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 update Fall 2001

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. Selective 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 "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 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 books 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 three Parts of 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 a 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 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 21 through 28 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.

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 lenghty 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 lessen 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 belong 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.

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

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

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

1
Overview

TABLE OF CONTENTS

This book is an introduction to the analysis of linear elastic structures by the Finite Element Method (FEM). It embodies three Parts:

I **Finite Element Discretization: Chapters 2-11**. This part provides an introduction to the discretization and analysis of skeletal structures by the Direct Stiffness Method.

II **Formulation of Finite Elements: Chapters 12-20**. This part presents the formulation of assumed-displacement isoparametric elements in one and two dimensions.

III **Computer Implementation of FEM: Chapters 21-28**. This part uses *Mathematica* as the implementation language.

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

§1.1. WHERE THIS MATERIAL FITS

The field of Mechanics can be subdivided into three major areas:

$$\text{Mechanics} \begin{cases} \textit{Theoretical} \\ \textit{Applied} \\ \textit{Computational} \end{cases} \tag{1.1}$$

Theoretical mechanics deals with fundamental laws and principles of mechanics 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 simulation through numerical methods implemented on digital computers.

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.

§1.1.1. Computational Mechanics

Several branches of computational mechanics can be distinguished according to the *physical scale* of the focus of attention:

$$\text{Computational Mechanics} \begin{cases} \textit{Nanomechanics and micromechanics} \\ \textit{Continuum mechanics} \begin{cases} \text{Solids and Structures} \\ \text{Fluids} \\ \text{Multiphysics} \end{cases} \\ \textit{Systems} \end{cases} \tag{1.2}$$

Nanomechanics deals with phenomena at the molecular and atomic levels of matter. As such it is closely interrelated with particle physics and chemistry. 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 averages. The two traditional areas of application are solid and fluid mechanics. The former includes *structures* which, for obvious reasons, are fabricated with solids. Computational solid mechanics takes a 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 areas are hydrodynamics, aerodynamics, atmospheric physics, and combustion.

Multiphysics is a more recent newcomer. This area is meant to include mechanical systems that transcend the classical boundaries of solid and fluid mechanics, as in interacting fluids and structures. 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, buildings, bridges, engines, cars, microchips, radio telescopes, robots, roller skates and garden sprinklers. Biological systems, such as a whale, amoeba or pine tree are included if studied from the viewpoint of biomechanics. Ecological, astronomical and cosmological entities also form systems.[1]

In this progression of (1.2) the *system* is the most general concept. A system is 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. Consequently there is a tradeoff art in deciding where to stop.[2]

§1.1.2. Statics vs. Dynamics

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

$$\text{Continuum mechanics} \begin{cases} \textit{Statics} \\ \textit{Dynamics} \end{cases} \tag{1.3}$$

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.

Problems in statics may also be time dependent but the inertial forces are ignored or neglected. Static problems may be classified into strictly static and quasi-static. For the former time need not be considered explicitly; any historical time-like response-ordering parameter (if one is needed) will do. In quasi-static problems such as foundation settlement, creep deformation, rate-dependent

[1] Except that their function may not be clear to us. "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).

[2] Thus in breaking down a car engine, say, the decomposition does not usually proceed beyond the components you can buy at a parts shop.

plasticity or fatigue cycling, a more realistic estimation of time is required but inertial forces are still neglected.

§1.1.3. Linear vs. Nonlinear

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

$$\text{Statics}\begin{cases}\textit{Linear}\\ \textit{Nonlinear}\end{cases}$$

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.

§1.1.4. Discretization methods

A final classification of CSM 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:

$$\text{Spatial discretization method}\begin{cases}\textit{Finite Element Method (FEM)}\\ \textit{Boundary Element Method (BEM)}\\ \textit{Finite Difference Method (FDM)}\\ \textit{Finite Volume Method (FVM)}\\ \textit{Spectral Method}\\ \textit{Mesh-Free Method}\end{cases}\tag{1.4}$$

For *linear* problems finite element methods currently dominate the scene, with boundary element methods posting a strong second choice in specific 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 dominant. Finite-volume methods, which address finite volume method conservation laws, are important in highly nonlinear problems of fluid mechanics. Spectral methods are based on transforms that map space and/or time dimensions to spaces where the problem is easier to solve.

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

§1.1.5. FEM Variants

The term *Finite Element Method* actually identifies a broad spectrum of techniques that share common features outlined in §1.3 and §1.4. Two subclassifications that fit well applications to structural mechanics are

$$\text{FEM Formulation}\begin{cases}\textit{Displacement}\\ \textit{Equilibrium}\\ \textit{Mixed}\\ \textit{Hybrid}\end{cases}\qquad \text{FEM Solution}\begin{cases}\textit{Stiffness}\\ \textit{Flexibility}\\ \textit{Mixed (a.k.a. Combined)}\end{cases}\tag{1.5}$$

(The distinction between these subclasses require advanced technical concepts, and will not be covered here.)

Using the foregoing classification, we can state the topic of this book more precisely: the *computational analysis of linear static structural problems* by the Finite Element Method. Of the variants listed in (1.5), emphasis is placed on the *displacement* formulation and *stiffness* solution. This combination is called the *Direct Stiffness Method* or DSM.

§1.2. WHAT DOES A FINITE ELEMENT LOOK LIKE?

The subject of this book is FEM. But what is a finite element? The 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 the numerical value of π.

Draw a circle of radius r and diameter $d = 2r$ as in Figure 1.1(a). Inscribe a regular polygon of n sides, where $n = 8$ in Figure 1.1(b). Rename polygon sides as *elements* and vertices as *nodal points* or *nodes*. Label nodes with integers 1, . . . 8. Extract a typical element, say that joining nodes 4–5 as shown in Figure 1.1(c). This is an instance of the *generic element* i–j shown in Figure 1.1(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 $\pi_n = L_n/d = n \sin(\pi/n)$.

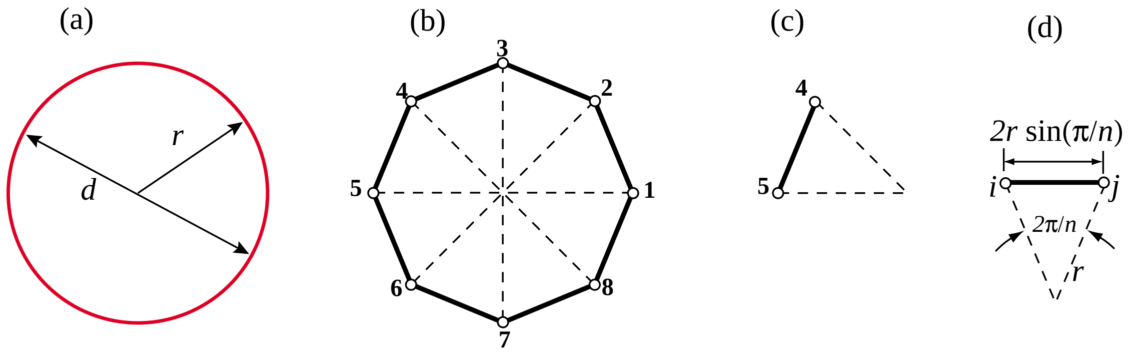


Figure 1.1. The "find π" problem treated with FEM concepts: (a) continuum object, (b) a discrete approximation (inscribed regular polygon), (c) disconnected element, (d), generic element.

Values of π_n obtained for $n = 1, 2, 4, \ldots$ 256 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 algorithm[3] into that shown in the third column. The last value displays 15-place accuracy.

Some of the key ideas behind the FEM can be identified in this simple 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 the nodes, a process called *disassembly* in the FEM. Upon disassembly

[3] A widely used extrapolation algorithm that speeds up the convergence of many sequences. See, e.g, J. Wimp, *Sequence Transformations and Their Applications*, Academic Press, New York, 1981.

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	0.000000000000000		
2	2.000000000000000		
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

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: length L_{ij}, can be computed in the generic element independently of the others, a property called *local support* in the FEM. Finally, the desired property: the 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 be used to rectify any smooth plane curve.[4]

This example has been offered in the FEM literature 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 Chapters 2–3, 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 $\pi = \lim_{n\to\infty} n\sin(\pi/n)$ uses the unknown π in the right hand side.[5] 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 in the next section.

§1.3. THE FEM ANALYSIS PROCESS

A model-based simulation process using FEM involves doing a sequence of steps. This sequence takes two canonical configurations depending on the environment in which FEM is used. These are reviewed next to introduce terminology.

[4] A similar limit process, however, may fail in three or more dimensions.

[5] This objection is bypassed if n is advanced as a power of two, as in Table 1.1, by using the half-angle recursion $\sqrt{2}\sin\alpha = \sqrt{1-\sqrt{1-\sin^2 2\alpha}}$, started from $2\alpha = \pi$ for which $\sin\pi = -1$.

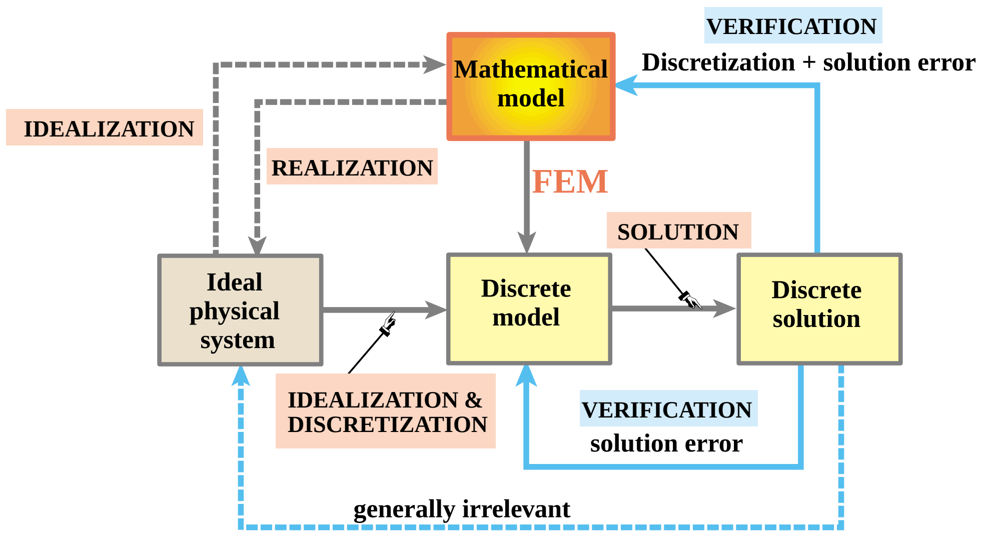


Figure 1.2. The Mathematical FEM. The mathematical model (top) is the source of the simulation process. Discrete model and solution follow from it. The ideal physical system (should one go to the trouble of exhibiting it) is inessential.

§1.3.1. The Mathematical FEM

The process steps are illustrated in Figure 1.2. The process centerpiece, from which everything emanates, is the *mathematical model*. This is often an ordinary or partial differential equation in space and time. A discrete finite element model is generated from a variational or weak form of the mathematical model.[6] This is the *discretization* step. The FEM equations are processed by an equation solver, which delivers a discrete solution (or solutions).

On the left Figure 1.2 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. For example, if the mathematical model is the Poisson's equation, realizations may be a heat conduction or a electrostatic charge distribution problem. This step is inessential and may be left out. Indeed 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*. 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.[7] Replacing into the ideal physical system would in principle quantify modeling errors. In the mathematical FEM this is largely irrelevant, however, since the ideal physical system is merely that: a figment of the imagination.

[6] The distinction between strong, weak and variational forms is discussed in advanced FEM courses. In the present course such forms will be stated as recipes.

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

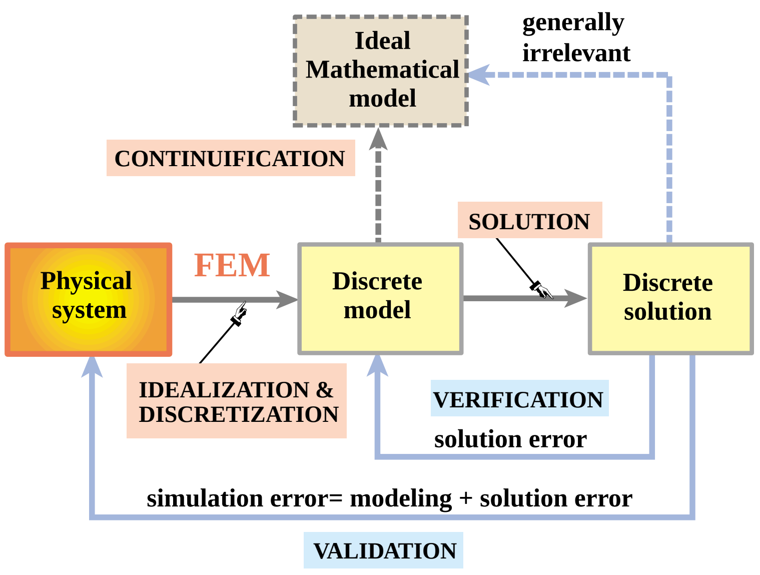


Figure 1.3. The Physical FEM. The physical system (left) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

§1.3.2. The Physical FEM

The second way of using FEM is the process illustrated in Figure 1.3. The centerpiece is now the *physical system* to be modeled. Accordingly, this sequence is called the *Physical FEM*. The processes of idealization and discretization are carried out *concurrently* to produce the discrete model. The solution is computed as before.

Just like Figure 1.2 shows an ideal physical system, 1.3 depicts 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, notably complex engineering systems, it makes no sense. Indeed FEM discretizations may be constructed and adjusted without reference to mathematical models, simply from experimental measurements.

The concept of *error* arises in the Physical FEM in two ways, known as *verification* and *validation*, respectively. Verification is the same as in the Mathematical FEM: the discrete solution is replaced into the discrete model to get the solution error. As noted above this error is not generally important. Substitution in the ideal mathematical model in principle provides the discretization error. This is rarely useful in complex engineering systems, however, because there is no reason to expect that the mathematical model exists, and 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. Since the latter is typically insignificant, the simulation error in practice can be identified with the modeling error.

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. Since the minimization conditions are generally nonlinear (even if the model is linear) the updating process is inherently iterative.

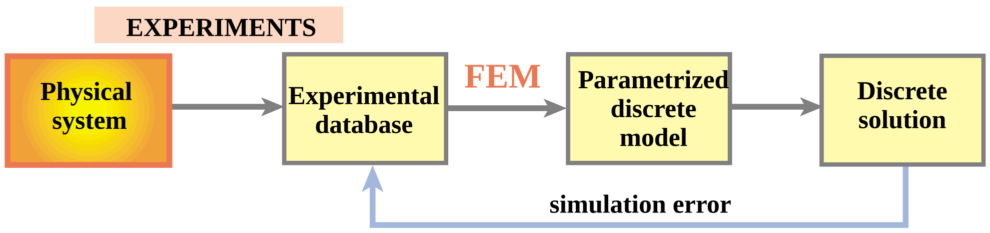


Figure 1.4. Model updating process in the Physical FEM.

§1.3.3. Synergy of Physical and Mathematical FEM

The foregoing physical and mathematical sequences are not exclusive but complementary. This synergy[8] is one of the reasons behind the power and acceptance of the method. Historically the Physical FEM was the first one to be developed to model very complex systems such as aircraft, as narrated in Appendix H. 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 differential equation that captures, at a continuum mechanics level,[9] 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 and view the rest as masses and forces, most of which are time-varying and nondeterministic. Second, consider the aircraft structure as built of *substructures*:[10] wings, fuselage, stabilizers, engines, landing gears, and so on. Take each substructure, and continue to decompose it into *components*: rings, ribs, spars, cover plates, actuators, etc, continuing 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 continuum 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 assembly* process is governed by the classical principles of Newtonian mechanics expressed in conservation form.

This multilevel decomposition process is diagramed in Figure 1.5, in which the intermediate substructure level is omitted for simplicity.

---

[8] This 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, 1620).

[9] Of course at the atomic and subatomic level quantum mechanics works for everything, from landing gears to passengers. But it would be slightly impractical to model the aircraft by 10^{36} interacting particles.

[10] A *substructure* is a part of a structure devoted to a specific function.

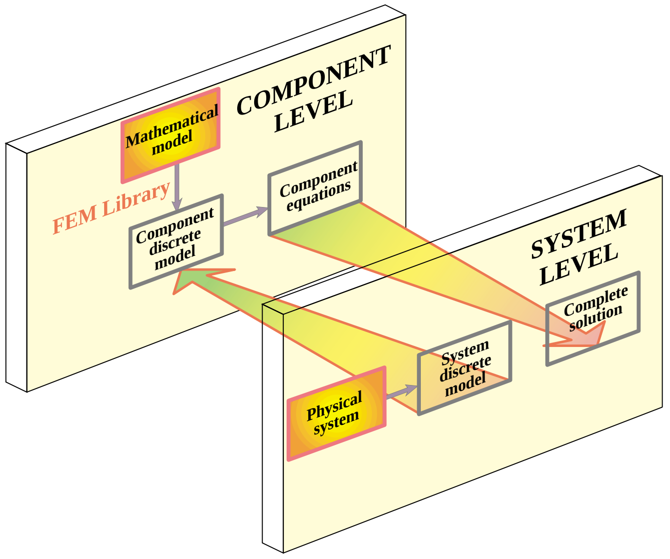


Figure 1.5. Combining physical and mathematical modeling through multilevel FEM. Only two levels (system and component) are shown for simplicity; intermediate substructure levels are omitted.

REMARK 1.2

More intermediate decomposition levels are used in some systems, such as offshore and ship structures, which are characterized by a modular fabrication process. In that case the decomposition mimics the way the system is actually constructed. The general technique, called *superelements*, is discussed in Chapter 11.

REMARK 1.3

There is no point in practice in going beyond a certain component level while considering the complete model, since the level of detail can become overwhelming without adding significant information. Further refinement or particular components is done by the so-called global-local analysis technique outlined in Chapter 11. This technique is an instance of multiscale analysis.

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 1.6. Multiple levels are unnecessary here. Of course the truss may be viewed as a substructure of the roof, and the roof as a a substructure of a building.

§1.4. INTERPRETATIONS OF THE FINITE ELEMENT METHOD

Just like there are two complementary ways of using the FEM, there are two complementary interpretations for teaching it. 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.

§1.4.1. Physical Interpretation

The physical interpretation focuses on the view of Figure 1.3. This interpretation has been shaped by the discovery and extensive use of the method in the field of structural mechanics. This relationship

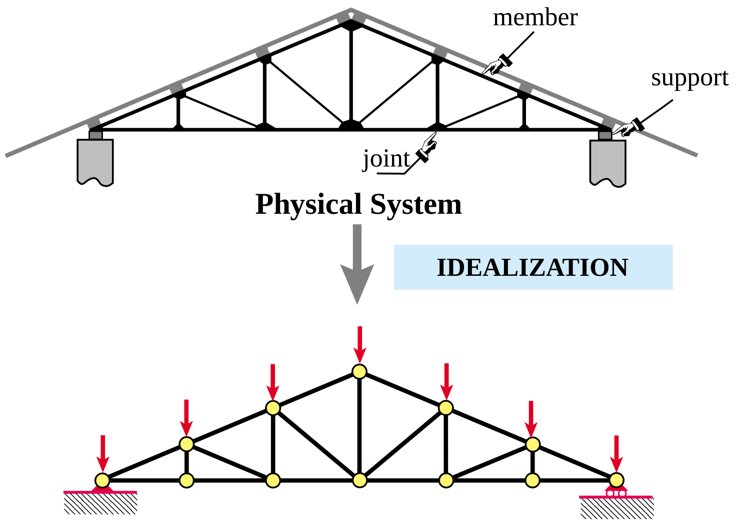


Figure 1.6. The idealization process for a simple structure. The physical system, here a roof truss, is directly idealized by the mathematical model: a pin-jointed bar assembly. For this particular structure, the idealization coalesces with the discrete model.

is reflected in the use of structural terms such as “stiffness matrix”, “force vector” and “degrees of freedom.” This terminology 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 §1.3, 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 can be physically interpreted and understood in terms that were eminently familiar to structural engineers.

§1.4.2. Mathematical Interpretation

This interpretation is closely aligned with the configuration of Figure 1.2. 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 thermal 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. 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.

§1.5. KEEPING THE COURSE

The first Part of this course, which is the subject of Chapters 2 through 11, stresses the physical interpretation in the framework of the Direct Stiffness Method (DSM) on account of its instructional advantages. Furthermore the computer implementation becomes more transparent because the sequence of computer operations can be placed in close correspondence with the DSM steps.

Subsequent Chapters incorporate ingredients of the mathematical interpretation when it is felt convenient to do so. However, the exposition avoids excessive entanglement with the mathematical theory when it may obfuscate the physics.

A historical outline of the evolution of Matrix Structural Analysis into the Finite Element Method is given in Appendix H, which provides appropriate references.

In Chapters 2 through 6 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 his thesis work on FEM under Ray Clough. Virtually all finite element codes are now based on the DSM and the computer implementation has not essentially changed since the late 1960s.

§1.6. *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 of solution.
3. Kirchhoff-based plate and shell elements.
4. Continuum-based plate and shell elements.
5. Variational methods in mechanics.
6. General mathematical theory of finite elements.
7. Vibration analysis.
8. Buckling analysis.
9. General stability analysis.
10. General nonlinear response analysis.
11. Structural optimization.
12. Error estimates and problem-adaptive discretizations.
13. Non-structural and coupled-system applications of FEM.
14. Structural dynamics.
15. Shock and wave-propagation dynamics.
16. Designing and building production-level FEM software and use of special hardware (*e.g.* vector and parallel computers)

Topics 1–7 pertain to what may be called "Advanced Linear FEM", whereas 9–11 pertain to "Nonlinear FEM". Topics 12-15 pertain to advanced applications, whereas 16 is an interdisciplinary topic that interweaves with computer science.

For pre-1990 books on FEM see Appendix G: Oldies but Goodies.

Homework Exercises for Chapter 1
Overview

EXERCISE 1.1

[A:15] Do Archimedes' problem using a circumscribed regular polygon, with $n = 1, 2, \ldots 256$. 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.

2

The Direct Stiffness Method: Breakdown

TABLE OF CONTENTS

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.

§2.1. WHY A PLANE TRUSS?

The simplest structural finite element is the bar (also called linear spring) element, which is illustrated in Figure 2.1(a). Perhaps the most complicated finite element (at least as regards number of degrees of freedom) is the curved, three-dimensional "brick" element depicted in Figure 2.1(b).

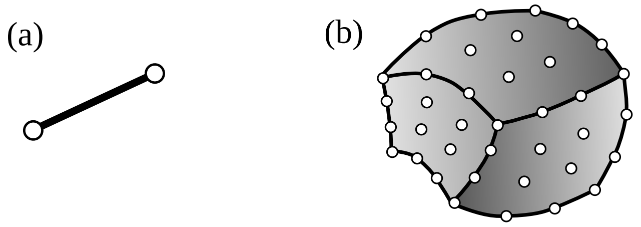


Figure 2.1. From the simplest to a highly complex structural finite element: (a) 2-node bar element for trusses, (b) 64-node tricubic, curved "brick" element for three-dimensional solid analysis.

Yet the remarkable fact is that, in the DSM, the simplest and most complex elements 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. A simple yet nontrivial structure is the *pin-jointed plane truss*.[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 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.

§2.2. 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

[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.

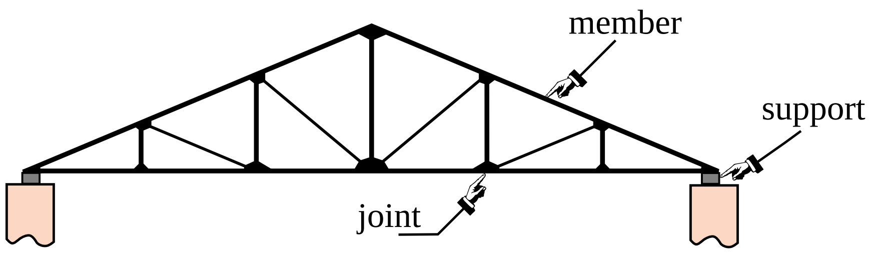


Figure 2.2. An actual plane truss structure. That shown is typical of a roof truss used in residential building construction.

of skeletal structures are frame structures or *frameworks*, which are common in reinforced concrete construction of building 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.

In this and the 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 which 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 5.

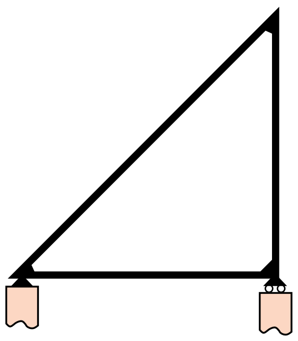


Figure 2.3. The example plane truss structure, called "example truss" in the sequel. It has three members and three joints.

To keep hand computations manageable in detail we use just about the simplest structure that can be called a plane truss, namely the three-member truss illustrated in Figure 2.3. The *idealized* model of the example truss as a pin-jointed assemblage of bars is shown in Figure 2.4(a), which also gives its geometric and material properties. In this idealization truss members carry only axial loads, have no bending resistance, and are connected by frictionless pins. Figure 2.4(b) displays support conditions as well as the applied forces applied to the truss joints.

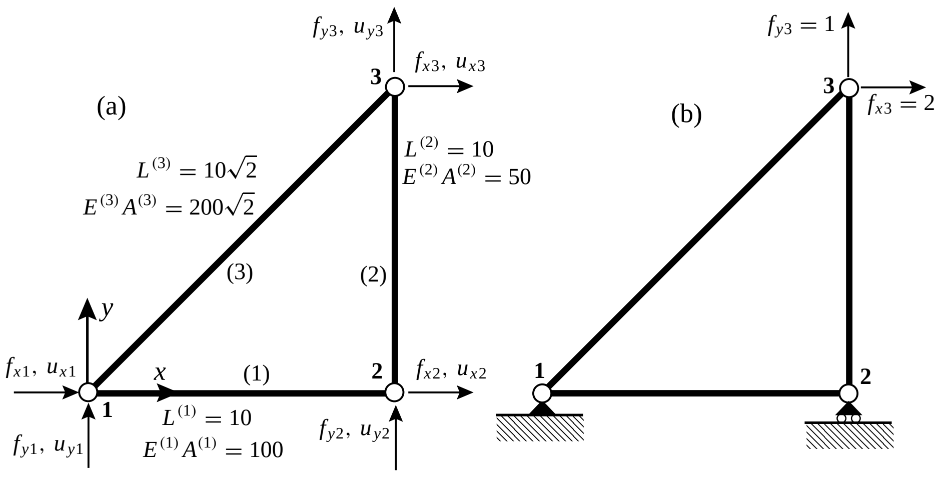


Figure 2.4. Pin-jointed idealization of example truss: (a) geometric and elastic properties, (b) support conditions and applied loads.

It should be noted that as a practical structure the example truss is not particularly useful — the one depicted in Figure 2.2 is far 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.

§2.3. IDEALIZATION

Although the pin-jointed assemblage of bars (as depicted in Figure 2.4) is sometimes presented as a real problem, it actually represents an *idealization* of a true truss structure. The axially-carrying members and frictionless pins of this structure are only an approximation of a real truss. 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; see Figure 2.2. Consequently members will carry some bending as well as direct axial loading.

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

This replacement of true by idealized is at the core of the *physical interpretation* of the finite element method discussed in §1.4.

§2.4. JOINT FORCES AND DISPLACEMENTS

The example truss shown in Figure 2.3 has three *joints*, which are labeled 1, 2 and 3, and three *members*, which are labeled (1), (2) and (3). These 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 $A^{(3)}$. 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, see below), which is usually enclosed in parentheses to avoid confusion with exponents. For example, the cross-section area of a generic member is $A^{(e)}$. Joints are generically identified by indices such as i, j or n. In the general FEM, the name "joint" and "member" is replaced by *node* and *element*, respectively. The 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 is called the *global coordinate system*. Other names for it in the literature are *structure coordinate system* and *overall coordinate system*.

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, which we call f_x and f_y, respectively. The components at joint i will be denoted as f_{xi} and f_{yi}, respectively. The set of all joint forces can be arranged as a 6-component column vector:

$$\mathbf{f} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix}. \tag{2.1}$$

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} and, like the joint forces, they are arranged into a 6-component vector:

$$\mathbf{u} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \tag{2.2}$$

In the DSM these six displacements are the primary unknowns. They are also called the *degrees of freedom* or *state variables* of the system.[2]

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

[2] *Primary unknowns* is the correct mathematical term whereas *degrees of freedom* has a mechanics flavor. The term *state variables* is used more often in nonlinear analysis.

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 for the hand computations.

§2.5. THE MASTER STIFFNESS EQUATIONS

The *master stiffness equations* relate the joint forces **f** of the complete structure to the joint displacements **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.[3]

If both assumptions hold the relation must be homogeneous and be expressable in component form as follows:

$$\begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} K_{x1x1} & K_{x1y1} & K_{x1x2} & K_{x1y2} & K_{x1x3} & K_{x1y3} \\ K_{y1x1} & K_{y1y1} & K_{y1x2} & K_{y1y2} & K_{y1x3} & K_{y1y3} \\ K_{x2x1} & K_{x2y1} & K_{x2x2} & K_{x2y2} & K_{x2x3} & K_{x2y3} \\ K_{y2x1} & K_{y2y1} & K_{y2x2} & K_{y2y2} & K_{y2x3} & K_{y2y3} \\ K_{x3x1} & K_{x3y1} & K_{x3x2} & K_{x3y2} & K_{x3x3} & K_{x3y3} \\ K_{y3x1} & K_{y3y1} & K_{y3x2} & K_{y3y2} & K_{y3x3} & K_{y3y3} \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \tag{2.3}$$

In matrix notation:

$$\mathbf{f} = \mathbf{K}\mathbf{u}. \tag{2.4}$$

Here **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 (2.3). 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 2.1

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

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \qquad \mathbf{f} = \begin{bmatrix} K_{x1x2} \\ K_{y1x2} \\ K_{x2x2} \\ K_{y2x2} \\ K_{x3x2} \\ K_{y3x2} \end{bmatrix}. \tag{2.5}$$

[3] 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 4.

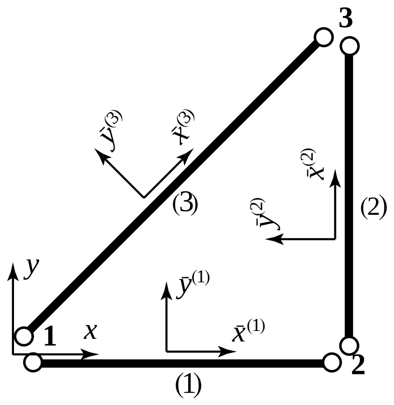


Figure 2.5. Breakdown of example truss into individual members (1), (2) and (3), and selection of local coordinate systems.

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 the property just noted is called *interpretation of stiffness coefficients as displacement influence coefficients,* and extends unchanged to the general finite element method.

§2.6. BREAKDOWN

The first three DSM steps are: (1) disconnection, (2) localization, and (3) computation of member stiffness equations. These are collectively called *breakdown steps* and are described below.

§2.6.1. Disconnection

To carry out the first step of the DSM we proceed to *disconnect* or *disassemble* the structure into its components, namely the three truss members. This step is illustrated in Figure 2.5.

To each member $e = 1, 2, 3$ is assigned a Cartesian system $\{\bar{x}^{(e)}, \bar{y}^{(e)}\}$. Axis $\bar{x}^{(e)}$ is aligned along the axis of the e^{th} member. See Figure 2.5. Actually $\bar{x}^{(e)}$ runs along the member longitudinal axis; it is shown offset in that Figure 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 called $\varphi^{(e)}$. The axes origin is arbitrary and may be placed at the member midpoint or at one of the end joints for convenience.

These systems are called *local coordinate systems* or *member-attached coordinate systems*. In the general finite element method they receive the name *element coordinate systems*.

§2.6.2. Localization

Next, we drop the member identifier (e) so that we are effectively dealing with a *generic* truss member as illustrated in Figure 2.6. The local coordinate system is $\{\bar{x}, \bar{y}\}$. The two end joints are called i and j.

As shown in Figure 2.6, a generic truss member has four joint force components and four joint displacement components (the member degrees of freedom). The member properties include the length L, elastic modulus E and cross-section area A.

§2.6.3. Computation of Member Stiffness Equations

The force and displacement components of Figure 2.7(a) are linked by the *member stiffness relations*

$$\bar{\mathbf{f}} = \bar{\mathbf{K}}\,\bar{\mathbf{u}}, \tag{2.6}$$

which written out in full is

$$\begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} \bar{K}_{xixi} & \bar{K}_{xiyi} & \bar{K}_{xixj} & \bar{K}_{xiyj} \\ \bar{K}_{yixi} & \bar{K}_{yiyi} & \bar{K}_{yixj} & \bar{K}_{yiyj} \\ \bar{K}_{xjxi} & \bar{K}_{xjyi} & \bar{K}_{xjxj} & \bar{K}_{xjyj} \\ \bar{K}_{yjxi} & \bar{K}_{yjyi} & \bar{K}_{yjxj} & \bar{K}_{yjyj} \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix}. \tag{2.7}$$

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 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 the element properties 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 2.6(a) as a linear spring of equivalent stiffness k_s, an interpretation depicted in Figure 2.7(b). If the member properties are *uniform* along its length, Mechanics of Materials bar theory tells us that[4]

$$k_s = \frac{EA}{L}, \tag{2.8}$$

Consequently the force-displacement equation is

$$F = k_s d = \frac{EA}{L} d, \tag{2.9}$$

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

$$F = \bar{f}_{xj} = -\bar{f}_{xi}, \qquad d = \bar{u}_{xj} - \bar{u}_{xi}, \tag{2.10}$$

which express force equilibrium and kinematic compatibility, respectively.

Combining (2.9) and (2.10) we obtain the matrix relation[5]

$$\bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \frac{EA}{L} \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix} = \bar{\mathbf{K}}\,\bar{\mathbf{u}}, \tag{2.11}$$

[4] See for example, Chapter 2 of F. P. Beer and E. R. Johnston, *Mechanics of Materials*, McGraw-Hill, 2nd ed. 1992.

[5] The detailed derivation of (2.11) is the subject of Exercise 2.3.

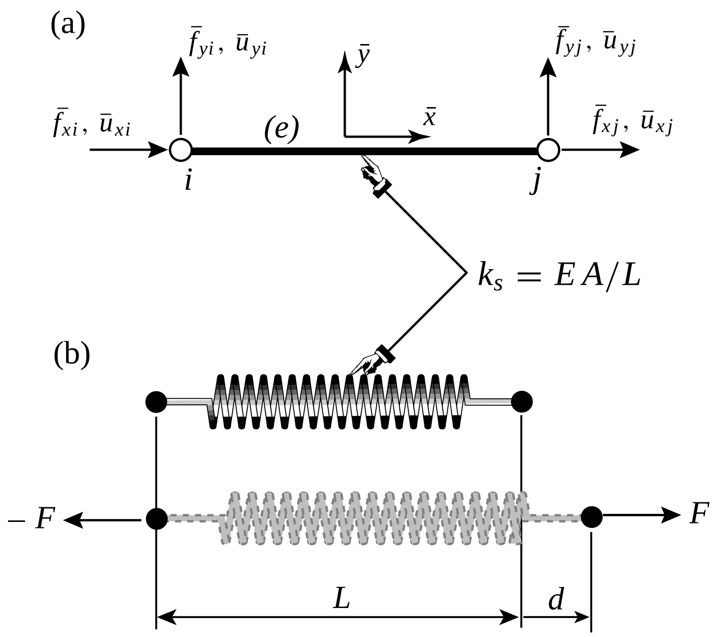

(a)
$\bar f_{yi},\ \bar u_{yi}$
$\bar y$
$\bar f_{yj},\ \bar u_{yj}$
$\bar f_{xi},\ \bar u_{xi}$
(e)
$\bar x$
$\bar f_{xj},\ \bar u_{xj}$
i
j
$k_s = EA/L$
(b)
$-F$
F
L
d

Figure 2.6. Generic truss member referred to its local coordinate system $\{\bar x, \bar y\}$: (a) idealization as bar element, (b) interpretation as equivalent spring.

Hence

$$\overline{\mathbf{K}} = \frac{EA}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{2.12}$$

This is the truss stiffness matrix in local coordinates.

Two other methods for obtaining the local force-displacement relation (2.9) are covered in Exercises 2.6 and 2.7.

In the following Chapter we will complete the main DSM steps by putting the truss back together and solving for the unknown forces and displacements.

Homework Exercises for Chapter 2
The Direct Stiffness Method: Breakdown

EXERCISE 2.1

[D:5] 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: think of translational rigid body modes.] Does the property hold also for the columns of that matrix?

EXERCISE 2.3

[A:15] Using matrix algebra derive (2.11) from (2.9) and (2.10).

EXERCISE 2.4

[A:15] By direct matrix multiplication verify that for the generic truss member $\bar{\mathbf{f}}^T\,\bar{\mathbf{u}} = F\,d$. Can you interpret this result physically? (Interpretation hint: look at (E2.3) below]

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

$$d = \mathbf{T}_d\,\bar{\mathbf{u}}, \qquad \bar{\mathbf{f}} = F\,\mathbf{T}_f, \tag{E2.1}$$

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 is a general result.)

EXERCISE 2.6

[A:20] Derive the equivalent spring formula $F = (EA/L)\,d$ of (2.9) 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

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

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 (2.9) 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

$$\Pi = \tfrac{1}{2}\int_0^L A\,\sigma\,e\;d\bar{x} - Fd \tag{E2.3}$$

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 $\sigma = Ae$ law to express Π as a function $\Pi(d)$ of the relative displacement d only. Then apply MPE by requiring that $\partial\Pi/\partial d = 0$.

3

The Direct Stiffness Method: Assembly and Solution

TABLE OF CONTENTS

§3.1. INTRODUCTION

Chapter 2 explained the breakdown of a truss structure into components called *members* or *elements*. Upon deriving the stiffness relations at the element level in terms of the local coordinate system, we are now ready to go back up to the original structure. This process is called *assembly*.

Assembly involves two substeps: *globalization*, through which the member stiffness equations are transformed back to the global coordinate system, and *merging* of those equations into the global stiffness equations. On the computer these 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 also embodies two substeps: *application of displacement boundary conditions* and *solution* for the unknown joint displacements. First, the free-free master stiffness equations are modified by taking into account which components of the joint displacements and forces are given and which are unknown. Then the modified equations are submitted to a linear equation solver, which returns the unknown joint displacements. On some equation solvers, both operations are done concurrently.

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

§3.2. ASSEMBLY

§3.2.1. Coordinate Transformations

Before describing the globalization step, we must establish matrix relations that connect joint displacements and forces in the global and local coordinate systems. The necessary transformations are easily obtained by inspection of Figure 3.1. For the displacements

$$\begin{aligned} \bar u_{xi} &= u_{xi} c + u_{yi} s, \qquad & \bar u_{yi} &= -u_{xi} s + u_{yi} c, \\ \bar u_{xj} &= u_{xj} c + u_{yj} s, \qquad & \bar u_{yj} &= -u_{xj} s + u_{yj} c, \end{aligned} \tag{3.1}$$

where $c = \cos\varphi$, $s = \sin\varphi$ and φ is the angle formed by $\bar x$ and x, measured positive counterclockwise from x. The matrix form of this relation is

$$\begin{bmatrix} \bar u_{xi} \\ \bar u_{yi} \\ \bar u_{xj} \\ \bar u_{yj} \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 \\ -s & c & 0 & 0 \\ 0 & 0 & c & s \\ 0 & 0 & -s & c \end{bmatrix} \begin{bmatrix} u_{xi} \\ u_{yi} \\ u_{xj} \\ u_{yj} \end{bmatrix}. \tag{3.2}$$

The 4×4 matrix that appears above is called a *displacement transformation matrix* and is denoted[1] by $\mathbf{T}$. The node forces transform as $f_{xi} = \bar f_{xi} c - \bar f_{yi} s$, etc., which in matrix form become

$$\begin{bmatrix} f_{xi} \\ f_{yi} \\ f_{xj} \\ f_{yj} \end{bmatrix} = \begin{bmatrix} c & -s & 0 & 0 \\ s & c & 0 & 0 \\ 0 & 0 & c & -s \\ 0 & 0 & s & c \end{bmatrix} \begin{bmatrix} \bar f_{xi} \\ \bar f_{yi} \\ \bar f_{xj} \\ \bar f_{yj} \end{bmatrix}. \tag{3.3}$$

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

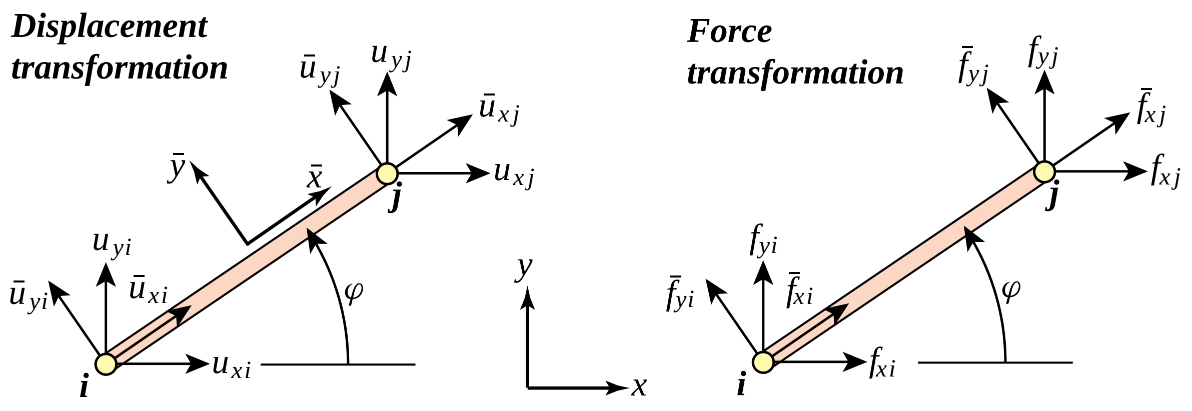


Figure 3.1. The transformation of node displacement and force components from the local system $\{\bar x, \bar y\}$ to the global system $\{x, y\}$.

The 4×4 matrix that appears above is called a *force transformation matrix*. A comparison of (3.2) and (3.3) 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.[2]

REMARK 3.1

Note that in (3.2) the local-system (barred) quantities appear on the left hand side, whereas in (3.3) they appear on the right-hand side. The expressions (3.2) and (3.3) are discrete counterparts of what are called covariant and contravariant transformations, respectively, in continuum mechanics. The counterpart of the transposition relation is the *adjointness* property.

REMARK 3.2

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 does not possess an ordinary inverse. However the congruential transformation relations (3.4)-(3.6) hold generally.

§3.2.2. Globalization

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

$$\mathbf{f}^{(e)} = \mathbf{K}^{(e)}\mathbf{u}^{(e)}. \tag{3.4}$$

The compact form of (3.2) and (3.3) for the e^{th} member is

$$\bar{\mathbf{u}}^{(e)} = \mathbf{T}^{(e)}\mathbf{u}^{(e)}, \qquad \mathbf{f}^{(e)} = (\mathbf{T}^{(e)})^T\,\bar{\mathbf{f}}^{(e)}. \tag{3.5}$$

Inserting these matrix expressions into (2.6) and comparing with (3.4) 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

[2] 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.

system $\{\bar{x}, \bar{y}\}$ through the congruential transformation

$$\mathbf{K}^{(e)} = (\mathbf{T}^{(e)})^T \, \bar{\mathbf{K}}^{(e)} \mathbf{T}^{(e)}. \tag{3.6}$$

Carrying out the matrix multiplications we get

$$\mathbf{K}^{(e)} = \frac{E^{(e)} A^{(e)}}{L^{(e)}} \begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix}, \tag{3.7}$$

in which $c = \cos\varphi^{(e)}$, $s = \sin\varphi^{(e)}$, with superscripts of c and s suppressed to reduce clutter. If the angle is zero we recover (2.11), as may be expected. $\mathbf{K}^{(e)}$ is called a *member stiffness matrix in global coordinates*. The proof of (3.6) and verification of (3.7) is left as Exercise 3.1.

The globalized member stiffness matrices for the example truss can now be easily obtained by inserting appropriate values into (3.7). For member (1), with end joints 1–2, angle $\varphi = 0°$ and the member data listed in Figure 2.4(a) we get

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \end{bmatrix} = 10 \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1}^{(1)} \\ u_{y1}^{(1)} \\ u_{x2}^{(1)} \\ u_{y2}^{(1)} \end{bmatrix}. \tag{3.8}$$

For member (2), with end joints 2–3, and angle $\varphi = 90°$:

$$\begin{bmatrix} f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = 5 \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x2}^{(2)} \\ u_{y2}^{(2)} \\ u_{x3}^{(2)} \\ u_{y3}^{(2)} \end{bmatrix}. \tag{3.9}$$

Finally, for member (3), with end joints 1–3, and angle $\varphi = 45°$:

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = 20 \begin{bmatrix} 0.5 & 0.5 & -0.5 & -0.5 \\ 0.5 & 0.5 & -0.5 & -0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \\ -0.5 & -0.5 & 0.5 & 0.5 \end{bmatrix} \begin{bmatrix} u_{x1}^{(3)} \\ u_{y1}^{(3)} \\ u_{x3}^{(3)} \\ u_{y3}^{(3)} \end{bmatrix}. \tag{3.10}$$

§3.2.3. Assembly 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 *merging* of individual members. The merge operation can be physically interpreted as reconnecting that member in the process of fabricating the complete structure. For a truss structure, reconnection means inserting the pins back into the joints. This is mathematically governed by two rules of structural mechanics:

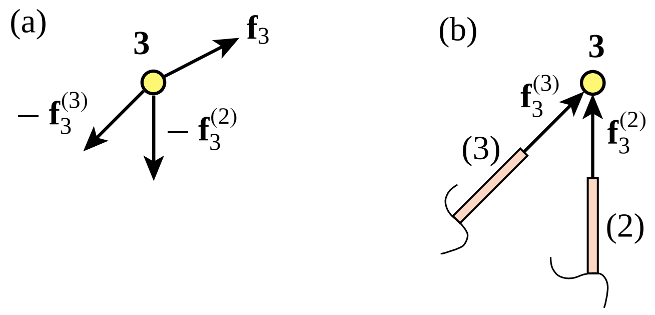


Figure 3.2. The force equilibrium of joint 3 of the example truss, depicted as a free body diagram in (a). Here $\mathbf{f}_3$ is the known external joint force applied on the joint. Joint forces $\mathbf{f}_3^{(2)}$ and $\mathbf{f}_3^{(3)}$ are applied *by the joint on the members*, as illustrated in (b). Consequently 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 if both 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 (3.11).

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

The first rule is physically obvious: reconnected joints must move as one entity. The second one can be visualized by considering a joint as a free body, but care is required in the interpretation of joint 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 this particular joint gives

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

$$\text{Rule 2:} \quad f_{x3} = f_{x3}^{(2)} + f_{x3}^{(3)} = f_{x3}^{(1)} + f_{x3}^{(2)} + f_{x3}^{(3)}, \quad f_{y3} = f_{y3}^{(2)} + f_{y3}^{(3)} = f_{y3}^{(1)} + f_{y3}^{(2)} + f_{y3}^{(3)}. \tag{3.11}$$

The addition of $f_{x3}^{(1)}$ and $f_{y3}^{(1)}$ to $f_{x3}^{(2)} + f_{x3}^{(3)}$ and $f_{y3}^{(2)} + f_{y3}^{(3)}$, respectively, changes nothing because member (1) is not connected to joint 3; we are simply adding zeros. But this augmentation enables us to write the matrix relation: $\mathbf{f} = \mathbf{f}^{(1)} + \mathbf{f}^{(2)} + \mathbf{f}^{(3)}$, which will be used in (3.19).

§3.2.4. Hand Assembly by Augmentation and Merge

To directly visualize how the two assembly rules translate to member merging rules, we first *augment* the member stiffness relations by adding zero rows and columns as appropriate to *complete* the force and displacement vectors.

For member (1):

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \\ f_{x3}^{(1)} \\ f_{y3}^{(1)} \end{bmatrix} = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1}^{(1)} \\ u_{y1}^{(1)} \\ u_{x2}^{(1)} \\ u_{y2}^{(1)} \\ u_{x3}^{(1)} \\ u_{y3}^{(1)} \end{bmatrix}. \tag{3.12}$$

For member (2):

$$\begin{bmatrix} f_{x1}^{(2)} \\ f_{y1}^{(2)} \\ f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 0 & 5 \end{bmatrix} \begin{bmatrix} u_{x1}^{(2)} \\ u_{y1}^{(2)} \\ u_{x2}^{(2)} \\ u_{y2}^{(2)} \\ u_{x3}^{(2)} \\ u_{y3}^{(2)} \end{bmatrix}. \tag{3.13}$$

For member (3):

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x2}^{(3)} \\ f_{y2}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 0 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & 0 & 10 & 10 \end{bmatrix} \begin{bmatrix} u_{x1}^{(3)} \\ u_{y1}^{(3)} \\ u_{x2}^{(3)} \\ u_{y2}^{(3)} \\ u_{x3}^{(3)} \\ u_{y3}^{(3)} \end{bmatrix}. \tag{3.14}$$

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

$$\begin{bmatrix} f_{x1}^{(1)} \\ f_{y1}^{(1)} \\ f_{x2}^{(1)} \\ f_{y2}^{(1)} \\ f_{x3}^{(1)} \\ f_{y3}^{(1)} \end{bmatrix} = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}, \tag{3.15}$$

$$\begin{bmatrix} f_{x1}^{(2)} \\ f_{y1}^{(2)} \\ f_{x2}^{(2)} \\ f_{y2}^{(2)} \\ f_{x3}^{(2)} \\ f_{y3}^{(2)} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 0 & 5 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}, \tag{3.16}$$

$$\begin{bmatrix} f_{x1}^{(3)} \\ f_{y1}^{(3)} \\ f_{x2}^{(3)} \\ f_{y2}^{(3)} \\ f_{x3}^{(3)} \\ f_{y3}^{(3)} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 0 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & 0 & 10 & 10 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \tag{3.17}$$

These three equations can be represented in direct matrix notation as

$$\mathbf{f}^{(1)} = \mathbf{K}^{(1)}\,\mathbf{u}, \qquad \mathbf{f}^{(2)} = \mathbf{K}^{(2)}\,\mathbf{u}, \qquad \mathbf{f}^{(3)} = \mathbf{K}^{(3)}\,\mathbf{u}. \tag{3.18}$$

According to the second rule

$$\mathbf{f} = \mathbf{f}^{(1)} + \mathbf{f}^{(2)} + \mathbf{f}^{(3)} = \big(\mathbf{K}^{(1)} + \mathbf{K}^{(2)} + \mathbf{K}^{(3)}\big)\mathbf{u} = \mathbf{K}\mathbf{u}, \tag{3.19}$$

so all we have to do is add the three stiffness matrices that appear above, and we arrive at the master stiffness equations:

$$\begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix}. \tag{3.20}$$

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 an equation solver we need to separate known and unknown components of **f** and **u**. In this Section a technique suitable for hand computation is described.

§3.3.1. Applying Displacement BCs by Reduction

If one attempts to solve the system (3.20) 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 **K** are linear combinations of each other (see Remark 3.4 below). The physical interpretation of singularity is that there are still unsuppressed *rigid body motions*: the truss "floats" in the $\{x, y\}$ plane.

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

$$u_{x1} = u_{y1} = u_{y2} = 0, \tag{3.21}$$

whereas the known applied forces are

$$f_{x2} = 0, \quad f_{x3} = 2, \quad f_{y3} = 1. \tag{3.22}$$

When solving the stiffness equations by hand, the simplest way to account for support conditions is to *remove* equations associated with known joint displacements from the master system. To apply (3.21) 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 **K** and the corresponding components from **f** and **u**. The reduced three-equation system is

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x2} \\ f_{x3} \\ f_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix}. \tag{3.23}$$

Equation (3.23) is called the *reduced master stiffness system*. The coefficient matrix of this system is no longer singular.

REMARK 3.3

In mathematical terms, the free-free master stiffness matrix **K** in (3.20) 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 **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 3.4

Conditions (3.21) represent the simplest type of support conditions, namely zero specified displacements. Subsequent Chapters discuss how more general constraint forms, such as prescribed nonzero displacements and multipoint constraints, are handled.

§3.3.2. Solving for Displacements

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

$$\begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0.4 \\ -0.2 \end{bmatrix}. \tag{3.24}$$

This is called a *partial displacement solution* because it excludes suppressed displacement components. This solution vector is *expanded* to six components by including the specified values

(3.21):

$$\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \\ -0.2 \end{bmatrix} \tag{3.25}$$

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

§3.4. POSTPROCESSING

The last major 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 (3.25) by **K** we get

$$\mathbf{f} = \mathbf{K}\mathbf{u} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0.4 \\ -0.2 \end{bmatrix} = \begin{bmatrix} -2 \\ -2 \\ 0 \\ 1 \\ 2 \\ 1 \end{bmatrix} \tag{3.26}$$

This vector recovers the known applied forces (3.22) as can be expected. Furthermore we get three *reaction forces*: $f_{x1} = f_{y1} = -2$ and $f_{y2} = 1$, which are associated with the support conditions (3.21). It is easy to check that the complete force system is in equilibrium; this is the topic of Exercise 3.2.

§3.4.2. Recovery of Internal Forces and Stresses

Frequently the structural engineer is not primarily interested in displacements but in *internal forces* and *stresses*. These are in fact the most important quantities for preliminary design.

In trusses the only internal forces are the axial member forces, which are depicted in Figure 3.3. These forces are denoted by $p^{(1)}$, $p^{(2)}$ and $p^{(3)}$ and collected in a vector **p**. The average axial stress $\sigma^{(e)}$ is easily obtained on dividing $p^{(e)}$ by the cross-sectional area of the member.

The axial force $p^{(e)}$ in member (e) can be obtained as follows. Extract the displacements of member (e) from the displacement solution **u** to form $\mathbf{u}^{(e)}$. Then recover local joint displacements from $\bar{\mathbf{u}}^{(e)} = \mathbf{T}^{(e)}\mathbf{u}^{(e)}$. Compute the deformation d (relative displacement) and recover the axial force from the equivalent spring constitutive relation:

$$d^{(e)} = \bar{u}^{(e)}_{xj} - \bar{u}^{(e)}_{xi}, \qquad p^{(e)} = \frac{E^{(e)}A^{(e)}}{L^{(e)}} d^{(e)}. \tag{3.27}$$

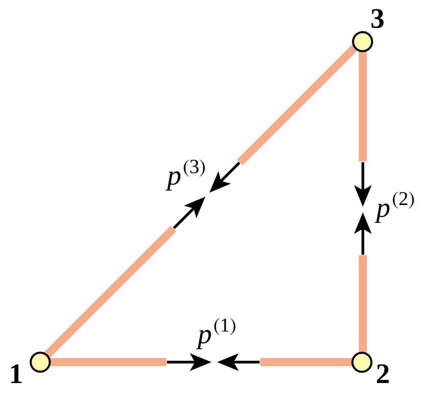


Figure 3.3. The internal forces in the example truss are the axial forces $p^{(1)}$, $p^{(2)}$ and $p^{(3)}$ in the members. Signs shown for these forces correspond to tension.

An alternative interpretation of (3.27) 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 $p^{(e)} = A^{(e)} \sigma^{(e)}$ as the axial force. This is more in tune with the Theory of Elasticity viewpoint discussed in Exercise 2.6.

§3.5. 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 §3.2.4. There are two major differences:

(I) Member stiffness matrices are *not* expanded. Their entries are directly merged into those of **K** through the use of a "freedom pointer array" called the *Element Freedom Table* or EFT.

(II) The master stiffness matrix **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 **K** is stored as a full square matrix, and study only (I). For the example truss the freedom-pointer technique expresses the entries of **K** as the sum

$$K_{pq} = \sum_{e=1}^{3} K_{ij}^{(e)} \qquad \text{for} \quad i = 1, \ldots 4, \;\; j = 1, \ldots 4, \;\; p = \text{EFT}^{(e)}(i), \;\; q = \text{EFT}^{(e)}(j). \tag{3.28}$$

Here $K_{ij}^{(e)}$ denote the entries of the 4×4 globalized member stiffness matrices in (3.9) through (3.11). Entries K_{pq} that do not get any contributions from the right hand side remain zero. $\text{EFT}^{(e)}$ denotes the Element Freedom Table for member (e). For the example truss these tables are

$$\text{EFT}^{(1)} = \{1, 2, 3, 4\}, \qquad \text{EFT}^{(2)} = \{3, 4, 5, 6\}, \qquad \text{EFT}^{(3)} = \{1, 2, 5, 6\}. \tag{3.29}$$

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 (3.28) involves 3 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 **K** to roughly halve the number of additions. Exercise 3.6 follows the process (3.28) by hand.

§3.5.2. Applying Displacement BCs by Modification

In §3.3.1 the support conditions (3.21) were applied by reducing (3.20) to (3.23). 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 on secondary storage.

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

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 2 \\ 1 \end{bmatrix}, \tag{3.30}$$

in which rows and columns for equations 1, 2 and 4 have been cleared. Solving this modified system yields the complete displacement solution (3.25).

REMARK 3.5

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 .

Homework Exercises for Chapter 3
The Direct Stiffness Method: Assembly and Solution

EXERCISE 3.1

[A:20] Derive (3.6) from $\bar{\mathbf{K}}^{(e)}\bar{\mathbf{u}}^{(e)} = \bar{\mathbf{f}}^{(e)}$, (3.4) and (3.5). (*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 (3.7). Use Falk's scheme for the multiplications.[3])

EXERCISE 3.2

[A:15] Draw a free body diagram of the nodal forces (3.26) acting on the free-free truss structure, and verify that this force system satisfies translational and rotational (moment) equilibrium.

EXERCISE 3.3

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

EXERCISE 3.4

[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.

EXERCISE 3.5

[A:20] Suppose that the third support condition in (3.21) is $u_{x2} = 0$ instead of $u_{y2} = 0$. Rederive the reduced system (3.23) 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 3.6

[N:20] Construct by hand the free-free master stiffness matrix of (3.20) using the freedom-pointer technique (3.28). Note: start from $\mathbf{K}$ initialized to the null matrix, then cycle over $e = 1, 2, 3$.

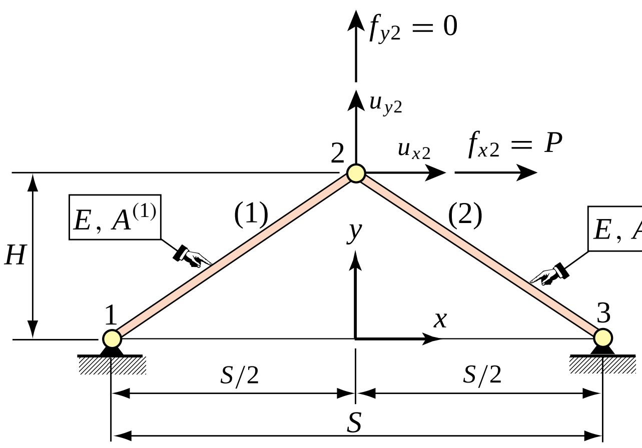


Figure E3.1. Truss structure for Exercise 3.7.

[3] This scheme is recommended to do matrix multiplication by hand. It is explained in §B.3.2 of Appendix B.

[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: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:

(a) Assemble the master stiffness equations. Any method: expanded element stiffness, or the more advanced "freedom pointer" technique explained in §3.5.1, is acceptable.

(b) Apply the displacement BCs and solve the reduced system for the crown displacements u_{x2} and u_{y2}. Partial result: $u_{x2} = 9/512 = 0.01758$.

(c) Recover the joint forces at all joints including reactions. Verify that overall force equilibrium (x forces, y forces, and moments about any point) is satisfied.

(d) Recover the axial forces in the two members. Result should be $p^{(1)} = -p^{(2)} = 15/2$.

4 The Direct Stiffness Method: Additional Topics

TABLE OF CONTENTS

Chapters 2 and 3 presented the “core” steps of the Direct Stiffness Method (DSM). These steps were illustrated with the hand analysis of a plane truss structure. This Chapter covers some additional topics that were left out from Chapters 2–3 for clarity. These include: the imposition of prescribed nonzero displacements, and the treatment of thermal effects.

§4.1. PRESCRIBED NONZERO DISPLACEMENTS

The support conditions considered in the example truss of Chapters 2–3 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. Mathematically these are called non-homogenous boundary conditions. The treatment of this generalization of the FEM equations is studied in the following subsections.

§4.1.1. Application of DBCs by Reduction

We describe first a reduction technique, analogous to that explained in §3.2.1, which is suitable for hand computations. Recall the master stiffness equations (3.20) for the example truss:

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \\ f_{x3} \\ f_{y3} \end{bmatrix} \tag{4.1}$$

Suppose that the applied forces are as for the example truss but the prescribed displacements are

$$u_{x1} = 0, \quad u_{y1} = -0.5, \quad u_{y2} = 0.4 \tag{4.2}$$

This means that joint 1 goes down vertically whereas joint 2 goes up vertically, as depicted in Figure 4.1. Inserting the known data into (4.1) we get

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ -0.5 \\ u_{x2} \\ 0.4 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ 0 \\ f_{y2} \\ 2 \\ 1 \end{bmatrix} \tag{4.3}$$

The first, second and fourth rows of (4.3) are removed, leaving only

$$\begin{bmatrix} -10 & 0 & 10 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ -0.5 \\ u_{x2} \\ 0.4 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \tag{4.4}$$

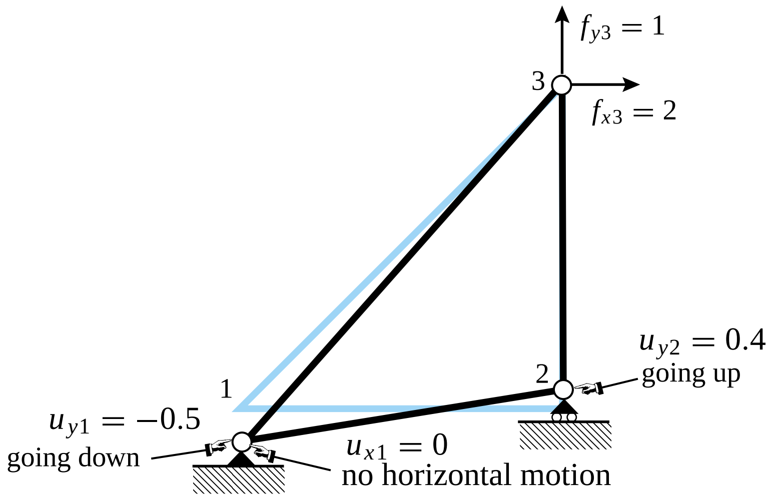


Figure 4.1. The example truss with prescribed nonzero vertical displacements at joints 1 and 2.

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

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} - \begin{bmatrix} (-10) \times 0 + 0 \times (-0.5) + 0 \times 0.4 \\ (-10) \times 0 + (-10) \times (-0.5) + 0 \times 0.4 \\ (-10) \times 0 + (-10) \times (-0.5) + (-5) \times 0.4 \end{bmatrix} = \begin{bmatrix} 0 \\ -3 \\ -2 \end{bmatrix}. \tag{4.5}$$

The matrix equation (4.5) is the *reduced system*. Note that its coefficient matrix is exactly the same as in the reduced system (3.23) 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*. Solving the reduced system yields

$$\begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0.2 \end{bmatrix}. \tag{4.6}$$

Filling the missing slots with the known values (4.2) yields the complete displacement solution

$$\mathbf{u} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0.4 \\ -0.5 \\ 0.2 \end{bmatrix} \tag{4.7}$$

Going through the postprocessing steps discussed in §3.3 with (4.7), we can find that the reaction forces and the 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.

§4.1.2. Application of DBCs by Modification

The computer-oriented modifification approach follows the same idea outlined in §3.5.2. As there, the key objective is to avoid rearranging the master stiffness equations. To understand the process it is useful to think of doing it 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 support conditions (4.2):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0.4 \\ 2 \\ 1 \end{bmatrix} \tag{4.8}$$

The solution of this system recovers (4.2) 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 detailed in (4.5). We thus arrive at

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 10 \\ 0 & 0 & 0 & 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ -0.5 \\ 0 \\ 0.4 \\ -3 \\ -2 \end{bmatrix} \tag{4.9}$$

As before, this is called the *modified master stiffness system*. Observe that the equations retain the original order. Solving this system yields the complete displacement solution (4.7).

Note that if all prescribed displacements were zero forces on the right hand side are not modified, and one would have (3.30).

REMARK 4.1

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* procedure. 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* procedure. It is essential that the procedures be executed in the indicated order, because stiffness terms must be used in the force modification before they are cleared.

§4.1.3. Matrix Forms of DBC Application Methods

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

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{K}_{12} \\ \mathbf{K}_{21} & \mathbf{K}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \end{bmatrix}. \tag{4.10}$$

In this matrix equation, subvectors $\mathbf{u}_2$ and $\mathbf{f}_1$ collect displacement and force components, respectively, that are *known, given* or *prescribed*. On the other hand, subvectors $\mathbf{u}_1$ and $\mathbf{f}_2$ collect force and displacement components, respectively, that are *unknown*. The force components in $\mathbf{f}_2$ are reactions on supports; consequently $\mathbf{f}_2$ is called the *reaction force vector*.

On transferring the known terms to the right hand side the first matrix equation becomes

$$\mathbf{K}_{11}\mathbf{u}_1 = \mathbf{f}_1 - \mathbf{K}_{12}\mathbf{u}_2. \tag{4.11}$$

This is the *reduced master equation system*. If the displacement B.C. 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:

$$\mathbf{K}_{11}\mathbf{u}_1 = \mathbf{f}_1. \tag{4.12}$$

Examples that illustrate (4.11) and (4.12) are (4.5) and (3.23), respectively.

The computer-oriented modification technique retains the same joint displacement vector as in (4.10) through the following rearrangement:

$$\begin{bmatrix} \mathbf{K}_{11} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{u}_1 \\ \mathbf{u}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{f}_1 - \mathbf{K}_{12}\mathbf{u}_2 \\ \mathbf{u}_2 \end{bmatrix}, \tag{4.13}$$

This *modified system* is simply the reduced equation (4.11) augmented by the trivial equation $\mathbf{I}\mathbf{u}_2 = \mathbf{u}_2$. This system is often denoted as

$$\widehat{\mathbf{K}}\mathbf{u} = \hat{\mathbf{f}}. \tag{4.14}$$

Solving (4.14) yields the complete displacement solution including the specified displacements $\mathbf{u}_2$.

For the computer implemenmtation it is important to note that the partitioned form (4.12) is used only to facilitate the use of matrix notation. The equations are not explicitly rearranged and retain their original numbers. For instance, in the example truss

$$\mathbf{u}_1 = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{y2} \end{bmatrix} \equiv \begin{bmatrix} \text{DOF \#1} \\ \text{DOF \#2} \\ \text{DOF \#4} \end{bmatrix}, \qquad \mathbf{u}_2 = \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} \equiv \begin{bmatrix} \text{DOF \#3} \\ \text{DOF \#5} \\ \text{DOF \#6} \end{bmatrix}. \tag{4.15}$$

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

§4.2. THERMOMECHANICAL EFFECTS

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."

[1] Called *initial stress* or *initial strain* effects by many authors. These names reflect what is regarded as the physical source of initial force effects at the continuum level.

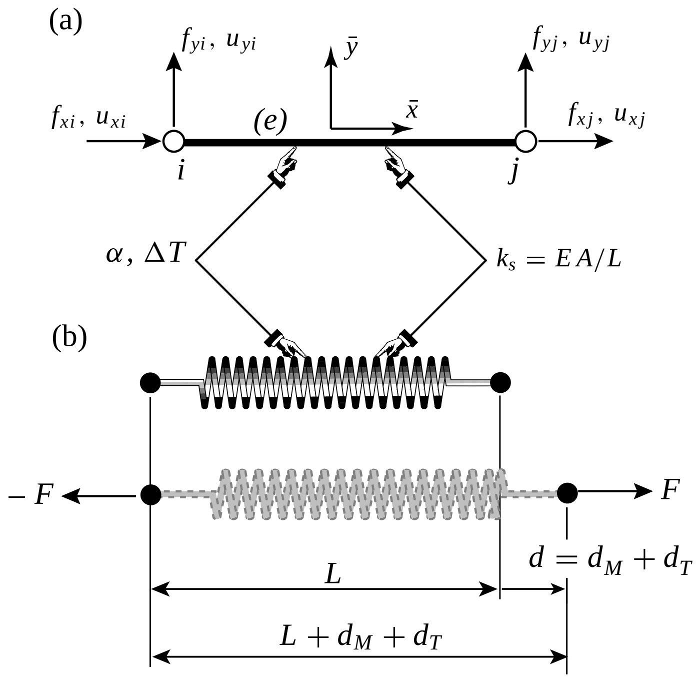


Figure 4.2. Generic truss member subjected to mechanical and thermal effects: (a) idealization as bar, (b) idealization as equivalent linear spring.

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 §4.3 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.

§4.2.1. Thermomechanical Behavior

Consider the generic plane-truss member shown in Figure 4.2. The member is prismatic and uniform. The temperature T is also uniform. For clarity the member identification subscript will be omitted in the following development until the transformation-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

[2] These are important in composite materials and geomechanics.

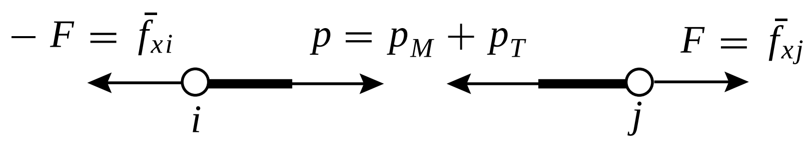


Figure 4.3. Equilibrium of generic truss member under thermomechanical forces. Here F and p denote effective forces.

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 is free to change from L to $L + d_T$. If the thermoelastic constitutive behavior is linear[3] then d_T is proportional to L and ΔT:

$$d_T = \alpha L \, \Delta T, \tag{4.16}$$

where 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

$$e_T = d_T/L = \alpha \, \Delta T. \tag{4.17}$$

Now suppose that the member is also subject to mechanical forces, more precisely the applied axial force F shown in Figure 4.2. The member axial stress is $\sigma = F/A$. In response to this stress the length changes by d_M. The *mechanical strain* is $e_M = d_M/L$. Hence the total strain $e = d/L = (d_M + d_T)/L$ is the sum of the mechanical and the thermal strains:

$$\boxed{e = e_M + e_T = \frac{\sigma}{E} + \alpha \Delta T} \tag{4.18}$$

This superposition of deformations is the basic assumption made in the thermomechanical analysis of trusses under static loads. It is physically obvious for an unconstrained member such as that depicted in Figure 4.2.

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 $\sigma = E e_M = -E e_T = -E\alpha \, \Delta T < 0$. This *thermal stress* is further discussed in Remark 4.2.

[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.

§4.2.2. Thermomechanical Stiffness Equations

Because $e = d/L$ and $d = \bar{u}_{xj} - \bar{u}_{xi}$, (4.18) can be developed as

$$\frac{\bar{u}_{xj} - \bar{u}_{xi}}{L} = \frac{\sigma}{E} + \alpha\,\Delta T, \tag{4.19}$$

To pass to internal forces (4.19) is multiplied through by EA:

$$\frac{EA}{L}(\bar{u}_{xj} - \bar{u}_{xi}) = A\sigma + EA\alpha\,\Delta T = p_M + p_T = F. \tag{4.20}$$

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 (4.20), $F = p = p_M + p_T$ follows from free-body member equilibrium; see Figure 4.3. Passing to matrix form:

$$F = \frac{EA}{L}\,[\,-1 \quad 0 \quad 1 \quad 0\,]\begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{yi} \\ \bar{u}_{yj} \end{bmatrix}. \tag{4.21}$$

Noting that $F = \bar{f}_{xj} = -\bar{f}_{xi}$, $\bar{f}_{yi} = \bar{f}_{yj} = 0$, we can relate joint forces to joint displacements as

$$\begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{yi} \\ \bar{f}_{xj} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} \bar{f}_{Mxi} \\ \bar{f}_{Myi} \\ \bar{f}_{Mxj} \\ \bar{f}_{Myj} \end{bmatrix} + \begin{bmatrix} \bar{f}_{Txi} \\ \bar{f}_{Tyi} \\ \bar{f}_{Txj} \\ \bar{f}_{Tyj} \end{bmatrix} = \begin{bmatrix} \bar{f}_{Mxi} \\ \bar{f}_{Myi} \\ \bar{f}_{Mxj} \\ \bar{f}_{Myj} \end{bmatrix} + E\alpha\,\Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \frac{EA}{L}\begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{yi} \\ \bar{u}_{xj} \\ \bar{u}_{yj} \end{bmatrix}, \tag{4.22}$$

In compact matrix form this is $\bar{\mathbf{f}} = \bar{\mathbf{f}}_M + \bar{\mathbf{f}}_T = \bar{\mathbf{K}}\bar{\mathbf{u}}$, or

$$\boxed{\bar{\mathbf{K}}\bar{\mathbf{u}} - \bar{\mathbf{f}}_T = \bar{\mathbf{f}}_M.} \tag{4.23}$$

Here $\bar{\mathbf{K}}$ is the same member stiffness matrix derived in §2.6.3. The new ingredient that appears is the vector

$$\bar{\mathbf{f}}_T = EA\alpha\,\Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \tag{4.24}$$

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 4.2

A useful physical interpretation of (4.23) is as follows. Suppose that the member is completely blocked against joint motions so that $\bar{\mathbf{u}} = \mathbf{0}$. Then $\bar{\mathbf{f}}_M = -\bar{\mathbf{f}}_T$. 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 — that is, $\Delta T > 0$ — 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.

