational beam elements with warping effects in instability problems, *Comp. Meths. Appl. Mech. Engrg.*, **191**, 1755–1789, 2002.
- [61] Battini, J.-M. and Pacoste, C., Plastic instability of beam structures using co-rotational elements, *Comp. Meths. Appl. Mech. Engrg.*, **191**, 5811–5831, 2002.
- [62] Bazant, Z. P. and Cedolin, L., *Stability of Structures: Elastic, Inelastic, Fracture and Damage Theories*, 2nd ed., World Scientific, Singapore, 2010. First edition 1988.
- [63] Bazeley, G. P., Cheung, Y. K., Irons, B. M., and Zienkiewicz, O. C., Triangular elements in plate bending – conforming and nonconforming solutions, in *Proc. 1st Conf. Matrix Meth. Struc. Mech.*, ed. by R. Bader et. al., AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 1966, 547–576.
- [64] Beck, M., Die Knicklast des eiseiting eigenspannen, tangential gedrückten Stabes, *Z. Angew. Math. Phys.*, **3**, No. 3, 1952.
- [65] Becker, M., *The Principles and Applications of Variational Methods*, MIT Press, Cambridge, 1964.
- [66] Beer, F. P. and Johnston, E. R., *Mechanics of Materials*, McGraw-Hill, 2nd ed. 1992.
- [67] Bellman, R., *Introduction to Matrix Analysis*, McGraw-Hill, New York, 1960.
- [68] Bellman, R., *Perturbation Techniques in Mathematics, Physics and Engineering*, Holt, Rinehart and Winston, New York, 1964. Reprinted by Dover, 2003.
- [69] Belytschko, T. and Hsieh, B. J., Nonlinear transient finite element analysis with convected coordinates, *Int. J. Numer. Meth. Engrg.*, **7**, 255–271, 1973.
- [70] Belytschko, T. and Mullen, R., On dispersive properties of finite element solutions, in *Modern Problems in Elastic Wave Propagation*, ed. by J. Miklowitz and J. D. Achenbach, Wiley, New York, 67–82, 1978.
- [71] Belytschko, T. and Hsieh, B. J., Application of higher order corotational stretch theories to nonlinear finite element analysis, *Computers & Structures*, **11**, 175–182, 1979.

- [72] Belytschko, T. and Hsieh, B. J., An overview of semidiscretization and time integration operators, Chapter 1 in *Computational Methods for Transient Analysis*, ed. by T. Belytschko and T. J. R. Hughes, North-Holland, Amsterdam, 1–66, 1983.
- [73] Belytschko, T. and Hughes, T. J. R. (eds.), *Computational Methods for Transient Analysis*, Elsevier Sci. Pubs., Ltd., 1983.
- [74] Belytschko, T., Stolarski, H., and Carpenter, N., A C^0 triangular plate element with one-point quadrature, *Int. J. Numer. Meth. Engrg.*, **20**, 787–802, 1984.
- [75] Belytschko, T., Stolarski, H., Liu, W. K., Carpenter, N., and Ong, J., Stress projection for membrane and shear locking in finite elements, *Comp. Meths. Appl. Mech. Engrg.*, **51**, 221–258, 1985.
- [76] Belytschko, T., Liu, W. K., and Engelmann, B. E., The gamma elements and related developments, in *Finite Element Methods for Plate and Shell Structures, Vol. I: Element Technology*, ed. by T. J. R. Hughes and E. Hinton, Pineridge Press, Swansea, U.K., 316–347, 1986.
- [77] Ben-Israel, A. and Greville, T. N. E., *Generalized Inverses: Theory and Applications* Springer-Verlag, New York, 2nd ed., 2003.
- [78] Bergan, P. G. and Horrigmoe, G., Incremental variational principles and finite element models for nonlinear problems, *Comp. Meths. Appl. Mech. Engrg.*, **7**, 201–217, 1976.
- [79] Bergan, P. G. and Hanssen, L., A new approach for deriving ‘good’ finite elements, MAFELAP II Conference, Brunel University, 1975, in *The Mathematics of Finite Elements and Applications – Volume II*, ed. by J. R. Whiteman, Academic Press, London, 483–497, 1976.
- [80] Bergan, P. G., Horrigmoe, G., Krakeland, B., and Søreide, T. H., Solution techniques for nonlinear finite element problems, *Int. J. Numer. Meth. Engrg.*, **12**, 1677–1696, 1978.
- [81] Bergan, P. G., Finite elements based on energy-orthogonal functions, *Int. J. Numer. Meth. Engrg.*, **15**, 1141–1555, 1980.
- [82] Bergan, P. G., Solution algorithms for nonlinear structural problems, *Computers & Structures*, **12**, 497–509, 1980.
- [83] Bergan, P. G. and Simons, J., Hyperplane displacement control methods in nonlinear analysis, in *Innovative Methods for Nonlinear Problems*, ed. by W. K. Liu, T. Belytschko and K. C. Park, Pineridge Press, Swansea, U.K., 345–364, 1984.
- [84] Bergan, P. G. and Wang, X., Quadrilateral plate bending elements with shear deformations, *Computers & Structures*, **19**, 25–34, 1984.
- [85] Bergan, P. G. and Nygård, M. K., Finite elements with increased freedom in choosing shape functions, *Int. J. Numer. Meth. Engrg.*, **20**, 643–664, 1984.
- [86] Bergan, P. G. and Felippa, C. A., A triangular membrane element with rotational degrees of freedom, *Comp. Meths. Appl. Mech. Engrg.*, **50**, 25–69, 1985.
- [87] Bergan, P. G., Bathe, K. J., and W. Wunderlich (eds), *Finite Element Methods for Nonlinear Problems*, Springer, Berlin, 1986.
- [88] Bergan, P. G. and Nygård, M. K., Nonlinear shell analysis using Free Formulation finite elements, in *Finite Element Methods for Nonlinear Problems*, Springer Verlag, Berlin, 317–338, 1989.
- [89] Bickford, W. B. B., *Advanced Mechanics of Materials*, Addison-Wesley Longman, Menlo Park, CA, 1998.
- [90] Biot, M. A., *The Mechanics of Incremental Deformations*, McGraw-Hill, New York, 1965.
- [91] Bird, R. B., Armstrong, R. C., and Hassager, O., *Dynamics of Polymeric Liquids. Vol 1: Fluid Dynamics*, Wiley, New York, 1977.

Appendix R: REFERENCES (IN PROGRESS)

- [92] Bjærum, R. O., Finite element formulations and solution algorithms for buckling and collapse analysis of thin shells. *Dr. Ing. Thesis*, Div. of Structural Mechanics, NTH, Trondheim, Norway, 1992.
- [93] Bodewig, E., *Matrix Calculus*, North-Holland, Amsterdam, 1956. (Second revised and enlarged edition 1959.)
- [94] Bogner, F. K., Fox, R. L., and Schmidt Jr., L. A., The generation of interelement compatible stiffness and mass matrices by the use of interpolation formulas, *Proc. Conf. on Matrix Methods in Structural Mechanics*, WPAFB, Ohio, 1965, in *AFFDL TR 66-80*, 397–444, 1966.
- [95] Boley, B. A. and Wiener, J. H., *Theory of Thermal Stresses*, Wiley, New York, 1960.
- [96] Bolotin, V. V., *Nonconservative Problems of the Theory of Elastic Stability*, Pergamon Press, Oxford, 1963; English translation from the 1961 Russian edition.
- [97] Bolotin, V. V., *The Dynamic Stability of Elastic Systems*, Holden Day, San Francisco, 1964; English translation from the 1960 Russian edition.
- [98] Boresi, A. P., Schmidt, R. J., and Sidebottom, O. M., *Advanced Mechanics of Materials*, 5th ed., Wiley, 1993.
- [99] Born, M. and Huang, K., *Dynamical Theory of Crystal Lattices*, Oxford, London, 1954.
- [100] Boggs, P. T., The solution of nonlinear systems of equations by A-stable integration techniques, *SIAM J. Numer. Anal.*, **8**, 767–785, 1971.
- [101] Branin, F. H. and Hoo, S. K., A method for finding multiple extrema of a function of n variables, in *Numerical Methods for Nonlinear Optimization*, ed. by F. A. Lootsma, Academic Press, London, 1972.
- [102] Brauer, A., Limits for the characteristic roots of matrices IV: Applications to stochastic matrices, *Duke Math. J.*, **19**, 75–91, 1952.
- [103] Brent, R. P., Some efficient algorithms for solving systems of nonlinear equations, *SIAM J. Numer. Anal.*, **10**, 327–344, 1973.
- [104] Brezzi, F., Rappaz, J., and Raviart, P. A., Finite dimensional approximation of nonlinear problems, Part 3: simple bifurcation points, *Numer. Math.*, **38**, 1–30, 1981.
- [105] Brew, J. S. and Morton, D. M., Nonlinear structural analysis by dynamic relaxation, *Int. J. Numer. Meth. Engrg.*, **3**, 463–483, 1971.
- [106] Brillouin, L., *Wave Propagation in Periodic Structures*, Dover, New York, 1946.
- [107] Brogan, F. A. and Almroth, B. O., Practical methods for elastic collapse analysis of shell structures, *AIAA J.*, **9**, 2321–2325, 1971.
- [108] Broyden, C. G., A class of methods for solving nonlinear simultaneous equations, *Math. Comp.*, **19**, 577–593, 1965.
- [109] Broyden, C. G., Quasi-Newton methods and their application to function minimization, *Math. Comp.*, **21**, 368–381, 1967.
- [110] Brush, D. O. and Almroth, B. O., *Buckling of Bars, Plates and Shells*, McGraw-Hill, New York, 1975.
- [111] Bushnell, D., A strategy for the solution of problems involving large deflections, plasticity and creep, *Int. J. Numer. Meth. Engrg.*, **11**, 683–708, 1977.
- [112] Bushnell, D., Buckling of shells – pitfall for designers, *AIAA J.*, **19**, 1183–1226, 1981.
- [113] Bushnell, D., Plastic buckling, Ch. 2.4 in *Pressure Vessels and Piping Design Technology – A Decade of Progress*, ed. by S. Y. Zamrik and D. Dietrich, Book No. G00213, ASME, New York, 47–117, 1982.
- [114] Bushnell, D., *Computerized Buckling Analysis of Shells*, M. Nijhoff Pubs., Dordrecht, 1985.

- [115] Butcher, J. C., *Numerical Methods for Ordinary Differential Equations*, Wiley, Chichester, 2nd ed., 2003.
- [116] Byrne, G. D. and Hall, C. A., *Numerical Solution of Systems of Nonlinear Algebraic Equations*, Academic Press, London, 1973.

C

- [117] Calladine, C. R., *Engineering Plasticity*, Pergamon Press, 1969.
- [118] Calladine, C. R., *Theory of Shell Structures*, Cambridge University Press, 1983.
- [119] Campbell, S. L. and Meyer, C. D., *Generalized Inverses of Linear Transformations*, Dover, New York, 1991.
- [120] Caramanlian, C., A solution to the C^1 continuity problem in plate bending, *Int. J. Numer. Meth. Engrg.*, **17**, 1291-1317, 1983.
- [121] Caratheodory, C., *Calculus of Variations and Partial Differential Equations of the First Order*, Chelsea, New York, 1982. Translated reprint of the original German edition, Teubner, Berlin, 1935.
- [122] Carey, G. F., A unified approach to three finite element theories for geometric nonlinearities, *Comp. Meths. Appl. Mech. Engrg.*, **4**, 69–79, 1974.
- [123] Carr, A. J., A refined finite element analysis of thin shell structures including dynamic loadings, *Ph. D. Dissertation*, Department of Civil Engineering, University of California at Berkeley, Berkeley, CA, 1968.
- [124] Cassel, A. C., Shells of revolution under arbitrary loading and the use of fictitious densities in dynamic relaxation, *Proc. Inst. Civil Engrs.*, **45**, 65–78, 1970.
- [125] Cayley, A. *Collected Works*, Cambridge Univ. Press, 1898.
- [126] Ceruzzi, P. E., *A History of Modern Computing*, The MIT Press, Cambridge, MA, 1998.
- [127] Chan, T. F. and Keller, H. B., Arclength continuation and multigrid techniques for nonlinear elliptic eigenvalue problems, *SIAM J. Sci. Statist. Comput.*, **3**, 1982.
- [128] Chandrasekhar, S., *Radiative Transfer*, Dover, New York, 1960.
- [129] Cheng, H. and Gupta, K. C., An historical note on finite rotations, *J. Appl. Mech.*, **56**, 139–145, 1989.
- [130] Chiesa, M., Skallerud, B. and Gross, D., Closed form line spring yield surfaces for deep and shallow cracks: formulation and numerical performance, *Computers & Structures*, **80**, 533–545, 2002.
- [131] Chow, S. N., Mallet-Paret, J., and Yorke, J. A., Finding zeros of maps: homotopy methods that are constructive with probability one, *Math. Comp.*, **32**, pp. 887-899, 1978.
- [132] Chow, S. N. and J. K. Hale, *Methods of Bifurcation Theory*, Springer-Verlag, New York, 1982.
- [133] Cline.Paper.SIAMR.1964 Cline, R. E., Note on the generalized inverse of the product of matrices, *SIAM Review*, **6**, 57–58, 1964.
- [134] Clough, R. W., The finite element method in plane stress analysis, *Proc. 2nd ASCE Conf. on Electronic Computation*, Pittsburgh, Pa, 1960.
- [135] Clough, R. W., The stress distribution of Norfolk Dam, Univ. of California at Berkeley, *Inst. Res. Rept.*, Ser. 100, **19**, March 1962, rev. August 1962.
- [136] Clough, R. W., The finite element method in structural mechanics, in *Stress Analysis*, ed. by O. C. Zienkiewicz and G. S. Holister, Wiley, London, 85–119, 1965.

Appendix R: REFERENCES (IN PROGRESS)

- [137] Clough, R. W. and Tocher, J. L., Finite element stiffness matrices for the analysis of plate bending. In *Proceedings 1st Conference on Matrix Methods in Structural Mechanics*, ed. by R. Bader et. al., AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 515–547, 1966.
- [138] Clough, R. W. and Felippa, C. A., A refined quadrilateral element for analysis of plate bending, Proc. 2nd Conf. on Matrix Methods in Structural Mechanics, WPAFB, Ohio, 1965, in *AFFDL TR 69-23*, 1969.
- [139] Clough, R. W., Analysis of structural vibrations and dynamic response, in *Recent Advances in Matrix Methods of Structural Analysis and Design*, ed by R. H. Gallagher, Y. Yamada and J. T. Oden, University of Alabama Press, Huntsville, AL, 441–486, 1971.
- [140] Clough, R. W. and Penzien, J., *Dynamics of Structures*, McGraw-Hill, 1975; 2nd ed., 1993.
- [141] Clough, R. W., Comparison of three-dimensional finite elements, in *Symposium on Application of Finite Element Methods in Civil Engineering*, ed. by W. H. Rowan and R. M. Hackett, TN ASCE, Nashville, 1969.
- [142] Clough, R. W., The finite element method after twenty-five years: a personal view, *Computers & Structures*, **12**, 361–370, 1980.
- [143] Clough, R. W., The finite element method – a personal view of its original formulation, in *From Finite Elements to the Troll Platform – the Ivar Holand 70th Anniversary Volume*, ed. by K. Bell, Tapir, Norway, 89–100, 1994.
- [144] Clough, R. W. and Wilson, E. L., Early finite element research at Berkeley, Proc. 5th US National Conf. Comp. Mech, Boulder, CO, August 1999.
- [145] Collatz, *The Numerical Treatment of Differential Equations*, 3rd ed., Springer, Berlin, 1960.
- [146] Cook, R. D., Malkus, D. S., and Plesha, M. E., *Concepts and Application of Finite Element Methods*, 3rd ed., Wiley, New York, 1989.
- [147] Cools, R., Constructing cubature formulas - the science behind the art, *Acta Numerica*, Cambridge University Press, **6**, 1–54, 1999.
- [148] Cools, R., Monomial cubature rules since “Stroud”: a compilation — Part 2, *J. Comput. Appl. Math.*, **112**, 21–27, 1999.
- [149] Cools, R., An encyclopædia of cubature formulas, *J. Complexity*, **19**, 445–453, 2003.
- [150] Courant, R. and Hilbert, D., *Methods of Mathematical Physics*, 2 vols, Interscience Pubs, 1962. Translation of *Methoden der Mathematischen Physik*, Springer-Verlag, Berlin, 1937.
- [151] Courant, R., Variational methods for the solution of problems in equilibrium and vibrations, *Bull. Amer. Math. Soc.*, **49**, 1–23, 1943; reprinted in *Int. J. Numer. Meth. Engrg.*, **37**, 643–645, 1994.
- [152] Cowper, G. R., Kosko, E., Lindberg, G. M., and Olson, M. O., Static and dynamic applications of a high-precision triangular plate bending element, *AIAA J.*, **7**, 1957–1965, 1969.
- [153] Coxeter, H.S.M., Barycentric coordinates, §13.7 in *Introduction to Geometry*, 2nd ed., Wiley, New York, 216–221, 1969.
- [154] Crabtree, D. E. and Haynsworth, E. V., An identity for the Schur complement of a matrix, *Proc. Amer. Math. Soc.*, **22**, 364–366, 1969.
- [155] Crandall, S. H., *Engineering Analysis: A Survey of Numerical Procedures*, McGraw-Hill, New York, 1956.
- [156] Crandall, M. G. and Rabinowitz, P. H., Bifurcations from simple eigenvalues, *J. Funct. Anal.*, **8**, 321–340, 1971.

- [157] Crisfield, M. A., A faster modified Newton-Raphson iteration, *Comp. Meths. Appl. Mech. Engrg.*, **20**, 267–278, 1979.
- [158] Crisfield, M. A., Incremental/iterative Solution procedures for nonlinear structural analysis, in *Numerical Methods for Nonlinear Problems, Vol. 1*, ed. by C. Taylor, E. Hinton and D. R. J. Owen, Pineridge Press, Swansea, U. K., 1980.
- [159] Crisfield, M. A., An incremental-iterative algorithm that handles snap-through, *Computers & Structures*, **13**, 55–62, 1981.
- [160] Crisfield, M. A., An arc-length method including line searches and accelerations, *Int. J. Numer. Meth. Engrg.*, **19**, 1269–1289, 1983.
- [161] Crisfield, M. A., A four-noded thin plate bending element using shear constraints – a modified version of Lyons’ element, *Comp. Meths. Appl. Mech. Engrg.*, **39**, 93–120, 1983.
- [162] Crisfield, M. A., A quadratic Mindlin element using shear constraints, *Computers & Structures*, **18**, 833–852, 1984.
- [163] Crisfield, M. A., Explicit integration and the isoparametric arch and shell elements, *Comm. Appl. Numer. Meth.*, **2**, 181–187, 1986.
- [164] Crisfield, M. A., A consistent corotational formulation for nonlinear three-dimensional beam element, *Comp. Meths. Appl. Mech. Engrg.*, **81**, 131–150, 1990.
- [165] Crisfield, M. A. and Moita, G. F., A unified co-rotational for solids, shells and beams, *Int. J. Solids Struc.*, **33**, 2969–2992, 1996.
- [166] Crisfield, M. A., *Nonlinear Finite Element Analysis of Solids and Structures. Vol 1: Basic Concepts*, Wiley, Chichester, 1991.
- [167] Crisfield, M. A., *Nonlinear Finite Element Analysis of Solids and Structures. Vol 2: Advanced Topics*, Wiley, Chichester, 1997.
- [168] Crowe, M. J., *A History of Vector Analysis: The Evolution of the Idea of a Vectorial System*, Dover, New York, 1967; reprinted with corrections 1994.
- [169] Cross, H., Analysis of continuous frames by distributing fixed-end moments, Proceedings American Society of Civil Engineers (ASCE), 919–928, 1930, and Transactions ASCE 96, 1–10, 1932; reprinted in *Numerical Methods of Analysis in Engineering*, ed. by L. E. Grinter, MacMillan, New York, 1–13, 1948.

D

- [170] D’Alembert, J. L., *Traité de Dynamique*, Paris, 1743. Reprinted by Gauthier-Villars, Paris, 1921; available from Google as eBook.
- [171] D’Alembert, J. L., *Traité de l’Équilibre et du Mouvement des Fluides pour servir de suite au Traité de Dynamique*, Paris, 2nd. ed., 1770; downloadable from European Cultural Heritage Online (ECHO).
- [172] Dahlquist, G. and Björk, A. , *Numerical Methods*, Prentice-Hall, Englewood Cliffs, N. J., 1974.
- [173] Davidenko, D. F., On a new method of numerical solution of systems of nonlinear equations (in Russian), *Dokl. Akad. Nauk. USSR*, **88**, 601–602, 1953.
- [174] Day, A. S., An introduction to dynamic relaxation, *The Engineer*, **219**, 218–221, 1965.
- [175] Davis, H. T., *Introduction to Nonlinear Differential and Integral Equations*, Dover, New York, 1962.
- [176] Decker, D. W. and Keller, H. B., Path following near bifurcation, *Comm. Pure Appl. Math. Anal.*, **34**, 149–175, 1984.

Appendix R: REFERENCES (IN PROGRESS)

- [177] Decker, D. W., Keller, H. B., and Kelley, C. T., Convergence rates for Newton's method at singular points, *SIAM J. Numer. Anal.*, **20**, 296–314, 1984.
- [178] Deist, F. H. and Sefor, L., Solution of systems of nonlinear equations by parameter variation, *Computing J.*, **10**, 78–82, 1967.
- [179] Den Heijer, C. and Rheinboldt, W. C., On steplength algorithms for a class of continuation methods, *SIAM J. Numer. Anal.*, **18**, 925–948, 1981.
- [180] Dennis, J. E. and Moré, J., Quasi-Newton methods: motivation and theory, *SIAM Review*, **19**, 46–84, 1977.
- [181] Dennis, J. E. and Schnabel, R., *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
- [182] Dhatt, G., An efficient triangular shell element, *AIAA J.*, **8**, No. 11, 2100–2102, 1970.
- [183] Doherty, W. P., Wilson E. L., and Taylor, R. L., Stress analysis of axisymmetric solids utilizing higher order quadrilateral finite elements, SESM Report 69-3, Department of Civil Engineering, University of California, Berkeley, 1969.
- [184] Donnell, L. H., *Beams, Plates and Shells*, McGraw-Hill, 1976.
- [185] Duncan, W. J. and Collar, A. R., A method for the solution of oscillations problems by matrices, *Phil. Mag.*, Series 7, **17**, 865–885, 1934.
- [186] Duncan, W. J. and Collar, A. R., Matrices applied to the motions of damped systems, *Phil. Mag.*, Series 7, **19**, 197–214, 1935.
- [187] Duncan, W. J., Some devices for the solution of large sets of simultaneous linear equations, *Phil. Mag.*, Series 7, **35**, 660–670, 1944.
- [188] Dvorkin, E. N. and Bathe, K.-J., A continuum mechanics based four-node shell element for general nonlinear analysis, *Engrg. Comput.*, **1**, 77–88, 1984.

E

- [189] Edelsbrunner, H., *Geometry and Topology for Mesh Generation*, Cambridge Univ. Press, Cambridge, 2001.
- [190] Egeland, O. and Araldsen, H., SESAM-69: A general purpose finite element method program, *Computers & Structures*, **4**, 41–68, 1974.
- [191] Ergatoudis, J., Irons, B. M., and Zienkiewicz, O. C., Curved, isoparametric, “quadrilateral” elements for finite element analysis, *Int. J. Solids Struc.*, **4**, 31–42, 1968.
- [192] Euler, L., *Theoria Motus Corporum Solidorum seu Rigidum ex Primis nostrae Cognitionis Principiis Stabilita et ad Omnis Motus, qui in hujusmodi Corpora Cadere Possunt*, 1765. Reprinted in *L. Euler Opera Omnia*, 3–4, 3–293, Natural Science Society, Berne, 1911—82.

F

- [193] Farhat, C. and Roux, F. X., A method of finite-element tearing and interconnecting and its parallel solution algorithm, *Int. J. Numer. Meth. Engrg.*, **40**, 1205–1227, 1991.
- [194] Farhat, C. and Roux, F. X., Implicit parallel processing in structural mechanics, *Comput. Mech. Advances*, **2**, No. 1, 1–124, 1994.
- [195] Farhat, C., Geuzaine, P. and Brown, G., Application of a three-field nonlinear fluid-structure formulation to the prediction of the aeroelastic parameters of an F-16 fighter, *Computers and Fluids*, **32**, 3–29, 2003.

- [196] Farlow, J., Hall, J. E., J. M. McDill, J. M., and West, B.H., *Differential Equations and Linear Algebra*, Prentice Hall, NJ, 2002.
- [197] Felippa, C. A., Refined finite element analysis of linear and nonlinear two-dimensional structures, *Ph.D. Dissertation*, Department of Civil Engineering, University of California at Berkeley, Berkeley, CA, 1966.
- [198] Felippa, C. A. and Clough R. W., The finite element method in solid mechanics, in *Numerical Solution of Field Problems in Continuum Physics*, ed. by G. Birkhoff and R. S. Varga, SIAM–AMS Proceedings II, American Mathematical Society, Providence, R.I., 210–252, 1969.
- [199] Felippa, C. A. An alphanumeric finite element mesh plotter, *Int. J. Numer. Meth. Engrg.*, **5**, 217–236, 1972.
- [200] Felippa, C. A., Finite element analysis of three-dimensional cable structures, in *Computational Methods in Nonlinear Mechanics*, ed. by J T. Oden *et al.*, The Texas Institute for Computational Mechanics, University of Texas, Austin, Texas, 311–324, 1974.
- [201] Felippa, C. A., Procedures for computer analysis of large nonlinear structural systems, in *Large Engineering Systems*, ed. by A. Wexler, Pergamon Press, London, 60–101, 1976.
- [202] Felippa, C. A., Error analysis of penalty function techniques for constraint definition in linear algebraic systems, *Int. J. Numer. Meth. Engrg.*, **11**, 709–728, 1977.
- [203] Felippa, C. A., Solution of equations with skyline–stored symmetric coefficient matrix, *Computers & Structures*, **5**, 13–25, 1975.
- [204] Felippa, C. A., Iterative procedures for improving penalty function solutions of algebraic systems, *Int. J. Numer. Meth. Engrg.*, **12**, 821–836, 1978.
- [205] Felippa, C. A. and Park, K. C., Computational aspects of time integration procedures in structural dynamics – I. Implementation, *J. Appl. Mech.*, **45**, 595–602, 1978.
- [206] Felippa, C. A., Dynamic relaxation and Quasi-Newton methods, in *Numerical Methods for Nonlinear Problems 2*, ed. by C. Taylor, E. Hinton, D. J. R. Owen, E. Oñate, Pineridge Press, Swansea, U. K., 27–38, 1984.
- [207] Felippa, C. A. and J. A. DeRuntz, Finite element analysis of shock-induced hull cavitation, *Comp. Meths. Appl. Mech. Engrg.*, **44**, 297–337, 1984.
- [208] Felippa, C. A., Dynamic relaxation under general increment control, in *Innovative Methods for Nonlinear Problems*, ed. by W. K. Liu, T. Belytschko and K. C. Park, Pineridge Press, Swansea, U.K., 103–133, 1984.
- [209] Felippa, C. A., Traversing critical points by penalty springs, Proceedings of NUMETA'87 Conference, Swansea, Wales, M. Nijhoff Pubs, Dordrecht, Holland, 1987.
- [210] Felippa, C. A. and Bergan, P. G., A triangular plate bending element based on an energy-orthogonal free formulation, *Comp. Meths. Appl. Mech. Engrg.*, **61**, 129–160, 1987.
- [211] Felippa, C. A., Penalty spring stabilization of singular Jacobians, *J. Appl. Mech.*, **54**, 730–733, 1987.
- [212] Felippa, C. A. and Geers, T. L., Partitioned analysis of coupled mechanical systems, *Engrg. Comput.*, **5**, 123–133, 1988.
- [213] Felippa, C. A. and Militello, C., Developments in variational methods for high performance plate and shell elements, in *Analytical and Computational Models for Shells*, CED Vol. 3, ed. by A. K. Noor, T. Belytschko and J. C. Simo, The American Society of Mechanical Engineers, ASME, New York, 191–216, 1989.

Appendix R: REFERENCES (IN PROGRESS)

- [214] Felippa, C. A., The extended free formulation of finite elements in linear elasticity, *J. Appl. Mech.*, **56**, 609–616, 1989.
- [215] Felippa, C. A., Parametrized multifield variational principles in elasticity: II. Hybrid functionals and the free formulation, *Comm. Appl. Numer. Meth.*, **5**, 79–88, 1989.
- [216] Felippa, C. A., Programming the isoparametric six-node triangle, *Engineering Comp.*, **7**, 173–177, 1990.
- [217] Felippa, C. A. and Crivelli, L. A., The core-congruential formulation of geometrically nonlinear TL finite elements, Chapter IV.4 in *Nonlinear Computational Mechanics — The State of the Art*, ed. by P. Wriggers and W. Wagner, Springer-Verlag, Berlin, 283–282, 1991.
- [218] Felippa, C. A., Parametrized variational principles encompassing compressible and incompressible elasticity, *Int. J. Solids Struc.*, **29**, 57–68, 1991.
- [219] Felippa, C. A., Militello, C., Membrane triangles with corner drilling freedoms: II. The ANDES element, *Finite Elem. Anal. Des.*, **12**, 189–201, 1992.
- [220] Felippa, C. A. and Alexander, S., Membrane triangles with corner drilling freedoms: III. Implementation and performance evaluation, *Finite Elem. Anal. Des.*, **12**, 203–239, 1992.
- [221] Felippa, C. A., Crivelli, L. A., and Haugen, B., A survey of the core-congruential formulation for geometrically nonlinear TL finite elements, *Archives of Computational Methods in Engineering*, **1**, 1–48, 1994.
- [222] Felippa, C. A., A survey of parametrized variational principles and applications to computational mechanics, *Comp. Meths. Appl. Mech. Engrg.*, **113**, 109–139, 1994.
- [223] Felippa, C. A., Haugen, B. and Militello, C., From the individual element test to finite element templates: evolution of the patch test. *Int. J. Numer. Meth. Engrg.*, **38**, 199–229, 1995.
- [224] Felippa, C. A., Parametrized unification of matrix structural analysis: classical formulation and d-connected mixed elements, *Finite Elem. Anal. Des.*, **21**, 45–74, 1995.
- [225] Felippa, C. A., Recent developments in parametrized variational principles for mechanics, *Comput. Mech.*, **18**, 159–174, 1996.
- [226] Felippa, C. A., Park, K. C., and Justino Filho, M. R., The construction of free-free flexibility matrices as generalized stiffness inverses, *Computers & Structures*, **88**, 411–418, 1997.
- [227] Felippa, C. A. and Park, K. C., A direct flexibility method, *Comp. Meths. Appl. Mech. Engrg.*, **149**, 319–337, 1997.
- [228] Felippa, C. A., Recent advances in finite element templates, Chapter 4 in *Computational Mechanics for the Twenty-First Century*, ed. by B. H. V. Topping, Saxe-Coburn Pubs., Edinburgh, 71–98, 2000.
- [229] Felippa, C. A., A systematic approach to the Element-Independent Corotational dynamics of finite elements. Report CU-CAS-00-03, Center for Aerospace Structures, College of Engineering and Applied Sciences, University of Colorado at Boulder, January 2000.
- [230] Felippa, C. A., Recent advances in finite element templates, Chapter 4 in *Computational Mechanics for the Twenty-First Century*, ed. by B.H.V. Topping, Saxe-Coburn Publications, Edinburgh, 71–98, 2000.
- [231] Felippa, C. A., Customizing high performance elements by Fourier methods, *Trends in Computational Mechanics*, ed. by W. A. Wall et. al., CIMNE, Barcelona, Spain, 283–296, 2001.
- [232] Felippa, C. A., Customizing the mass and geometric stiffness of plane thin beam elements by Fourier methods, *Engrg. Comput.*, **18**, 286–303, 2001.
- [233] Felippa, C. A., A historical outline of matrix structural analysis: a play in three acts, *Computers & Structures*, **79**, 1313–1324, 2001.

- [234] Felippa, C. A., Park, K. C. and Farhat, C., Partitioned analysis of coupled mechanical systems, *Comp. Meths. Appl. Mech. Engrg.*, **190**, 3247–3270, 2001.
- [235] Felippa, C. A. and Park, K. C., The construction of free-free flexibility matrices for multilevel structural analysis, *Comp. Meths. Appl. Mech. Engrg.*, **191**, 2111–2140, 2002.
- [236] Felippa, C. A., A study of optimal membrane triangles with drilling freedoms, *Comp. Meths. Appl. Mech. Engrg.*, **192**, 2125–2168, 2003.
- [237] Felippa, C. A., A compendium of FEM integration rules for finite element work, *Engrg. Comput.*, **21**, 867–890, 2004.
- [238] Felippa, C. A., A template tutorial, Chapter 3 in *Computational Mechanics: Theory and Practice*, ed. by K. M. Mathisen, T. Kvamsdal and K. M. Okstad, CIMNE, Barcelona, 29–68, 2004.
- [239] Felippa, C. A., The amusing history of shear flexible beam elements, *IACM Expressions*, Issue 17, 15–19, 2005.
- [240] Felippa, C. A. and Oñate, E., Nodally exact Ritz discretizations of 1D diffusion-absorption and Helmholtz equations by variational FIC and modified equation methods, *Comput. Mech.*, 2006.
- [241] Felippa, C. A., and Oñate, E., Nodally exact Ritz discretizations of 1D diffusion-absorption and Hemholtz equations by variational FIC and modified equation methods. *Comput. Mech.*, **39**, 91–112, 2007.
- [242] Felippa, C. A., Oñate, E., and Idelsohn, S. R., FIC-based fluid-solid variational formulation encompassing incompressibility: I. Static Analysis, in preparation.
- [243] Felippa, C. A., Oñate, E., and Idelsohn, S. R., FIC-based fluid-solid variational formulation encompassing incompressibility: II. Spectral analysis, in preparation.
- [244] Felippa, C. A., Oñate, E., and Idelsohn, S. R., FIC-based fluid-solid variational formulation encompassing incompressibility: III. Time-domain dynamic analysis, in preparation.
- [245] Felippa, C. A., Web-posted Lectures on *Advanced Finite Element Methods*, at <http://caswww.colorado.edu/courses.d/AFEM.d/Home.html>, updated biyearly.
- [246] Felippa, C. A., Web-posted Lectures on *Advanced Finite Variational Methods in Mechanics*, at <http://caswww.colorado.edu/courses.d/AFEM.d/Home.html>, updated biyearly.
- [247] Felippa, C. A., Web-posted Lectures on *Introduction to Finite Element Methods*, at <http://caswww.colorado.edu/courses.d/IFEM.d/Home.html>, updated yearly.
- [248] Felippa, C. A., Web-posted Lectures on *Nonlinear Finite Element Methods*, at <http://caswww.colorado.edu/courses.d/NFEM.d/Home.html>, updated biyearly.
- [249] Ferrers, N. M., Extension of Lagrange’s equations, *Quart. J. Pure Appl. Math.*, **12**, 1–5, 1873.
- [250] Ficken, F., The continuation method for nonlinear functional equations, *Comm. Pure Appl. Math. Anal.*, **4**, 435–456, 1951.
- [251] Fill, J. A. and Fishkind, D. E., The Moore-Penrose generalized inverse for sum of matrices, *SIAM J. Matrix Anal. Appl.*, **21**, 629–635, 1999.
- [252] Finlayson, B. A. and Scriven, L. E., The method of weighted residuals — a review, *Appl. Mech. Rev.*, **19**, 735–748, 1966.
- [253] Finlayson, B. A., *The Methods of Weighted Residuals and Variational Principles*, Academic Press, 1972.
- [254] Fjeld, S. A., A three-dimensional theory of elasticity, in *Finite Element Methods for Stress Analysis*, ed. by I. Holland and K. Bell, Tapir, Trondheim, Norway, 1969.