§4.2.3. Globalization

At this point we restore the member superscript so that the member stiffness equations(4.22) are rewritten as

$$\bar{\mathbf{K}}^{(e)}\bar{\mathbf{u}}^{(e)} - \bar{\mathbf{f}}_T^{(e)} = \bar{\mathbf{f}}_M^{(e)}. \tag{4.25}$$

Use of the transformation rules developed in §3.1 to change displacements and forces to the global system $\{x, y\}$ yields

$$\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{f}_T^{(e)} = \mathbf{f}_M^{(e)}, \tag{4.26}$$

where $\mathbf{T}^{(e)}$ is the displacement transformation matrix (3.1), and the transformed quantities are

$$\mathbf{K}^{(e)} = \left(\mathbf{T}^{(e)}\right)^T \bar{\mathbf{K}}^{(e)}\mathbf{T}^{(e)}, \qquad \mathbf{f}_M^{(e)} = \left(\mathbf{T}^{(e)}\right)^T \bar{\mathbf{f}}^{(e)}, \qquad \mathbf{f}_T^{(e)} = \left(\mathbf{T}^{(e)}\right)^T \bar{\mathbf{f}}_T^{(e)}. \tag{4.27}$$

These globalized member equations are used to assemble the free-free master stiffness equations by a member merging process.

§4.2.4. Merge

The merge process is based on two assembly rules:

1. *Compatibility of deformations*: The joint displacements of all members meeting at a joint are the same.
2. *Force equilibrium*: The sum of *effective* forces exerted by all members that meet at a joint balances the *external mechanical* forces that act on that joint.

Comparing these to the rules stated in §3.1.3 for purely mechanical actions, reveals that the only difference pertains to Rule 2, in which internal mechanical forces become effective forces. This is the reason for keeping $\mathbf{f}_M$ and $\mathbf{f}_T$ on both sides of (4.25) and (4.26).

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.[4] Upon completion of the assembly process we arrive at the free-free master stiffness equations

$$\mathbf{K}\mathbf{u} - \mathbf{f}_T = \mathbf{f}_M. \tag{4.28}$$

[4] An illustrative example is provided in §4.2.7.

§4.2.5. Solution

For the solution phase it is convenient to write (4.28) as

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_T = \mathbf{f}.} \tag{4.29}$$

This has formally the same configuration as the master stiffness equations (2.3). The only difference is that the *effective* joint force vector **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 §3.2.1, §3.4.1 and §4.1. Processing the reduced or modified system by a linear equation solver yields the displacement solution **u**.

§4.2.6. Postprocessing

The postprocessing steps described in §3.4 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

$$\mathbf{f}_M = \mathbf{K}\mathbf{u} - \mathbf{f}_T \tag{4.30}$$

To recover mechanical internal forces in member (e), obtain $p^{(e)}$ by the procedure outlined in §3.4.2, and subtract the thermal component:

$$p_M^{(e)} = p^{(e)} - E^{(e)} A^{(e)} \alpha^{(e)}\, \Delta T^{(e)}. \tag{4.31}$$

The mechanical axial stress is then $\sigma^{(e)} = p_M^{(e)}/A^{(e)}$.

§4.2.7. A Worked-Out Example

To go over the steps by hand, suppose that the example truss is now *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 first task is to get the thermal forces for each member in global coordinates. Using (4.26) and the third of (4.27) these are easily worked out to be

$$\begin{aligned}
\mathbf{f}_T^{(1)} &= E^{(1)} A^{(1)} \alpha^{(1)}\, \Delta T^{(1)} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 100\alpha\, \Delta T \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \\
\mathbf{f}_T^{(2)} &= E^{(2)} A^{(2)} \alpha^{(2)}\, \Delta T^{(2)} \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 50\alpha\, \Delta T \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix}, \\
\mathbf{f}_T^{(3)} &= E^{(3)} A^{(3)} \alpha^{(3)}\, \Delta T^{(3)} \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = 200\alpha\, \Delta T \begin{bmatrix} -1 \\ -1 \\ 1 \\ 1 \end{bmatrix}.
\end{aligned} \tag{4.32}$$

Merging the contribution of these 3 members gives the master thermal force vector

$$\mathbf{f}_T = \alpha\,\Delta T \begin{bmatrix} -100+0-200 \\ 0+0-200 \\ 100+0+0 \\ 0-50+0 \\ 0+0+200 \\ 0+50+200 \end{bmatrix} = \alpha\,\Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix} \tag{4.33}$$

The master stiffness matrix **K** does not change. Consequently the master stiffness equations are

$$\begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x1}=0 \\ u_{y1}=0 \\ u_{x2} \\ u_{y2}=0 \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} f_{Mx1} \\ f_{My1} \\ f_{Mx2}=0 \\ f_{My2} \\ f_{Mx3}=0 \\ f_{My3}=0 \end{bmatrix} + \alpha\,\Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix} \tag{4.34}$$

in which f_{Mx1}, f_{My1} and f_{My2} are the unknown mechanical reaction forces, and the known forces and displacements have been marked. Because the prescribed displacements are zero, the reduced system is simply

$$\begin{bmatrix} 10 & 0 & 0 \\ 0 & 10 & 10 \\ 0 & 10 & 15 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \alpha\,\Delta T \begin{bmatrix} 100 \\ 200 \\ 250 \end{bmatrix} = \alpha\,\Delta T \begin{bmatrix} 100 \\ 200 \\ 250 \end{bmatrix}. \tag{4.35}$$

Solving (4.35) gives $u_{x2} = u_{x3} = u_{y3} = 10\alpha\,\Delta T$. Completing **u** with the prescribed zero displacements and premultiplying by **K** gives the complete effective force vector:

$$\mathbf{f} = \mathbf{K}\mathbf{u} = \begin{bmatrix} 20 & 10 & -10 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & -5 & 10 & 15 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 10 \\ 0 \\ 10 \\ 10 \end{bmatrix} \alpha\,\Delta T = \alpha\,\Delta T \begin{bmatrix} -300 \\ -200 \\ 100 \\ -50 \\ 200 \\ 250 \end{bmatrix} = \mathbf{f}_T. \tag{4.36}$$

Consequently

$$\mathbf{f}_M = \mathbf{K}\mathbf{u} - \mathbf{f}_T = \mathbf{0}. \tag{4.37}$$

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. Such structures do not develop thermal stresses for *any* combination of temperature changes.

§4.3. INITIAL FORCE EFFECTS

As previously noted, a wide spectrum of mechanical and non-mechanical effects can be acommodated under the umbrella of the *initial force* concept. The stiffness equations at the local (member) level are

$$\boxed{\bar{\mathbf{K}}\bar{\mathbf{u}} = \bar{\mathbf{f}}_M + \bar{\mathbf{f}}_I = \bar{\mathbf{f}},} \tag{4.38}$$

and at the global (assembled structure) level:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I = \mathbf{f}.} \tag{4.39}$$

In these equations subscripts M and I identify mechanical and initial joint forces, respectively. The sum of the two: $\bar{\mathbf{f}}$ at the member level and $\mathbf{f}$ at the structure level, are called *effective* forces.

A physical interpretation of (4.39) 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 initial stresses (often called prestresses).

Local effects that lead to initial forces at the member level are: temperature changes (studied in §5.2, where $\mathbf{f}_I \equiv \mathbf{f}_T$), moisture diffusion, residual stresses, lack of fit in fabrication, and in-member prestressing. Global effects include prescribed nonzero joint displacements (studied in §5.1) and multimember prestressing (for example, by cable pretensioning).

As can be seen there is a wide variety of physical effects that lead to nonzero initial forces. The good news is that once the member equations (4.38) are formulated, the remaining DSM steps (globalization, merge and solution) are identical. This nice property extends to the general Finite Element Method.

§4.4. PSEUDOTHERMAL INPUTS

Some commercial FEM programs do not have a way to handle directly effects such as moisture, lack or fit or prestress. But all can handle temperature variation inputs. Since all these effects can be treated as initial forces, it is possible (at least for bar elements) to model them as fictitious thermomechanical effects, by inputting fictitious 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 input $\Delta T + \Delta T_P$.

If this device is used, some care should be exercised in interpreting results for internal forces given by the program. The trick is not necessary for personal or open-source codes over which you have full control.

Homework Exercises for Chapter 4

The Direct Stiffness Method: Additional Topics

EXERCISE 4.1

[N:20] Resolve items (a) through (c) — omitting (d) — of the problem of Exercise 3.7 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 a reduction scheme to apply the displacement BCs.

EXERCISE 4.2

[N:25] Use the same data of Exercise 3.7 except that $P = 0$ and so there are no applied mechanical forces. Both members have the same dilatation coefficient $\alpha = 10^{-6}$ 1/°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 $\Delta T = 120\,^\circ$F above T_{ref}, whereas member (2) stays at T_{ref}.

EXERCISE 4.3

[A:15] Consider the generic truss member of Figure 2.6. 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 (4.38). *Hint*: find the mechanical forces that would compensate for δ and restore the desired length.

EXERCISE 4.4

[A:10] Show that the lack of fit of the previous exercise can be viewed as equivalent to a prestress force of $-(EA/L)\delta$.

5 Analysis of Example Truss by a CAS

TABLE OF CONTENTS

§5.1. 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 Sy**mbolic **Ma**nipulator) — 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.

§5.1.1. 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. 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. Its competitor *Maple* is primarily used on Unix systems.
2. Up-to-date user interface with above average graphics. On all machines *Mathematica* offers a graphics user interface called the Notebook front-end. This is mandatory for serious work.
3. A powerful programming language.
4. Good documentation and abundance of application books at all levels.

One common disadvantage of CAS, and *Mathematica* is not 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.

§5.2. PROGRAMMING STYLE AND PREREQUISITES

The following material assumes that you are a moderately experienced user of *Mathematica*, or are willing to learn to be one. The *Mathematica Book* is just a reference manual and not good for training. But there is an excellent tutorial available: *The Beginner's Guide to Mathematica* by Jerry Glynn and Theodore W. Gray.[3]

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.

---

[1] Some 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. As of now, only humans can do mathematics.

[2] Another commonly used program for engineering computations: *Matlab*, does only numerical computations although an interface to *Maple* can be purchased as a toolbox.

[3] This is also available 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*.

When approaching that level you may notice that functions in *Mathematica* display many aspects similar to C.[4] 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.

Although function arguments can be modified, in practice this 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.[5]

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.[6] The program will not be particularly modular (in the computer science sense) because *Mathematica* is not suitable for that programming style.[7] The code below 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.

§5.3. THE ELEMENT STIFFNESS MODULE

As our first code segment, the top box of Figure 5.1 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 `<Enter>`, gives the output shown in the bottom box. The contents of the figure is described in further detail below.

§5.3.1. Module Description

The module is called `ElemStiff2DTwoNodeBar`. Such descriptive names are permitted by the language. It takes two arguments:

`{ { x1 , y1 }, { y1 , y2 } }`	A two-level list[8] containing the $\{x, y\}$ coordinates of the bar end nodes labelled as 1 and 2.[9]
`{ Em , A }`	A level-one list containing the bar elastic modulus, E and the member cross section area, A. See §5.3.3 as to why name `E` cannot be used.

[4] Simple functions can be implemented in `Mathematica` directly, for instance `DotProduct[x_,y_]:=x.y;` more complex ones are handled by the `Module` construct emphasized here.

[5] 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.

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

[7] And indeed none of the CAS packages in popular use is designed for strong modularity because of historical and interactivity constraints.

[8] 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.

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

```
ElemStiff2DTwoNodeBar[{{x1_,y1_},{x2_,y2_}},{Em_,A_}] :=
  Module[{c,s,dx=x2-x1,dy=y2-y1,L,Ke},
    L=Sqrt[dx^2+dy^2]; c=dx/L; s=dy/L;
    Ke=(Em*A/L)* {{ c^2, c*s,-c^2,-c*s},
                  { c*s, s^2,-s*c,-s^2},
                  {-c^2,-s*c, c^2, s*c},
                  {-s*c,-s^2, s*c, s^2}};
    Return[Ke]
 ];
Ke= ElemStiff2DTwoNodeBar[{{0,0},{10,10}},{100,2*Sqrt[2]}];
Print["Numerical elem stiff matrix:"]; Print[Ke//MatrixForm];
Ke= ElemStiff2DTwoNodeBar[{{0,0},{L,L}},{Em,A}];
Ke=Simplify[Ke,L>0];
Print["Symbolic elem stiff matrix:"]; Print[Ke//MatrixForm];
```

```
Numerical elem stiff matrix:

 10   10  -10  -10
 10   10  -10  -10
-10  -10   10   10
-10  -10   10   10

Symbolic elem stiff matrix:
```

$$\begin{pmatrix} \frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} \\ \frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} \\ -\frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} \\ -\frac{A\,Em}{2\sqrt{2}\,L} & -\frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} & \frac{A\,Em}{2\sqrt{2}\,L} \end{pmatrix}$$

Figure 5.1. Module `ElemStiff2DTwoNodeBar` to form the element stiffness of a 2D bar element in global coordinates, test program and its output.

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 aspect 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 standard programming languages. The format in fact closely matches the mathematical expression (3.4).

§5.3.2. Programming Comments

The function in Figure 5.1 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.[10] In the `Module[ ...]` construct this is done by listing those names in a list immediately 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 use of 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:

[10] The "global by default" choice is the worst one, but we must live with the rules of the language.

the `Return` statement makes clear what the `Module` gives back to its invoker and is self-documenting.

§5.3.3. 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. If, for example, you cannot use `E` because of its built-in meaning as the base of natural logarithths.[11]

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, following 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 the latter is a reserved symbol.

§5.3.4. Testing the Member Stiffness Module

Following the definition of `ElemStiff2DTwoNodeBar` in Figure 5.1 there are several statements that constitute the *module test program* that call the module and print the 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 5.1.

The first test returns the member stiffness matrix (3.10) 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, which is the subject of an Exercise. The ability to carry along variables is of course a fundamental capability of any CAS and the major reasons for which such programs are used.

§5.4. MERGING A MEMBER INTO THE MASTER STIFFNESS

The next fragment of *Mathematica* code, listed in Figure 5.2, 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 matrix. The module takes three arguments:

`Ke`	The 4×4 member stiffness matrix to be merged. This is a level-two list.
`eftab`	The column of the Element Freedom Table, defined in §3.4.1, appropriate to the member being merged; cf. (3.29). Recall that the EFT lists the global equation numbers for the four member degrees of freedom. 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. Thus we encounter here a novelty: an input-output

[11] 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.

```
MergeElemIntoMasterStiff[Ke_,eftab_,Kin_]:=Module[
{i,j,ii,jj,K=Kin},
   For [i=1, i<=4, i++, ii=eftab[[i]];
      For [j=i, j<=4, j++, jj=eftab[[j]];
           K[[jj,ii]]=K[[ii,jj]]+=Ke[[i,j]]
      ]
   ]; Return[K]
];
K=Table[0,{6},{6}];
Print["Initialized master stiffness matrix:"];
Print[K//MatrixForm]
Ke=ElemStiff2DTwoNodeBar[{{0,0},{10,10}},{100,2*Sqrt[2]}];
Print["Member stiffness matrix:"]; Print[Ke//MatrixForm];
K=MergeElemIntoMasterStiff[Ke,{1,2,5,6},K];
Print["Master stiffness after member merge:"];
Print[K//MatrixForm];
```

```
Initialized master stiffness matrix:

 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

Member stiffness matrix:

  10   10  -10  -10
  10   10  -10  -10
 -10  -10   10   10
 -10  -10   10   10

Master stiffness after member merge:

  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
```

Figure 5.2. Module `MergeElemIntoMasterStiff` to merge a 4×4 bar element stiffness into the master stiffness matrix, test program and its output.

argument. Because a formal argument cannot be modified, the situation is handled by copying the incoming `Kin` into `K` on entry. It is the copy which is 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 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 5.4. The result is as expected.

§5.5. ASSEMBLING THE MASTER STIFFNESS

The module `MasterStiffOfExampleTruss`, listed in the top box of Figure 5.3, makes use of the foregoing two modules: `ElemStiff2DTwoNodeBar` and `MergeElemIntoMasterStiff`, to form the

```
AssembleMasterStiffOfExampleTruss[]:=
Module[{Ke,K=Table[0,{6},{6}]},
    Ke=ElemStiff2DTwoNodeBar[{{0,0},{10,0}},{100,1}];
    K= MergeElemIntoMasterStiff[Ke,{1,2,3,4},K];
    Ke=ElemStiff2DTwoNodeBar[{{10,0},{10,10}},{100,1/2}];
    K= MergeElemIntoMasterStiff[Ke,{3,4,5,6},K];
    Ke=ElemStiff2DTwoNodeBar[{{0,0},{10,10}},{100,2*Sqrt[2]}];
    K= MergeElemIntoMasterStiff[Ke,{1,2,5,6},K];
    Return[K]
  ];
K=AssembleMasterStiffOfExampleTruss[];
Print["Master stiffness of example truss:"]; Print[K//MatrixForm];
```

```
Master stiffness of example truss:

 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
```

Figure 5.3. Module `MasterStiffOfExampleTruss` that forms the 6×6 master stiffness matrix of the example truss, test program and its output.

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 argumentless Fortran or C functions. It takes no arguments. All the example truss data is "wired in."

The output from the test program in is shown in the lower box of Figure 5.3. The output stiffness matches that in Equation (3.20), as can be expected if all fragments used so far work correctly.

§5.6. MODIFYING THE MASTER SYSTEM

Following the assembly process the master stiffness equations $\mathbf{K}\mathbf{u} = \mathbf{f}$ must be modified to account for single-freedom displacement boundary conditions. This is done through the computer-oriented equation modification process described in §3.4.2.

Module `ModifiedMasterStiffForDBC` carries out this process for the master stiffness matrix **K**, whereas `ModifiedMasterForcesForDBC` does this for the nodal force vector **f**. These two modules are listed in the top box of Figure 5.4, along with test statements. The logic of both functions, but especially that of `ModifiedMasterForcesForBC`, 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 Chapter 21.

Function `ModifiedMasterStiffnessForDBC` 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}.
`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`,

```
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 5.4. Modules `ModifiedMasterStiff` and `ModifiedMasterForce` that modify the master stiffness matrix and force vector of a truss to impose displacement BC.

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, however, 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 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 are purposedly 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 5.4, which should be self explanatory. The force vector and its modified form are printed as row vectors to save space.

```
IntForce2DTwoNodeBar[{{x1_,y1_},{x2_,y2_}},{Em_,A_},eftab_,u_]:=
  Module[ {c,s,dx=x2-x1,dy=y2-y1,L,ix,iy,jx,jy,ubar,e},
    L=Sqrt[dx^2+dy^2]; c=dx/L; s=dy/L; {ix,iy,jx,jy}=eftab;
    ubar={c*u[[ix]]+s*u[[iy]],-s*u[[ix]]+c*u[[iy]],
          c*u[[jx]]+s*u[[jy]],-s*u[[jx]]+c*u[[jy]]};
    e=(ubar[[3]]-ubar[[1]])/L; Return[Em*A*e]
];
p =IntForce2DTwoNodeBar[{{0,0},{10,10}},{100,2*Sqrt[2]},
         {1,2,5,6},{0,0,0,0,0.4,-0.2}];
Print["Member int force (numerical):"]; Print[N[p]];
p =IntForce2DTwoNodeBar[{{0,0},{L,L}},{Em,A},
         {1,2,5,6},{0,0,0,0,ux3,uy3}];
Print["Member int force (symbolic):"]; Print[Simplify[p]];
```

```
Member int force (numerical):
2.82843
Member int force (symbolic):
A Em (ux3 + uy3)
----------------
      2 L
```

Figure 5.5. Module `IntForce2DTwoNodeBar` for computing the internal force in a bar element.

§5.7. RECOVERING INTERNAL FORCES

Mathematica provides built-in matrix operations for solving a linear system of equations and multiplying matrices by vectors. Thus we do not need to write application functions for the solution of the modified stiffness equations and for the recovery of nodal forces. Consequently, the last application functions we need are those for internal force recovery.

Function `IntForce2DTwoNodeBar` listed in the top box of Figure 5.5 computes the internal force in an individual bar element. It is somewhat similar in argument sequence and logic to `ElemStiff2DTwoNodeBar` (Figure 5.1). The first two arguments are identical. Argument `eftab` provides the Element Foreedom 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. 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 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 5.5. The first test is for member (3) of the example truss using the actual nodal displacements (3.24). It also illustrates the use of the *Mathematica* built in function `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 5.6 lists a higher-level function, `IntForcesOfExampleTruss`, which has a single argument: `u`, which is the complete 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 actual node displacements (3.24) to `IntForcesOfExampleTruss`. Running the functions with the test statements produces the output

```
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 5.6. Module `IntForceOfExampleTruss` that computes internal forces in the 3 members of the example truss.

shown in the bottom box of Figure 5.6. The internal forces are $p^{(1)} = 0$, $p^{(2)} = -1$ and $p^{(3)} = 2\sqrt{2} = 2.82843$.

§5.8. PUTTING THE PIECES TOGETHER

After all this development and testing effort documented in Figures 5.1 through 5.6 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 5.7. This *driver program* uses the previously described modules

```
ElemStiff2DTwoNodeBar
MergeElemIntoMasterStiff
MasterStiffOfExampleTruss
ModifiedMasterStiffForDBC
ModifiedMasterForcesForDBC
IntForce2DTwoNodeTruss
IntForcesOfExampleTruss
```
(5.1)

These functions must have been defined (”compiled”) at the time the driver programs 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. (An even simpler procedure would to group them all in one cell, but that would make placing separate test statements difficult.)

The program listed in the top box of Figure 5.7 first assembles the master stiffness matrix through `MasterStiffOfExampleTruss`. 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

```
u=Inverse[Kmod].fmod
```

which takes advantage of two built-in *Mathematica* functions. `Inverse` returns the inverse of its matrix argument[12] The dot operator signifies matrix multiply (here, matrix-vector multiply.) The enclosing

[12] This is a highly inefficient way to solve $\mathbf{Ku} = \mathbf{f}$ if this system becomes large. It is done here to keep simplicity.

```
f={0,0,0,0,2,1};
K=AssembleMasterStiffOfExampleTruss[];
Kmod=ModifiedMasterStiffForDBC[{1,2,4},K];
fmod=ModifiedMasterForcesForDBC[{1,2,4},f];
u=Simplify[Inverse[Kmod].fmod];
Print["Computed nodal displacements:"]; Print[u];
f=Simplify[K.u];
Print["External node forces including reactions:"]; Print[f];
p=Simplify[IntForcesOfExampleTruss[u]];
Print["Internal member forces:"]; Print[p];
```

```
Computed nodal displacements:
```
$\{0, 0, 0, 0, \frac{2}{5}, -\frac{1}{5}\}$
```
External node forces including reactions:
{-2, -2, 0, 1, 2, 1}
Internal member forces:
```
$\{0, -1, 2\sqrt{2}\}$

Figure 5.7. Driver program for numerical analysis of example truss and its output.

`Simplify` function is placed to simplify the expression of vector `u` in case of symbolic calculations; it is actually redundant if all computations are numerical as in Figure 5.7.

The remaining calculations recover the node vector including reactions by the matrix-vector multiply `f = K.u` (recall that `K` contains the unmodified master stiffness matrix) and the member internal forces `p` through `IntForcesOfExampleTruss`. The program prints `u`, `f` and `p` as row vectors to conserve space.

Running the program of the top box of Figure 5.7 produces the output shown in the bottom box of that figure. The results confirm the hand calculations of Chapter 3.

```
f={0,0,0,0,fx3,fy3};
K=AssembleMasterStiffOfExampleTruss[];
Kmod=ModifiedMasterStiffForDBC[{1,2,4},K];
fmod=ModifiedMasterForcesForDBC[{1,2,4},f];
u=Simplify[Inverse[Kmod].fmod];
Print["Computed nodal displacements:"]; Print[u];
f=Simplify[K.u];
Print["External node forces including reactions:"]; Print[f];
p=Simplify[IntForcesOfExampleTruss[u]];
Print["Internal member forces:"]; Print[p];
```

```
Computed nodal displacements:
```
$\{0, 0, 0, 0, \frac{1}{10}(3\,\mathrm{fx3} - 2\,\mathrm{fy3}), \frac{1}{5}(-\mathrm{fx3} + \mathrm{fy3})\}$
```
External node forces including reactions:
{-fx3, -fx3, 0, fx3 - fy3, fx3, fy3}
Internal member forces:
```
$\{0, -\mathrm{fx3} + \mathrm{fy3}, \sqrt{2}\,\mathrm{fx3}\}$

Figure 5.8. Driver program for symbolic analysis of example truss and its output.

At this point you may wonder whether all of this is worth the trouble. After all, a hand calculation (typically helped by a programable calculator) would be quicker in terms of flow time. Writing 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 Chapter.) For larger problems, however, *Mathematica* would certainly beat hand-plus-calculator computations, the cross-over typically appearing for 10-20 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 10000 equations, *Matlab* would be the best compromise between human and computer flow time. Beyond 10000 equations a program in a low-level language, such as C or Fortran, would be most efficient in terms of computer time.

One distinct advantage of computer algebra systems appear 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 5.8, press the Enter key and shortly get the symbolic answer shown in the bottom box of Figure 5.8. This is the answer to an infinite number of numerical problems. Although one may try to undertake such studies by hand, the likelyhood 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 5.1

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 this Universe to store that, but barely ...

Homework Exercises for Chapter 5

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 5 Index and click on the link). Open this Notebook file using version 4.0 or a later one. The first eight cells contain the modules and test statements listed in the top boxes of Figures 5.1–8. The first six of these are marked as *initialization cells*. Before running driver programs, they should be executed by picking up Kernel → Initialize → Execute Initialization. Verify that the output of those six cells agrees with that shown in the bottom boxes of Figures 5.1–6. Then execute the driver programs in Cells 7–8 by clicking on the cells and hitting <Enter>, and compare the output with that shown in Figures 5.7–8. If the output checks out, you may proceed to the Exercises.

EXERCISE 5.1

[C:10] Explain why the `Simplify` command in the test statements of Figure 5.1 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[L^2]` to `L` unless one specifies the sign of `L` in the `Simplify` command?

EXERCISE 5.2

[C:10] Explain the logic of the `For` loops in the merge function `MergeElemIntoMasterStiff` of Figure 5.2. What does the operator `+=` do?

EXERCISE 5.3

[C:10] Explain the reason behind the use of `Length` in the modules of Figure 5.4. Why not simply set `nk` and `np` to 6 and 3, respectively?

EXERCISE 5.4

[C:15] Of the seven modules listed in Figures 5.1 through 5.6, 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 5.5

[C:20] Modify the modules `MasterStiffOfExampleTruss`, `IntForcesOfExampleTruss` and the driver program of Figure 5.7 to solve numerically the three-node, two-member truss of Exercise 3.7. Verify that the output reproduces the solution given for that problem.

EXERCISE 5.6

[C:25] Expand the logic `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 E5.1. Mystery program for Exercise 5.7.

EXERCISE 5.7

[C:20] Explain what the program of Figure E5.1 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?

6 Constructing MoM Members

TABLE OF CONTENTS

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 or matrix operations. Non-simplex elements are presented as a recipe, since 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* connected at *joints*.

§6.1. FORMULATION METHOD

§6.1.1. The Class of MoM Members

MoM-based formulations are largely restricted to *intrinsically one-dimensional members*. These are structural components one of whose dimensions, called the *longitudinal dimension*, is significantly larger than the other two, which are called the *tranverse dimensions*. Such members are amenable to the simplified structural theories developed in MoM textbooks. We shall study only *straight* members with geometry defined by the two end joints. The member *cross sections* are defined by the intersection of planes normal to the longitudinal dimension with the member. See Figure 6.1. Note that although the individual member will be idealized as being one-dimensional in its intrinsic or local coordinate system, it is generally part 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.

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

The structures fabricated with these kinds of 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 in cases where the nature of the geometry and loading allows the reduction of the analysis to one plane of symmetry, which is chosen to be the $\{x, y\}$ plane. Plane trusses and plane frameworks are examples of such simplifications.

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 6.1.

[1] Mechanics of Materials was called Strength of Materials in older texts. The scope of this subject includes bars, beams, shafts, arches, thin plates and shells, but only one-dimensional models are covered in basic undergraduate courses. MoM involves *ab initio* kinematic assumptions such as "plane sections remain plane."

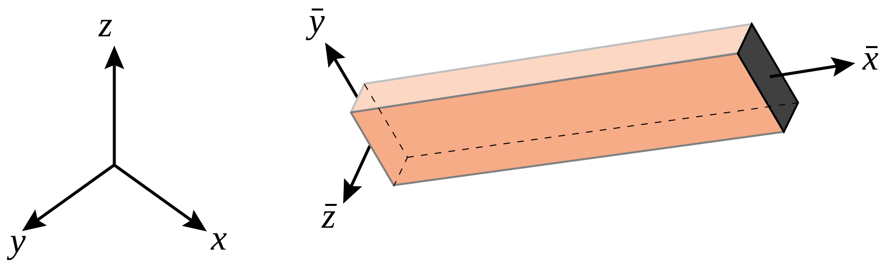


Figure 6.1. A Mechanics of Materials (MoM) member is a structural element one of whose dimensions (the longitudinal dimension) is significantly larger than the other two. Local axes $\{\bar{x}, \bar{y}, \bar{z}\}$ are chosen as indicated. Although the depicted member is prismatic, some applications utilize tapered or stepped members, the cross section of which varies as a function of $\bar{x}$.

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 6.2.

§6.1.2. End Quantities and Degrees of Freedom

The set of mathematical variables used to interconnect 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 linked by the member stiffness equations.

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

The vector of joint forces $\bar{\mathbf{f}}$ collects 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 z-moment $\bar{m}_{zj}$.

§6.1.3. Internal Quantities

Internal quantities are mechanical actions that take place inside 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 $\mathbf{p}$. This set globally characterizes the forces resisted by the material. They are also called generalized stresses in structural theory. Stresses at any point in the member 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 axial deformation, or with the total elongation. Pairs that mutually correspond in the sense of the

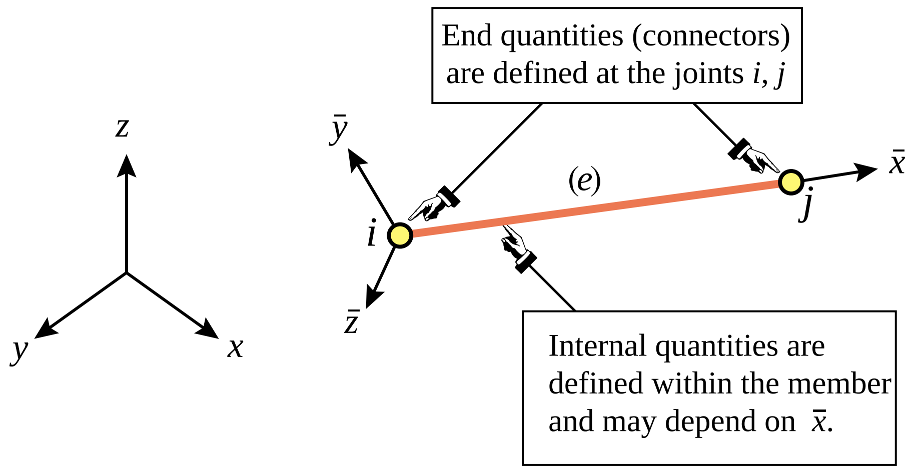

End quantities (connectors) are defined at the joints i, j
z
y
x
$\bar{y}$
$\bar{z}$
$\bar{x}$
i
j
(e)
Internal quantities are defined within the member and may depend on $\bar{x}$.

Figure 6.2. 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. The internal quantities characterize the stresses and deformations in the member.

Principle of Virtual Work are called *conjugate*. Unlike the case of end quantities, conjugacy is not a mandatory requirement but simplifies some derivations.

§6.1.4. The Discrete Field Equations

The matrix equations that connect $\bar{\mathbf{u}}$, $\mathbf{v}$, $\mathbf{p}$ and $\mathbf{f}$ 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 compability conditions, also called the deformation-displacement or strain-displacement equations:

$$\boxed{\mathbf{v} = \mathbf{B}\bar{\mathbf{u}}.} \tag{6.1}$$

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:

$$\boxed{\mathbf{p} = \mathbf{S}\mathbf{v}.} \tag{6.2}$$

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

$$\boxed{\bar{\mathbf{f}} = \mathbf{A}^T\mathbf{p}.} \tag{6.3}$$

where the transpose of $\mathbf{A}$ is used for convenience.[2]

[2] If $\mathbf{p}$ is a function of $\bar{x}$ the relation is of differential type and is studied in §6.3.

$$\begin{tikzcd}[column sep=huge, row sep=huge]
\bar{\mathbf{u}} \arrow[r, "\bar{\mathbf{f}} = \mathbf{A}^T \mathbf{S}\,\mathbf{B}\,\mathbf{u} = \bar{\mathbf{K}}\,\bar{\mathbf{u}}"] \arrow[d, "\mathbf{v} = \mathbf{B}\,\bar{\mathbf{u}}", "\text{Kinematic}"'] & \bar{\mathbf{f}} \\
\mathbf{v} \arrow[r, "\mathbf{p} = \mathbf{S}\,\mathbf{v}", "\text{Constitutive}"'] & \mathbf{p} \arrow[u, "\bar{\mathbf{f}} = \mathbf{A}^T \mathbf{p}", "\text{Equilibrium}"']
\end{tikzcd}$$

Figure 6.3. Tonti diagram of the three discrete field equations (6.1)–(6.3) and the stiffness equation (6.4) for a *simplex* MoM element. Internal and end quantities appear inside the orange and yellow boxes, respectively.

These equations can be presented graphically as shown in Figure 6.3. This is a discrete variant of the so-called *Tonti diagrams*, which represent the governing equations as arrows linking boxes containing kinematic and static quantities.

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

A	Equilibrium
S	Rigidity, material, constitutive[3]
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.

§6.2. STIFFNESS EQUATIONS FOR SIMPLEX MOM ELEMENTS

In 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, and the derivation of the element stiffness equations is particularly simple.

Under the constancy assumption, elimination of the interior quantities **p** and **v** from (6.1)-(6.3) yields the element stiffness relation

$$\bar{\mathbf{f}} = \mathbf{A}^T\mathbf{S}\mathbf{B}\,\bar{\mathbf{u}} = \bar{\mathbf{K}}\bar{\mathbf{u}}. \tag{6.4}$$

Hence the element stiffness matrix is

$$\boxed{\bar{\mathbf{K}} = \mathbf{A}^T\mathbf{S}\mathbf{B}.} \tag{6.5}$$

The four preceding matrix equations are diagrammed in Figure 6.3.

[3] 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 **E**, as in §6.3.

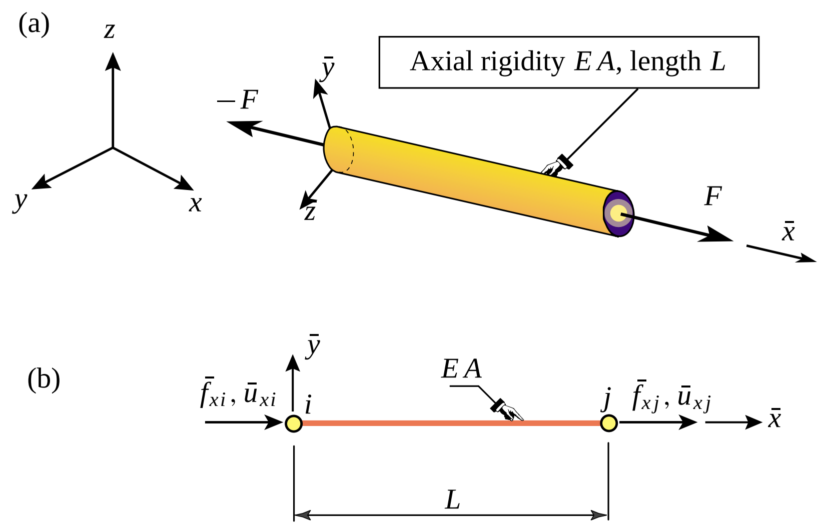

(a)
z
y
x
$\bar{y}$
$-F$
Axial rigidity EA, length L
$\bar{z}$
F
$\bar{x}$
(b)
$\bar{y}$
EA
$\bar{f}_{xi}, \bar{u}_{xi}$
i
j
$\bar{f}_{xj}, \bar{u}_{xj}$
$\bar{x}$
L

Figure 6.4. The prismatic truss (also called bar) member: (a) individual member in 3D space, (b) idealization as generic member.

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

$$\boxed{\bar{\mathbf{K}} = \mathbf{B}^T \mathbf{S} \mathbf{B}.} \tag{6.6}$$

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

REMARK 6.1

If $\bar{\mathbf{f}}$ and $\bar{\mathbf{u}}$ are conjugate 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.

§6.2.1. The Truss Element Revisited

The simplest example of a MoM element is the prismatic truss (bar) element already derived in Chapter 2. See Figure 6.4. This qualifies as a simplex MoM element since 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 since 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 studied below. They illustrate that the resulting element stiffness equations coalesce, as can be expected, since the external quantities are 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 $v = d$ and $p = F$, respectively. The chain of discrete field equations is easily constructed:

$$
\begin{aligned}
d &= [\,-1 \quad 1\,] \begin{bmatrix} \bar{u}_{xi} \\ \bar{u}_{xj} \end{bmatrix} = \mathbf{B}\bar{\mathbf{u}}, \\
F &= \frac{EA}{L} d = S d, \\
\bar{\mathbf{f}} &= \begin{bmatrix} \bar{f}_{xi} \\ \bar{f}_{xj} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} F = \mathbf{A}^T F
\end{aligned}
\tag{6.7}
$$

Hence

$$
\bar{\mathbf{K}} = \mathbf{A}^T S \mathbf{B} = S \mathbf{B}^T \mathbf{B} = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{6.8}
$$

Note that $\mathbf{A} = \mathbf{B}$ because F and d are conjugate.

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 $[\,-1 \quad 1\,]/L$ whereas S becomes EA. Matrix $\mathbf{A}$ does not change. The product $\mathbf{A}^T S \mathbf{B}$ gives the same $\bar{\mathbf{K}}$ as in (6.8), 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 in 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 the mean axial stress $\bar{\sigma} = F/A$, (which is *not* conjugate to $\bar{e}$) is taken as internal force measure. Now $\mathbf{B} = [\,-1 \quad 1\,]/L$, $S = E$ and $\mathbf{A}^T = A[\,-1 \quad 1\,]$. The product $\mathbf{A}^T S \mathbf{B}$ gives again the same $\bar{\mathbf{K}}$ shown in (6.8).

§6.2.2. The Spar Element

The *spar* or *shear-web* member has two joints, i and j. It 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 6.5. This element is often used in modeling aircraft wing structures, as illustrated in Figure 6.6.

The active degrees of freedom for the generic element of length L depicted in Figure 6.5(b) are $\bar{u}_{yi}$ and $\bar{u}_{yj}$. Let G be the shear modulus and A_s the effective shear area.[4] The shear rigidity is GA_s. As deformation measure the mean shear strain $\gamma = V/(GA_s)$ is chosen. The deformation-displacement, constitutive, and equilibrium equations are

$$
\begin{aligned}
\gamma &= \frac{1}{L} [\,-1 \quad 1\,] \begin{bmatrix} \bar{u}_{yi} \\ \bar{u}_{yj} \end{bmatrix} = \mathbf{B}\bar{\mathbf{u}}, \\
V &= GA_s \gamma = S\gamma, \\
\bar{\mathbf{f}} &= \begin{bmatrix} \bar{f}_{yi} \\ \bar{f}_{yj} \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix} V = \mathbf{A}^T V.
\end{aligned}
\tag{6.9}
$$

Therefore the member stiffness equations are

$$
\bar{\mathbf{f}} = \begin{bmatrix} \bar{f}_{yi} \\ \bar{f}_{yj} \end{bmatrix} = \mathbf{A}^T S \mathbf{B} \bar{\mathbf{u}} = \frac{GA_s}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{u}_{yi} \\ \bar{u}_{yj} \end{bmatrix} = \bar{\mathbf{K}} \bar{\mathbf{u}}. \tag{6.10}
$$

[4] A concept developed in Mechanics of Materials. For a narrow rectangular cross section, $A_s = 5A/6$.

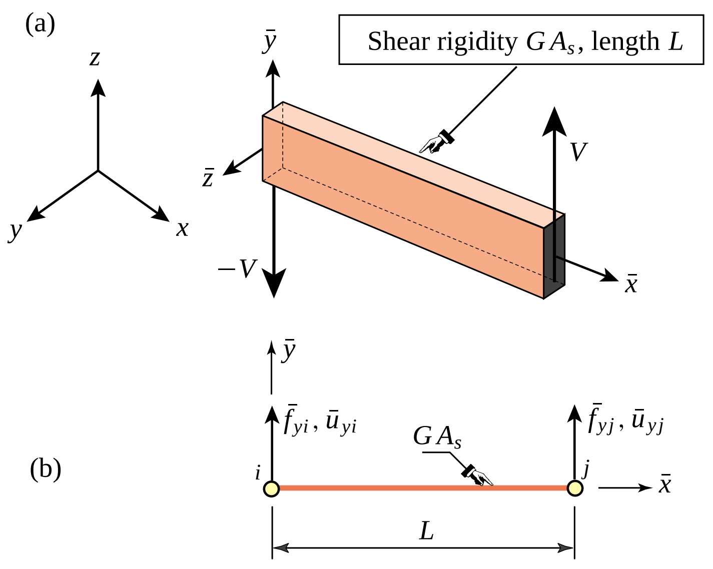

(a)
z
y
x
$\bar{y}$
$\bar{z}$
$\bar{x}$
Shear rigidity GA_s, length L
V
$-V$
(b)
$\bar{y}$
$\bar{f}_{yi}, \bar{u}_{yi}$
$\bar{f}_{yj}, \bar{u}_{yj}$
GA_s
i
j
$\bar{x}$
L

Figure 6.5. The prismatic spar (also called shear-web) member: (a) individual member in 3D space, (b) idealization as generic member.

Note that $\mathbf{A} \neq \mathbf{B}$ as V and γ are not conjugate. However, the difference is easily adjusted for.

The equations (6.10) may be augmented with zero rows and columns to bring in the node displacements in the $\bar{x}$ and $\bar{z}$ directions, and then referred to the global axis $\{x, y, z\}$ through a congruential transformation.

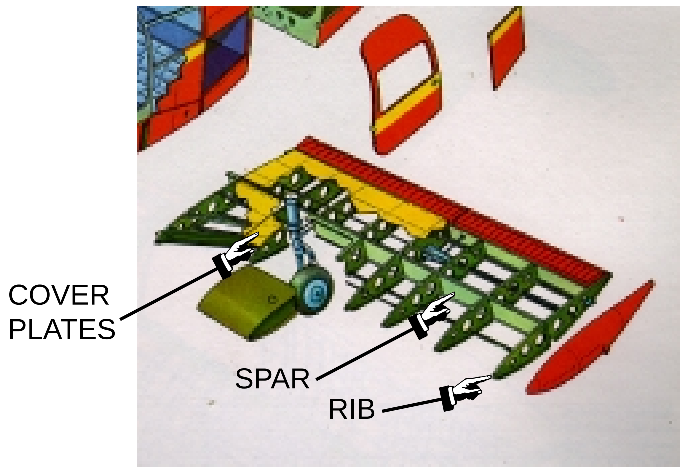
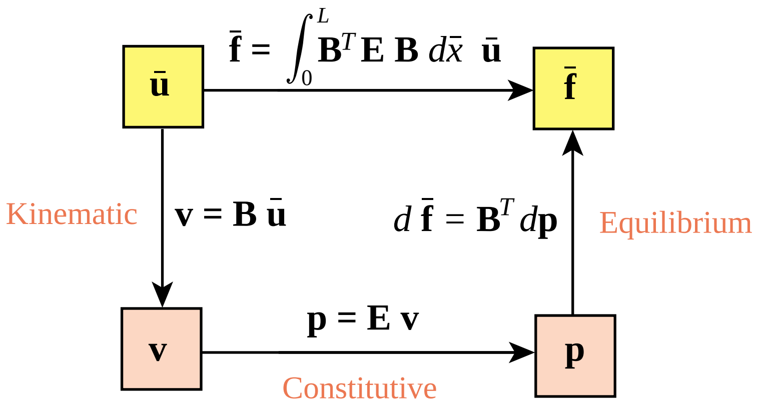


Figure 6.6. Spar members in aircraft wing (Piper Cherokee).



$\bar{\mathbf{f}} = \int_0^L \mathbf{B}^T \mathbf{E}\, \mathbf{B}\, d\bar{x}\ \ \bar{\mathbf{u}}$
$\bar{\mathbf{u}}$
$\bar{\mathbf{f}}$
Kinematic
$\mathbf{v} = \mathbf{B}\, \bar{\mathbf{u}}$
$d\, \bar{\mathbf{f}} = \mathbf{B}^T d\mathbf{p}$
Equilibrium
$\mathbf{v}$
$\mathbf{p} = \mathbf{E}\, \mathbf{v}$
$\mathbf{p}$
Constitutive

Figure 6.7. Diagram of the discrete field equations for a non-simplex MoM element.

§6.3. MEMBERS WITH VARIABLE INTERNAL QUANTITIES

In the general case, the internal quantities $\mathbf{p}$ and $\mathbf{v}$ may vary over the member; that is, depend on $\bar{x}$. This dependence may be due to element type, variable cross section, or both. Consequently one or more of the matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{S}$, depend on $\bar{x}$. Such elements are called *non-simplex*.

If the element falls under this category, the straightforward matrix multiplication recipe (6.5) cannot be used to construct the element stiffness matrix $\bar{\mathbf{K}}$. This fact 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, $\mathbf{K}$ must be independent of $\bar{x}$ because it relates the end quantities $\bar{\mathbf{u}}$ and $\bar{\mathbf{f}}$.

The derivation of non-simplex MoM elements requires the use of work principles of mechanics, for example the Principle of Virtual Work or PVW. More care must be exercised in the selection of conjugate internal quantities. The following rules can be justified through the variational arguments discussed in Part II. They are stated here only as recipe.

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[5] that the force equilibrium equation exists only in a differential sense: $\mathbf{B}^T d\mathbf{p} = d\bar{\mathbf{f}}$. Here d denotes differentiation with respect to $\bar{x}$.[6]

Rule 3. The constitutive matrix $\mathbf{E}(\bar{x})$ that relates $\mathbf{p}(\bar{x})$ to $\mathbf{v}(\bar{x})$ must be symmetric.[7]

Relations that emanate from these rules are diagrammed in Figure 6.7. Internal quantities are now eliminated using the differential equilibrium relation:

$$d\bar{\mathbf{f}} = \mathbf{B}^T d\mathbf{p} = \mathbf{B}^T \mathbf{p}\, d\bar{x} = \mathbf{B}^T \mathbf{E}\, \mathbf{v}\, d\bar{x} = \mathbf{B}^T \mathbf{E}\mathbf{B}\bar{\mathbf{u}}\, d\bar{x} = \mathbf{B}^T \mathbf{E}\mathbf{B}\, d\bar{x}\, \bar{\mathbf{u}}. \tag{6.11}$$

[5] The proof is done by equating the virtual work of a slice of length $d\bar{x}$: $d\bar{\mathbf{f}}^T.\delta\bar{\mathbf{u}} = d\mathbf{p}^T.\delta\mathbf{v} = d\mathbf{p}^T.(\mathbf{B}\,\bar{\mathbf{u}}) = (\mathbf{B}^T.d\mathbf{p})^T.\delta\bar{\mathbf{u}}$. Since $\delta\bar{\mathbf{u}}$ is arbitrary, $\mathbf{B}^T d\mathbf{p} = d\bar{\mathbf{f}}$.

[6] 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 Equation (6.12).

[7] Note that symbol $\mathbf{E}$ replaces the $\mathbf{S}$ of the previous section. $\mathbf{E}$ applies to a specific cross section, $\mathbf{S}$ to the entire member. See Exercise 6.4.

Integrating both sides over the member length L yields

$$\bar{\mathbf{f}} = \int_0^L d\bar{\mathbf{f}} = \int_0^L \mathbf{B}^T \mathbf{E} \mathbf{B} \, d\bar{x} \; \bar{\mathbf{u}} = \bar{\mathbf{K}} \bar{\mathbf{u}}, \tag{6.12}$$

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

$$\boxed{\bar{\mathbf{K}} = \int_0^L \mathbf{B}^T \, \mathbf{E} \, \mathbf{B} \; d\bar{x}} \tag{6.13}$$

The result (6.13) will be justified in Part II of the course. It will be seen there that this formula applies to arbitrary displacement-assumed finite elements in any number of dimensions. It is applied to the derivation of the stiffness equations of the plane beam element in Chapter 13.

Homework Exercises for Chapter 6

Constructing MoM Members

EXERCISE 6.1

[A:15] Formulate a prismatic *shaft member* of length L that can only transmit a torque T along the longitudinal ($\bar{x}$) direction. The only active degrees of freedom of this member are the twist-angle rotations $\bar{\theta}_{xi}$ and $\bar{\theta}_{xj}$ of the end joints about $\bar{x}$. The constitutive equation furnished by Mechanics of Materials is $T = (GJ/L)\phi$, where GJ is the torsional rigidity and $\phi = \bar{\theta}_{xj} - \bar{\theta}_{xi}$ is the relative twist angle.

The result should be

$$\begin{bmatrix} \bar{m}_{xi} \\ \bar{m}_{xj} \end{bmatrix} = \frac{GJ}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \bar{\theta}_{xi} \\ \bar{\theta}_{xj} \end{bmatrix} \tag{E6.1}$$

in which $\bar{m}_{xi} = -T$ and $\bar{m}_{xj} = T$ are the nodal moments about $\bar{x}$.

EXERCISE 6.2

[A:10] Explain how to select the deformation variable v (paired to V) of the spar member formulated in §6.2.2, so that $\mathbf{A} = \mathbf{B}$.

EXERCISE 6.3

[A/C:25] Derive the 4×4 global element stiffness matrix of a prismatic spar element in a two dimensional Cartesian system $\{x, y\}$. Start from the local stiffness (6.10) and proceed as in §3.2.1 and §3.2.2 for the bar element. Since $\bar{\mathbf{K}}$ in (6.10) is 2×2, $\mathbf{T}$ is 2×4. Show that $\mathbf{T}$ consists of rows 2 and 4 of the matrix of (3.2).

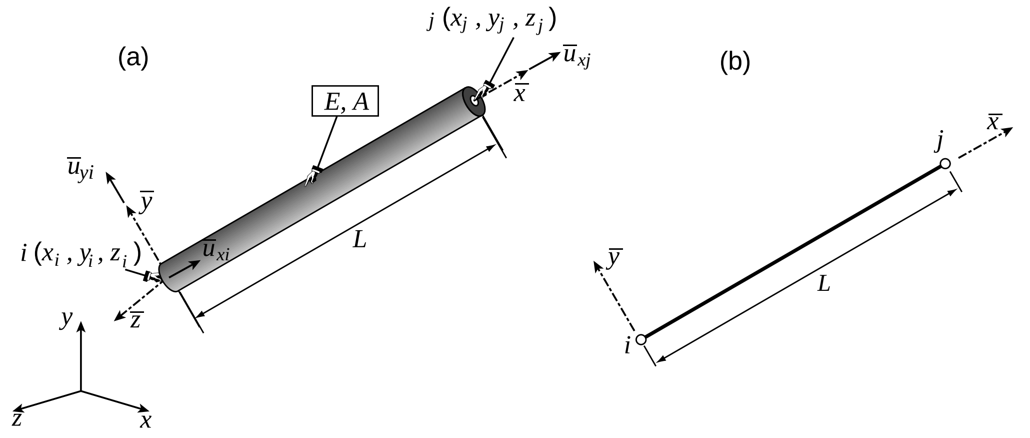


Figure E6.1. Bar element in 3D for Exercise 6.4.

EXERCISE 6.4

[A+N:15] A bar element moving in three dimensional space is 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 E6.1. The 2×6 displacement transformation matrix $\mathbf{T}$ relates $\bar{\mathbf{u}}^{(e)} = \mathbf{T}\mathbf{u}^{(e)}$. Here $\bar{\mathbf{u}}^{(e)}$ contains the local axial displacements $\bar{u}_{xi}$ and $\bar{u}_{xj}$ whereas $\mathbf{u}^{(e)}$ contains the global displacements $u_{xi}, u_{yi}, u_{zi}, u_{xj}, u_{yj}, u_{zj}$. Show that

$$\mathbf{T} = \frac{1}{L} \begin{bmatrix} x_{ji} & y_{ji} & z_{ji} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{ji} & y_{ji} & z_{ji} \end{bmatrix} = \begin{bmatrix} c_{xji} & c_{yji} & c_{zji} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{xji} & c_{yji} & c_{zji} \end{bmatrix} \tag{E6.2}$$

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

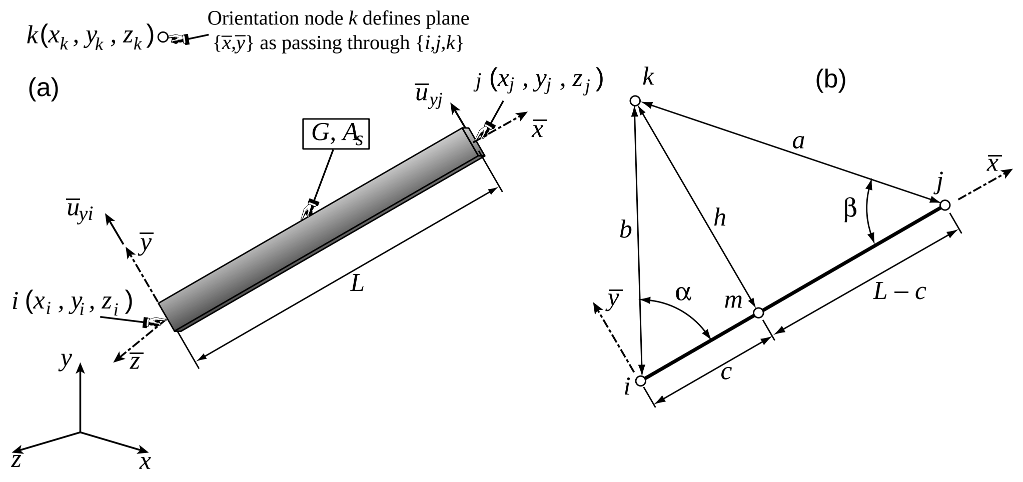


Figure E6.2. Spar element in 3D for Exercise 6.5.

EXERCISE 6.5

[A+N:25 (5+20)] 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 E6.2. The problem is that axis $\bar{y}$, which defines the direction of shear force transmission, is not uniquely defined by i and j.[8] Most FEM programs use the *orientation node* method to complete the definition. A third node k, not colinear with i and j, is provided by the user. Nodes $\{i, j, k\}$ define the $\{\bar{x}, \bar{y}\}$ plane and consequently $\bar{z}$. The projection of k on line ij is point m and the distance $h > 0$ from m to k is called h as shown in Figure E6.2. The 2×6 displacement transformation matrix $\mathbf{T}$ 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

$$\mathbf{T} = \frac{1}{h}\begin{bmatrix} x_{km} & y_{km} & z_{km} & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{km} & y_{km} & z_{km} \end{bmatrix} = \begin{bmatrix} c_{xkm} & c_{ykm} & c_{zkm} & 0 & 0 & 0 \\ 0 & 0 & 0 & c_{xkm} & c_{ykm} & c_{zkm} \end{bmatrix} \tag{E6.3}$$

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.

(b) Work out the formulas to compute the coordinates of point m in terms of the coordinates of $\{i, j, k\}$. Using the notation of Figure E6.2(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), $\cos\alpha = (a^2 - b^2 - L^2)/(2bL)$, $\cos\beta = (b^2 - a^2 - L^2)/(2aL)$, $c = b\cos\alpha$, $L - c = a\cos\beta$, $x_m = x_i(L-c)/L + x_j c/L$, etc. Evaluate $\mathbf{T}$ for a bar going from node i at $\{1, 2, 3\}$ to node j at $\{3, 8, 6\}$. with k at $\{4, 5, 6\}$.

EXERCISE 6.6

[A:15] If the matrices $\mathbf{B}$ and $\mathbf{E}$ are constant over the element length L, show that expression (6.13) of the element stiffness matrix for a variable-section element reduces to (6.6), in which $\mathbf{S} = L\mathbf{E}$.

EXERCISE 6.7

[A:20] Explain in detail the quick derivation of footnote 5. (Knowledge of the PVW is required to do this exercise.)

[8] This full specification of the local system is also required for 3D beam elements.

EXERCISE 6.8

[A:15] Explain how thermal effects can be generally incorporated in the constitutive equation (6.2) to produce an initial force vector.

EXERCISE 6.9

[A:20] Consider a non-simplex MoM element in which $\mathbf{E}$ varies with $\bar{x}$ but $\mathbf{B}$ is constant. From (6.13) show that

$$\bar{\mathbf{K}} = L\mathbf{B}^T\bar{\mathbf{E}}\,\mathbf{B}, \quad \text{with} \quad \bar{\mathbf{E}} = \frac{1}{L}\int_0^L \mathbf{E}(\bar{x})\,d\bar{x} \tag{E6.4}$$

Here $\bar{\mathbf{E}}$ is an element-averaged constitutive matrix. Apply this result to form $\bar{\mathbf{K}}$ for a truss member of tapered cross section area $A(\bar{x})$. Assume that the area is 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 joints i and j, respectively. Take $\mathbf{B}$ to be the same as in the prismatic case discussed in §6.2.1.

EXERCISE 6.10

[A/C+N:30] A prismatic bar element in 3D space is referred to a global coordinate system $\{x, y, z\}$, as in Figure E6.1. 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 $L = \sqrt{x_{21}^2 + y_{21}^2 + z_{21}^2}$. Show that the element stiffness matrix in *global* coordinates can be compactly written[10]

$$\mathbf{K}^{(e)} = \frac{EA}{L^3}\mathbf{B}^T\mathbf{B} \tag{E6.5}$$

where

$$\mathbf{B} = [\,-x_{21} \quad -y_{21} \quad -z_{21} \quad x_{21} \quad y_{21} \quad z_{21}\,] \tag{E6.6}$$

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 of the course:

```
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};
```

[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

$$\tfrac{1}{2}(L^2 - L_0^2) = x_{21}(u_{x2} - u_{x1}) + y_{21}(u_{y2} - u_{y1}) + z_{21}(u_{z2} - u_{z1}) + R \approx \mathbf{Bu}$$

in which R is a quadratic function of node displacements which is therefore dropped in the small-displacement linear theory. But