Appendix R: REFERENCES (IN PROGRESS)

- [255] Flaggs, D. L., Symbolic analysis of the finite element method in structural mechanics, *Ph. D. Dissertation*, Dept of Aeronautics and Astronautics, Stanford University, 1988.
- [256] Flanagan, D. P. and Belytschko, T., A uniform strain hexahedron and quadrilateral with orthogonal hourglass control, *Int. J. Numer. Meth. Engrg.*, **17**, 679–706, 1981.
- [257] Flanders, H., *Differential Forms, With Applications to the Physical Sciences*, Dover, 1989.
- [258] Fletcher, C. A. J., *Computational Galerkin Methods*, Springer-Verlag, Berlin, 1984.
- [259] Fletcher, R., *Practical Methods of Optimization*, 2nd ed., Wiley, New York, 1987.
- [260] Flügge, W., *Stresses in Shells*, 2nd printing, Springer-Verlag, Berlin, 1973.
- [261] Forsyth, A. R., *Calculus of Variations*, Dover, 1960.
- [262] Fourier, J., *Theorie Analytique de la Chaleur*, Chez Firmin Didot, Père et Fils, Paris, 1822.
- [263] Fox, C., *An Introduction to the Calculus of Variations*, Oxford, 1963; Dover reprint 1987.
- [264] Fox, R. L. and Schmit, L. A., Advances in the integrated approach to structural synthesis, AIAA/ASME Material Conference, 1964.
- [265] Fraeijs de Veubeke, B. M., Diffusion des inconnues hyperstatiques dans les voilures à longeron couplés, *Bull. Serv. Technique de L'Aéronautique No. 24*, Imprimerie Marcel Hayez, Bruxelles, 1951.
- [266] Fraeijs de Veubeke, B. M., Upper and lower bounds in matrix structural analysis, in *AGARDograph 72: Matrix Methods of Structural Analysis*, ed. by B. M. Fraeijs de Veubeke, Pergamon Press, New York, 174–265, 1964.
- [267] Fraeijs de Veubeke, B. M., Displacement and equilibrium models, in *Stress Analysis*, ed. by O. C. Zienkiewicz and G. Hollister, Wiley, London, 145–197, 1965; reprinted in *Int. J. Numer. Meth. Engrg.*, **52**, 287–342, 2001.
- [268] Fraeijs de Veubeke, B., A conforming finite element for plate bending, *Int. J. Solids Struc.*, **4**, 95–108, 1968.
- [269] Fraeijs de Veubeke, B. M., Matrix structural analysis: Lecture Notes for the International Research Seminar on the Theory and Application of Finite Element Methods, Calgary, Alberta, Canada, July-August 1973; reprinted in *B. M. Fraeijs de Veubeke Memorial Volume of Selected Papers*, ed. by M. Geradin, Sithoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 509–568, 1980.
- [270] Fraeijs de Veubeke, B. M., Stress function approach, *Proc. World Congr. on Finite Element Methods*, October 1975, Woodlands, England; reprinted in *B. M. Fraeijs de Veubeke Memorial Volume of Selected Papers*, ed. by M. Geradin, Sithoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 663–715, 1980.
- [271] Fraeijs de Veubeke, B. M. The dynamics of flexible bodies, *Int. J. Engrg. Sci.*, **14**, 895-913, 1976. Reprinted in *B. M. Fraeijs de Veubeke Memorial Volume of Selected Papers*, ed. by M. Geradin, Sithoff & Noordhoff, Alphen aan den Rijn, The Netherlands, 717–752, 1980.
- [272] Franca, L. P., Analysis and finite element approximation of compressible and incompressible linear isotropic elasticity based upon a variational principle, *Comp. Meths. Appl. Mech. Engrg.*, **76**, 259–273, 1989.
- [273] Frazer, R. A. and Duncan, W. J., *The Flutter of Airplane Wings*, Reports & Memoranda 1155, Aeronautical Research Committee, London, 1928.
- [274] Frazer, R. A., Duncan, W. J., and Collar, A. R., *Elementary Matrices, and some Applications to Dynamics and Differential Equations*, Cambridge Univ. Press, 1st ed. 1938, 7th (paperback) printing 1963.

- [275] Freudenstein, F. and Roth, B., Numerical solutions of systems of nonlinear equations, *J. Assoc. Comp. Mach.*, **10**, 550–556, 1963.
- [276] Fried, I., Orthogonal trajectory accession to the nonlinear equilibrium curve, *Comp. Meths. Appl. Mech. Engrg.*, **47**, 283–297, 1984.
- [277] Fried, I. and Malkus, D. S., Finite element mass lumping by numerical integration with no convergence rate loss, *Int. J. Solids Struc.*, **11**, 461–466, 1975.
- [278] Fung, Y. C., *Foundations of Solid Mechanics*, Prentice-Hall, 1965.

G

- [279] Gallaguer, R. H., Padlog, J., and Bijlard, P. P., Stress analysis of heated complex shapes, *J. Am. Rock. Soc.*, 700-707, 1962.
- [280] Gallaguer, R. H., *A Correlation Study of Methods of Matrix Structural Analysis*, Pergamon, Oxford, 1964.
- [281] Gallagher, R. H., *Perturbation Procedures in Nonlinear Finite Element Structural Analysis*, Lecture Notes in Mathematics No. 461, Springer-Verlag, New York, 75–89, 1975.
- [282] Gantmacher, F. R., *The Theory of Matrices*, 2 vols, Chelsea, New York, 1960.
- [283] Garbow, B. S., Boyle, J. M., Dongarra, J. J., and Moler, C. B., Matrix Eigensystem Routines - EISPACK Guide Extension, Lecture Notes in Computer Science Vol. 51, Springer-Verlag, New York, 1986.
- [284] Gear, C. W., *Numerical Initial Value Problems in Ordinary Differential Equations*, Prentice-Hall, Englewood Cliffs, N.J., 1971.
- [285] Geers, T. L., Residual potential and approximate methods for three-dimensional fluid-structure interaction, *J. Acoust. Soc. Am.*, **45**, 1505–1510, 1971.
- [286] Geers, T. L., Doubly asymptotic approximations for transient motions of general structures, *J. Acoust. Soc. Am.*, **45**, 1500–1508, 1980.
- [287] Geers, T. L. and Felippa, C. A., Doubly asymptotic approximations for vibration analysis of submerged structures, *J. Acoust. Soc. Am.*, **73**, 1152–1159, 1980.
- [288] Geers, T. L., Boundary element methods for transient response analysis, in: Chapter 4 of *Computational Methods for Transient Analysis*, ed. by T. Belytschko and T. J. R. Hughes, North-Holland, Amsterdam, 221–244, 1983.
- [289] Gelfand, I. M. and Fomin, S. V., *Calculus of Variations*, Prentice-Hall, Englewood Cliffs, NJ, 1963. Reprinted by Dover, 2000.
- [290] Georg, K., Numerical integration of the Davidenko equation, in *Numerical Solution of Nonlinear Equations*, ed. by E. L. Allgower, K. Glashoff and H.-O. Peitgen, Lecture Notes in Mathematics 878, Springer-Verlag, Berlin, 1981.
- [291] Georg, K., On tracing an implicitly defined curve by Quasi-Newton steps and calculating bifurcation by local perturbations, *SIAM J. Sci. Statist. Comput.*, **2**, 35–50, 1981.
- [292] Geradin, M., Hogge, M. and Idelsohn, S., Nonlinear structural analysis via Newton and Quasi-Newton methods, *Nuclear Engrg. Design*, **58**, 339–348, 1980.
- [293] Geradin, M., Hogge, M., and Idelsohn, S., Implicit finite element methods, Ch. 9 in *Computational Methods for Transient Analysis*, ed. by T. Belytschko and Hughes, T. J. R., North-Holland, Amsterdam, 1983.

Appendix R: REFERENCES (IN PROGRESS)

- [294] Geradin, M., Finite element approach to kinematic and dynamic analysis of mechanisms using Euler parameters, in *Numerical Methods for Nonlinear Problems II*, ed. by C. Taylor, E. Hinton and D. R. J. Owen, Pineridge Press, Swansea, 1984.
- [295] Geradin, M. and Rixen, D., *Mechanical Vibrations: Theory and Applications to Structural Dynamics*, Wiley, New York, 1997.
- [296] Gill, P. E., Murray, W., and Wright, M. H., *Practical Optimization*, Academic Press, London, 1981.
- [297] Glynn, J. and Gray, T. H., *The Beginner's Guide to Mathematica Version 4*, Cambridge Univ. Press, 1999.
- [298] Goldberg, J. E. and Richards, R. H., Analysis of nonlinear structures, *J. ASCE Struct. Div.*, **89**, 333–336, 1963.
- [299] Goldstein, H. *Classical Mechanics*. Addison-Wesley, 1st ed. 1950; 3rd ed. updated by C. Poole and J. Safko, 2001.
- [300] Goldstine, H. H., *A History of Numerical Analysis*, Springer-Verlag, New York, 1977.
- [301] Golub, G. H. and Van Loan, C. F., *Matrix Computations*, Johns Hopkins Univ. Press, 2nd ed., 1983.
- [302] Golubitsky, M. and Schaeffer, D. G., *Singularities and Groups in Bifurcation Theory*, Springer-Verlag, New York, 1985.
- [303] González, L. A., Park, K. C. and Felippa, C. A., Partitioned formulation of frictional contact problems using localized Lagrange multipliers, *Commun. Numer. Meth. Engrg.*, **22**, 319–333, 2006.
- [304] Gorman, D. J., *Vibration Analysis of Plates by the Superposition Method*, World Scientific Pub. Co, 1999.
- [305] Graff, K. F., *Wave Motion in Elastic Solids*, Dover, New York, 1991.
- [306] Green, A. E., On Reissner's theory of bending of elastic plates, *Quart. Appl. Math.*, **7**, 223–228, 1949.
- [307] Griffiths, D. F. and Mitchell, A. R., Nonconforming elements, in *The Mathematical Basis of Finite Element Methods*, ed. by D. F. Griffiths, Clarendon Press, Oxford, 41–70, 1984.
- [308] Griffiths, D. and Sanz-Serna, J., On the scope of the method of modified equations. *SIAM J. Sci. Statist. Comput.*, **7**, 994–1008, 1986.
- [309] Guggenheimer, H. W., *Differential Geometry*, Dover, 1977..
- [310] Gurtin, M., The Linear Theory of Elasticity, in *Encyclopedia of Physics* VIa, Vol II, ed. by C. Truesdell, Springer-Verlag, Berlin, 1–295, 1972; reprinted as *Mechanics of Solids* Vol II, Springer-Verlag, Berlin, 1984.
- [311] Guttman, R. J., Enlargement methods for computing the inverse matrix, *Ann. Math. Stat.*, **317**, 336–343, 1946.
- [312] Guyan, R. J., Reduction of stiffness and mass matrices, *AIAA J.*, **3**, 380, 1965.

H

- [313] Hageman, L. A. and Young, D. M., *Applied Iterative Methods*, Academic Press, New York, 1981.
- [314] Hager, W. W., Updating the inverse of a matrix, *SIAM Review*, **31**, 221–239, 1989.
- [315] Hairer, E., Nørsett, S. P., and Wanner, G., *Solving Ordinary Differential Equations I: NonStiff Problems*, Springer-Verlag, Berlin, 2nd ed., 1993.
- [316] Hairer, E., Backward analysis of numerical integrators and symplectic methods, *Annals Numer. Math.*, **1**, 107–132, 1994.

- [317] Hairer, E., and Wanner, G., *Analysis by Its History*, Springer-Verlag, New York, 1996.
- [318] Hairer, E., Wanner, G., and Lubich, C., *Geometrical Numeric Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, Springer-Verlag, Berlin, 2002.
- [319] Haisler, W. E., Stricklin, J. H., and Key, J. E., Displacement incrementation in nonlinear structural analysis by the self-correcting method, *Int. J. Numer. Meth. Engrg.*, **11**, 3–10, 1977.
- [320] Hammer, P. C. and Stroud, A. H., Numerical integration over simplices, *Math. Tables Aids Comput.*, **10**, 137–139, 1956.
- [321] Hammermesh, M., *Group Theory and Its Application to Physical Problems*, Dover, New York, 1989.
- [322] Hammer, P. C. and Stroud, A. H., Numerical evaluation of multiple integrals, *Math. Tables Aids Comput.*, **12**, 272–280, 1958.
- [323] Hamming, R. W., *Digital Filters*, Dover, New York, 3rd ed., 1998.
- [324] Hamming, R. W., *Numerical Methods for Scientists and Engineers*, Dover, New York, 2nd ed., 1986.
- [325] Hammond, P., *Energy Methods in Electromagnetics*, Clarendon Press, Oxford, 1981.
- [326] Hanssen, L., Bergan, P. G., and Syversten, T. J., Stiffness derivation based on element convergence requirements, in *The Mathematics of Finite Elements and Applications – Volume III*, ed. by J. R. Whiteman, Academic Press, London, 83–96, 1979.
- [327] Hanson, A. J., *Visualizing Quaternions*, Elsevier, Oxford, 2006.
- [328] Hardy, G. H., *Divergent Series*, American Mathematical Society, Providence, 1991 (reprint of 1949 Oxford edition).
- [329] Hartung, R. F. (ed.), *Numerical Solution of Nonlinear Structural Problems*, ASME/AMD Vol. 6, ASME, New York, 1973.
- [330] Haselgrove, C. B., The solution of nonlinear equations and of differential equations with two-point boundary conditions, *Computer J.*, **4**, 255–259, 1961.
- [331] Haugen, B., Buckling and stability problems for thin shell structures using high-performance finite elements, *Ph. D. Dissertation*, Dept. of Aerospace Engineering Sciences, University of Colorado, Boulder, CO, 1994.
- [332] Haugen, B. and Felippa, C. A., A unified formulation of small-strain corotational finite elements: II. Applications to shells and mechanisms, in preparation.
- [333] Havner, K. S., The theory of finite plastic deformations in chrySTALLINE solids, in *Mechanics of Solids — The Rodney Hill 60th Anniversary Volume*, ed. by H. G. Hopkins and M. J. Sewell, Pergamon Press, Oxford, 265–302, 1982.
- [334] Haynsworth, E. V., On the Schur complement, *Basel Mathematical Notes*, #BMN20, **20**, 17pp, 1968.
- [335] Haynsworth, E. V., Determination of the inertia of a partitioned Hermitian matrix, *Lin. Alg. Appl.*, **1**, 73–81, 1968.
- [336] Haynsworth, E. V. and Ostrowski, A., On the inertia of some classes of partitioned matrices, *Lin. Alg. Appl.*, **1**, 299–301, 1968.
- [337] Hellinger, E., Die allgemeine Ansätze der Mechanik der Kontinua, *Encyklopædia der Mathematische Wissenschaften*, Vol 4⁴, ed. by F. Klein and C. Müller, Teubner, Leipzig, 1914.
- [338] Henderson, H. V., and Searle S. R., On deriving the inverse of a sum of matrices, *SIAM Review*, **23**, 53–60, 1981.
- [339] Henrici, P., *Discrete Variable Methods for Ordinary Differential Equations*, Wiley, New York, 1962.

Appendix R: REFERENCES (IN PROGRESS)

- [340] Henrici, P., *Error Propagation for Difference Methods*, Wiley, New York, 1963.
- [341] Henrici, P., *Applied and Computational Complex Analysis*, Vol II, Wiley, 1977.
- [342] Herrmann, L. R., Elasticity equations for nearly incompressible materials by a variational theorem, *AIAA J.*, **3**, 1896–1900, 1965.
- [343] Herrmann, L. R., A bending analysis for plates, in *Proceedings 1st Conference on Matrix Methods in Structural Mechanics*, ed. by R. Bader et. al., AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 577–604, 1966.
- [344] Hertz, H. *Die Prinzipien der Mechanik in neuem Zusammenhange dargestellt*, Barth, Leipzig, 1984. Reprinted by Kessinger, 2009. English translation: *The Principles of Mechanics Presented in a New Form*, Dover, New York, 1956; Phoenix Edition, 2004.
- [345] Hestenes, M. R., Multiplier and gradient methods, *J. Opt. Theory Appl.*, **4**, 303–320, 1969.
- [346] Hetenyi, M., *Beams on Elastic Foundation: Theory with Applications in the Fields of Civil and Mechanical Engineering*, Univ. of Michigan Press, 1946; 8th printing, 1967.
- [347] Hibbitt, H. D., Marcal, P. V., and Rice, J. R., A finite element formulation for problems of large strain and large displacement, *Int. J. Solids Struc.*, **6**, 1069–1086, 1970.
- [348] Higham, N. J., *Functions of Matrices: Theory and Computation*, SIAM, Philadelphia, 2008.
- [349] Hildebrand, F. B., *Introduction to Numerical Analysis*, Dover, 2nd ed., 1987; 1st ed., McGraw-Hill, 1974.
- [350] Hill, R., On constitutive equations for simple materials, *J. Mech. Phys. Solids*, **15**, 229–242, 1968.
- [351] Hill, R., Aspects of invariance in solid mechanics, in *Advances in Applied Mechanics*, ed. by C. S. Yih, **18**, 1–75, 1978.
- [352] Hinton, E., Rock T. and Zienkiewicz, O. C., A note on mass lumping and related processes in the finite element method, *Earthquake Engrg. Struc. Dynamics*, **4**, 245–249, 1976.
- [353] Holand, I., Stiffness matrices for plate bending, in *Finite Element Methods in Stress Analysis*, ed. by I. Holand and K. Bell, Tapir, Trondheim, 1969.
- [354] Holmes, M. H., *Introduction to Perturbation Methods*, Springer-Verlag, New York, 1994.
- [355] Horn, R. A. and Johnson, C. R., *Matrix Analysis*, Cambridge University Press, 1991. Corrected reprint edition 1990.
- [356] Horn, R. A. and Johnson, C. R., *Topics in Matrix Analysis*, Cambridge University Press, 1991. Corrected reprint edition 1994.
- [357] Horrigmoe, Thesis.NTH.1977 Horrigmoe, G., Finite element instability analysis of free-form shells, *Dr. Ing. Thesis*, Div. of Structural Mechanics, NTH, Trondheim, Norway, 1977.
- [358] Horrigmoe, G. and Bergan, P. G., Instability analysis of free-form shells by flat finite elements, *Comp. Meths. Appl. Mech. Engrg.*, **16**, 11–35, 1978.
- [359] Householder, A. S., *The Theory of Matrices in Numerical Analysis*, Blaisdell, New York, 1964. Dover reprint 1975.
- [360] Hrabok, M. M. and Hrudey, T. M., A review and catalogue of plate bending finite elements, *Computers & Structures*, **19**, 479–498, 1984.
- [361] Hrenikoff, A., Solution of problems in elasticity by the framework method, *J. Appl. Mech.*, **8**, A169–A175, 1941.
- [362] Hu, H. *Variational Principles of Theory of Elasticity with Applications*, Gordon and Breach Science Publishers Inc, New York, 1984; available online as Google eBook.

- [363] Huang, N. C. and Nachbar, W., Dynamic snap-through of imperfect viscoelastic shallow arches, *J. Appl. Mech.*, **35**, 289–296, 1968.
- [364] Huang, H. C. and Hinton, E., A new nine node degenerated shell element with enhanced membrane and shear interpolation, *Int. J. Numer. Meth. Engrg.*, **22**, 73–92, 1986.
- [365] Huang, D. Y., Unified approach to quadratically convergent algorithms for function minimization, *J. Opt. Theory Appl.*, **5**, 405–423, 1970.
- [366] Hughes, T. J. R., Taylor, R., and Kanolkulchai, W., A simple and efficient finite element for plate bending, *Int. J. Numer. Meth. Engrg.*, **11**, 1529–1543, 1977.
- [367] Hughes, T. J. R. and Malkus, D. S., Mixed finite element methods – reduced and selective integration techniques: a unification of concepts, *Comp. Meths. Appl. Mech. Engrg.*, **15**, 63–81, 1978.
- [368] Hughes, T. J. R. and Cohen, M., The Heterosis finite element for plate bending, *Computers & Structures*, **9**, 445–450, 1980.
- [369] Hughes, T. J. R., Generalization of selective integration procedures to anisotropic and nonlinear media, *Int. J. Numer. Meth. Engrg.*, **15**, 1413–148, 1980.
- [370] Hughes, T. J. R. and Winget, J., Finite rotation effects in numerical integration of rate constitutive equations arising in large deformation analysis, *Int. J. Numer. Meth. Engrg.*, **15**, 1862–1867, 1980.
- [371] Hughes, T. J. R. and Liu, W.-K., Nonlinear finite element analysis of shells: Part I. Three-dimensional shells, *Comp. Meths. Appl. Mech. Engrg.*, **26**, 331–362, 1981.
- [372] Hughes, T. J. R. and Liu, W.-K., Nonlinear finite element analysis of shells: Part II. Two-dimensional shells, *Comp. Meths. Appl. Mech. Engrg.*, **27**, 167–182, 1981.
- [373] Hughes, T. J. R. and Malkus, D. S., A general penalty mixed equivalence theorem for anisotropic, incompressible finite elements, in *Hybrids and Mixed Finite Element Methods*, ed. by S. N. Atluri, R. H. Gallagher and O. C. Zienkiewicz, Wiley, London, 1983.
- [374] Hughes, T. J. R., *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice Hall, Englewood Cliffs, NJ, 1987; Dover reprint, 2000.
- [375] Hurwitz, A., Über die Bedingungen, unter welchen eine Gleichung nur Wurzeln mit negativen reellen Theilen besitzt, *Math. Ann.*, **46**, 273–284, 1895. English translation: On the conditions under which an equation has only roots with negative real part, in *Selected papers on Mathematical Trends in Control Theory*, ed. by R. Bellman and R. Kalaba, Dover, 72–82, 1964.
- [376] Hussein, R., *Composite Panels and Plates: Analysis and Design*, Technomic Pub. Co., Lancaster, PA, 1986

I

- [377] Idelsohn, S. R., Oñate, E., and Del Pin, F., The Particle Finite Element Method: A powerful tool to solve incompressible flows with free surfaces and breaking waves, *Int. J. Numer. Meth. Engrg.*, **61**, 964–989, 2004.
- [378] Idelsohn, S. R., Del Pin, F., Rossi, R., and Oñate, E., Fluid-structure interaction problems with strong added-mass effect, *Int. J. Numer. Meth. Engrg.*, **10**, 1261–1294, 2009.
- [379] Irons, B. M. and Barlow, J., Comments on ‘matrices for the direct stiffness method’ by R. J. Melosh, *AIAA J.*, **2**, 403, 1964.
- [380] Irons, B. M. and Draper, K., Inadequacy of nodal connections in a stiffness solution for plate bending, *AIAA J.*, **3**, 965–966, 1965.

Appendix R: REFERENCES (IN PROGRESS)

- [381] Irons, B. M., Engineering application of numerical integration in stiffness methods, *AIAA J.*, **4**, 2035–2037, 1966.
- [382] Irons, B. M., A frontal solution program for finite element analysis, *Int. J. Numer. Meth. Engrg.*, **12**, 5–32, 1970.
- [383] Irons, B. M. and A. Razzaque, A., Experiences with the patch test for convergence of finite elements, in *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, ed. by A. K. Aziz, Academic Press, New York, 557–587, 1972.
- [384] Irons, B. M. and Ahmad, S., *Techniques of Finite Elements*, Ellis Horwood Ltd, Chichester, UK, 1980.
- [385] Irons, B. M., Putative high-performance plate bending element, Letter to Editor, *Int. J. Numer. Meth. Engrg.*, **19**, 310, 1983.
- [386] Irons, B. M. and Loikannen, M., An engineer's defense of the patch test, *Int. J. Numer. Meth. Engrg.*, **19**, 1391–1401, 1983.
- [387] Irvine, M., *Cable Structures*, Dover, 1992. Reprint of 1st ed., MIT Press, 1981.

J

- [388] Jensen, P. S., Transient analysis of structures by stiffly stable methods, *Computers & Structures*, **4**, 615–626, 1974.
- [389] Jones, H., *The Theory of Brillouin Zones and Electronic States in Crystals*, North Holland, Amsterdam, 1960.
- [390] Jury, E. I., *Theory and Applications of the Z-Transform Method*, Wiley, 1964.
- [391] Jury, E. I., *Inners and Stability of Dynamic Systems*, 2nd ed., Krieger, 1982.

K

- [392] Kaneko, I., Lawo, M. and Thierauf, G., On computational procedures for the force method, *Int. J. Numer. Meth. Engrg.*, **18**, 1469–1495, 1982.
- [393] Kaneko, I. and Plemmons, R. J., Minimum norm solutions to linear elastic analysis problems, *Int. J. Numer. Meth. Engrg.*, **20**, 983–998, 1984.
- [394] Kantorovich, L. V. and Krylov, V. I., *Approximate Methods of Higher Analysis*, Noordhoff, Groningen, 1958.
- [395] Kavanagh, K. and Key, S. W., A note on selective and reduced integration techniques in the finite element method, *Int. J. Numer. Meth. Engrg.*, **4**, 148–150, 1972.
- [396] Keller, H. B., Nonlinear bifurcation, *J. Diff. Eqs.*, **7**, 417–434, 1970.
- [397] Keller, H. B., Numerical solution of bifurcation and nonlinear eigenvalue problems, in *Application of Bifurcation Theory*, ed. by P. H. Rabinowitz, Academic Press, New York, 359–384, 1977.
- [398] Keller, H. B., Global homotopies and Newton methods, in *Recent Advances in Numerical Analysis*, ed. by C. de Boor and G. H. Golub, Academic Press, New York, 1978.
- [399] Keller, H. B., Geometrically isolated nonisolated solutions and their approximation, *SIAM J. Numer. Anal.*, **18**, 822–838, 1981.
- [400] Kikuchi, N. and Oden, J. T., Contact problems in elastostatics, in *Finite Elements: Special Problems in Solid Mechanics*, ed. by Oden, J. T. and G. Carey, Prentice-Hall, Englewood Cliffs, N. J., 1984.
- [401] Kikuchi, N. and Oden, J. T., *Contact Problems in Elasticity: A Study of Variational Inequalities and Finite Element Methods*, SIAM Studies in Applied Mathematics, SIAM, Philadelphia, 1988.

- [402] Kirchhoff, G., Über das Gleichgewicht und die Bewegung einer elastischen Scheibe, *Crelles J.*, **40**, 51–88, 1850.
- [403] Kirchhoff, G., *Vorlesungen über Mathematischen Physik. Mechanik*, Leipzig, 1876.
- [404] Klopffestein, R. W., Zeros of nonlinear functions, *J. Assoc. Comp. Mach.*, **8**, 366–373, 1961.
- [405] Knops, R. J. and Wilkes, E. W., Theory of elastic stability, in *Encyclopedia of Physics* Vol VIa/1-4, ed. by S. Flugge, Springer-Verlag, Berlin, 1973.
- [406] Knuth, D. E., *The T_EXbook*, Addison-Wesley, Reading, MA, 1984.
- [407] Kråkeland, B., Large displacement analysis of shells considering elastoplastic and elastoviscoplastic materials, *Dr. Ing. Thesis*, Div. of Structural Mechanics, NTH, Trondheim, Norway, 1977.
- [408] Krieg, R. D. and Key, S. W., Transient shell analysis by numerical time integration, in *Advances in Computational Methods for Structural Mechanics and Design*, ed. by J. T. Oden, R. W. Clough and Y. Yamamoto, UAH Press, Huntsville, Alabama, 237–258, 1972.
- [409] G. Kron, Tensorial analysis and equivalent circuits of elastic structures, *J. Franklin Inst.*, **238**, 399–442, 1944.
- [410] Kröplin, B., Dinkler, D., and Hillmann, J., Global constraints in nonlinear solution strategies, in *Finite Element Methods for Nonlinear Problems*, ed. by P. G. Bergan, K. J. Bathe, and W. Wunderlich, Springer, Berlin, 1986.
- [411] Kubíček, M. and Hlaváček, V., *Numerical Solution of Nonlinear Boundary Value Problems with Applications*, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.
- [412] Kubíček, M., and Marek, M., *Computational Methods in Bifurcation Theory and Dissipative Structures*, Springer-Verlag, New York, 1983.
- [413] Küpper, T., Mittelman, H. D., and Weber, H. (eds.) *Numerical Methods for Bifurcation Problems*, Birhäuser Verlag, Basel, 1984.
- [414] Küpper, T., Seydel, R., and Troger, H. (eds.) *Bifurcation: Analysis, Algorithms, Applications*, Birhäuser Verlag, Basel, 1987.
- [415] Kuztesov, E. N., *Unconstrained Structure Systems*, Springer-Verlag, New York, 1991.

L

- [416] Lagrange, J. L., *Mécanique Analytique*, Paris, 1788. Reprinted by J. Gabay, Paris, 1989; downloadable as Google eBook.
- [417] Lambert, J. D., *Computational Methods in Ordinary Differential Equations*, Wiley, London, 1973.
- [418] Lancaster, P. and Tismenetsky M., *The Theory of Matrices*, Academic Press, Orlando, 2nd ed., 1985.
- [419] Lanczos, C., *The Variational Principles of Mechanics*, Dover, 4th edition, 1970. (First edition 1949).
- [420] Langhaar, H. L., *Energy Methods in Applied Mechanics*, Wiley, New York, 1962.
- [421] Lapidus, L. and Seinfeld, J. H., *Numerical Solution of Ordinary Differential Equations*, Academic Press, New York, 1971.
- [422] Lautersztajn-S, N. and Samuelsson, A., Further discussion on four-node isoparametric elements in plane bending, *Int. J. Numer. Meth. Engrg.*, **47**, 129–140, 2000.
- [423] Lesne, A., Discrete versus continuous controversy in physics, *Math. Struct. Comp. Sci.*, **17**, 185–223, 2007.

Appendix R: REFERENCES (IN PROGRESS)

- [424] Letnitskii, S. G., *Theory of Elasticity of an Anisotropic Elastic Body*, Russian translation, Holden-Day, 1963.
- [425] Letnitskii, S. G., *Anisotropic Plates*, Russian translation, Gordon and Breach, 1968.
- [426] Levold, E., Solid mechanics and material models including large deformations, *Dr. Ing. Thesis*, Div. of Structural Mechanics, NTH, Trondheim, Norway, 1990.
- [427] Levy, S., Computation of influence coefficients for aircraft structures with discontinuities and sweepback, *J. Aero. Sci.*, **14**, 547–560, 1947.
- [428] Levy, S., Structural analysis and influence coefficient for delta wings, *J. Aero. Sci.*, **20**, 449–454, 1953.
- [429] Liew, K. M., et al. (eds.), *Vibration of Mindlin Plates: Programming the P-Version Ritz Method*, Elsevier Science Ltd, 1998.
- [430] Ling, F. F., (ed.), *Vibrations of Elastic Plates : Linear and Nonlinear Dynamical Modeling of Sandwiches, Laminated Composites, and Piezoelectric Layers*, Springer-Verlag, New York, 1995.
- [431] Livesley, R. K., *Matrix Methods of Structural Analysis*, Pergamon Press, London, 1964. Reprinted by Dover.
- [432] Loikannen, M. J. and Irons, B. M., An 8-node brick finite element, *Int. J. Numer. Meth. Engrg.*, **20**, 523–528, 1984.
- [433] Love, A. E. H., *Theory of Elasticity*, Cambridge, 4th ed 1927, Dover reprint, 1944.
- [434] Lowe, P. G., *Basic Principles of Plate Theory*, Surrey Univ. Press, Blackie Group, London, 1982.
- [435] Lützen, J., *Mechanistic Images in Geometric Form: Heinrich Hertz's Principles of Mechanics*, Oxford Univ. Press, 2005.

M

- [436] MacNeal, R. H. (ed.), *The NASTRAN Theoretical Manual*, NASA SP-221, 1970.
- [437] MacNeal, R. H. and McCormick, C. W., The NASTRAN computer program for structural analysis, *Computers & Structures*, **1**, 389–412, 1971.
- [438] MacNeal, R. H., Derivation of element stiffness matrices by assumed strain distribution, *Nuclear Engrg. Design*, **70**, 3–12, 1978.
- [439] MacNeal, R. H., A simple quadrilateral shell element, *Computers & Structures*, **8**, 175–183, 1978.
- [440] MacNeal, R. H., The evolution of lower order plate and shell elements in MSC/NASTRAN, in *Finite Element Methods for Plate and Shell Structures, Vol. I: Element Technology*, ed. by T. J. R. Hughes and E. Hinton, Pineridge Press, Swansea, U.K., 85–127, 1986.
- [441] MacNeal, R. H. and Harder, R. L., A proposed standard set of problems to test finite element accuracy, *Finite Elem. Anal. Des.*, **1**, 3–20, 1985.
- [442] MacNeal, R. H., A theorem regarding the locking of tapered four noded membrane elements, *Int. J. Numer. Meth. Engrg.*, **24**, 1793–1799, 1987.
- [443] MacNeal, R. H., On the limits of finite element perfectibility, *Int. J. Numer. Meth. Engrg.*, **35**, 1589–1601, 1992.
- [444] MacNeal, R. H., *Finite Elements: Their Design and Performance*, Marcel Dekker, New York, 1994.
- [445] Mallet, R. H. and Marcal, P. V., Finite element analysis of nonlinear structures, *J. ASCE Struct. Div.*, **94**, ST9, 2081–2105, 1968.