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

also because of small displacements. Hence the small axial strain is $e = \Delta L/L = (1/L^2)\mathbf{Bu}^{(e)}$, which begins the Tonti diagram. Next is $F = EA\,e$, and finally you should show that force equilibrium at nodes requires $\mathbf{f}^{(e)} = (1/L)\mathbf{B}^T F$. Multiplying through you get (E6.5). Another way is to start from the local stiffness (6.8) and transform to global using the transformation matrix (E6.2).

```
    {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;
  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.

7 FEM Modeling: Introduction

TABLE OF CONTENTS

Chapters 2 through 6 cover material technically known as Matrix Structural Analysis or MSA. This is a subject that historically preceded the Finite Element Method. This Chapter starts the coverage of the FEM proper, which is distinguished from MSA by the more prevalent role of continuum and variational mechanics.

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

§7.1. FEM TERMINOLOGY

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

Classical analytical mechanics is that invented by Euler and Lagrange and developed by Hamilton and Jacobi as a systematic formulation of Newtonian mechanics. Its objects of attention are models of mechanical systems ranging from particles composed of sufficiently large of molecules, through airplanes, to the Solar System.[1] The spatial configuration of any such system is described by its *degrees of freedom*. 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 of the model is finite, the model is called *discrete*, and *continuous* otherwise.

Because FEM is a discretization method, the number of degrees of freedom of a FEM model is necessarily finite. The freedoms are collected in a column vector called $\mathbf{u}$. This vector is generally 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*. 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.

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 stiffness equations:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}.} \tag{7.1}$$

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

[1] For cosmological scales, such as the full Universe, the general theory of relativity is necessary. For the sub-particle world, quantum mechanics is required.

[2] In variational mathematics this is called a duality pairing.

Table 7.1. Physical Significance of Vectors u and f in Miscellaneous FEM Applications

Application Problem	*State (DOF) vector* **u** *represents*	*Conjugate vector* **f** *represents*
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

The physical significance of the vectors **u** and **f** varies according to the application being modeled, as illustrated in Table 7.1.

If the relation between forces and displacements is linear but not homogeneous, equation (7.1) generalizes to

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I\,.} \tag{7.2}$$

Here $\mathbf{f}_I$ is the initial node force vector introduced in Chapter 4 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 7.1, 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 illsutrates terminology. It shows the three key simulation steps are: *idealization*, *discretization* and *solution*,and indicates that 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 ill-posed problems.

The idealization and discretization steps, briefly mentioned in Chapter 1, deserve further discussion. The solution step is dealt with in the last Part of this course.

§7.2. IDEALIZATION

Idealization passes from the actual system to a model thereof. This is the most important step in engineering practice.

§7.2.1. 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:

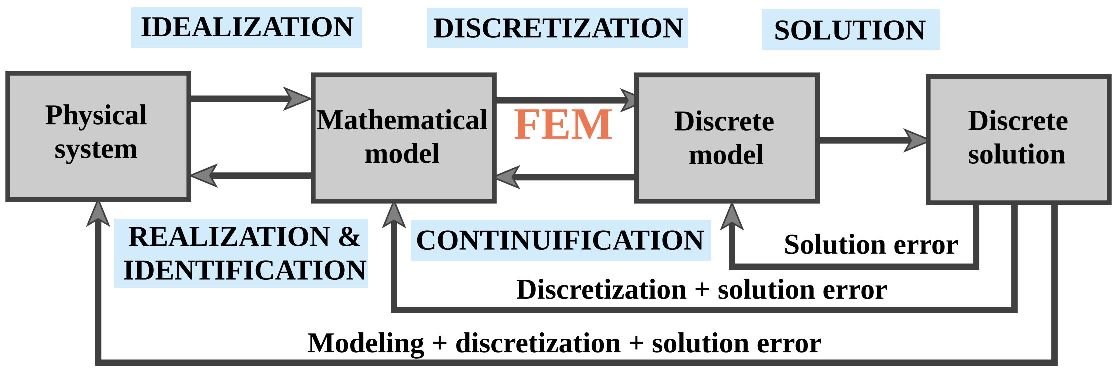


Figure 7.1. A simplified view of the physical simulation process, primarily useful to illustrate modeling terminology.

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

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. 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.[3]

§7.2.2. Mathematical Models

Mathematical modeling, or *idealization*, is a process by which the engineer passes from the actual physical system under study, to a *mathematical model* of the system, where the term *model* is understood in the wider sense defined above.

The process is called *idealization* because the mathematical model is necessarily an abstraction of the physical reality. (Note the phrase *aspects of behavior* in the foregoing definition.) The analytical or numerical results obtained for the mathematical model are re-interpreted in physical terms only for those aspects.[4]

To give an example on the choices an engineer may face, suppose that the structure is a flat plate structure subjected to transverse loading. Here is a 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.
3. A *moderately thick* plate model, for example 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. The decision may be different in static

[3] A problem-definition file or a stress plot are examples of the latter.

[4] Whereas idealization can be reasonably taught in advanced design courses, the converse process of "realization" or "parameter identification" generally requires considerable physical understanding and maturity that can only be gained through professional experience.

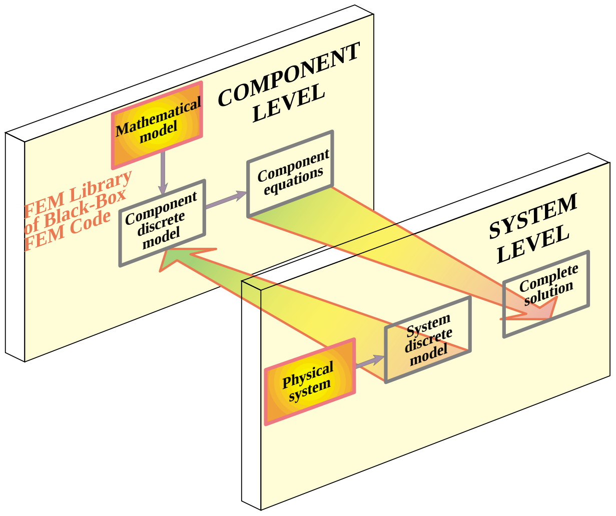


Figure 7.2. A reproduction of Figure 1.5 with some relabeling. Illustrates implicit modeling: picking elements from an existing FEM code consents to an idealization.

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 controlled. This is achieved by “filtering out” physical details that are not relevant to the 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, choosing a mathematical model is equivalent to choosing an information filter.

§7.2.3. Implicit vs. Explicit Modeling

As noted the diagram of Figure 7.1 is an oversimplification of engineering practice. The more common scenario is that pictured in Figures 1.2, 1.4 and 1.5. The latter is reproduced in Figure 7.2 for convenience.

A common scenario in industry is: you have to analyze a structure or a substructure, 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.

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 can 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.

§7.3. DISCRETIZATION

§7.3.1. Purpose

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.

At this point one faces the choice of trying 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 symbolic 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, often have to be 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.[5] The practical way out is numerical simulation. Here is where finite element methods and the digital computer 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*. 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 course deals only with static problems, we need not consider the time dimension and are free to concentrate on *spatial discretization*.

[5] This statement has to be tempered in two respects. First, the wider availability and growing power of computer algebra systems, discussed in Chapter 5, 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 to reduce the dimensionality of the problem and to facilitate parameter studies. Important examples are provided by Fourier analysis, perturbation and boundary-element methods.

§7.3.2. Error Sources and Approximation

Figure 7.1 tries to convey 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 such *model verification and validation* requires access to and comparison with experimental results. These may be either scarce, or unavailable in the case of a new product.

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.

§7.3.3. Other Discretization Methods

It was stated in the first chapter 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 gas dynamics.

§7.4. THE FINITE ELEMENT METHOD

§7.4.1. Interpretation

The finite element method (FEM) is the dominant discretization technique in structural mechanics. As noted in Chapter 1, the FEM can be interpreted from either a physical or mathematical standpoint. The treatment has so far emphasized 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.

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.

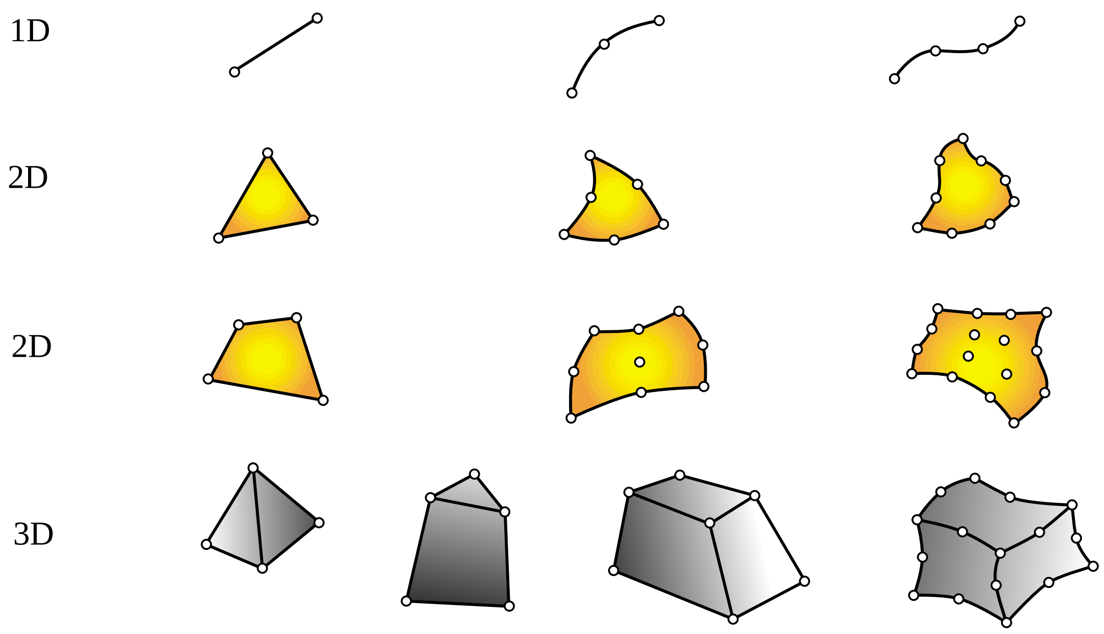


Figure 7.3. Typical finite element geometries in one through three dimensions.

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*.

§7.4.2. 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 disconnection and localization steps, which were described for the truss example in Chapter 2. This 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. After these two steps we can consider *generic* elements: a bar element, a beam element, and so on. From the standpoint of 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 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.

Intrinsic Dimensionality. Elements can have one, two or three space dimensions.[6] There are also special elements with zero dimensionality, such as lumped springs or point masses.

Nodal points. Each element possesses a set of distinguishing points called *nodal points* or *nodes* for short. Nodes serve two purposes: definition of element geometry, and home for degrees of freedom. They are usually located at the corners or end points of elements, as illustrated in Figure

[6] In dynamic analysis, time appears as an additional dimension.

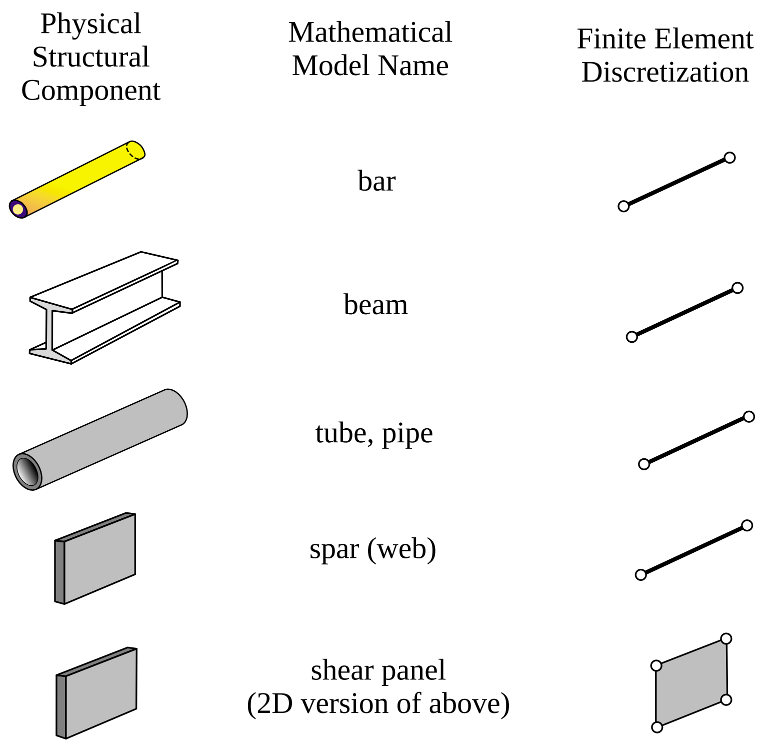


Figure 7.4. Examples of primitive structural elements.

7.3; in the so-called refined or higher-order elements nodes are also placed on sides or faces, as well as the interior of the element.

Geometry. The geometry of the element is defined by the placement of the 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 three common shapes are tetrahedra, pentahedra (also called wedges or prisms), and hexahedra (also called cuboids or "bricks"). See Figure 7.3.

Degrees of freedom. The 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 nodal 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.

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.

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.

Fabrication properties. For mechanical elements these are fabrication properties which have been

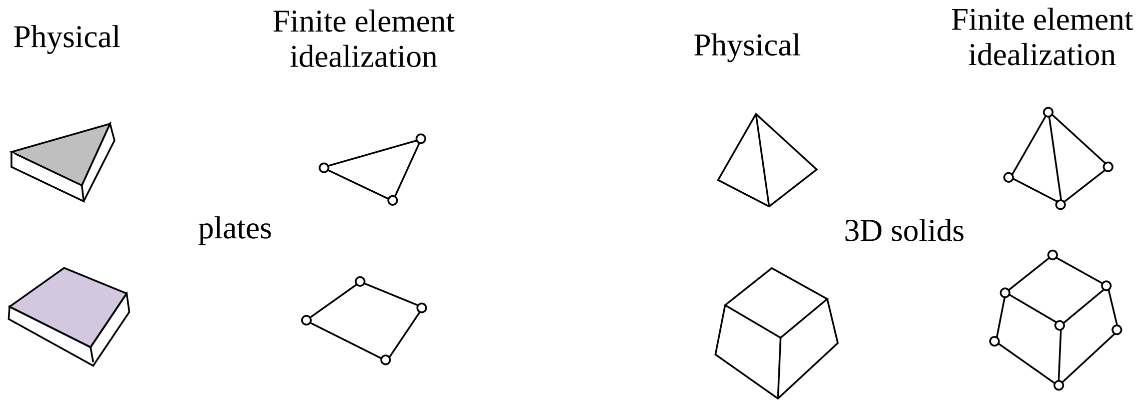


Figure 7.5. Continuum element examples.

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.

This data is used by the element generation subroutines to compute element stiffness relations in the local system.

§7.5. 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 many points that appear over and over in subsequent sections and provides insight into advanced modeling techniques such as hierarchical breakdown and global-local analysis.

§7.5.1. Primitive Structural Elements

These resemble fabricated structural components, and are often drawn as such; see Figure 7.4. The qualifier *primitive* is used to distinguish them from macroelements, which is another element class described below. It means that they are not decomposable into simpler elements. These elements are usually derived from Mechanics-of-Materials arguments, and are better understood from a physical, rather than mathematical, standpoint. Examples are the elements discussed in Chapter 6: bars, cables, beams, shafts, spars.

§7.5.2. 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 7.5.

§7.5.3. 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

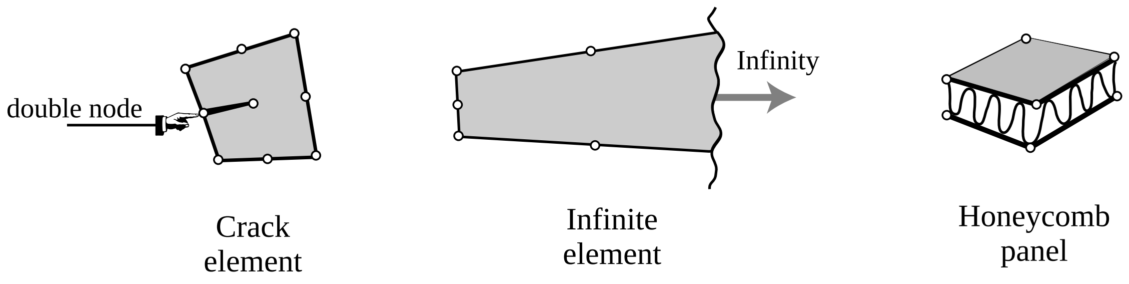


Figure 7.6. Special element examples.

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 7.6.

§7.5.4. 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 7.7.

§7.5.5. Substructures

Also called structural modules and superelements. These are macroelements with a well defined structural function, typically obtained by cutting the complete structure into functional components. Examples: the wings and fuselage in an airplane, the deck and cables in a suspension bridge. It should be noted that the distinction between complete structures, substructures and macroelements is not clear-cut. The term *superelement* is often used in a collective sense to embrace all levels beyond that of primitive elements. This topic is further covered in Chapter 11.

§7.6. 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 processs involves two basic steps:

Globalization. The element equations are transformed to a common *global* coordinate system.

Merge. The element stiffness equations are merged into the master stiffness equations by appropriate indexing and entry addition.

The computer implementation of this process is not necessarily as simple as the hand calculations of the truss example suggest. The master stiffness relations in practical cases may involve thousands (or even millions) of degrees of freedom. To conserve storage and processing time the use of sparse matrix techniques as well as peripheral storage is required. But this inevitably increases the programming complexity. The topic is elaborated upon in the last part of this course.

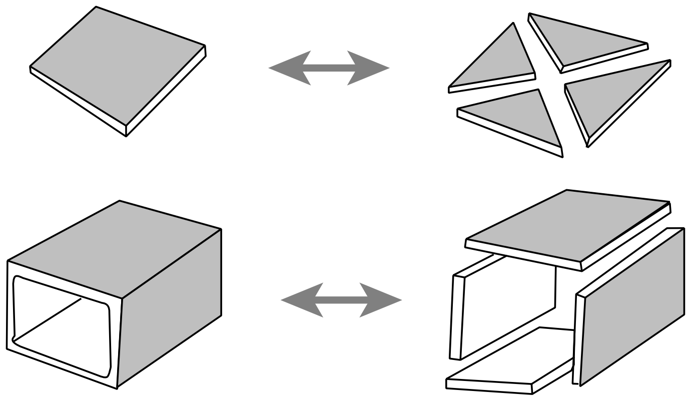


Figure 7.7. Macroelement examples.

§7.7. 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. One of the biggest hurdles a FEM newcomer faces is the understanding and proper handling of boundary conditions. In the present Section we summarize some basic rules for treating boundary conditions. The following Chapter provides specific rules and examples.

§7.7.1. Essential and Natural B.C.

The important thing to remember is that boundary conditions (BCs) come in two basic flavors, essential and natural.

Essential BCs are those that directly affect the degrees of freedom, and are imposed on the left-hand side vector **u**.

Natural BCs are those that do not directly affect the degrees of freedom, and are imposed on the right-hand side vector **f**.

The mathematical justification for this distinction requires use of the variational calculus, and is consequently relegated to Part II of the course. 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.

§7.7.2. Boundary Conditions in Structural Problems

In mechanical problems, essential boundary conditions are those that involve *displacements* (but not strain-type displacement derivatives). The support conditions for the truss 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 the number of equations to be solved.

Ignorable freedoms. To suppress displacements that are irrelevant to the problem. (In classical dynamics these are called *ignorable coordinates*.) Even experienced users of finite element programs are sometimes baffled by this kind. An example are rotational degrees of freedom normal to 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*, which can be notoriously difficult to handle from a numerical standpoint. These will be covered in Chapters 9 and 10.

8
FEM Modeling: Mesh, Loads and BCs

TABLE OF CONTENTS

This Chapter continues the exposition of finite element modeling principles. After some general recommendations, it provides guidelines on layout of finite element meshes, conversion of distributed loads to node forces, and how to handle the simplest forms of support boundary conditions. The following Chapters deal with more complicated forms of boundary conditions called multifreedom constraints.

The presentation is "recipe oriented" and illustrated by specific examples. All examples are from structural mechanics; most of them are two-dimensional. No attempt is given at a rigorous justification of rules and recommendations, because that would require mathematical tools beyond the scope of this course.

§8.1. GENERAL RECOMMENDATIONS

The general rules that should guide you in the use of commercial or public FEM packages, are:

- 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, and there is little point in using complicated models that will not survive design iterations. The time for refined models is when the design has stabilized and you have a better view picture of the underlying physics, possibly reinforced by experiments or observation.

§8.2. 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.

§8.2.1. Mesh Refinement

Use a relatively fine (coarse) discretization in regions where you expect a high (low) *gradient* of strains and/or stresses. Regions to watch out for high gradients are:

- Near entrant corners, 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 8.1 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.

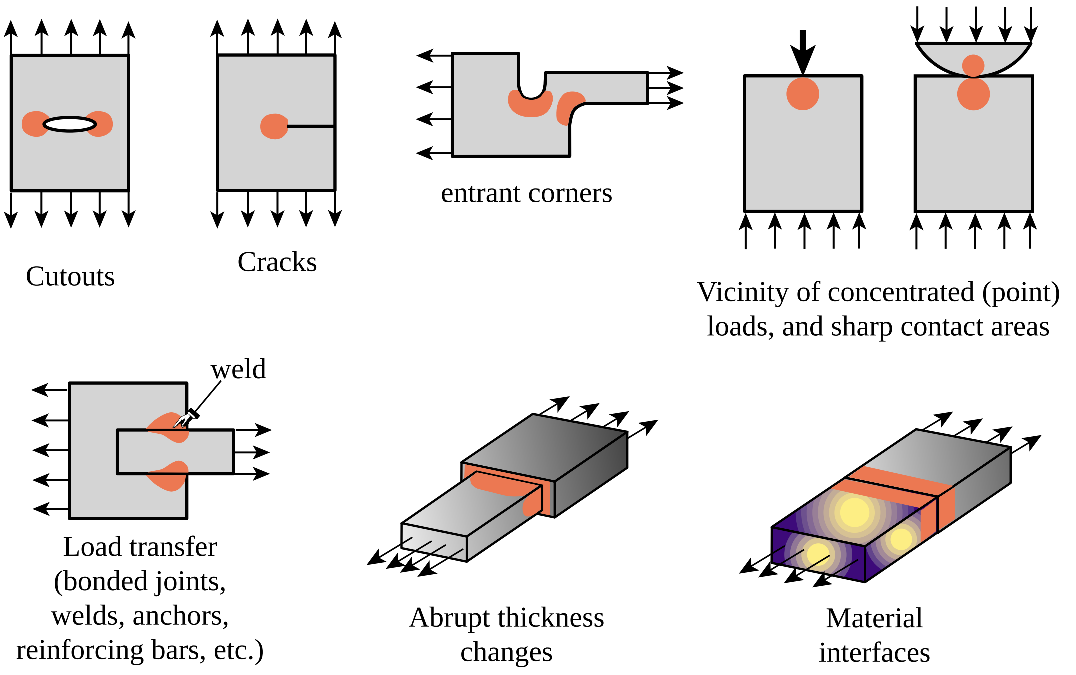

Cutouts
Cracks
entrant corners
Vicinity of concentrated (point) loads, and sharp contact areas
weld
Load transfer (bonded joints, welds, anchors, reinforcing bars, etc.)
Abrupt thickness changes
Material interfaces

Figure 8.1. Some situations where a locally refined finite element discretization (in the red-colored areas) is recommended.

§8.2.2. Element Aspect Ratios

When discretizing two and three dimensional problems, try to avoid finite elements of high aspect ratios: elongated or “skinny” elements, as the ones illustrated in Figure 8.2.[1] 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.

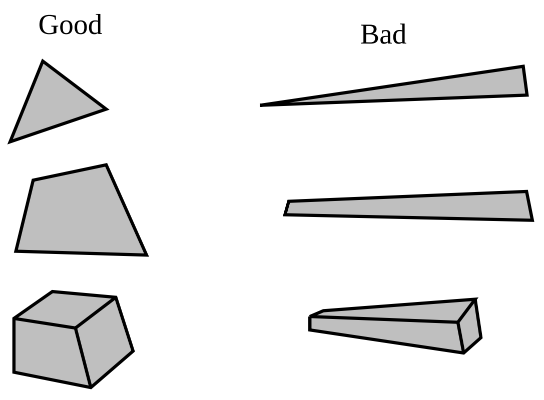


Figure 8.2.Elements of good and bad aspect ratios.

[1] The aspect ratio of a two- or three-dimensional element is the ratio between its largest and smallest dimension.

REMARK 8.1

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.

§8.2.3. 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 8.3.

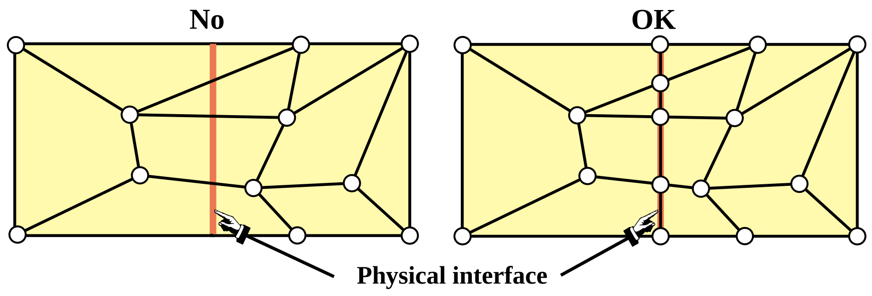


Figure 8.3. Illustration of the rule that elements should not cross material interfaces.

§8.2.4. Preferred Shapes

In two-dimensional 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 three dimensional FE modeling, prefer strongly bricks over wedges, and wedges over tetrahedra. The latter should be used only if there is no viable alternative. The main problem with tetrahedra and wedges is that they can produce wrong stress results even if the displacement solution looks reasonable.

§8.3. DIRECT LUMPING OF DISTRIBUTED LOADS

In practical structural problems, distributed loads are more common than concentrated (point) loads.[2] 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, 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.

[2] In fact, one of the objectives of a good design is to avoid or alleviate stress concentrations produced by concentrated forces.

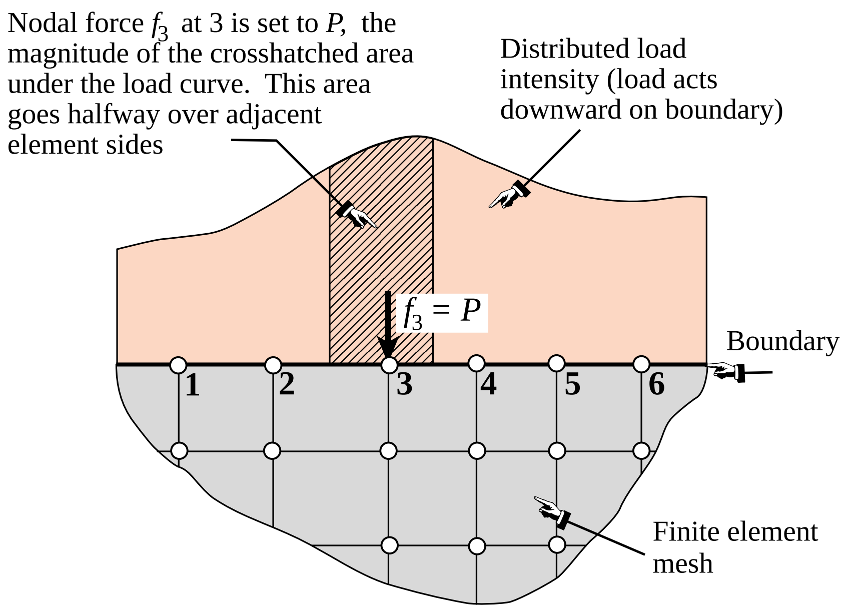

Nodal force f_3 at 3 is set to P, the magnitude of the crosshatched area under the load curve. This area goes halfway over adjacent element sides
Distributed load intensity (load acts downward on boundary)
$f_3 = P$
Boundary
1
2
3
4
5
6
Finite element mesh

Figure 8.4. NbN direct lumping of distributed load, illustrated for a 2D problem.

A derived type: line loads, result from the integration of surface loads along one transverse direction, or of volume loads along two transverse directions. 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 eventually end up in the right-hand side of the master stiffness equations.

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 more mathematically impeccable but complicated variational approach. Two variants of this technique are described below for distributed surface loads.

§8.3.1. Node by Node (NbN) Lumping

The node by node (NbN) lumping method is graphically explained in Figure 8.4. This example shows a distributed surface loading acting on 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 8.4, each boundary node is assigned a *tributary region* around it that extends halfway to the 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 computation of centroids, as required in the EbE technique discussed in the next subsection. For this reason it is often preferred in hand computations. It can be extended to three-dimensional meshes as well as volume loads.[3] It should

[3] The computation of tributary areas and volumes can be done through the so-called Voronoi diagrams.

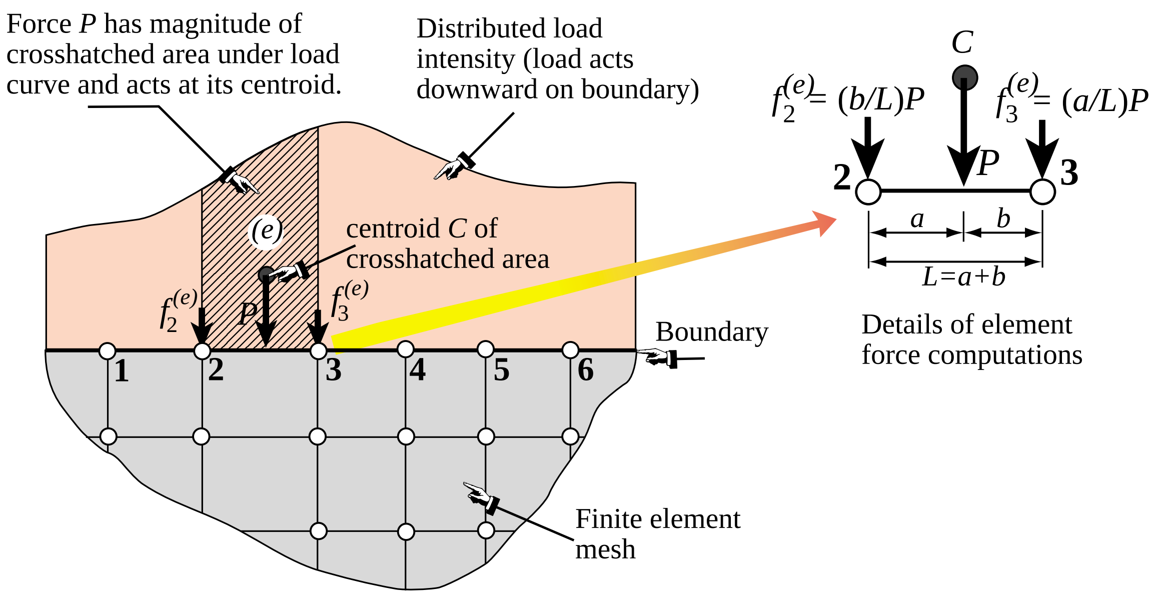

Force P has magnitude of crosshatched area under load curve and acts at its centroid.
Distributed load intensity (load acts downward on boundary)
C
$f_2^{(e)} = (b/L)P$
$f_3^{(e)} = (a/L)P$
2
P
3
a
b
L=a+b
(e)
centroid C of crosshatched area
$f_2^{(e)}$
P
$f_3^{(e)}$
Boundary
Details of element force computations
1
2
3
4
5
6
Finite element mesh

Figure 8.5. EbE direct lumping of distributed load, illustrated for a 2D problem.

be avoided, however, when the applied forces vary rapidly (within element length scales) or act only over portions of the tributary regions.

§8.3.2. Element by Element (EbE) 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 8.5, which shows details of the computation over segment 2-3. The total force at node 3, for instance, would that contributed by segments 2-3 and 3-4.

If applicable, the EbE procedure is more accurate than NbN lumping. In fact it agrees with the consistent node lumping for simple elements that possess only corner nodes. In those cases it is not affected by the sharpness of the load variation and can be used for point loads that are not applied at the nodes.

The 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 approach is preferable.

§8.4. 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 Chapter we discuss next 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.

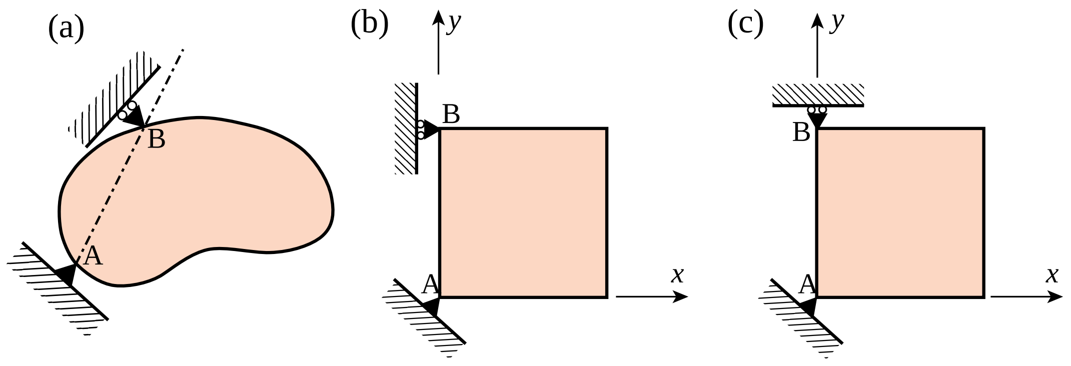


Figure 8.6. Examples of restraining a body against two-dimensional rigid body motions.

In structural problems formulated by the DSM, the recipe of §7.7.1 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.

§8.5. 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.[4] The resulting boundary conditions are often called *motion constraints*. In what follows we analyze two- and three-dimensional motions separately.

§8.5.1. Supporting Two Dimensional Bodies

Figure 8.6 shows two-dimensional bodies that displace in the plane of the paper. If a body is not restrained, an applied load will cause infinite displacements. Regardless of the loading conditions, the structure must be restrained against two translations and one rotation. Consequently the minimum number of constraints that has to be imposed in two dimensions is *three*.

In Figure 8.4, 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 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 displacement constraints.

Engineers call A and B *reaction-to-ground points*. This means that if the supports are conceptually removed, the applied loads are automatically balanced by reactive forces at points A and B, in accordance with Newton's third law. Additional freedoms may be removed to model greater restraint by the environment. However, Figure 8.6(a) does illustrate the *minimal* number of constraints.

[4] 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.

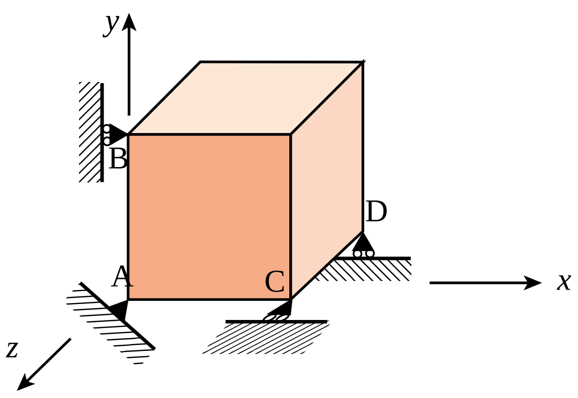


Figure 8.7. Suppressing RBM freedoms in a three-dimensional body.

Figure 8.6(b) shows a simplified version of Figure 8.6(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 6(c), it becomes ineffective in constraining the infinitesimal rotational motion about point A because the rolling direction is normal to AB. The configuration of Figure 6(c) is called a *kinematic mechanism*, and can be flagged by a singular modified stiffness matrix.

§8.5.2. Supporting Three Dimensional Bodies

Figure 8.7 illustrates the extension of the freedom deletion concept to three dimensions. The minimal number of freedoms that have to be deleted is now *six* and many combinations are possible. In the example of the figure, all three degrees of freedom at point A have been deleted to prevent rigid body translations. 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.

§8.6. 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.[5]

§8.6.1. Visualization

Recognition of symmetry and antisymmetry conditions can be done by either visualization of the displacement field, or by imagining certain rotational ot 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 8.8(a). Alternatively, a 180° rotation of the body about the symmetry line reproduces exactly the original problem.

[5] Even if the conditions are not explicitly applied through BCs, they provide valuable checks on the computed solution.

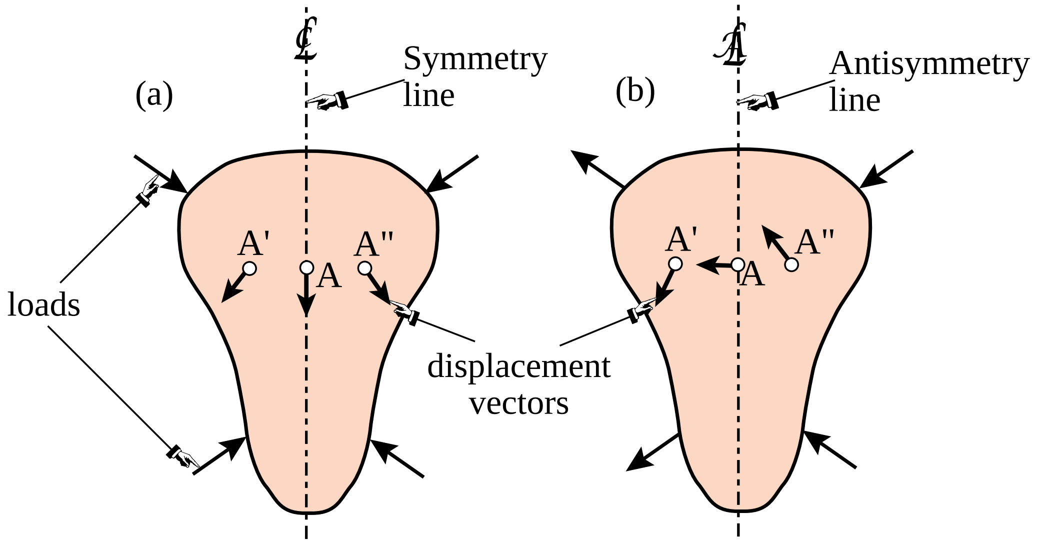


Figure 8.8. Visualizing symmetry and antisymmetry lines.

An *antisymmetry line* can be recognized by the displacement pattern illustrated in Figure 8.8(b). 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) and *rotational symmetry* can be treated in a similar manner, but will not be discussed here.

§8.6.2. 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.

Consider the plate structure shown in Figure 8.9(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 8.8(b).

On the other hand if the loading is *antisymmetric*, as shown in Figure 8.10(a), then the x axis becomes an *antisymmetry line* because none of the $y = 0$ points can move along the x direction. The boundary conditions to be imposed on the finite element model are also different, as shown in Figure 8.10(b).

REMARK 8.2

For the antisymmetric loading case, one node point has to be constrained against vertical motion. The choice is arbitrary and amounts only to an adjustment on the overall (rigid-body) vertical motion. In Figure 8.10(b) the center point C has been chosen to be that vertically-constrained node. 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.

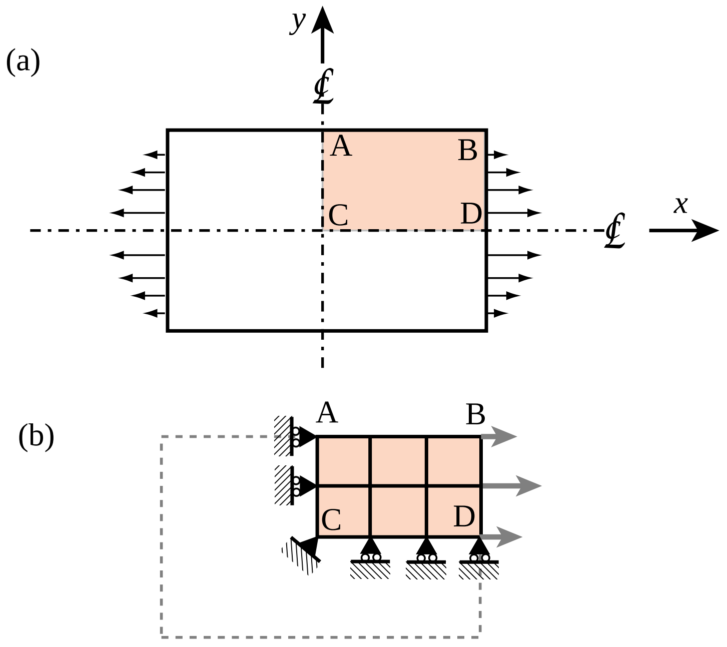


Figure 8.9. A doubly symmetric structure under symmetric loading.

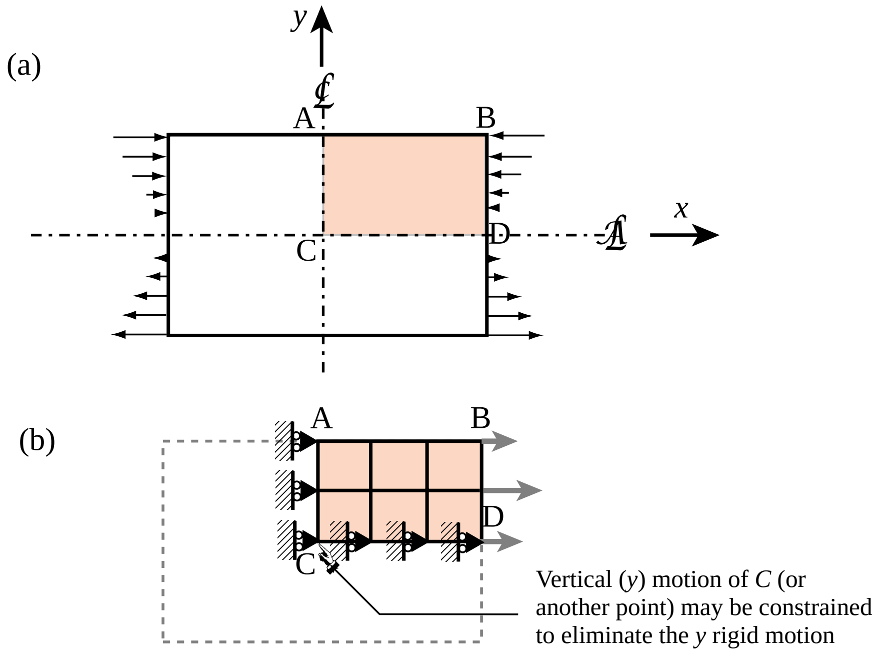

(a)
y
℄
A
B
C
D
A
x
(b)
A
B
D
C
Vertical (y) motion of C (or another point) may be constrained to eliminate the y rigid motion

Figure 8.10. A doubly symmetric structure under antisymmetric loading.

Homework Exercises for Chapters 7 and 8

FEM Modeling

EXERCISE 8.1

[D:10] The plate structure shown in Figure E8.1 is loaded and deforms in the plane of the figure. The applied load at D and the supports at I and N extend over a fairly narrow area. Give a list of 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.

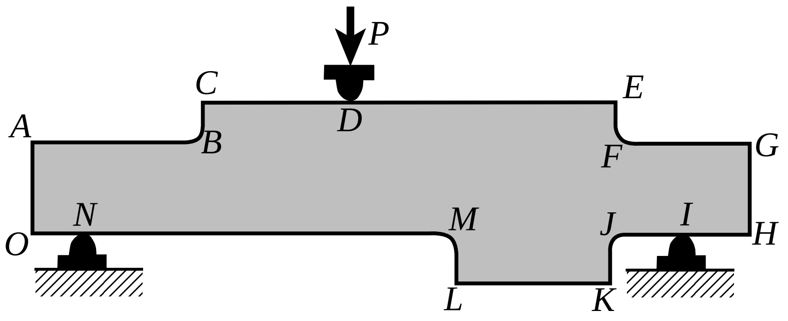


Figure E8.1. Plate structure for Exercise 8.1.

EXERCISE 8.2

[D:15] Part of a two-dimensional FE mesh has been set up as indicated in Figure E8.2. 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.

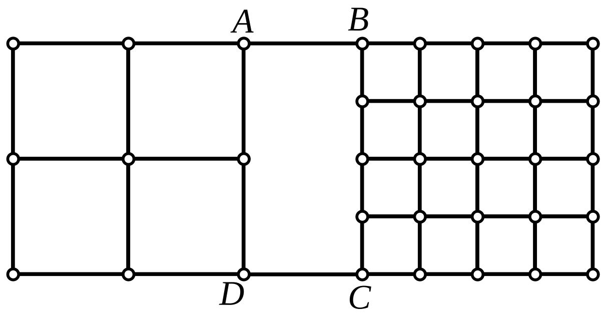


Figure E8.2. Plate structure for Exercise 8.2.

EXERCISE 8.3

[A:15] Compute the "lumped" nodal forces f_1, f_2, f_3 and f_4 equivalent to the linearly-varying distributed surface load q for the finite element layout defined in Figure E8.3. Use both NbN and EbE lumping. For example, $f_1 = 3q/8$ for NbN. Check that $f_1 + f_2 + f_3 + f_4 = 6q$ for both schemes (why?). Note that q is given as a force per unit of vertical length.

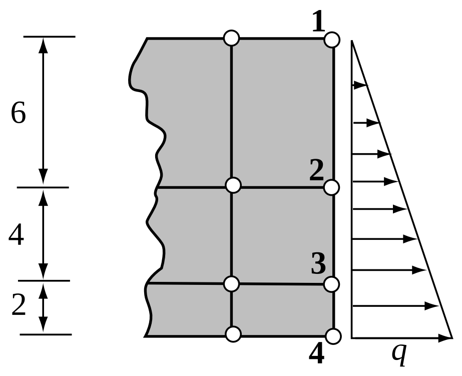


Figure E8.3. Mesh layout for Exercise 8.3.

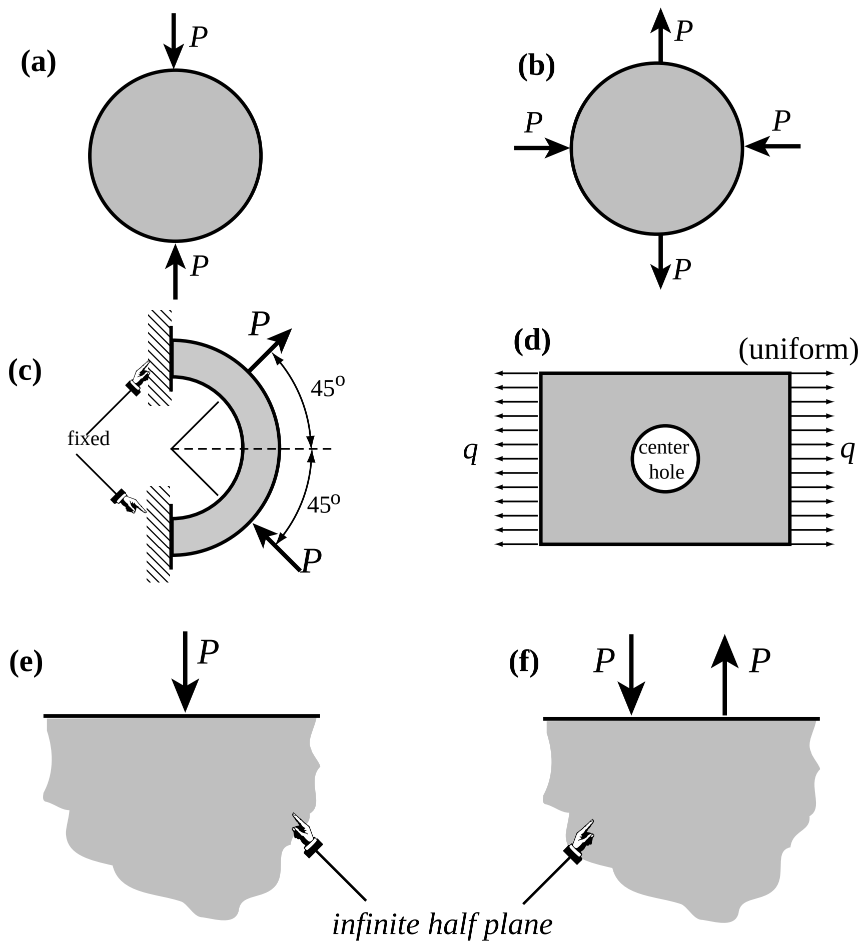


Figure E8.4. Problems for Exercise 8.4.

EXERCISE 8.4

[D:15] Identify the symmetry and antisymmetry lines in the two-dimensional problems illustrated in Figure E8.4. 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

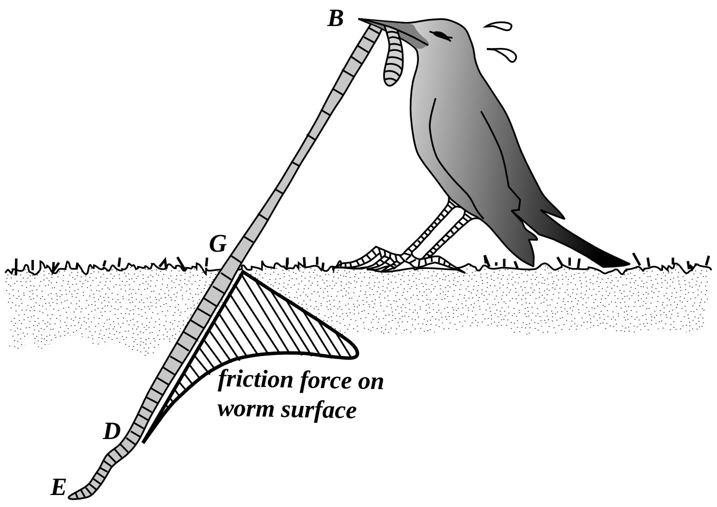


Figure E8.5. The hungry bird.

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.[6]

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 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 8.5

[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 E8.5.

After a long struggle, however, the worm wins. While hungrily looking for a smaller one your thoughts wonder to FEM and how the bird 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.

(a) 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.

(b) On the above model, how would you represent boundary conditions, applied forces and friction forces?

(c) Next you want a more refined anaysis of the worm that distinguishes skin and insides. What type of finite element model would be appropriate?

(d) (Advanced) Finally, point out what need to be added to the model of (c) to include the soil as an elastic medium.

Briefly explain your decisions. Dont write equations.

[6] 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.

EXERCISE 8.6

[A/D:20] Explain from kinematics why two antisymmetry lines in 2D cannot cross at a finite point. As a corollary, investigate whether it is possible to have more than one antisymmetric line in a 2D elasticity problem.

EXERCISE 8.7

[A/D:15] Explain from kinematics why a symmetry line and an antisymmetry line must cross at right angles.

EXERCISE 8.8

[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 C is fixed.

EXERCISE 8.9

[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 planes 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 8.10

[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 8.11

[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 8.12

[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)

9
MultiFreedom Constraints I

TABLE OF CONTENTS

§9.1. CLASSIFICATION OF CONSTRAINT CONDITIONS

In previous Chapters we have considered structural support conditions that are mathematically expressable as constraints on individual degrees of freedom:

$$\boxed{\text{nodal displacement component} = \text{prescribed value.}} \tag{9.1}$$

These are called *single-freedom constraints*. Chapters 3 and 4 explain how to incorporate constraints of this form into the master stiffness equations, using hand- or computer-oriented techniques. The displacement BCs studied in Chapter 8, including the modeling of symmetry and antisymmetry, lead to constraints of this form.

For example:

$$u_{x4} = 0, \qquad u_{y9} = 0.6. \tag{9.2}$$

These are two single-freedom constraints. The first one is homogeneous and the second one non-homogeneous.

§9.1.1. MultiFreedom Constraints

The next step up in complexity involves *multifreedom equality constraints*, or *multifreedom constraints* for short, the last name being often acronymed to MFC. These are functional equations that connect *two or more* displacement components:

$$\boxed{F(\text{nodal displacement components}) = \text{prescribed value,}} \tag{9.3}$$

where function F vanishes if all its nodal displacement arguments do. Equation (9.3), where 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 transfering all terms that depend on displacement components to the left-hand side, the right-hand side — the "prescribed value" in (9.3) — is zero. It is called *non-homogeneous* otherwise.

In this and next Chapter only linear constraints will be considered. Furthermore more attention is devoted to the homogeneous case, because it arises more often in practice.

REMARK 9.1

The most general constraint class is that of inequality constraints, such as $u_{y5} - 2u_{x2} \geq 0$. 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.

§9.1.2. MFC Examples

Here are three examples of MFCs:

$$u_{x2} = \tfrac{1}{2} u_{y2}, \quad u_{x2} - 2u_{x4} + u_{x6} = 0.25, \quad (x_5 + u_{x5} - x_3 - u_{x3})^2 + (y_5 + u_{y5} - y_3 - u_{y3})^2 = 0. \quad (9.4)$$

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 in the $\{x, y\}$ plane is to remain constant. This kind of constraint appears in geometrically nonlinear analysis of structures, which is a topic beyond the scope of this course.

§9.1.3. Matrix Forms

Matrix forms of linear MFCs are often convenient for compact notation. An individual constraint such as the second one in (9.4) can be written

$$[\,1 \quad -2 \quad 1\,] \begin{bmatrix} u_{x2} \\ u_{x4} \\ u_{x6} \end{bmatrix} = 0.25 \qquad (9.5)$$

or, in direct matrix notation:

$$\bar{\mathbf{a}}_i \bar{\mathbf{u}}_i = b_i, \qquad \text{(no sum on } i\text{)} \qquad (9.6)$$

in which index i ($i = 1, 2, \ldots$) 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 b_i is the right hand side scalar. The bar over **a** and **u** distinguishes (9.6) from the expanded form (9.8) 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 (9.5) is part of a two-dimensional finite element model with 12 freedoms: $u_{x1}, u_{y1}, \ldots u_{y6}$, the left-hand side row vector may be expanded with nine zeros as follows

$$[\,0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad -2 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0\,] \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ \vdots \\ u_{y6} \end{bmatrix} = 0.25, \qquad (9.7)$$

in which case the matrix notation

$$\mathbf{a}_i \mathbf{u} = b_i \qquad (9.8)$$

is used. Finally, all multifreedom constraints expressed as (9.8) may be collected into a single matrix relation:

$$\mathbf{A}\mathbf{u} = \mathbf{g}, \qquad (9.9)$$

where rectangular matrix **A** is formed by stacking the $\mathbf{a}_i$'s as rows and column vector **g** is formed by stacking the b_is as entries. If there are 12 degrees of freedom in **u** and 5 multifreedom constraints then **A** will be 5×12.

§9.2. 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. The modification process is also called *constraint application* or *constraint imposition*. The modified system is the one submitted to the equation solver.

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 there 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 for completeness but is considered an advanced topic.

Conceptually, imposing MFCs is not different from the procedure discussed in Chapters 3 and 4 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 — particularly 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 9.2

The three methods are also applicable, as can be expected, to the simpler case of a single-freedom constraint such as (9.2). 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 9.3

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.

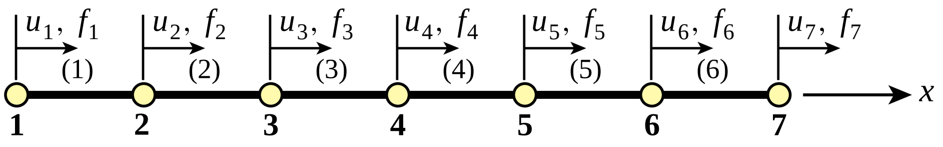


Figure 9.1. A one-dimensional problem discretized with six bar finite elements. The seven nodes may move only along the x direction. Subscript x is omitted from the u's and f's for simplicity.

§9.3. THE EXAMPLE STRUCTURE

The one-dimensional finite element discretization shown in Figure 9.1 will be used throughout 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

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \tag{9.10}$$

or

$$\mathbf{K}\mathbf{u} = \mathbf{f}. \tag{9.11}$$

The nonzero stiffness coefficients K_{ij} in (9.10) depend on the bar rigidity properties. For example, if $E^{(e)}A^{(e)}/L^{(e)} = 100$ for each element $e = 1, \ldots, 6$, then $K_{11} = K_{77} = 100$, $K_{22} = \ldots = K_{66} = 200$, $K_{12} = K_{23} = \ldots = K_{67} = -100$. 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 are to move by the same amount:

$$u_2 = u_6. \tag{9.12}$$

Passing all node displacements to the right hand side gives the canonical form:

$$\boxed{u_2 - u_6 = 0.} \tag{9.13}$$

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]

[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.

We now study the imposition of constraint (9.13) on the master equations (9.11) by the methods mentioned above. In this Chapter the master-slave method is treated. Two more general methods: penalty augmentation and Lagrange multiplier adjunction, are discussed in the following Chapter.

§9.4. 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 congruential 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.

§9.4.1. A One-Constraint Example

The mechanics of the process is best seen by going through an example. To impose (9.13) choose u_6 as slave and u_2 as master. Relate the original unknowns $u_1, \ldots u_7$ to the new set in which u_6 is missing:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix}, \tag{9.14}$$

This is the required transformation relation. In compact form:

$$\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}. \tag{9.15}$$

Replacing (9.15) into (9.11) and premultiplying by $\mathbf{T}^T$ yields the modified system

$$\boxed{\hat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T\mathbf{K}\mathbf{T}, \qquad \hat{\mathbf{f}} = \mathbf{T}^T\mathbf{f}.} \tag{9.16}$$

Carrying out the indicated matrix multiplications yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 \\ f_3 \\ f_4 \\ f_5 \\ f_7 \end{bmatrix}, \tag{9.17}$$

Equation (9.17) 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 (9.12).

REMARK 9.4

The form of modified system (9.16) can be remembered by a simple mnemonic rule: premultiply both sides of $\mathbf{u} = \mathbf{T}\,\hat{\mathbf{u}}$ by $\mathbf{T}^T\,\mathbf{K}$, and replace $\mathbf{K}\,\mathbf{u}$ by $\mathbf{f}$ on the right hand side.

REMARK 9.5

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 congruential transformation is then nothing more than the elimination of u_4 by striking out rows and columns from the master stiffness equations.

REMARK 9.6

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:

$$\begin{bmatrix} K_{11} & 0 & 0 & 0 & K_{12} & 0 \\ 0 & K_{33} & K_{34} & 0 & K_{23} & 0 \\ 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ K_{12} & K_{23} & 0 & K_{56} & K_{22}+K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_3 \\ f_4 \\ f_5 \\ f_2+f_6 \\ f_7 \end{bmatrix}, \tag{9.18}$$

Although (9.17) and (9.18) are algebraically equivalent, the latter would be processed faster if a skyline solver (Part III of course) is used for the modified equations.

§9.4.2. Several Homogeneous MFCs

The matrix equation (9.16) 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:

$$u_2 - u_6 = 0, \qquad u_1 + 4u_4 = 0, \qquad 2u_3 + u_4 + u_5 = 0, \tag{9.19}$$

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

$$u_6 = u_2, \qquad u_4 = -\tfrac{1}{4}u_1, \qquad u_3 = -\tfrac{1}{2}(u_4 + u_5) = \tfrac{1}{8}u_1 - \tfrac{1}{2}u_5. \tag{9.20}$$

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 (9.20) is

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{8} & 0 & -\frac{1}{2} & 0 \\ -\frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \\ u_7 \end{bmatrix}, \tag{9.21}$$

The modified system is now formed through the congruential transformation (9.16). 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 for computer, as further discussed later.

REMARK 9.7

The three MFCs (9.20) 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:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 0 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \end{bmatrix} \tag{9.22}$$

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 (9.21). This general matrix form is considered in §9.4.4. Note that non-singularity of $\mathbf{A}_s$ is essential for this method to work.

§9.4.3. Nonhomogeneous MFCs

Extension to non-homogeneous constraints is immediate. In such a case the transformation equation becomes non-homogeneous. For example suppose that (9.15) contains a nonzero prescribed value:

$$\boxed{u_2 - u_6 = 0.2} \tag{9.23}$$

Nonzero RHS values such as 0.2 in (9.23) may be often interpreted as constraint "gaps." Again we chose u_6 as slave: $u_6 = u_2 - 0.2$, and build the transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.2 \\ 0 \end{bmatrix}. \tag{9.24}$$

In compact matrix notation,

$$\mathbf{u} = \mathbf{T}\hat{\mathbf{u}} - \mathbf{g}. \tag{9.25}$$

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 9.4, premultiply both sides of (9.25) by $\mathbf{T}^T\mathbf{K}$ and replace $\mathbf{K}\mathbf{u}$ by $\mathbf{f}$:

$$\boxed{\hat{\mathbf{K}}\,\hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T\mathbf{K}\mathbf{T}, \quad \hat{\mathbf{f}} = \mathbf{T}^T\mathbf{f} - \mathbf{T}^T\mathbf{K}\mathbf{g}.} \tag{9.26}$$

For the example problem this gives

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 - 0.2K_{66} \\ f_3 \\ f_4 \\ f_5 - 0.2K_{56} \\ f_7 - 0.2K_{67} \end{bmatrix}. \tag{9.27}$$

See Exercises for multiple non-homogeneous MFCs.

§9.4.4. *The General Case

For implementation in general-purpose programs the master-slave method can be described as follows. The degrees of freedoms in **u** are classified into three types: independent or uncommitted, masters and slaves. (The uncommitted 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:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um} & \mathbf{K}_{us} \\ \mathbf{K}_{um}^T & \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{us}^T & \mathbf{K}_{ms}^T & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \mathbf{f}_m \\ \mathbf{f}_s \end{bmatrix} \tag{9.28}$$

The MFCs may be written in matrix form as

$$\mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s \mathbf{u}_s = \mathbf{g}, \tag{9.29}$$

where $\mathbf{A}_s$ is assumed square and nonsingular. If so we can solve for the slave freedoms:

$$\mathbf{u}_s = -\mathbf{A}_s^{-1}\mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s^{-1}\mathbf{g} = \mathbf{T}\mathbf{u}_m + \mathbf{g}, \tag{9.30}$$

Inserting into the partitioned stiffness matrix and symmetrizing

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um}\mathbf{T} \\ \mathbf{T}^T\mathbf{K}_{um}^T & \mathbf{T}^T\mathbf{K}_{mm}\mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u - \mathbf{K}_{us}\mathbf{g} \\ \mathbf{f}_m - \mathbf{K}_{ms}\mathbf{g} \end{bmatrix} \tag{9.31}$$

It is seen that the misleading simplicity of the handworked examples is gone.

§9.4.5. *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 **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.

With a bit of trickiness it is possible to maintain the original freedom ordering. Let us display it for the example problem under (9.13). Instead of (9.14), use the *square* transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix}, \tag{9.32}$$

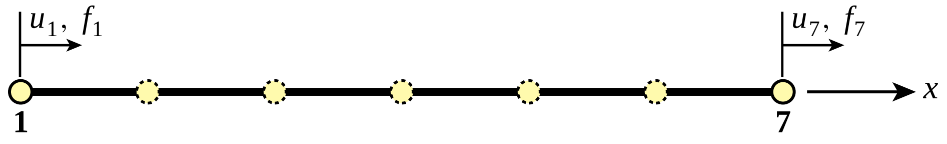


Figure 9.2. Model reduction of the example structure of Figure 9.1 to the end freedoms.

where $\tilde{u}_6$ is a *placeholder* for the slave freedom u_6. The modified equations are

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & 0 & K_{56} & 0 & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{67} & 0 & 0 & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2+f_6 \\ f_3 \\ f_4 \\ f_5 \\ 0 \\ f_7 \end{bmatrix}, \tag{9.33}$$

which are submitted to the equation solver. If the solver is not trained to skip zero row 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 the many points in common with the computer-oriented placeholder technique described in §3.4 to handle single-freedom constraints.

§9.4.6. Model Reduction by Kinematic Constraints

The congruential transformation equations (9.16) and (9.26) 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. Often the reduced model is used in subsequent calculations as component of a larger system, particularly during design or in parameter identification.

To illustrate the method for a static model, consider the bar assembly of Figure 9.1. Assume that the only masters are the end motions u_1 and u_7, as illustrated in Figure 9.2, and interpolate all freedoms linearly:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5/6 & 1/6 \\ 4/6 & 2/6 \\ 3/6 & 3/6 \\ 2/6 & 4/6 \\ 1/6 & 5/6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix}, \qquad \text{or} \quad \mathbf{u} = \mathbf{T}\,\hat{\mathbf{u}}. \tag{9.34}$$

The reduced-model 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

$$\begin{bmatrix} \hat{K}_{11} & \hat{K}_{17} \\ \hat{K}_{17} & \hat{K}_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix} = \begin{bmatrix} \hat{f}_1 \\ \hat{f}_7 \end{bmatrix}, \tag{9.35}$$

where

$$\begin{aligned}
\hat{K}_{11} &= \tfrac{1}{36}(36K_{11}+60K_{12}+25K_{22}+40K_{23}+16K_{33}+24K_{34}+9K_{44}+12K_{45}+4K_{55}+4K_{56}+K_{66})\\
\hat{K}_{17} &= \tfrac{1}{36}(6K_{12}+5K_{22}+14K_{23}+8K_{33}+18K_{34}+9K_{44}+18K_{45}+8K_{55}+14K_{56}+5K_{66}+6K_{67})\\
\hat{K}_{77} &= \tfrac{1}{36}(K_{22}+4K_{23}+4K_{33}+12K_{34}+9K_{44}+24K_{45}+16K_{55}+40K_{56}+25K_{66}+60K_{67}+36K_{77})\\
\hat{f}_1 &= \tfrac{1}{6}(6f_1+5f_2+4f_3+3f_4+2f_5+f_6), \qquad \hat{f}_7 = \tfrac{1}{6}(f_2+2f_3+3f_4+4f_5+5f_6+6f_7).
\end{aligned} \tag{9.36}$$

This reduces the order of the FEM model from 7 to 2. The key feature is that the masters are picked *a priori*, as the freedoms to be retained in the model for further use.

REMARK 9.8

Model reduction can also be done by the static condensation method explained in Chapter 11. As the 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.

§9.4.7. Assessment of the Master-Slave Method

What are the good and bad points of this constraint imposition 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. The main implementation drawback is the complexity of the general case as can be seen by studying (9.28) through (9.31). 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 §9.4.5.
2. An auxiliary linear system, namely (9.30), has to be assembled and solved to produce the transformation matrix **T** and vector **g**.
3. The transformation process may generate many additional matrix terms. If a sparse matrix storage scheme is used for **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. At one extreme, if **K** is stored as full matrix and slave freedom coupling in the MFCs is disallowed the logic is simple.[2] At the other extreme, if arbitrary couplings are permitted and **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 programmmed 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:

$$\tfrac{1}{6}u_2 + \tfrac{1}{2}u_4 = u_6, \qquad u_3 + 6u_6 = u_7 \tag{9.37}$$

[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 largest 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:

$$\tfrac{1}{6}u_2 = u_6, \qquad \tfrac{1}{5}u_3 + 6u_6 = u_7, \qquad u_2 + u_3 - u_7 = 0. \tag{9.38}$$

The last constraint is an exact linear combination of the first two. If the program blindly choses 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 (9.38), can be recognized by a rank analysis of the $\mathbf{A}_s$ matrix defined in (9.29). In inexact floating-point arithmetic, however, such detection may fail.[3]

The complexity of slave selection is in fact equivalent to that of automatically selecting kinematic redundancies in the force method. It has led implementors of programs that use this method to require masters and slaves be prescribed in the input data, thus transfering the burden to users.

The method is not generally extendible to nonlinear constraints without extensive reworking.

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 this method, however, because of its great importance of the congruential transformation technique in *model reduction* for static and dynamic problems.

[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 9

MultiFreedom Constraints I

EXERCISE 9.1

[C+N:20] The example structure of Figure 9.1 has $E^{(e)}A^{(e)}/L^{(e)} = 100$ for each element $e = 1, \ldots, 6$. Consequently $K_{11} = K_{77} = 100$, $K_{22} = \ldots = K_{66} = 200$, $K_{12} = K_{23} = \ldots = K_{67} = -100$. 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 (9.30) 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 effective recovered node forces are

$$\begin{aligned} \mathbf{u}^T &= [\,0 \quad 0.270 \quad 0.275 \quad 0.250 \quad 0.185 \quad 0.070 \quad 0.140\,] \\ (\mathbf{K}\mathbf{u})^T &= [\,-27 \quad 26.5 \quad 3 \quad 4 \quad 5 \quad -18.5 \quad 7\,] \end{aligned} \tag{E9.1}$$

Use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* program solves this Exercise:

```
(* Exercise 9.1 - Master-Slave Method *)
K11=100; K77=100;K12=0; K22=K33=K44=K55=K66=200;
K12=K23=K34=K45=K56=K67=-100;
f1=1;f2=2;f3=3;f4=4;f5=5;f6=6;f7=7;
K={{K11,K12,0,0,0,0,0},{K12,K22,K23,0,0,0,0},
   {0,K23,K33,K34,0,0,0},{0,0,K34,K44,K45,0,0},
   {0,0,0,K45,K55,K56,0},{0,0,0,0,K56,K66,K67},
   {0,0,0,0,0,K67,K77}};  Print["K=",K//MatrixForm];
f={f1,f2,f3,f4,f5,f6,f7}; Print["f=",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}};   (* Freedom transformation matrix *)
b={0,0,0,0,0,-1/5,0}; (* Constraint gap vector *)
Print["Transformation matrix T=",T//MatrixForm];
Print["Constraint gap vector b=",b];
Khat=Simplify[Transpose[T].K.T]; fhat=Simplify[Transpose[T].(f-K.b)];
Khat[[1,1]]=1; Khat[[1,2]]=Khat[[2,1]]=0; fhat[[1]]=0; (* fix left end *)
Print["Modified Stiffness after BC:",Khat//MatrixForm];
Print["Modified f after BC:",fhat];
uhat=N[Inverse[Khat].fhat];
Print["Computed uhat=",uhat];
u=Join[uhat[[{1,2,3,4,5}]],{uhat[[2]]-1/5},{uhat[[6]]}]; (* complete u *)
Print["Solution u=",u];
Print["Recovered node forces=",K.u];
```

EXERCISE 9.2

[C+N:25] As in the previous Exercise but applying the three MFCs (9.22).

EXERCISE 9.3

[A:25] Can the MFCs be pre-processed to make sure that no slave freedom in a MFC appears as master in another?

EXERCISE 9.4

[A:25] In the general case discussed in §9.4.4, under which condition is the matrix $\mathbf{A}_s$ of (9.32) diagonal and thus trivially invertible?

EXERCISE 9.5

[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 9.6

[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 9.7

[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).

10 MultiFreedom Constraints II

TABLE OF CONTENTS

In this Chapter we continue the discussion of methods to treat multifreedom constraints (MFCs). The master-slave method described in the previous Chapter was found to exhibit serious shortcomings in 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 of these techniques are better suited to general implementations of the Finite Element Method.

§10.1. THE PENALTY METHOD

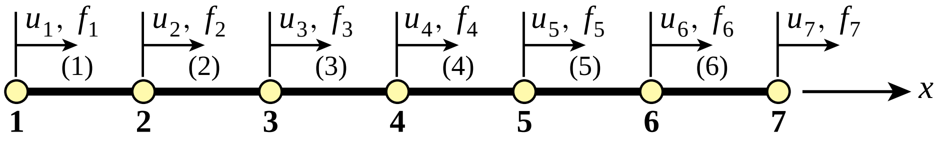


Figure 10.1. The example structure of Chapter 9, repeated for convenience.

§10.1.1. Physical Interpretation

The penalty method will be first presented using a physical interpretation, leaving the mathematical formulation to a subsequent section. Consider again the example structure of Chapter 9, which is reproduced in Figure 10.1 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 10.2. This bar is called a *penalty element* and w is its *penalty weight*.

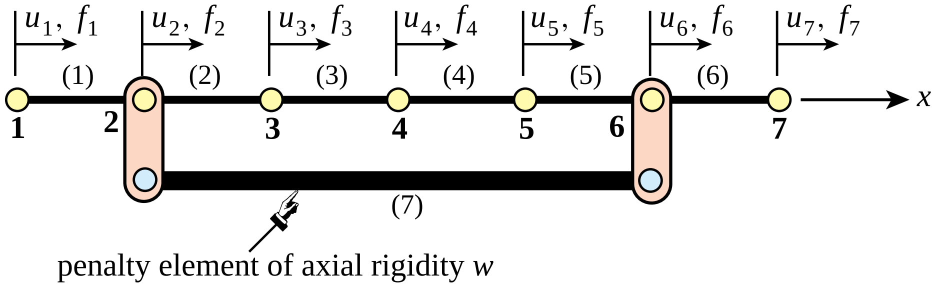


Figure 10.2. Adjunction of a fictitious penalty bar of axial stiffness w, identified as element 7,to enforce the MFC $u_2 = u_6$.

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, $\mathbf{K}^{(7)}\mathbf{u}^{(7)} = \mathbf{f}^{(7)}$, can be shown to be

$$w \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} u_2 \\ u_6 \end{bmatrix} = \begin{bmatrix} f_2^{(7)} \\ f_6^{(7)} \end{bmatrix} \tag{10.1}$$

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: $\widehat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}$, which shown in detail are

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+w & K_{23} & 0 & 0 & -w & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & -w & 0 & 0 & K_{56} & K_{66}+w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \tag{10.2}$$

This system can now be submitted to the equation solver. Note that $\hat{\mathbf{u}} \equiv \mathbf{u}$, and only $\mathbf{K}$ has changed.

§10.1.2. Choosing the Penalty Weight

What happens when (10.2) 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 w: 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.

Consequently, 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 (10.2) 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 $e = 1, \ldots 6$ are unity, that $w >> 1$, 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.[1] The modified stiffness matrix of (10.2) becomes

$$\widehat{\mathbf{K}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2+w & -1 & 0 & 0 & -w & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & -w & 0 & 0 & -1 & 2+w & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{10.3}$$

Clearly as $w \to \infty$ rows 2 and 6, as well as columns 2 and 6, tend to become linearly dependent; in fact the negative of each other. Linear dependence means singularity; hence $\widehat{\mathbf{K}}$ approaches singularity as $w \to \infty$. In fact, if w exceeds $1/\epsilon_f = 10^{16}$ the computer will not be able to distinguish $\widehat{\mathbf{K}}$ from an exactly singular matrix. If $w << 10^{16}$ but $w >> 1$, 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.

[1] Such definitions are more rigurously done by working with binary numbers and base-2 arithmetic but for the present conceptual discussion the use of decimal powers is sufficient.

§10.1.3. The Square Root Rule

Obviously 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.[2] Then choose penalty weights to be of order $10^{k+p/2}$ with the proviso that such a choice would not cause arithmetic overflow.[3]

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 more precision on the computer. This is not easy to do when using standard programming languages.

The name "square root" arises because the recommended w is in fact $10^k\sqrt{10^p}$. Thus it is seen that the choice of penalty weight by this rule involves knowledge of both stiffness magnitudes and floating-point hardware properties of the computer used.

§10.1.4. Penalty Elements for General MFCs

For the constraint $u_2 = u_6$ the physical interpretation of the penalty element is clear. Points 2 and 6 must move in lockstep along x, which can be approximately enforced by the fat bar device shown in Figure 10.2. 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 has not yet been introduced (it is given in the next subsection), the procedure used for constructing a penalty element is stated here as a recipe. Consider the homogeneous constraint

$$3u_3 + u_5 - 4u_6 = 0. \tag{10.4}$$

Rewrite this equation in matrix form

$$[\,3 \quad 1 \quad -4\,] \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 0, \tag{10.5}$$

and premultiply both sides by the transpose of the coefficient matrix:

$$\begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix} [\,3 \quad 1 \quad -4\,] \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \bar{\mathbf{K}}^{(e)} \mathbf{u}^{(e)} = 0. \tag{10.6}$$

[2] Such order-of-magnitude estimates can be readily found by scanning the diagonal of **K** because the largest stiffness coefficient of the actual structure is usually a diagonal entry.

[3] 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 (10.1) with the scaled penalty element (10.6) yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33}+9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55}+w & K_{56}-4w & 0 \\ 0 & 0 & -12w & 0 & K_{56}-4w & K_{66}+16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \tag{10.7}$$

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

$$[\,3 \quad 1 \quad -4\,] \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 1. \tag{10.8}$$

Premultiply both sides by the transpose of the coefficient matrix:

$$\begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix}. \tag{10.9}$$

Scaling by w and assembling yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33}+9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55}+w & K_{56}-4w & 0 \\ 0 & 0 & -12w & 0 & K_{56}-4w & K_{66}+16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3+3w \\ f_4 \\ f_5+w \\ f_6-4w \\ f_7 \end{bmatrix}, \tag{10.10}$$

§10.1.5. *The Theory Behind the Recipe

The rule comes from the following mathematical theory. Suppose we have a set of m linear MFCs. Using the matrix notation introduced in §9.1.3, these will be stated as

$$\mathbf{a}_p \mathbf{u} = b_p, \quad p = 1, \ldots m \tag{10.11}$$

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"

$$P = \sum_{p=1}^{m} P_p, \qquad \text{with} \quad P_p = \mathbf{u}^T \left(\tfrac{1}{2}\mathbf{a}_p^T \mathbf{a}_p \mathbf{u} - w_p \mathbf{a}_p^T b_p\right) = \tfrac{1}{2}\mathbf{u}^T \mathbf{K}^{(p)} \mathbf{u} - \mathbf{u}^T \mathbf{f}^{(p)}, \tag{10.12}$$

where we have called $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and $\mathbf{f}^{(p)} = w_p \mathbf{a}^T b_i$. 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 with $\mathbf{u}$ yields

$$\Big(\mathbf{K}\mathbf{u} + \sum_{p=1}^{m} \mathbf{K}^{(p)}\Big)\,\mathbf{u} = \mathbf{f} + \sum_{p=1}^{m} \mathbf{f}^{(p)}. \tag{10.13}$$

Each term of the sum on p, which derives from term P_p in (10.12), 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 $\mathbf{f}^{(p)} = w_p \mathbf{a}_p^T b_p$.

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

$$(\mathbf{K} + \mathbf{A}^T\mathbf{W}\mathbf{A})\,\mathbf{u} = \mathbf{f} + \mathbf{W}\mathbf{A}^T\mathbf{b}, \tag{10.14}$$

where $\mathbf{W}$ is a diagonal matrix of penalty weights. This compact form, however, conceals the structure of the penalty elements.

§10.1.6. Assessment of the Penalty Method

The main advantage of the penalty function method is its straightforward computer implementation. Looking at modifified systems such as (10.7) and (10.10) it is obvious that the equations need not be re-arranged. 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 implemented 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 (upon processsing for single-freedom constraints if necessary) can be passed to the equation solver.

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 bar elements connecting 2 and 6 will be inserted, doubling the intended weight but otherwise not causing undue harm.

An advantage with respect to the Lagrange multiplier method described in §10.2 is that positive definiteness is not lost. Such loss can affect the performance of certain numerical processes. Finally, it is worth noting that the 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 mentioned previously often works, although effective use calls for knowledge of the the magnitude of stiffness coefficients. Such knowledge may be difficult to extract from a general purpose "black box" program. For complex cases selection of appropriate weights may require extensive numerical experimentation, wasting the user time with numerical games that have no bearing on the real objective, which is getting a solution.

The deterioration of the condition number of the penalty-augmented stiffness matrix can have an even more serious side effects in some solution procedures such as eigenvalue extraction or iterative solvers. Finally, even if optimal weights are selected, the combined solution error cannot be lowered beyond a treshold value.

From these comments 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.

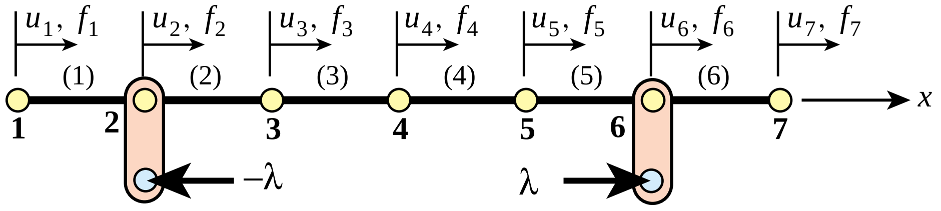


Figure 10.3. Physical interpretation of Lagrange multiplier adjunction to enforce the MFC $u_2 = u_6$.

§10.2. LAGRANGE MULTIPLIER ADJUNCTION

§10.2.1. 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 10.3. These are called the *constraint forces*. Incorporating these forces into the original stiffness equations (9.10) we get

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 - \lambda \\ f_3 \\ f_4 \\ f_5 \\ f_6 + \lambda \\ f_7 \end{bmatrix}. \tag{10.15}$$

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:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \tag{10.16}$$

But now we have 7 equations in 8 unknowns. To render the system determinate, the constraint condition $u_2 - u_8 = 0$ is appended as eight equation:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \end{bmatrix}, \qquad (10.17)$$

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.

§10.2.2. 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

$$u_2 - u_6 = 0, \qquad 5u_2 - 8u_7 = 3, \qquad 3u_3 + u_5 - 4u_6 = 1, \qquad (10.18)$$

Adjoin these MFCs as the eighth, nineth and tenth equations:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}, \qquad (10.19)$$

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 adjoining 3 columns that are the transpose of the 3 last rows, and filling the bottom right-hand corner with zeros:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 & 5 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 & -8 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}, \qquad (10.20)$$

§10.2.3. *The Theory Behind the Recipe

The recipe illustrated by (10.20) comes from a well known technique of variational calculus. Using the matrix notation introduced in §9.1.3, compactly denote the set of m MFCs by $\mathbf{Au} = \mathbf{b}$, where $\mathbf{A}$ is $m \times n$. The potential energy of the unconstrained finite element model is $\Pi = \frac{1}{2}\mathbf{u}^T\mathbf{Ku} - \mathbf{u}^T\mathbf{f}$. To impose the constraint, adjoin m Lagrange multipliers collected in vector $\boldsymbol{\lambda}$ and form the Lagrangian

$$L(\mathbf{u}, \boldsymbol{\lambda}) = \Pi + \lambda^T(\mathbf{Au} - \mathbf{b}) = \tfrac{1}{2}\mathbf{u}^T\mathbf{Ku} - \mathbf{u}^T\mathbf{f} + \lambda^T(\mathbf{Au} - \mathbf{b}). \tag{10.21}$$

Extremization of L with respect to $\mathbf{u}$ and $\boldsymbol{\lambda}$ yields the multiplier-augmented form

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix} \tag{10.22}$$

The master stiffness matrix $\mathbf{K}$ in (10.22) 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 $-\mathbf{A}^T\boldsymbol{\lambda}$.

§10.2.4. 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 computation errors). It provides directly the constraint forces, which are of interest in many applications. It does not require any 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. It renders the augmented stiffness matrix indefinite, a property that may cause grief with some solution methods such as Cholesky factorization or conjugate gradients. 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 the Lagrangian multiplier method appear to be the most elegant 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.

§10.3. *THE AUGMENTED LAGRANGIAN METHOD

The general matrix forms of the penalty function and Lagrangian multiplier methods are given by expressions (10.14) and (10.22), respectively. A useful connection between these methods can be established as follows.

Because the lower diagonal block of the bordered stiffness matrix in (10.22) is null, it is not possible to directly eliminate $\boldsymbol{\lambda}$. To allow that, 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, $\epsilon\mathbf{S}^{-1}\boldsymbol{\lambda}$ is added to the right-hand side:

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \epsilon\mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \epsilon\mathbf{S}^{-1}\boldsymbol{\lambda}^P \end{bmatrix} \tag{10.23}$$

Here superscript P (for "predicted value") is attached to the $\boldsymbol{\lambda}$ on the right-hand side as a "tracer." We can now formally solve for $\boldsymbol{\lambda}$ and subsequently for $\mathbf{u}$. The results may be presented as

$$\begin{aligned}(\mathbf{K} + w\mathbf{A}^T\mathbf{S}\mathbf{A})\,\mathbf{u} &= \mathbf{f} + w\mathbf{A}^T\mathbf{S}\mathbf{b} - \mathbf{A}^T\boldsymbol{\lambda}^P,\\ \boldsymbol{\lambda} &= \boldsymbol{\lambda}^P + w\mathbf{S}(\mathbf{b} - \mathbf{A}\mathbf{u}),\end{aligned} \tag{10.24}$$

Setting $\boldsymbol{\lambda}^P = \mathbf{0}$ in the first matrix equation yields

$$(\mathbf{K} + w\mathbf{A}^T\mathbf{S}\mathbf{A})\,\mathbf{u} = \mathbf{f} + w\mathbf{A}^T\mathbf{S}\mathbf{b}. \tag{10.25}$$

On taking $\mathbf{W} = w\mathbf{S}$, the general matrix equation (10.14) 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.*[4] This strategy circumvents the aforementioned ill-conditioning problems when the weight w is gradually increased. One such method is easily constructed by inspecting (10.24). Using superscript k as an iteration index and keeping w fixed, solve equations (10.24) in tandem as follows:

$$\begin{aligned}(\mathbf{K} + \mathbf{A}^T\mathbf{W}\mathbf{A})\,\mathbf{u}^k &= \mathbf{f} + \mathbf{A}^T\mathbf{W}\mathbf{b} - \mathbf{A}^T\boldsymbol{\lambda}^k,\\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k + \mathbf{W}(\mathbf{b} - \mathbf{A}\mathbf{u}^k),\end{aligned} \tag{10.26}$$

for $k = 0, 1, \ldots,$ beginning with $\boldsymbol{\lambda}^0 = \mathbf{0}$.[5] 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 (10.23) directly.

The family of iterative procedures that may be precipitated from (10.24) collectively pertains to the class of *augmented Lagrangian methods*. These have received much attention since the late 1960s, when they originated in the field of constrained optimization.

	Master-Slave Elimination	Penalty Function	Lagrange Multiplier
Generality	fair	excellent	excellent
Ease of implementation	poor to fair	good	fair
Sensitivity to user decisions	high	high	slight
Accuracy	variable	mediocre	excellent
Sensitivity as regards constraint dependence	high	none	high
Retains positive definiteness	yes	yes	no
Modification of DOF vector required	yes	no	yes

Figure 10.4. Assessment summary of MFC application methods.

[4] C. A. Felippa, Iterative procedures for improving penalty function solutions of algebraic systems, *Int. J. Numer. Meth. Engrg.*, **12**, 821–836, 1978.

[5] This form of the stiffness equations is discussed in C. A. Felippa, Iterative procedures for improving penalty function solutions of algebraic systems, *Int. J. Numer. Meth. Engrg.*, **12**, 821–836, 1978.

§10.4. SUMMARY

The treatment of linear MFCs in finite element systems can be carried out by several methods. Three of these: the 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, numerical behavior and simplicity of implementation. See Figure 10.4 for a summary.

For a general purpose program that tries to approach "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.

Homework Exercises for Chapter 10

MultiFreedom Constraints II

EXERCISE 10.1

[C+N:25] This is identical to Exercise 9.1, 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 $k = 3, 4, 5, \ldots 14$. For each sample w compute the Euclidean-norm solution error $e(w) = \sqrt{||\mathbf{u}^p(w) - \mathbf{u}^{ex}||_2}$, where $\mathbf{u}^p$ is the computed solution and $\mathbf{u}^{ex}$ is the exact solution listed in (E9.1). Plot $k = \log_{10} w$ versus $\log_{10} e$ and report for which weight e attains a minimum. (See Slides for a check). Does it roughly agree with the square root rule (§10.1.3) if the computations carry 16 digits of precision?

As in Exercise 9.1, use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* program solves this Exercise:

```
(* Exercise 10.1 - Penalty Function Method *)
K11=100; K77=100;K12=0; K22=K33=K44=K55=K66=200;
K12=K23=K34=K45=K56=K67=-100;
f1=1;f2=2;f3=3;f4=4;f5=5;f6=6;f7=7;
K={{K11,K12,0,0,0,0,0},{K12,K22,K23,0,0,0,0},
   {0,K23,K33,K34,0,0,0},{0,0,K34,K44,K45,0,0},
   {0,0,0,K45,K55,K56,0},{0,0,0,0,K56,K66,K67},
   {0,0,0,0,0,K67,K77}};  Print["K=",K];
f={f1,f2,f3,f4,f5,f6,f7}; Print["f=",f];
nw=12; ew=Table[{0,w},{nw}];
w=100;
Do [ Khat=K; fhat=f;  w=w*10;  (* increase w by 10 every pass *)
   Print["Penalty weight w=",N[w]//ScientificForm];
   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;
   Khat[[1,1]]=1; Khat[[1,2]]=Khat[[2,1]]=0;  fhat[[1]]=0;
   (*Print["Modified Stiffness after BC:",Khat];*)
   (*Print["Modified f after BC:",fhat];*)
   u=uhat=Inverse[N[Khat]].N[fhat];
   (*Print["Solution u=",u]; Print["Recovered node forces=",K.u];*)
   uexact= {0,0.27,0.275,0.25,0.185,0.07,0.14};
   e=Sqrt[(u-uexact).(u-uexact)]; Print["L2 solution error=",e//
   ScientificForm];
   ew[[i]]={Log[10,w],Log[10,e]},
{i,1,nw}];
ListPlot[ew,AxesOrigin->{5,-8},PlotStyle->PointSize[0.02],
         PlotJoined->True,AxesLabel->{"Log10(w)","Log10(u error)"}
        ];
```

Note: If you run the above program, you may get several beeps from *Mathematica* as it is processing the systems with very large weights. Don't be alarmed: those are only warnings. The `Inverse` function is simply alerting you that the coefficient matrices $\hat{\mathbf{K}}$ for weights of order 10^{12} or bigger are very ill-conditioned.

EXERCISE 10.2

[C+N:25] Again identical to Exercise 9.1, 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 $\mathbf{Ku}$ should agree with (E9.1).

Use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* program solves this Exercise:

```
(* Exercise 10.2 - Lagrange Multiplier Method *)
K11=100; K77=100;K12=0;K22=K33=K44=K55=K66=200;
K12=K23=K34=K45=K56=K67=-100;
f1=1;f2=2;f3=3;f4=4;f5=5;f6=6;f7=7;
K={{K11,K12,0,0,0,0,0},{K12,K22,K23,0,0,0,0},
   {0,K23,K33,K34,0,0,0},{0,0,K34,K44,K45,0,0},
   {0,0,0,K45,K55,K56,0},{0,0,0,0,K56,K66,K67},
   {0,0,0,0,0,K67,K77}};
Kb={{K11,K12,0,0,0,0,0, 0},{K12,K22,K23,0,0,0,0, 1},
   {0,K23,K33,K34,0,0,0,0},{0,0,K34,K44,K45,0,0,0},
   {0,0,0,K45,K55,K56,0,0},{0,0,0,0,K56,K66,K67,-1},
   {0,0,0,0,0,K67,K77,0},{0,1,0,0,0,-1,0,0}};
fb={f1,f2,f3,f4,f5,f6,f7, 1/5};
Kb[[1,1]]=1;Kb[[1,2]]=Kb[[2,1]]=0;fb[[1]]=0;
Print["Bordered K=",Kb];Print["Bordered f=",fb];
ub=N[Inverse[Kb].fb];  u=Take[ub,7];
Print[" Solution u=",u ,",   lambda=",ub[[8]]];
Print["Recovered node forces=",K.u];
```

EXERCISE 10.3

[A:15] for the example structure, show which penalty elements would implement the following MFCs:

$$\begin{aligned} &\text{(a)} \quad u_2 + u_6 = 0, \\ &\text{(b)} \quad u_2 = 3u_6 = 0. \end{aligned} \tag{E10.1}$$

As answer, show the stiffness equations of those two elements in a manner similar to (10.1).

EXERCISE 10.4

[A/C+N:15+15+10] Suppose that the assembled stiffness equations for a one-dimensional finite element model before imposing constraints are

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}. \tag{E10.2}$$

This system is to be solved subject to the multipoint constraint

$$u_1 = u_3. \tag{E10.3}$$

(a) Impose the constraint (E10.3) by the master-slave method taking u_1 as master, and solve the resulting 2×2 system of equations by hand.

(b) Impose the constraint (E10.3) 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 $w \to \infty$ 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 $(6w + 5)/(4w + 4)$.

(c) Impose the constraint (E10.3) by the Lagrange multiplier method. Show the 4×4 multiplier-augmented system of equations analogous to (10.13) and solve it by computer or calculator.

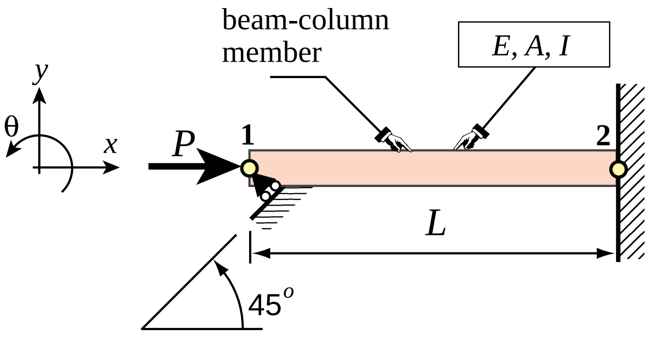


Figure E10.1. Beam-colum member for Exercise 10.5.

EXERCISE 10.5

[A/C:20+20+20] The beam-column member shown in Figure E10.1 rests on a skew roller that forms a 45° angle with the horizontal axis x, and is loaded axially by a force P.

The finite element equations upon removing the fixed right end, but before imposing the skew-roller MFC, are

$$\begin{bmatrix} EA/L & 0 & 0 \\ 0 & 12EI/L^3 & 6EI/L^2 \\ 0 & 6EI/L^2 & 4EI/L \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_1 \end{bmatrix} = \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix}, \tag{E10.4}$$

where E, A, and I are member properties, θ_1 is the left end rotation, and L is the member length. To simplify the following calculations set $P = \alpha EA$, and $I = \beta AL^2$, in which α and β are dimensionless parameters.

(a) Apply the skew roller constraint by the master-slave method (make u_{y1} slave) and solve for u_{x1} in terms of L, α and β. This may be done by hand or CAS.

(b) Apply the skew roller constraint by the penalty method by adjoining a penalty truss member of axial stiffness $k = wEA$ normal to the roller, and compute u_{x1} (Cramer's rule is recommended if solved by hand). Verify that as $w \to \infty$ the exact answer obtained in (a) is recovered.

(c) Apply the skew roller constraint by Lagrangian multiplier adjunction, and solve the resulting 4×4 system of equations using a CAS.

EXERCISE 10.6

[A:25] 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

$$\Pi_{MS} = \tfrac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} + \boldsymbol{\lambda}^T(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \tag{E10.5}$$

and write down the stationarity conditions of Π_{MS} with respect to $\mathbf{u}$, $\boldsymbol{\lambda}$ and $\hat{\mathbf{u}}$ in matrix form.

EXERCISE 10.7

[A:35] Check the matrix equations (10.23) through (10.26) quoted for the Augmented Lagrangian method.

EXERCISE 10.8

[A:40] (Advanced, close to a research problem) 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

$$\Pi_P = \tfrac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} + \tfrac{1}{2}w(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}})^T(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \tag{E10.6}$$

Write down the matrix stationarity conditions with respect to to **u** $\hat{\mathbf{u}}$ and eliminate **u**. Using the Woodbury formula show that

$$\overline{\mathbf{K}} = \mathbf{T}\mathbf{K}^{-1}\mathbf{T}^T \qquad \text{(E10.7)}$$

and that using Woodbury's formula (Appendix C, §C.5.2) we get

$$(\mathbf{K} + w\mathbf{T}^T\mathbf{S}\mathbf{T})^{-1} = \mathbf{K}^{-1} - \mathbf{K}^{-1}\mathbf{T}^T(\overline{\mathbf{K}} + w^{-1}\mathbf{S}^{-1})^{-1}\mathbf{T}\,\mathbf{K}^{-1}. \qquad \text{(E10.8)}$$

11 Superelements and Global-Local Analysis

TABLE OF CONTENTS

§11.1. SUPERELEMENT CONCEPT

Superelements are groupings of finite elements that, upon assembly, may be considered as an *individual element* for computational purposes. These purposes may be related to modeling or solution needs.

A random assortment of elements does not necessarily make up a superelement. To be considered as such, an element grouping must meet certain conditions. Informally we can say that the grouping must form a structural component on its own. This imposes certain conditions stated mathematically in §11.1.3. 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 7, 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 the 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 in the 1970s to take up the entire spectrum, ranging from *individual elements* to *complete structures*. This universality is helped by the common features noted below.

Both macroelements and substructures are treated exactly the same way in so far as matrix processing is concerned. The basic processing rule is that associated with *condensation* of internal degrees of freedom. This is illustrated in the following section with a very simple example. The reader should note, however, that the technique applies to *any* superelement, whether composed of two or a million elements.

§11.1.1. Where Does the Idea Comes From?

Substructuring was invented by aerospace engineers in the early 1960s[1] to carry out a first-level breakdown of complex systems such as a complete airplane, as depicted in Figures 11.1 and 11.2. The decomposition may continue hierarchically through additional levels as illustrated in Figure 11.3. The concept is also natural for space vehicles operating in stages, such as the Apollo short stack depicted in Figure 11.4.

One obvious advantage of this idea results if the structure is built of several identical units. For example, the wing substructures S_2 and S_3 are largely identical except for a reflection about the fuselage midplane, and so are the stabilizers S_4 and S_5. Even if the loading is not symmetric, taking account of the structural symmetry reduces mesh preparation time.

The concept was then picked up and developed extesively by the offshore and shipbuilding industries,

[1] For a bibliography of early work, see J. S. Przeminiecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968 (also in Dover ed).

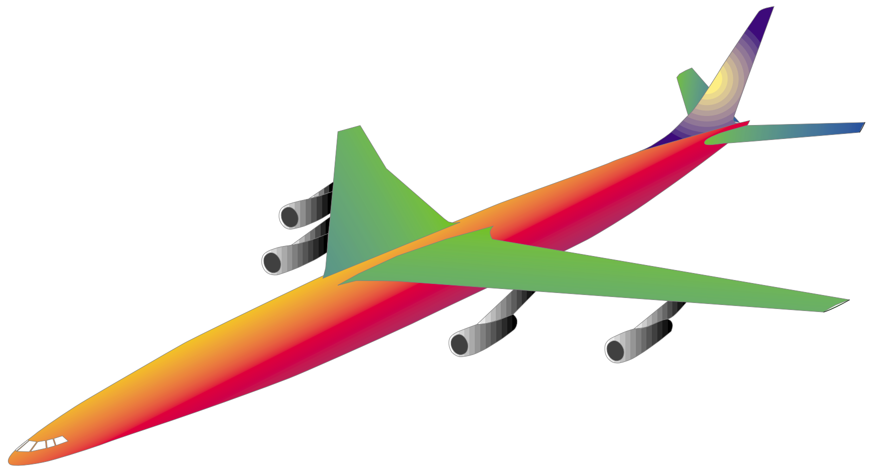


Figure 11.1. A complete airplane.

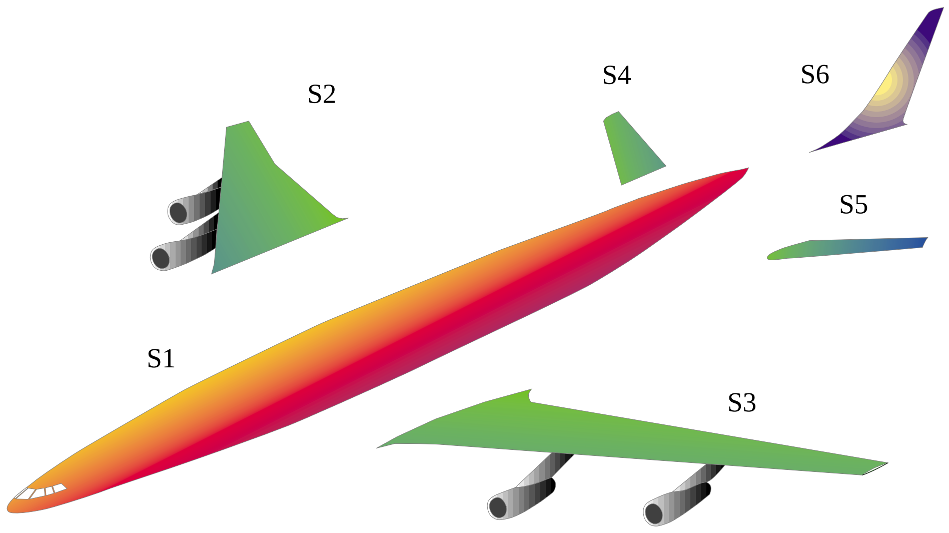


Figure 11.2. Airplane broken down into six level one substructures identified as S_1 through S_6.

the products of which tend to be very modular and repetitive to reduce fabrication costs. As noted above, repetition of structural components favors the use of substructuring techniques.

At the other extreme, mesh units appeared in the early days of finite element methods. They were motivated by user convenience. For example, in hand preparation of finite element 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.

§11.1.2. 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*.

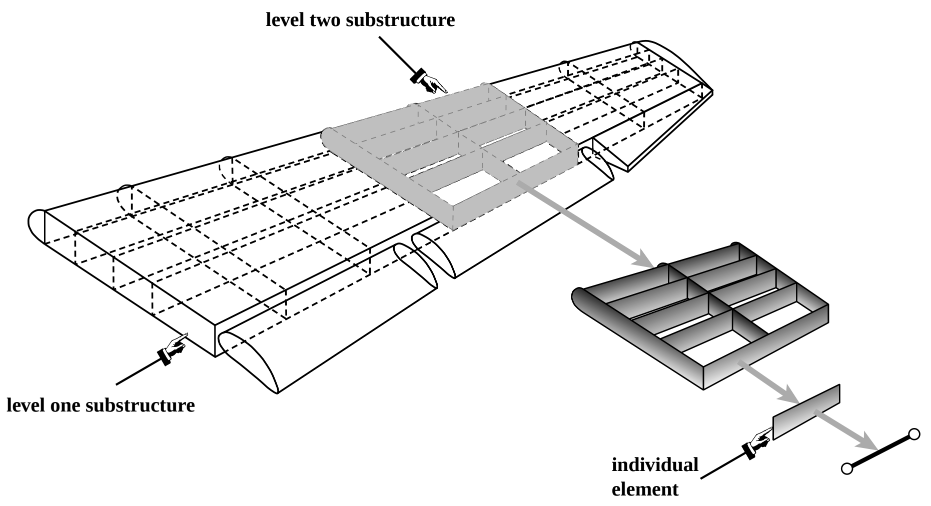


Figure 11.3. Further breakdown of wing structure. The decomposition process may continue down to the individual element level.

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.

§11.1.3. *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.

§11.2. 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. Node 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 11.5.

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.

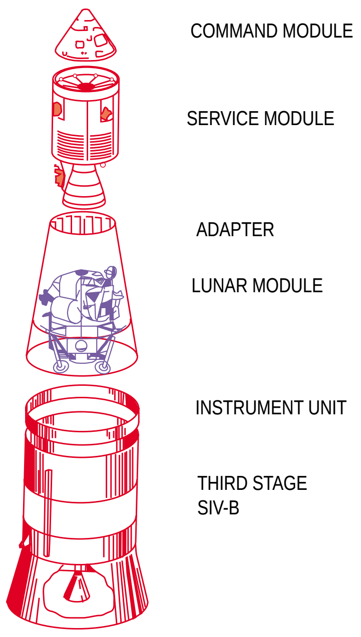


Figure 11.4. The Apollo short stack.

§11.2.1. Condensation by Explicit Matrix Operations

To carry out the condensation process, the assembled stiffness equations of the superelement are partitioned as follows:

$$\begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{bi} \\ \mathbf{K}_{ib} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \mathbf{f}_i \end{bmatrix}. \tag{11.1}$$

where subvectors $\mathbf{u}_b$ and $\mathbf{u}_i$ collect *boundary* and *interior* degrees of freedom, respectively. Take the second matrix equation:

$$\mathbf{K}_{ib}\mathbf{u}_b + \mathbf{K}_{ii}\mathbf{u}_i = \mathbf{f}_i, \tag{11.2}$$

If $\mathbf{K}_{ii}$ is nonsingular we can solve for the interior freedoms:

$$\mathbf{u}_i = \mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{K}_{ib}\mathbf{u}_b), \tag{11.3}$$

Replacing into the first matrix equation of (11.2) yields the *condensed stiffness equations*

$$\tilde{\mathbf{K}}_{bb}\mathbf{u}_b = \tilde{\mathbf{f}}_b. \tag{11.4}$$

In this equation,

$$\tilde{\mathbf{K}}_{bb} = \mathbf{K}_{bb} - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib}, \qquad \tilde{\mathbf{f}}_b = \mathbf{f}_b - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{f}_i, \tag{11.5}$$

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.

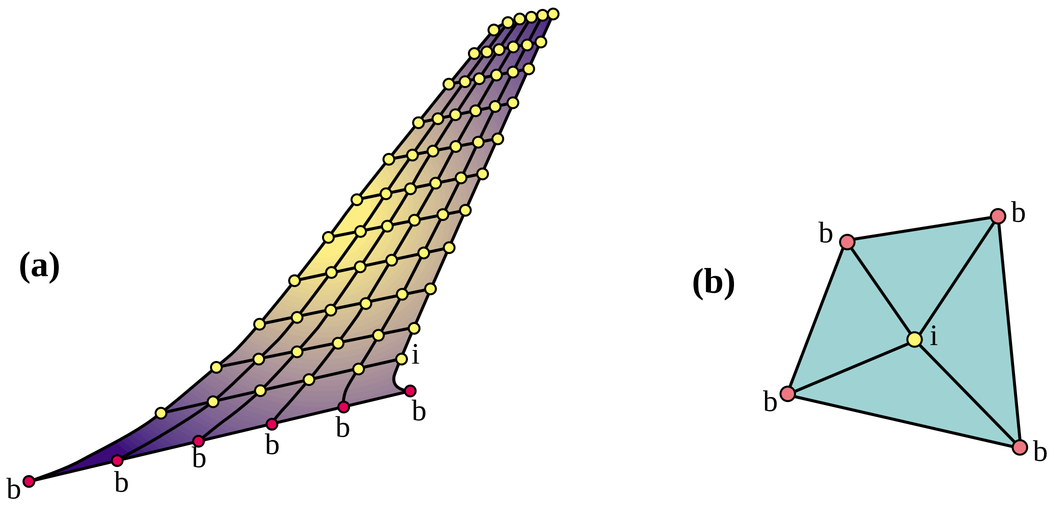


Figure 11.5. Classification of superelement freedoms into boundary and internal. (a) shows the vertical stabilizer substructure S_6 of Figure 11.2. (The FE mesh is depicted as two-dimensional for illustrative purposes; for an actual aircraft it will be three dimensional with an internal structure similar to the one displayed in Figure 11.3.) 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.

REMARK 11.1

The feasibility of the condensation process (11.5) hinges on the non-singularity of $\mathbf{K}_{ii}$. This matrix is nonsingular if the superelement is rank-sufficient in the sense stated in §11.1.3, 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 we cite the concepts of projectors and generalized inverses.

§11.2.2. Condensation by Symmetric Gauss Elimination

In the computer implementation of the 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 is 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:

$$\begin{bmatrix} 6 & -2 & -1 & -3 \\ -2 & 5 & -2 & -1 \\ -1 & -2 & 7 & -4 \\ -3 & -1 & -4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \\ 0 \end{bmatrix}. \tag{11.6}$$

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

Cell 11.1 Program to Condense Out the Last Freedom from $\mathbf{Ku} = \mathbf{f}$

```
CondenseLastFreedom[K_,f_]:=Module[{n=Length[K],c,pivot,Kc,fc},
   If [n<=0,Return[{K,f}]];
   Kc=Table[0,{n-1},{n-1}]; fc=Table[0,{n-1},{1}];
   pivot=K[[n,n]]; If [pivot==0,Print["Singular Matrix"]; Return[{K,f}]];
   Do[ c=K[[i,n]]/pivot; fc[[i,1]]=f[[i,1]]-c*f[[n,1]];
      Do[  Kc[[j,i]]=Kc[[i,j]]=K[[i,j]]-c*K[[n,j]],
     {j,1,i}],
   {i,1,n-1}];
   Return[{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["K=",K, "  f=",f];
{K,f}=CondenseLastFreedom[K,f]; Print["Upon condensing freedom 4: ",K,f];
{K,f}=CondenseLastFreedom[K,f]; Print["Upon condensing freedom 3: ",K,f];
```

and column:

$$\begin{bmatrix} 6-\frac{(-3)\times(-3)}{8} & -2-\frac{(-1)\times(-3)}{8} & -1-\frac{(-4)\times(-3)}{8} \\ -2-\frac{(-3)\times(-1)}{8} & 5-\frac{(-1)\times(-1)}{8} & -2-\frac{(-4)\times(-1)}{8} \\ -1-\frac{(-3)\times(-4)}{8} & -2-\frac{(-1)\times(-4)}{8} & 7-\frac{(-4)\times(-4)}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3-\frac{0\times(-3)}{8} \\ 6-\frac{0\times(-1)}{8} \\ 4-\frac{0\times(-4)}{8} \end{bmatrix}, \tag{11.7}$$

or

$$\begin{bmatrix} \frac{39}{8} & -\frac{19}{8} & -\frac{5}{2} \\ -\frac{19}{8} & \frac{39}{8} & -\frac{5}{2} \\ -\frac{5}{2} & -\frac{5}{2} & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix}. \tag{11.8}$$

Repeat the process for the third row and column to eliminate u_3:

$$\begin{bmatrix} \frac{39}{8}-\frac{(-5/2)\times(-5/2)}{5} & -\frac{19}{8}-\frac{(-5/2)\times(-5/2)}{5} \\ -\frac{19}{8}-\frac{(-5/2)\times(-5/2)}{5} & \frac{39}{8}-\frac{(-5/2)\times(-5/2)}{5} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 3-\frac{4\times(-5/2)}{5} \\ 6-\frac{4\times(-5/2)}{5} \end{bmatrix}, \tag{11.9}$$

or

$$\begin{bmatrix} \frac{29}{8} & -\frac{29}{8} \\ -\frac{29}{8} & \frac{29}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}. \tag{11.10}$$

These are the condensed stiffness equations. Cell 11.1 shows a *Mathematica* program that executes the foregoing steps.

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 in any particular order. For example, suppose that the assembled substructure contains originally eight degrees of freedom and that the freedoms to

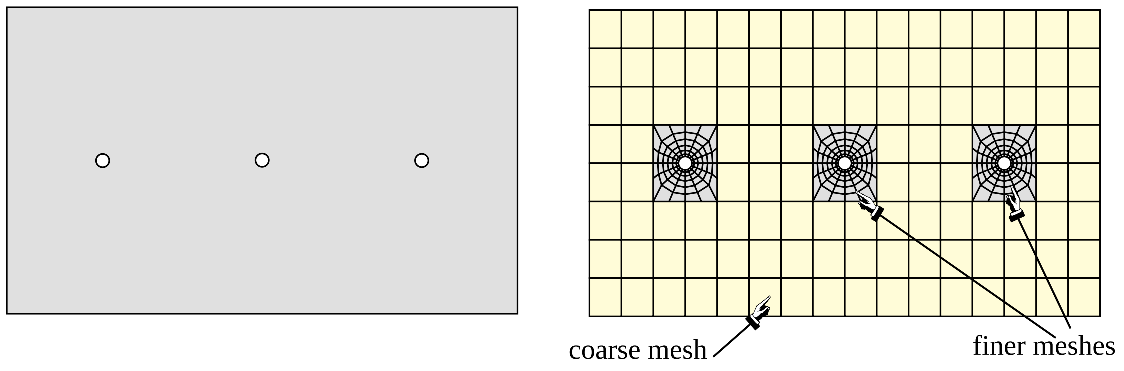


Figure 11.6. Left: example panel structure for global-local analysis. Right: a FEM mesh for a one-shot analysis.

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.

REMARK 11.2

The symmetric Gauss elimination procedure, as illustrated in steps (11.7) through (11.10), 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.

REMARK 11.3

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 $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A}$ is $n \times n$ square and $\mathbf{x}$ and $\mathbf{y}$ are n-vectors, is partitioned as

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}. \tag{11.11}$$

Assuming the appropriate inverses to exist, then the following are easily verified matrix identities:

$$\begin{bmatrix} \mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix}. \tag{11.12}$$

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 (11.12) it is necessary to change the sign of one of the subvectors.

§11.3. GLOBAL-LOCAL ANALYSIS

As noted 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.

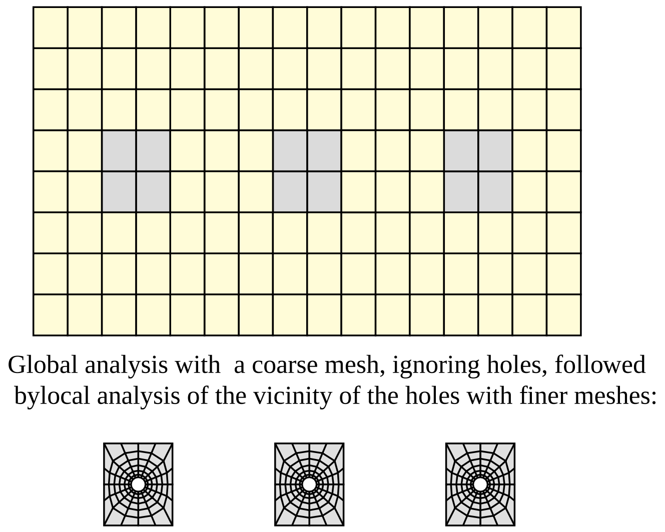

Global analysis with a coarse mesh, ignoring holes, followed
bylocal analysis of the vicinity of the holes with finer meshes:

Figure 11.7. Global-local analysis of problem of Figure 11.5.

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 incorporated in a series of local analyses.

The gist of the global-local approach is explained in the example illustrated in Figures 11.6 and 11.7. Although the structure is admittedly two 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 11.6, 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 11.6 illustrates the global-local analysis procedure. The global analysis is done with a coarse but regular FEM mesh which *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.

The BC choice noted above gives rise to two basic variations of the global-local approach. Experience accumulated over several decades[2] 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.

[2] Particularly in the aerospace industry, in which the global-local technique has been used since the mid 1960s.

Homework Exercises for Chapter 11

Superelements and Global-Local Analysis

EXERCISE 11.1

[N:15] Suppose that the assembled stiffness equations for a one-dimensional superelement are

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}. \tag{E11.1}$$

Eliminate u_2 from the unconstrained stiffness equations (E10.1) by static condensation, and show (but do not solve) the condensed equation system. Use either the explicit inverse formulas (11.5) or the symmetric Gauss elimination process explained in §11.2.2. Hint: regardless of method the result should be

$$\begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \tag{E11.2}$$

EXERCISE 11.2

[C+N:20] Generalize the program of **Cell 11.1** to a module `CondenseFreedom[K,f,i]` that is able to condense the `i`th degree of freedom from `K` and `f`, which is not necessarily the last one. That is, `i` may range from 1 to `n`, where `n` is the dimension of $\mathbf{Ku} = \mathbf{f}$. Apply that program to solve Exercise 11.1.

EXERCISE 11.3

[D:15] Explain the similarities and differences between superelement analysis and global-local FEM analysis.

EXERCISE 11.4

[A:25] Show that the static condensation process can be viewed as a master-slave transformation method (Chapter 9) in which the interior freedoms $\mathbf{u}_i$ are taken as slaves linked by the transformation relation

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{22}^{-1}\mathbf{K}_{21} \end{bmatrix} [\,\mathbf{u}_b\,] - \begin{bmatrix} \mathbf{0} \\ -\mathbf{K}_{22}^{-1}\mathbf{f}_i \end{bmatrix} = \mathbf{T}\mathbf{u}_b - \mathbf{g}. \tag{E11.3}$$

Hint: apply (9.26) in which $\hat{\mathbf{u}} \equiv \mathbf{u}_b$ are the masters and compare the result to (11.6)-(11.7).

EXERCISE 11.5

[D:30] (Requires thinking) Explain the conceptual and operational differences between standard (one-stage) FEM analysis and global-local analysis of a problem such as that illustrated in Figures 11.5-6. Are the answers the same? What is gained, if any, by the global-local approach over the one-stage FEM analysis?

12 Variational Formulation of Bar Element

TABLE OF CONTENTS

§12.1. 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: assembly and solution, remain the same as in Part I. These operations are not element dependent.

Part II constructs *individual elements* beginning with the simplest ones and progressing to more complicated ones. The formulation of two-dimensional 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.

The simplest finite elements from a geometrical standpoint are one-dimensional or *line elements*. The term means that the *intrinsic dimensionality* is one, although they 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 as the truss member.

In this Chapter the stiffness equations of that bar element 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.

§12.2. DEFINITION OF BAR MEMBER

In structural mechanics a *bar* is a structural component characterized by two properties:

(1) One bar dimension: the *longitudinal dimension* or *axial dimension* is much larger that 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 12.1.

(2) The bar resists an internal axial force along its longitudinal dimension.

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 10.

We will consider here only *straight bars,* although their cross section may vary. The one-dimensional mathematical model assumes that the bar material is linearly elastic, obeying Hooke's law, and that displacements and strains are infinitesimal. Figure 12.2 pictures the relevant quantities for a fixed-free bar. Table 12.1 collects the necessary terminology for the governing equations.

Figure 12.3 displays the governing equations of the bar in a graphic format called a *Tonti diagram*. The formal similarity with the diagrams used in Chapter 6 to explain MoM elements should be noted, although the diagram of Figure 12.3 pertains to the continuum model rather than to the discrete one.

Table 12.1 Nomenclature for Mathematical Model of Axially Loaded Bar

Quantity	*Meaning*
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
$p = A\sigma = EAe = EAu'$	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.

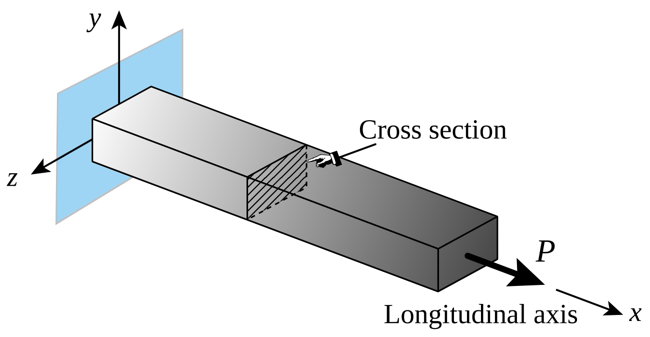


Figure 12.1. A fixed-free bar member.

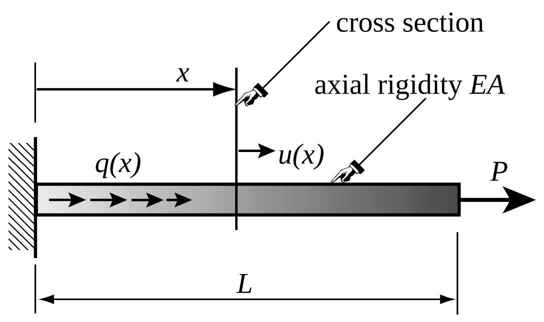


Figure 12.2. Quantities that appear in analysis of bar.

§12.3. VARIATIONAL FORMULATION

To illustrate the variational formulation, the finite element equations of the bar will be derived from the Minimum Potential Energy principle.

§12.3.1. 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

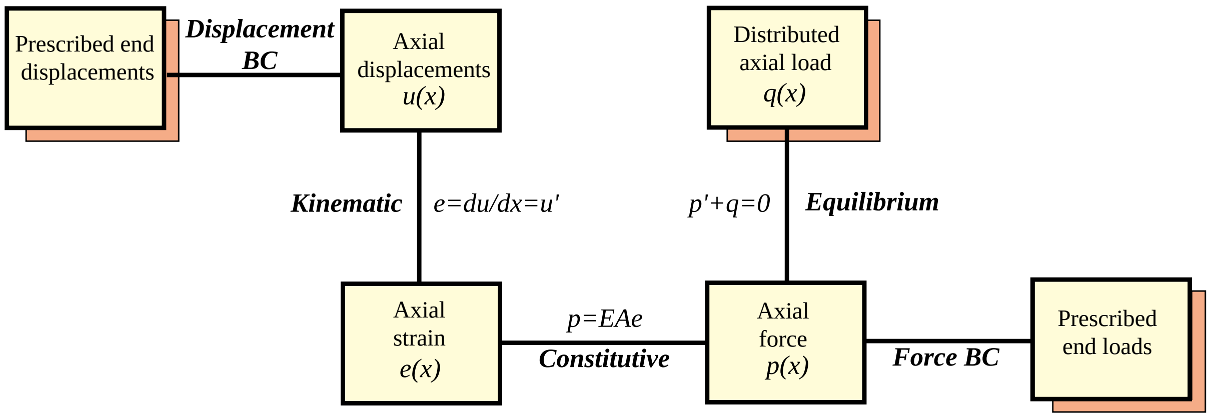


Figure 12.3. Tonti diagram for the mathematical model of a bar member. The diagram displays the field equations and boundary conditions as lines connecting the boxes. Boxes shown in relief contain known (given) quantities.

over the volume of the bar gives the total internal energy

$$U = \tfrac{1}{2}\int_0^L pe\,dx = \tfrac{1}{2}\int_0^L (EAu')u'\,dx = \tfrac{1}{2}\int_0^L u'EAu'\,dx, \tag{12.1}$$

in which all integrand quantities may depend on x.

The *external energy* due to applied mechanical loads pools 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 applied end load(s). For the fixed-free example of Figure 12.2 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 $q(x)$, where $\delta(a)$ denotes the one-dimensional Dirac delta function at $x = a$. If this is done the external energy can be concisely expressed as

$$W = \int_0^L qu\,dx. \tag{12.2}$$

The total potential energy of the bar is given by

$$\boxed{\Pi = U - W} \tag{12.3}$$

Mathematically this is a functional, called the *Total Potential Energy* functional or TPE. It depends only on the axial displacement $u(x)$. In variational calculus this 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, $\Pi[u(x)] = U[u(x)] - W[u(x)]$.

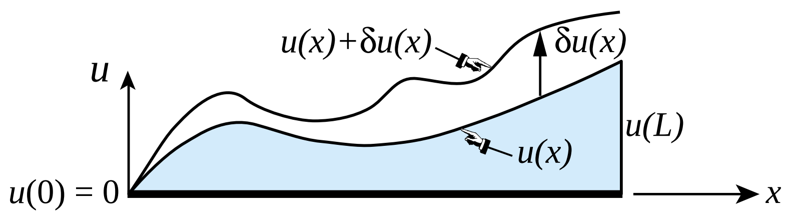


Figure 12.4. Concept of variation of the axial displacement functional $u(x)$. For convenience $u(x)$ is plotted normal to the bar longitudinal axis. Both the $u(x)$ and $u(x) + \delta u(x)$ shown in the figure are kinematically admissible, and so is the variation $\delta u(x)$.

REMARK 12.1

According to the rules of Variational Calculus, the Euler-Lagrange equation for Π is

$$\frac{\partial \Pi}{\partial u} - \frac{d}{dx}\frac{\partial \Pi}{\partial u'} = -q - (E A\, u')' = 0 \tag{12.4}$$

This is the equation of equilibrium in terms of the axial displacement, usually written $(E A\, u')' + q = 0$. This equation is not used in the FEM development.

§12.3.2. Variation of an Admissible Function

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 (12.4) 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 12.4, where for convenience $u(x)$ is plotted normal to x. The functional changes from Π to $\Pi + \delta\Pi$. 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), which agrees with the conditions stated in the classic variational calculus. A kinematically admissible axial displacement $u(x)$ obeys two conditions:

(i) It is continuous over the bar length, that is, $u(x) \in \mathcal{C}_0$ in $x \in [0, L]$.

(ii) It satisfies exactly any displacement boundary condition, such as the fixed-end specification $u(0) = 0$ of Figure 12.2.

The variation $\delta\, u(x)$ depicted in Figure 12.4 is kinematically admissible because both $u(x)$ and $u(x) + \delta\, u(x)$ satisfy the foregoing conditions. The physical meaning of (i)–(ii) is the subject of Exercise 12.1.

[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.

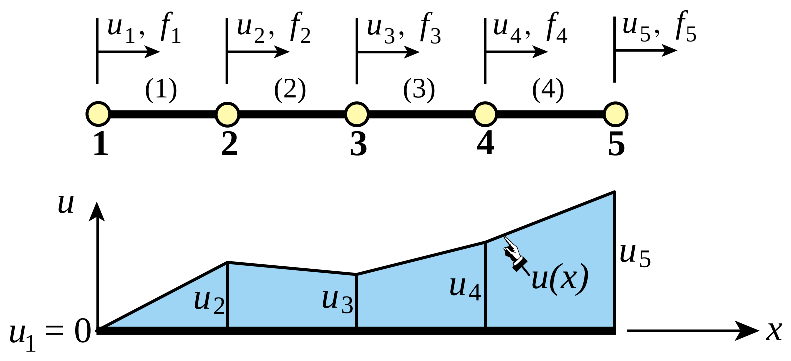


Figure 12.5. FEM discretization of bar member. A piecewise-linear admissible displacement trial function $u(x)$ is drawn below the mesh. It is assumed that the left end is fixed, hence $u_1 = 0$.

§12.3.3. The Minimum Potential Energy Principle

The Minimum Potential Energy (MPE) principle states[2] that the actual displacement solution $u^*(x)$ that satisfies the governing equations is that which renders Π stationary:

$$\boxed{\delta\Pi = \delta U - \delta W = 0 \quad \text{iff} \quad u = u^*} \tag{12.5}$$

with respect to *admissible* variations $u = u^* + \delta u$ of the exact displacement field $u^*(x)$. (The symbol "iff" in (12.5) is an abbreviation for "if and only if".)

REMARK 12.2

Using standard techniques of variational calculus[3] it can be shown that if $EA > 0$ the solution $u^*(x)$ of (12.5) exists, is unique, and renders $\Pi[u]$ a minimum over the class of kinematically admissible displacements. The last attribute explains the "mininum" in the name of the principle.

§12.3.4. TPE Discretization

To apply the TPE functional (12.4) to the derivation of finite element equations we replace the continuum mathematical model by a discrete one consisting of a union of bar elements. For example, Figure 12.5 illustrates the subdivision of a bar member into four two-node elements.

Functionals are scalars. Consequently, corresponding to a discretization such as that shown in Figure 12.5, the TPE functional (12.4) may be decomposed into a sum of contributions of individual elements:

$$\boxed{\Pi = \Pi^{(1)} + \Pi^{(2)} + \ldots + \Pi^{(N_e)}} \tag{12.6}$$

[2] The proof may be found in texts on variational methods in mechanics, e.g., H. L. Langhaar, *Energy Methods in Applied Mechanics*, McGraw-Hill, 1960. This is the most readable "old fashioned" treatment of the energy principles of structural mechanics, with a beautiful treatment of virtual work. Out of print but used copies may be found from web sites.

[3] See for example, I. M. Gelfand and S. V. Fomin [12.1].

where N_e is the number of elements. The same decomposition applies to the internal and external energies, as well as to the stationarity condition (12.5):

$$\boxed{\delta\Pi = \delta\Pi^{(1)} + \delta\Pi^{(2)} + \ldots + \delta\Pi^{(N_e)} = 0.} \tag{12.7}$$

Using the fundamental lemma of variational calculus,[4] it can be shown that (12.7) implies that for a generic element e we may write

$$\boxed{\delta\Pi^{(e)} = \delta U^{(e)} - \delta W^{(e)} = 0.} \tag{12.8}$$

This *variational equation*[5] is the basis for the derivation of element stiffness equations once the displacement field has been discretized over the element, as described next.

§12.3.5. Bar Element Discretization

Figure 12.5 depicts a generic bar element e. The element is referred to its local axis x. Note that there is no need to call it $\bar{x}$ because in one dimension local and global coordinates coalesce. The two degrees of freedom are u_i and u_j.

The mathematical concept of bar finite elements is based on *approximation* of the axial displacement $u(x)$ over the element. The exact displacement u^* is replaced by an approximate displacement

$$u^*(x) \approx u(x) \tag{12.9}$$

over the finite element mesh. The value of $u(x)$ over element e is denoted by $u^{(e)}(x)$. This approximate displacement, $u(x)$, taken over all elements, is called the *finite element trial expansion* or simply *trial function*. See Figure 12.5.

This FE trial expansion must belong to the class of kinematically admissible displacements defined in §12.3.2. Consequently, it must be $\mathcal{C}_0$ continuous over and between elements.

§12.3.6. Shape Functions

For a two node bar element the only possible variation of the displacement $u^{(e)}$ which satisfies the interelement continuity requirement stated above is *linear*, and expressable by the interpolation formula

$$\boxed{u^{(e)}(x) = N_i^{(e)} u_i^{(e)} + N_j^{(e)} u_j^{(e)} = [\, N_i^{(e)} \quad N_j^{(e)} \,] \begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \mathbf{N}\mathbf{u}^{(e)}.} \tag{12.10}$$

The functions $N_i^{(e)}$ and $N_j^{(e)}$ that multiply the node displacements u_i and u_j are called *shape functions*. These functions *interpolate* the internal displacement $u^{(e)}$ directly from the node values.

[4] See Chapter II of Gelfand and Fomin [12.1].

[5] Mathematically called a *weak form* or *Galerkin form*. Equation (12.7) also states the Principle of Virtual Work 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 [12.4].

See Figure 12.6. For our element the shape functions are linear:

$$N_i^{(e)} = 1 - \frac{x}{\ell} = 1 - \zeta, \qquad N_j^{(e)} = \frac{x}{\ell} = \zeta, \tag{12.11}$$

in which $\ell = L^{(e)}$ denotes the element length, and $\zeta = x/\ell$ is a dimensionless coordinate, also known as a *natural coordinate*. Note that the shape function $N_i^{(e)}$ has the value 1 at node i and 0 at node j. Conversely, shape function $N_j^{(e)}$ has the value 0 at node i and 1 at node j. This is a general property of shape functions, required by the fact that that element displacement interpolations such as (12.10) are based on physical node values.

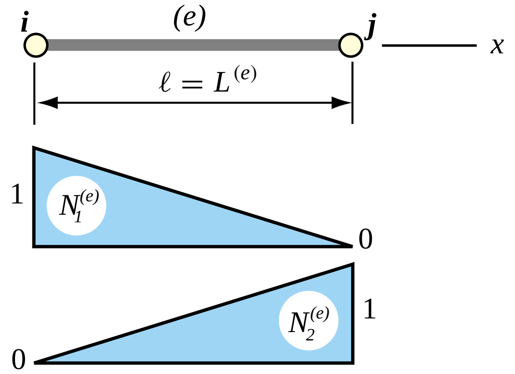


Figure 12.6. The shape functions of the generic bar element.

REMARK 12.3

In addition to continuity, shape 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 (12.11) do satisfy this requirement.

§12.3.7. The Strain-Displacement Equation

The axial strain over the element is

$$e = \frac{du^{(e)}}{dx} = (u^{(e)})' = \begin{bmatrix} \dfrac{dN_i^{(e)}}{dx} & \dfrac{dN_j^{(e)}}{dx} \end{bmatrix} \begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \frac{1}{\ell}\,[\,-1 \quad 1\,] \begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \mathbf{B}\mathbf{u}^{(e)}, \tag{12.12}$$

where

$$\mathbf{B} = \frac{1}{\ell}\,[\,-1 \quad 1\,] \tag{12.13}$$

is called the *strain-displacement* matrix.

§12.4. THE FINITE ELEMENT EQUATIONS

In linear finite element analysis, the discretization process for the TPE functional invariably leads to the following algebraic form

$$\Pi^{(e)} = U^{(e)} - W^{(e)}, \quad U^{(e)} = \tfrac{1}{2}(\mathbf{u}^{(e)})^T\mathbf{K}^{(e)}\mathbf{u}^{(e)}, \quad W^{(e)} = (\mathbf{u}^{(e)})^T\mathbf{f}^{(e)}, \tag{12.14}$$

where $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ are called the *element stiffness matrix* and the *element consistent nodal force vector*, respectively. Note that in (12.14) the three energies are only function of the node displacements $\mathbf{u}^{(e)}$. $U^{(e)}$ and $W^{(e)}$ depend quadratically and linearly, respectively, on those displacements.

Taking the variation of the discretized TPE of (12.14) with respect to the node displacements gives[6]

$$\delta\Pi^{(e)} = \left(\delta\mathbf{u}^{(e)}\right)^T \frac{\partial\Pi^{(e)}}{\partial\mathbf{u}^{(e)}} = \left(\delta\mathbf{u}^{(e)}\right)^T \left[\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{f}^{(e)}\right] = 0. \tag{12.15}$$

[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.

Because the variations $\delta\mathbf{u}^{(e)}$ can be arbitrary, the bracketed quantity must vanish, which yields

$$\boxed{\mathbf{K}^{(e)}\mathbf{u}^{(e)} = \mathbf{f}^{(e)}} \tag{12.16}$$

These are the element stiffness equations. Hence the foregoing names given to $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ are justified *a posteriori*.

§12.4.1. The Stiffness Matrix

For the two-node bar element, the internal energy $U^{(e)}$ is

$$U^{(e)} = \tfrac{1}{2}\int_0^{\ell} e\,EA\,e\,dx, \tag{12.17}$$

where the strain e is related to the nodal displacements through (12.12). This form is symmetrically expanded by inserting $e = \mathbf{B}\mathbf{u}^{(e)}$ into the second e and $e = e^T = (\mathbf{u}^{(e)})^T\mathbf{B}^T$ into the first e:

$$U^{(e)} = \tfrac{1}{2}\int_0^{\ell} [\,u_i^{(e)} \quad u_j^{(e)}\,]\,\frac{1}{\ell}\begin{bmatrix} -1 \\ 1 \end{bmatrix} EA\,\frac{1}{\ell}\,[\,-1 \quad 1\,]\begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} dx. \tag{12.18}$$

The nodal displacements can be moved out of the integral, giving

$$U^{(e)} = \tfrac{1}{2}\,[\,u_i^{(e)} \quad u_j^{(e)}\,]\int_0^{\ell}\frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx\;\begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \tfrac{1}{2}\big(\mathbf{u}^{(e)}\big)^T\mathbf{K}^{(e)}\mathbf{u}^{(e)}. \tag{12.19}$$

in which

$$\boxed{\mathbf{K}^{(e)} = \int_0^{\ell}\frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx = \int_0^{\ell} EA\,\mathbf{B}^T\mathbf{B}\,dx,} \tag{12.20}$$

is the element stiffness matrix. If the rigidity EA is constant over the element,

$$\mathbf{K}^{(e)} = EA\,\mathbf{B}^T\mathbf{B}\int_0^{\ell} dx = \frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\ell = \frac{EA}{\ell}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{12.21}$$

This is the same element stiffness matrix of the prismatic truss member derived in Chapters 2 and 6 by Mechanics of Materials arguments, but now obtained through a variational method.

§12.4.2. The Consistent Node Force Vector

The *consistent node force vector* $\mathbf{f}^{(e)}$ introduced in (12.14) comes from the element contribution to the external work potential W:

$$W^{(e)} = \int_0^{\ell} qu\,dx = \int_0^{\ell} q\,\mathbf{N}^T\mathbf{u}^{(e)}\,dx = \big(\mathbf{u}^{(e)}\big)^T\int_0^{\ell} q\begin{bmatrix} 1-\zeta \\ \zeta \end{bmatrix} dx = \big(\mathbf{u}^{(e)}\big)^T\,\mathbf{f}^{(e)}, \tag{12.22}$$

in which $\zeta = x/\ell$. Consequently

$$\boxed{\mathbf{f}^{(e)} = \int_0^{\ell} q\begin{bmatrix} 1-\zeta \\ \zeta \end{bmatrix} dx} \tag{12.23}$$

If the force q is constant over the element, one obtains the same results as with the EbE load-lumping method of Chapter 8. See Exercise 12.3.

§12.5. *NODAL EXACTNESS AND SUPERCONVERGENCE

Suppose that the following three conditions are satisfied:

1. The bar properties are constant along the length (prismatic member).
2. The distributed load $q(x)$ is zero between nodes.
3. The only applied loads are point forces applied at the nodes.

If so, a linear axial displacement $u(x)$ as defined by (12.10) and (12.11) is the exact solution over each element because constant strain and stress satisfy, element by element, all of the governing equations listed in Figure 12.3.[7] 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. Adding extra elements and nodes would not change the solution. That is the reason behind the truss discretizations used in Chapters 2–4: *one element per member is sufficient* if the members are prismatic and the only loads are applied at the joints.[8] Such FEM 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. For an infinite one-dimensional lattice of equal-length 2-node bar elements, however, an interesting result is *the solution is nodally exact for any loading if consistent node forces are correctly computed.* This can be proven using Fourier analysis, following the technique outlines in Exercise E12.8. This result underlies the importance of computing node forces correctly.

If the 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) if nodes forces are consistently computed. This phenomenon is called *superconvergence* in the FEM literature.

§12.6. NOTES AND REFERENCES

The foregoing development pertains to the simplest finite element possible: the linear two-node bar element. If we stay within the realm of one-dimensional elements the technique may be generalized in the following directions:

Refined bar elements. 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 plate edge reinforcements.) For that reason they are not considered here. The 3-node bar element is developed in Exercise 16.5.

Two and three dimensional structures built with bar elements. The only required extra ingredient are the nodal-displacement transformation matrices discussed in Chapters 3 and 6.

Beam-type elements. These require the introduction of displacement-derivative nodal freedoms (interpretable as nodal rotations). Plane beam elements are treated in the next Chapter.

Curved elements. These are derivable using isoparametric mapping. This device will be introduced later when considering triangular and quadrilateral elements in Chapter 16.

[7] The internal equilibrium equation $p' + q = 0$ is trivially verified because $p' = q = 0$.

[8] In fact, adding more elements per member makes the stiffness equations singular in 2D and 3D because zero-energy mechanisms appear. Removing those mechanisms would require the application of MFCs at intermediate nodes. See Exercise E12.7.

Matrices for straight bar elements are available in any finite element book; for example Przemieniecki [12.5]. The mathematical fundamentals of Variational Calculus may be studied in Gelfand and Fomin [12.1] which is available now in an inexpensive Dover edition.

Nodal exactness for arbitrary loading was proven by Park and Flaggs [12.2,12.3]. The concept of superconvergence is readably presented in the book of Sttrang and Fix [12.6].

References

[12.1] I. M. Gelfand and S. V. Fomin, *Calculus of Variations*, Prentice-Hall, 1963. Reprinted by Dover, 2000.

[12.2] Park, K. C. and Flaggs, D. L. (1984a), “An operational procedure for the symbolic analysis of the finite element method,” *Comp. Meths. Appl. Mech. Engrg.*, **46**, 65–81, 1984.

[12.3] Park, K. C. and Flaggs, D. L. (1984b), “A Fourier analysis of spurious modes and element locking in the finite element method,” *Comp. Meths. Appl. Mech. Engrg.*, **42**, 37–46, 1984.

[12.4] E. P. Popov, *Engineering Mechanics of Solids*, Prentice Hall, Englewood Cliffs, N. J., 2nd ed., 1991.

[12.5] J. S. Przemieniecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968; Dover edition 1986.

[12.6] G. Strang and G. Fix, *An Analysis of the Finite Element Method.* Prentice-Hall, 1973.

Homework Exercises for Chapter 12
Variational Formulation of Bar Element

EXERCISE 12.1

[D:10] Explain the kinematic admissibility requirements stated in §12.3.2 in terms of physics, namely ruling out the possibility of gaps or interpenetration as the bar material deforms.

EXERCISE 12.2

[A/C:15] Using the expression (12.20), derive the stiffness matrix for a *tapered* bar element in which the cross section area varies linearly along the element length:

$$A = A_i(1-\zeta) + A_j\,\zeta \tag{E12.1}$$

where A_i and A_j are the areas at the end nodes, and $\zeta = x/\ell$ is the dimensionless coordinate defined in §12.3.6. 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:[9]

```
ClearAll[Le,x,Em,A,Ai,Aj];
Be={{-1,1}}/Le; ξ=x/Le; A=Ai*(1-ξ)+Aj*ξ;
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 12.3

[A:10] Using the expression (12.23), 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.

EXERCISE 12.4

[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 (E12.1) 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:[10]

```
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];
```

[9] 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.

[10] The `ClearAll[A]` before the last statement is essential; else `A` would retain the previous assignation.

EXERCISE 12.5

[A/C:20] A tapered bar element of length ℓ, end areas A_i and A_j with A interpolated as per (E12.1), 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. 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 $A = A_i = A_j$.

EXERCISE 12.6

[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 Equation (12.23), with $q(x) = Q\,\delta(a)$, 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];
ξ=x/Le; Ne={{1-ξ,ξ}}; 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 12.7

[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 5 and verify that the solution "blows up" because the modified master stiffness is singular. Explain physically what happens.

13 Variational Formulation of Plane Beam Element

TABLE OF CONTENTS

§13.1. 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.

§13.2. WHAT IS A BEAM?

Beams are the most common type of structural component, particularly in Civil and Mechanicsl Engineering. A *beam* is a bar-like structural member whose primary function is to support *transverse* loading and carry to the supports. See Figure 13.1. 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.

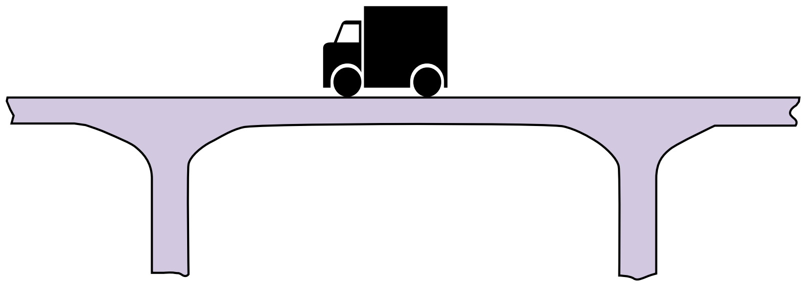


Figure 13.1. A beam is a structural member designed to resist transverse loads.

A beam resists transverse loads mainly through *bending action*, as illustrated in Figure fig-BeamVFResistance. Such 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.

§13.2.1. 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.

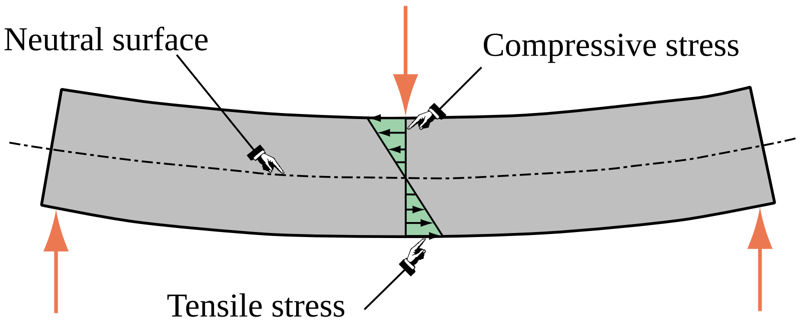


Figure 13.2. Beam transverse loads are primarily resisted by bending action.

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.

§13.2.2. 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* theory, also called *classical beam theory* or *engineering beam theory*, and the *Timoshenko beam theory*. The Bernoulli-Euler theory is that taught in introductory Mechanics of Materials, and is the one emphasized in this Chapter. The Timoshenko beam model is briefly outlined on §13.7, which contains advanced material.

Both models can be used to formulate beam finite elements. The classical beam theory used here leads to the so-called *Hermitian* beam elements.[1] These are also known as C^1 elements for the reason explained in §13.5.1. This model neglects transverse shear deformations. 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.

§13.2.3. Assumptions of Classical Beam Theory

The classical (Bernoulli-Euler) theory for *plane beams* is based 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 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 beam member accounts only for bending moment deformations. All other effects, notably transverse shear and axial force effects, are ignored.

[1] The qualifier “Hermitian” relates to the use of a interpolation formula studied by the French mathematician Hermite. The term has nothing to do with the beam model used.

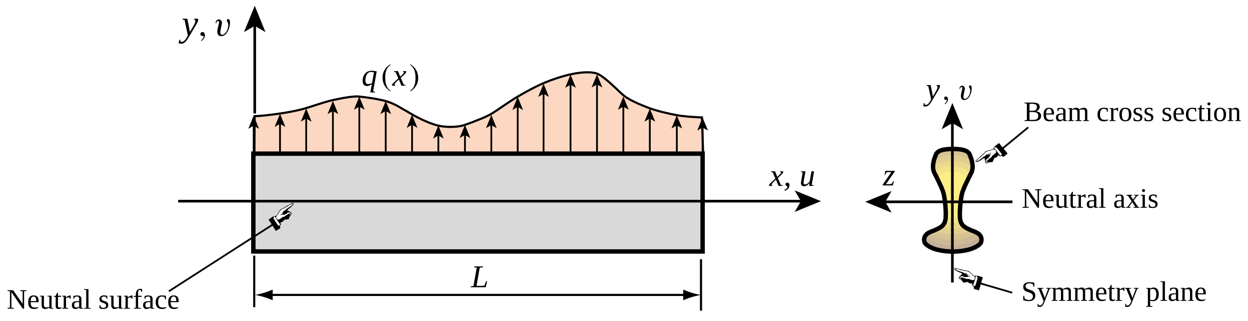


Figure 13.3. Terminology in Bernoulli-Euler model of plane beam.

5. *Linearization*. Transverse deflections, rotations and deformations are considered so small that the assumptions of infinitesimal deformations apply.
6. *Elastic behavior*. The beam is fabricated of material assumed to be elastic and isotropic. Heterogeneous beams fabricated with several materials, such as reinforced concrete, are not excluded.

§13.3. THE CLASSICAL BEAM THEORY

§13.3.1. The Neutral Axis

Under transverse loading one of the beam surfaces shortens while the other elongates; see Figure 13.2. Therefore a *neutral surface* exists between the top and the bottom that undergoes no elongation or contraction. The intersection of this surface with each cross section defines the *neutral axis* of that cross section. If the beam is fabricated of uniform material, the position of the neutral axis is only a function of the cross section geometry.

§13.3.2. Element Coordinate Systems

The Cartesian axes for plane beam analysis are chosen as follows:

1. x along the longitudinal beam axis, at neutral axis height.[2]
2. z along the neutral axis at the origin section.
3. y upwards forming a RHS system with x and z.

The origin of the x, y, z system is placed at the the leftmost section. The total length of the beam member is L. See Figure 13.3.

[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 axes passes through the centroid of an "equivalent" cross section. This topic is covered in Mechanics of Materials textbooks; for example Popov [13.10].

§13.3.3. Kinematics

The *motion* of plane beam member in the x, y plane is described by the two dimensional displacement field

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}, \tag{13.1}$$

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 primarity due to Poisson's ratio effects, is of no interest. The normality assumption of the classical (Bernoulli-Euler) model can be represented mathematically as

$$u(x,y) = -y\frac{\partial v(x)}{\partial x} = -yv' = -y\theta, \qquad v(x,y) = v(x). \tag{13.2}$$

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 counterclockwise.

§13.3.4. 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 13.3.

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 counterclockwise, it contributes $C\delta'(b)$ to $q(x)$, where δ' denotes a doublet acting at $x = b$.

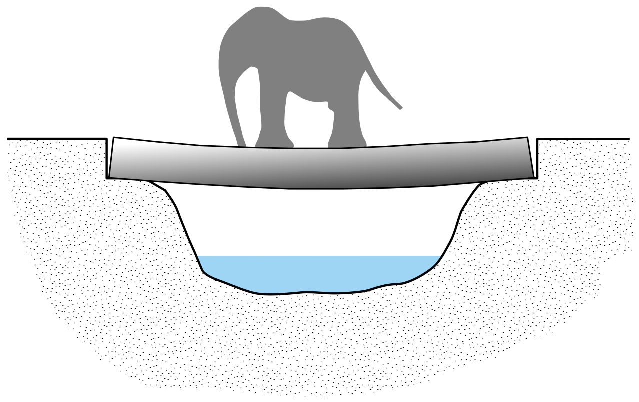


Figure 13.4. A simply supported beam has end supports that preclude transverse displacements but permit end rotations.

§13.3.5. Support Conditions

Support conditions for beams exhibit far more variety than for bar members. Two canonical cases often encountered in engineering practice: simple support and cantilever support. These are illustrated in Figures 13.4 and 13.5, respectively. Beams often appear as components of skeletal structures called frameworks, in which case the support conditions are of more complex type.

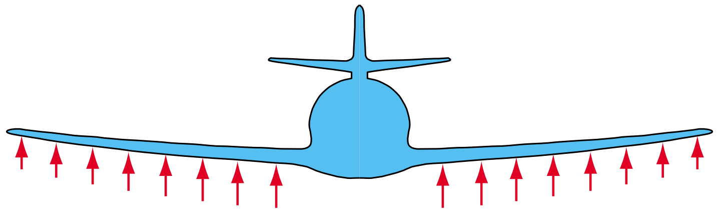


Figure 13.5. A cantilever beam is clamped at one end and free at the other. Airplane wings and stabilizers provide examples of this configuration.

§13.3.6. Strains, Stresses and Bending Moments

The classical (Bernoulli-Euler) 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 (13.2):

$$e = \frac{\partial u}{\partial x} = -y\frac{\partial^2 v}{\partial x^2} = -y\frac{d^2 v}{dx^2} = -y\kappa. \tag{13.3}$$

Here κ denotes the deformed beam axis curvature, which to first order is $\partial^2 v/\partial x^2 \approx v''$. The bending stress $\sigma = \sigma_{xx}$ is linked to e through the one-dimensional Hooke's law

$$\sigma = Ee = -Ey\frac{d^2 v}{dx^2} = -Ey\kappa, \tag{13.4}$$

where E is the elastic modulus.

The most important stress resultant in classical beam theory is the *bending moment* M, which is defined as the cross section integral

$$M = \int_A -y\sigma\, dx = E\frac{d^2 v}{dx^2}\int_A y^2\, dA = EI\,\kappa. \tag{13.5}$$

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. M is considered positive if it compresses the upper portion: $y > 0$, of the beam cross section, as illustrated in Figure 13.6. This explains the negative sign of y in the integral (13.5). The product EI is called the *bending rigidity* of the beam with respect to flexure about the z axis.

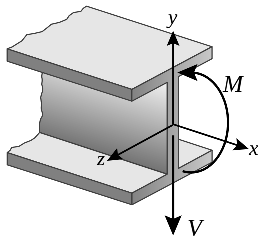


Figure 13.6. Positive sign convention for M and V.

The governing equations of the Bernoulli-Euler beam model are summarized in the Tonti diagram of Figure 13.7.

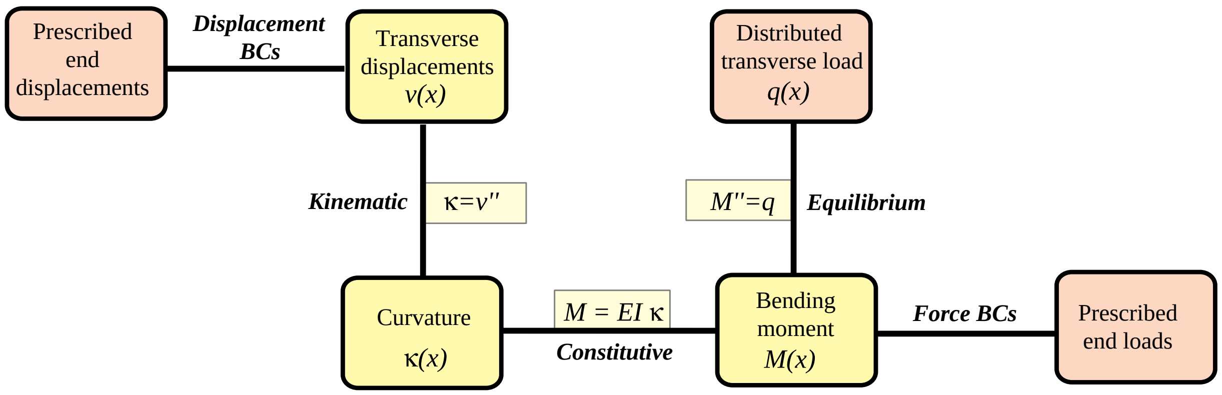


Figure 13.7. The Tonti diagram for the governing equations of the Bernoulli-Euler beam model.

§13.4. TOTAL POTENTIAL ENERGY FUNCTIONAL

The total potential energy of the beam is

$$\boxed{\Pi = U - W} \tag{13.6}$$

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:

$$\boxed{U = \tfrac{1}{2}\int_V \sigma e\, dV = \tfrac{1}{2}\int_0^L M\kappa\, dx = \tfrac{1}{2}\int_0^L EI\kappa^2\, dx = \tfrac{1}{2}\int_0^L EI\left(v''\right)^2 dx = \tfrac{1}{2}\int_0^L v'' EI v''\, dx.} \tag{13.7}$$

The external work W accounts for the applied transverse force:

$$\boxed{W = \int_0^L q\, v\, dx.} \tag{13.8}$$

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 $W[v]$.

Observe that $\Pi[v]$ includes up to second derivatives in v, because $v'' = \kappa$ appears in U. This number (the order of the highest derivatives present in the functional) is called the *variational index*. Variational calculus tells us that since the variational index is 2, admissible displacements $v(x)$ must be continuous, have continuous first derivatives (slopes or rotations), and satisfy the displacement boundary conditions exactly.

This continuity requirement can be succintly stated by saying that admissible displacements must be C^1 continuous. This condition guides the construction of beam finite elements described below.

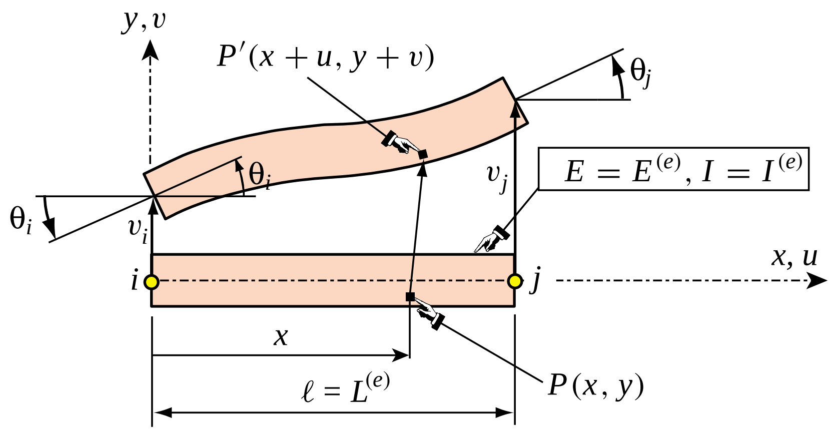


Figure 13.8. The two-node Bernouilli-Euler plane beam element with four degrees of freedom.

REMARK 13.1

If there is an applied distributed moment $m(x)$ per unit of beam length, the external energy (13.8) must be augmented with a $\int_0^L m(x)\theta(x)\,dx$ term. This is further elaborated in Exercises 13.4 and 13.5. Such kind of distributed loading is uncommon in practice although in framework analysis occassionally the need arises for treating a concentrated moment C between nodes.

§13.5. BEAM FINITE ELEMENTS

Beam finite elements are obtained by subdividing beam members longitudinally. The simplest Bernoulli-Euler plane beam element, depicted in Figure 13.8, has two end nodes, i and j, and four degrees of freedom collected in the node displacement vector

$$\mathbf{u}^{(e)} = [\, v_i^{(e)} \quad \theta_i^{(e)} \quad v_j^{(e)} \quad \theta_j^{(e)} \,]^T . \tag{13.9}$$

§13.5.1. Finite Element Trial Functions

The freedoms (13.9) must be used to define uniquely the variation of the transverse displacement $v^{(e)}(x)$ over the element. The C^1 continuity requirement stated at the end of the previous Section says that both w and the slope $\theta = v'$ must be continuous over the entire beam member, and in particular between beam elements.

C^1 continuity can be trivially satisfied within each element by choosing polynomial interpolation functions as shown below. Matching the nodal displacements and rotations with adjacent beam elements enforces the necessary interelement continuity.

§13.5.2. Shape Functions

The simplest shape functions that meet the C^1 continuity requirements for the nodal freedom configuration (13.9) are called the *Hermitian cubic* shape functions. The interpolation formula based on these functions may be written as

$$v^{(e)} = [\, N^{(e)}_{v_i} \quad N^{(e)}_{\theta_i} \quad N^{(e)}_{v_j} \quad N^{(e)}_{\theta_j} \,] \begin{bmatrix} v^{(e)}_i \\ \theta^{(e)}_i \\ v^{(e)}_j \\ \theta^{(e)}_j \end{bmatrix} = \mathbf{N}\,\mathbf{u}^{(e)}. \tag{13.10}$$

These shape functions are conveniently written in terms of the dimensionless "natural" coordinate

$$\xi = \frac{2x}{\ell} - 1, \tag{13.11}$$

which varies from -1 at node i ($x = 0$) to $+1$ at node j ($x = \ell$); ℓ being the element length:

$$\boxed{\begin{aligned} N^{(e)}_{v_i} &= \tfrac{1}{4}(1-\xi)^2(2+\xi), \\ N^{(e)}_{\theta_i} &= \tfrac{1}{8}\ell(1-\xi)^2(1+\xi), \\ N^{(e)}_{v_j} &= \tfrac{1}{4}(1+\xi)^2(2-\xi), \\ N^{(e)}_{\theta_j} &= -\tfrac{1}{8}\ell(1+\xi)^2(1-\xi). \end{aligned}} \tag{13.12}$$

These four functions are depicted in Figure 13.9.

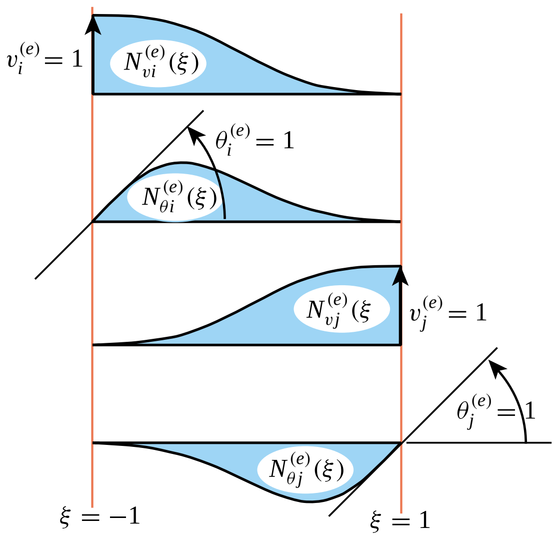


Figure 13.9. Cubic shape functions of plane beam element.

The curvature κ that appears in U can be expressed in terms of the nodal displacements by differentiating twice with respect to x:

$$\kappa = \frac{d^2 v^{(e)}(x)}{dx^2} = \frac{4}{\ell^2}\frac{d^2 v^{(e)}(\xi)}{d\xi^2} = \frac{4}{\ell^2}\frac{d\mathbf{N}^{(e)}}{d\xi^2}\mathbf{u}^{(e)} = \mathbf{B}\mathbf{u}^{(e)} = \mathbf{N}''\mathbf{u}^{(e)}. \tag{13.13}$$

Here $\mathbf{B} = \mathbf{N}''$ is the 1×4 curvature-displacement matrix

$$\boxed{\mathbf{B} = \frac{1}{\ell}\Big[\, 6\frac{\xi}{\ell} \quad 3\xi - 1 \quad -6\frac{\xi}{\ell} \quad 3\xi + 1 \,\Big]}. \tag{13.14}$$

REMARK 13.2

The $4/\ell^2$ factor that shows up in (13.13) comes from the differentiation chain rule. If $f(x)$ is a function of x, and $\xi = 2x/\ell - 1$,

$$\begin{aligned} \frac{df(x)}{dx} &= \frac{df(\xi)}{d\xi}\frac{d\xi}{dx} = \frac{2}{\ell}\frac{df(\xi)}{d\xi}, \\ \frac{d^2 f(x)}{dx^2} &= \frac{d(2/\ell)}{dx}\frac{df(\xi)}{d\xi} + \frac{2}{\ell}\frac{d}{dx}\left(\frac{df(\xi)}{d\xi}\right) = \frac{4}{\ell^2}\frac{d^2 f(\xi)}{d\xi^2}, \end{aligned} \tag{13.15}$$

because $d(2/\ell)/dx = 0$.

§13.6. THE FINITE ELEMENT EQUATIONS

Insertion of (13.12) and (13.14) into the TPE functional specialized to the element, yields the quadratic form in the nodal displacements

$$\Pi^{(e)} = \tfrac{1}{2}\mathbf{u}^{(e)\,T}\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{u}^{(e)\,T}\mathbf{f}^{(e)}, \tag{13.16}$$

where

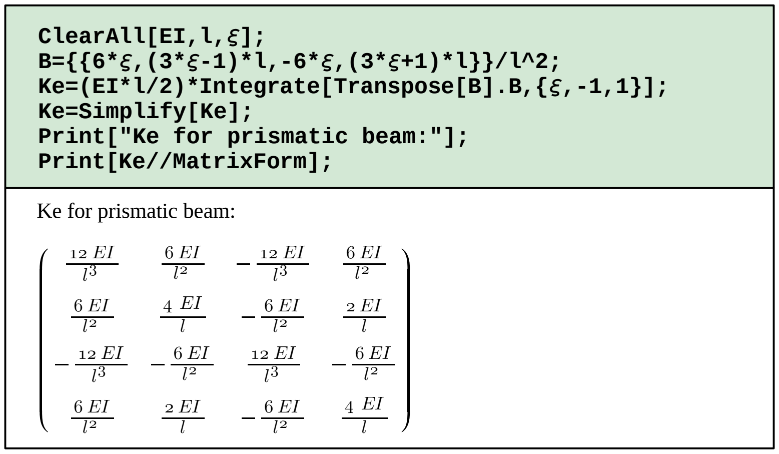
$$\boxed{\mathbf{K}^{(e)} = \int_0^\ell EI\ \mathbf{B}^T\mathbf{B}\,dx = \int_{-1}^{1} EI\ \mathbf{B}^T\mathbf{B}\ \tfrac{1}{2}\ell\,d\xi,} \tag{13.17}$$

is the element stiffness matrix and

$$\boxed{\mathbf{f}^{(e)} = \int_0^\ell \mathbf{N}^T q\,dx = \int_{-1}^{1} \mathbf{N}^T q\ \tfrac{1}{2}\ell\,d\xi,} \tag{13.18}$$

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.



```
ClearAll[EI,l,ξ];
B={{6*ξ,(3*ξ-1)*l,-6*ξ,(3*ξ+1)*l}}/l^2;
Ke=(EI*l/2)*Integrate[Transpose[B].B,{ξ,-1,1}];
Ke=Simplify[Ke];
Print["Ke for prismatic beam:"];
Print[Ke//MatrixForm];
```

Ke for prismatic beam:

$$\begin{pmatrix} \frac{12\,EI}{l^3} & \frac{6\,EI}{l^2} & -\frac{12\,EI}{l^3} & \frac{6\,EI}{l^2} \\ \frac{6\,EI}{l^2} & \frac{4\,EI}{l} & -\frac{6\,EI}{l^2} & \frac{2\,EI}{l} \\ -\frac{12\,EI}{l^3} & -\frac{6\,EI}{l^2} & \frac{12\,EI}{l^3} & -\frac{6\,EI}{l^2} \\ \frac{6\,EI}{l^2} & \frac{2\,EI}{l} & -\frac{6\,EI}{l^2} & \frac{4\,EI}{l} \end{pmatrix}$$

Figure 13.10. Using *Mathematica* to form $\mathbf{K}^{(e)}$ for a prismatic beam element.