- [446] Malkus, D. S. and Plesha, M. E., Zero and negative masses in finite element vibration and transient analysis, *Comp. Meths. Appl. Mech. Engrg.*, **59**, 281–306, 1986.
- [447] Malkus, D. S., Plesha, M. E., and Liu, M. R., Reversed stability conditions in transient finite element analysis, *Comp. Meths. Appl. Mech. Engrg.*, **68**, 97–114, 1988.
- [448] Marcal, P. V., A stiffness method for elastic-plastic problems, *Int. J. Mech. Sci.*, 229–238, 1965.
- [449] Marcal, P. V., Finite element analysis with material nonlinearities – theory and practice, in *Recent Advances in Matrix Methods in Structural Analysis and Design*, ed. by R. H. Gallagher, Y. Yamada and J. T. Oden, University of Alabama Press, Huntsville, Ala., 1970.
- [450] Martin, H. C., On the derivation of stiffness matrices for the analysis of large deflection and stability problems, in *Proc. 1st Conf. on Matrix Methods in Structural Mechanics*, ed. by J. S. Przemieniecki et al, AFFDL-TR-66-80, Air Force Institute of Technology, 697–716, 1966.
- [451] Martin, H. C., *Introduction to Matrix Methods of Structural Analysis*, McGraw-Hill, New York, 1966.
- [452] Martin, H. C., Large deflection and stability analysis by the direct stiffness method, in *Recent Advances in Matrix Methods in Structural Analysis and Design*, ed. by R. H. Gallagher, Y. Yamada and J. T. Oden, University of Alabama Press, Huntsville, Ala., 1970.
- [453] Martin, H. C. and Carey, G. F., *Introduction to Finite Element Analysis*, McGraw-Hill, New York, 1973.
- [454] Mathisen, K. M., Large displacement analysis of flexible and rigid systems considering displacement-dependent loads and nonlinear constraints. *Dr. Ing. Thesis*, Div. of Structural Mechanics, NTH, Trondheim, Norway, 1990.
- [455] Mathisen, K. M., Kvamsdal T., and Okstad, K. M., Adaptive strategies for nonlinear finite element analysis of shell structures, In: *Numerical Methods in Engineering '92*, C. Hirsch et al. (eds.), Elsevier Science Publishers B. V., 1992.
- [456] Mattiasson, K., On the corotational finite element formulation for large deformation problems, *Dr. Ing. Thesis*, Dept. of Structural Mechanics, Chalmers University of Technology, Göteborg, 1983.
- [457] Mattiasson, K. and Samuelsson, A., Total and updated Lagrangian forms of the co-rotational finite element formulation in geometrically and materially nonlinear analysis, in *Numerical Methods for Nonlinear Problems II*, ed. by C. Taylor, E. Hinton and D. R. J. Owen, Pineridge Press, Swansea, 134–151, 1984.
- [458] Mattiasson, K., Bengtson, A., and Samuelsson, A., On the accuracy and efficiency of numerical algorithms for geometrically nonlinear structural analysis, in *Finite Element Methods for Nonlinear Problems*, ed. by P. G. Bergan, K. J. Bathe and W. Wunderlich, Springer-Verlag, 3–23, 1986.
- [459] Matthies, H. and Strang, G., The solution of nonlinear finite element equations, *Int. J. Numer. Meth. Engrg.*, **14**, 1613–1626, 1979.
- [460] McHenry, D., A lattice analogy for the solution of plane stress problems, *J. Inst. Civ. Engrs.*, **21**, 59–82, 1943.
- [461] Meek, J. L. and Tan, H. S., Geometrically nonlinear analysis of space frames by an incremental-iterative technique, *Comp. Meths. Appl. Mech. Engrg.*, **47**, 261–282, 1984.
- [462] Meirovitch, L., *Methods of Analytical Dynamics*, McGraw-Hill, New York, 1970.
- [463] Meirovitch, L., *Computational Methods in Structural Dynamics*, Kluwer Acad. Pubs, 1980.
- [464] Melhem, R. G. and Rheinboldt, W. C., A comparison of methods for determining turning points of nonlinear equations, *Computing J.*, **29**, 201–226, 1982.
- [465] Melosh, R. J. and Merritt, R. G., Evaluation of spar matrices for stiffness analysis, *J. Aero. Sci.*, **25**, 537–543, 1959.

Appendix R: REFERENCES (IN PROGRESS)

- [466] Melosh, R. J., A stiffness matrix for the analysis of thin plates in bending, *J. Aero. Sci.*, **28**, 34–40, 1961.
- [467] Melosh, R. J., Development of the stiffness method to define bounds on the elastic behavior of structures, *Ph.D. Dissertation*, University of Washington, Seattle, 1962.
- [468] Melosh, R. J., Bases for the derivation of matrices for the direct stiffness method, *AIAA J.*, **1**, 1631–1637, 1963.
- [469] Melosh, R. J., Structural analysis of solids, *J. ASCE Struct. Div.*, **ST4-89**, 205–223, 1963.
- [470] Melosh, R. J., A flat triangular shell element stiffness matrix, Proc. Conf. on Matrix Methods in Structural Mechanics, WPAFB, Ohio, 1965, in *AFFDL TR 66-80*, 503–509, 1966.
- [471] Meyer, G., On solving nonlinear equations with a one-parameter operator embedding, *SIAM J. Numer. Anal.*, **5**, 739–752, 1968.
- [472] Mikhlin, S. G., *Variational Methods in Mathematical Physics*, Pergamon Press and Macmillan, New York, 1964.
- [473] Mikhlin, S. G., *The Problem of the Minimum of a Quadratic Functional*, PHolden-Day, San Francisco, 1965.
- [474] Militello, C. and Felippa, C. A., C. A., The individual element patch revisited, in *The Finite Element Method in the 1990's — a book dedicated to O. C. Zienkiewicz*, ed. by E. Oñate, J. Periaux and A. Samuelsson, CIMNE, Barcelona and Springer-Verlag, Berlin, 554–564, 1990.
- [475] Militello, C. and Felippa, C. A., The first ANDES elements: 9-DOF plate bending triangles, *Comp. Meths. Appl. Mech. Engrg.*, **93**, 217–246, 1991.
- [476] Mindlin, R. D., Influence of rotary inertia and shear on flexural vibrations of isotropic, elastic plates, *J. Appl. Mech.*, **18**, 31–38, 1951.
- [477] Misner, C. W., , Thorne, K., and Wheeler, J. A., *Gravitation*, W. H. Freeman, San Francisco, 1973.
- [478] Möbius, A.F., *Der Barycentrische Calcul*, Georg Olms, Hildesheim, Germany, 1976. Original edition: Leipzig, Germany, 1827.
- [479] Moore, G., The numerical treatment of non-trivial bifurcation points, *Numer. Funct. Anal. Opt.*, **17** 567–576, 1980.
- [480] Moore, G. and Spence, A., The calculation of turning points of nonlinear equations, *SIAM J. Numer. Anal.*, **17** 567–576, 1980.
- [481] Morley, L. S. D., *Skew Plates and Structures*, Pergamon Press, 1963.
- [482] Morley, L. S. D., The constant-moment plate bending element, *J. Strain Analysis*, **6**, 20–24, 1971.
- [483] Morse, P. M. and Feshbach, H., *Methods of Theoretical Physics*, 2 vols, McGraw-Hill, 1953.
- [484] Muir, T., (Sir), *Theory of Determinants*, Dover, New York, 1960.
- [485] Murray, D. W. and Wilson, E. L., Finite element large-deflection analysis of plates, *J. ASCE Mech. Div.*, **95**, EM5, 143–165, 1969.
- [486] Murray, D. W. and Wilson, E. L., Finite element analysis of nonlinear structures, *AIAA J.*, **7**, 1915–1920, 1969.

N

- [487] Nadukandi, P., Stabilized finite element methods for convection-diffusion-reaction, Helmholtz and Navier-Stokes problems, Thesis, Universidad Politécnica de Catalunya, 2011.

- [488] Nayfeh, A. H., *Perturbation Methods*, Wiley, New York, 1973.
- [489] Newton, I., *Philosophiae Naturalis Principia Mathematica*, (*Mathematical Principles of Natural Philosophy*, also known as the *Principia*), 1st.ed., London, 1687; 2nd.ed., Cambridge, 1712; 3rd.ed., London, 1726, final ed. (prepared by Newton, in English translation by A. Motte), London, 1729.
- [490] Noor, A. K. and J. M. Peters, Reduced basis technique for nonlinear analysis of structures, *AIAA J.*, **18**, 455–462, 1980.
- [491] Noor, A. K., Recent advances in reduction methods for nonlinear problems, *Computers & Structures*, **13**, 31–44, 1983.
- [492] Nour-Omid, B. and Wriggers, P., Solution methods for contact problems, *Comp. Meths. Appl. Mech. Engrg.*, **52**, 1986.
- [493] Nour-Omid, B. and Rankin, C. C., Finite rotation analysis and consistent linearization using projectors, *Comp. Meths. Appl. Mech. Engrg.*, **93**, 353–384, 1991.
- [494] Novozhilov, V. V., *Foundations of the Nonlinear Theory of Elasticity*, Graylock Press, Rochester, 1953.
- [495] Nygård, M. K., The Free Formulation for nonlinear finite elements with applications to shells, *Dr. Ing. Thesis*, Division of Structural Mechanics, NTH, Trondheim, Norway, 1986.
- [496] Nygård, M. K. and Bergan, P. G., Advances in treating large rotations for nonlinear problems, Chapter 10 in *State-of-the Art Surveys on Computational Mechanics*, ed. by A. K. Noor and J. T. Oden, ASME, New York, 305–332, 1989.

O

- [497] Oden, J. T., Calculation of geometric stiffness matrices for thin Shells of arbitrary shape, *AIAA J.*, **4**, 1480–1482, 1966.
- [498] Oden, J. T., Numerical formulation of nonlinear elasticity problems, *J. ASCE Struct. Div.*, **93**, 235–255, 1967.
- [499] Oden, J. T., *Finite Elements of Nonlinear Continua*, McGraw-Hill, New York, 1972.
- [500] Oden, J. T. and Key, S. W., Analysis of static nonlinear response by explicit time integration, *Int. J. Numer. Meth. Engrg.*, **7**, 225–240, 1973.
- [501] Oden, J. T. et al. (eds.), *Computational Methods in Nonlinear Mechanics*, The Texas Institute for Computational Mechanics (TICOM), University of Texas, Austin, Texas, 1974.
- [502] Oden, J. T., Exterior penalty methods for contact problems in elasticity, in *Nonlinear Finite Element Analysis in Structural Mechanics*, ed. by W. Wunderlich, E. Stein and K. J. Bathe, Springer, Berlin, 1981.
- [503] Oden, J. T. and Reddy, J. N., *Variational Methods in Theoretical Mechanics*, Springer-Verlag, Berlin, 1982.
- [504] Okuma, M. and Shi, Q., Identification of the principal rigid body modes under free-free boundary condition, *Trans. ASME J. of Vibration and Acoustics*, **119**(3), 341–345, 1997.
- [505] Oñate, E., Derivation of the stabilization equations for advective-diffusive fluid transport and fluid flow problems. *Comp. Meths. Appl. Mech. Engrg.*, **151**, 233–267, 1998.
- [506] Oñate, E., Possibilities of finite calculus in computational mechanics, *Int. J. Numer. Meth. Engrg.*, **60**, 255–281, 2004.
- [507] Oñate, E., Taylor, R. L., O. C. Zienkiewicz and J. Rojek, A residual correction method based on finite calculus, *Engrg. Comput.*, **20**, 629–638, 2003.

Appendix R: REFERENCES (IN PROGRESS)

- [508] Oñate, E., Rojek, J., Taylor, R. L., and Zienkiewicz, O. C., Finite calculus formulation for incompressible solids using linear triangles and tetrahedra, *Int. J. Numer. Meth. Engrg.*, **59**, 1473–1500, 2004.
- [509] Oñate, E., Miquel, J., and Hauke, G., A stabilized finite element method for the one-dimensional advection diffusion-absorption equation using finite calculus, *Comp. Meths. Appl. Mech. Engrg.*, **195**, 3926–3946, 2006.
- [510] E. Oñate, S. R. Idelsohn, and C. A. Felippa, Simple and accurate pressure Laplacian stabilization for incompressible continua via higher order Finite Calculus, submitted to *Int. J. Numer. Meth. Engrg.*, 2010.
- [511] Oñate, E., Nadukandi, P., Idelsohn, S. R., and Felippa, C. A., A family of residual-based stabilized finite element methods for Stokes flow, *Int. J. Numer. Meth. Engrg.*, **65**, 106–134, 2011.
- [512] Ortega, J. M. and Rheinboldt, W. C., *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.
- [513] Otter, J. R. H., Computations for prestressed concrete reactor pressure vessels using dynamic relaxation, *Nuclear Struct. Engrg.*, **1**, 61-75, 1965.
- [514] Ostrowski, A., A new proof of Haysnworth’s quotient formula for Schur complements, *Lin. Alg. Appl.*, **4**, 389–392, 1971.
- [515] Ostrowski, A., On Schur’s complement, *J. Combin. Theory Series A*, **14**, 319–323, 1973.
- [516] Özişik, M. N., *Boundary Value Problems of Heat Conduction*, Dover edition, 1989.

P

- [517] Pacoste, C., Co-rotational flat facet triangular elements for shell instability analysis, *Comp. Meths. Appl. Mech. Engrg.*, **156**, 75-110, 1998.
- [518] Padovan, J., Self-adaptive predictor-corrector algorithm for static nonlinear structural analysis, Report NASA CR-165410 to Lewis Research Center, The University of Akron, Akron, Ohio, 1981.
- [519] Padovan, J. and Tovichakchaikul, S., Self-Adaptive predictor-corrector algorithm for static nonlinear structural analysis, *Computers & Structures*, **15**, 365–377, 1982.
- [520] Panovko, Y. K. and Gubanov, I. I., *Stability and Oscillations of Physical Systems: Paradoxes, Fallacies and New Concepts*, Consultants Bureau, New York, 1965.
- [521] Papadrakakis, M., Post-buckling analysis of spatial structures by vector iteration methods, *Computers & Structures*, **12**, 393–402, 1981.
- [522] Papenfuss, S. W., Lateral plate deflection by stiffness methods and application to a marquee, M. S. Thesis, Department of Civil Engineering, University of Washington, Seattle, WA, 1959.
- [523] Parlett, B. N., *The Symmetric Eigenvalue Problem*, Prentice-Hall, 1980. Reprinted by SIAM Publications, 1998.
- [524] Park, K. C., Felippa, C. A., and DeRuntz, J. A., Stabilization of staggered solution procedures for fluid-structure interaction analysis, in *Computational Methods for Fluid-Structure Interaction Problems*, ed. by T. Belytschko and Geers, T. L., AMD Vol. 26, American Society of Mechanical Engineers, ASME, New York, 95–124, 1977.
- [525] Park, K. C., and Felippa, C. A., Computational aspects of time integration procedures in structural dynamics – II. Error Propagation, *J. Appl. Mech.*, **45**, 603–611, 1978.
- [526] Park, K. C. and Felippa, C. A., Direct time integration methods in nonlinear structural dynamics, *Comp. Meths. Appl. Mech. Engrg.*, **17/18**, pp. 277–313, 1979.

- [527] Park, K. C., Partitioned transient analysis procedures for coupled-field problems: stability analysis, *J. Appl. Mech.*, **47**, 370–376, 1980.
- [528] Park, K. C., Felippa, C. A. and Ohayon, R., Partitioned formulation of internal fluid-structure interaction problems via localized Lagrange multipliers, *Comp. Meths. Appl. Mech. Engrg.*, **190**, 2989–3007, 2001.
- [529] Park, K. C. and P. G. Underwood, A variable step central difference method for structural dynamics analysis: theoretical aspects, *Comp. Meths. Appl. Mech. Engrg.*, **22**, 241–258, 1980.
- [530] Park, K. C., Time integration of structural dynamics: a survey, Ch. 4.2 in *Pressure Vessels and Piping Design Technology – A Decade of Progress*, ASME, New York, 1982.
- [531] Park, K. C., A family of solution algorithms for nonlinear structural analysis based on the relaxation equations, *Int. J. Numer. Meth. Engrg.*, **18**, 1337–1347, 1982.
- [532] Park, K. C. and Felippa, C. A., Partitioned analysis of coupled systems, in: Chapter 3 of *Computational Methods for Transient Analysis*, ed. by T. Belytschko and T. J. R. Hughes, North-Holland, Amsterdam, pp. 157–219, 1983.
- [533] Park, K. C., Symbolic Fourier analysis procedures for C^0 finite elements, in *Innovative Methods for Nonlinear Problems*, ed. by W. K. Liu, T. Belytschko and K. C. Park, Pineridge Press, Swansea, UK, 269–293, 1984.
- [534] Park, K. C. and Felippa, C. A., Recent advances in partitioned analysis procedures, in: Chapter 11 of *Numerical Methods in Coupled Problems*, ed. by R. Lewis, P. Bettess and E. Hinton, Wiley, Chichester, 327–352, 1984.
- [535] Park, K. C. and Flaggs, D. L., A Fourier analysis of spurious modes and element locking in the finite element method, *Comp. Meths. Appl. Mech. Engrg.*, **42**, 37–46, 1984.
- [536] Park, K. C. and Flaggs, D. L., An operational procedure for the symbolic analysis of the finite element method, *Comp. Meths. Appl. Mech. Engrg.*, **46**, 65–81, 1984.
- [537] Park, K. C. and Stanley, G. M., A curved C^0 shell element based on assumed natural-coordinate strains, *J. Appl. Mech.*, **53**, 278–290, 1986.
- [538] Park, K. C. and W. K. Belvin, A partitioned solution procedure for control-structure interaction simulations, *J. Guidance, Control and Dynamics*, **14**, 59–67, 1991.
- [539] Park, K. C., Justino, M. R. Jr., and Felippa, C. A., An algebraically partitioned FETI method for parallel structural analysis: algorithm description, *Int. J. Numer. Meth. Engrg.*, **40**, 2717–2737, 1997.
- [540] Park, K. C. and Felippa, C. A., A variational framework for solution method developments in structural mechanics, *J. Appl. Mech.*, **65**, 242–249, 1998.
- [541] Park, K. C., Gumaste, U. and Felippa, C. A., A localized version of the method of Lagrange multipliers and its applications, *Comput. Mech.*, **24**, 476–490, 2000.
- [542] Park, K. C. and Felippa, C. A., A variational principle for the formulation of partitioned structural systems, *Int. J. Numer. Meth. Engrg.*, **47**, 395–418, 2000.
- [543] Park, K. C., Felippa, C. A. and Ohayon, R., Partitioned formulation of internal fluid-structure interaction problems via localized Lagrange multipliers, *Comp. Meths. Appl. Mech. Engrg.*, **190**(24–25), 2989–3007, 2001.
- [544] Park, K. C., Felippa, C. A., and Ohayon, R., Localized formulation of multibody systems, in: *Computational Aspects of Nonlinear Systems with Large Rigid Body Motion*, ed. by J. Ambrosio and M. Kleiber, NATO Science Series, IOS Press, 253–274, 2001.
- [545] Park, K. C. and Felippa, C. A., A variational principle for the formulation of partitioned structural systems, *Int. J. Numer. Meth. Engrg.*, **47**, 395–418, 2002.

Appendix R: REFERENCES (IN PROGRESS)

- [546] Park, K. C., Felippa, C. A., and Ohayon, R., The principal D'Alembert-Lagrange equations and applications to flexible floating systems, *Int. J. Numer. Meth. Engrg.*, **77**, 1072–1099, 2009.
- [547] Pars, L. A., *A Treatise in Analytical Dynamics*, Heinemann, London, 1965.
- [548] Patnaik, S. N., An integrated force method for discrete analysis, *Int. J. Numer. Meth. Engrg.*, **6**, 237–251, 1973.
- [549] Patnaik, S. N., The variational energy formulation for the integrated force method, *AIAA J.*, **24**, 129–136, 1986.
- [550] Pawsey, S. F. and Clough, R. W., Improved numerical integration of thick shell finite elements, *Int. J. Numer. Meth. Engrg.*, **3**, 545–586, 1971.
- [551] Peaceman, D. W. and Rachford Jr., H. H., The numerical solution of parabolic and elliptic differential equations, *SIAM J.*, **3**, 28–41, 1955.
- [552] Peano, A., Hierarchics of conforming elements for elasticity and plate bending, *Comput. Math. Appl.*, **2**, 3–4, 1976.
- [553] Pestel, E. C. and Leckie, F. A., *Matrix Methods in Elastomechanics*, McGraw-Hill, New York, 1963.
- [554] Pian, T. H. H., Derivation of element stiffness matrices by assumed stress distributions, *AIAA J.*, **2**, 1333–1336, 1964.
- [555] Pian, T. H. H., Element stiffness matrices for boundary compatibility and for prescribed boundary stresses, in *Proc. 1st Conf. on Matrix Methods in Structural Mechanics*, AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 457–478, 1966.
- [556] Pian, T. H. H. and Tong, P., Basis of finite element methods for solid continua, *Int. J. Numer. Meth. Engrg.*, **1**, 3–29, 1969.
- [557] Pian, T. H. H. and Sumihara, K., Rational approach for assumed stress finite elements, *Int. J. Numer. Meth. Engrg.*, **20**, 1685–1695, 1984.
- [558] Pian, T. H. H. and Tong, P., Relations between incompatible displacement model and hybrid stress model, *Int. J. Numer. Meth. Engrg.*, **22**, 173–181, 1986.
- [559] Pian, T. H. H., Some notes on the early history of hybrid stress finite element method, *Int. J. Numer. Meth. Engrg.*, **47**, 419–425, 2000.
- [560] Pietraszkiewicz, W., Finite rotations in shells, in *Theory of Shells, Proc. 3rd IUTAM Symp on Shell Theory, Tsibili 1978*, 445–471, North Holland, Amsterdam, 1980.
- [561] Pietraszkiewicz, W., (ed.), *Finite Rotations in Structural Mechanics*, Lecture Notes in Engineering 19, Springer-Verlag, Berlin, 1985.
- [562] Piperno, S. and Farhat, C., Design of efficient partitioned procedures for the transient solution of aeroelastic problems, *Revue Européenne Eléments Finis*, **9**, 655–680, 2000.
- [563] Piperno, S., and Farhat, C., Partitioned procedures for the transient solution of coupled aeroelastic problems: an energy transfer analysis and three-dimensional applications, *Comp. Meths. Appl. Mech. Engrg.*, **190**, 3147–3170, 2001.
- [564] Pönish, G. and Schwetlik, H., Computing turning points using curves implicitly defined by nonlinear equations depending on a parameter, *Computing J.*, **26**, 107–121, 1981.
- [565] Pönish, G., Computing simple bifurcation points using a minimally extended system of nonlinear equations, *Computing J.*, **35**, 277–294, 1985.
- [566] Pope, G. G., The application of the matrix displacement method in plane elasto-plastic problems, *Proceedings Conference on Matrix Methods in Structural Engineering*, ed. by R. Bader et. al., AFFDL-TR-66-80, Wright-Patterson AFB, Dayton, Ohio, 635–654, 1966.

- [567] Popov, E. P., *Engineering Mechanics of Solids*, Prentice Hall, Englewood Cliffs, N. J., 2nd ed., 1991.
- [568] Poston, T. and Steward, I., *Catastrophe Theory and its Applications*, Pitman, London, 1978.
- [569] Powell, G. H. and Simons, J., Improved iteration strategy for nonlinear structures, *Int. J. Numer. Meth. Engrg.*, **17**, 1455–1467, 1981.
- [570] Powell, M. J. D., A method for nonlinear constraints in optimization problems, in *Optimization*, ed. by R. Fletcher, Academic Press, London, 283–298, 1969.
- [571] Prager, W. and Synge, J. L., Approximations in elasticity based on the concept of function space, *Quart. Appl. Meth.*, **5**, 241–269, 1947.
- [572] Prager, W., Variational principles for linear elastostatics for discontinuous displacements, strains and stresses, in *Recent Progress in Applied Mechanics*, The Folke-Odgvist Volume, ed. by B. Broger, J. Hult and F. Niordson, Almquist and Wiksell, Stockholm, 463–474, 1967.
- [573] Prange, G., Der Variations- und MinimalPrinzip der Statik der Baukonstruktionen, *Habilitationsschrift*, Tech. Univ. Hanover, 1916.
- [574] Press, W. J. et al., *Numerical Recipes: The Art of Scientific Computing*, 2nd ed., Cambridge Univ. Press, 1992.
- [575] Przemieniecki, J. S., *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968; Dover edition 1986.
- [576] Pugh, E. D., Hinton, E., and Zienkiewicz, O. C., A study of quadrilateral plate bending elements with reduced integration, *Int. J. Numer. Meth. Engrg.*, **12**, 1059–1078, 1978.
- [577] Punch, E. F. and Atluri, S. N., Applications of isoparametric three-dimensional hybrid stress elements with least-order fields, *Computers & Structures*, **19**, 406–430, 1984.
- [578] Punch, E. F. and Atluri, S. N., Development and testing of stable, invariant, isoparametric curvilinear 2- and 3D hybrid stress elements, *Comp. Meths. Appl. Mech. Engrg.*, **47**, 331–356, 1984.

Q

R

- [579] Rabinowitz, P. H., *Numerical Methods for Nonlinear Algebraic Equations*, Gordon and Breach, New York, 1970.
- [580] Raimes, S., *The Wave Mechanics of Electrons in Metals*, North-Holland, Amsterdam, 1967.
- [581] Rajasekaran, S. and Murray, D. W., Incremental finite element matrices, *J. Str. Div. ASCE*, **99**, 2423–2438, 1973.
- [582] Rall, L. B., *Computational Solution of Nonlinear Operator Equations*, Wiley, New York, 1969.
- [583] Rall, L. B. (ed.), *Nonlinear Functional Analysis and Applications*, Academic Press, New York, 1971.
- [584] Ralston, A. and Rabinowicz, P., *A First Course in Numerical Analysis*, 2nd ed., Dover, New York, 2001.
- [585] Ramm, E., A plate/shell element for large deflections and rotations, *Proc. US-Germany Symposium on Formulations and Algorithms in Finite Element Analysis*, MIT-Cambridge, Mass., 264–293, 1976.
- [586] Ramm, E., Strategies for tracing the nonlinear response near limit points, in *Proc. Europe-US Workshop on Nonlinear Finite Element Analysis in Structural Mechanics*, Bochum 80, Springer-Verlag, Berlin, 1981.

Appendix R: REFERENCES (IN PROGRESS)

- [587] Ramm, E., The Riks/Wempner approach – an extension of the displacement control method in nonlinear analysis, in *Recent Advances in Nonlinear Computational Mechanics*, ed. by E. Hinton *et. al.*, Pineridge Press, Swansea, U.K. 63–86, 1982.
- [588] Rand, T., An approximate method for computation of stresses in sweptback wings, *J. Aero. Sci.*, **18**, 61–63, 1951.
- [589] Rankin, C. C., Brogan, F. A., Loden, W. A., and Cabiness, H., *STAGS User Manual*, Lockheed Mechanics, Materials and Structures Report P032594, Version 3.0, 1998.
- [590] Rankin, C. C. and Brogan, F. A., An element-independent corotational procedure for the treatment of large rotations, *ASME J. Pressure Vessel Technology*, **108**, 165–174, 1986.
- [591] Rankin, C. C., Consistent linearization of the element-independent corotational formulation for the structural analysis of general shells, *NASA Contractor Report 278428*, Lockheed Palo Alto Res. Lab., CA, 1988.
- [592] Rankin, C. C., Consistent linearization of the element-independent corotational formulation for the structural analysis of general shells, *NASA Contractor Report 278428*, Lockheed Palo Alto Research Laboratory, Palo Alto, CA, 1988.
- [593] Rankin, C. C. and Nour-Omid B., The use of projectors to improve finite element performance, *Computers & Structures*, **30**, 257–267, 1988.
- [594] Rankin, C. C., On the choice of best possible corotational element frame, in *Modeling and Simulation Based Engineering*, ed by S. N. Atluri and P. E. O’Donoghue, Tech Science Press, Palmdale, CA, 1998.
- [595] Rankin, C. C., Brogan, F. A., W. A. Loden and H. Cabiness, *STAGS User Manual*, LMMS P032594, Version 3.0, January 1998.
- [596] Rao, C. R. and Mitra, S. K., *Generalized Inverse of Matrices and its Applications*, Wiley, New York, 1971.
- [597] Rashid, Y. S., Three dimensional analysis of elastic solids: I. Analysis procedure, *Int. J. Solids Struc.*, **5**, 1311–1331, 1969.
- [598] Rashid, Y. S., Three dimensional analysis of elastic solid: II. The computational problem, *Int. J. Solids Struc.*, **6**, 195–207, 1970.
- [599] Razzaque, A., Program for triangular bending elements with derivative smoothing, *Int. J. Numer. Meth. Engrg.*, **6**, 333–343, 1973.
- [600] Reddy, J. N., *Energy and Variational Methods in Applied Mechanics*, Wiley, 1986.
- [601] Reddy, J. N., *Mechanics of Laminated Composite Plates*, CRC Press, Boca Raton, FL, 1997.
- [602] Reid, J. K., On the method of conjugate gradients for the solution of large sparse systems of equations, in *Large Sparse Sets of Linear Equations*, ed. by J. K. Reid, Academic Press, London 231–254, 1971.
- [603] Reissner, E., The effect of transverse shear deformation on the bending of elastic plates, *J. Appl. Mech.*, **12**, 69–77, 1945.
- [604] Reissner, E., On bending of elastic plates, *Quart. Appl. Meth.*, **5**, 55–68, 1947.
- [605] Reissner, E., On a variational theorem in elasticity, *J. Math. Phys.*, **29**, 90–95, 1950.
- [606] Rheinboldt, W. C., Numerical methods for a class of finite-dimensional bifurcation problems, *SIAM J. Numer. Anal.*, **15**, 1–11, 1978.
- [607] Rheinboldt, W. C., Numerical analysis of continuation methods for nonlinear structural problems, *Computers & Structures*, **13**, 130–141, 1981.

- [608] Rheinboldt, W. C., Computation of critical boundaries on equilibrium manifolds, *SIAM J. Numer. Anal.*, **19** 653–669, 1982.
- [609] Rheinboldt, W. C. and Burkardt, J. V., A locally parametrized continuation process, *ACM Trans. Math. Software*, **9**, 215–235, 1983.
- [610] Rheinboldt, W. C., *Numerical Analysis of Parametrized Nonlinear Equations*, Wiley, New York, 1986.
- [611] Richtmyer, R. L. and Morton, K. W., *Difference Methods for Initial Value Problems*, 2nd ed., Interscience Pubs., New York, 1967.
- [612] Riks, E., The application of Newton's method to the problem of elastic stability, *J. Appl. Mech.*, **39**, 1060–1065, 1972.
- [613] Riks, E., An incremental approach to the solution of snapping and buckling problems, *Int. J. Solids Struc.*, **15**, 329–351, 1979.
- [614] Riks, E., Some computational aspects of the stability analysis of nonlinear structures, *Comp. Meths. Appl. Mech. Engrg.*, **47**, 219–260, 1984.
- [615] Riks, E., Bifurcation and stability: a numerical approach, in *Innovative Methods for Nonlinear Problems*, ed. by W. K. Liu, T. Belytschko and K. C. Park, Pineridge Press, Swansea, U.K., 313–344, 1984.
- [616] Riks, E., Progress in collapse analysis, *J. Pressure Vessels Tech.*, **109**, 33–41, 1987.
- [617] Robinson, J., An evaluation of skew sensitivity of 33 plate bending elements in 19 FEM systems, *Finite Element News*, January 1985, also reprinted in *Finite Element Standards Forum – Vol 2*, ed. by K. J. Forsberg and H. H. Fong, held at 26th SDM Conference, Orlando, Florida, 1985.
- [618] Robinson, J. and G. W. Haggemacher, G. W., LORA – An accurate four-node stress plate bending element, *Int. J. Numer. Meth. Engrg.*, **14**, pp. 296–306, 1979
- [619] Rodrigues, O., Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacement considérées indépendent des causes qui peuvent les produire, *J. de Mathématiques Pures et Appliquées*, **5**, 380–400, 1840.
- [620] Roark, R. J., Budynas, R. G., and Young, W. C., *Roark's Formulas for Stress and Strain*, McGraw-Hill, New York, 7th ed., 2001.
- [621] Robinson, J., *Structural Matrix Analysis for the Engineer*, Wiley, New York, 1966.
- [622] Robinson, J., *EarlyFEM Pioneers*, Robinson and Associates, Dorset, 1966.
- [623] Ross, M. R., *Coupling and Simulation of Acoustic Fluid-Structure Interaction Systems Using Localized Lagrange Multipliers*, Ph.D. Thesis, Department of Aerospace Engineering Science, University of Colorado, Boulder, 2006.
- [624] Ross, M. R., Felippa, C. A., Park, K. C., and Sprague, M. A., Acoustofluid-structure interaction by localized Lagrange multipliers: formulation, *Comp. Meths. Appl. Mech. Engrg.*, **197**, 3057–3079, 2008.
- [625] M. R. Ross, M. A. Sprague, C. A. Felippa and K. C. Park, Treatment of acoustic fluid-structure interaction by localized Lagrange multipliers and comparison to alternative interface coupling methods, *Comp. Meths. Appl. Mech. Engrg.*, **198**, 986–1005, 2009.
- [626] Routh, E. J., *A Treatise on the Stability of a Given State of Motion, Particularly Steady Motion*, Macmillan, London, 1877; reprinted by Pranava Books, 2008.
- [627] Ruskeepaa, H. *Mathematica Navigator: Mathematics, Statistics, and Graphics*, Academic Press, 2004.

Appendix R: REFERENCES (IN PROGRESS)

- [628] Ruskeepaa, H., *Mathematica Navigator: Graphics and Methods of Applied Mathematics*, Academic Press, 2004.