§13.6.1. 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 (13.17):

$$\mathbf{K}^{(e)} = \tfrac{1}{2}EI\,\ell \int_{-1}^{1} \mathbf{B}^T\mathbf{B}\,d\xi = \frac{EI}{2\ell}\int_{-1}^{1} \begin{bmatrix} \dfrac{6\xi}{\ell} \\ 3\xi-1 \\ \dfrac{-6\xi}{\ell} \\ 3\xi+1 \end{bmatrix} \begin{bmatrix} \dfrac{6\xi}{\ell} & 3\xi-1 & \dfrac{-6\xi}{\ell} & 3\xi+1 \end{bmatrix} d\xi. \tag{13.19}$$

Expanding and integrating over the element yields

$$\mathbf{K}^{(e)} = \frac{EI}{2\ell^3}\int_{-1}^{1}\begin{bmatrix} 36\xi^2 & 6\xi(3\xi-1)\ell & -36\xi^2 & 6\xi(3\xi+1)\ell \\ & (3\xi-1)^2\ell^2 & -6\xi(3\xi-1)\ell & (9\xi^2-1)\ell^2 \\ & & 36\xi^2 & -6\xi(3\xi+1)\ell \\ symm & & & (3\xi+1)^2\ell^2 \end{bmatrix} d\xi$$

$$= \frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ & 4\ell^2 & -6\ell & 2\ell^2 \\ & & 12 & -6\ell \\ symm & & & 4\ell^2 \end{bmatrix} \tag{13.20}$$

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 13.10 processes (13.20) using the `Integrate` function. The output, shown in the bottom box, corroborates the hand integration result.

§13.6.2. Consistent Nodal Force Vector for Uniform Load

If q does not depend on x it can be moved out of (13.18), giving

$$\mathbf{f}^{(e)} = \tfrac{1}{2}q\ell\int_{-1}^{1}\mathbf{N}^T\,d\xi = \tfrac{1}{2}q\ell\int_{-1}^{1}\begin{bmatrix} \frac{1}{4}(1-\xi)^2(2+\xi) \\ \frac{1}{8}\ell(1-\xi)^2(1+\xi) \\ \frac{1}{4}(1+\xi)^2(2-\xi) \\ -\frac{1}{8}\ell(1+\xi)^2(1-\xi) \end{bmatrix} d\xi = q\ell\begin{bmatrix} \frac{1}{2} \\ \frac{1}{12}\ell \\ \frac{1}{2} \\ -\frac{1}{12}\ell \end{bmatrix}. \tag{13.21}$$

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 FEM literature.

```
ClearAll[q,l,ξ]
Ne={{2*(1-ξ)^2*(2+ξ), (1-ξ)^2*(1+ξ)*l,
     2*(1+ξ)^2*(2-ξ),-(1+ξ)^2*(1-ξ)*l}}/8;
fe=(q*l/2)*Integrate[Ne,{ξ,-1,1}]; fe=Simplify[fe];
Print["fe^T for uniform load q:"];
Print[fe//MatrixForm];
```

fe^T for uniform load q:

$$\begin{pmatrix} \frac{l\,q}{2} & \frac{l^2\,q}{12} & \frac{l\,q}{2} & -\frac{l^2\,q}{12} \end{pmatrix}$$

Figure 13.11. Using *Mathematica* to form $\mathbf{f}^{(e)}$ for uniform transverse load q.

The hand result (13.21) can be verified with the *Mathematica* script displayed in Figure 13.11, in which $\mathbf{f}^{(e)}$ is printed as a row vector to save space.

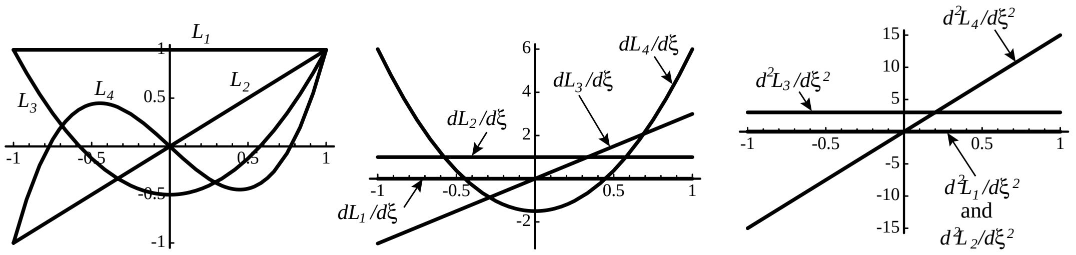


Figure 13.12. The Legendre polynomials and their first two derivatives plotted over $\xi \in [-1, 1]$.

§13.7. *GENERALIZED INTERPOLATION

For derivation of nonstandard and C^0 beam elements it is convenient to use a transverse-displacement interpolation in which the nodal freedoms w_i, w_j, θ_i and θ_j are replaced by *generalized coordinates* q_1 through q_4:

$$v(\xi) = N_{q1}\, q_1 + N_{q2}\, q_2 + N_q\, q_3 + N_{q4}\, q_4 = \mathbf{N}_q\, \mathbf{q}. \tag{13.22}$$

The N_{qi} are functions of ξ that satisfy the completeness and continuity requirements discussed in Chapter 19. Note that $\mathbf{N}_q$ is a $1{\times}4$ matrix whereas $\mathbf{q}$ is a column 4-vector. This is called a *generalized interpolation*. It includes the physical interpolation (13.10) when $q_1 = w_i$, $q_2 = \theta_i$, $q_3 = w_j$ and $q_4 = \theta_j$. One well known generalized form is the polynomial in ξ:

$$v(\xi) = q_1 + q_2\xi + q_3\xi^2 + q_4\xi^3. \tag{13.23}$$

A particularly seminal form is

$$v(\xi) = L_1\, q_1 + L_2\, q_2 + L_3\, q_3 + L_4\, q_4 = \mathbf{L}\,\mathbf{q}, \tag{13.24}$$

where the L_i are the first four Legendre polynomials[3]

$$L_1(\xi) = 1, \quad L_2(\xi) = \xi, \quad L_3(\xi) = \tfrac{1}{2}(3\xi^2 - 1), \quad L_4(\xi) = \tfrac{1}{2}(5\xi^3 - 3\xi). \tag{13.25}$$

These functions and their first two derivatives are plotted in Figure 13.12. Unlike the shape functions (13.12), the L_i have a clear physical meaning as can be gathered from the picture:

L_1 Translational rigid body mode.

L_2 Rotational rigid body mode.

L_3 Constant-curvature symmetric deformation mode (symmetric with respect to $\xi = 0$)

L_4 Linear-curvature antisymmetric deformation mode (antisymmetric with respect to $\xi = 0$)

These properties are also shared by the standard polynomial interpolation (13.23). What distinguishes the set (13.25) are the orthogonality properties

$$\mathbf{Q} = \int_0^\ell \mathbf{L}^T\,\mathbf{L}\,dx = \ell \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \\ 0 & 0 & 1/5 & 0 \\ 0 & 0 & 0 & 1/7 \end{bmatrix}, \qquad \mathbf{S} = \int_0^\ell (\mathbf{L}'')^T\,\mathbf{L}''\,dx = \frac{48}{\ell^3} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 25 \end{bmatrix}. \tag{13.26}$$

[3] This is a FEM oriented nomenclature; L_1 through L_4 are called P_0 through P_3 in mathematical handbooks; e.g. Chapter 22 of Abramowitz and Stegun [13.1]. They are normalized as per the usual conventions: $L_i(1) = 1$, $L_i(-1) = (-1)^{i-1}$. One important application in numerical analysis is the construction of Gauss integration rules: the abscissas of the p-point rule are the zeros of $L_{p+1}(\xi)$.

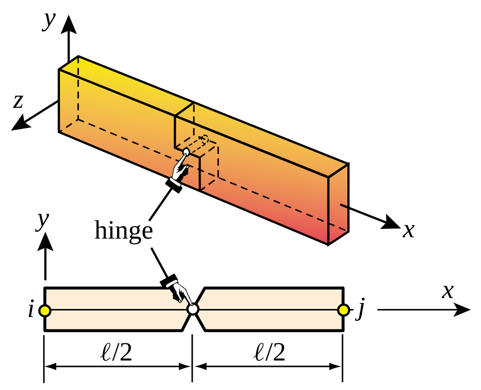


Figure 13.13. Hinged beam element: hinge is at midspan $\xi = 0$. The upper figure sketches a possible fabrication.

The stiffness matrix in terms of the q_i is called the generalized stiffness. Denoting the bending rigidity by R:

$$\mathbf{K}_q = \int_0^\ell R\,(\mathbf{L}'')^T\,\mathbf{L}''\,dx. \tag{13.27}$$

For a prismatic Bernoulli-Euler plane beam element $R = EI$ is constant and $\mathbf{K}_q = EI\,\mathbf{S}$, where $\mathbf{S}$ is the second diagonal matrix in (13.26). With view to future applications it is convenient to differentiate between symmetric and antisymmetric bending rigidities R_s and R_a, which are associated with the responses to deformation modes L_3 and L_4, respectively. Assuming R_s and R_a to be uniform along the element we get

$$\mathbf{K}_q = \mathbf{K}_{qs} + \mathbf{K}_{qa}, \qquad \mathbf{K}_{qs} = \frac{144R_s}{\ell^3}\begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&1&0\\ 0&0&0&0 \end{bmatrix}, \qquad \mathbf{K}_{qa} = \frac{1200R_a}{\ell^3}\begin{bmatrix} 0&0&0&0\\ 0&0&0&0\\ 0&0&0&0\\ 0&0&0&1 \end{bmatrix}. \tag{13.28}$$

For a Bernoulli-Euler model, the generalized coordinates q_i of (13.24) can be connected to the physical DOFs by the transformations

$$\begin{bmatrix} w_i\\ \theta_i\\ w_j\\ \theta_j \end{bmatrix} = \begin{bmatrix} 1&-1&1&-1\\ 0&2/\ell&-6/\ell&12/\ell\\ 1&1&1&1\\ 0&2/\ell&6/\ell&12/\ell \end{bmatrix}\begin{bmatrix} q_1\\ q_2\\ q_3\\ q_4 \end{bmatrix} = \mathbf{G}\mathbf{q}, \quad \begin{bmatrix} q_1\\ q_2\\ q_3\\ q_4 \end{bmatrix} = \frac{1}{60}\begin{bmatrix} 30&5\ell&30&-5\ell\\ -36&-3\ell&36&-3\ell\\ 0&-5\ell&0&-5\ell\\ 6&3\ell&-6&3\ell \end{bmatrix}\begin{bmatrix} w_i\\ \theta_i\\ w_2\\ \theta_j \end{bmatrix} = \mathbf{H}\mathbf{u}^{(e)}. \tag{13.29}$$

in which $\mathbf{H} = \mathbf{G}^{-1}$. The physical stiffness is

$$\mathbf{K}^{(e)} = \mathbf{H}^T\mathbf{K}_q\mathbf{H} = \frac{1}{\ell^3}\begin{bmatrix} 12R_a & 6R_a\ell & -12R_a & 6R_a\ell\\ 6R_a\ell & (3R_a+R_s)\ell^2 & -6R_a\ell & (3R_a-R_s)\ell^2\\ -12R_a & -6R_a\ell & 12R_a & -6R_a\ell\\ 6R_a\ell & (3R_a-R_s)\ell^2 & -6R_a\ell & (3R_a+R_s)\ell^2 \end{bmatrix} \tag{13.30}$$

If $R_s = R_a = EI$ the well known stiffness matrix (13.20) is recovered, as can be expected. The additional power of (13.30) is exhibited below in two advanced applications.

§13.7.1. *Hinged Plane Beam Element

The 2-node prismatic plane beam element depicted in Figure 13.13 has a mechanical hinge at midspan ($\xi = 0$). The cross sections on both sides of the hinge can rotate respect to each other. Consequently both curvature κ

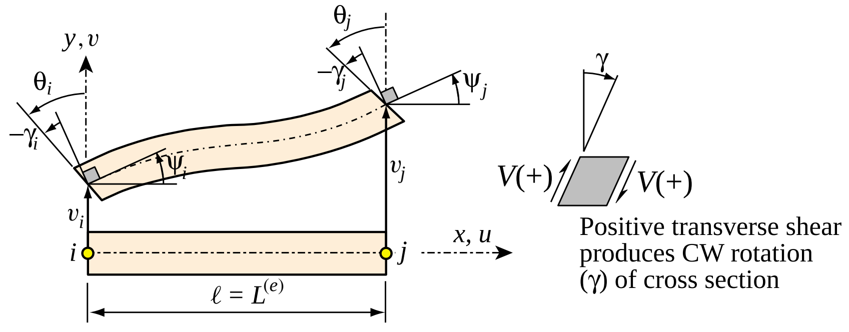

y, v
θ_i
−γ_i
ψ_i
θ_j
−γ_j
ψ_j
v_i
v_j
i
j
x, u
ℓ = L^(e)
γ
V(+)
V(+)
Positive transverse shear produces CW rotation (γ) of cross section

Figure 13.14. A 2-node Timoshenko beam element.

and the bending moment M must vanish there. But in a element described by a cubic interpolation of w the curvature must vary linearly in both ξ and x. Consequently the mean curvature, which is controlled by mode L_2, must be zero. This kinematic constraint can be enforced by setting the symmetric rigidity $R_s = 0$ whereas the antisymmetric rigidity is $R_a = EI$. Inserting these into (13.30) immediately gives

$$\mathbf{K}^{(e)} = \frac{EI}{\ell^3}\begin{bmatrix} 12 & 6\ell & -12 & 6 \\ 6\ell & 3\ell^2 & -6 & 3\ell^2 \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & 3\ell^2 & -6\ell & 3\ell^2 \end{bmatrix} \tag{13.31}$$

This matrix has rank 1, as it should be. It can be derived by more sophisticated methods, but the present technique is the most expedient one.

§13.7.2. *Timoshenko Plane Beam Element

As noted in 13.2, the Timoshenko model includes a first order correction for transverse shear. The kinematic assumption of classical beam theory is changed to “plane sections remain plane but not necessarily normal to the deformed neutral surface.” This is depicted in Figure 13.14 for a plane beam element. The cross section rotation is still called θ, positive counterclockwise, and the displacement field is

$$u(x, y) = u(x) - y\theta, \qquad v(x, y) = v(x) \tag{13.32}$$

But now

$$\theta = \frac{\partial v}{\partial x} - \gamma, \qquad \gamma = \frac{V}{GA_s}. \tag{13.33}$$

Here V is the transverse shear force, $\gamma = \gamma(x)$ the “shear rotation” averaged over the cross section, G the shear modulus and A_s the effective shear area.[4] The negative sign comes from the sign convention for V commonly used in Mechanics of Materials and employed here: a positive V produces a clockwise shear rotation. See Figure 13.14. For convenience define the dimensionless factor $\phi = 12EI/(GA_s\ell^2)$. This characterizes the “shear slenderness” of the beam element; if $\phi \to 0$ the Timoshenko model reduces to the Bernoulli-Euler one.

From equilibrium: $V = M' = EI\,v'''$ and $\gamma = V/(GA_s) = EIv'''/(GA_s) = v'''\ell^2/(12\phi)$. From the Legendre interpolation (13.24), $v''' = 120q_4/\ell^3$ and $\gamma = 10q_4/(12\phi\ell)$, which is constant over the element. The element internal energy is now

$$U^{(e)} = \tfrac{1}{2}\int_0^\ell (EI\kappa^2 + GA_s\gamma^2)\,dx. \tag{13.34}$$

[4] A concept defined in Mechanics of Materials; see e.g. Chapter 10 of Popov [13.10].

The remaining derivations involve lengthy algebra and are relegated to Exercise 13.11. The final result is that the element stiffness fits the form (13.30) in which $R_s = EI$ and $R_a = EI/(1+\phi)$. This gives

$$\mathbf{K}^{(e)} = \frac{EI}{\ell^3(1+\phi)} \begin{bmatrix} 12 & 6\ell & -12 & 6\ell \\ 6\ell & \ell^2(4+\phi) & -6\ell & \ell^2(2-\phi) \\ -12 & -6\ell & 12 & -6\ell \\ 6\ell & \ell^2(2-\phi) & -6\ell & \ell^2(4+\phi) \end{bmatrix} \tag{13.35}$$

If $\phi = 0$ this reduces to the Bernoulli-Euler beam stiffness, as can be expected.

§13.8. NOTES AND REFERENCES

A comprehensive source for matrices of plane and spatial beams is the book by Przemieniecki [13.11]. The derivation of stiffness matrices in Chapter 5 of that book is carried out using differential equations rather than energy methods. This was in fact the common practice before 1963 [13.9]. However results for prismatic elements are identical. The derivation using energy methods was popularized by Archer [13.2,13.3] and Martin [13.8].

The Legendre interpolation (13.24) to construct optimal mass-stiffness combinations for beam elements is further studied in [13.5,13.5] in the context of finite element templates. The diagonal covariance matrix **Q** given in (13.26) plays a key role in the customization of the mass matrix.

The hinged element stiffness (13.31) is rederived in the AFEM book using a mixed variational principle.

The equilibrium based form of the Timoshenko beam element (13.35) is derived in Section 5.6 of [13.11] using differential equations. This form is optimal in that it solves nodally-loaded discretizations exactly. This beam model belongs to the class of "C^0 elements", which have been extensively studied in the FEM litearture after 1968. The book of Hughes [13.7] provides a comprehensive treatment for beams and plates.

References

[13.1] M. Abramowitz and L. A. Stegun (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.

[13.2] J. S. Archer, Consistent mass matrix for distributed mass systems, *J. Str. Div. Proc. ASCE*, **89**, 161–178, 1963.

[13.3] J. S. Archer, Consistent mass matrix formulation for structural analysis using finite element techniques, *AIAA J.*, **3**, 1910–1918, 1965.

[13.4] C. A. Felippa, A survey of parametrized variational principles and applications to computational mechanics, *Comp. Meths. Appl. Mech. Engrg.*, **113**, 109–139, 1994.

[13.5] C. A. Felippa, Customizing high performance elements by Fourier methods, *Trends in Computational Mechanics*, ed. by W. A. Wall, K.-U. Bleitzinger and K. Schweizerhof, CIMNE, Barcelona, Spain, 283-296, 2001.

[13.6] C. A. Felippa, Customizing the mass and geometric stiffness of plane thin beam elements by Fourier methods, *Engrg. Comput.*, **18**, 286–303, 2001.

[13.7] T. J. R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis*, Prentice Hall, Englewood Cliffs, N. J., 1987.

[13.8] H. C. Martin, 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.

[13.9] E. C. Pestel and F. A. Leckie, *Matrix Methods in Elastomechanics*, McGraw-Hill, New York, 1963.

[13.10] E. P. Popov, *Engineering Mechanics of Solids*, Prentice Hall, Englewood Cliffs, N. J., 2nd ed., 1991.

[13.11] J. S. Przemieniecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968; Dover edition 1986.

Homework Exercises for Chapter 13
Variational Formulation of Plane Beam Element

EXERCISE 13.1

[A/C:20] Use (13.17) to derive the element stiffness matrix $\mathbf{K}^{(e)}$ of a Hermitian beam element of variable bending rigidity given by the inertia law

$$I(x) = I_i(1-\frac{x}{\ell}) + I_j\frac{x}{\ell} = I_i\,\tfrac{1}{2}(1-\xi) + I_j\,\tfrac{1}{2}(1+\xi). \tag{E13.1}$$

Use of *Mathematica* or similar CAS tool is recommended since the integrals are time consuming. *Mathematica* hint: write

```
EI = EIi*(1-ξ)/2 + EIj*(1+ξ)/2;                (E13.2)
```

and keep `EI` inside the argument of `Integrate`. Check whether you get back (13.20) if `EI=EIi=EIj`. If you use *Mathematica*, this check can be simply done after you got and printed the tapered beam `Ke`, by writing `ClearAll[EI]; Ke=Simplify[ Ke/.{EIi->EI,EIj->EI}];` and printing this matrix.[5]

EXERCISE 13.2

[A/C:20] Use (13.18) to derive the consistent node force vector $\mathbf{f}^{(e)}$ for a Hermitian beam element under linearly varying transverse load q defined by

$$q(x) = q_i(1-\frac{x}{\ell}) + q_j\frac{x}{\ell} = q_i\,\tfrac{1}{2}(1-\xi) + q_j\,\tfrac{1}{2}(1+\xi). \tag{E13.3}$$

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 = qi*(1-ξ)/2 + qj*(1+ξ)/2;                (E13.4)
```

and keep `q` inside the argument of `Integrate`. Check whether you get back (13.21) if $q_i = q_j = q$ (See previous Exercise for *Mathematica* procedure).

EXERCISE 13.3

[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 $0 \le a \le \ell$. 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 $0 \le a \le \ell$.

EXERCISE 13.4

[A:25] Derive the consistent node force vector $\mathbf{f}^{(e)}$ of a Hermitian beam element subject to a uniformly distributed z-moment m per unit length. 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

$$\mathbf{f}^{(e)} = \int_0^\ell \frac{\partial \mathbf{N}^T}{\partial x} m\,dx = \int_{-1}^{1} \frac{\partial \mathbf{N}^T}{\partial \xi}\frac{2}{\ell} m\,\tfrac{1}{2}\ell\,d\xi = \int_{-1}^{1} \mathbf{N}_\xi^T m\,d\xi, \tag{E13.5}$$

where $\mathbf{N}_\xi^T$ denote the column vectors of beam shape function derivatives with respect to ξ. Can you see a shortcut that avoids the integral for constant m?

[5] `ClearAll[EI]` discards the previous definition (E13.2) of `EI`; the same effect can be achieved by writing `EI=.` (dot).

EXERCISE 13.5

[A:20] Obtain the consistent node force vector $\mathbf{f}^{(e)}$ of a Hermitian beam element subject to a concentrated moment C applied at $x = a$. Use the expression (E13.5) in which $m(x) = C\,\delta(a)$, where $\delta(a)$ denotes the Dirac's delta function at $x = a$.

EXERCISE 13.6

[A/C:25] Consider the one-dimensional Gauss integration rules.[6]

$$\text{One point :}\qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq 2\,f(0) \tag{E13.6}$$

$$\text{Two points:}\qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq f(-1/\sqrt{3}) + f(1/\sqrt{3}) \tag{E13.7}$$

$$\text{Three points:}\qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq \frac{5}{9}\,f(-\sqrt{3/5}) + \frac{8}{9}\,f(0) + \frac{5}{9}\,f(\sqrt{3/5}) \tag{E13.8}$$

Try each rule on the monomial integrals

$$\int_{-1}^{1} d\xi,\qquad \int_{-1}^{1} \xi\,d\xi,\qquad \int_{-1}^{1} \xi^2\,d\xi,\quad \ldots \tag{E13.9}$$

until the rule fails. In this way verify that the rules (E13.6), (E13.7) and (E13.8) 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 13.7

[A/C:25] Repeat the derivation of Exercise 13.1 using the two-point Gauss rule (E13.7) 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 (E13.8), 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 13.8

[A/C:25] As above but for Exercise 13.2.

EXERCISE 13.9

[A/C:25=15+10] Consider a cantilever beam of constant rigidity EI and length L, which is modeled with one element. The beam is end loaded by a transverse force P.

(a) Find the displacement v_j and the rotation θ_j in terms of P, E, I and L Compare with the analytical values $PL^3/(3EI)$ and $PL^2/(2EI)$.

(b) Why does the finite element model provides the exact answer with one element?

[6] Gauss integration is studied further in Chapter 17.

EXERCISE 13.10

[A/C:30] Derive the classical beam stiffness matrix using the method of differential equations. To do this integrate the homegeneous differential equation $EIv'''' = 0$ four times over a cantilever beam clamped at node i 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 i and the two force BCs at node j. 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 i translate and rotate.

EXERCISE 13.11

[A/C:35] Carry out the complete derivation of the Timoshenko beam element stiffness (13.35).

14 The Plane Stress Problem

TABLE OF CONTENTS

§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 H, continuum finite elements were invented in the aircraft industry[1] to solve this kind of problem when it arose in the design and analysis of delta wing panels.

The problem is presented here within the framework of the linear theory of elasticity.

§14.1.1. Plate in Plane Stress

In structural mechanics, a flat thin sheet of material is called a *plate*.[2] The distance between the plate faces is called the *thickness* and denoted by h. The *midplane* lies halfway between the two faces. The direction normal to the midplane is called the *transverse* direction. Directions parallel to the midplane are called *in-plane* directions. The global axis z will be oriented along the transverse direction. Thus the midplane equation is $z = 0$. Axis x and y are placed in the midplane, forming a right-handed Rectangular Cartesian Coordinate (RCC) system. See Figure 14.1.

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. All displacements, strains and stresses can be taken to be uniform through the thickness.
4. All stress components in the z direction are zero or negligible.

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 inplane 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 though the thickness. Such plates are called *transversely homogeneous* or *monocoque* plates.

The last assumption excludes wall constructions of importance in aerospace, in particular composite and honeycomb plates. The development of 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 lead 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 is precluded. Although the construction of finite element models for those states has many common points with plane stress, we shall not

[1] At Boeing over the period 1952-53.

[2] If it is relatively thick, as in concrete pavements, or cheese slices, the term *slab* is also used but not for plane stress conditions.

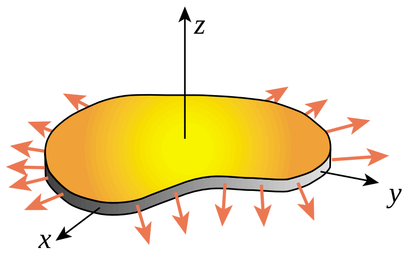


Figure 14.1. A plate structure in plane stress.

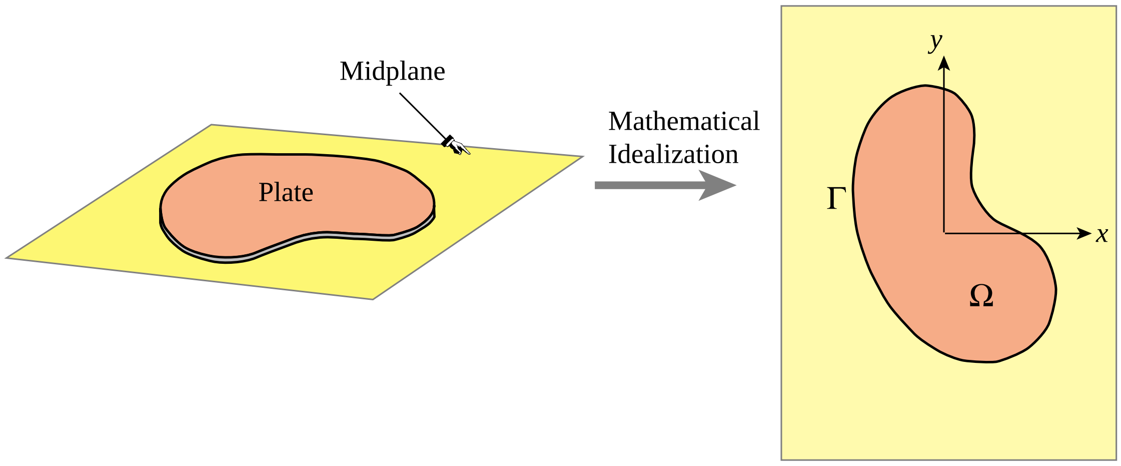


Figure 14.2. Mathematical model of plate in plane stress.

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 more advanced courses.

§14.1.2. Mathematical Model

The mathematical model of the plate in plane stress is a two-dimensional boundary value problem (BVP). This problem is posed over a plane domain Ω with a boundary Γ, as illustrated in Figure 14.2.

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 inplane stresses through the thickness. See Figure 14.3.

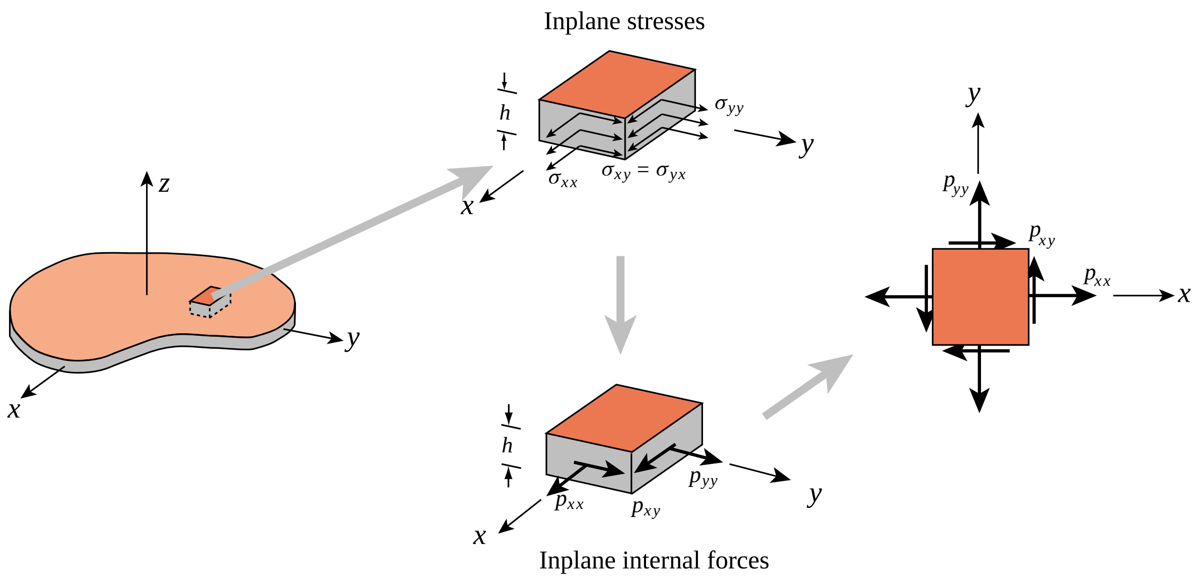


Figure 14.3. Internal stresses and plate forces.

§14.2. PLANE STRESS PROBLEM DESCRIPTION

§14.2.1. Problem Data

The given data includes:

Domain geometry. This is defined by the boundary Γ illustrated in Figure 14.2.

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.

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 these 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.

Displacement Boundary Conditions. These specify how the plate is supported. Points on the plate boundary may be fixed, allowed to move in one direction, or subject to multipoint constraints. In addition symmetry and antisymmetry lines may be identified as discussed in Chapter 8.

If no displacement boundary conditions are imposed, the plate structure is said to be *free-free*.

§14.2.2. Problem Unknowns

The three 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*. Consequently 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:

$$\mathbf{u}(x, y) = \begin{bmatrix} u_x(x, y) \\ u_y(x, y) \end{bmatrix} \tag{14.1}$$

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 conventionally arranged to form a 3-component "strain vector"

$$\mathbf{e}(x, y) = \begin{bmatrix} e_{xx}(x, y) \\ e_{yy}(x, y) \\ 2e_{xy}(x, y) \end{bmatrix} \tag{14.2}$$

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 effects. This strain does not enter the governing equations as unknown because the associated stress σ_{zz} is zero, which 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 conventionally arranged to form a 3-component "stress vector"

$$\boldsymbol{\sigma}(x, y) = \begin{bmatrix} \sigma_{xx}(x, y) \\ \sigma_{yy}(x, y) \\ \sigma_{xy}(x, y) \end{bmatrix} \tag{14.3}$$

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,

$$p_{xx} = \sigma_{xx}h, \quad p_{yy} = \sigma_{yy}h, \quad p_{xy} = \sigma_{xy}h. \tag{14.4}$$

These p's also form a tensor. They are often called *membrane forces* in the literature. See Figure 14.3.

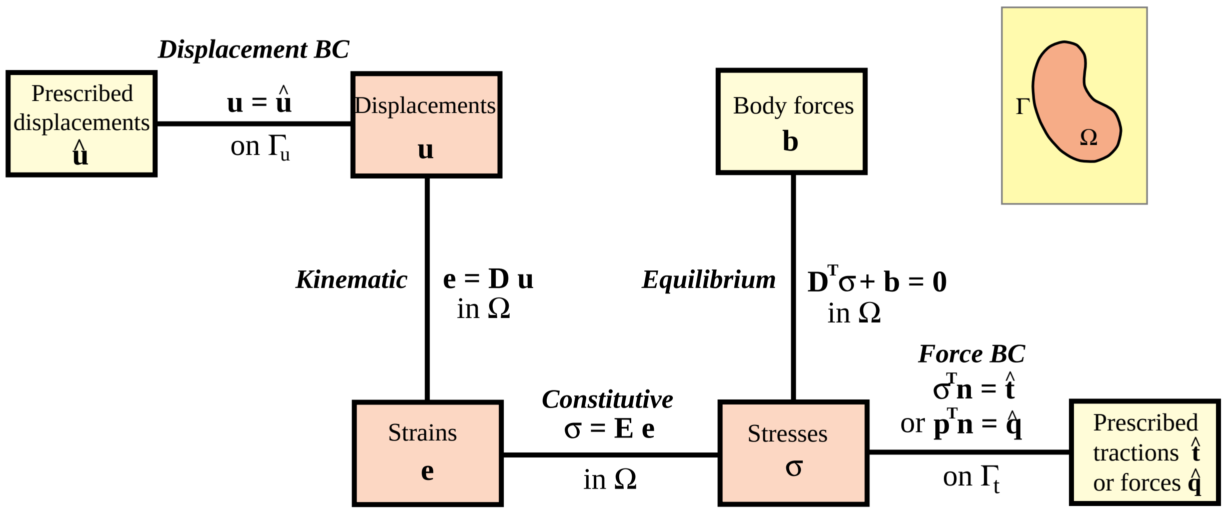


Figure 14.4. The Strong Form of the plane stress equations of linear elastostatics displayed as a Tonti diagram. Yellow boxes identify prescribed fields whereas orange boxes denote unknown fields. The distinction between Strong and Weak Forms is explained in §14.3.3.

§14.3. LINEAR ELASTICITY 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 14.4.

§14.3.1. Governing Equations

The three internal fields: displacements, strains and stresses (14.2)-(14.4) are connected by three field equations: kinematic, constitutive and internal-equilibrium equations. If initial strain effects are ignored, these equations read

$$\begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix},$$

$$\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \tag{14.5}$$

$$\begin{bmatrix} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

In these equations, b_x and b_y are the components of the interior body force **b**, **E** is the 3×3 stress-strain matrix of plane stress elastic moduli, **D** is the 3×2 symmetric-gradient operator and its transpose the 2×3 tensor-divergence operator. The dependence on (x, y) has been omitted to reduce clutter.

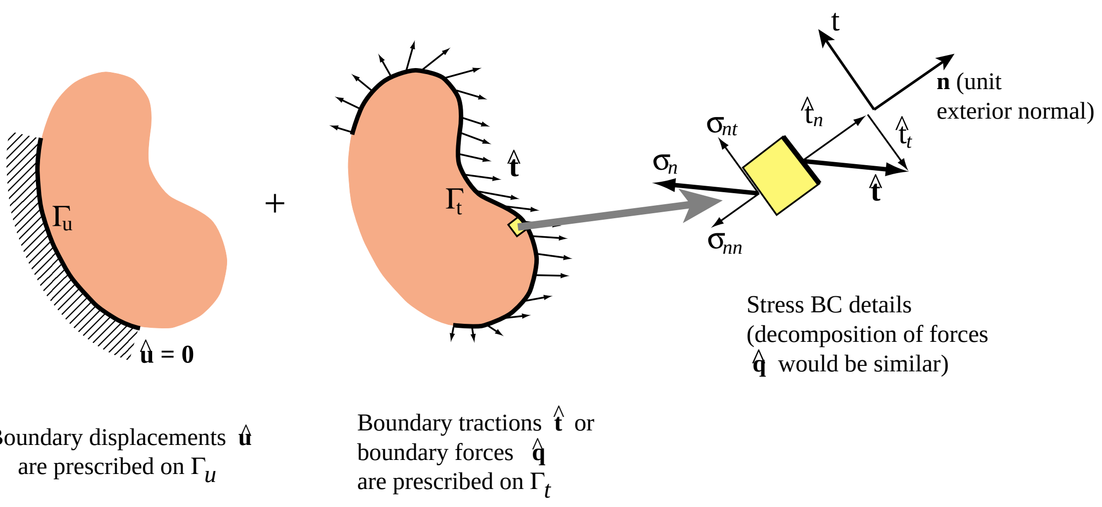

Γ_u
$\hat{\mathbf{u}} = \mathbf{0}$
+
Γ_t
$\hat{\mathbf{t}}$
t
$\mathbf{n}$ (unit exterior normal)
$\hat{t}_n$
$\hat{t}_t$
σ_{nt}
σ_n
σ_{nn}
$\hat{\mathbf{t}}$
Stress BC details (decomposition of forces $\hat{\mathbf{q}}$ would be similar)
Boundary displacements $\hat{\mathbf{u}}$ are prescribed on Γ_u
Boundary tractions $\hat{\mathbf{t}}$ or boundary forces $\hat{\mathbf{q}}$ are prescribed on Γ_t

Figure 14.5. Displacement and force (stress, traction) boundary conditions for the plane stress problem.

The compact matrix version of (14.5) is

$$\boxed{\mathbf{e} = \mathbf{D}\mathbf{u}, \qquad \boldsymbol{\sigma} = \mathbf{E}\mathbf{e}, \qquad \mathbf{D}^T\boldsymbol{\sigma} + \mathbf{b} = \mathbf{0},} \tag{14.6}$$

in which $\mathbf{E} = \mathbf{E}^T$.

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 (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 14.5.

Displacement boundary conditions are prescribed on Γ_u in the form

$$\boxed{\mathbf{u} = \hat{\mathbf{u}}.} \tag{14.7}$$

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 14.5.

Force boundary conditions (also called stress BCs and traction BCs in the literature) are specified on Γ_t. They take the form

$$\boxed{\boldsymbol{\sigma}_n = \hat{\mathbf{t}}.} \tag{14.8}$$

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 14.5.

An alternative form of (14.8) uses the internal plate forces:

$$\boxed{\mathbf{p}_n = \hat{\mathbf{q}}.} \tag{14.9}$$

Here $\mathbf{p}_n = \boldsymbol{\sigma}_n h$ 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 nonhomogeneous.

The components of $\boldsymbol{\sigma}_n$ in Cartesian coordinates follow from Cauchy's stress transformation formula

$$\boldsymbol{\sigma}_n = \begin{bmatrix} \sigma_{xx} n_x + \sigma_{xy} n_y \\ \sigma_{xy} n_x + \sigma_{yy} n_y \end{bmatrix} = \begin{bmatrix} n_x & 0 & n_y \\ 0 & n_y & n_x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{14.10}$$

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 $\hat{t}_y = \sigma_{ny}$.

It is sometimes convenient to write the condition (14.8) in terms of normal n and tangential t directions:

$$\sigma_{nn} = \hat{t}_n, \qquad \sigma_{nt} = \hat{t}_t \tag{14.11}$$

in which $\sigma_{nn} = \sigma_{nx} n_x + \sigma_{ny} n_y$ and $\sigma_{nt} = -\sigma_{nx} n_y + \sigma_{ny} n_x$.

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. To cover these situations one needs either a generalization of the 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.

§14.3.3. Weak Forms versus Strong Form

We introduce now some further terminology from variational calculus. The Tonti diagram of Figure 14.4 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. 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 14.6. The internal equilibrium equations and stress BC become weak links, which are indicated 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.

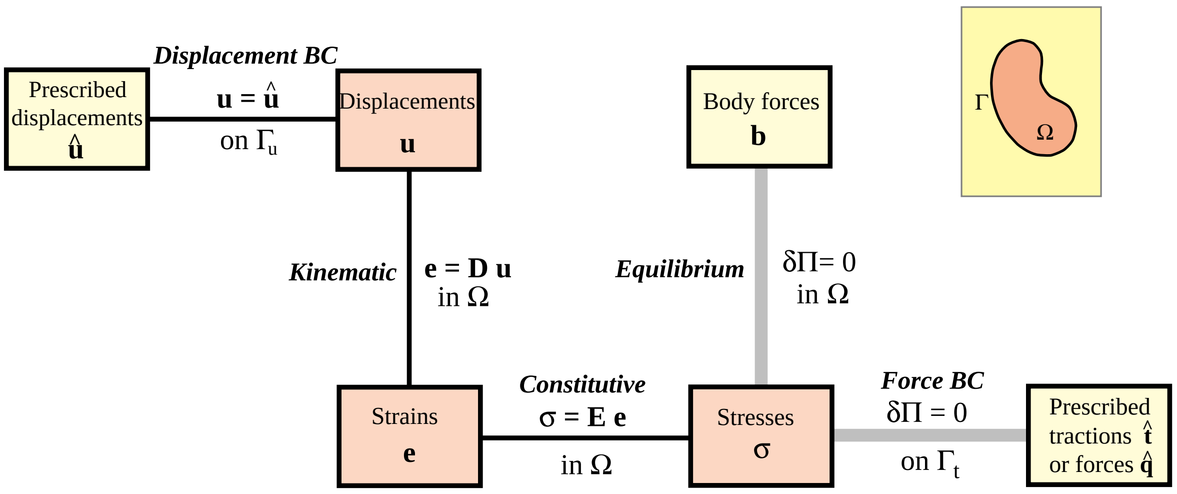

Displacement BC
Prescribed displacements $\hat{\mathbf{u}}$
$\mathbf{u} = \hat{\mathbf{u}}$ on Γ_u
Displacements $\mathbf{u}$
Body forces $\mathbf{b}$
Γ
Ω
Kinematic
$\mathbf{e} = \mathbf{D}\,\mathbf{u}$ in Ω
Equilibrium
$\delta\Pi = 0$ in Ω
Constitutive
$\sigma = \mathbf{E}\,\mathbf{e}$ in Ω
Strains $\mathbf{e}$
Stresses σ
Force BC
$\delta\Pi = 0$ on Γ_t
Prescribed tractions $\hat{\mathbf{t}}$ or forces $\hat{\mathbf{q}}$

Figure 14.6. The TPE-based Weak Form of the plane stress equations of linear elastostatics. Weak links are marked with grey lines.

§14.3.4. Total Potential Energy

As usual the Total Potential Energy functional for the plane stress problem is given by

$$\boxed{\Pi = U - W.} \tag{14.12}$$

The internal energy is the elastic strain energy:

$$\boxed{U = \tfrac{1}{2}\int_\Omega h\,\boldsymbol{\sigma}^T\mathbf{e}\,d\Omega = \tfrac{1}{2}\int_\Omega h\,\mathbf{e}^T\mathbf{E}\mathbf{e}\,d\Omega.} \tag{14.13}$$

The derivation details are relegated to Exercise E14.5. The external energy is the sum of contributions from known interior and boundary forces:

$$\boxed{W = \int_\Omega h\,\mathbf{u}^T\mathbf{b}\,d\Omega + \int_{\Gamma_t} h\,\mathbf{u}^T\hat{\mathbf{t}}\,d\Gamma.} \tag{14.14}$$

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 are collected here and expressed in matrix form. The domain of Figure 14.7(a) is discretized by a finite element mesh as illustrated in Figure 14.7(b). From this mesh we extract a generic element labeled (e) with $n \geq 3$ node points. In the subsequent derivation the number n is kept *arbitrary*. Therefore, the formulation is applicable to arbitrary two-dimensional elements, for example those sketched in Figure 14.8.

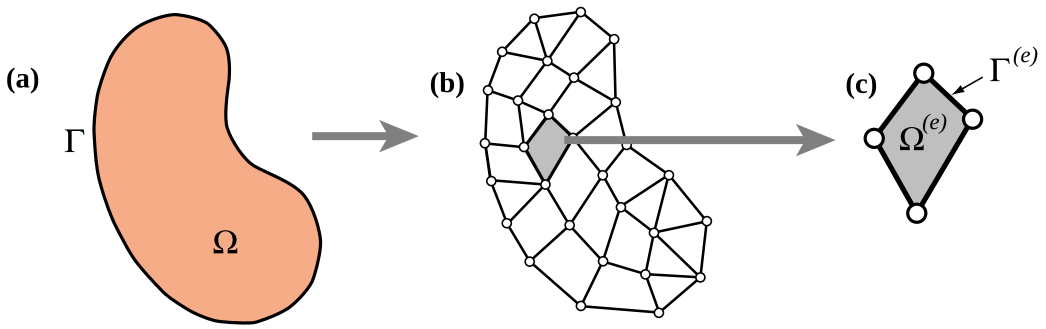


Figure 14.7. Finite element discretization and extraction of generic element.

Departing from previous practice in 1D elements, the element node points will be labelled 1 through n. These are called *local node numbers*. The element domain and boundary are denoted by $\Omega^{(e)}$ and $\Gamma^{(e)}$, respectively. The element has $2n$ degrees of freedom. These are collected in the element node displacement vector

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad \ldots \quad u_{xn} \quad u_{yn} \,]^T \,. \tag{14.15}$$

§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

$$u_x(x, y) = \sum_{i=1}^{n} N_i^{(e)}(x, y)\, u_{xi}, \qquad u_y(x, y) = \sum_{i=1}^{n} N_i^{(e)}(x, y)\, u_{yi}, \tag{14.16}$$

where $N_i^{(e)}(x, y)$ are the element shape functions. In matrix form:

$$\boxed{\mathbf{u}(x, y) = \begin{bmatrix} u_x(x, y) \\ u_y(x, y) \end{bmatrix} = \begin{bmatrix} N_1^{(e)} & 0 & N_2^{(e)} & 0 & \ldots & N_n^{(e)} & 0 \\ 0 & N_1^{(e)} & 0 & N_2^{(e)} & \ldots & 0 & N_n^{(e)} \end{bmatrix} \mathbf{u}^{(e)} = \mathbf{N}^{(e)}\mathbf{u}^{(e)}.} \tag{14.17}$$

The minimum conditions on $N_i^{(e)}(x, y)$ is that it must take the value one at the i^{th} node and zero at all others, so 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:

$$\mathbf{e}(x, y) = \begin{bmatrix} \frac{\partial N_1^{(e)}}{\partial x} & 0 & \frac{\partial N_2^{(e)}}{\partial x} & 0 & \ldots & \frac{\partial N_n^{(e)}}{\partial x} & 0 \\ 0 & \frac{\partial N_1^{(e)}}{\partial y} & 0 & \frac{\partial N_2^{(e)}}{\partial y} & \ldots & 0 & \frac{\partial N_n^{(e)}}{\partial y} \\ \frac{\partial N_1^{(e)}}{\partial y} & \frac{\partial N_1^{(e)}}{\partial x} & \frac{\partial N_2^{(e)}}{\partial y} & \frac{\partial N_2^{(e)}}{\partial x} & \ldots & \frac{\partial N_n^{(e)}}{\partial y} & \frac{\partial N_n^{(e)}}{\partial x} \end{bmatrix} \mathbf{u}^{(e)} = \mathbf{B}\,\mathbf{u}^{(e)}. \tag{14.18}$$

[3] This is the so called *element isotropy* condition, which is studied and justified in advanced FEM courses.

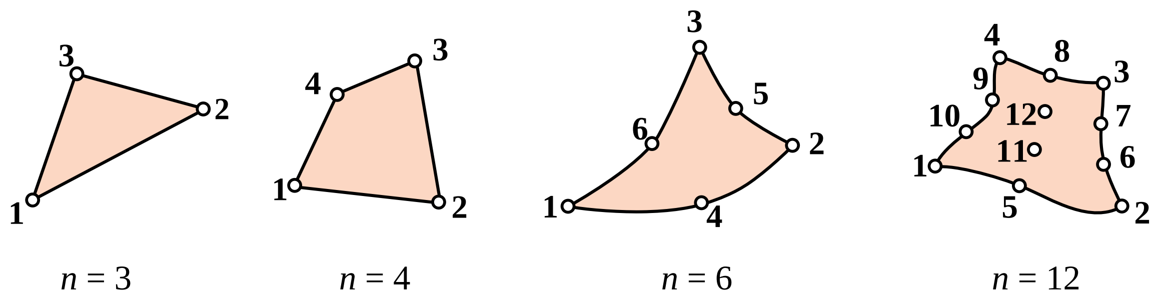


Figure 14.8. Example two-dimensional finite elements, characterized by their number of nodes n.

The strain-displacement matrix $\mathbf{B} = \mathbf{D}\mathbf{N}^{(e)}$ is $3 \times 2n$. The superscript (e) is omitted from it to reduce clutter. 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:

$$\boxed{\delta\Pi^{(e)} = \delta U^{(e)} - \delta W^{(e)} = 0.} \tag{14.19}$$

where

$$\boxed{U^{(e)} = \tfrac{1}{2}\int_{\Omega^{(e)}} h\,\boldsymbol{\sigma}^T\mathbf{e}\,d\Omega^{(e)} = \tfrac{1}{2}\int_{\Omega^{(e)}} h\,\mathbf{e}^T\mathbf{E}\mathbf{e}\,d\Omega^{(e)}} \tag{14.20}$$

and

$$\boxed{W^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{u}^T\mathbf{b}\,d\Omega^{(e)} + \int_{\Gamma^{(e)}} h\,\mathbf{u}^T\,\hat{\mathbf{t}}\,d\Gamma^{(e)}} \tag{14.21}$$

Note that in (14.21) $\Gamma_t^{(e)}$ has been taken equal to the complete boundary $\Gamma^{(e)}$ 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 $\Pi^{(e)}$ yields the quadratic form in the nodal displacements

$$\Pi^{(e)} = \tfrac{1}{2}\mathbf{u}^{(e)\,T}\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{u}^{(e)\,T}\mathbf{f}^{(e)}, \tag{14.22}$$

where the element stiffness matrix is

$$\boxed{\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{B}^T\mathbf{E}\mathbf{B}\,d\Omega^{(e)},} \tag{14.23}$$

and the consistent element nodal force vector is

$$\boxed{\mathbf{f}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{N}^T\mathbf{b}\,d\Omega^{(e)} + \int_{\Gamma^{(e)}} h\,\mathbf{N}^T\hat{\mathbf{t}}\,d\Gamma^{(e)}.} \tag{14.24}$$

In the second integral the matrix **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.

Homework Exercises for Chapter 14

The Plane Stress Problem

EXERCISE 14.1

[A:25] 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

$$\begin{aligned}
\text{plane stress:}\quad & \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \\
\text{plane strain:}\quad & \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}.
\end{aligned} \tag{E14.1}$$

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 and suppressing the stars. Find the expression of E^* and ν^* in terms of E and ν. This device permits "reusing" a plane stress FEM program to do plane strain, as long as the material is isotropic.

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

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = G = \frac{E}{2(1+\nu)}. \tag{E14.2}$$

Conversely

$$E = \frac{\mu(3\lambda + 2\mu)}{\lambda + \mu}, \quad \nu = \frac{\lambda}{2(\lambda + \mu)}. \tag{E14.3}$$

Substitute (E14.3) into (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

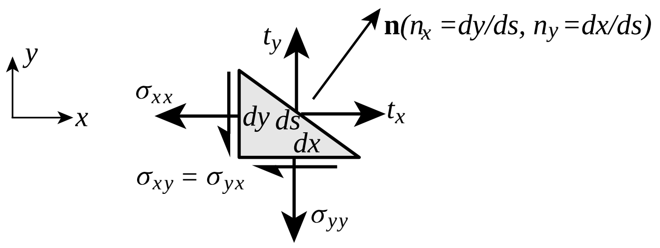
$$\mathbf{E} = \mathbf{E}_\mu + \mathbf{E}_\lambda \tag{E14.4}$$

in which $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ contain only μ and λ, respectively, with $\mathbf{E}_\mu$ 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 $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ also in terms of E and ν.

For the plane stress case perform a similar splitting in which where $\mathbf{E}_\lambda$ contains only $\bar{\lambda} = 2\lambda\mu/(\lambda + 2\mu)$ with $E_{\lambda 33} = 0$, and $\mathbf{E}_\mu$ is a diagonal matrix function of μ and $\bar{\lambda}$.[5] Express $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ 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 meaning is lost, however, for plane stress.

[5] For the physical significance of $\bar{\lambda}$ see I. Sokolnikoff, *The Mathematical Theory of Elasticity*, McGraw-Hill, 2nd ed., 1956, p. 254ff.



y
x
t_y
$\mathbf{n}(n_x = dy/ds,\ n_y = dx/ds)$
σ_{xx}
t_x
dy
ds
dx
$\sigma_{xy} = \sigma_{yx}$
σ_{yy}

Figure E14.1. Geometry for deriving (14.10).

EXERCISE 14.3

[A:20] Derive the Cauchy stress-to-traction equations (14.10) using force equilibrium along x and y and the geometric relations shown in Figure E14.1. Hint: $t_x\, ds = \sigma_{xx}\, dy + \sigma_{xy}\, dx$, etc.

EXERCISE 14.4

[A:15] Include thermoelastic effects in the plane stress field equations, assuming a thermally isotropic material with coefficient of linear expansion α.

EXERCISE 14.5

[A:25=5+5+15] A plate is in linearly elastic plane stress. It is shown in courses in elasticity that the internal strain energy density stored per unit volume is

$$\mathcal{U} = \tfrac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + \sigma_{xy}e_{xy} + \sigma_{yx}e_{yx}) = \tfrac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + 2\sigma_{xy}e_{xy}). \tag{E14.5}$$

(a) Show that (E14.5) can be written in terms of strains only as

$$\mathcal{U} = \tfrac{1}{2}\mathbf{e}^T \mathbf{E}\, \mathbf{e}, \tag{E14.6}$$

and hence justify (14.13).

(b) Show that (E14.5) can be written in terms of stresses only as

$$\mathcal{U} = \tfrac{1}{2}\boldsymbol{\sigma}^T \mathbf{C}\, \boldsymbol{\sigma}, \tag{E14.7}$$

where $\mathbf{C} = \mathbf{E}^{-1}$ is the elastic compliance (strain-stress) matrix.

(c) Suppose you want to write (E14.5) 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

$$\mathcal{U} = \tfrac{1}{2}\begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}^T \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ -A_{13} & -A_{23} & A_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{E14.8}$$

and explain how the entries A_{ij} can be calculated[6] in terms of the elastic moduli E_{ij}.

[6] The process of computing **A** is an instance of “partial inversion” of matrix **E**. See Remark 11.3.

15 The Linear Plane Stress Triangle

TABLE OF CONTENTS

§15.1. INTRODUCTION

This Chapter presents the element equations of a three-node triangle with assumed linear displacements for the plane stress problem formulated in Chapter 14. This element is called the *linear triangle*. This particular element is distinguished in several respects:

(1) It belongs to both the isoparametric and superparametric element families, which are covered in the next Chapter.

(2) It allows closed form derivations for its stiffness and consistent forces without the need for numerical integration.

(3) It cannot be improved by the addition of internal degrees of freedom.

In addition the linear triangle has historical importance.[1] Although it 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. One reason is that triangular meshes are easily generated over arbitrary domains using techniques such as Delaunay triangulations.

§15.1.1. Parametric Representation of Functions

The concept of *parametric representation* of functions is crucial in modern FEM presentations. Together with numerical integration, it has become a key tool for the systematic development of elements in two and three space dimensions. Without these two tools the element developer would become lost in an algebraic maze as element geometric shapes get more complicated.

The essentials of the idea of parametric representation can be illustrated through a simple example. Consider the following alternative representations of the unit-circle function, $x^2 + y^2 = 1$:

$$y = \sqrt{1 - x^2} \tag{15.1}$$

$$x = \cos\theta, \quad y = \sin\theta \tag{15.2}$$

The direct representation (15.1) 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 representation (15.2) is parametric: both x 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 many 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. TRIANGLE GEOMETRY AND COORDINATE SYSTEMS

The geometry of the 3-node triangle shown in Figure 15.1(a) is specified by the location of its three corner nodes on the $\{x, y\}$ plane. The nodes are labelled 1, 2, 3 while traversing the sides in *counterclockwise* fashion. The location of the corners is defined by their Cartesian coordinates: $\{x_i, y_i\}$ for $i = 1, 2, 3$.

[1] This was one of the two continuum elements presented by Martin, Turner, Clough and Topp in their landmark 1956 paper. Its publication is widely regarded as the start of the present FEM. See Appendix H.

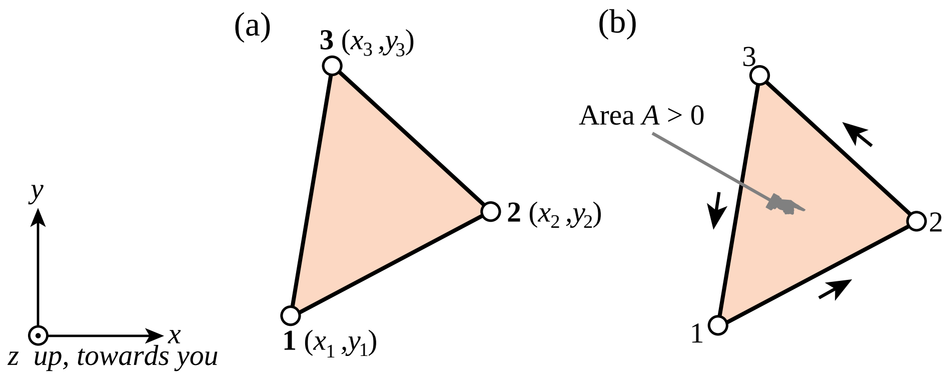


Figure 15.1. The three-node, linear-displacement plane stress triangular element: (a) geometry; (b) area and positive boundary traversal.

The element has six degrees of freedom, defined by the six nodal 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 area of the triangle is denoted by A and is given by

$$2A = \det \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} = (x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) + (x_1 y_2 - x_2 y_1). \tag{15.3}$$

The area given by (15.3) is a *signed* quantity. It is positive if the corners are numbered in cyclic counterclockwise order as shown in Figure 15.1(b). This convention is followed in the sequel.

§15.2.1. Triangular Coordinates

Points of the triangle may also be located in terms of a *parametric* coordinate system:

$$\zeta_1,\ \zeta_2,\ \zeta_3. \tag{15.4}$$

In the literature these three parameters receive an astonishing number of names, as the list given 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

$$\zeta_i = constant \tag{15.5}$$

represent a set of straight lines parallel to the side opposite to the i^{th} corner, as depicted in Figure 15.2. The equation of sides 1–2, 2–3 and 3–1 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:

$$\zeta_1 + \zeta_2 + \zeta_3 = 1. \tag{15.6}$$

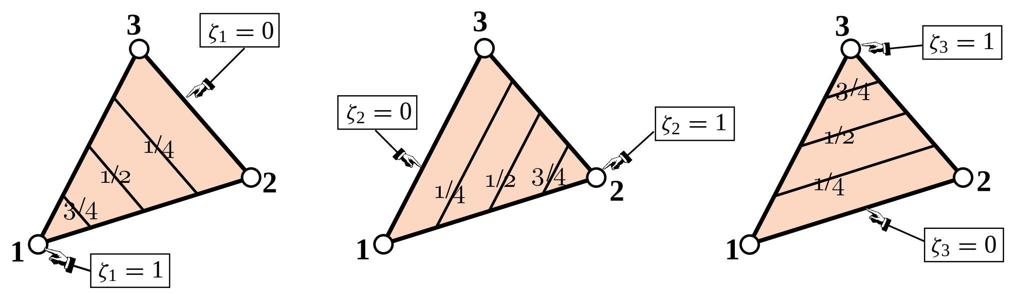


Figure 15.2. Triangular coordinates.

REMARK 15.1

In older (pre-1970) FEM publications triangular coordinates are often called *area coordinates*. This name comes from the following interpretation: $\zeta_i = A_{jk}/A$, where A_{jk} is the area subtended by the triangle formed by the point P and corners j and k, in which j and k are cyclic permutations of i. Historically this was the way the coordinates were defined in 1960s papers. Unfortunately the interpretation does not carry over to general isoparametric triangles with curved sides and thus it is not used here.

Table 15.1 Names of element parametric coordinates in the FEM literature

Name	*Applicable to*
natural coordinates	all elements
isoparametric coordinates	isoparametric elements
shape function coordinates	isoparametric elements
barycentric coordinates	triangles, tetrahedra
Möbius coordinates	triangles
triangular coordinates	all triangles
area coordinates	straight-sided triangles

§15.2.2. Linear Interpolation

Consider a function $f(x, y)$ that varies *linearly* over the triangle domain. In terms of Cartesian coordinates it may be expressed as

$$f(x, y) = a_0 + a_1 x + a_2 y, \tag{15.7}$$

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:

$$f_1, \quad f_2, \quad f_3 \tag{15.8}$$

The expression in triangular coordinates makes direct use of those three values:

$$f(\zeta_1, \zeta_2, \zeta_3) = f_1\zeta_1 + f_2\zeta_2 + f_3\zeta_3 = [\, f_1 \quad f_2 \quad f_3 \,] \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = [\, \zeta_1 \quad \zeta_2 \quad \zeta_3 \,] \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix}. \tag{15.9}$$

Expression (15.9) is called a *linear interpolant* for f.

§15.2.3. Coordinate Transformations

Quantities that are closely linked with the element geometry are naturally expressed in triangular coordinates. On the other hand, quantities such as displacements, strains and stresses are often expressed in the Cartesian system x, y. We therefore need transformation equations through which we can pass from one coordinate system to the other.

Cartesian and triangular coordinates are linked by the relation

$$\boxed{\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}.} \tag{15.10}$$

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 apply the interpolant formula (15.9) to the Cartesian coordinates: $x = x_1\zeta_1 + x_2\zeta_2 + x_3\zeta_3$ and $y = y_1\zeta_1 + y_2\zeta_2 + y_3\zeta_3$.

Inversion of (15.10) yields

$$\begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} x_2y_3 - x_3y_2 & y_2 - y_3 & x_3 - x_2 \\ x_3y_1 - x_1y_3 & y_3 - y_1 & x_1 - x_3 \\ x_1y_2 - x_2y_1 & y_1 - y_2 & x_2 - x_1 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} 2A_{23} & y_{23} & x_{32} \\ 2A_{31} & y_{31} & x_{13} \\ 2A_{12} & y_{12} & x_{21} \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \end{bmatrix}. \tag{15.11}$$

Here $x_{jk} = x_j - x_k$, $y_{jk} = y_j - y_k$, A is the triangle area given by (15.3) 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, $A_{23} = A_{31} = A_{12} = A/3$.

§15.2.4. Partial Derivatives

From equations (15.10) and (15.11) we immediately obtain the following relations between partial derivatives:

$$\frac{\partial x}{\partial \zeta_i} = x_i, \qquad \frac{\partial y}{\partial \zeta_i} = y_i, \tag{15.12}$$

$$2A\frac{\partial \zeta_i}{\partial x} = y_{jk}, \qquad 2A\frac{\partial \zeta_i}{\partial y} = x_{kj}. \tag{15.13}$$

In (15.13) j and k denote the *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 (15.13) and application of the chain rule:

$$\boxed{\begin{aligned} \frac{\partial f}{\partial x} &= \frac{1}{2A}\left(\frac{\partial f}{\partial \zeta_1}y_{23} + \frac{\partial f}{\partial \zeta_2}y_{31} + \frac{\partial f}{\partial \zeta_3}y_{12}\right) \\ \frac{\partial f}{\partial y} &= \frac{1}{2A}\left(\frac{\partial f}{\partial \zeta_1}x_{32} + \frac{\partial f}{\partial \zeta_2}x_{13} + \frac{\partial f}{\partial \zeta_3}x_{21}\right) \end{aligned}} \tag{15.14}$$

which in matrix form is

$$\begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} y_{23} & y_{31} & y_{12} \\ x_{32} & x_{13} & x_{21} \end{bmatrix} \begin{bmatrix} \dfrac{\partial f}{\partial \zeta_1} \\ \dfrac{\partial f}{\partial \zeta_2} \\ \dfrac{\partial f}{\partial \zeta_3} \end{bmatrix}. \tag{15.15}$$

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 linear triangle.

§15.3. ELEMENT DERIVATION

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*. The shape functions are simply the triangular coordinates. That is, $N_i^{(e)} = \zeta_i$ for $i = 1, 2, 3$.

§15.3.1. Displacement Interpolation

For the plane stress problem we will select the linear interpolation (15.9) for the displacement components u_x and u_y at an arbitrary point $P(\zeta_1, \zeta_2, \zeta_3)$:

$$\boxed{\begin{aligned} u_x &= u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3, \\ u_y &= u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3, \end{aligned}} \tag{15.16}$$

as illustrated in Figure 15.3.

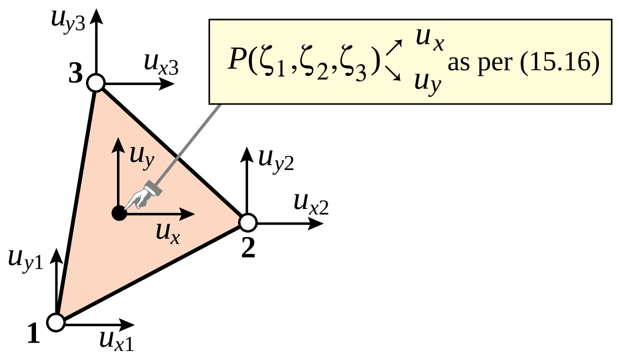


Figure 15.3. Displacement interpolation over triangle.

These relations can be combined in a matrix form that befits the general expression (14.17) for an arbitrary plane stress element:

$$\begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \zeta_1 & 0 & \zeta_2 & 0 & \zeta_3 & 0 \\ 0 & \zeta_1 & 0 & \zeta_2 & 0 & \zeta_3 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{N}^{(e)}\mathbf{u}^{(e)}. \tag{15.17}$$

§15.3.2. Strain-Displacement Equations

The strains within the elements are obtained by differentiating the shape functions with respect to x and y. Using (15.14) and the general form (14.18) we get

$$\mathbf{e} = \mathbf{D}\mathbf{N}^{(e)}\mathbf{u}^{(e)} = \frac{1}{2A}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}\begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{B}\mathbf{u}^{(e)}, \tag{15.18}$$

in which $\mathbf{D}$ denotes the symbolic strain-to-displacement differentiation operator given in (14.6).

Note that the strains are *constant* over the element. This is the origin of the name *constant strain triangle* (CST) given it in many finite element publications.

§15.3.3. 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:

$$\boldsymbol{\sigma} = \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix}\begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \mathbf{E}\mathbf{e}, \tag{15.19}$$

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.

§15.3.4. The Stiffness Matrix

The element stiffness matrix is given by the general formula (14.23), which is repeated here for convenience:

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{B}^T\mathbf{E}\mathbf{B}\; d\Omega^{(e)}, \tag{15.20}$$

where $\Omega^{(e)}$ is the triangle domain, and h is 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:

$$\begin{aligned}\mathbf{K}^{(e)} &= \mathbf{B}^T\mathbf{E}\mathbf{B}\int_{\Omega^{(e)}} h\, d\Omega^{(e)} \\ &= \frac{1}{4A^2}\begin{bmatrix} y_{23} & 0 & x_{32} \\ 0 & x_{32} & y_{23} \\ y_{31} & 0 & x_{13} \\ 0 & x_{13} & y_{31} \\ y_{12} & 0 & x_{21} \\ 0 & x_{21} & y_{12} \end{bmatrix}\begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}\int_{\Omega^{(e)}} h\, d\Omega^{(e)}.\end{aligned} \tag{15.21}$$

If the thickness h is uniform over the element the integral in (15.21) is simply hA, and we obtain the closed form

$$\boxed{\mathbf{K}^{(e)} = Ah\,\mathbf{B}^T\mathbf{E}\mathbf{B} = \frac{h}{4A}\begin{bmatrix} y_{23} & 0 & x_{32} \\ 0 & x_{32} & y_{23} \\ y_{31} & 0 & x_{13} \\ 0 & x_{13} & y_{31} \\ y_{12} & 0 & x_{21} \\ 0 & x_{21} & y_{12} \end{bmatrix} \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{21} & y_{12} \end{bmatrix}.} \tag{15.22}$$

Exercise 15.1 deals with the case of a linearly varying thickness.

§15.3.5. The Consistent Nodal Force Vector

For simplicity we consider here only internal body forces[2] defined by the vector field

$$\mathbf{b} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} \tag{15.23}$$

which is specified per unit of volume. The consistent nodal force vector $\mathbf{f}^{(e)}$ is given by the general formula (14.23) of the previous Chapter:

$$\mathbf{f}^{(e)} = \int_{\Omega^{(e)}} h\,(\mathbf{N}^{(e)})^T \mathbf{b}\,d\Omega^{(e)} = \int_{\Omega^{(e)}} h \begin{bmatrix} \zeta_1 & 0 \\ 0 & \zeta_1 \\ \zeta_2 & 0 \\ 0 & \zeta_2 \\ \zeta_3 & 0 \\ 0 & \zeta_3 \end{bmatrix} \mathbf{b}\;d\Omega^{(e)}. \tag{15.24}$$

The simplest case is when the body force components (15.23) as well as the thickness h are constant over the element. Then we need the integrals

$$\int_{\Omega^{(e)}} \zeta_1\;d\Omega^{(e)} = \int_{\Omega^{(e)}} \zeta_2\;d\Omega^{(e)} = \int_{\Omega^{(e)}} \zeta_3\;d\Omega^{(e)} = \tfrac{1}{3}A \tag{15.25}$$

which replaced into (15.24) gives

$$\boxed{\mathbf{f}^{(e)} = \frac{Ah}{3}\begin{bmatrix} b_x \\ b_y \\ b_x \\ b_y \\ b_x \\ b_y \end{bmatrix}.} \tag{15.26}$$

This agrees with the simple force-lumping procedure, which assigns one third of the total force along the $\{x, y\}$ directions: Ahb_x and Ahb_y, to each corner.

2 For consistent force computations corresponding to distributed boundary loads over a side, see Exercise 15.4.

REMARK 15.2

The integral (15.25) is a particular case of the general integration formula of monomials in triangular coordinates:

$$\frac{1}{2A}\int_{\Omega^{(e)}} \zeta_1^i\,\zeta_2^j\,\zeta_3^k\;d\Omega^{(e)} = \frac{i!\,j!\,k!}{(i+j+k+2)!},\qquad i\ge 0,\ j\ge 0,\ k\ge 0. \tag{15.27}$$

which can be proven by recursive integration by parts. This formula *only holds for triangles with straight sides*, and consequently is useless for higher order curved elements. The result (15.25) is obtained by setting $i=1,\ j=k=0$ in (15.27).

§15.4. *CONSISTENCY VERIFICATION

It remains to check whether the piecewise linear expansion (15.16) for the element displacements meets the completeness and continuity criteria studied in more detail in Chapter 19 for finite element trial functions. Such *consistency* conditions are sufficient to insure convergence towards the exact solution of the mathematical model as the mesh is refined.

The variational index for the plane stress problem is $m = 1$. Consequently the trial functions should be 1-complete, C^0 continuous, and C^1 piecewise differentiable.

§15.4.1. *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*. For example, over the side 1–2 of an individual triangle:

$$\begin{aligned} u_x &= u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3 = u_{x1}\zeta_1 + u_{x2}\zeta_2,\\ u_y &= u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3 = u_{y1}\zeta_1 + u_{y2}\zeta_2, \end{aligned} \tag{15.28}$$

because $\zeta_3 = 0$ along that side.

An identical argument holds for that side when it belongs to an adjacent triangle, such as elements $(e1)$ and $(e2)$ shown in Figure 15.4. Since the node values on all elements that meet at a node are the same, u_x and u_y match along the side, and the trial function is C^0 continuous across elements.Because the functions are continuous inside the elements, it follows that the conformity requirement is met.

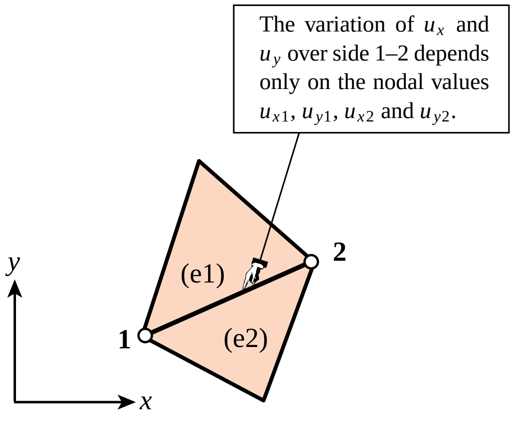

The variation of u_x and u_y over side 1–2 depends only on the nodal values u_{x1}, u_{y1}, u_{x2} and u_{y2}.
(e1)
(e2)
1
2
y
x

Figure 15.4. Interelement continuity check.

§15.4.2. *Checking Completeness

The completeness condition for variational order $m = 1$ require that the shape functions $N_i = \zeta_i$ be able to represent exactly any linear displacement field:

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y,\qquad u_y = \beta_0 + \beta_1 x + \beta_1 y. \tag{15.29}$$

To check this we obtain the nodal values associated with the motion (15.29)

$$\left.\begin{aligned} u_{xi} &= \alpha_0 + \alpha_1 x_i + \alpha_2 y_i\\ u_{yi} &= \beta_0 + \beta_1 x_i + \beta_2 y_i \end{aligned}\right\}\qquad i = 1, 2, 3, \tag{15.30}$$

replace them into (15.17) and see if we recover (15.29). Here are the detailed calculations for component u_x:

$$
\begin{aligned}
u_x &= \sum_i u_{xi}\zeta_i = \sum_i (\alpha_0 + \alpha_1 x_i + \alpha_2 y_i)\zeta_i = \sum_i (\alpha_0\zeta_i + \alpha_1 x_i\zeta_i + \alpha_2 y_i\zeta_i) \\
&= \alpha_0 \sum_i \zeta_i + \alpha_1 \sum_i (x_i\zeta_i) + \alpha_2 \sum_i (y_i\zeta_i) = \alpha_0 + \alpha_1 x + \alpha_2 y.
\end{aligned}
\tag{15.31}
$$

Component u_y can be similarly checked. Consequently (15.17) 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 continuity requirements are satisfied.

§15.5. *THE TONTI DIAGRAM OF THE LINEAR TRIANGLE

For further developments covered in advanced FEM course, it is convenient to split the governing equations of the element. In the case of the linear triangle they are, omitting element superscripts:

$$\epsilon = \mathbf{Bu}, \quad \boldsymbol{\sigma} = \mathbf{Ee}, \quad \mathbf{f} = \mathbf{A}^T\boldsymbol{\sigma} = V\mathbf{B}^T\boldsymbol{\sigma}. \tag{15.32}$$

in which $V = h_m A$ is the volume of the element, h_m being the mean thickness.

The equations (15.32) may be represented with the matrix diagram shown in Figure 15.5.

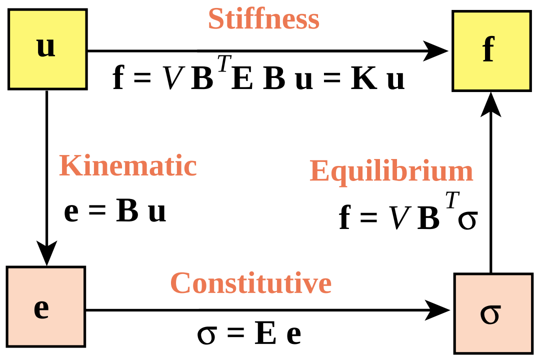


Figure 15.5. Tonti matrix diagram for linear triangle.

§15.6. *DERIVATION USING NATURAL STRAINS AND STRESSES

The element derivation in §15.3 uses Cartesian strains and stresses, as well as $x - y$ displacements. The only intrinsic quantities are the triangle coordinates. This subsection 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.

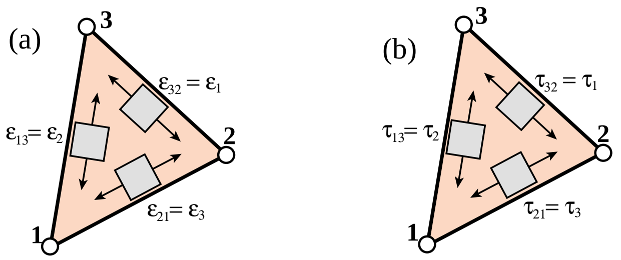


Figure 15.6. Geometry-intrinsic strain and stress fields for the 3-node linear triangle: (a) natural strains ϵ_i, (b) natural stresses τ_i.

§15.6.1. *Natural Strains and Stresses

Natural strains are extensional strains directed parallel to the triangle sides, as shown in Figure 15.6(a). They are denoted by $\epsilon_{21} \equiv \epsilon_3$, $\epsilon_{32} \equiv \epsilon_1$, and $\epsilon_{13} \equiv \epsilon_2$. Because they are constant over the triangle, no node value association is needed.

Natural stresses are normal stresses directed parallel to the triangle sides, as shown in Figure 15.6(b). They are denoted by $\tau_{21} \equiv \tau_3$, $\tau_{32} \equiv \tau_1$, and $\tau_{13} \equiv \tau_2$. Because they are constant over the triangle, no node value association is needed.

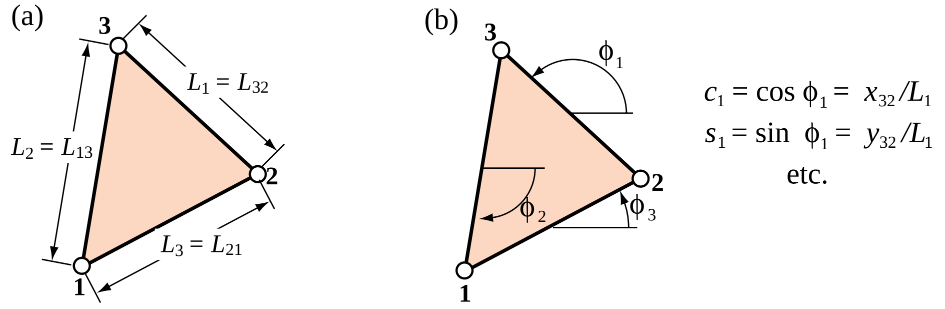


Figure 15.7. Quantities appearing in natural strain and stress calculations: (a) side lengths, (b) side directions.

The natural strains can be related to Cartesian strains by the tensor transformation[3]

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} c_1^2 & s_1^2 & s_1c_1 \\ c_2^2 & s_2^2 & s_2c_2 \\ c_3^2 & s_3^2 & s_3c_3 \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \mathbf{T}_e^{-1}\mathbf{e}. \tag{15.33}$$

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 sine/cosines of the side directions with respect to $\{x, y\}$, as illustrated in Figure 15.7. The inverse of this relation is

$$\mathbf{e} = \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \frac{1}{4A^2} \begin{bmatrix} y_{31}y_{21}L_1^2 & y_{12}y_{32}L_2^2 & y_{23}y_{13}L_3^2 \\ x_{31}x_{21}L_1^2 & x_{12}x_{32}L_2^2 & x_{23}x_{13}L_3^2 \\ (y_{31}x_{12} + x_{13}y_{21})L_1^2 & (y_{12}x_{23} + x_{21}y_{32})L_2^2 & (y_{23}x_{31} + x_{32}y_{13})L_3^2 \end{bmatrix} \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \mathbf{T}_e\boldsymbol{\epsilon}. \tag{15.34}$$

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

$$\mathcal{U} = \tfrac{1}{2}\mathbf{e}^T\mathbf{E}\mathbf{e} = \tfrac{1}{2}\boldsymbol{\epsilon}^T\mathbf{E}_n\boldsymbol{\epsilon}, \qquad \mathbf{E}_n = \mathbf{T}_e^T\mathbf{E}\mathbf{T}_e. \tag{15.35}$$

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.

[3] This is the “straingage rosette” transformation studied in Mechanics of Materials books.

§15.6.2. *Covariant Node Displacements

Covariant node displacements d_i are directed along the side directions, as shown in Figure 15.8, which defines the notation used for them. They are related to the Cartesian node displacements by

$$\mathbf{d} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \begin{bmatrix} c_3 & s_3 & 0 & 0 & 0 & 0 \\ c_2 & s_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & c_1 & s_1 & 0 & 0 \\ 0 & 0 & c_3 & s_3 & 0 & 0 \\ 0 & 0 & 0 & 0 & c_2 & s_2 \\ 0 & 0 & 0 & 0 & c_1 & s_1 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \mathbf{T}_d \mathbf{u}. \tag{15.36}$$

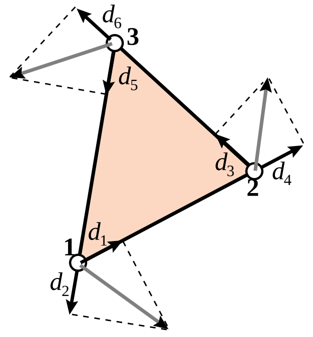


Figure 15.8. Covariant node displacements d_i.

The inverse relation is

$$\mathbf{u} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \\ u_{x3} \\ u_{y3} \end{bmatrix} = \frac{1}{2A} \begin{bmatrix} L_3 y_{31} & L_2 y_{21} & 0 & 0 & 0 & 0 \\ L_3 x_{13} & L_2 x_{12} & 0 & 0 & 0 & 0 \\ 0 & 0 & L_1 y_{12} & L_3 y_{32} & 0 & 0 \\ 0 & 0 & L_1 x_{21} & L_3 x_{23} & 0 & 0 \\ 0 & 0 & 0 & 0 & L_2 y_{23} & L_1 y_{13} \\ 0 & 0 & 0 & 0 & L_2 x_{32} & L_1 x_{31} \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \mathbf{T}_d^{-1} \mathbf{d}. \tag{15.37}$$

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:

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \end{bmatrix} = \begin{bmatrix} 0 & 0 & -1/L_1 & 0 & 0 & 1/L_1 \\ 0 & 1/L_2 & 0 & 0 & -1/L_2 & 0 \\ -1/L_3 & 0 & 0 & 1/L_3 & 0 & 0 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \end{bmatrix} = \mathbf{B}_\epsilon \mathbf{d}. \tag{15.38}$$

§15.6.3. *The Natural Stiffness Matrix

The natural stiffness matrix for constant h is

$$\mathbf{K}_n = (Ah)\, \mathbf{B}_\epsilon^T \mathbf{E}_n \mathbf{B}_\epsilon, \quad \mathbf{E}_n = \mathbf{T}_e^T \mathbf{E} \mathbf{T}_e. \tag{15.39}$$

The Cartesian stiffness matrix is

$$\mathbf{K} = \mathbf{T}_d^T \mathbf{K}_n \mathbf{T}_d. \tag{15.40}$$

Comparing with $\mathbf{K} = (Ah)\, \mathbf{B}^T \mathbf{E} \mathbf{B}$ we see that

$$\mathbf{B} = \mathbf{T}_e \mathbf{B}_\epsilon \mathbf{T}_d, \qquad \mathbf{B}_\epsilon = \mathbf{T}_e^{-1} \mathbf{B} \mathbf{T}_d^{-1}. \tag{15.41}$$

§15.7. *NOTES AND BIBLIOGRAPHY

The linear triangle, as a structural element, was first developed in the 1956 paper by Turner, Clough, Martin and Topp [15.8]. The envisioned application was modeling of skin panels of delta wings. 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 of Ray Clough [15.3] at Berkeley.

The method of [15.8] would look unfamiliar to present FEM practicioners 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 $\boldsymbol{\sigma}$. Lumping the stress field to the nodes fives 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 $\mathbf{e}$: $\mathbf{K} = \mathbf{L}\mathbf{E}\mathbf{A}$. It happens that $\mathbf{L} = V\mathbf{A}^T$ and so $\mathbf{K} = V\mathbf{A}^T\mathbf{E}\mathbf{A}$ happily turned out to be symmetric. (This $\mathbf{A}$ is the $\mathbf{B}$ of (15.13) times $2A$.)

The derivation from assumed displacements evolved slowly, and it is not clear who presented it first. The equivalence, through energy principles, had been noted by Gallagher [15.6]. Early displacement derivations typically started by starting from linear polynomials in the Cartesian coordinates. For example Przemieniecki [15.7] begins with

$$u_x = c_1 x + c_2 y + c_3, \quad u_y = c_4 x + c_5 y + c_6. \tag{15.42}$$

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 [15.4]. Even for this simple element the approach is unnecessarily complicated and leads to long computations. The elegant derivation in natural coordinates was popularized by Argyris [15.2].

The idea of using linear interpolation over a triangular net precedes [15.8] by 13 years. It appears in the Appendix of an article by Courant [15.5], 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 the original derivation in [15.8] was not based on displacements.

References

[15.1] Argyris, J. H., Kelsey, S., *Energy Theorems and Structural Analysis*, London, Butterworth, 1960; Part I reprinted from *Aircr. Engrg.*, **26**, Oct-Nov 1954 and **27**, April-May 1955.

[15.2] Argyris J. H., Continua and discontinua, *Proceedings 1st Conference on Matrix Methods in Structural Mechanics*, AFFDL-TR-66-80, Air Force Institute of Technology, Dayton, Ohio, 10–170, 1965.

[15.3] Clough, R. W., The finite element method in plane stress analysis, *Proc. 2nd ASCE Conf. on Electronic Computation*, Pittsburgh, Pa, 1960.

[15.4] 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.

[15.5] 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.

[15.6] Gallaguer, R. H., *A Correlation Study of Methods of Matrix Structural Analysis*, Pergamon, Oxford, 1964.

[15.7] Przemieniecki, J. S., *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968; Dover edition 1986.

[15.8] Turner, M. J., Clough, R. W., Martin, H. C., Topp, L. J., Stiffness and deflection analysis of complex structures, *J. Aero. Sci.*, **23**, 805–824, 1956.

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

$$h(\zeta_1, \zeta_2, \zeta_3) = h_1\zeta_1 + h_2\zeta_2 + h_3\zeta_3, \tag{E15.1}$$

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 (15.23) but with h replaced by the mean thickness $h_m = (h_1 + h_2 + h_3)/3$. *Hint*: use (15.22) and (15.28).

EXERCISE 15.2

[A:20] The exact integrals of triangle-coordinate monomials over a straight-sided triangle are given by the formula

$$\frac{1}{2A}\int_A \zeta_1^i\,\zeta_2^j\,\zeta_3^k\,dA = \frac{i!\,j!\,k!}{(i+j+k+2)!} \tag{E15.2}$$

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 $i + j + k \le 3$, 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 $i \ge j \ge k$).

EXERCISE 15.3

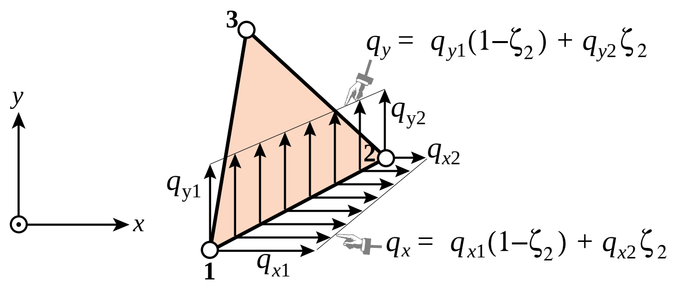
[A/C:20] Compute the consistent node force vector $\mathbf{f}^{(e)}$ for body loads over a linear triangle, if the element thickness varies as per (E15.1), $b_x = 0$, and $b_y = b_{y1}\zeta_1 + b_{y2}\zeta_2 + b_{y3}\zeta_3$. Check that for $h_1 = h_2 = h_3 = h$ and $b_{y1} = b_{y2} = b_{y3} = b_y$ you recover (15.27). For the integrals over the triangle area use the formula (E15.2). 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 linear 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

$$q_x = q_{x1}\zeta_1 + q_{x2}\zeta_2 = q_{x1}(1-\zeta_2) + q_{x2}\zeta_2, \quad q_y = q_{y1}\zeta_1 + q_{y2}\zeta_2 = q_{y1}(1-\zeta_2) + q_{y2}\zeta_2. \tag{E15.3}$$

This "line force" $\mathbf{q}$ has dimension of force per unit of side length.



3
$q_y = q_{y1}(1-\zeta_2) + q_{y2}\zeta_2$
q_{y2}
2
q_{x2}
y
x
q_{y1}
$q_x = q_{x1}(1-\zeta_2) + q_{x2}\zeta_2$
1
q_{x1}

Figure E15.1. Line forces on triangle side 1-2 for Exercise 15.4.

Procedure. 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,

$$W^{(e)} = \left(\mathbf{u}^{(e)}\right)^T \mathbf{f}^{(e)} = \int_{\Gamma^{(e)}} \mathbf{u}^T\mathbf{q}\;d\Gamma^{(e)} = \int_0^1 \mathbf{u}^T\mathbf{q}\,L_{21}\,d\zeta_2, \tag{E15.4}$$

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 (E15.4). 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];
```

EXERCISE 15.5

[C+N:15] Compute the entries of $\mathbf{K}^{(e)}$ for the following plane stress triangle:

$$x_1 = 0,\ y_1 = 0,\ x_2 = 3,\ y_2 = 1,\ x_3 = 2,\ y_3 = 2,$$
$$\mathbf{E} = \begin{bmatrix} 100 & 25 & 0 \\ 25 & 100 & 0 \\ 0 & 0 & 50 \end{bmatrix}, \qquad h = 1. \tag{E15.5}$$

This may be done by hand (it is a good exercise in matrix multiplication) or (more quickly) using the following *Mathematica* script:

```
Stiffness3NodePlaneStressTriangle[{{x1_,y1_},{x2_,y2_},{x3_,y3_}},
 Emat_,{h_}]:=Module[{x21,x13,x32,y12,y31,y23,A,Be,Ke},
   A=Simplify[(x2*y3-x3*y2+(x3*y1-x1*y3)+(x1*y2-x2*y1))/2];
   {x21,x13,x32}={x2-x1,x1-x3,x3-x2};
   {y12,y31,y23}={y1-y2,y3-y1,y2-y3};
   Be={{y23,0,y31,0,y12,0},{0,x32,0,x13,0,x21},
       {x32,y23,x13,y31,x21,y12}}/(2*A);
   Ke=A*h*Transpose[Be].Emat.Be;Return[Ke]];

Ke=Stiffness3NodePlaneStressTriangle[{{0,0},{3,1},{2,2}},
      {{100,25,0},{25,100,0},{0,0,50}},{1}];
Print["Ke=",Ke//MatrixForm];
Print["eigs of Ke=",Chop[Eigenvalues[N[Ke]]]];
Show[Graphics[Line[{{0,0},{3,1},{2,2},{0,0}}]],Axes->True];
```

Check it out: $K_{11} = 18.75$, $K_{66} = 118.75$. The last statement draws the triangle.

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. Check it with the foregoing script.

EXERCISE 15.7

[A/C:30] Let $p(\zeta_1, \zeta_2, \zeta_3)$ represent a *polynomial* expression in the natural coordinates. The integral

$$\int_{\Omega^{(e)}} p(\zeta_1, \zeta_2, \zeta_3)\, d\Omega \tag{E15.6}$$

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 15.8

[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* 4.1 Notebook that does plane stress FEM analysis using the 3-node linear 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,[4] 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
   ........
```

[4] 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
```

16 The Isoparametric Representation

TABLE OF CONTENTS

§16.1. INTRODUCTION

The technique used in Chapter 15 for the formulation of the linear triangle can be formally extended to construct quadrilateral elements as well as higher order triangles. But it quickly runs into technical difficulties in two tasks:

1. The construction of shape functions satisfying consistency requirements for higher order elements with curved boundaries becomes increasingly difficult.
2. Integrals that appear in the expressions of the element stiffness matrix and consistent nodal force vector can no longer be carried out in closed form.

These difficulties 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 form the basis of 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 is then illustrated on specific elements. In the next Chapter we apply these techniques, combined with numerical integration, 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:

$$\begin{bmatrix} 1 \\ x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix}, \tag{16.1}$$

$$\begin{aligned} u_x &= u_{x1} N_1^{(e)} + u_{x2} N_2^{(e)} + u_{x3} N_3^{(e)} = u_{x1}\zeta_1 + u_{x2}\zeta_2 + u_{x3}\zeta_3 \\ u_y &= u_{y1} N_1^{(e)} + u_{y2} N_2^{(e)} + u_{y3} N_3^{(e)} = u_{y1}\zeta_1 + u_{y2}\zeta_2 + u_{y3}\zeta_3 \end{aligned}. \tag{16.2}$$

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 the element geometry and element displacements are not treated equally. If we proceed to higher order triangles with straight sides, only the displacement expansion is refined whereas the geometry definition remains the same. Elements built according to this prescription are called *superparametric*, a term that emphasizes that unequal treatment.

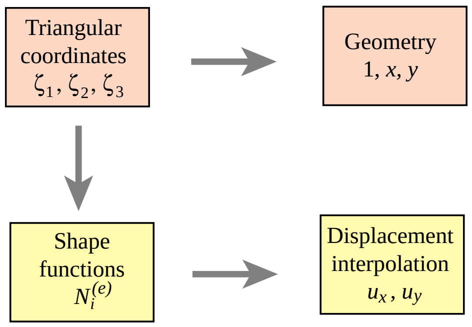

Triangular coordinates $\zeta_1, \zeta_2, \zeta_3$
Geometry $1, x, y$
Shape functions $N_i^{(e)}$
Displacement interpolation u_x, u_y

Figure 16.1. Superparametric representation of triangular element.

§16.2.2. Equalizing Geometry and Displacements

On first inspection (16.1) and (16.2) 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:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{y3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{y3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \end{bmatrix}. \tag{16.3}$$

This form emphasizes that geometry and element displacements are given by the *same* parametric representation, as illustrated in Figure 16.2.

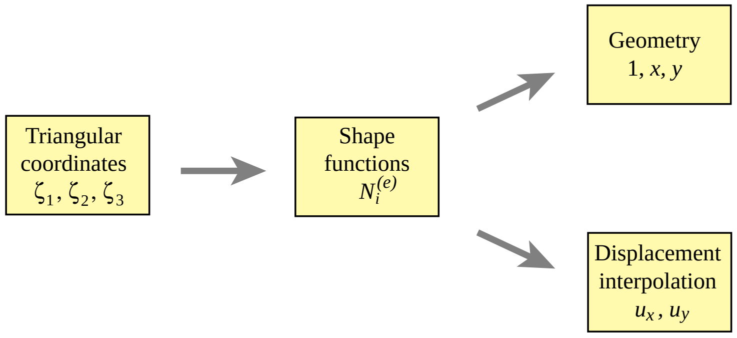

Triangular coordinates $\zeta_1, \zeta_2, \zeta_3$
Shape functions $N_i^{(e)}$
Geometry $1, x, y$
Displacement interpolation u_x, u_y

Figure 16.2. Isoparametric representation of triangular elements.

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" as illustrated in Figure 16.3.

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.

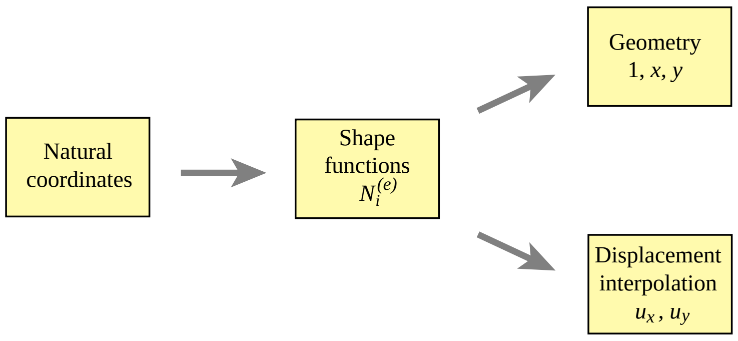

Natural coordinates
Shape functions $N_i^{(e)}$
Geometry $1, x, y$
Displacement interpolation u_x, u_y

Figure 16.3. Isoparametric representation of arbitrary 2D elements: triangles or quadrilaterals. For 3D elements, expand the geometry list to $\{1, x, y, z\}$ and the displacements to $\{u_x, u_y, u_z\}$.

REMARK 16.1

The terms *isoparametric* and *superparametric* were introduced by Irons and coworkers at Swansea in 1966. 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:

$$\boxed{1 = \sum_{i=1}^{n} N_i^{(e)}, \quad x = \sum_{i=1}^{n} x_i N_i^{(e)}, \quad y = \sum_{i=1}^{n} y_i N_i^{(e)}.} \tag{16.4}$$

Displacement interpolation:

$$\boxed{u_x = \sum_{i=1}^{n} u_{xi} N_i^{(e)}, \quad u_y = \sum_{i=1}^{n} u_{yi} N_i^{(e)}.} \tag{16.5}$$

These two sets of equations may be combined in matrix form[1] as

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ y_1 & y_2 & \ldots & y_n \\ u_{x1} & u_{x2} & \ldots & u_{xn} \\ u_{y1} & u_{y2} & \ldots & u_{yn} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ \vdots \\ N_n^{(e)} \end{bmatrix}. \tag{16.6}$$

[1] This unified matrix representation for isoparametric elements was introduced by C. A. Felippa and R. W. Clough, 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.

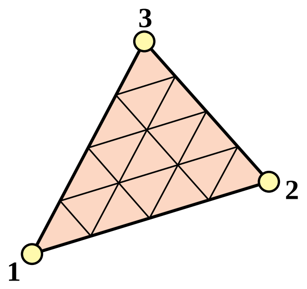


Figure 16.4. The three-node linear triangle.

The first three equations express the geometry definition, and the last two the displacement expansion. Note that additional rows may be added to this matrix expression if additional variables are interpolated by the same shape functions. For example, suppose that the thickness h and a temperature field T is interpolated from the n node values:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \\ h \\ T \end{bmatrix} = \begin{bmatrix} 1 & 1 & \ldots & 1 \\ x_1 & x_2 & \ldots & x_n \\ y_1 & y_2 & \ldots & y_n \\ u_{x1} & u_{x2} & \ldots & u_{xn} \\ u_{y1} & u_{y2} & \ldots & u_{yn} \\ h_1 & h_2 & \ldots & h_n \\ T_1 & T_2 & \ldots & T_n \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ \vdots \\ N_n^{(e)} \end{bmatrix}. \tag{16.7}$$

Note that the column of shape functions is not touched.

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 they use different natural coordinates.

§16.4. TRIANGULAR ELEMENTS

§16.4.1. The Linear Triangle

The three-noded linear triangle studied in Chapter 15 and reproduced in Figure 16.4, may be presented as an isoparametric element:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \end{bmatrix}. \tag{16.8}$$

The shape functions are simply the triangular coordinates:

$$N_1^{(e)} = \zeta_1, \qquad N_2^{(e)} = \zeta_2, \qquad N_3^{(e)} = \zeta_3. \tag{16.9}$$

The linear triangle is the only triangular element that is both superparametric and isoparametric.

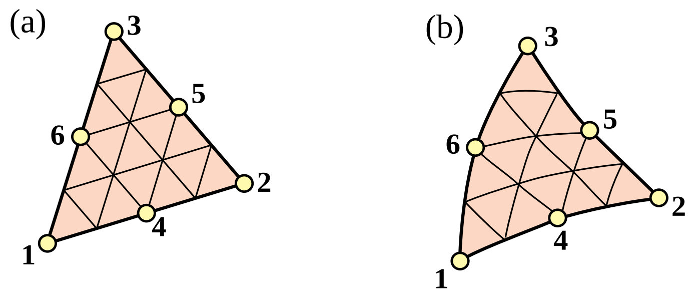


Figure 16.5. The six-node quadratic triangle: (a) the superparametric version, with straight sides and midside nodes at midpoints; (b) isoparametric version.

§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:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \end{bmatrix}. \tag{16.10}$$

The shape functions are

$$\begin{aligned} &N_1^{(e)} = \zeta_1(2\zeta_1 - 1), \quad N_2^{(e)} = \zeta_2(2\zeta_2 - 1), \quad N_3^{(e)} = \zeta_3(2\zeta_3 - 1), \\ &N_4^{(e)} = 4\zeta_1\zeta_2, \quad N_5^{(e)} = 4\zeta_2\zeta_3, \quad N_6^{(e)} = 4\zeta_3\zeta_1. \end{aligned} \tag{16.11}$$

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.

§16.5. QUADRILATERAL ELEMENTS

§16.5.1. Quadrilateral Coordinates and Iso-P Mappings

Before presenting examples of quadrilateral elements, we have to 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 straight sided and curved side quadrilaterals. These are called *quadrilateral coordinates*.

These coordinates vary from -1 on one side to $+1$ at the other, taking the value zero on the medians. This particular variation range (instead of, say, 0 to 1) was chosen by the investigators

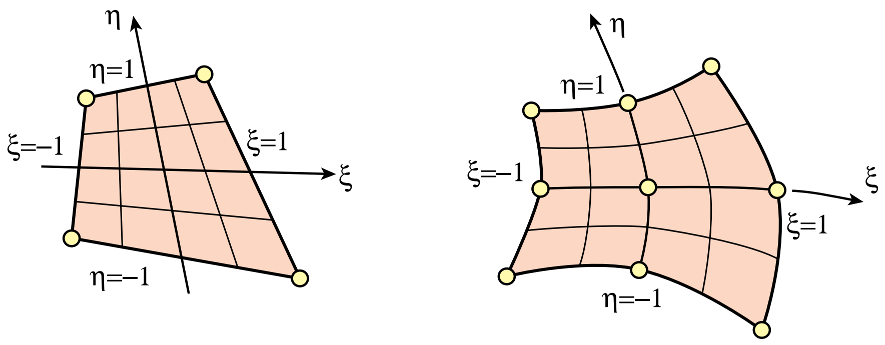


Figure 16.6. Quadrilateral coordinates.

who originally developed isoparametric quadrilaterals to facilitate the use of the standard Gauss integration formulas. Those formulas are discussed in the following Chapter.

In some FEM derivations it is useful 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

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \end{bmatrix}. \tag{16.12}$$

The shape functions are

$$\begin{aligned} N_1^{(e)} &= \tfrac{1}{4}(1-\xi)(1-\eta), & N_2^{(e)} &= \tfrac{1}{4}(1+\xi)(1-\eta), \\ N_3^{(e)} &= \tfrac{1}{4}(1+\xi)(1+\eta), & N_4^{(e)} &= \tfrac{1}{4}(1-\xi)(1+\eta). \end{aligned} \tag{16.13}$$

These functions vary *linearly* on quadrilateral coordinate lines $\xi = const$ and $\eta = const$, but are not linear polynomials as in the case of the three-node triangle.

§16.5.3. The Biquadratic Quadrilateral

The nine-node quadrilateral shown in Figure 16.8(a) is the next *bicomplete* member of the quadri-

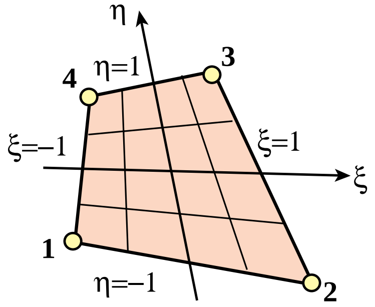


Figure 16.7. The four-node bilinear quadrilateral.

lateral family. It has eight external nodes and one internal node. It is defined by

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & x_9 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & y_9 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} & u_{x9} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} & u_{y9} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ \vdots \\ N_9^{(e)} \end{bmatrix}. \tag{16.14}$$

The shape functions are

$$\begin{array}{lll} N_1^{(e)} = \frac{1}{4}(1-\xi)(1-\eta)\xi\eta & N_5^{(e)} = -\frac{1}{2}(1-\xi^2)(1-\eta)\eta & \\ N_2^{(e)} = -\frac{1}{4}(1+\xi)(1-\eta)\xi\eta & N_6^{(e)} = \frac{1}{2}(1+\xi)(1-\eta^2)\xi & N_9^{(e)} = (1-\xi^2)(1-\eta^2) \\ \ldots & \ldots & \end{array} \tag{16.15}$$

These functions vary *quadratically* along the coordinate lines $\xi = const$ and $\eta = const$. The shape function associated with the internal node 9 is called a *bubble function* in the FEM literature.

Figure 16.8(b) depicts an widely used eight-node variant called the “serendipity” quadrilateral.[2] The internal node is eliminated by kinematic constraints.

§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

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y, \qquad u_y = \beta_0 + \beta_1 x + \beta_2 y. \tag{16.16}$$

[2] A name that originated from circumstances surrounding the element discovery.

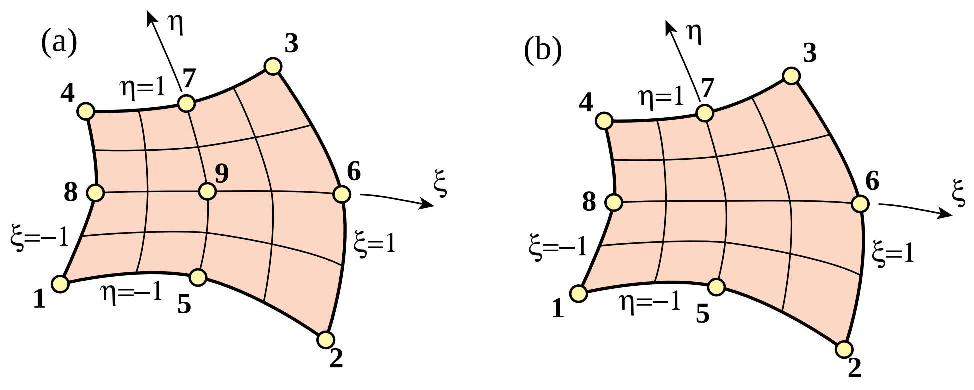


Figure 16.8. Two widely used higher order quadrilaterals: (a) the nine-node biquadratic quadrilateral; (b) the eight-node "serendipity" quadrilateral.

To carry out the check, evaluate (16.16) at the nodes

$$u_{xi} = \alpha_0 + \alpha_1 x_i + \alpha_2 y_i \qquad u_{yi} = \beta_0 + \beta_1 x_i + \beta_2 y_i, \qquad i = 1, \ldots n. \tag{16.17}$$

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:

$$u_x = \sum_{i=1}^{n} (\alpha_0 + \alpha_1 x_i + \alpha_2 y_i)\, N_i^{(e)} = \alpha_0 \sum_i N_i^{(e)} + \alpha_1 \sum_i x_i N_i^{(e)} + \alpha_2 \sum_i y_i N_i^{(e)} = \alpha_0 + \alpha_1 x + \alpha_2 y. \tag{16.18}$$

For the last step we have used the geometry definition relations (16.4), reproduced here for convenience:

$$\boxed{1 = \sum_{i=1}^{n} N_i^{(e)}, \quad x = \sum_{i=1}^{n} x_i N_i^{(e)}, \quad y = \sum_{i=1}^{n} y_i N_i^{(e)}.} \tag{16.19}$$

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.[3]

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 checked by hand for simple elements. Here are two examples.

Check for the linear triangle: directly from the definition of triangular coordinates,

$$N_1^{(e)} + N_2^{(e)} + N_3^{(e)} = \zeta_1 + \zeta_2 + \zeta_3 = 1. \tag{16.20}$$

[3] This derivation is due to B. M. Irons, see B. M. Irons and S. Ahmad, *Techniques of Finite Elements*, Ellis Horwood Ltd, Chichester, UK, 1980. The property was known since the mid 1960s and contributed substantially to the rapid acceptance of iso-P elements.

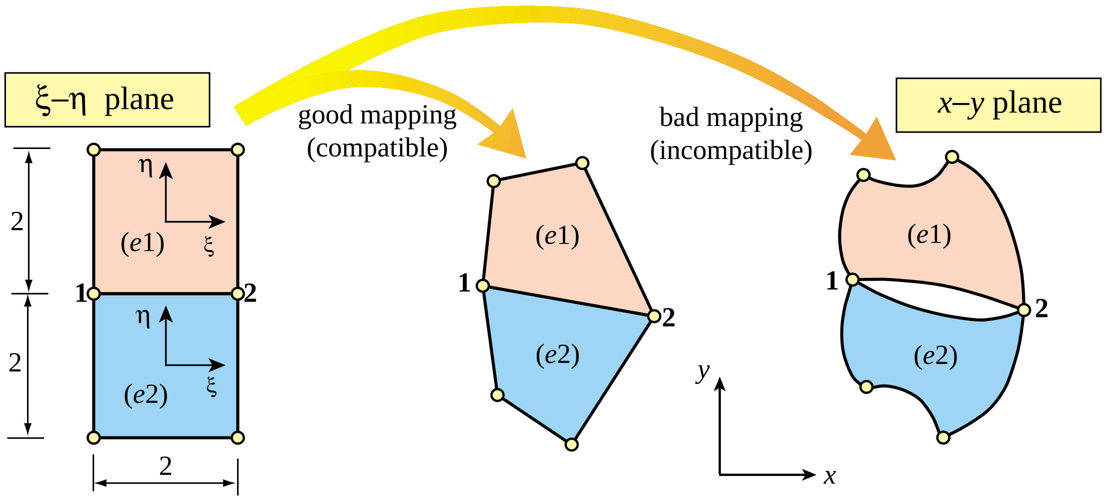


Figure 16.9. Good and bad isoparametric mappings of 4-node quadrilateral from the $\{\xi, \eta\}$ reference plane to the $\{x, y\}$ physical plane.

Check for the 4-node bilinear quadrilateral:

$$\begin{aligned} N_1^{(e)} + N_2^{(e)} + N_3^{(e)} + N_4^{(e)} &= \tfrac{1}{4}(1-\xi-\eta+\xi\eta) + \tfrac{1}{4}(1+\xi-\eta-\xi\eta) \\ &+ \tfrac{1}{4}(1+\xi+\eta+\xi\eta) + \tfrac{1}{4}(1-\xi+\eta-\xi\eta) = 1 \end{aligned} \tag{16.21}$$

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

$$x = \sum_{i=1}^{4} x_i N_i^{(e)}(\xi, \eta), \quad y = \sum_{i=1}^{4} y_i N_i^{(e)}(\xi, \eta). \tag{16.22}$$

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.[4]

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 posed 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:

$$\boxed{\text{Unit-sum condition + interelement compatibility} \rightarrow \text{completeness.}} \tag{16.23}$$

[4] 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 bilinear elements in §23.3.

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*.[5] 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 subject of Exercises 16.4 through 16.6.

[5] 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 expressable 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:

$$N_1^{(e)}(\xi) = a_0 + a_1\xi + a_2\xi^2, \quad N_2^{(e)}(\xi) = b_0 + b_1\xi + b_2\xi^2, \quad N_3^{(e)}(\xi) = c_0 + c_1\xi + c_2\xi^2. \tag{E16.1}$$

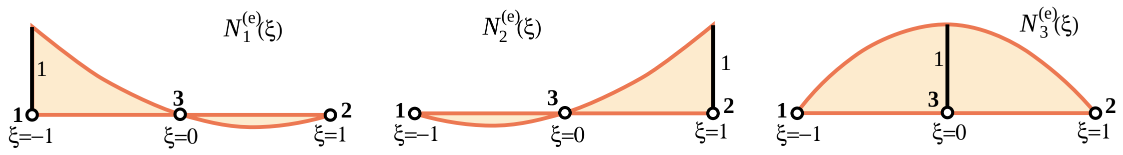


Figure E16.1. Isoparametric shape functions for 3-node bar element (sketch). Node 3 has been drawn at the 1–2 midpoint but it may be moved away from it.

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

$$N_1^{(e)}(\xi) = -\tfrac{1}{2}\xi(1-\xi), \qquad N_2^{(e)}(\xi) = \tfrac{1}{2}\xi(1+\xi), \qquad N_3^{(e)}(\xi) = 1-\xi^2. \tag{E16.2}$$

Verify that their sum is identically one.

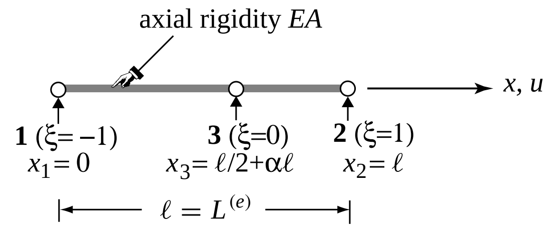


Figure E16.2. The 3-node bar element in its local system.

EXERCISE 16.5

[A/C:10+10+15+5+10] 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

$$\begin{bmatrix} 1 \\ x \\ u \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ u_1 & u_2 & u_3 \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \end{bmatrix}, \tag{E16.3}$$

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

$$x_1 = 0, \qquad x_2 = \ell, \qquad x_3 = (\tfrac{1}{2} + \alpha)\ell, \tag{E16.4}$$

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.

(a) 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 $-1 \le \xi \le 1$; (ii) if $\alpha = 0$, $J = \ell/2$ is constant over the element.

(b) 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).

(c) Show that the element stiffness matrix is given by

$$\mathbf{K}^{(e)} = \int_0^{\ell} EA\,\mathbf{B}^T\mathbf{B}\,dx = \int_{-1}^{1} EA\,\mathbf{B}^T\mathbf{B}\,J\,d\xi. \tag{E16.5}$$

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.

(d) What is the minimum number of Gauss points needed to integrate $\mathbf{K}^{(e)}$ exactly if $\alpha = 0$?

(e) This item addresses the question of why $\mathbf{K}^{(e)}$ was computed by numerical integration in (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,alpha,xi]; (* Define J and B={{B1,B2,B3}} here *)
Ke=Simplify[Integrate[EA*Transpose[B].B*J,{xi,-1,1}]];
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 fourth line of which will trigger error messages. Comment on why the exact stiffness cannot be evaluated directly at $\alpha = 0$, and why it contains complex numbers. A Taylor series expansion about $\alpha = 0$ removes both problems but the 2-point Gauss integration rule gives the same answer without the gyrations.

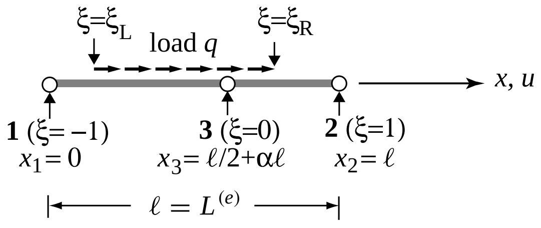


Figure E16.3. The 3-node bar element under a "box" axial load q.

EXERCISE 16.6

[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 $-1 \le \xi_L < \xi_R \le 1$. See Figure E16.3. Use

$$\mathbf{f}^{(e)} = \int_{-\xi_L}^{\xi_R} q\, \mathbf{N}^T\, J\, d\xi, \tag{E16.6}$$

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

§17.1. INTRODUCTION

In this Chapter the isoparametric representation of element geometry and shape functions discussed in Chapter 16 is used to construct *quadrilateral* elements for the plane stress problem. The formulas presented in Chapter 14 for the stiffness matrix and consistent load vector of general plane stress elements are of course applicable to the element formulation. For a practical implementation, however, we must go through the following steps:

1. Construction of shape functions.
2. Computations of shape function derivatives to evaluate the strain-displacement matrix.
3. Numerical integration over the element by Gauss quadrature rules.

The first topic was dealt with in Chapter 16 in recipe form, and is systematically covered in the next Chapter. 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 more systematically in Chapter 23.

We shall not cover 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. Since the 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 the following derivations we need the Jacobian of two-dimensional transformations that connect the differentials of $\{x, y\}$ to those of $\{\xi, \eta\}$ and vice-versa:

$$\begin{bmatrix} dx \\ dy \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \mathbf{J}^T \begin{bmatrix} d\xi \\ d\eta \end{bmatrix}, \quad \begin{bmatrix} d\xi \\ d\eta \end{bmatrix} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \xi}{\partial x} \\ \frac{\partial \eta}{\partial x} & \frac{\partial \eta}{\partial y} \end{bmatrix} \begin{bmatrix} dx \\ dy \end{bmatrix} = \mathbf{J}^{-T} \begin{bmatrix} dx \\ dy \end{bmatrix}, \tag{17.1}$$

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):

$$\mathbf{J} = \frac{\partial(x, y)}{\partial(\xi, \eta)} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}, \qquad \mathbf{J}^{-1} = \frac{\partial(\xi, \eta)}{\partial(x, y)} = \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix} \tag{17.2}$$

In finite element work $\mathbf{J}$ and $\mathbf{J}^{-1}$ are often called the *Jacobian* and *inverse Jacobian*, respectively; the fact that it is a matrix being understood. The scalar symbol J means the determinant of

J: $J = |\mathbf{J}| = \det \mathbf{J}$. In one dimension **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

Note that the matrices relating the differentials in (17.1) are the transposes of **J** and $\mathbf{J}^{-1}$. The reason is that differentials transform as covariant quantities as per the chain rule: $dx = (\partial x/\partial\xi)\, d\xi + (\partial x/\partial\eta)\, d\eta$, etc. But Jacobians have traditionally been arranged as in (17.2) because of earlier use in contravariant transformations: $\partial\phi/\partial\xi = (\partial\phi/\partial x)(\partial x/\xi) + (\partial\phi/\partial y)(\partial y/\xi)$, as in (17.6) below.

REMARK 17.2

To show that **J** and $\mathbf{J}^{-1}$ are in fact inverses of each other we form their product:

$$\mathbf{J}^{-1}\mathbf{J} = \begin{bmatrix} \frac{\partial x}{\partial\xi}\frac{\partial\xi}{\partial x} + \frac{\partial x}{\partial\eta}\frac{\partial\eta}{\partial x} & \frac{\partial y}{\partial\xi}\frac{\partial\xi}{\partial x} + \frac{\partial y}{\partial\eta}\frac{\partial\eta}{\partial x} \\ \frac{\partial x}{\partial\xi}\frac{\partial\xi}{\partial y} + \frac{\partial x}{\partial\eta}\frac{\partial\eta}{\partial y} & \frac{\partial y}{\partial\xi}\frac{\partial\xi}{\partial y} + \frac{\partial y}{\partial\eta}\frac{\partial\eta}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial x} & \frac{\partial y}{\partial x} \\ \frac{\partial x}{\partial y} & \frac{\partial y}{\partial y} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{17.3}$$

where we have taken into account that

$$x = x(\xi, \eta), \qquad y = y(\xi, \eta), \tag{17.4}$$

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 §16.7. The derivatives with respect to x and y are given by the chain rule:

$$\begin{aligned} \frac{\partial N_i^{(e)}}{\partial x} &= \frac{\partial N_i^{(e)}}{\partial\xi}\frac{\partial\xi}{\partial x} + \frac{\partial N_i^{(e)}}{\partial\eta}\frac{\partial\eta}{\partial x}, \\ \frac{\partial N_i^{(e)}}{\partial y} &= \frac{\partial N_i^{(e)}}{\partial\xi}\frac{\partial\xi}{\partial y} + \frac{\partial N_i^{(e)}}{\partial\eta}\frac{\partial\eta}{\partial y}. \end{aligned} \tag{17.5}$$

In matrix form:

$$\begin{bmatrix} \dfrac{\partial N_i^{(e)}}{\partial x} \\ \dfrac{\partial N_i^{(e)}}{\partial y} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial\xi}{\partial x} & \dfrac{\partial\eta}{\partial x} \\ \dfrac{\partial\xi}{\partial y} & \dfrac{\partial\eta}{\partial y} \end{bmatrix} \begin{bmatrix} \dfrac{\partial N_i^{(e)}}{\partial\xi} \\ \dfrac{\partial N_i^{(e)}}{\partial\eta} \end{bmatrix} = \frac{\partial(\xi,\eta)}{\partial(x,y)} \begin{bmatrix} \dfrac{\partial N_i^{(e)}}{\partial\xi} \\ \dfrac{\partial N_i^{(e)}}{\partial\eta} \end{bmatrix} = \mathbf{J}^{-1} \begin{bmatrix} \dfrac{\partial N_i^{(e)}}{\partial\xi} \\ \dfrac{\partial N_i^{(e)}}{\partial\eta} \end{bmatrix}. \tag{17.6}$$

where $\mathbf{J}^{-1}$ is defined in (17.4). The computation of **J** is addressed in the next subsection.

§17.2.3. Computing the Jacobian Matrix

To compute the entries of **J** at any quadrilateral location we make use of the last two geometric relations in (16.4), which are repeated here for convenience:

$$x = \sum_{i=1}^{n} x_i N_i^{(e)}, \qquad y = \sum_{i=1}^{n} y_i N_i^{(e)}. \tag{17.7}$$

Differentiating with respect to the quadrilateral coordinates,

$$\begin{aligned} \frac{\partial x}{\partial \xi} &= \sum_{i=1}^{n} x_i \frac{\partial N_i^{(e)}}{\partial \xi}, \qquad & \frac{\partial y}{\partial \xi} &= \sum_{i=1}^{n} y_i \frac{\partial N_i^{(e)}}{\partial \xi}, \\ \frac{\partial x}{\partial \eta} &= \sum_{i=1}^{n} x_i \frac{\partial N_i^{(e)}}{\partial \eta}, \qquad & \frac{\partial y}{\partial \eta} &= \sum_{i=1}^{n} y_i \frac{\partial N_i^{(e)}}{\partial \eta}. \end{aligned} \tag{17.8}$$

because the x_i and y_i do not depend on ξ and η. In matrix form:

$$\mathbf{J} = \mathbf{P}\mathbf{X} = \begin{bmatrix} \dfrac{\partial N_1^{(e)}}{\partial \xi} & \dfrac{\partial N_2^{(e)}}{\partial \xi} & \dots & \dfrac{\partial N_n^{(e)}}{\partial \xi} \\ \dfrac{\partial N_1^{(e)}}{\partial \eta} & \dfrac{\partial N_2^{(e)}}{\partial \eta} & \dots & \dfrac{\partial N_n^{(e)}}{\partial \eta} \end{bmatrix} \begin{bmatrix} x_1 & y_1 \\ x_2 & y_2 \\ \vdots & \vdots \\ x_n & y_n \end{bmatrix}. \tag{17.9}$$

Given a quadrilateral point of coordinates ξ, η we can calculate the entries of **J** using (17.9). The inverse Jacobian $\mathbf{J}^{-1}$ may be obtained by numerically inverting this 2×2 matrix.

REMARK 17.3

The symbolic inversion of **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 factors that motivated the use of Gaussian numerical quadrature, as discussed below.

§17.2.4. The Strain-Displacement Matrix

The strain-displacement matrix **B** that appears in the computation of the element stiffness matrix is given by the general expression (14.18), reproduced here for convenience:

$$\mathbf{e} = \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial N_1^{(e)}}{\partial x} & 0 & \dfrac{\partial N_2^{(e)}}{\partial x} & 0 & \dots & \dfrac{\partial N_n^{(e)}}{\partial x} & 0 \\ 0 & \dfrac{\partial N_1^{(e)}}{\partial y} & 0 & \dfrac{\partial N_2^{(e)}}{\partial y} & \dots & 0 & \dfrac{\partial N_n^{(e)}}{\partial y} \\ \dfrac{\partial N_1^{(e)}}{\partial y} & \dfrac{\partial N_1^{(e)}}{\partial x} & \dfrac{\partial N_2^{(e)}}{\partial y} & \dfrac{\partial N_2^{(e)}}{\partial x} & \dots & \dfrac{\partial N_n^{(e)}}{\partial y} & \dfrac{\partial N_n^{(e)}}{\partial x} \end{bmatrix} \mathbf{u}^{(e)} = \mathbf{B}\mathbf{u}^{(e)}. \tag{17.10}$$

The entries of the $3 \times 3n$ matrix **B** are partials of the shape functions with respect to x and y. The calculation of these entries is done via (17.9) and (17.6).

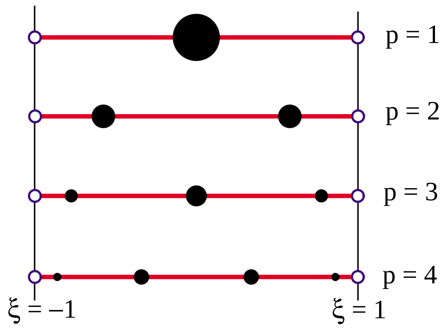


Figure 17.1. The first four one-dimensional Gauss rules $p = 1, 2, 3, 4$ depicted over the line segment $\xi = -1$ to $\xi = +1$. Gauss point locations are marked with black circles. The radii of these circles are proportional to the integration weights.

§17.3. NUMERICAL INTEGRATION BY GAUSS RULES

The use of numerical integration is essential for evaluating element integrals of isoparametric elements. The standard practice has been to use *Gauss integration*[1] because such rules use a *minimal number of sample points to achieve a desired level of accuracy*. This property is important for efficient element calculations because we shall see that at each sample point we must evaluate a matrix product. The fact that the location of the sample points in the Gauss rules is usually given by non-rational numbers is of no concern in digital computation.

§17.3.1. One Dimensional Rules

The standard Gauss integration rules in one dimension are

$$\int_{-1}^{1} F(\xi)\, d\xi \approx \sum_{i=1}^{p} w_i F(\xi_i). \tag{17.11}$$

Here $p \geq 1$ is the number of Gauss integration points, w_i are the integration weights, and ξ_i are sample-point abcissae in the interval $[-1,1]$. The use of the interval $[-1,1]$ is no restriction, because an integral over another range, say from a to b can be transformed to the standard interval via a simple linear transformation of the independent variable, as shown in the Remark below. The first four one-dimensional Gauss rules, also depicted in Figure 17.1, are:

$$\begin{aligned}
&\text{One point:} && \int_{-1}^{1} F(\xi)\, d\xi \approx 2F(0),\\
&\text{Two points:} && \int_{-1}^{1} F(\xi)\, d\xi \approx F(-1/\sqrt{3}) + F(1/\sqrt{3}),\\
&\text{Three points:} && \int_{-1}^{1} F(\xi)\, d\xi \approx \frac{5}{9}F(-\sqrt{3/5}) + \frac{8}{9}F(0) + \frac{5}{9}F(\sqrt{3/5}),\\
&\text{Four points:} && \int_{-1}^{1} F(\xi)\, d\xi \approx w_1F(\xi_1) + w_2F(\xi_2) + w_3F(\xi_3) + w_4F(\xi_4).
\end{aligned} \tag{17.12}$$

[1] Sometimes called Gauss-Legendre quadrature. Gauss presented these rules, derived from first principles, in 1814. See H. H. Goldstine, *A History of Numerical Analysis*, Springer-Verlag, New York, 1977, Sec 4.11. The name of Legendre is sometimes adjoined because the abcissas of the 1D sample points are zeros of Legendre polynomials.

For the 4-point rule, $\xi_3 = -\xi_2 = \sqrt{(3-2\sqrt{6/5})/7}$, $\xi_4 = -\xi_1 = \sqrt{(3+2\sqrt{6/5})/7}$, $w_1 = w_4 = \frac{1}{2} - \frac{1}{6}\sqrt{5/6}$, and $w_2 = w_3 = \frac{1}{2} + \frac{1}{6}\sqrt{5/6}$.

The four rules (17.12) integrate exactly polynomials in ξ of orders up to 1, 3, 5 and 7, 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.

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 $[-1,1]$ as follows:

$$\int_a^b F(x)\,dx = \int_{-1}^1 F(\xi)\,J\,d\xi, \tag{17.13}$$

in which ξ and J are defined by the mapping $x = \frac{1}{2}a(1-\xi) + \frac{1}{2}b(1+\xi) = \frac{1}{2}(a+b) + \frac{1}{2}\ell\xi$, that is, $\xi = (2/\ell)(x - \frac{1}{2}(a+b))$ and $J = dx/d\xi = \frac{1}{2}\ell$.

REMARK 17.5

Higher order Gauss rules are tabulated in standard manuals for numerical computation. For example, the widely used Handbook of Mathematical Functions[2] tabulates rules with up to 96 points. For $p > 4$ the abscissas and weights of sample points are not expressible as rational numbers or radicals, and are only given as floating-point numbers.

§17.3.2. Mathematica Implementation of 1D Rules

The following *Mathematica* module returns exact or floating-point information for the first four unidimensional Gauss rules:

```
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]},
   i,info=Null}, i=point;
   If [rule==1, info={0,2}];
   If [rule==2, info={g2[[i]],1}];
   If [rule==3, info={g3[[i]],w3[[i]]}];
   If [rule==4, info={g4[[i]],w4[[i]]}];
   If [numer, Return[N[info]], Return[Simplify[info]]];
];
```

To get information for the i^{th} point of the p^{th} rule, in which $1 \le i \le p$ and $p = 1,2,3,4$, call the module as `{ xi,wi }=LineGaussRuleInfo[{ p,numer },i]`. Here logical flag `numer` is `True` to get numerical (floating-point) information, or `False` to get exact information. `LineGaussRuleInfo` returns the sample point abcissa ξ_i in `xi` and the weight w_i in `wi`. For

[2] M. Abramowitz and I. A. Stegun (eds.) *Handbook of Mathematical Functions*, U S Department of Commerce, National Bureau of Standards, AM 55, 1964, Table 25.4.

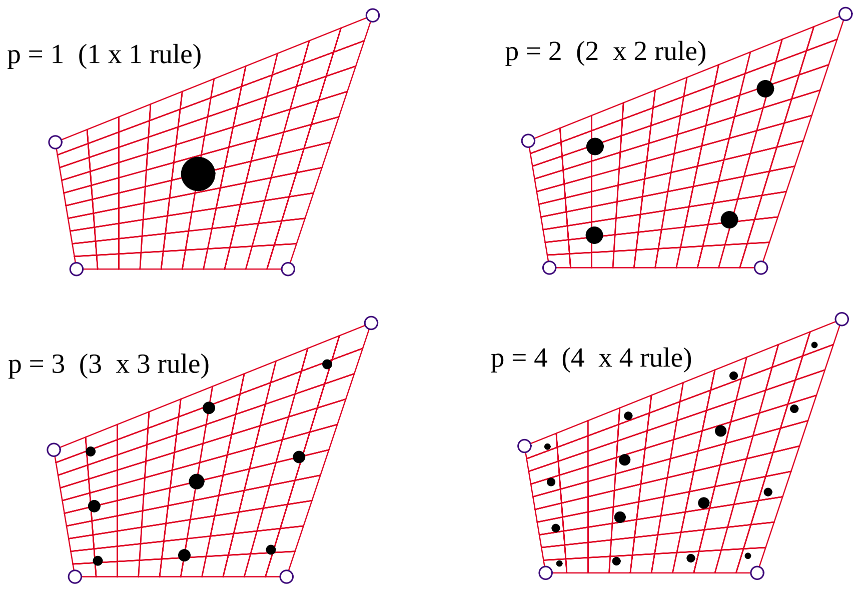


Figure 17.2. The first four two-dimensional Gauss product rules $p = 1, 2, 3, 4$ depicted over a straight-sided quadrilateral region. Gauss points are marked with black circles. The areas of these circles are proportional to the integration weights.

example, `LineGaussRuleInfo[{3,False},2]` returns `{0,8/9}`. If `p` is not 1 through 4, the module returns `Null`.

§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:

$$\int_{-1}^{1}\int_{-1}^{1} F(\xi,\eta)\,d\xi\,d\eta = \int_{-1}^{1} d\eta \int_{-1}^{1} F(\xi,\eta)\,d\xi. \tag{17.14}$$

Once this is done we can process numerically each integral in turn:

$$\int_{-1}^{1}\int_{-1}^{1} F(\xi,\eta)\,d\xi\,d\eta = \int_{-1}^{1} d\eta \int_{-1}^{1} F(\xi,\eta)\,d\xi \approx \sum_{i=1}^{p_1}\sum_{j=1}^{p_2} w_i w_j F(\xi_i,\eta_j). \tag{17.15}$$

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 17.2.

§17.3.4. Mathematica Implementation of 2D Gauss Rules

The following *Mathematica* module implements the two-dimensional product Gauss rules with 1 through 4 points in each direction. The number of points in each direction may be the same or different.

```
QuadGaussRuleInfo[{rule_,numer_},point_]:= Module[
 {xi,eta,p1,p2,i1,i2,w1,w2,k,info=Null},
  If [Length[rule] ==2, {p1,p2}=rule, p1=p2=rule];
  If [Length[point]==2, {i1,i2}=point,
     k=point; i2=Floor[(k-1)/p1]+1; i1=k-p1*(i2-1) ];
  {xi, w1}=  LineGaussRuleInfo[{p1,numer},i1];
  {eta,w2}=  LineGaussRuleInfo[{p2,numer},i2];
  info={{xi,eta},w1*w2};
  If [numer, Return[N[info]], Return[Simplify[info]]];
];
```

If the rule has the same number of points p in both directions the module is called in either of two ways:

```
{{xii,etaj},wij}=QuadGaussRuleInfo[{p,numer}, {i,j}]
{{xii,etaj},wij}=QuadGaussRuleInfo[{p,numer}, k ]
```
(17.16)

The first form is used to get information for point $\{i, j\}$ of the $p \times p$ rule, in which $1 \le i \le p$ and $1 \le j \le p$. The second form specifies that point by a "visiting counter" k that runs from 1 through p^2; if so $\{i, j\}$ are internally extracted[3] as `j=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:

```
{{xii,etaj},wij}=QuadGaussRuleInfo[{{p1,p2},numer}, {i,j}]
{{xii,etaj},wij}=QuadGaussRuleInfo[{{p1,p2},numer}, k ]
```
(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=Floor[(k-1)/p1]+1; i=k-p1*(i-1)`.

In all four forms flag `numer` 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.

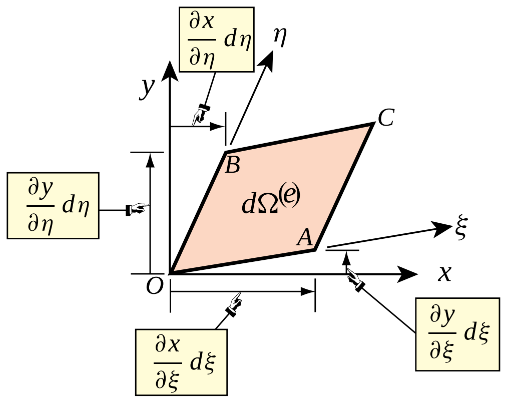
§17.4. THE STIFFNESS MATRIX

The stiffness matrix of a general plane stress element is given by the expression (14.23), which is reproduced here:

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\, \mathbf{B}^T \mathbf{E} \mathbf{B}\, d\Omega^{(e)} \tag{17.18}$$

Of the terms that appear in (17.18) the strain-displacement matrix **B** has been discussed previously. The thickness h, if variable, may be interpolated via the shape functions. The stress-strain matrix **E** is usually constant in elastic problems, but we could in principle interpolate it as appropriate should it vary over the element.

[3] Indices i and j are denoted by `i1` and `i2`, respectively, inside the module.



$\frac{\partial x}{\partial \eta}\,d\eta$
η
y
C
B
$\frac{\partial y}{\partial \eta}\,d\eta$
$d\Omega^{(e)}$
A
ξ
O
x
$\frac{\partial y}{\partial \xi}\,d\xi$
$\frac{\partial x}{\partial \xi}\,d\xi$

Figure 17.3. Geometric interpretation of the Jacobian-determinant formula.

To apply a Gauss product rule for the numerical integration of this equation we have to reduce it to the canonical form (17.15), *i.e.*

$$\mathbf{K}^{(e)} = \int_{-1}^{1}\int_{-1}^{1} \mathbf{F}(\xi,\eta)\,d\xi\,d\eta. \tag{17.19}$$

If ξ and η are the quadrilateral coordinates, everything in (17.19) 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)

$$d\Omega^{(e)} = dx\,dy = \det\mathbf{J}\;d\xi\,d\eta. \tag{17.20}$$

We therefore have

$$\mathbf{F}(\xi,\eta) = h\,\mathbf{B}^T\mathbf{E}\mathbf{B}\,\det\mathbf{J}. \tag{17.21}$$

This matrix function can be numerically integrated over the domain $-1 \le \xi \le +1$, $-1 \le \eta \le +1$ by an appropriate Gauss product rule.

REMARK 17.6

To geometrically justify the area transformation formula, consider the element of area OACB depicted in Figure 17.3. Then

$$dA = \vec{OB}\times\vec{OA} = \frac{\partial x}{\partial \xi}d\xi\,\frac{\partial y}{\partial \eta}d\eta - \frac{\partial x}{\partial \eta}d\eta\,\frac{\partial y}{\partial \xi}d\xi = \begin{vmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{vmatrix}\,d\xi\,d\eta = |\mathbf{J}|\,d\xi\,d\eta = \det\mathbf{J}\,d\xi\,d\eta. \tag{17.22}$$

§17.5. *EFFICIENT COMPUTATION OF ELEMENT STIFFNESS

This section is relevant only if the stiffness is coded in a low-level language such as Fortran or C. A Mathematica implementation for the 4-node quadrilateral is introduced in the Exercises and then elaborated in Chapter 23 for more complex elements.

Efficiency considerations in a low-level programming dictate that we take a close look at the matrix products that appear in (17.21). It is evident from a glance at (17.10) that the strain-displacement matrix **B** contains many zero entries. It is therefore of interest to form the matrix product $\mathbf{B}^T\mathbf{E}\mathbf{B}$ at each of the Gaussian points while avoiding multiplications by zero. This goal can be achieved as follows. To develop a simple expression in matrix form, suppose for the moment that the node displacement vector $\mathbf{u}^{(e)}$ is arranged as

$$[\,u_{x1} \quad u_{x2} \quad \ldots u_{xn} \quad u_{y1} \quad u_{y2} \quad \ldots \quad u_{yn}\,]^T \tag{17.23}$$

If so $\mathbf{B}$ can be expressed in partitioned-matrix form

$$\mathbf{B} = \begin{bmatrix} \mathbf{N}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{N}_y \\ \mathbf{N}_y & \mathbf{N}_x \end{bmatrix} \tag{17.24}$$

where $\mathbf{N}_x$ and $\mathbf{N}_y$ are row vectors of length n that contain the partial derivatives of the shape functions with respect to x and y, respectively. Then

$$\mathbf{E}\mathbf{B} = \begin{bmatrix} E_{11}\mathbf{N}_x + E_{13}\mathbf{N}_y & E_{12}\mathbf{N}_y + E_{13}\mathbf{N}_x \\ E_{12}\mathbf{N}_x + E_{23}\mathbf{N}_y & E_{22}\mathbf{N}_y + E_{23}\mathbf{N}_x \\ E_{13}\mathbf{N}_x + E_{33}\mathbf{N}_y & E_{23}\mathbf{N}_y + E_{33}\mathbf{N}_x \end{bmatrix}, \qquad \mathbf{B}^T\mathbf{E}\mathbf{B} = \begin{bmatrix} \mathbf{S}_{xx} & \mathbf{S}_{xy} \\ \mathbf{S}_{xy}^T & \mathbf{S}_{yy} \end{bmatrix}, \tag{17.25}$$

in which

$$\begin{aligned} \mathbf{S}_{xx} &= E_{11}\mathbf{N}_x^T\mathbf{N}_x + E_{13}(\mathbf{N}_x^T\mathbf{N}_y + \mathbf{N}_y^T\mathbf{N}_x) + E_{33}\mathbf{N}_y^T\mathbf{N}_y \\ \mathbf{S}_{xy} &= E_{13}\mathbf{N}_x^T\mathbf{N}_x + E_{12}\mathbf{N}_x^T\mathbf{N}_y + E_{33}\mathbf{N}_y^T\mathbf{N}_x + E_{23}\mathbf{N}_y^T\mathbf{N}_y \\ \mathbf{S}_{yy} &= E_{33}\mathbf{N}_x^T\mathbf{N}_x + E_{23}(\mathbf{N}_x^T\mathbf{N}_y + \mathbf{N}_y^T\mathbf{N}_x) + E_{22}\mathbf{N}_y^T\mathbf{N}_y \end{aligned} \tag{17.26}$$

In actual implementations the node displacement arrangement (17.23) is not used, but rather the x and y components are grouped node by node:

$$[\,u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad \ldots \quad u_{xn} \quad u_{yn}\,]^T \tag{17.27}$$

This change is easily implemented through appropriate array indexing.

As noted above, these considerations are important when programming in a low-level language such as C or Fortran. It should be avoided when programming in *Mathematica* or *Matlab* because in such high-level languages explicit indexing of arrays and lists is costly; that overhead in fact overwhelms any advantage gained from skipping zero operations.

§17.6. *INTEGRATION VARIANTS

Several deviations from the standard integration schemes described in the foregoing sections are found in the FEM literature. Two variations are described below and supplemented with motivation Exercises.

§17.6.1. *Weighted Integration

It is sometimes useful to form the element stiffness as a linear combination of stiffnesses produced by two different integration rules Such schemes are known as *weighted integration* methods. They are distinguished from the selective-integration schemes described in the next subsection in that the constitutive properties are not modified.

For the 4-node bilinear element weighted integration is done by combining the stiffnesses $\mathbf{K}^{(e)}_{1\times1}$ and $\mathbf{K}^{(e)}_{2\times2}$ produced by 1×1 and 2×2 Gauss product rules, respectively:

$$\mathbf{K}^{(e)}_{\beta} = (1-\beta)\mathbf{K}^{(e)}_{1\times1} + \beta\mathbf{K}^{(e)}_{2\times2}. \tag{17.28}$$

Here β is a scalar in the range [0, 1]. If $\beta = 0$ or $\beta = 1$ one recovers the element integrated by the 1×1 or 2×2 rule, respectively.[4]

The idea behind (17.28) is that $\mathbf{K}^{(e)}_{1\times1}$ is rank-deficient and too soft whereas $\mathbf{K}^{(e)}_{2\times2}$ 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.

§17.6.2. *Selective Integration

In the FEM literature the term *selective integration* is used to described a scheme for forming $\mathbf{K}^{(e)}$ as the sum of two or more matrices computed with different integration rules *and* different constitutive properties.[5] We consider here the case of a two-way decomposition. Split the plane stress constitutive matrix $\mathbf{E}$ into two:

$$\mathbf{E} = \mathbf{E}_{\rm I} + \mathbf{E}_{\rm II} \tag{17.29}$$

This is called a *stress-strain splitting*. Inserting (17.29) into (17.18) the expression of the stiffness matrix becomes

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{B}^T \mathbf{E}_{\rm I} \mathbf{B}\, d\Omega^{(e)} + \int_{\Omega^{(e)}} h\,\mathbf{B}^T \mathbf{E}_{\rm II} \mathbf{B}\, d\Omega^{(e)} = \mathbf{K}^{(e)}_{\rm I} + \mathbf{K}^{(e)}_{\rm II}. \tag{17.30}$$

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 (17.29) that improve the in-plane bending performance of rectangular elements.

[4] For programming the combination (17.28) may be regarded as a 5-point integration rule with weights $w_1 = 4(1-\beta)$ at the sample point at $\xi = \eta = 0$ and $w_i = \beta$ ($i = 2, 3, 4, 5$) at the four sample points at $\xi = \pm 1/\sqrt{3}$, $\eta = \pm 1/\sqrt{3}$.

[5] This technique is also called "selective reduced integration" to reflect the fact that one of the rules (the "reduced rule") underintegrates the element.

Homework Exercises for Chapter 17

Isoparametric Quadrilaterals

The *Mathematica* module `Quad4IsoPMembraneStiffness` 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_,mprop_,fprop_,options_]:=
  Module[{i,k,p=2,numer=False,Emat,th=1,h,qcoor,c,w,Nf,
   dNx,dNy,Jdet,B,Ke=Table[0,{8},{8}]},  Emat=mprop[[1]];
  If [Length[options]==2, {numer,p}=options, {numer}=options];
  If [Length[fprop]>0, th=fprop[[1]]];
  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]==0, h=th, h=th.Nf]; c=w*Jdet*h;
        B={ 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[B].(Emat.B)];
      ]; Return[Ke]
   ];
Quad4IsoPShapeFunDer[ncoor_,qcoor_]:= Module[
  {Nf,dNx,dNy,dNξ,dNη,i,J11,J12,J21,J22,Jdet,ξ,η,x1,x2,x3,x4,
   y1,y2,y3,y4,x,y},
  {ξ,η}=qcoor; {{x1,y1},{x2,y2},{x3,y3},{x4,y4}}=ncoor;
   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={x1,x2,x3,x4}; y={y1,y2,y3,y4};
   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 E17.1. Mathematica modules for Exercises 17.1–3.

The module makes use also of the Gauss integration modules `QuadGaussRuleInfo` and `LineGaussRuleInfo`. These are not shown in Figure E17.1 since they were listed in §17.3.2 and §17.3.4, but are included in the web-posted Notebook `Quad4Stiffness.nb`.[6] The arguments of the module are:

`ncoor`	Quadrilateral node coordinates arranged in two-dimensional list form: `{ { x1,y1 },{ x2,y2 },{ x3,y3 },{ x4,y4 } }`.
`mprop`	Material properties supplied as the list `{ Emat,rho,alpha }`. `Emat` is a two-dimensional list storing the 3 × 3 plane stress matrix of elastic moduli:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \tag{E17.1}$$

[6] 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.

If the material is isotropic with elastic modulus E and Poisson's ratio ν, this matrix becomes

$$\mathbf{E} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix} \tag{E17.2}$$

The other two items in `mprop` are not used in this module so zeros may be inserted as placeholders.

`fprop` Fabrication properties. The plate thickness specified as a four-entry list: { `h1,h2,h3,h4` }, a one-entry list: { `h` }, or an empty list: { }.

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 `h`. If an empty list appears the module assumes a uniform unit 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 forced to proceed in floating point arithmetic. For symbolic or exact arithmetic work set `numer` to `False`.[7]

`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.[8] If omitted `p` = 2 is assumed.

The module returns `Ke` as an 8×8 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4} \,]^T . \tag{E17.3}$$

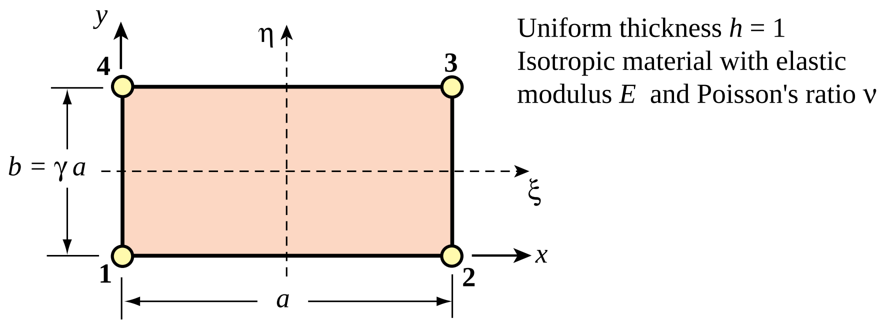

y
η
4
3
Uniform thickness h = 1
Isotropic material with elastic
modulus E and Poisson's ratio ν
b = γ a
ξ
x
1
2
a

Figure 17.2. Element for Exercises 17.1 to 17.3.

For the following three exercises we consider the specialization of the general 4-node bilinear quadrilateral (16.12)-(16.13) 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 (E17.2). The stiffness matrix of this element can be expressed in closed form.[9] For convenience define $\gamma = b/a$, $\psi_1 = (1+\nu)\gamma$, $\psi_2 = (1-3\nu)\gamma$,

[7] 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 complex.

[8] The rank of an element stiffness is discussed in Chapter 19.