S

- [629] Saad, Y., *Numerical Methods For Large Eigenvalue Problems*, second revised edition, SIAM, Philadelphia, 2011.
- [630] Sander, G. and Beckers, P. The influence of the choice of connectors in the Finite Element Method, in *The Mathematical Aspects of the Finite Element Method*, Lecture Notes in Mathematics, Vol. 606, Springer-Verlag, Berlin, 316ff, 1977.
- [631] Santilli, R., *Foundations of Theoretical Mechanics I*, Springer-Verlag, Berlin, 1978.
- [632] Scales, L. E., *Introduction to Non-Linear Optimization*, Springer-Verlag, New York, NY, 1985.
- [633] Schmidt, W. F., Adaptive stepsize selection for use with the continuation method, *Int. J. Numer. Meth. Engrg.*, **12**, 677–694, 1978.
- [634] Schuerch, H. U., Delta wing design analysis, presented at the *Natl. Aeron. Meeting*, Soc. Autom. Engrg, Preprint 441, Los Angeles, 1953.
- [635] Schur, I., Über Potenzreihen, die im Innern des Einheitnetkreises beschränkt sind [I]., *J. Reine Angew. Math.*, **147**, 205–232, 1917. English translation: in On power series which are bounded in the interior of the unit circle, in *Schur Methods in Operator Theory and Signal Processing. Operator Theory: Advances and Applications OT18*, ed. by I. Gohberg, Birkäuser Verlag, Basel, 31–59 and 61–88, 1986.
- [636] Senge, P., *The Fifth Discipline: The Art and Practice of the Learning Organization*, Doubleday, New York, Paris, 1990.
- [637] Seydel, R., *From Equilibrium to Chaos — Practical Bifurcation and Stability Analysis*, Elsevier, New York, 1988.
- [638] Sewell, M. J., On the connexion between stability and the shape of the equilibrium surface, *J. Mech. Phys. Solids*, **14**, 203–230, 1966.
- [639] Sewell, M. J., A general theory of equilibrium paths through critical points I-II, *Proc. Roy. Soc. London*, **A306**, 201–238, 1968.
- [640] Sewell, M. J., *Maximum and Minimum Principles*, Cambridge Univ. Press, Cambridge, 1987.
- [641] Shabana, A. A., *Dynamics of Multibody Systems*, Cambridge University Press, Cambridge, 1998.
- [642] Sharifi, P. and Popov, E. P., , Nonlinear buckling analysis of sandwich arches, *J. Engrg. Div. ASCE* **97**, 1397–1411, 1971.
- [643] Sharifi, P. and Popov, E. P., Nonlinear finite element analysis of sandwich shells of revolution, *AIAA J.*, **11**, 715–722, 1973.
- [644] Sherman, J. and Morrison, W. J., Adjustment of an inverse matrix corresponding to changes in the elements of a given column or a given row of the original matrix, *Ann. Math. Stat.*, **20**, 621, 1949; also: Adjustments of an inverse matrix corresponding to a change in one element of a given matrix, *Ann. Math. Stat.*, **21**, 124, 1950.
- [645] Simo, J. C., Wriggers, P., Schweizerhof, K. H., and Taylor, R. L., Finite deformation postbuckling analysis involving inelasticity and contact constraints, in *Innovative Methods for Nonlinear Problems*, ed. by W. K. Liu, T. Belytschko and K. C. Park, Pineridge Press, Swansea, U.K., 365–388, 1984.
- [646] Simo, J. C., A finite strain beam formulation. Part I: The three-dimensional dynamic problem, *Comp. Meths. Appl. Mech. Engrg.*, **49**, 55–70, 1985.

- [647] Simo, J. C. and L. Vu-Quoc, A finite strain beam formulation. Part II: Computational aspects, *Comp. Meths. Appl. Mech. Engrg.*, **58**, 79–116, 1986.
- [648] Simo, J. C. and Hughes, T. J. R., On the variational foundations of assumed strain methods, *J. Appl. Mech.*, **53**, 51–54, 1986.
- [649] Simo, J. C. and Rifai, M. S., A class of mixed assumed strain methods and the method of incompatible modes, *Int. J. Numer. Meth. Engrg.*, **29**, 1595–1638, 1990.
- [650] Simmonds, J. G. and Danielson, D. A., Nonlinear shell theory with finite rotation and stress function vectors, *J. Appl. Mech.*, **39**, 1085–1090, 1972.
- [651] Sivertsen, O. I., *Virtual Testing of Mechanical Systems: Theories and Techniques*, Swets & Zeitlinger Pubs., Heereweg, Netherlands, 2001.
- [652] Skallerud, B. and Haugen, B., Collapse of thin shell structures: Stress resultant plasticity modeling within a co-rotated ANDES finite element formulation, *Int. J. Numer. Meth. Engrg.*, **46**, 1961–1986, 1999.
- [653] Skallerud, B., Holthe, K., and Haugen, B., Combining high-performance thin shell and surface crack finite elements for simulation of combined failure modes, *Proc. 7th US Nat. Congress in Computational Mechanics*, Albuquerque, NM, July 2003.
- [654] Skeie, G. and Felippa, C. A., A local hyperelliptic constraint for nonlinear analysis, *Proceedings of NUMETA'90 Conference*, Swansea, Wales, Elsevier Sci. Pubs, 1990.
- [655] Skeie, G., The Free Formulation: linear theory and extensions with applications to tetrahedral elements with rotational freedoms, *Ph. D. Dissertation*, Division of Structural Mechanics, NTH, Trondheim, Norway, 1991.
- [656] Sobel, L. H. and Thomas, K. (eds.), *Collapse Analysis of Structures*, PVP Vol. 84, ASME, New York, 1984.
- [657] Soedel, W., *Vibrations of Plates and Shells*, Marcel Dekker, 1993.
- [658] Sokolnikoff, I., *The Mathematical Theory of Elasticity*, McGraw-Hill, 2nd ed., 1956.
- [659] Sommerfeld, A., *Mechanics: Lectures in Theoretical Physics Vol. 1*, Academic Press, London, 1964. (English translation from the 4th German edition of 1942.)
- [660] Sortais, Y. R., *La Géométrie du Triangle*, Hermann, Paris, 1987.
- [661] Spilker, R. and Singh, S. P., Three-dimensional hybrid-stress isoparametric quadratic displacement elements, *Int. J. Numer. Meth. Engrg.*, **18**, 445–465, 1982.
- [662] Sprague, M. A. and Geers, T. L., A spectral-element method for modeling cavitation in transient fluid-structure interaction, *Int. J. Numer. Meth. Engrg.*, **60**, 2467–2499, 2004.
- [663] Spurrier, R. A., A comment on singularity-free extraction of a quaternion from a direction cosine matrix, *J. Spacecrafts & Rockets*, **15**, p. 255, 1978.
- [664] Stanley, G. M., Continuum-based shell elements, *Ph. D. Dissertation*, Stanford University, 1985.
- [665] Stanley, G. M., Park, K. C., and Hughes, T. J. R., Continuum based resultant shell elements, in *Finite Element Methods for Plate and Shell Structures, Vol. I: Element Technology*, ed. by T. J. R. Hughes and E. Hinton, Pineridge Press, Swansea, U.K., 1986, 1–45.
- [666] Stein, E., Wagner, W., and Wriggers, P., Finite element analysis of stability problems with contact, in *Finite Element Methods for Nonlinear Problems*, ed. by P. G. Bergan, K. J. Bathe, and W. Wunderlich, Springer, Berlin, 1986.
- [667] Stewart, G. W., *Introduction to Matrix Computations*, Academic Press, New York, 1973.

Appendix R: REFERENCES (IN PROGRESS)

- [668] Stewart, G. W. and Sun, J. G., *Matrix Perturbation Theory*, Academic Press, Boston, 1990.
- [669] Stewart, G. W., *Matrix Algorithms. Vol 2: Eigenproblems*, SIAM, Philadelphia, 2001.
- [670] Stoer, J., Conjugate gradient type methods, in *Mathematical Programming: The State of the Art*, ed. by A. Bachem, M. Grötschel and B. Korte, Springer-Verlag, Berlin, 540–565, 1983.
- [671] Strang, G., Variational crimes in the finite element method, in *The Mathematical Foundations of the Finite Element Method with Applications to Partial Differential Equations*, ed. by A. K. Aziz, Academic Press, New York, 689–710, 1972.
- [672] Strang, G. and Fix, G., *An Analysis of the Finite Element Method*. Prentice-Hall, 1973.
- [673] Strang, G., *Linear Algebra and its Applications*, Academic Press, New York, 1976.
- [674] Strang, G., The Quasi-Newton method in finite element computations, in *Computational Methods in Nonlinear Mechanics*, ed. by J. T. Oden, North-Holland, Amsterdam, 451–456, 1980.
- [675] Stricklin, J., Haisler, W., Tisdale, P., and Gunderson, R., A rapidly converging triangular plate bending element, *AIAA J.*, **7**, 180–181, 1969.
- [676] Stricklin, J. A., Haisler, W. E. and Von Riesenmann, W. A., Self-correcting initial value formulations in nonlinear structural mechanics, *AIAA J.*, **9**, 2066–2067, 1971.
- [677] Stricklin, J. A., Von Riesenmann, W. A., Tillerson, J. R., and Haisler, W. E., Static geometric and material nonlinear analysis, in *Proc. 2nd U.S.-Japan Seminar on Advances in Computational Methods in Structural Mechanics and Design*, ed. by J. T. Oden, R. W. Clough and Y. Yamamoto, UAH Press, University of Alabama, Huntsville, 301–324, 1972.
- [678] Stricklin, J. A., Haisler, W. E. and von Riesenmann, W. A., Evaluation of solution procedures for nonlinear structural analysis, *AIAA J.*, **11**, 292–299, 1973.
- [679] Stricklin, J. A. and Haisler, W. E., Formulation and solution procedures for nonlinear structural analysis, *Computers & Structures*, **7**, 125–136, 1977.
- [680] Stroud, A. H. and Secrest, D., *Gaussian Quadrature Formulas*, Prentice-Hall, Englewood Cliffs, NJ, 1966.
- [681] Stroud, A. H., *Approximate Calculation of Multiple Integrals*, Prentice-Hall, Englewood Cliffs, NJ, 1971.
- [682] Struik, D. J., *Lectures in Classical Differential Geometry*, Addison-Wesley, 2nd ed., 1961.
- [683] Stuart, A. M. and Humphries, A. R., *Dynamic Systems and Numerical Analysis*, Cambridge Univ. Press, Cambridge, 1996.
- [684] Stummel, F., The limitations of the patch test, *Int. J. Numer. Meth. Engrg.*, **15**, 177–188, 1989.
- [685] Stummel, F., The generalized patch test, *SIAM J. Numer. Anal.*, **16**, 449–471, 1979.
- [686] Synge, J. R. *The Hypercircle in Mathematical Physics*, Cambridge Univ. Press, Cambridge, 1957.
- [687] Synge, J. L., Classical dynamics, in *Handbuch der Physik*, vol. III/1, ed. by S. Flügge, Springer-Verlag, Berlin, 1–225, 1960.
- [688] Szabo, B. and Babuska, I., *Finite Element Analysis*, Wiley, New York, 1991.
- [689] Szilard, R., *Theory and Analysis of Plates*, Prentice-Hall, 1974.
- [690] Szwabowicz, M. L., Variational formulation in the geometrically nonlinear thin elastic shell theory, *Int. J. Solids Struc.*, **22**, 1161–1175, 1986.

T

- [691] Taig, I. C. and Kerr, R. I., Some problems in the discrete element representation of aircraft structures, in *Matrix Methods of Structural Analysis*, ed. by B. M. Fraeijs de Veubeke, Pergamon Press, London, 1964.
- [692] Tanner, R., *Engineering Rheology*, Oxford University Press, 1985.
- [693] Taylor, C., Hinton, E., and Owen, D. J. R. (eds.), *Numerical Methods for Nonlinear Problems I*, Pineridge Press, Swansea, U. K., 1981.
- [694] Taylor, C., Hinton, E., Owen, D. J. R., and Oñate, E., (eds.), *Numerical Methods for Nonlinear Problems II*, Pineridge Press, Swansea, U. K., 1984.
- [695] Taylor, R. L., Pister, K. S., and Herrmann, L. R., A variational principle for incompressible and nearly incompressible orthotropic elasticity, *Int. J. Solids Struc.*, **4**, 875-883, 1968.
- [696] Taylor, R. L., Wilson, E. L., and Beresford, P. J., A nonconforming element for stress analysis, *Int. J. Numer. Meth. Engrg.*, **10**, 1211-1219, 1976.
- [697] Taylor, R. L., Simo, J. C., Zienkiewicz, O. C., and Chan, A. C., The patch test: a condition for assessing FEM convergence, *Int. J. Numer. Meth. Engrg.*, **22**, 39-62, 1986.
- [698] Teigen, J. G., Nonlinear analysis of concrete structures based on a 3D shear-beam element formulation. *Ph.D Dissertation*, Department of Mathematics, Mechanics Division, University of Oslo, Oslo, Norway, 1994.
- [699] Tessler, A., On a conforming, Mindlin-type plate element, in *Proc. MAFELAP 1981*, ed. by J. H. Whiteman, Academic Press, London, 119-126, 1981.
- [700] Tessler, A. and Hughes, T. J. R., A three-node Mindlin plate element with improved transverse shear, *Comp. Meths. Appl. Mech. Engrg.*, **50**, 71-101, 1985.
- [701] Thomson, W. and Tait, P. G., *Treatise on Natural Philosophy*, Part I, Cambridge University Press, Cambridge, 1867.
- [702] Thompson, J. M. T. and Hunt, G. W., *A General Theory of Elastic Stability*, Wiley, London, 1973.
- [703] Thompson, J. M. T., *Instabilities and Catastrophes in Science and Engineering*, Wiley, London, 1982.
- [704] Thompson, J. M. T. and Hunt, G. W., *Static and Dynamic Instability Phenomena*, Wiley, London, 1983.
- [705] Thurston, G. A., Continuation of Newton's method through bifurcation points, *J. Appl. Mech.*, **36**, 425-430, 1969.
- [706] Tillerson, J. R., Stricklin, J. A., and Haisler, W. E., Numerical methods for the solution of nonlinear problems in structural analysis, in *Numerical Solution of Nonlinear Problems*, ed. by R. F. Hartung, AMD Vol. 6, ASME, New York, 1972.
- [707] Timoshenko, S. P., On the correction for shear of the differential equation for transverse vibration of prismatic bars, *Phil. Mag.*, **XLI**, 744-46, 1921. Reprinted in *The Collected Papers of Stephen P. Timoshenko*, McGraw-Hill, London, 1953. See also S. P. Timoshenko and D. H. Young, *Vibration Problems in Engineering*, 3rd edition, Van Nostrand, 329-331, 1954.
- [708] Timoshenko, S. P. and Gere, J. M., *Theory of Elastic Stability*, McGraw-Hill, New York, expanded 2nd ed., 1961; reprinted by Dover, New York, 2009. First edition by S. P. Timoshenko published by McGraw-Hill, 1936.
- [709] Timoshenko, S. P. and Goodier, J. N., *Theory of Elasticity*, McGraw-Hill, New York, 1951.
- [710] Timoshenko, S. P. and Young, D. N., *Vibration Problems in Engineering*, Van Nostrand, Princeton, N.J., 1955.

Appendix R: REFERENCES (IN PROGRESS)

- [711] Timoshenko, S. P. and Woinowsky-Krieger, S., *Theory of Plates and Shells*, McGraw-Hill, New York, 1959.
- [712] Tocher, J. L., Analysis of plate bending using triangular elements, *Ph. D. Dissertation*, Dept. of Civil Engineering, Univ. of California, Berkeley, CA, 1962.
- [713] Tocher, J. L. and Kapur, K. K., Discussion of 'Basis for derivation of matrices for the direct stiffness method' by R. J. Melosh, *AIAA J.*, **3**, 1215–1216, 1965.
- [714] Tocher, J. L. and Felippa, C. A., Computer graphics applied to production structural analysis, in *Proceedings IUTAM Symposium on High-Speed Computing of Elastic Structures*, ed. by B. M. Fraeijs de Veubeke, Université de Liège Press, Liège, Belgium, 1971.
- [715] Tocher, J. L. and Herness, E. D., A critical view of NASTRAN, in: *Numerical and Computer Codes in Structural Mechanics*, ed. by S. J. Fenves, N. Perrone, A. R. Robinson, and W. C. Schnobrich, Academic Press, New York, 151–173, 1973.
- [716] Tong, P., Exact solution of certain problems by the finite element method, *AIAA J.*, **7**, 179–180, 1969.
- [717] Tong, P., New displacement finite element method for solid continua, *Int. J. Numer. Meth. Engrg.*, **2**, 73–83, 1970.
- [718] Tonti, E., The reason for analogies between physical theories, *Appl. Math. Model.*, **1**, 37–50, 1977.
- [719] Traub, J. F., *Iterative Methods for the Solution of Equations*, Prentice-Hall, Englewood Cliffs, New Jersey, 1964.
- [720] Truesdell, C., The mechanical foundations of elasticity and fluid dynamics, *J. Rat. Mech. Anal.*, **1**, 125–300, 1952. Corrected and expanded reprint in: Truesdell, C., *Continuum Mechanics I: The Mechanical Foundations of Elasticity and Fluid Dynamics*, Gordon & Breach, New York, 1966.
- [721] Truesdell, C. and Toupin, R. A., The classical field theories, in *Handbuch der Physik*, ed. by S. Flügge, vol. III/1, 226–790, Springer Verlag, Berlin, 1960.
- [722] Truesdell, C., *The Tragical History of Thermodynamics*, Springer-Verlag, Berlin, 1980.
- [723] Truesdell, C., *An Idiot's Fugitive Essays in Science: Methods, Criticism, Training, Circumstances*, Springer-Verlag, Berlin, 1984.
- [724] Truesdell, C., Hypoelasticity, *J. Rat. Mech. Anal.*, **4**, 83–133, 1019–1020, 1955. Reprinted in C. Truesdell (ed.), *Continuum Mechanics III: Foundations of Elasticity Theory*, Gordon & Breach, New York, 1965.
- [725] Truesdell, C., and Noll, W., The nonlinear field theories of mechanics, in: *Handbuch der Physik*, ed. by S. Flügge, Vol. III/3, Springer-Verlag, 1965.
- [726] Turnbull, H. W., *The Theory of Determinants, Matrices and Invariants*, Blackie and Sons, London, 1929. Expanded reprint by Dover, 1960.
- [727] Turner, M. J., Clough, R. W., Martin, H. C., and Topp, L. J., Stiffness and deflection analysis of complex structures, *J. Aero. Sci.*, **23**, 805–824, 1956.
- [728] Turner, M. J., The direct stiffness method of structural analysis, Structural and Materials Panel Paper, AGARD Meeting, Aachen, Germany, 1959.
- [729] Turner, M. J., Dill, E. H., Martin, H. C., and Melosh, R.J., Large deflection analysis of complex structures subjected to heating and external loads, *J. Aero. Sci.*, **27**, 97–107, 1960.

- [730] Turner, M. J., Martin, H. C., and Weikel, B. C., Further developments and applications of the stiffness method, in *Matrix Methods of Structural Analysis*, ed. by B. M. Fraeijs de Veubeke, AGARDograph 72, Pergamon Press, Oxford, 203–266, 1964.

U

- [731] Udwadia, F. E., and Kalaba, R. E., *Analytical Dynamics*, Cambridge, 1996.
- [732] Utku, S., Stiffness matrices for thin triangular elements of nonzero Gaussian curvature, *AIAA J.*, **37**, 1659–1667, 1967.
- [733] Underwood, P. G., Dynamic relaxation, Ch. 5 in *Computational Methods for Transient Dynamic Analysis*, ed. by T. Belytschko and T. J. R. Hughes, North-Holland, Amsterdam, 1983.
- [734] Uspenky, J. V., *Theory of Equations*, McGraw-Hill, New York, 1948.

V

- [735] Vainberg, M. M., *Variational Methods for the Study of Nonlinear Operators*, Holden-Day, 1964.
- [736] Valanis, K. C., A theory of viscoplasticity without a yield surface, Part II: Application to mechanical behavior of metals, *Arch. Mech.*, **23**, 535–551, 1971.
- [737] Van Dyke, M. D., *Perturbation Methods in Fluid Mechanics*, 2nd annotated ed., Parabolic Press, 1975.
- [738] Venkayya, V. B., Khot, N. S. and Reddy, V. S., Optimization of structures based on the study of energy distribution, in *Proceedings 2nd Conference on Matrix Methods in Structural Mechanics*, ed. by L. Berke et. al., AFFDL-TR-68-150, Air Force Institute of Technology, Dayton, Ohio, 111–154, 1968.
- [739] Verchery, G., Régularisation du système de l' équilibre des structures élastiques discrètes, *Comptes Rendus à l'Académie des Sciences*, Paris, t. 311, Série II, 585–589, 1990.
- [740] von Bertalanffy, L., *General Systems Theory: Foundations, Development, Applications*, Braziller, New York, 1968; revised edition 1970.
- [741] von Kármán, T., Festigkeitsprobleme im Maschinenbau, *Encyklopädie der Mathematischen Wissenschaften*, **4**, 311–385, 1910.
- [742] Vujanovic, B. D. and S. E. Jones, S. E., *Variational Methods in Nonconservative Phenomena*, Academic Press, New York, 1989.

W

- [743] Wachpress, E. I., *A Rational Finite Element Basis*, Academic Press, New York, 1975.
- [744] Wacker, H. J. (ed.), *Continuation Methods*, Academic Press, New York, 1978.
- [745] Waltz, J. E., Fulton, R. E., and Cyrus, N. J., Accuracy and convergence of finite element approximations, *Proc. Second Conf. on Matrix Methods in Structural Mechanics*, WPAFB, Ohio, Sep. 1968, in *AFFDL TR 68-150*, 995–1028, 1968.
- [746] Wahlbin, L. B., *Superconvergence in Galerkin Finite Element Methods*, Lecture Notes in Mathematics 1605, Springer-Verlag, Berlin, 1995.
- [747] Wang, D. W., Katz, I. M., and Szabo, B. A., *h*- and *p*-version finite element analysis of a rhombic plate, *Int. J. Numer. Meth. Engrg.*, **20**, 1399–1405, 1984.
- [748] Warming, R. F. and Hyett, B. J., The modified equation approach to the stability and accuracy analysis of finite difference methods, *J. Comp. Physics*, **14**, 159–179, 1974.

Appendix R: REFERENCES (IN PROGRESS)

- [749] Washizu, K., *Variational Methods in Elasticity and Plasticity*, Pergamon Press, 1972. Second expanded edition 1981.
- [750] Watson, L. T., An Algorithm that is globally convergent with probability one for a class of nonlinear two-point boundary value problems, *SIAM J. Numer. Anal.*, **16**, 394–401, 1979.
- [751] Watson, L. T. and Holzer, S. M., Quadratic convergence of Crisfield's method, *Computers & Structures*, **17**, 69–72, 1983.
- [752] Wasserstrom, E., Numerical solutions by the continuation method, *SIAM Review*, **15**, 89–119, 1973.
- [753] Weinstock, R., *Calculus of Variations, with Applications to Physics and Engineering*, McGraw-Hill, 1952. In Dover edition since 1974.
- [754] Weisstein, E. W., *CRC Concise Encyclopedia of Mathematics*, Chapman-Hall CRC, 2nd ed., 2002.
- [755] Wempner, G. A., Finite elements, finite rotations and small strains of flexible shells, *Int. J. Solids Struc.*, Vol 5, 117–153, 1969.
- [756] Wempner, G. A. , Discrete approximations related to nonlinear theories of solids, *Int. J. Solids Struc.*, **7**, 1581–1599, 1971.
- [757] Werner, B. and Spence, A., The computation of symmetry-breaking bifurcation points, *SIAM J. Numer. Anal.*, **8**, 767–785, 1971.
- [758] White, R. E., *An Introduction to the Finite Element Method with Applications to Nonlinear Problems*, Wiley, New York, 1985.
- [759] Whitney, J. K., *Structural Analysis of Laminated Anisotropic Plates*, Technomic Publ. Co., Lancaster, PA, 1987.
- [760] Wilf, H. S., *Generatingfunctionology*, Academic Press, New York, 1991.
- [761] Wilkinson, J. H., *The Algebraic Eigenvalue Problem*, Oxford Univ. Press, New York, 1965.
- [762] Wilkinson, J. H. and C. H. Reinsch, C. H. (eds.), *Handbook for Automatic Computation. Linear Algebra*, vol 2., Springer-Verlag, Berlin, 1971.
- [763] Willam, K. J., Finite element analysis of cellular structures, *Ph. D. Dissertation*, Dept. of Civil Engineering, Univ. of California, Berkeley, CA, 1969.
- [764] Willam, K. J., Numerical solution of inelastic rate processes, *Computers & Structures*, **8**, 511–531, 1978.
- [765] Wilson, E. L., Finite element analysis of two-dimensional structures, *Ph. D. Dissertation*, Department of Civil Engineering, University of California at Berkeley, 1963.
- [766] Wilson, E. L., Taylor, R. L., Doherty, W. P., and Ghaboussi, J., Incompatible displacement models, in *Numerical and Computer Models in Structural Mechanics*, ed. by S. J. Fenves, N. Perrone, A. R. Robinson, and W. C. Schnobrich, Academic Press, New York, 43–57, 1973.
- [767] Wilson, E. L., The static condensation algorithm, *Int. J. Numer. Meth. Engrg.*, **8**, 198–203, 1978.
- [768] Wilson, E. L., Automation of the finite element method — a historical view, *Finite Elem. Anal. Des.*, **13**, 91–104, 1993.
- [769] Wilson, E. L., *Three Dimensional Static and Dynamic Analysis of Structures: A Physical Approach with Emphasis on Earthquake Engineering*, Computers & Structures, Inc., 1998.
- [770] Wimp, J., *Sequence Transformations and Their Applications*, Academic Press, New York, 1981.
- [771] Wolfram, S., *The Mathematica Book*, Wolfram Media Inc., 4th ed. 1999. (Last edition in print).
- [772] Wood, A., *Acoustics*, Blackie And Sons, London, 1940. Reprinted by Dover, 1966.

- [773] Wood, R. H., *Plastic Analysis of Slabs and Plates*, Thames and Hudson, 1961.
- [774] Woodbury, M., Inverting modified matrices, Memorandum Report 42, Statistical Research Group, Princeton University, Princeton, NJ, 1950.
- [775] Wu, C. C. and Cheung, Y. K., On optimization approaches of hybrid stress elements, *Finite Elem. Anal. Des.*, **21**, 111–128, 1995.
- [776] Wunderlich, W., Stein, E., and Bathe, K. J. (eds.), *Nonlinear Finite Element Analysis in Structural Mechanics*, Springer, Berlin, 1981.

X

- [777] Xie, X. C., *Dynamic Stability of Structures*, Cambridge, 2006.

Y

- [778] Yanenko, N. N., *The Method of Fractional Steps*, Springer, Berlin, 1991.
- [779] Yang, Y.-B. and McGuire, W., A work control method for geometrically nonlinear analysis, *Proceedings NUMETA 85 Conference -Vol. 2*, A. Balkema Pubs, Rotterdam, 913–921, 1985.
- [780] Yourgrau W. and Mandelstam, S., *Variational Principles in Dynamics and Quantum Theory*, Dover, New York, 1968.

Z

- [781] Zhang, F. (ed.), *The Schur Complement and Its Applications*, Springer-Verlag, New York, 2005.
- [782] Zhechev, M. M., On the admissability of given acceleration-dependent forces in mechanics, *J. Appl. Mech.*, **74**, 107–110, 2007.
- [783] Ziegler, H., On the concept of elastic stability, in *Advances in Applied Mechanics Vol. 4*, ed. by H. L. Dryden and Th. von Karman, Academic Press, 351–403, 1956.
- [784] Zienkiewicz, O. C. and Cheung, Y. K., Finite elements in the solution of field problems, *The Engineer*, 507–510, 1965.
- [785] Zienkiewicz, O. C. and Cheung, Y. K., *The Finite Element Method in Engineering Science*, McGraw-Hill, London, 1967.
- [786] Zienkiewicz, O. C., Valliappan, S., and King, I. P., Elasto-plastic solutions of engineering problems: ‘initial stress’, finite element approach, *Int. J. Numer. Meth. Engrg.*, **1**, 75–100, 1969.
- [787] Zienkiewicz, O. C., Irons, B. M., Ergatoudis, J., Ahmad, S., and Scott, F. C., Iso-parametric and associated element families for two- and three-dimensional analysis, in *Finite Element Methods for Stress Analysis*, ed. by I. Holland and K. Bell, Tapir, Trondheim, Norway, 1969.
- [788] Zienkiewicz, O. C., Taylor, R. L., and Too, J. M., Reduced integration technique in general analysis of plates and shells, *Int. J. Numer. Meth. Engrg.*, **3**, 275–290, 1971.
- [789] Zienkiewicz, O. C., Finite elements: the background story, in *The Mathematics of Finite Elements and Applications*, ed. by J. R. Whiteman, Academic Press, New York, 1973.
- [790] Zienkiewicz, O. C., Incremental displacement in nonlinear analysis, *Int. J. Numer. Meth. Engrg.*, **3**, 587–592, 1973.
- [791] Zienkiewicz, O. C., *The Finite Element Method in Engineering Science*, 3rd ed., McGraw-Hill, 1976.
- [792] Zienkiewicz, O. C., *The Finite Element Method*, 3rd ed., McGraw-Hill, London, 1977.

Appendix R: REFERENCES (IN PROGRESS)

- [793] Zienkiewicz, O. C. and Löhner, R., Accelerated ‘relaxation’ or direct solution?, future prospects for FEM, *Int. J. Numer. Meth. Engrg.*, **21**, 1–11, 1985.
- [794] Zienkiewicz, O. C. and Taylor, R. E., *The Finite Element Method*, 4th ed., McGraw-Hill, London, Vol. I: 1988, Vol II: 1993.
- [795] Zienkiewicz, O. C., preface to reprint of B. M. Fraeijs de Veubeke’s “Displacement and equilibrium models” in *Int. J. Numer. Meth. Engrg.*, **52**, 287–342, 2001. Reprint available from <http://www3.interscience.wiley.com/cgi-bin/fulltext/85006363/PDFSTART>
- [796] Ziman, J. M., *Principles of the Theory of Solids*, North-Holland, Amsterdam, 1967.

===== Additional refs to be eventually merged w/above, and dups weeded =====

S

Spatial Applications of Matrices

TABLE OF CONTENTS

		Page
§S.1.	Points, Planes	S-3
§S.2.	Lines	S-3

In this Appendix we summarize some geometric applications of matrices in 3D space. Indexed homogeneous Cartesian coordinates¹ $\{x_0, x_1, x_2, x_3\}$ are used. To pass to physical coordinates, divide x_1, x_2 and x_3 by x_0 : $\{x_1/x_0, x_2/x_0, x_3/x_0\}$. If $x_0 = 0$ the physical coordinates are at infinity.

§S.1. Points, Planes

Points in 3D space will be identified by X, Y, P, Q , etc. Their coordinates are put in the 4-vectors

$$\mathbf{x} = [x_0 \ x_1 \ x_2 \ x_3]^T, \quad \mathbf{y} = [y_0 \ y_1 \ y_2 \ y_3]^T, \quad \mathbf{p} = [p_0 \ p_1 \ p_2 \ p_3]^T, \text{ etc} \quad (\text{S.1})$$

Planes in 3D space will be identified by A, B, C , etc. The equation of plane A is written $a_0x_0 + a_1x_1 + a_2x_2 + a_3x_3 = 0$ or

$$\mathbf{a}^T \mathbf{x} = 0, \quad \text{or} \quad \mathbf{x}^T \mathbf{a} = 0 \quad (\text{S.2})$$

The plane A is thus defined by the 4-vector \mathbf{a} .

Example S.1. Find the coordinates of the point where line joining points P and Q intersects plane A .

Solution. Any point of PQ is R where

$$\mathbf{r} = \lambda \mathbf{p} + \mu \mathbf{q} \quad (\text{S.3})$$

If R is on plane A , then $\mathbf{a}^T \mathbf{x} = 0$ so that $\lambda \mathbf{a}^T \mathbf{p} + \mu \mathbf{a}^T \mathbf{q} = 0$. Absorbing a suitable multiplier into the coordinates of R we obtain its vector in the form

$$\mathbf{r} = (\mathbf{a}^T \mathbf{q}) \mathbf{p} - (\mathbf{a}^T \mathbf{p}) \mathbf{q}. \quad (\text{S.4})$$

§S.2. Lines

Let \mathbf{x} and \mathbf{y} be the coordinates of points X and Y on a given line L . The 4×4 antisymmetric matrix

$$\mathbf{L} = \mathbf{xy}^T - \mathbf{yx}^T \quad (\text{S.5})$$

is called the *coordinate matrix* of the line. It can be shown that \mathbf{L} determines the line L to within a scale factor.

Let A and B be two planes through L . The 4×4 antisymmetric matrix

$$\mathbf{L}^* = \mathbf{ab}^T - \mathbf{ba}^T \quad (\text{S.6})$$

is called the *dual coordinate matrix* of line L . It can be shown that²

$$\mathbf{L}^* \mathbf{L} = \mathbf{0}. \quad (\text{S.7})$$

¹ These coordinates were independently invented in 1827 by Møbius and Feuerbach, and further developed in 1946 by E. A. Maxwell at Cambridge. See E. A. Maxwell, *General Homogeneous Coordinates in Space of Three Dimensions*, Cambridge University Press, 1951.

² See E. A. Maxwell, loc. cit., page 150.

Example S.2. Find where line L defined by two points X and Y meets a plane A .

Solution. Consider the vector

$$\mathbf{p} = \mathbf{L}\mathbf{a} = (\mathbf{xy}^T - \mathbf{yx}^T)\mathbf{a} = (\mathbf{y}^T\mathbf{a})\mathbf{x} - (\mathbf{x}^T\mathbf{a})\mathbf{y} \quad (\text{S.8})$$

From the last form the point P of coordinates \mathbf{p} must lie on the line that joins X and Y . Moreover since \mathbf{L} is antisymmetrical, $\mathbf{a}^T\mathbf{L}\mathbf{a} = 0$ so that $\mathbf{a}^T\mathbf{p} = 0$. Thus P is the intersection of line L and plane A .

Example S.3. Find the plane A that joins line L to a point X .

Solution.

$$\mathbf{a} = \mathbf{L}^* \mathbf{x} \quad (\text{S.9})$$

where \mathbf{L}^* is the dual of \mathbf{L} . The demonstration is trivial.

Two immediate corollaries: (i) Line L lies in the plane A if $\mathbf{L}\mathbf{a} = \mathbf{0}$; (ii) Line L passes through the point X if $\mathbf{L}^*\mathbf{x} = \mathbf{0}$.

Example S.4.

Consider two lines L_1 and L_2 with coordinate matrices $\mathbf{L}_1, \mathbf{L}_2$ and dual matrices $\tilde{\mathbf{L}}_1$ and $\tilde{\mathbf{L}}_2$, respectively. Find the conditions for the lines to intersect.

Solution. Any of the four equivalent conditions

$$\mathbf{L}_1\mathbf{L}_2^*\mathbf{L}_1 = \mathbf{0}, \quad \mathbf{L}_1^*\mathbf{L}_2\mathbf{L}_1^* = \mathbf{0}, \quad \mathbf{L}_2\mathbf{L}_1^*\mathbf{L}_2 = \mathbf{0}, \quad \mathbf{L}_2^*\mathbf{L}_1\mathbf{L}_2^* = \mathbf{0}. \quad (\text{S.10})$$

For the proof see E. A. Maxwell, loc. cit, page 154.