[9] This closed form can be obtained by either exact integration, or numerical integration with a 2×2 or higher Gauss rule.

$\psi_3 = 1-\nu+2\gamma^2$, $\psi_4 = 2+(1-\nu)\gamma^2$, $\psi_5 = 1-\nu-4\gamma^2$, $\psi_6 = 1-\nu-\gamma^2$, $\psi_7 = 4-(1-\nu)\gamma^2$ and $\psi_8 = 1-(1-\nu)\gamma^2$. Then

$$\mathbf{K}^{(e)} = \frac{Eh}{24\gamma(1-\nu^2)} \begin{bmatrix} 4\psi_3 & -3\psi_1 & 2\psi_5 & 3\psi_2 & -4\psi_6 & -3\psi_2 & -2\psi_3 & 3\psi_1 \\ & 4\psi_4 & -3\psi_2 & 4\psi_8 & 3\psi_2 & -2\psi_7 & 3\psi_1 & -2\psi_4 \\ & & 4\psi_3 & 3\psi_1 & -2\psi_3 & -3\psi_1 & -4\psi_6 & 3\psi_2 \\ & & & 4\psi_4 & -3\psi_1 & -2\psi_4 & -3\psi_2 & -2\psi_7 \\ & & & & 4\psi_3 & 3\psi_1 & 2\psi_5 & -3\psi_2 \\ & & & & & 4\psi_4 & 3\psi_2 & 4\psi_8 \\ & & & & & & 4\psi_3 & -3\psi_1 \\ \text{symm} & & & & & & & 4\psi_4 \end{bmatrix}. \qquad \text{(E17.4)}$$

EXERCISE 17.1

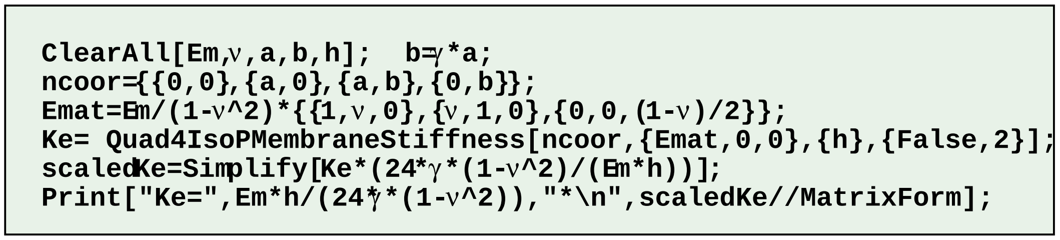
[C:15] Exercise the *Mathematica* module of Figure E17.1 with the following script:

```
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,0,0},{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 discussed in detail in Chapter 19.) Question: why does the stiffness matrix stays exactly the same for $p \geq 2$? Hint: take a look at the integrand $h\mathbf{B}^T\mathbf{E}\mathbf{B}\,J$ — for a *rectangular geometry* is it a polynomial in ξ and η or a rational function?

EXERCISE 17.2

[C:20] Check the rectangular element stiffness closed form given in (E17.4). This may be done by hand (takes a few days) or running the following script that calls the *Mathematica* module of Figure E17.1:



```
ClearAll[Em,ν,a,b,h];  b=γ*a;
ncoor={{0,0},{a,0},{a,b},{0,b}};
Emat=Em/(1-ν^2)*{{1,ν,0},{ν,1,0},{0,0,(1-ν)/2}};
Ke= Quad4IsoPMembraneStiffness[ncoor,{Emat,0,0},{h},{False,2}];
scaledKe=Simplify[Ke*(24*γ*(1-ν^2)/(Em*h))];
Print["Ke=",Em*h/(24*γ*(1-ν^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 (E17.4) by printout inspection and report any typos to instructor.

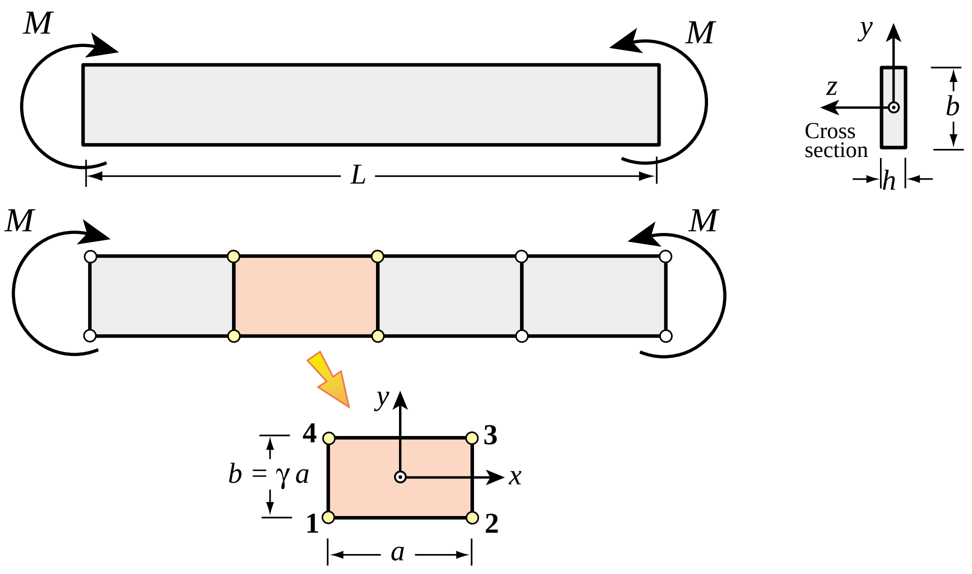


Figure E17.4. Pure bending of Bernoulli-Euler plane beam of thin rectangular cross section, for Exercises 17.3–7. The beam is modeled by one layer of 4-node iso-P bilinear quadrilaterals through its height.

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 $\kappa = M/(E I_z)$, in which $I_z = \frac{1}{12} h b^3$ 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

$$u_x = -\kappa x y, \quad u_y = \tfrac{1}{2}\kappa (x^2 + \nu y^2), \tag{E17.5}$$

where κ is the deformed beam curvature M/EI. The stiffness equations of the typical rectangular element are given by the close form expression (E17.4).

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.

(a) Check that (E17.5), 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 (E17.2) 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.

(b) Compute the strain energy $U_{\rm quad} = \frac{1}{2}(\mathbf{u}_{\rm beam})^T \mathbf{K}^{(e)} \mathbf{u}_{\rm beam}$ absorbed by the 4-node element under nodal displacements $\mathbf{u}_{\rm beam}$ constructed by evaluating (E17.5) at the nodes 1,2,3,4. To simplify this calculation, it is convenient to decompose that vector as follows:

$$\begin{aligned}\mathbf{u}_{\rm beam} = \mathbf{u}^x_{\rm beam} + \mathbf{u}^y_{\rm beam} &= \tfrac{1}{4}\kappa ab\,[\,-1 \quad 0 \quad 1 \quad 0 \quad -1 \quad 0 \quad 1 \quad 0\,]^T \\ &\quad + \tfrac{1}{8}\kappa(a^2+\nu b^2)\,[\,0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1\,]^T\end{aligned} \tag{E17.6}$$

Explain why $\mathbf{K}^{(e)}\mathbf{u}^y_{\rm beam}$ must vanish and consequently

$$U_{\rm quad} = \tfrac{1}{2}(\mathbf{u}^x_{\rm beam})^T \mathbf{K}^{(e)} \mathbf{u}^x_{\rm beam}. \tag{E17.7}$$

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 $\mathbf{u}^x_{\rm beam}$ 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 $\kappa = M/(EI_z) = 12M/(Eha^3\gamma^3)$.

(c) 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 $U_{beam} = \frac{1}{2}M\kappa a = M^2 a/(2EI_z) = 6M^2/(Eha^2\gamma^3)$. 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:

$$r = r(\gamma,\nu) = \frac{2\gamma^2(1-\nu^2)}{1+2\gamma^2-\nu}. \tag{E17.8}$$

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 $0 \le \nu \le \frac{1}{2}$. Moreover if $b << a$, $r << 1$; for example if $a = 10b$ and $\nu = 0$, $r \approx 1/50$ and the 2D model gives only about 2% of the correct solution.[10] 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.[11]

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 $\mathbf{K}^{(e)}_{1\times1}$ and $\mathbf{K}^{(e)}_{2\times2}$ denote the element stiffnesses produced by 1×1 and 2×2 Gauss product rules, respectively. Take

$$\mathbf{K}^{(e)}_{\beta} = (1-\beta)\mathbf{K}^{(e)}_{1\times1} + \beta\mathbf{K}^{(e)}_{2\times2} \tag{E17.9}$$

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 $\mathbf{K}^{(e)}_{\beta}$ into the test introduced there verify that

$$r = \frac{2\gamma^2(1-\nu^2)}{\beta(1+2\gamma^2-\nu)}. \tag{E17.10}$$

[10] This phenomenon is referred to in the FEM literature as *shear locking*, because overstiffness 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.

[11] Note that even if we make $a \to 0$ and $\gamma = b/a \to \infty$ by taking an infinite number of rectangular elements along x, the energy ratio r remains less than one if $\nu > 0$ since $r \to 1-\nu^2$. Thus the 2D model would not generally converge to the correct solution if we keep one layer through the height.

Whence show that if

$$\beta = \frac{2\gamma^2(1-\nu^2)}{1+2\gamma^2-\nu}, \tag{E17.11}$$

then $r \equiv 1$ for all $\{\gamma, \nu\}$ and the element is x-bending exact. A deficiency of 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.[12] Consider the stress-strain splitting

$$\mathbf{E} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix} \alpha & \beta & 0 \\ \beta & \alpha & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} + \frac{E}{1-\nu^2}\begin{bmatrix} 1-\alpha & \nu-\beta & 0 \\ \nu-\beta & 1-\alpha & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{E}_{\rm I} + \mathbf{E}_{\rm II}, \tag{E17.12}$$

where α and β are scalars. Show that if

$$\chi = \sqrt{\frac{1-\nu^2}{3(1-\alpha)}} \tag{E17.13}$$

the resulting element stiffness $\mathbf{K}_{\rm I}^{(e)} + \mathbf{K}_{\rm II}^{(e)}$ is bending exact for any $\{\alpha, \beta\}$. As a corollary show that that if $\alpha = \nu^2$, which corresponds to the splitting

$$\mathbf{E} = \frac{E}{1-\nu^2}\begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} = \frac{E}{1-\nu^2}\begin{bmatrix} \nu^2 & \beta & 0 \\ \beta & \nu^2 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} + \frac{E}{1-\nu^2}\begin{bmatrix} 1-\nu^2 & \nu-\beta & 0 \\ \nu-\beta & 1-\nu^2 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{E}_{\rm I} + \mathbf{E}_{\rm II}, \tag{E17.14}$$

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.[13] Show that if one selects the stress-strain splitting (E17.12) and

$$\chi = \sqrt{\frac{2(1-\nu^2)}{3(1-\alpha)}} \tag{E17.15}$$

the resulting element stiffness $\mathbf{K}_{\rm I}^{(e)} + \mathbf{K}_{\rm II}^{(e)}$ is bending exact for any $\{\alpha, \beta\}$. Discuss which choices of α reduce χ to $1/\sqrt{3}$ and $\sqrt{2/3}$, respectively.

[12] For a rectangular geometry these sample points lie on the diagonals. In the case of the standard 2-point Gauss product rule $\chi = 1/\sqrt{3}$.

[13] This is called a 4-point median rule, since the four points are located on the quadrilateral medians.

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:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \tag{E17.16}$$

The rules for the selective integration scheme are as described in Exercise 17.5. The appropriate stress-strain splitting is

$$\mathbf{E} = \mathbf{E}_{\rm I} + \mathbf{E}_{\rm II} = \begin{bmatrix} E_{11}\,\alpha_1 & E_{12}\beta & E_{13} \\ E_{12}\,\beta & E_{22}\,\alpha_2 & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} + \begin{bmatrix} E_{11}(1-\alpha_1) & E_{12}(1-\beta) & 0 \\ E_{12}(1-\beta) & E_{22}(1-\alpha_2) & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{E17.17}$$

in which β is arbitrary and

$$\begin{gathered} 1-\alpha_1 = \frac{|\mathbf{E}|}{3\chi^2 E_{11}(E_{22}E_{33}-E_{23}^2)} = \frac{1}{3\chi^2 C_{11}}, \qquad 1-\alpha_2 = \frac{|\mathbf{E}|}{3\chi^2 E_{22}(E_{11}E_{33}-E_{13}^2)} = \frac{1}{3\chi^2 C_{22}}, \\ |\mathbf{E}| = \det(\mathbf{E}) = E_{11}E_{22}E_{33} + 2E_{12}E_{13}E_{23} - E_{11}E_{23}^2 - E_{22}E_{13}^2 - E_{33}E_{12}^2, \\ C_{11} = E_{11}(E_{22}E_{33}-E_{13}^2)/|\mathbf{E}|, \quad C_{22} = E_{22}(E_{11}E_{33}-E_{13}^2)/|\mathbf{E}|. \end{gathered} \tag{E17.18}$$

Show that the resulting rectangular element is bending exact for any $\mathbf{E}$ and $\chi \neq 0$. (In practice one would select $\chi = 1/\sqrt{3}$.)

18 Shape Function Magic

TABLE OF CONTENTS

§18.1. REQUIREMENTS

This Chapter explains, through a series of examples, how two-dimensional isoparametric shape functions can be directly constructed by geometric considerations. 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 along 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 on any element boundary that includes node i.

(D) *Completeness condition*. The interpolation may represent exactly any displacement field which is a linear polynomial in x and y; in particular, a constant value.

Requirement (A) follows directly from 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 the isoparametric shape functions 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 along a side common to two elements must uniquely depend only on its nodal values on that side.

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), it is sufficient to check that the sum of shape functions is identically unity in view of the discussion in §16.6.

§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 natural consequence of the definition of natural coordinates. In general a shape function can be expressed as the product of m factors:

$$N_i^{(e)} = c_i\, L_1\, L_2\, \ldots\, L_m, \tag{18.1}$$

where

$$L_j = 0, \qquad j = 1, \ldots m \tag{18.2}$$

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.

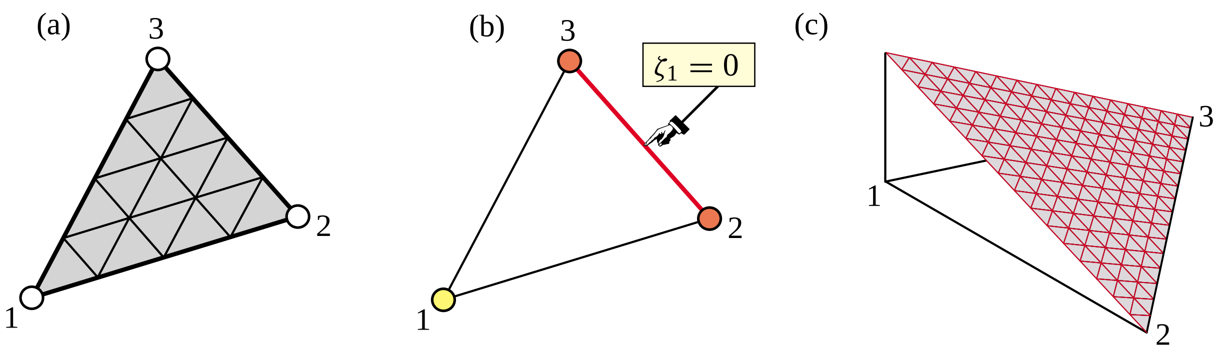


Figure 18.1. The three-node linear triangle: (a) element geometry; (b) equation of side opposite corner 1; (c) perspective view of the shape function $N_1 = \zeta_1$.

For two-dimensional isoparametric elements, the ingredients in (18.1) are chosen according to the following rules.

1. 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. Primary choices are the element sides and medians. The examples below illustrate how this is done.
2. Set coefficient c_i so that $N_i^{(e)}$ has the value 1 at the i^{th} node.
3. Check the polynomial order variation over each interelement boundary that contains node i. If this order is n, there must be exactly $n+1$ nodes on the boundary for the compatibility condition to hold.
4. Perform completeness checks.

Specific two-dimensional examples in the following subsections show these rules in action. Essentially the same technique is applicable to one- and three-dimensional elements.

§18.3. TRIANGULAR ELEMENT SHAPE FUNCTIONS

We first illustrate 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, which 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 $L_{j\text{-}k} = \zeta_i = 0$, where j and k are the cyclic permutations of i. Here symbol $L_{j\text{-}k}$ 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

$$N_i \overset{\text{guess}}{=} c_i L_i. \qquad (18.3)$$

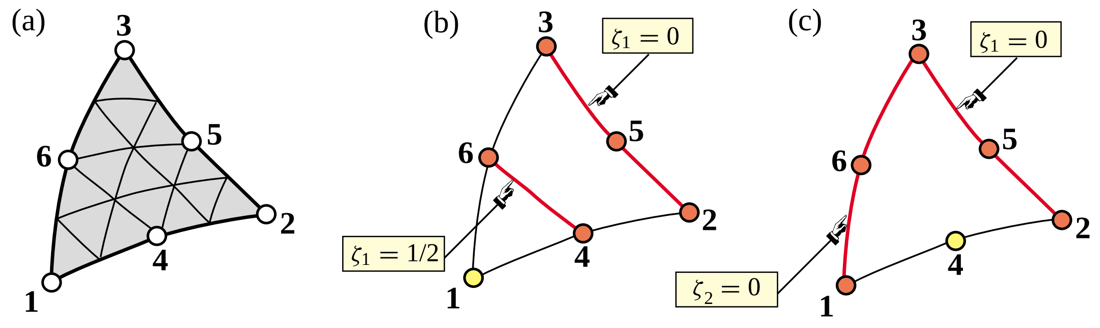


Figure 18.2. The six-node quadratic triangle: (a) element geometry; (b) sides whose product yields $N_1^{(e)}$; (c) Sides whose product yields $N_4^{(e)}$.

This satisfies conditions (A) and (B) except the unit value at node i; this follows if $c_i = 1$. The compatibility condition (C) follows from the fact that the variation of ζ_i along the sides meeting at node i is linear and that there are two nodes per side; cf. §15.4.2. Completeness is automatic since $\zeta_1 + \zeta_1 + \zeta_3 = 1$. Figure 18.1(c) displays the shape function $N_1 = \zeta_1$ 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

$$N_1^{(e)} \overset{\text{guess}}{=} c_1 \, L_{2\text{-}3} \, L_{4\text{-}6}, \tag{18.4}$$

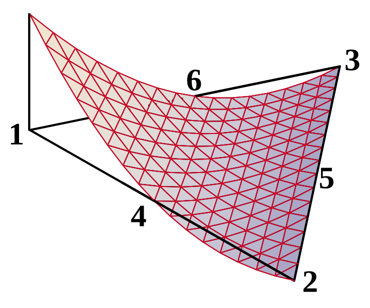
Why is (18.4) expected to work? Clearly $N_1^{(e)}$ will vanish along both sides 2-3 and 4-6. This will make the function zero at nodes 2 through 6, as is obvious upon inspection of Figure 18.2(b), while being nonzero at node 1. This value can be adjusted to be unity if c_1 is appropriately chosen. Furthermore, $N_1^{(e)}$ will vanish along the element boundary 2-3 that does not contain node 1, thus satisfying rule (B) of §18.1. The equations of the lines that appear in (18.4) are

$$L_{2\text{-}3}: \quad \zeta_1 = 0, \qquad\qquad L_{4\text{-}6}: \quad \zeta_1 - \tfrac{1}{2} = 0. \tag{18.5}$$

Replacing into (18.3) we get

$$N_1^{(e)} = c_1 \, \zeta_1 (\zeta_1 - \tfrac{1}{2}), \tag{18.6}$$

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: $N_1^{(e)}(1, 0, 0) = c_1 \times 1 \times \frac{1}{2} = 1$, from



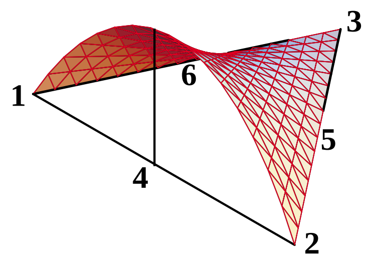


Figure 18.3. Perspective view of shape functions N_1 and N_4 for the quadratic triangle. The plot is done over a straight side triangle for programming simplicity.

which $c_1 = 2$ and finally

$$N_1^{(e)} = 2\zeta_1(\zeta_1 - \tfrac{1}{2}) = \zeta_1(2\zeta_1 - 1), \tag{18.7}$$

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

$$N_4^{(e)} \overset{\text{guess}}{=} c_4 \, L_{2\text{-}3} \, L_{1\text{-}3} \tag{18.8}$$

Evidently (18.8) satisfies requirements (A) and (B) if c_4 is appropriately normalized. The equation of sides $L_{2\text{-}3}$ and $L_{1\text{-}3}$ 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 $c_4 = 4$ and finally

$$N_4^{(e)} = 4\zeta_1\zeta_2 \tag{18.9}$$

as listed in §16.5.2. Figure 18.3 shows a perspective view. 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 checks that the sum of the N_i is unity; thus the element is complete.

§18.4. QUADRILATERAL ELEMENT SHAPE FUNCTIONS

Three quadrilateral elements which are common in the applications are used as examples to illustrate the construction of their shape functions.

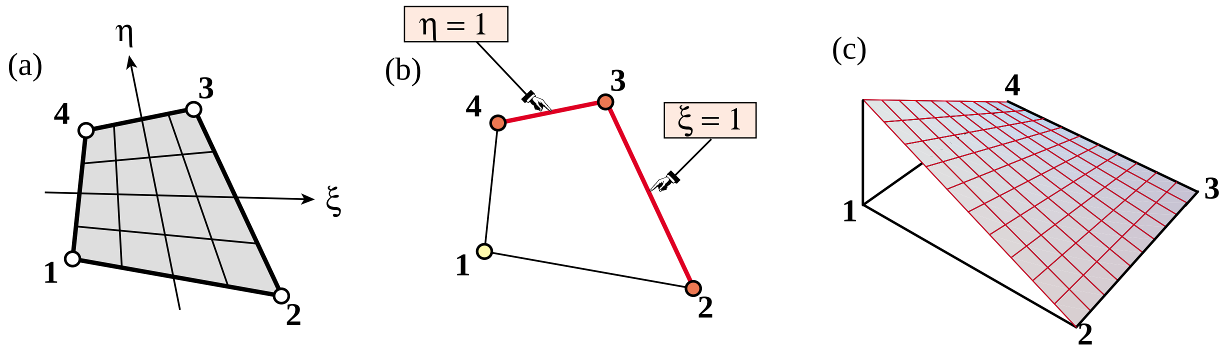


Figure 18.4. The four-node bilinear quadrilateral: (a) element geometry; (b) equations of sides opposite corner 1; (c) perspective view of the shape function $N_1^{(e)}$.

§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(b) suggests trying

$$N_1^{(e)} \overset{\text{guess}}{=} c_1\, L_{2\text{-}3}\, L_{3\text{-}4} \tag{18.10}$$

This clearly vanishes over nodes 2, 3 and 4, and can be normalized to unity at node 1 by adjusting c_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

$$N_1^{(e)}(\xi,\eta) = c_1(\xi-1)(\eta-1) = c_1(1-\xi)(1-\eta). \tag{18.11}$$

To find coefficient c_1, evaluate this function at node 1, the natural coordinates of which are $\xi = \eta = -1$:

$$N_1^{(e)}(-1,-1) = c_1 \times 2 \times 2 = 4c_1 = 1 \tag{18.12}$$

Hence $c_1 = \frac{1}{4}$ and the shape function is

$$N_1^{(e)} = \tfrac{1}{4}(1-\xi)(1-\eta), \tag{18.13}$$

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

$$N_i^{(e)} = \tfrac{1}{4}(1+\xi_i\,\xi)(1+\eta_i\,\eta). \tag{18.14}$$

The continuity check proceeds as follows, using node 1 as example. Node 1 belongs to interelement boundaries 1–2 and 1–3. On 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). On 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.

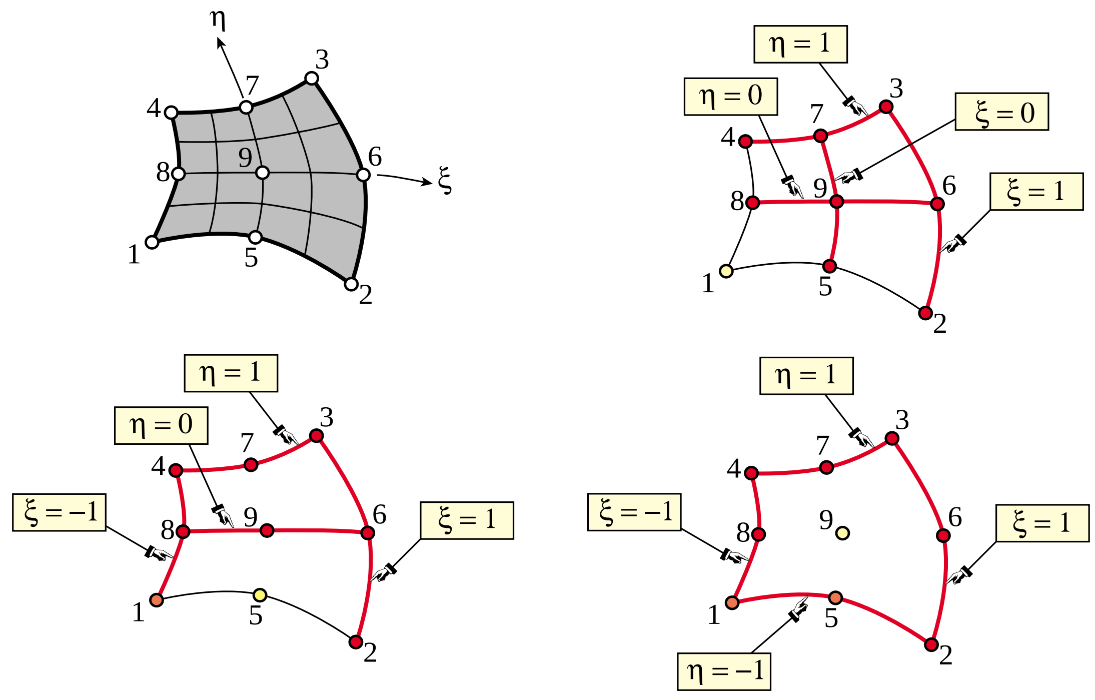
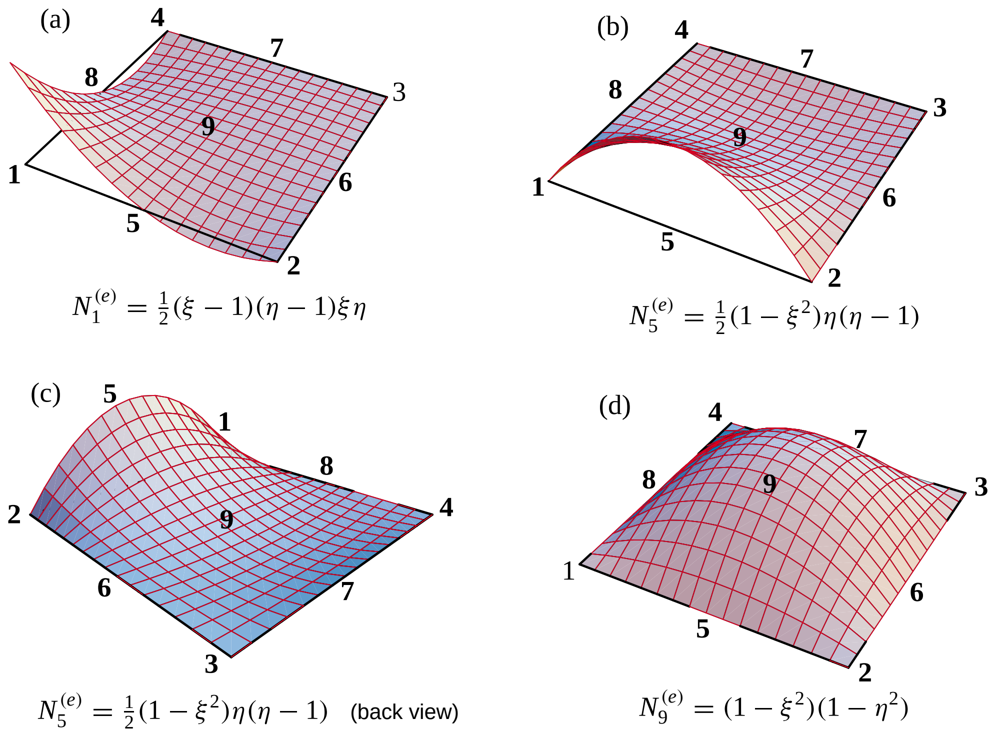


Figure 18.5. The nine-node biquadratic quadrilateral: (a) element geometry; (b,c,d) lines whose product make up the shape functions $N_1^{(e)}$, $N_5^{(e)}$ and $N_9^{(e)}$, respectively.



(a)
(b)
(c)
(d)
1
2
3
4
5
6
7
8
9
$N_1^{(e)} = \frac{1}{2}(\xi-1)(\eta-1)\xi\eta$
$N_5^{(e)} = \frac{1}{2}(1-\xi^2)\eta(\eta-1)$
$N_5^{(e)} = \frac{1}{2}(1-\xi^2)\eta(\eta-1)$ (back view)
$N_9^{(e)} = (1-\xi^2)(1-\eta^2)$

Figure 18.6. Perspective view of the shape functions for nodes 1, 5 and 9 of the nine-node biquadratic quadrilateral.

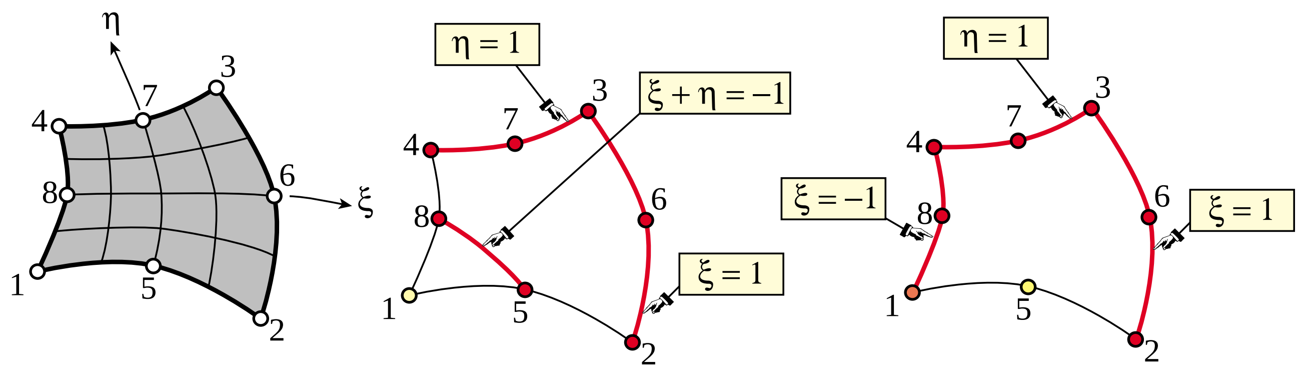


Figure 18.7. The eight-node serendipity quadrilateral: (a) element geometry; (b,c) lines whose product make up the shape functions $N_1^{(e)}$ and $N_5^{(e)}$, respectively.

§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:

$$N_1^{(e)} = c_1 L_{2\text{-}3} L_{3\text{-}4} L_{5\text{-}7} L_{6\text{-}8} = c_1(\xi - 1)(\eta - 1)\xi\eta \tag{18.15}$$

$$N_5^{(e)} = c_5 L_{2\text{-}3} L_{1\text{-}4} L_{6\text{-}8} L_{3\text{-}4} = c_5\,(\xi - 1)(\xi + 1)\eta(\eta - 1) = c_5\,(1 - \xi^2)\eta(1 - \eta) \tag{18.16}$$

$$N_9^{(e)} = c_9\, L_{1\text{-}2} L_{2\text{-}3} L_{3\text{-}4} L_{4\text{-}1} = c_9\,(\xi - 1)(\eta - 1)(\xi + 1)(\eta + 1) = c_9\,(1 - \xi^2)(1 - \eta^2) \tag{18.17}$$

Imposing the normalization conditions we find

$$c_1 = \tfrac{1}{4}, \qquad c_5 = -\tfrac{1}{2}, \qquad c_9 = 1, \tag{18.18}$$

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 over any interelement boundary is two and there are three nodes on each boundary. 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 was widely used in commercial codes during the 70s and 80s but it is gradually being phased out in favor of the 9-node quadrilateral.

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.

$$N_1^{(e)} = c_1 L_{2\text{-}3} L_{3\text{-}4} L_{5\text{-}8} = c_1(\xi-1)(\eta-1)(1+\xi+\eta) = c_1(1-\xi)(1-\eta)(1+\xi+\eta), \quad (18.19)$$

$$N_5^{(e)} = c_5 L_{2\text{-}3} L_{3\text{-}4} L_{4\text{-}1} = c_5\,(\xi-1)(\xi+1)(\eta-1) = c_5\,(1-\xi^2)(1-\eta). \quad (18.20)$$

Imposing the normalization conditions we find

$$c_1 = -\tfrac{1}{4}, \qquad c_5 = \tfrac{1}{2} \quad (18.21)$$

The other shape functions follow by appropriate permutation of nodal indices.

The interelement continuity and completeness verification is similar to that carried out for the nine-node element, and is left as an exercise.

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];
```

§18.5. *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.

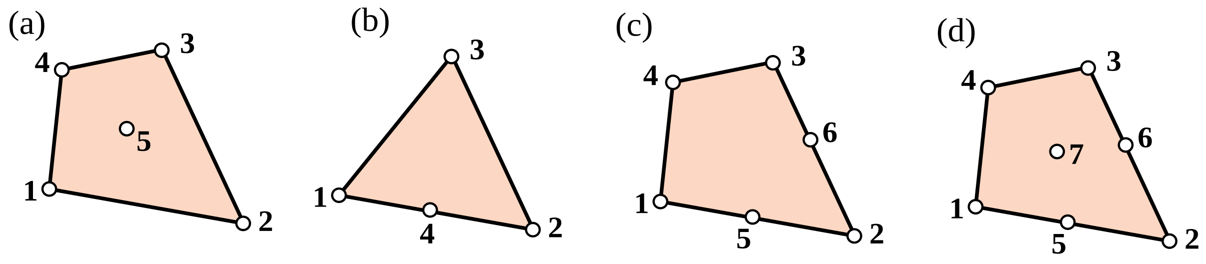


Figure 18.8. Node configurations for which the magic wand (18.1-18.2) does not work.

§18.6. DOES THE MAGIC WAND ALWAYS WORK?

The recipe (18.1-18.2) is not foolproof. It fails for certain node configurations although it is a reasonable way to start. The method runs into difficulties, for example, in the problem posed in Exercise 18.6, which deals with the 5-node quadrilateral depicted in Figure 18.8(a). If for corner 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 $N_1 = -(1-\xi)(1-\eta)(\xi+\eta)/8$. This satisfies the interpolation condition (A) and the local support condition (B). However, it violates the compatibility condition (C) along sides 1-2 and 4-1, because it varies quadratically over them with only two nodes to define its variation.

A more general technique relies on a *correction approach*, which uses 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

$$N_i^{(e)} = c_i\, L_1^c\, L_2^c\, \ldots\, L_m^c + d_i\, L_1^d\, L_2^d\, \ldots\, L_n^d, \tag{18.22}$$

Here two normalization coefficients: c_i and d_i, appear. Exercise 18.6 gives an example of this correction technique for the element of Figure 18.8(a).

The recommended procedure for a new element is to start with the one-product form (18.1). If more than one pattern of lines is possible, all of them are systematically tried. If none works in the sense of satisfying conditions (A) through (D), proceed to the next level combination such as (18.22).

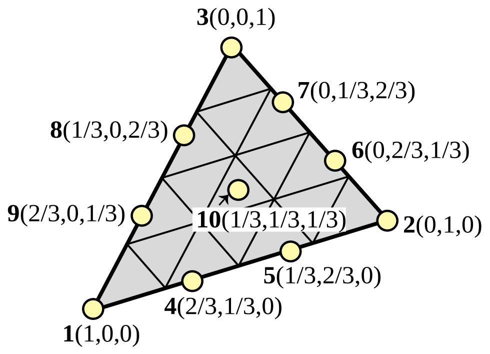
This correction technique is particularly useful for *transition elements*, which have corner nodes but midnodes only over certain sides. Three examples are provided in Figure 18.8(b,c,d). Correct shape functions can be derived with one, two and three corrections, respectively. The general method is more easily presented through the framework of *hierarchical shape functions*, which is covered in more advanced FEM courses.

Homework Exercises for Chapter 18

Shape Function Magic

EXERCISE 18.1

[A/C:15+15] The complete cubic triangle for plane stress has 10 nodes located as shown in Figure E18.1, with their triangular coordinates listed in parentheses.



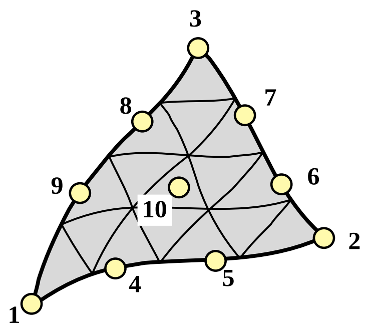


Figure E18.1. Ten-node cubic triangle for Exercise 18.1. The left picture displays the superparametric element whereas the right picture shows the general isoparametric version with curved sides.

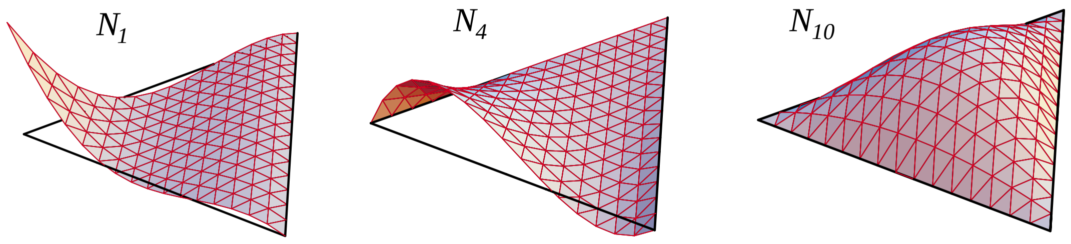


Figure E18.2. Perspective plots of the shape functions N_1, N_4 and N_{10} for the 10-node cubic triangle.

(a) Construct the cubic shape functions $N_1^{(e)}$, $N_4^{(e)}$ and $N_{10}^{(e)}$ for nodes 1, 4, and 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.

(b) Construct the remaining 7 shape functions by appropriate node number permutations, and verify that the sum of the 10 functions is identically one.

EXERCISE 18.2

[A:20] 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.

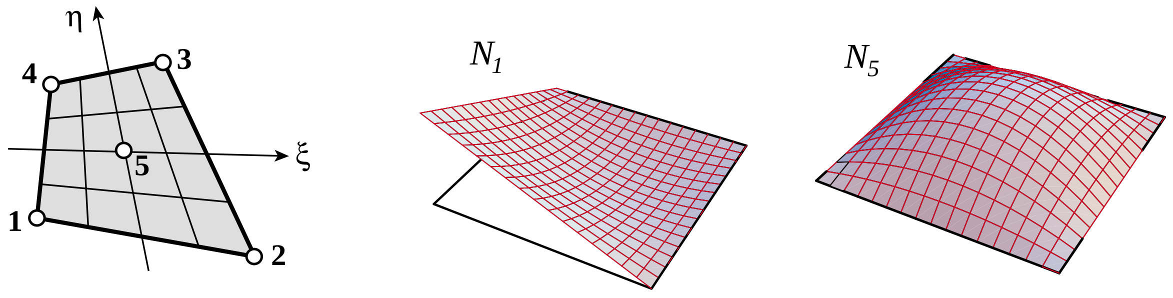


Figure E18.3. Five node quadrilateral element for Exercise 18.6.

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 derived in §18.4.3 satisfy the interelement compatibility condition (C) stated in §18.1. Check whether 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.

EXERCISE 18.6

[A:20] 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 : $\mu = +1$

[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 shown in Figure E18.3 must vanish at node 5, that is, at $\xi = \eta = 0$. On the other hand, the $N_1^{(e)}$ of Figure 18.4 takes the value $\frac{1}{4}$ there.

19 FEM Convergence Requirements

TABLE OF CONTENTS

§19.1. OVERVIEW

Chapters 12 through 18 have listed, in piecemeal fashion, various requirements for shape functions of isoparametric elements. These requirements are motivated by *convergence*: as the finite element mesh is refined, the 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 must 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 satify 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*. The famous Lax-Wendroff theorem[2] says that consistency and stability imply convergence.

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. Nonetheless, the satisfaction of the three criteria guarantees convergence and may therefore be regarded as a safe choice.

§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, that is a different and far more difficult problem.

[2] Proven originally for classical finite difference discretizations.

EXAMPLE 19.1

In the bar problem discussed in Chapter 12,

$$\Pi[u] = \int_0^L \left(\tfrac{1}{2}\, u' E A u' - q u \right) dx. \tag{19.1}$$

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 13,

$$\Pi[v] = \int_0^L \left(\tfrac{1}{2}\, v'' E I v'' - q v \right) dx. \tag{19.2}$$

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

$$u_x = \alpha_0 + \alpha_1 x + \alpha_2 y, \qquad u_y = \alpha_0 + \alpha_1 x + \alpha_2 y \tag{19.3}$$

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. The 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

$$v = \alpha_0 + \alpha_1 x + \alpha_2 x^2 \tag{19.4}$$

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.

§19.3.2. Compatibility

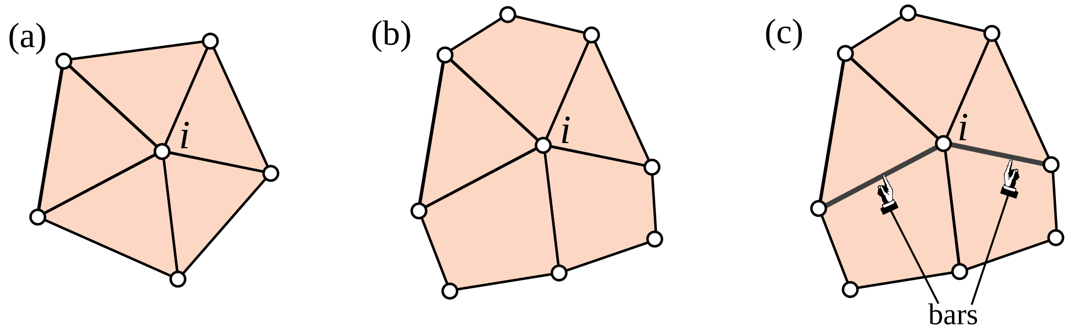


Figure 19.1. An element patch is the set of all elements attached to a patch node, herein labeled i. (a) illustrates a patch of triangles; (b) a mixture of triangles and quadrilaterals; (c) a mixture of triangles, quadrilaterals, and bars.

To state this requirement succintly, 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. This node is 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.

With the benefit 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.

In the common case $m = 1$, 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]

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.

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 19.1

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 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.

§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 be 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.

[3] The FEM literature is 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.

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 x 2
8-node quadrilateral	8	16	13	5	3 x 3
9-node quadrilateral	9	18	15	5	3 x 3
16-node quadrilateral	16	32	29	10	4 x 4

* These triangle integration rules are introduced in §24.2.

§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,

$$d = (n_F - n_R) - r \tag{19.5}$$

is called the rank deficiency.

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

$$r = \min(n_F - n_R, n_E\, n_G) \tag{19.6}$$

To attain rank sufficiency, $n_E n_G$ must equal or exceed $n_F - n_R$:

$$\boxed{n_E\, n_G \geq n_F - n_R} \tag{19.7}$$

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 Chapter 14. Also $n_R = 3$. Consequently $r = \min(n_F - 3, 3n_G)$ and $3n_G \geq n_F - 3$.

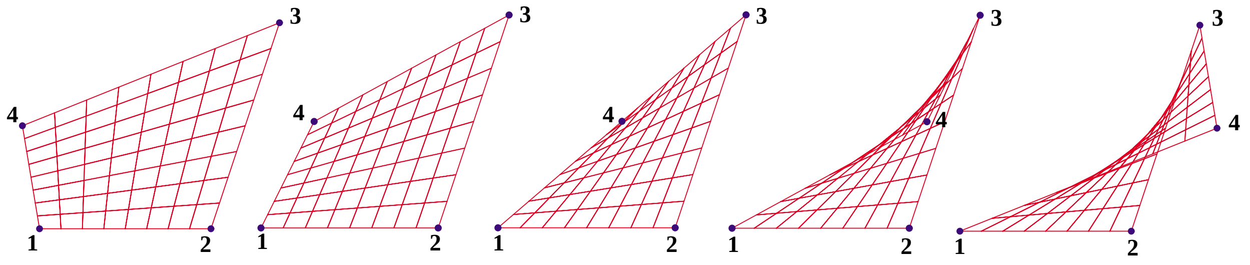


Figure 19.2. Effect of displacing node 4 of the four-node bilinear quadrilateral shown on the leftmost picture, to the right.

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 must be such that the determinant $J = \det \mathbf{J}$ of the Jacobian matrix defined[4] in §17.2, is positive everywhere. As illustrated in Equation (17.20), J characterizes the local metric of the element natural coordinates.

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 19.2 for a 4-node quadrilateral in which node 4 is gradually moved to the right. The quadrilateral 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 contradicts the (erroneous) advise of some FE books, which state that quadrilaterals can be reduced to triangles as special cases, thereby rendering triangular elements unnecessary.

[4] This definition applies to quadrilateral elements. The Jacobian determinant of an arbitrary triangular element is defined in §24.2.

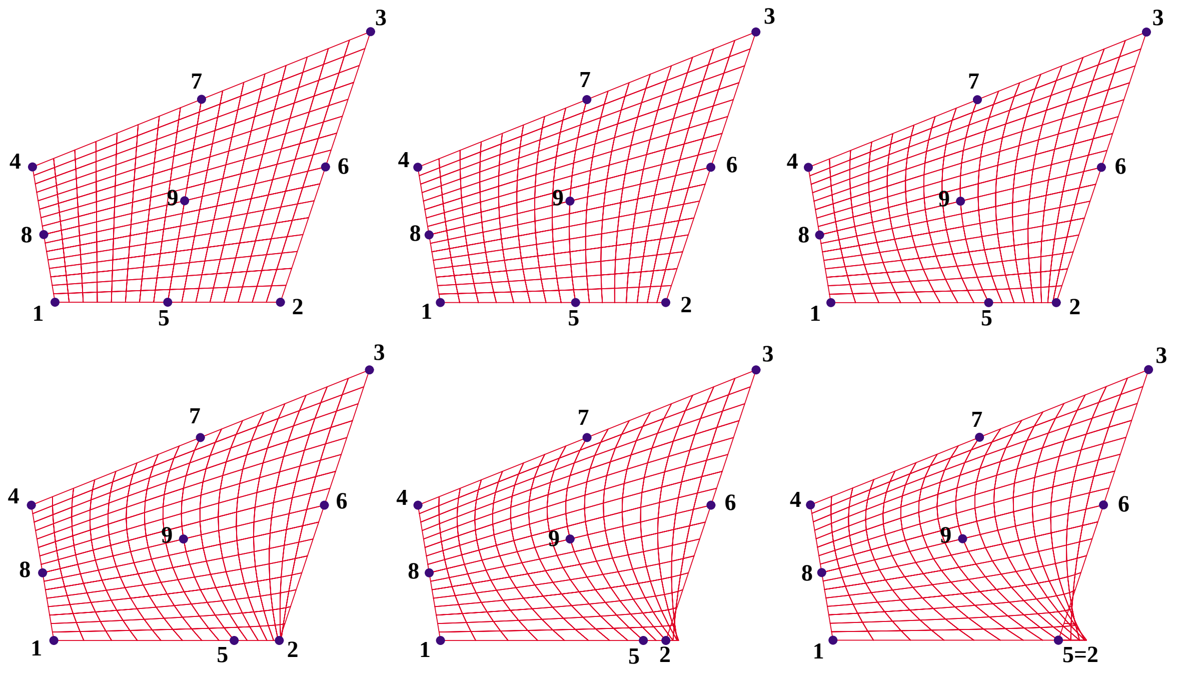


Figure 19.3. Effect of moving midpoint 5 of a 9-node biquadratic quadrilateral tangentially toward corner 2.

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 19.3 and 19.4.

Figure 19.3 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.

Displacing midside nodes normally to the sides is comparatively more forgiving, as illustrated in Figure 19.4. 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" without the metric breaking down.

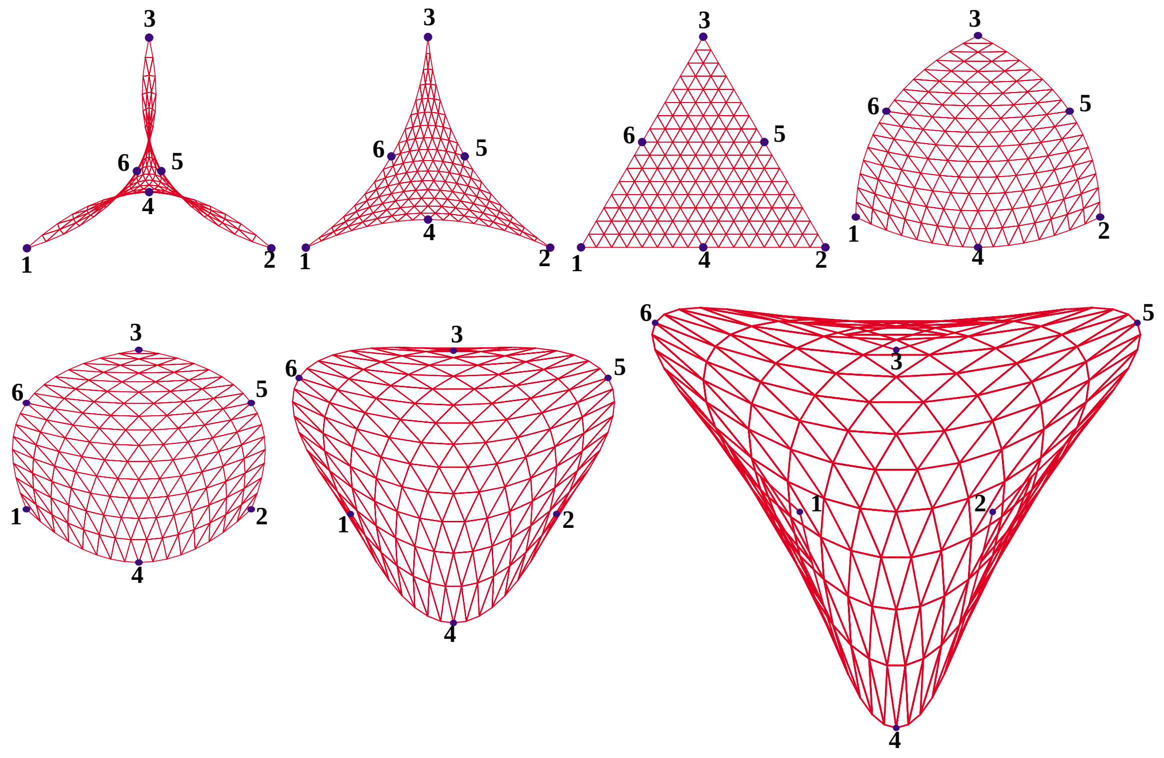


Figure 19.4. Effect of displacing midpoints 4, 5 and 6 of an equilateral 6-node triangle along the midpoint normals. Motion is inwards in first two top frames, outwards in the last four. In the lower leftmost picture nodes 1 through 6 lie on a circle.

Homework Exercises for Chapter 19
FEM Convergence Requirements

EXERCISE 19.1

[D:15] Draw a picture of a 2D non-matching mesh in which element nodes on two sides of a boundary do not share the same locations. Discuss why enforcing compatibility becomes difficult.

EXERCISE 19.2

[A:25] The isoparametric definition of the straight 3-node bar element in its local system $\bar{x}$ is

$$\begin{bmatrix} 1 \\ \bar{x} \\ \bar{v} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ \bar{x}_1 & \bar{x}_2 & \bar{x}_3 \\ \bar{u}_1 & \bar{u}_2 & \bar{u}_3 \end{bmatrix} \begin{bmatrix} N_1^{(e)}(\xi) \\ N_2^{(e)}(\xi) \\ N_3^{(e)}(\xi) \end{bmatrix} \tag{E19.1}$$

where 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.

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 a an end point.

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, $1\times1\times1$, $2\times2\times2$, etc) if the number of node points is $n = 8$, 20, 27, or 64.

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, and 9 at the center of the square. Displace 5 tangentially towards 4 until the jacobian determinant 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] A plane stress triangular element has 3 nodes located at the midpoints of the sides. Develop the 3 shape functions and study whether the element satisfies compatibility and completeness.

20 Miscellaneous Element Formulation Topics

TABLE OF CONTENTS

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.

§20.1. *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 seemlessly 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*, 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 accordining 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.

§20.1.1. *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

$$x = x(\xi), \quad u = u(\xi) \tag{20.1}$$

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

$$e_{\xi\xi} = \frac{dx}{d\xi}\frac{du}{d\xi} = J u_\xi \tag{20.2}$$

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:

$$e = \frac{du}{dx} = J^{-1}J\frac{du}{dx} = J^{-1}\frac{dx}{d\xi}\frac{du}{d\xi}\frac{d\xi}{dx} = J^{-1}e_{\xi\xi}J^{-1} = J^{-2}e_{\xi\xi} \tag{20.3}$$

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 (20.2). 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 20.1.

§20.1.2. *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 $\vec{\mathbf{x}}$ and the inplane displacement field into a 2-vector $\vec{\mathbf{u}}$. Then the natural strains are defined as

$$e_{\xi\xi} = \frac{\partial \vec{\mathbf{x}}}{\partial \xi} \cdot \frac{\partial \vec{\mathbf{u}}}{\partial \xi}, \quad e_{\eta\eta} = \frac{\partial \vec{\mathbf{x}}}{\partial \eta} \cdot \frac{\partial \vec{\mathbf{u}}}{\partial \eta}, \quad e_{\xi\eta} = \frac{\partial \vec{\mathbf{x}}}{\partial \xi} \cdot \frac{\partial \vec{\mathbf{u}}}{\partial \eta}, \quad e_{\eta\xi} = \frac{\partial \vec{\mathbf{x}}}{\partial \eta} \cdot \frac{\partial \vec{\mathbf{u}}}{\partial \xi}, \quad \gamma_{\xi\eta} = e_{\xi\eta} + e_{\eta\xi}. \tag{20.4}$$

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

$$\mathbf{e}_{\xi\eta} = \begin{bmatrix} e_{\xi\xi} & e_{\xi\eta} \\ e_{\eta\xi} & e_{\eta\eta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{bmatrix} \frac{\partial u_x}{\partial \xi} & \frac{\partial u_x}{\partial \eta} \\ \frac{\partial u_y}{\partial \xi} & \frac{\partial u_y}{\partial \eta} \end{bmatrix} = \mathbf{J}\,\mathbf{G}, \tag{20.5}$$

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

$$\mathbf{J}^{-1}\mathbf{e}_{\xi\eta}\mathbf{J}^{-T} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} \end{bmatrix} = \begin{bmatrix} e_{xx} & e_{yx} \\ e_{xy} & e_{yy} \end{bmatrix}. \tag{20.6}$$

from which the Cartesian components may be easily extracted and rearranged as a 3-vector. Exercise 20.4 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.

§20.1.3. *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

$$\mathbf{e}_{\xi\eta\zeta} = \begin{bmatrix} e_{\xi\xi} & e_{\xi\eta} & e_{\xi\eta} \\ e_{\eta\xi} & e_{\eta\eta} & e_{\eta\zeta} \\ e_{\zeta\xi} & e_{\zeta\eta} & e_{\zeta\zeta} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} & \frac{\partial z}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} & \frac{\partial z}{\partial \eta} \\ \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} & \frac{\partial z}{\partial \zeta} \end{bmatrix} \begin{bmatrix} \frac{\partial u_x}{\partial \xi} & \frac{\partial u_x}{\partial \eta} & \frac{\partial u_x}{\partial \zeta} \\ \frac{\partial u_y}{\partial \xi} & \frac{\partial u_y}{\partial \eta} & \frac{\partial u_y}{\partial \zeta} \\ \frac{\partial u_z}{\partial \xi} & \frac{\partial u_z}{\partial \eta} & \frac{\partial u_z}{\partial \zeta} \end{bmatrix} = \mathbf{J}\,\mathbf{G}, \tag{20.7}$$

where $\mathbf{J}$ is the Jacobian matrix for bricks. To pass to Cartesian strains use

$$\mathbf{J}^{-1}\mathbf{e}_{\xi\eta\zeta}\mathbf{J}^{-T} = \begin{bmatrix} \frac{\partial u_x}{\partial x} & \frac{\partial u_y}{\partial x} & \frac{\partial u_z}{\partial x} \\ \frac{\partial u_x}{\partial y} & \frac{\partial u_y}{\partial y} & \frac{\partial u_z}{\partial y} \\ \frac{\partial u_x}{\partial z} & \frac{\partial u_y}{\partial z} & \frac{\partial u_z}{\partial z} \end{bmatrix} = \begin{bmatrix} e_{xx} & e_{yx} & e_{zx} \\ e_{xy} & e_{yy} & e_{zy} \\ e_{xz} & e_{yz} & e_{zz} \end{bmatrix}. \tag{20.8}$$

from which the 6-vector of Cartesian strains is easily extracted.

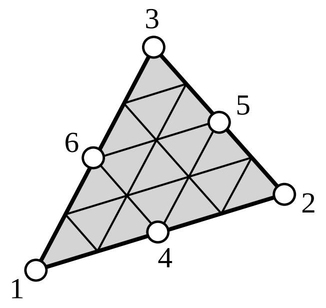


Figure 20.1. The six-node quadratic triangle (once more)

§20.2. *HIERARCHICAL SHAPE FUNCTIONS

Hierarchical shape functions is a topic motivated by the following factors.

(1) *Unification of element formation software.* The same element formation subroutine can generate a wide range of elements within a geometric family, including transition elements.

(2) *Simplified model preparation*. The specification of model geometry and boundary conditions is considerably simplified for elements with side nodes.

(3) *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.

In what follows we shall emphasize only factor the first factor.

§20.2.1. The Six Node Triangle, Revisited

Consider (again) the six-node quadratic triangle, which is pictured in Figure 20.1. The isoparametric definition of this element is repeated here for convenience:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \end{bmatrix}. \tag{20.9}$$

The shape functions are

$$\begin{aligned} N_1^{(e)} &= \zeta_1(2\zeta_1 - 1), & N_4^{(e)} &= 4\zeta_1\zeta_2 \\ N_2^{(e)} &= \zeta_2(2\zeta_2 - 1), & N_5^{(e)} &= 4\zeta_2\zeta_3 \\ N_3^{(e)} &= \zeta_3(2\zeta_3 - 1), & N_6^{(e)} &= 4\zeta_3\zeta_1 \end{aligned} \tag{20.10}$$

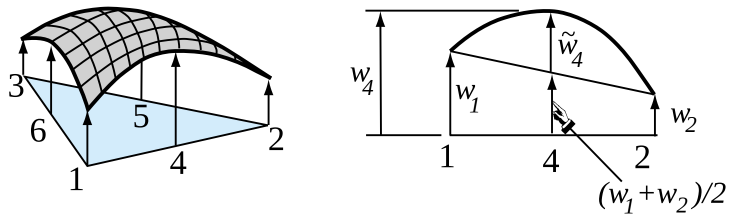


Figure 20.2. Interpretation of hierarchical midnode value $\tilde{w}_4$.

Now consider a generic scalar function, w, that is interpolated over the six-node triangle with the shape functions (20.3):

$$w = [\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \,] \begin{bmatrix} \zeta_1(2\zeta_1 - 1) \\ \zeta_2(2\zeta_2 - 1) \\ \zeta_3(2\zeta_3 - 1) \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}. \tag{20.11}$$

Symbol w may represent x, y, u_x or u_y in the isoparametric representation (20.1), or other element-varying quantities such as thickness, temperature, etc.

§20.2.2. Hierarchical Representation

The key step in going to a hierarchical representation of this element is to express the values of w at the midnodes 4, 5 and 6 as *deviations from the linear interpolation*:

$$\begin{aligned} w_4 &= \tfrac{1}{2}(w_1 + w_2) + \tilde{w}_4, \\ w_5 &= \tfrac{1}{2}(w_2 + w_3) + \tilde{w}_5, \\ w_4 &= \tfrac{1}{2}(w_3 + w_1) + \tilde{w}_6. \end{aligned} \tag{20.12}$$

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 20.2 illustrates the meaning of the hierarchical value $\tilde{w}_4$ at midnode 4.

If we insert (20.4) into (20.3), we get the *hierarchical interpolation* formula

$$w = [\, w_1 \quad w_2 \quad w_3 \quad \tilde{w}_4 \quad \tilde{w}_5 \quad \tilde{w}_6 \,] \begin{bmatrix} \zeta_1 \\ \zeta_2 \\ \zeta_3 \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}. \tag{20.13}$$

On comparing (20.5) and (20.3) two points become evident:

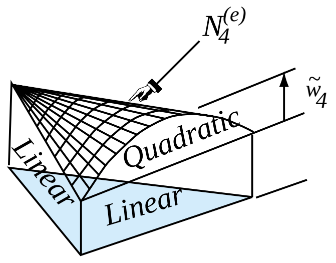


Figure 20.3. The midnode hierarchical shape functions as measured from the linear triangle functions.

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 20.3 for $\tilde{N}_4^{(e)} = 4\zeta_1\zeta_2$.

We can now insert the hierarchical interpolation formula (20.5) into rows 2 through 4 of (20.1) 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:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ x_1 & x_2 & x_3 & \tilde{x}_4 & \tilde{x}_5 & \tilde{x}_6 \\ y_1 & y_2 & y_3 & \tilde{y}_4 & \tilde{y}_5 & \tilde{y}_6 \\ u_{x1} & u_{x2} & u_{x3} & \tilde{u}_{x4} & \tilde{u}_{x5} & \tilde{u}_{x6} \\ u_{y1} & u_{y2} & u_{y3} & \tilde{u}_{y4} & \tilde{u}_{y5} & \tilde{u}_{y6} \end{bmatrix} \begin{bmatrix} \tilde{N}_1^{(e)} \\ \tilde{N}_2^{(e)} \\ \tilde{N}_3^{(e)} \\ \tilde{N}_4^{(e)} \\ \tilde{N}_5^{(e)} \\ \tilde{N}_6^{(e)} \end{bmatrix}, \tag{20.14}$$

where only the first three shape functions are different:

$$\begin{aligned} \tilde{N}_1^{(e)} &= \zeta_1, & \tilde{N}_4^{(e)} &= N_4^{(e)} = 4\zeta_1\zeta_2, \\ \tilde{N}_2^{(e)} &= \zeta_2, & \tilde{N}_5^{(e)} &= N_5^{(e)} = 4\zeta_2\zeta_3, \\ \tilde{N}_3^{(e)} &= \zeta_3, & \tilde{N}_6^{(e)} &= N_6^{(e)} = 4\zeta_3\zeta_1. \end{aligned} \tag{20.15}$$

The linear and quadratic parts of the element are now neatly separated:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} \text{Linear} & \text{Quadratic} \end{bmatrix} \begin{bmatrix} \text{Linear} \\ \text{Quadratic} \end{bmatrix}. \tag{20.16}$$

§20.2.3. *Hierarchical Stiffness Matrix and Load Vector

The element stiffness equations for the conventional (non-hierarchical) shape function form are

$$\mathbf{K}^{(e)}\mathbf{u}^{(e)} = \mathbf{f}^{(e)}. \tag{20.17}$$

We would like to transform these equations to the hierarchical form

$$\tilde{\mathbf{K}}^{(e)}\tilde{\mathbf{u}}^{(e)} = \tilde{\mathbf{f}}^{(e)}. \tag{20.18}$$

The nodal displacement vector may be partitioned as

$$\tilde{\mathbf{u}}^{(e)} = \begin{bmatrix} \tilde{u}_x^{(e)} \\ \tilde{u}_y^{(e)} \end{bmatrix}, \tag{20.19}$$

where $\tilde{\mathbf{u}}_x$ and $\tilde{\mathbf{u}}_y$ denote the 6×1 vectors

$$\tilde{\mathbf{u}}_x^{(e)} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \tilde{u}_{x4} \\ \tilde{u}_{x5} \\ \tilde{u}_{x6} \end{bmatrix}, \qquad \tilde{\mathbf{u}}_y^{(e)} = \begin{bmatrix} \tilde{u}_{y1} \\ \tilde{u}_{y2} \\ \tilde{u}_{y3} \\ \tilde{u}_{y4} \\ \tilde{u}_{y5} \\ \tilde{u}_{y6} \end{bmatrix}. \tag{20.20}$$

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:

$$\mathbf{u}_x^{(e)} = \tilde{\mathbf{T}}_x\tilde{\mathbf{u}}_x^{(e)}, \qquad \mathbf{u}_y^{(e)} = \tilde{\mathbf{T}}_y\tilde{\mathbf{u}}_y^{(e)}. \tag{20.21}$$

To construct these transformation matrices, observe that

$$\begin{bmatrix} u_{x1} \\ u_{x2} \\ u_{x3} \\ u_{x4} \\ u_{x5} \\ u_{x6} \end{bmatrix} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \frac{1}{2}(u_{x1}+u_{x2})+\tilde{u}_{x4} \\ \frac{1}{2}(u_{x2}+u_{x3})+\tilde{u}_{x5} \\ \frac{1}{2}(u_{x3}+u_{x1})+\tilde{u}_{x6} \end{bmatrix} = \begin{bmatrix} \tilde{u}_{x1} \\ \tilde{u}_{x2} \\ \tilde{u}_{x3} \\ \frac{1}{2}(\tilde{u}_{x1}+\tilde{u}_{x2})+\tilde{u}_{x4} \\ \frac{1}{2}(\tilde{u}_{x2}+\tilde{u}_{x3})+\tilde{u}_{x5} \\ \frac{1}{2}(\tilde{u}_{x3}+\tilde{u}_{x1})+\tilde{u}_{x6} \end{bmatrix}, \tag{20.22}$$

with exactly the same relation holding for u_y. We can present (20.14) in the matrix form (20.13), with

$$\tilde{\mathbf{T}}_x = \tilde{\mathbf{T}}_y = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 & 1 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 & 1 \end{bmatrix}. \tag{20.23}$$

Combining the two matrix equations in one:

$$\mathbf{u}^{(e)} = \tilde{\mathbf{T}}\,\tilde{\mathbf{u}}^{(e)} \tag{20.24}$$

where $\tilde{\mathbf{T}}$ is the 12×12 transformation matrix

$$\tilde{\mathbf{T}} = \begin{bmatrix} \tilde{\mathbf{T}}_x & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{T}}_y \end{bmatrix}. \tag{20.25}$$

Inserting (20.17) into (20.9) and premultiplying by $\tilde{\mathbf{T}}^T$ we obtain (20.10), in which

$$\begin{aligned} \tilde{\mathbf{K}}^{(e)} &= \tilde{\mathbf{T}}^T \mathbf{K}^{(e)} \tilde{\mathbf{T}} \\ \tilde{\mathbf{f}}^{(e)} &= \tilde{\mathbf{T}}^T \mathbf{f}^{(e)} \end{aligned}. \tag{20.26}$$

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 very useful for checking element derivations and computer programs.

§20.2.4. *Equation Nesting and Node Dropping

To take full advantage of the "node dropping" feature described in §20.6, 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

$$\begin{gathered} [\,\tilde{u}_{x1} \quad \tilde{u}_{x2} \quad \tilde{u}_{x3} \quad \tilde{u}_{x4} \quad \tilde{u}_{x5} \quad \tilde{u}_{x6} \quad \tilde{u}_{y1} \quad \tilde{u}_{y2} \quad \tilde{u}_{y3} \quad \tilde{u}_{y4} \quad \tilde{u}_{y5} \quad \tilde{u}_{y6}\,] \\ \text{to} \\ [\,\tilde{u}_{x1} \quad \tilde{u}_{y1} \quad \tilde{u}_{x2} \quad \tilde{u}_{y2} \quad \tilde{u}_{x3} \quad \tilde{u}_{y3} \quad \tilde{u}_{x4} \quad \tilde{u}_{y4} \quad \tilde{u}_{x5} \quad \tilde{u}_{y5} \quad \tilde{u}_{x6} \quad \tilde{u}_{y6}\,]. \end{gathered} \tag{20.27}$$

With this rearrangement of nodal freedoms, the hierarchical stiffness equations of the six-node triangle take the *nested form*

$$\begin{bmatrix} \text{Linear} & \\ & \\ & \text{Quadratic} \end{bmatrix} \begin{bmatrix} \text{Linear} \\ \\ \text{Quadratic} \end{bmatrix} = \begin{bmatrix} \text{Linear} \\ \\ \text{Quadratic} \end{bmatrix}, \tag{20.28}$$

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 20.4 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

$$\tilde{u}_{x6} = \tilde{u}_{y6} = 0, \tag{20.29}$$

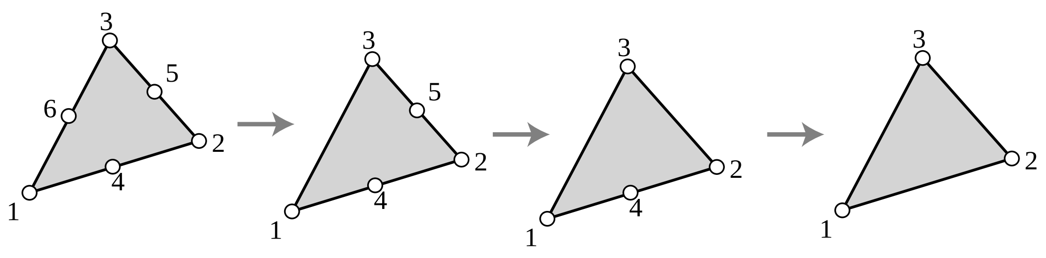


Figure 20.4. Transition triangular elements

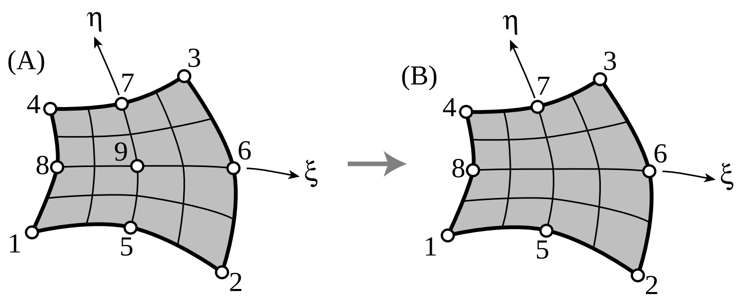


Figure 20.5. Conventional eight-node quadrilateral (A) and nine-node quadrilateral (B) with hierarchical internal node.

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 20.4.

Dropping node 5 through a similar procedure we get the four-node transition element ($n = 4$) shown in Figure 20.4. Finally, dropping node 4 we reduce the element to the three-node linear triangle.

§20.2.5. *Internal Node Injection

The hierarchical representation can also be applied to *internal* nodes. The algebra is somewhat more 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 20.5: 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 §20.6 to build transition elements.

The isoparametric representation of (A) is

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \end{bmatrix}, \tag{20.30}$$

with the shape functions

$$\begin{array}{ll} N_1^{(e)} = -\frac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta), & N_5^{(e)} = \frac{1}{2}(1-\xi^2)(1-\eta) \\ N_2^{(e)} = -\frac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta), & N_6^{(e)} = \frac{1}{2}(1-\eta^2)(1+\xi) \\ N_3^{(e)} = -\frac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta), & N_7^{(e)} = \frac{1}{2}(1-\xi^2)(1+\eta) \\ N_4^{(e)} = -\frac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta), & N_8^{(e)} = \frac{1}{2}(1-\eta^2)(1-\xi) \end{array}. \tag{20.31}$$

Express a generic quantity w over element (A) as

$$w^A = [\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \,] \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \end{bmatrix}. \tag{20.32}$$

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:

$$w_9 = w_9^A(0,0) + \tilde{w}_9 \tag{20.33}$$

The center value is obtained by evaluating (20.24) at $\xi = \eta = 0$, which gives

$$w_9^A(0,0) = -\tfrac{1}{4}(w_1 + w_2 + w_3 + w_4) + \tfrac{1}{2}(w_5 + w_6 + w_7 + w_8), \tag{20.34}$$

so that the full form of (20.25) is

$$w_9 = -\tfrac{1}{4}(w_1 + w_2 + w_3 + w_4) + \tfrac{1}{2}(w_5 + w_6 + w_7 + w_8) + \tilde{w}_9. \tag{20.35}$$

The transformation between non-hierarchical and hierarchical values may be expressed in matrix form:

$$\begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \\ w_7 \\ w_8 \\ w_9 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & -\frac{1}{4} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & \frac{1}{2} & 1 \end{bmatrix} \begin{bmatrix} \tilde{w}_1 \\ \tilde{w}_2 \\ \tilde{w}_3 \\ \tilde{w}_4 \\ \tilde{w}_5 \\ \tilde{w}_6 \\ \tilde{w}_7 \\ \tilde{w}_8 \\ \tilde{w}_9 \end{bmatrix}. \tag{20.36}$$

The shape function associated with the internal node is the bubble function

$$N_9^{(e)} = (1-\xi^2)(1-\eta^2) \tag{20.37}$$

With these results the generic interpolation formula for (B) becomes

$$w^B = [\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \quad w_7 \quad w_8 \quad \tilde{w}_9 \,] \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \\ N_9^{(e)} \end{bmatrix}, \tag{20.38}$$

where the shape functions are (20.23) and (20.29). [No tildes on the N's are necessary, because none of these functions changes in going from (A) to (B).] Finally, on specializing (20.30) to x, y, etc, we obtain the isoparametric representation of element (B):

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 & x_7 & x_8 & \tilde{x}_9 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 & y_7 & y_8 & \tilde{y}_9 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} & u_{x7} & u_{x8} & \tilde{u}_{x9} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} & u_{y7} & u_{y8} & \tilde{u}_{y9} \end{bmatrix} \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \\ N_7^{(e)} \\ N_8^{(e)} \\ N_9^{(e)} \end{bmatrix}. \tag{20.39}$$

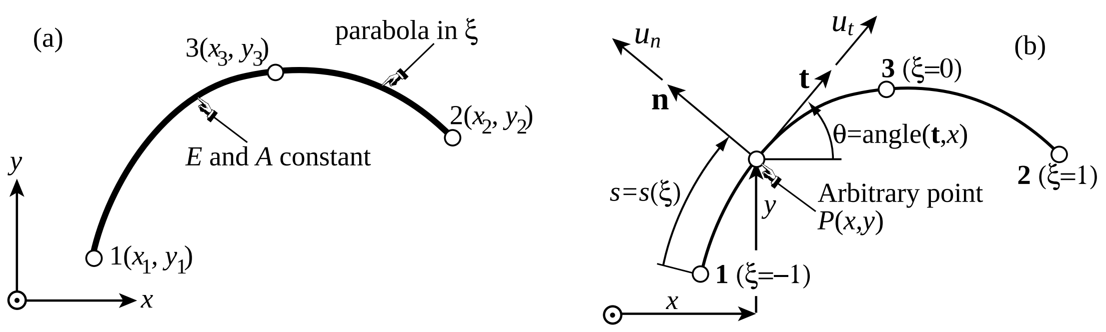


Figure 20.6. Figure (a) shows a 3-node curved bar element with constant modulus E and area A. This element has six degrees of freedom and can only transmit an axial force $N = EAe$. Figure 20.6(b) depicts geometric details covered in the text.

§20.3. *A 3-NODE CURVED BAR ELEMENT

This section formulates a *curved* 3-node bar isoparametric element in which node 3 is allowed to be off alignment from 1 and 2, as depicted in Figure 20.6(a). The derivation provides a first glimpse of techniques used in advanced FEM expositions to develop curved beam and shell elements.

§20.3.1. *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 $N = E A\, e$, where e is the axial strain.

Additional geometric details are given in Figure 20.6(b). At an arbitrary point $P(x, y)$ of the element one defines the tangential and normal directions by the *unit* vectors **t** and **n**, respectively.[1] The positive sense of **t** corresponds to traversing along from 1 to 2. The normal vector **n** is at 90° CCW from **t**. The angle formed by **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 $\mathbf{u} = [\, u_x \quad u_y \,]^T$. The displacement components in the directions **t** and **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

$$e = \frac{du_t}{ds} - \kappa u_n \tag{20.40}$$

where κ is the inplane bar curvature, the expression of which is derived in §20.3.3. The corrective term κu_n is known as the *hoop strain* in the theory of curved bars and rods.[2]

The isoparametric definition of this element is[3]

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix} \tag{20.41}$$

[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.

[3] Supercript (e) is omitted for brevity.

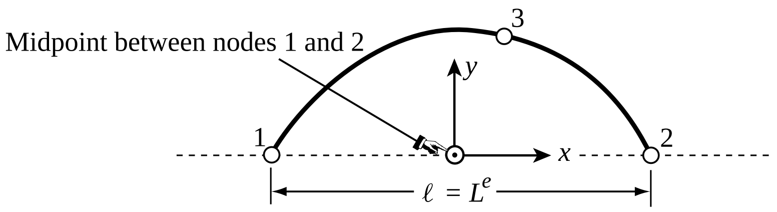


Figure 20.7. Choice of Cartesian local axes to simplify derivations.

Here

$$N_1 = -\tfrac{1}{2}\xi(1-\xi), \quad N_2 = \tfrac{1}{2}\xi(1+\xi), \quad N_3 = 1-\xi^2 \tag{20.42}$$

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

$$U^{(e)} = \tfrac{1}{2}\int_{\text{arclength}} EA\,e^2\,ds = \tfrac{1}{2}\int_{-1}^{1} EA\,e^2\,J\,d\xi = \tfrac{1}{2}EA\int_{-1}^{1} e^2\,J\,d\xi, \quad J = ds/d\xi = s'. \tag{20.43}$$

To simplify the analytical derivations that follow, we orient $\{x,y\}$ as shown in Figure 20.7, 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 §20.3.3, is

$$\begin{aligned} e &= \frac{du_n}{ds} - \kappa u_n = \frac{d(\mathbf{u}^T\mathbf{t})}{ds} - \kappa u_n = \left(\frac{d\mathbf{u}}{ds}\right)^T \mathbf{t} + \mathbf{u}^T\frac{d\mathbf{t}}{ds} - \kappa u_n \\ &= \left(\frac{d\mathbf{u}}{d\xi}\right)^T \frac{d\xi}{ds}\,\mathbf{t} + \mathbf{u}^T(\kappa\mathbf{n}) - \kappa u_n = J^{-1}\mathbf{t}^T\mathbf{u}' + \kappa\,u_n - \kappa\,u_n = J^{-1}\mathbf{t}^T\mathbf{u}'. \end{aligned} \tag{20.44}$$

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 §20.3.3. The values of $\mathbf{u}$ and $\mathbf{u}'$ can be calculated from the last two rows of the isoparametric definition:

$$\mathbf{u} = \begin{bmatrix} u_x \\ u_y \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1 \\ N_2 \\ N_3 \end{bmatrix}, \tag{20.45}$$

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

$$\mathbf{u}' = \begin{bmatrix} u_x' \\ u_y' \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} N_1' \\ N_2' \\ N_3' \end{bmatrix} = \begin{bmatrix} u_{x1} & u_{x2} & u_{x3} \\ u_{y1} & u_{y2} & u_{y3} \end{bmatrix} \begin{bmatrix} \xi - \frac{1}{2} \\ \xi + \frac{1}{2} \\ -2\xi \end{bmatrix}. \tag{20.46}$$

Collecting terms and replacing the expression of $\mathbf{t}$ given in §20.3.3:

$$e = \mathbf{B}\,\mathbf{u}^{(e)}, \tag{20.47}$$

[4] Axes $\{x,y\}$ in Figure 20.7 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 20.8. 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.

$$\mathbf{B} = J^{-2}\,[\,x'N_1' \quad y'N_1' \quad x'N_2' \quad y'N_2' \quad x'N_3' \quad y'N_3'\,], \tag{20.48}$$

The derivatives $x' = dx/d\xi$ and $y' = dy/d\xi$ are obtained from rows 2-3 of the isoparametric element definition:

$$x' = x_1N_1' + x_2N_2' + x_3N_3' = (\tfrac{1}{2} - 2\alpha\xi)\ell, \qquad y' = y_1N_1' + y_2N_2' + y_3N_3' = -2\beta\xi\ell, \tag{20.49}$$

whence

$$J = \sqrt{(x')^2 + (y')^2} = \ell\sqrt{(\tfrac{1}{2} - 2\alpha\xi)^2 + 4\beta^2\xi^2}, \tag{20.50}$$

Finally,

$$\mathbf{B} = \frac{\ell}{J^2}\,[\,(\tfrac{1}{2} - 2\alpha\xi)(\xi - \tfrac{1}{2}) \quad 2\beta\xi(\xi - \tfrac{1}{2}) \quad (\tfrac{1}{2} - 2\alpha\xi)(\xi + \tfrac{1}{2}) \quad -2\beta\xi(\xi + \tfrac{1}{2}) \quad -2(\tfrac{1}{2} - 2\alpha\xi)\xi \quad 2\beta\xi^2\,]. \tag{20.51}$$

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 **B** reduces to

$$\mathbf{B} = \frac{1}{\ell(\frac{1}{2} - 2\alpha\xi)}\,[\,\xi - \tfrac{1}{2} \quad 0 \quad \xi + \tfrac{1}{2} \quad 0 \quad -2\xi \quad 0\,]. \tag{20.52}$$

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 **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 **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}
                                       1
Check unif strain for straight bar={---}
                                      ell
```

These checks can also be done by hand, but there is some error-prone algebra involved.

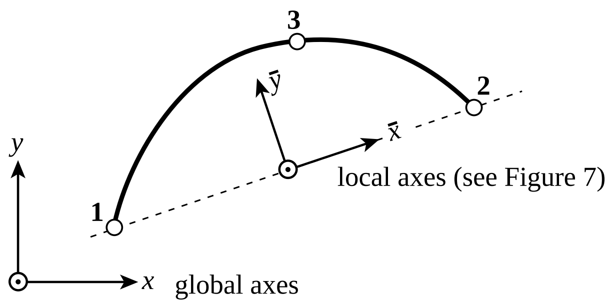


Figure 20.8. Global and local axes for the transformations used in forming the global stiffness matrix.

§20.3.2. 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 `{ { x1,y1 },{ x2,y2 },{ x3,y3 } }`.
`mprop`	Material properties supplied as `{ Em }`, 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 20..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

$$\begin{aligned} \bar{x} &= (x - x_m)\,c + (y - y_m)\,s, & \bar{y} &= -(x - x_m)\,s + (y - y_m)\,c \\ x &= \bar{x}\,c - \bar{y}\,s + x_m, & y &= \bar{x}\,s + \bar{y}\,c + y_m, \end{aligned} \tag{20.53}$$

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 20.8. Whence

$$\begin{aligned} \alpha = \bar{x}_3/\ell &= \frac{(x_3 - x_m)\,c + (y_3 - y_m)\,s}{\ell} = \frac{(x_3 - x_m)\,x_{21} + (y_3 - y_m)\,y_{21}}{\ell^2}, \\ \beta = \bar{y}_3/\ell &= \frac{-(x_3 - x_m)\,s + (y_3 - y_m)\,c}{\ell} = \frac{-(x_3 - x_m)\,y_{21} + (y_3 - y_m)\,x_{21}}{\ell^2}. \end{aligned} \tag{20.54}$$

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 `Kebar= Em*A*(J1*Transpose[B1].B1+J2*Transpose[B2].B2`, where `B1`, `J1`, etc., are function evaluations at the Gauss points of the 2-point rule. From inspection, the global stiffness matrix is given by

$$\mathbf{K}^{(e)} = \mathbf{T}^T\,\bar{\mathbf{K}}^{(e)}\,\mathbf{T}, \qquad \text{where} \quad \mathbf{T} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & c & s & 0 & 0 \\ 0 & 0 & -s & c & 0 & 0 \\ 0 & 0 & 0 & 0 & c & s \\ 0 & 0 & 0 & 0 & -s & c \end{bmatrix} \tag{20.55}$$

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:

$$\mathbf{K}^{(e)} = \frac{1}{\ell^2}\,\hat{\mathbf{T}}^T\,\bar{\mathbf{K}}^{(e)}\,\hat{\mathbf{T}}, \qquad \text{where} \quad \hat{\mathbf{T}} = \begin{bmatrix} x_{21} & y_{21} & 0 & 0 & 0 & 0 \\ -y_{21} & x_{21} & 0 & 0 & 0 & 0 \\ 0 & 0 & x_{21} & y_{21} & 0 & 0 \\ 0 & 0 & -y_{21} & x_{21} & 0 & 0 \\ 0 & 0 & 0 & 0 & x_{21} & y_{21} \\ 0 & 0 & 0 & 0 & -y_{21} & x_{21} \end{bmatrix}, \tag{20.56}$$

This is the transformation implemented in `Stiffness3NodePlaneCurvedBar`, in which $\hat{\mathbf{T}}$ is called `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}
Ke.rm3={0, 0, 0, 0, 0, 0}
```

The eigenvalue distribution show a rank deficiency of 1 (4 zero eigenvalues, one more that $6 - 3 = 3$). This property is explained in §20.3.4. 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:

```
                                                1  1
alpha,beta in Stiffness3NodePlaneCurvedBar={-, -}
                                                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.

§20.3.3. Geometric Properties of Plane Curves

The following geometric properties of plane curves are collected to help in the analytical derivations of §20.3.1. They can be found in any book on differential geometry, such as the well known textbook by Struik.[5]

Consider a smooth plane curve given in parametric form

$$x = x(\xi), \qquad y = y(\xi). \tag{20.57}$$

The arclength s is also a function $s = s(\xi)$ of the coordinate ξ; cf. Figure 20.6.

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

$$J = s' = x'\frac{dx}{ds} + y'\frac{dy}{ds} = x'\cos\theta + y'\sin\theta = \sqrt{(x')^2 + (y')^2}, \tag{20.58}$$

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

$$\mathbf{t} = \begin{bmatrix} dx/ds \\ dy/ds \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix} = J^{-1}\begin{bmatrix} x' \\ y' \end{bmatrix} \tag{20.59}$$

$$\mathbf{n} = \begin{bmatrix} -dy/ds \\ dx/ds \end{bmatrix} = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix} = J^{-1}\begin{bmatrix} -y' \\ x' \end{bmatrix} \tag{20.60}$$

Here $\mathbf{n}$ has been taken to be at $+90^\circ = +\pi/2$ radians, from $\mathbf{t}$. The curvature is given by

$$\kappa = \frac{x'y'' - x''y'}{J^3}. \tag{20.61}$$

Expression (20.61) is not needed for a bar element. It would be required, however, for a curved beam element as in Exercise 20.8.

The derivative of $\mathbf{t}$ with respect to the arclength s is given by the first Frénet formula

$$\frac{d\mathbf{t}}{ds} = \kappa\, \mathbf{n}, \tag{20.62}$$

§20.3.4. *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

[5] D. J. Struik, *Lectures on Classical Differential Geometry*, Addison-Wexley, New York, 2nd ed., 1961.

[6] These are also called “kinematic mechanisms” in the FEM literature.

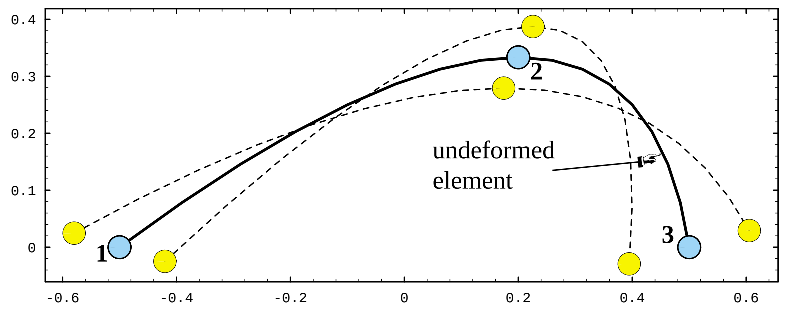


Figure 20.9. The inextensional zero-energy mode (IZEM) of a 3-node curved bar. Depicted for an element with $\ell = 1$, $\alpha = 1/5$ and $\beta = 1/3$.

looks like, the bar element with $E = A = \ell = 1$, $\alpha = 1/5$ and $\beta = 1/5$ previously treated in §20.3.2 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:

```
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 20.9. 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 not change length (in the linear approximation), and thus takes no axial strain energy. If bending energy is taking into account, however, the element becomes rank-sufficient.

Homework Exercises for Chapter 20

Miscellaneous Element Formulation Topics

EXERCISE 20.1

[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 compatibilty and completeness requirements. Explain how you would construct the strain-displacement matrix (this is not a trivial problem).

EXERCISE 20.2

[A:35] As above, but for a two-dimensional isoparametric element.

EXERCISE 20.3

[A:30] Develop a theory of natural strains and stresses for the 3-node curved bar element formulated in §20.3.

EXERCISE 20.4

[A:20] Prove the transformation (20.6). 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 20.5

[A:25] As in the previous Exercise, but for the 3D case discussed in §20.1.3.

EXERCISE 20.6

[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 (20.23) 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 (20.23) but with 10 matrix columns, and a 10×10 transformation matrix. Verify that rigid body motions and constant strain states are preserved.

EXERCISE 20.7

[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 20.8

[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.

21
Implementation of One-Dimensional Elements

TABLE OF CONTENTS

This Chapter begins Part III of the course. This Part deals with the computer implementation of the Finite Element Method. 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.

Theis Chapter illustrates, through specific examples, the programming of one-dimensional elements: bars and beams, using *Mathematica* as implementation language. The programming of both stiffness and mass matrices are illustrated although only the stiffness matrix is used in this course.

§21.1. 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 21.1 for conveniency. It has two nodes and four degrees of freedom. The element node displacements and congugate forces are

$$\mathbf{u}^{(e)} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ u_{y2} \end{bmatrix}, \quad \mathbf{f}^{(e)} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{x2} \\ f_{y2} \end{bmatrix}. \tag{21.1}$$

The element geometry is described by the coordinates $\{x_i, y_i\}$, $i = 1, 2$ of the two end nodes. The two material properties involved in the stiffness and mass computations are the modulus of elasticity E and the mass density per unit volume ρ. The only fabrication property required is the cross section area A. All of these properties are taken to be constant over the element.

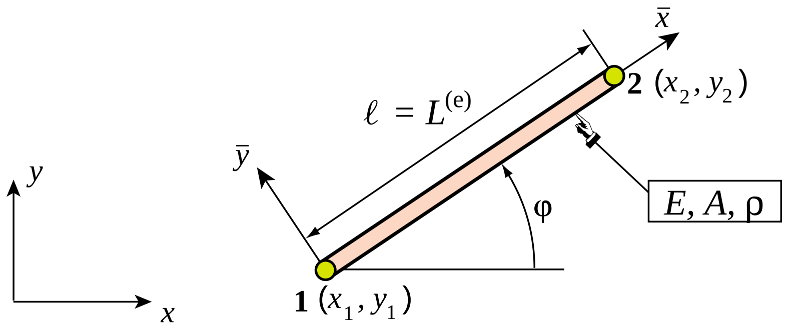


Figure 21.1. Plane bar element.

§21.1.1. Element Formulation

The element stiffness matrix in global $\{x, y\}$ coordinates is given by the explicit expression derived in §3.1:

$$\mathbf{K}^{(e)} = \frac{EA}{\ell} \begin{bmatrix} c^2 & sc & -c^2 & -sc \\ sc & s^2 & -sc & -s^2 \\ -c^2 & -sc & c^2 & sc \\ -sc & -s^2 & sc & s^2 \end{bmatrix} = \frac{EA}{\ell^3} \begin{bmatrix} x_{21}x_{21} & x_{21}y_{21} & -x_{21}x_{21} & -x_{21}y_{21} \\ x_{21}y_{21} & y_{21}y_{21} & -x_{21}y_{21} & -y_{21}y_{21} \\ -x_{21}x_{21} & -x_{21}y_{21} & x_{21}x_{21} & x_{21}y_{21} \\ -x_{21}y_{21} & -y_{21}y_{21} & x_{21}y_{21} & y_{21}y_{21} \end{bmatrix}. \tag{21.2}$$

```
PlaneBar2Stiffness[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,numer,L,LL,LLL,Ke},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=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]
];
PlaneBar2ConsMass[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,numer,L,MeC},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=mprop; {A}=fprop; {numer}=opt;
  If [numer,{x21,y21,ρ,A}=N[{x21,y21,ρ,A}]];
  L=Simplify[PowerExpand[Sqrt[x21^2+y21^2]]];
  MeC=(ρ*A*L/6)*{{2,0,1,0},{0,2,0,1},{1,0,2,0},{0,1,0,2}};
  Return[MeC]
];
PlaneBar2LumpMass[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,numer,L,MeL},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=mprop; {A}=fprop; {numer}=opt;
  If [numer,{x21,y21,ρ,A}=N[{x21,y21,ρ,A}]];
  L=Simplify[PowerExpand[Sqrt[x21^2+y21^2]]];
  MeL=(ρ*A*L/2)*IdentityMatrix[4];
  Return[MeL]
];
```

Figure 21.2. *Mathematica* modules to form stiffness and mass matrices of a 2-node, prismatic plane bar element.

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 21.1). The second matrix expression in (21.2) is useful in symbolic work, because it enhances simplification possibilities.

The consistent and lumped mass matrix are given by

$$\mathbf{M}_C^{(e)} = \frac{\rho A\ell}{6}\begin{bmatrix} 2 & 0 & 1 & 0 \\ 0 & 2 & 0 & 1 \\ 1 & 0 & 2 & 0 \\ 0 & 1 & 0 & 2 \end{bmatrix}, \qquad \mathbf{M}_L^{(e)} = \frac{\rho A\ell}{2}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{21.3}$$

Both mass matrices are independent of the orientation of the element, that is, do not depend on the angle φ.[1]

[1] For the derivation of the consistent mass matrix, see, e.g., the book by Przemieniecki, *Matrix Structural Analysis*, Dover, 1968.

§21.1.2. Element Formation Modules

Figure 21.2 lists three *Mathematica* modules:

`PlaneBar2Stiffness` returns the element stiffness matrix $\mathbf{K}^{(e)}$ of a 2-node, prismatic plane bar element, given by (21.2). `PlaneBar2ConsMass` returns the consistent mass matrix $\mathbf{M}_C^{(e)}$ of a 2-node, prismatic plane bar element, given by (21.3). `PlaneBar2LumpMass` returns the lumped mass matrix $\mathbf{M}_L^{(e)}$ of a 2-node, prismatic plane bar element, given by (21.3).

The argument sequence of the three modules is exactly the same:

```
[ ncoor, mprop, fprop, opt ]
```

These four arguments are actually *lists* that collect the following data:

node coordinates, material properties, fabrication properties, options

The internal structure of these lists is

```
ncoor    {{x1,y1},{x2,y2}}
mprop    {Em,Gm,rho,alpha}
fprop    {A}
opt      {numer}
```
(21.4)

Here `x1,y1` and `x2,y2` are the coordinates of the end nodes, and `Em`, `Gm`, `rho`, `alpha` and `A` stand for E, G, ρ, α and A, respectively. For the stiffness matrix only E and A are used. In the mass matrix modules only ρ and A are used. Entries `Gm` and `alpha` are included to take care of other 1D elements.

The only option is `numer`, which is a logical flag with the value `True` or `False`. If `True` the computations are carried out in numerical floating-point arithmetic.

§21.1.3. Testing the Plane Bar Element Modules

The modules are tested by the statements listed in Figures 21.3 and 21.5.

The script of Figure 21.3 tests a numerically defined element with end nodes located at (0, 0) and (30, 40), with $E = 1000$, $A = 5$, $\rho = 6/10$, and `numer` set to `True`. Executing the statements in Figure 21.3 produces the results listed in Figure 21.4. The tests consist of the following operations:

Testing $\mathbf{K}^{(e)}$. The stiffness matrix returned in `Ke` 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 EA/ℓ. The symmetry of `Ke` is also checked but the output is not shown to save space.

Testing $\mathbf{M}_C^{(e)}$. The consistent mass matrix returned in `MeC` is printed. Its four eigenvalues are computed and printed. As expected they are all positive and form two pairs. The symmetry is also checked but the output is now shown.

Testing $\mathbf{M}_L^{(e)}$. The lumped mass matrix returned in `MeL` is printed. This is a diagonal matrix and consequently its eigenvalues are the same as the diagonal entries.

```
ncoor={{0,0},{30,40}}; mprop={1000,0,6/10,0};
fprop={5}; opt={True};

Ke= PlaneBar2Stiffness[ncoor,mprop,fprop,opt];
Print["Numerical Elem Stiff Matrix: "];
Print[Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]]]];
Print["Symmetry check=",Simplify[Transpose[Ke]-Ke]];

MeC= PlaneBar2ConsMass[ncoor,mprop,fprop,opt];
Print["Numerical Consistent Mass Matrix: "];
Print[MeC//MatrixForm];
Print["Eigenvalues of MeC=",Eigenvalues[N[MeC]]];
Print["Symmetry check=",Simplify[Transpose[MeC]-MeC]];

MeL= PlaneBar2LumpMass[ncoor,mprop,fprop,opt];
Print["Numerical Lumped Mass Matrix: "];
Print[MeL//MatrixForm];
Print["Eigenvalues of MeL=",Eigenvalues[N[MeL]]];
```

Figure 21.3. Test of plane bar element modules with numerical inputs.

```
Numerical Elem Stiff Matrix:
 36.   48.  -36.  -48.
 48.   64.  -48.  -64.
-36.  -48.   36.   48.
-48.  -64.   48.   64.
Eigenvalues of Ke={200., 0, 0, 0}
Numerical Consistent Mass Matrix:
 50.   0   25.   0
  0   50.   0   25.
 25.   0   50.   0
  0   25.   0   50.
Eigenvalues of MeC={75., 75., 25., 25.}
Numerical Lumped Mass Matrix:
 75.   0    0    0
  0   75.   0    0
  0    0   75.   0
  0    0    0   75.
Eigenvalues of MeL={75., 75., 75., 75.}
```

Figure 21.4. Output from test statements of Figure chapdot3
(outputs from symmetry checks not shown).

The script of Figure 21.5 tests a symbolically defined element with end nodes located at (0, 0) and $(L, 0)$, which is aligned with the x axis. The element properties E, ρ and A are kept symbolic. Executing the statements in Figure 21.4 produces the results shown in Figure 21.5.

The sequence of tests on the symbolic element is essentially the same carried out before, but a

```
ClearAll[A,Em,ρ,L,opt];
ncoor={{0,0},{L,0}}; mprop={Em,0,ρ,0}; fprop={A}; opt={False};
Ke=  PlaneBar2Stiffness[ncoor,mprop,fprop,opt];
kfac=Em*A/L; Ke=Simplify[Ke/kfac];
Print["Symbolic Elem Stiff Matrix: "];
Print[kfac," ",Ke//MatrixForm];
Print["Eigenvalues of Ke=",kfac,"*",Eigenvalues[Ke]];
MeC= PlaneBar2ConsMass[ncoor,mprop,fprop,opt];
mfac=ρ*L*A/6; MeC= Simplify[MeC/mfac];
Print["Symbolic Consistent Mass Matrix: "];
Print[mfac," ",MeC//MatrixForm];
Print["Eigenvalues of MeC=",mfac,"*",Eigenvalues[MeC]];
Print["Squared frequencies=",Simplify[kfac/mfac],"*",
         Simplify[Eigenvalues[Inverse[MeC].Ke]]];
MeL= PlaneBar2LumpMass[ncoor,mprop,fprop,opt];
mfac=ρ*L*A/2; MeL= Simplify[MeL/mfac];
Print["Symbolic Lumped Mass Matrix: "];
Print[mfac," ",MeL//MatrixForm];
Print["Eigenvalues of MeL=",mfac,"*",Eigenvalues[MeL]];
Print["Squared frequencies=",Simplify[kfac/mfac],"*",
         Simplify[Eigenvalues[Inverse[MeC].Ke]]];
```

Figure 21.5. Test of plane bar element modules with symbolic inputs.

```
Symbolic Elem Stiff Matrix:
         ⎛ 1   0  −1  0 ⎞
 A Em    ⎜ 0   0   0  0 ⎟
 ────    ⎜−1   0   1  0 ⎟
  L      ⎝ 0   0   0  0 ⎠
                     A Em
 Eigenvalues of Ke= ──── *{0, 0, 0, 2}
                      L
 Symbolic Consistent Mass Matrix:
          ⎛2  0  1  0⎞
  A L ρ   ⎜0  2  0  1⎟
  ─────   ⎜1  0  2  0⎟
    6     ⎝0  1  0  2⎠
                      A L ρ
 Eigenvalues of MeC= ───── *{1, 1, 3, 3}
                        6
                       6 Em
 Squared frequencies= ───── *{0, 0, 0, 2}
                      L² ρ
 Symbolic Lumped Mass Matrix:
          ⎛1  0  0  0⎞
  A L ρ   ⎜0  1  0  0⎟
  ─────   ⎜0  0  1  0⎟
    2     ⎝0  0  0  1⎠
                      A L ρ
 Eigenvalues of MeL= ───── *{1, 1, 1, 1}
                        2
                       2 Em
 Squared frequencies= ───── *{0, 0, 0, 2}
                      L² ρ
```

Figure 21.6. Output from test statements of Figure 21.5.

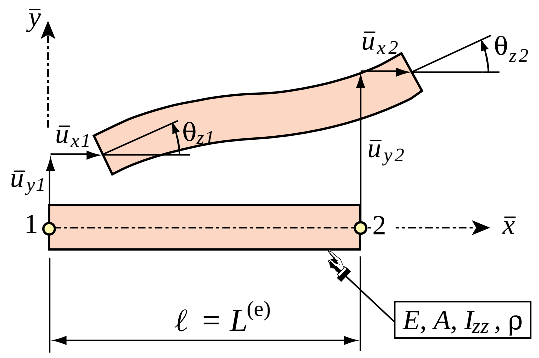


Figure 21.7. Plane beam-column element in its local system.

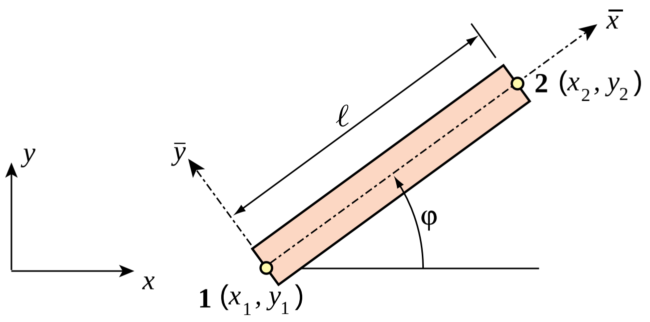


Figure 21.8. Plane beam-column element transformation to global system.

frequency test has been added. This consists of solving the one-element vibration eigenproblem

$$\mathbf{K}^{(e)}\mathbf{v}_i = \omega_i^2 \mathbf{M}_C^{(e)}\mathbf{v}_i, \qquad \mathbf{K}^{(e)}\mathbf{v}_i = \omega_i^2 \mathbf{M}_L^{(e)}\mathbf{v}_i, \tag{21.5}$$

in which ω_i are circular frequencies and $\mathbf{v}_i$ the associated eigenvectors or vibration mode shapes. Because the 3 rigid body modes are solution of (21.5), three zero frequencies are expected, which is borned out by the tests. The single positive nonzero frequency $\omega_a > 0$ corresponds to the vibration mode of axial extension and contraction. The consistent mass yields $\omega_a^2 = 12E/(\rho L^2)$ whereas the lumped mass yields $\omega_a^2 = 4E/(\rho L^2)$. The exact continuum solution for this axial mode is $\omega_a^2 = \pi^2 E/(\rho L^2) \approx 9.86E/(\rho L^2)$. Hence the consistent mass overestimates the true frequency whereas the lumped mass underestimates it, which is a well known property.

Running the script of Figure 21.5 produces the output shown in Figure 21.6. One thing to be noticed is the use of stiffness and mass matrix scaling factors, called `kfac` and `mfac`, respectively, in Figure 21.5. These embody symbolic quantities that can be taken out as matrix factors, for example EA/L in $\mathbf{K}^{(e)}$. The effect is to clean up matrix and vector output, as can be observed in Figure 21.6.

§21.2. 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 comnmon in buildings. A plane beam-column element is a combination of a plane bar (such as that considered in §21.1), and a plane beam.

We consider such an element in its local system $(\bar{x}, \bar{y})$ as shown in Figure 21.7, and then in the global system (x, y) as shown in Figure 21.8. The six degrees of freedom and conjugate node forces of the elements are:

$$\bar{\mathbf{u}}^{(e)} = \begin{bmatrix} \bar{u}_{x1} \\ \bar{u}_{y1} \\ \theta_{z1} \\ \bar{u}_{x2} \\ \bar{u}_{y2} \\ \theta_{z2} \end{bmatrix}, \quad \bar{\mathbf{f}}^{(e)} = \begin{bmatrix} \bar{f}_{x1} \\ \bar{f}_{y1} \\ m_{z1} \\ \bar{u}_{x2} \\ \bar{u}_{y2} \\ m_{z2} \end{bmatrix}, \quad \mathbf{u}^{(e)} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_{z1} \\ u_{x2} \\ u_{y2} \\ \theta_{z2} \end{bmatrix}, \quad \mathbf{f}^{(e)} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ m_{z1} \\ f_{x2} \\ f_{y2} \\ m_{z2} \end{bmatrix}. \tag{21.6}$$

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.

The element geometry is described by the coordinates $\{x_i, y_i\}$, $i = 1, 2$ of the two end nodes. The element length is ℓ. The two material properties involved in the stiffness and mass computations are the modulus of elasticity E and the mass density per unit volume ρ. The fabrication properties required are the cross section area A and the bending moment of inertia $I = I_{zz}$ about the neutral axis, which is taken as defining the position of the z axis. All of these properties are taken to be constant over the element.

§21.2.1. Element Formulation

For the plane bar and plane beam components we use the elements derived in Chapters 12 and 13, respectively. (For beam bending this is the Euler-Bernoulli mathematical model.) Combining these two we obtain the stiffness matrix of prismatic beam-column element in the local system $\bar{x}, \bar{y}$ as

$$\bar{\mathbf{K}}^{(e)} = \frac{EA}{\ell} \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & 1 & 0 & 0 \\ & & & & 0 & 0 \\ symm & & & & & 0 \end{bmatrix} + \frac{EI}{\ell^3} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & 12 & 6\ell & 0 & -12 & 6\ell \\ & & 4\ell^2 & 0 & -6\ell & 2\ell^2 \\ & & & 0 & 0 & 0 \\ & & & & 12 & -6\ell \\ symm & & & & & 4\ell^2 \end{bmatrix} \tag{21.7}$$

The first matrix on the right is the contribution from the bar stiffness, but in which rows and columns have been rearranged in accordance with the nodal freedoms (21.6). The second matrix is the contribution from the Bernoulli-Euler bending stiffness; again freedoms have been rearranged as appropriate.

The consistent mass matrix in the local system $\bar{x}, \bar{y}$ is

$$\bar{\mathbf{M}}_C^{(e)} = \frac{\rho A \ell}{6} \begin{bmatrix} 2 & 0 & 0 & 1 & 0 & 0 \\ & 0 & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 & 0 \\ & & & 2 & 0 & 0 \\ & & & & 0 & 0 \\ symm & & & & & 0 \end{bmatrix} + \frac{\rho A \ell}{420} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ & 156 & 21\ell & 0 & 54 & -13\ell \\ & & 4\ell^2 & 0 & 13\ell & -3\ell^2 \\ & & & 0 & 0 & 0 \\ & & & & 156 & -21\ell \\ symm & & & & & 4\ell^2 \end{bmatrix} \tag{21.8}$$

```
PlaneBeamColumn2Stiffness[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,Izz,numer,c,s,L,LL,
   LLL,ra,rb,T,Kebar,Ke},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=mprop; {A,Izz}=fprop; {numer}=opt;
   LL=Simplify[x21^2+y21^2]; L=PowerExpand[Sqrt[LL]]; LLL=L*LL;
   c=x21/L; s=y21/L; ra=Em*A/L; rb= 2*Em*Izz/LLL;
   Kebar= ra*{
  { 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}} +
          rb*{
  { 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}};
   T={{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}};
   Ke=Transpose[T].Kebar.T; If [numer,Ke=N[Ke]];
   Return[Ke]
 ];
```

Figure 21.9. *Mathematica* module to form the stiffness matrix of a 2-node, prismatic plane beam-column element.

The first matrix on the right is the contribution from the axial (bar) inertia, whereas the second one comes from the bending (beam) inertia. The expression for the latter neglect the shear and rotatory inertia. Their derivation may be followed in the book of Przemieniecki cited in footnote 1.

The displacement transformation matrix between local and global systems is

$$\bar{\mathbf{u}}^{(e)} = \begin{bmatrix} \bar{u}_{x1} \\ \bar{u}_{y1} \\ \theta_{z1} \\ u_{x2} \\ \bar{u}_{y2} \\ \theta_{z2} \end{bmatrix} = \begin{bmatrix} c & s & 0 & 0 & 0 & 0 \\ -s & c & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & c & s & 0 \\ 0 & 0 & 0 & -s & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_{z1} \\ u_{x2} \\ u_{y2} \\ \theta_{z2} \end{bmatrix} = \mathbf{T}\,\mathbf{u}^{(e)} \tag{21.9}$$

where $c = \cos\varphi$, $s = \sin\varphi$, and φ is the angle between $\bar{x}$ and x, measured positive-counterclockwise from x; see Figure 21.3. The stiffness and consistent mass matrix in the global system are obtained through the congruential transformation $\mathbf{K}^{(e)} = \mathbf{T}^T\bar{\mathbf{K}}^{(e)}\mathbf{T}$, $\quad \mathbf{M}_C^{(e)} = \mathbf{T}^T\bar{\mathbf{M}}^{(e)}\mathbf{T}$. The lumped mass matrix is diagonal and is the same in the local and global systems:

$$\mathbf{M}_L^{(e)} = \bar{\mathbf{M}}_L^{(e)} = \frac{\rho\ell}{2}\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & m_\theta & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & m_\theta \end{bmatrix}. \tag{21.10}$$

Here the rotational lumped mass is shown as m_θ. There are several ways to compute m_θ, including the simplest one of leaving it zero. In the implementation shown in Figure 21.6, $m_\theta = \rho A\ell^3/78$.

```
PlaneBeamColumn2ConsMass[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,Izz,numer,c,s,L,LL,m,
   T,MeClocal,MeC},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=mprop; {A,Izz}=fprop; {numer}=opt;
   LL=x21^2+y21^2; L=PowerExpand[Sqrt[LL]];
   c=x21/L; s=y21/L; m=ρ*L*A;
   MeClocal= (m/6)*{
    {2,0,0,1,0,0},{0,0,0,0,0,0},{0,0,0,0,0,0},
    {1,0,0,2,0,0},{0,0,0,0,0,0},{0,0,0,0,0,0}}+
        +(m/420)*{
    {0,0,0,0,0,0},{0,156,22*L,0,54,-13*L},
                  {0,22*L,4*LL,0,13*L,-3*LL},
    {0,0,0,0,0,0},{0,54,13*L,0,156,-22*L},
                  {0,-13*L,-3*LL,0,-22*L,4*LL}};
   T={{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}};
   MeC=Transpose[T].MeClocal.T;  If [numer,MeC=N[MeC]];
   Return[MeC]
 ];
PlaneBeamColumn2LumpMass[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,x21,y21,Em,Gm,ρ,α,A,Izz,numer,L,LL,MeL},
  {{x1,y1},{x2,y2}}=ncoor; {x21,y21}={x2-x1,y2-y1};
  {Em,Gm,ρ,α}=mprop; {A,Izz}=fprop;  {numer}=opt;
  LL=Simplify[x21^2+y21^2]; L=PowerExpand[Sqrt[LL]];
  (* HRZ lumping scheme for rotational masses *)
  MeL=(ρ*A*L/2)*
     {{1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,LL/39,0,0,0},
      {0,0,0,1,0,0},{0,0,0,0,1,0},{0,0,0,0,0,LL/39}};
  If [numer,MeL=N[MeL]];
  Return[MeL]
];
```

Figure 21.10. *Mathematica* modules to form mass matrices of a 2-node, prismatic plane beam-column element.

This is called the “HRZ lumping scheme” in the FEM literature.[2]

§21.2.2. Element Formation Modules

Figures 21.9 and 21.10 list three *Mathematica* modules:

`PlaneBeamColumn2Stiffness` returns the element stiffness matrix $\mathbf{K}^{(e)}$ of a 2-node, prismatic plane beam-column element, given by (21.7). `PlaneBeamColumn2ConsMass` returns the consistent mass matrix $\mathbf{M}_C^{(e)}$ of a 2-node, prismatic plane beam-column element, given by (21.8). `PlaneBeamColumn2LumpMass` returns the lumped mass matrix $\mathbf{M}_L^{(e)}$ of a 2-node, prismatic plane beam-column element, given by (21.10).

[2] See e.g., R. D. Cook, D. S. Malkus and M. E. Plesha, *Concepts and Applications of Finite Element Analysis*, 3rd ed., Wiley, 1989.

```
ncoor={{0,0},{3,4}}; mprop={100,0,84/5,0}; fprop={125,250};
opt={True};

Ke= PlaneBeamColumn2Stiffness[ncoor,mprop,fprop,opt];
Print["Numerical Elem Stiff Matrix: "];
Print[Ke//MatrixForm];
Print["Eigenvalues of Ke=",Chop[Eigenvalues[Ke]]];

MeC= PlaneBeamColumn2ConsMass[ncoor,mprop,fprop,opt];
Print["Numerical Consistent Mass Matrix: "];
Print[MeC//MatrixForm];
Print["Eigenvalues of MeC=",Eigenvalues[MeC]];
Print["Squared frequencies (consistent)=",
       Chop[Eigenvalues[Inverse[MeC].Ke],10^(-7)]];

MeL= PlaneBeamColumn2LumpMass[ncoor,mprop,fprop,opt];
Print["Numerical Lumped Mass Matrix: "];
Print[MeL//MatrixForm];
Print["Eigenvalues of MeL=",Eigenvalues[MeL]];
Print["Squared frequencies (lumped)=",
       Chop[Eigenvalues[Inverse[MeL].Ke],10^(-7)]];
```

Figure 21.11. Test of 2-node plane beam-column element with numeric inputs.

```
Numerical Elem Stiff Matrix:
( 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. )
Eigenvalues of Ke={34800., 10000., 5000., 0, 0, 0}
Numerical Consistent Mass Matrix:
( 3756.   -192.  -2200.   1494.   192.    1300. )
( -192.   3644.   1650.   192.    1606.   -975. )
(-2200.   1650.   2500.  -1300.   975.   -1875. )
( 1494.   192.   -1300.   3756.   -192.   2200. )
(  192.   1606.   975.    -192.   3644.  -1650. )
( 1300.   -975.  -1875.   2200.  -1650.   2500. )
Eigenvalues of MeC={9209.32, 5250., 3068.05, 1750., 415.679, 106.949}
Squared frequencies (consistent)={160., 13.7143, 2.85714, 0, 0, 0}
Numerical Lumped Mass Matrix:
(5250.   0.      0.      0.     0.      0.    )
(  0.   5250.    0.      0.     0.      0.    )
(  0.    0.    3365.38   0.     0.      0.    )
(  0.    0.      0.    5250.    0.      0.    )
(  0.    0.      0.      0.   5250.     0.    )
(  0.    0.      0.      0.     0.    3365.38 )
Eigenvalues of MeL={5250., 5250., 5250., 5250., 3365.38, 3365.38}
Squared frequencies (lumped)={9.82857, 2.97143, 0.952381, 0, 0, 0}
```

Figure 21.12. Output from test statements of Figure 21.11.

```
ClearAll[L,Em,ρ,A,Izz,opt];
ncoor={{0,0},{L,0}}; mprop={Em,0,ρ,0}; fprop={A,Izz};
opt={False};

Ke=  PlaneBeamColumn2Stiffness[ncoor,mprop,fprop,opt];
Print["Symbolic Elem Stiff Matrix:"]; kfac=Em*A/L;
Ke=Simplify[Ke/kfac]; Print[kfac," ",Ke//MatrixForm];
Print["Eigenvalues of Ke=",kfac,"*",Eigenvalues[Ke]];

MeC= PlaneBeamColumn2ConsMass[ncoor,mprop,fprop,opt];
Print["Symbolic Consistent Mass Matrix:"]; mfac=ρ*A*L;
MeC= Simplify[MeC/mfac]; Print[mfac," ",MeC//MatrixForm];
Print["Eigenvalues of MeC=",mfac,"*",Eigenvalues[MeC]];
Print["Squared frequencies=",Simplify[kfac/mfac],"*",
       Simplify[Eigenvalues[Inverse[MeC].Ke]]];

MeL= PlaneBeamColumn2LumpMass[ncoor,mprop,fprop,opt];
Print["Symbolic Lumped Mass Matrix:"]; mfac=ρ*A*L;
MeL= Simplify[MeL/mfac]; Print[mfac," ",MeL//MatrixForm];
Print["Eigenvalues of MeL=",Eigenvalues[MeL]];
Print["Squared frequencies=",Simplify[kfac/mfac],"*",
       Simplify[Eigenvalues[Inverse[MeL].Ke]]];
```

Figure 21.13. Test of 2-node plane beam-column element with symbolic inputs.

As in the case of the plane bar, the calling argument sequence of the three modules is exactly the same:

`[ ncoor, mprop, fprop, opt ]`

These four arguments are *lists* that collect the following data:

node coordinates, material properties, fabrication properties, options

The internal structure of these lists is

```
ncoor    {{x1,y1},{x2,y2}}
mprop    {Em,Gm,rho,alpha}
fprop    {A,Izz}                        (21.11)
opt      {num}
```

Here `x1,y1` and `x2,y2` are the coordinates of the end nodes, while `Em`, `Gm`, `rho`, `alpha`, `A` and `Izz` stand for E, G, ρ, α, A and I_{zz}, respectively. For the stiffness matrix only E, A and I_{zz} are used. In the mass matrix modules only ρ and A are used. Entries `Gm` and `alpha` are included to take care of other 1D elements.

The only option is `numer`, which is a logical flag with the value `True` or `False`. If `numer=True` the computations are carried out in numerical floating-point arithmetic.

§21.2.3. Testing the Beam-Column Element Modules

The modules are tested by the statements collected in the scripts of Figures 21.11 and 21.13.

Symbolic Elem Stiff Matrix:

$$\frac{\mathrm{A\,Em}}{\mathrm{L}}\begin{pmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & \frac{12\,\mathrm{Izz}}{\mathrm{A\,L}^2} & \frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & 0 & -\frac{12\,\mathrm{Izz}}{\mathrm{A\,L}^2} & \frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} \\ 0 & \frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & \frac{4\,\mathrm{Izz}}{\mathrm{A}} & 0 & -\frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & \frac{2\,\mathrm{Izz}}{\mathrm{A}} \\ -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & -\frac{12\,\mathrm{Izz}}{\mathrm{A\,L}^2} & -\frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & 0 & \frac{12\,\mathrm{Izz}}{\mathrm{A\,L}^2} & -\frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} \\ 0 & \frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & \frac{2\,\mathrm{Izz}}{\mathrm{A}} & 0 & -\frac{6\,\mathrm{Izz}}{\mathrm{A\,L}} & \frac{4\,\mathrm{Izz}}{\mathrm{A}} \end{pmatrix}$$

Eigenvalues of Ke= $\frac{\mathrm{A\,Em}}{\mathrm{L}}*\left\{0,\ 0,\ 0,\ 2,\ \frac{2\,\mathrm{Izz}}{\mathrm{A}},\ \frac{6\,(4\,\mathrm{Izz}+\mathrm{Izz\,L}^2)}{\mathrm{A\,L}^2}\right\}$

Symbolic Consistent Mass Matrix:

$$\mathrm{A\,L}\,\rho\begin{pmatrix} \frac{1}{3} & 0 & 0 & \frac{1}{6} & 0 & 0 \\ 0 & \frac{13}{35} & \frac{11\,\mathrm{L}}{210} & 0 & \frac{9}{70} & -\frac{13\,\mathrm{L}}{420} \\ 0 & \frac{11\,\mathrm{L}}{210} & \frac{\mathrm{L}^2}{105} & 0 & \frac{13\,\mathrm{L}}{420} & -\frac{\mathrm{L}^2}{140} \\ \frac{1}{6} & 0 & 0 & \frac{1}{3} & 0 & 0 \\ 0 & \frac{9}{70} & \frac{13\,\mathrm{L}}{420} & 0 & \frac{13}{35} & -\frac{11\,\mathrm{L}}{210} \\ 0 & -\frac{13\,\mathrm{L}}{420} & -\frac{\mathrm{L}^2}{140} & 0 & -\frac{11\,\mathrm{L}}{210} & \frac{\mathrm{L}^2}{105} \end{pmatrix}$$

Squared frequencies= $\frac{\mathrm{Em}}{\mathrm{L}^2\,\rho}*\left\{0,\ 0,\ 0,\ 12,\ \frac{720\,\mathrm{Izz}}{\mathrm{A\,L}^2},\ \frac{8400\,\mathrm{Izz}}{\mathrm{A\,L}^2}\right\}$

Symbolic Lumped Mass Matrix:

$$\mathrm{A\,L}\,\rho\begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{\mathrm{L}^2}{78} & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{\mathrm{L}^2}{78} \end{pmatrix}$$

Squared frequencies= $\frac{\mathrm{Em}}{\mathrm{L}^2\,\rho}*\left\{0,\ 0,\ 0,\ 4,\ \frac{156\,\mathrm{Izz}}{\mathrm{A\,L}^2},\ \frac{516\,\mathrm{Izz}}{\mathrm{A\,L}^2}\right\}$

Figure 21.14. Results from test statements of Figure 21.13. Output of eigenvalues of mass matrices deleted to save space.

The script of Figure 21.7 tests a numerically defined element of length $\ell = 5$ with end nodes located at $(0, 0)$ and $(3, 4)$ respectively, with $E = 100$, $\rho = 84/5$, $A = 125$ and $I_{zz} = 250$. The output is shown in Figure 21.12. The tests consist of the following operations:

Testing $\mathbf{K}^{(e)}$. 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.

Testing $\mathbf{M}_C^{(e)}$. The consistent mass matrix returned in `MeC` is printed. Its symmetry is checked. The six eigenvalues are computed and printed. As expected they are all positive. A frequency test is also carried out.

Testing $\mathbf{M}_L^{(e)}$. The lumped mass matrix returned in `MeL` is printed. This is a diagonal matrix and thus its eigenvalues are the same as the diagonal entries. A frequency test is also carried out.

The script of Figure 21.13 tests a symbolically defined element with end nodes located at $(0, 0)$ and

$(L, 0)$, which is aligned with the x axis. The element properties E, ρ, A and I_{zz} are kept in symbolic form. The output is shown in shown in Figure 21.14, except that the output of the eigenvalues of the mass matrices has been deleted to save space.

The sequence of tests on the symbolic element is similar to that performed for the bar element in Figure 21.4. The vibration eigenproblems are

$$\mathbf{K}^{(e)}\mathbf{v}_i = \omega_i^2 \mathbf{M}_C^{(e)}\mathbf{v}_i, \qquad \mathbf{K}^{(e)}\mathbf{v}_i = \omega_i^2 \mathbf{M}_L^{(e)}\mathbf{v}_i, \tag{21.12}$$

where ω_i are circular frequencies and $\mathbf{v}_i$ the associated eigenvectors or vibration mode shapes. Because the three rigid body modes are solution of (21.12), three zero frequencies are expected for the consistent mass eigenproblem, which is borned out by the tests. The three positive nonzero frequencies corresponds to one free-free axial and two free-free bending modes. The consistent mass gives for the latter $\omega^2 = 720EI/(\rho AL\ell^4)$ and $\omega^2 = 8400EI/(\rho A\ell^4)$. These are upper bounds to the exact continuum solution for the first two free-free bending frequencies, which are are $502EI/(\rho A\ell^4)$ and $1382EI/(\rho A\ell^4)$.

The lumped mass vibration eigenproblem, using the HRZ lumping scheme for the rotational masses, yields three zero frequencies, one finite positive axial vibration frequency, which is the same as that provided by the bar element, and two bending frequencies $\omega^2 = 156EI/(\rho AL\ell^4)$ and $\omega^2 = 516EI/(\rho A\ell^4)$. These are lower bounds to the exact free-free continuum frequencies given above.

§21.3. THE SPACE BAR ELEMENT

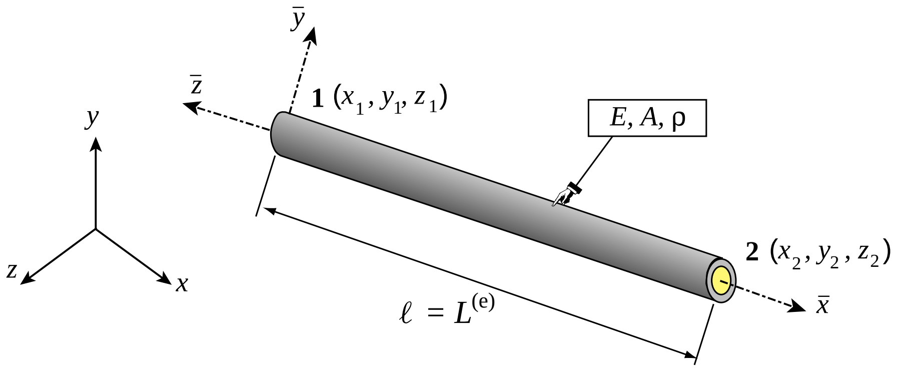


Figure 21.15. The space (3D) bar element.

To provide a taste of the world of space structures, this section outlines the extension of the bar element to three dimensions.

The two-node, prismatic, three-dimensional bar element is shown in Figure 21.15. It has two nodes and six degrees of freedom. The element node displacements and congugate forces are

$$\mathbf{u}^{(e)} = \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{z1} \\ u_{x2} \\ u_{y2} \\ u_{z2} \end{bmatrix}, \quad \mathbf{f}^{(e)} = \begin{bmatrix} f_{x1} \\ f_{y1} \\ f_{z1} \\ f_{x2} \\ f_{y2} \\ f_{z1} \end{bmatrix}. \tag{21.13}$$

22

FEM Programs for Plane Trusses and Frames

TABLE OF CONTENTS

§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 assignment.

The description is done in "bottom up" fashion. That means the basic modules are presented, then the driver program. Graphic modules are provided in a posted Notebook but not described.

§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 **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 **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: `{{x1,y1},{x2,y2}, ... {xn,yn}}`, where `n` is the total number of nodes.
`elenod`	Element end nodes arranged as a two-dimensional list: `{{i1,j1}, {i2,j2}, ... {ie,je}}`, where `e` is the total number of elements.
`elemat`	Element material properties arranged as a two-dimensional list: `{{Em1,Gm1,rho1,alpha1}, ... {Eme,Gme,rhoe,alphae}}`, where `e` is the total number of elements. Only the elastic modulus `Em` is used in this

```
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.

program. For the other properties zeros may be stored as placeholders.

`elefab`	Element fabrication properties arranged as a two-dimensional list: `{ { A1 }, ... { Ae } }`, where `e` is the total number of elements, and `A` the cross section area.
`eleopt`	Element processing option: set to `{ True }` to tell `PlaneBar2Stiffness` to carry out element stiffness computations in floating-point arithmetic. Else set to `{ 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 **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 **K**, whereas `ModifiedNodalForces` does this for the nodal force vector **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 **K**, places ones on the diagonal, and returns the thus modified **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 **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:

$$\{\texttt{ux1, uy1 ... uxn, uyn}\} \tag{22.1}$$

`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.

```
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.

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.

§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

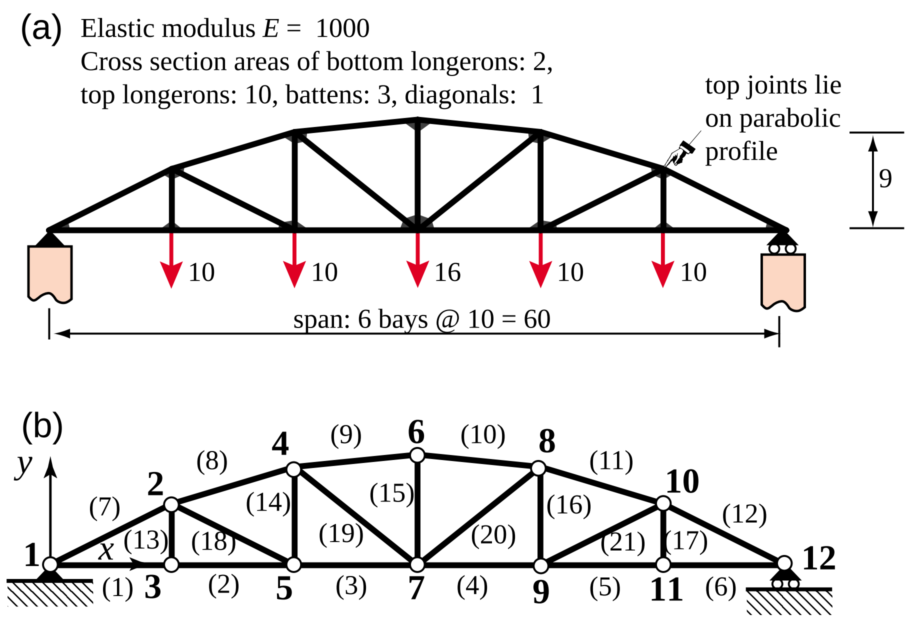


Figure 22.4. Six-bay bridge truss used as example problem: (a) truss structure showing supports and applied loads; (b) finite element idealization as pin-jointed truss.

```
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.

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 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=LinearSolve[Kmod,fmod]` (22.2)

which takes advantage of the built-in linear solver provided by `Mathematica`.

```
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.

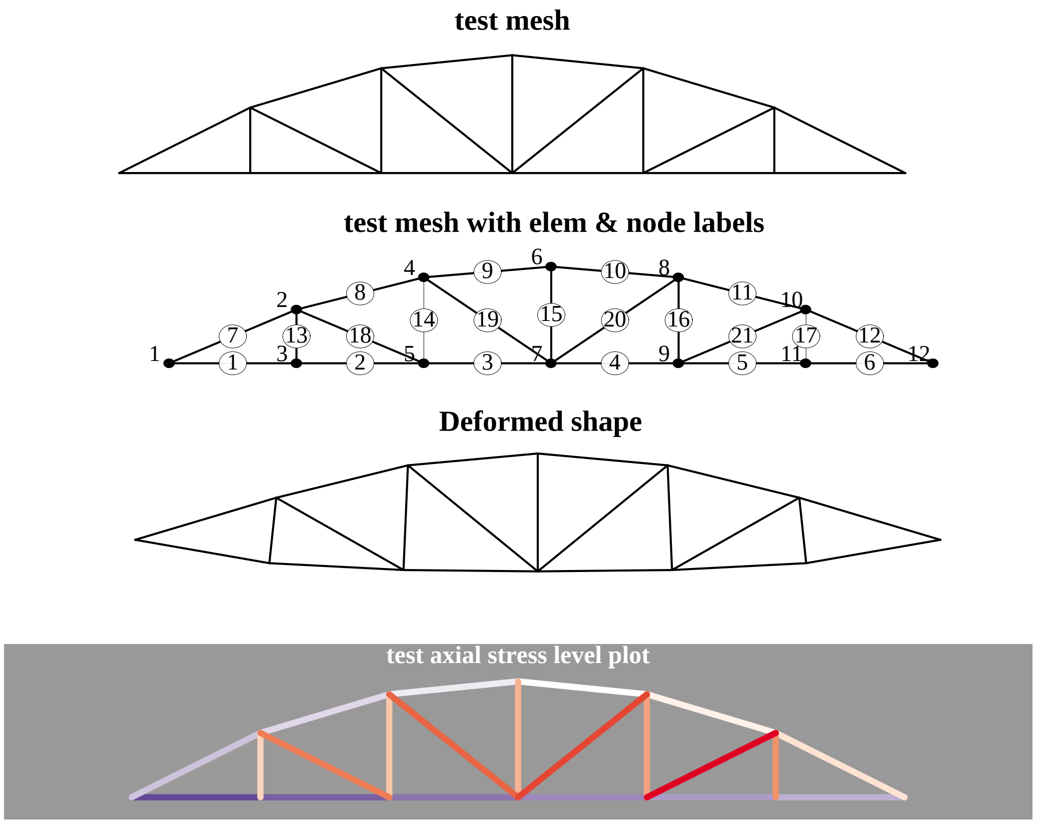

test mesh
test mesh with elem & node labels
Deformed shape
test axial stress level plot

Figure 22.8. Graphics output from the bridge truss analysis.

The remaining calculations recover the node vector including reactions by the matrix-vector multiply `f = K.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.

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. Test the assembler on a simple beam problem such as that shown in Figure E22.1.

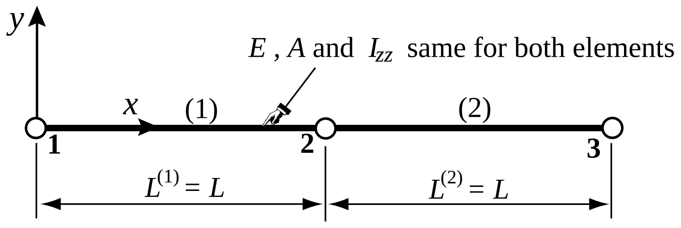


Figure E22.1. Suggested beam assembly tester.

The assembled stiffness should be

$$\begin{bmatrix}
EA/L & 0 & 0 & -EA/L & 0 & 0 & 0 & 0 & 0 \\
0 & 12EI/L^3 & 6EI/L^2 & 0 & -12EI/L^3 & 6EI/L^2 & 0 & 0 & 0 \\
 & & 4EI/L & 0 & -6EI/L^2 & 2EI/L & 0 & 0 & 0 \\
 & & & 2EA/L & 0 & 0 & -EA/L & 0 & 0 \\
 & & & & 24EI/L^3 & 0 & 0 & -12EI/L^3 & 6EI/L^2 \\
 & & & & & 8EI/L & 0 & -6EI/L^2 & 2EI/L \\
 & & & & & & EA/L & 0 & 0 \\
 & & & & & & 0 & 12EI/L^3 & -6EI/L^2 \\
symm & & & & & & & & 4EI/L
\end{bmatrix} \qquad (E22.1)$$

[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.

Cell 2: BC Applicator. No modifications are necessary. The two modules should work correctly for this problem since they dont 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.

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 E22.2. The frame is fixed[3] 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.

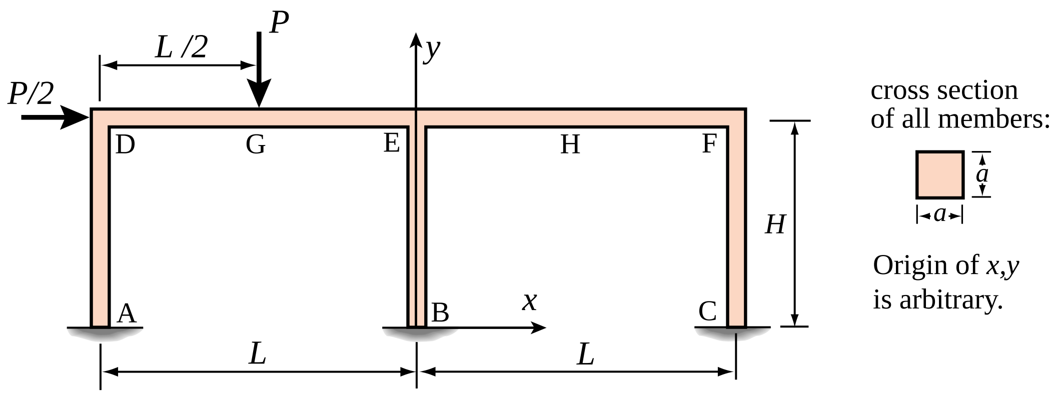


Figure E22.2. Structure for Exercise 22.2.

The SI physical units to be used are: mm for lengths, N for forces, and M Pa=N/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 E22.3.

As solution to this exercise list the driver program you used and the displacement and internal force outputs. The results of primary interest are:

[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.

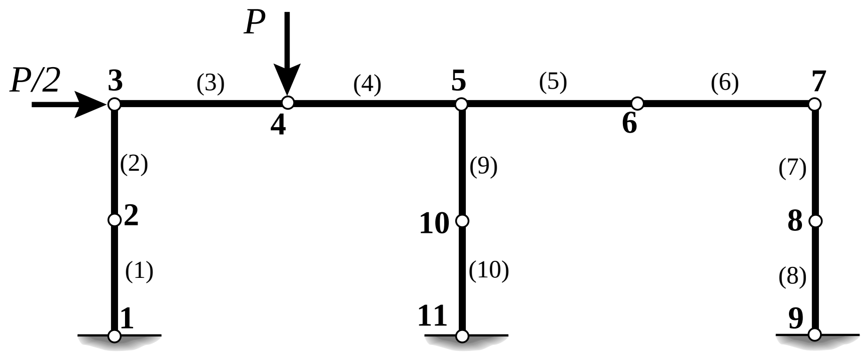


Figure E22.3. Recommended FEM discretization for Exercise 22.2.

1. The horizontal displacement at D, and the vertical displacement at G (mm).
2. The axial forces (N) in columns AD, BE and FG, with appropriate sign[4].
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 FG.[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, to be used as targets.

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.

[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.

23 Implementation of Iso-P Quadrilateral Elements

TABLE OF CONTENTS

§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 and consistent node force vector for a body force field. The organization of the computations is typical of isoparametric-element modules in any number of space dimensions.

§23.2. IMPLEMENTATION OF THE BILINEAR QUADRILATERAL

We consider the implementation of the 4-node blinear quadrilateral for plane stress, depicted in Figure 23.1.

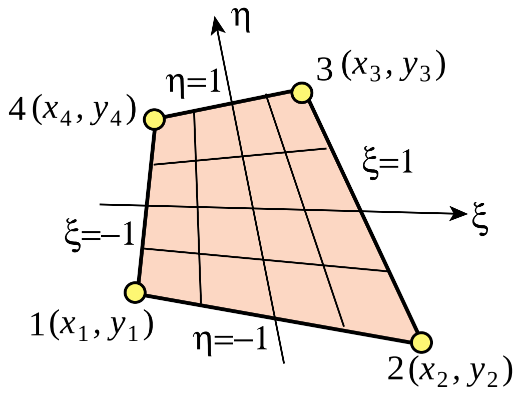


Figure 23.1. The 4-node bilinear quadrilateral element.

The element stiffness matrix of simple one-dimensional elements, such as the ones discussed in Chapters 21 and 22, can be easily packaged in a simple module. For 2D and 3D elements, however, it is convenient to break up the implementation into *application dependent* and *application independent* modules, as sketched in Figure 23.2. The application independent modules can be "reused" in other FEM applications, for example to form thermal, fluid or electromagnetic elements.

For the 4-node quadrilateral studied here, the subdivision of Figure 23.2 is done through the following modules:

```
Stiffness4NodePlaneStressQuad - forms Ke of 4-node bilinear quad in plane stress
   QuadGaussRuleInfo -          returns Gauss quadrature product rules of order 1-4
   IsoQuad4ShapeFunctions -     evaluates shape functions and their x/y derivatives
```

These modules are described in further detail in the following subsections, in a "bottom up" fashion.

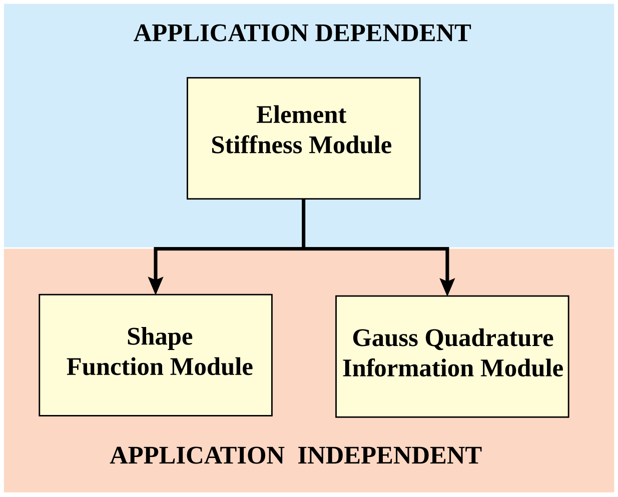


Figure 23.2. Organization of element stiffness modules.

```
QuadGaussRuleInfo[{rule_,numer_},point_]:= Module[
 {xi,eta,p1,p2,i1,i2,w1,w2,k,info=Null},
  If [Length[rule] ==2, {p1,p2}=rule, p1=p2=rule];
  If [Length[point]==2, {i1,i2}=point,
      k=point; i2=Floor[(k-1)/p1]+1; i1=k-p1*(i2-1) ];
  {xi, w1}=  LineGaussRuleInfo[{p1,numer},i1];
  {eta,w2}=  LineGaussRuleInfo[{p2,numer},i2];
  info={{xi,eta},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]},
   i,info=Null},  i=point;
   If [rule==1, info={0,2}];
   If [rule==2, info={g2[[i]],1}];
   If [rule==3, info={g3[[i]],w3[[i]]}];
   If [rule==4, info={g4[[i]],w4[[i]]}];
   If [numer, Return[N[info]], Return[Simplify[info]]];
];
```

Figure 23.3. Module to get Gauss-product quadrature information for a quadrilateral.

§23.2.1. Gauss Quadrature Rule Information

Recall from §17.3 that Gauss quadrature rules for isoparametric quadrilateral elements have the canonical form

$$\int_{-1}^{1}\int_{-1}^{1} \mathbf{F}(\xi,\eta)\,d\xi\,d\eta = \int_{-1}^{1} d\eta \int_{-1}^{1} \mathbf{F}(\xi,\eta)\,d\xi \doteq \sum_{i=1}^{p_1}\sum_{j=1}^{p_2} w_i w_j \mathbf{F}(\xi_i,\eta_j). \tag{23.1}$$

Here $\mathbf{F} = h\mathbf{B}^T\mathbf{E}\mathbf{B}\,J$ is the matrix to be integrated, and 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 QuadGaussRuleInfo that implements the two-dimensional product Gauss rules with 1 through 4 points in each direction. The number of points in each direction may be the same or different. Use 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.

```
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; J21=dNξ.y; J12=dNη.x; J22=dNη.y;
   Jdet=Simplify[J11*J22-J12*J21];
   dNx= ( J22*dNξ-J21*dNη)/Jdet;  dNx=Simplify[dNx];
   dNy= (-J12*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 as function value 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.

```
Quad4IsoPMembraneStiffness[ncoor_,mprop_,fprop_,options_]:=
  Module[{i,k,p=2,numer=False,Emat,th=1,h,qcoor,c,w,Nf,
   dNx,dNy,Jdet,B,Ke=Table[0,{8},{8}]},  Emat=mprop[[1]];
  If [Length[options]==2, {numer,p}=options, {numer}=options];
  If [Length[fprop]>0, th=fprop[[1]]];
  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]==0, h=th, h=th.Nf]; c=w*Jdet*h;
        B={ 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[B].(Emat.B)];
      ]; Return[Ke]
   ];
```

Figure 23.5. Element stiffness formation module for 4-node bilinear quadrilateral.

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 Figure 23.5) test it for a specific configuration.

The arguments of the module are:

`ncoor` Quadrilateral node coordinates arranged in two-dimensional list form: `{ { x1,y1 },{ x2,y2 },{ x3,y3 },{ x4,y4 } }`.

`mprop` Material properties supplied as the list `{ Emat,rho,alpha }`. `Emat` is a two-dimensional list storing the 3 × 3 plane stress matrix of elastic moduli:

$$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \tag{23.2}$$

If the material is isotropic with elastic modulus E and Poisson's ratio ν, this matrix becomes

$$\mathbf{E} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{bmatrix} \tag{23.3}$$

The other two items in `mprop` are not used in this module so zeros may be inserted as placeholders.

`fprop` Fabrication properties. The plate thickness specified as a four-entry list: `{ h1,h2,h3,h4 }`, a one-entry list: `{ h }`, or an empty list: { }.

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 `h`. If an empty list appears the module assumes a uniform unit 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 forced to proceed in floating point arithmetic. For symbolic or exact arithmetic work set `numer` to `False`.

`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. If omitted `p` = 2 is assumed.

The module returns `Ke` as an 8×8 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4} \,]^T . \tag{23.4}$$

§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. Both geometries will be used to illustrate the effect of the numerical integration rule.

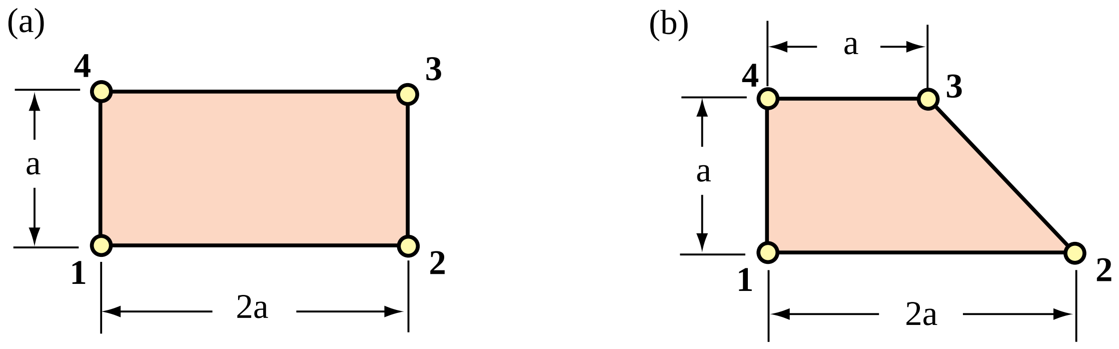


Figure 23.6. Test quadrilateral element geometries.

§23.3.1. Test of Rectangular Geometry

The test statements of Figure 23.7 compute and print the stiffness of the rectangular element shown in Figure 23.6(a). This is a rectangle of base $2a$ and height a. The element has unit thickness and isotropic material with $E = 96$ and $\nu = 1/3$, giving the stress-strain constitutive matrix

$$\mathbf{E} = \begin{bmatrix} 108 & 36 & 0 \\ 36 & 108 & 0 \\ 0 & 0 & 36 \end{bmatrix} \tag{23.5}$$

```
ClearAll[Em,nu,a,b,e,h,p,num];  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 *) num=False; (* exact symbolic arithmetic *)
Ke=Quad4IsoPMembraneStiffness[ncoor,{Emat,0,0},{h},{num,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).

Using a 2 × 2 Gauss integration rule returns the stiffness matrix

$$\mathbf{K}^{(e)} = \begin{bmatrix} 42 & 18 & -6 & 0 & -21 & -18 & -15 & 0 \\ 18 & 78 & 0 & 30 & -18 & -39 & 0 & -69 \\ -6 & 0 & 42 & -18 & -15 & 0 & -21 & 18 \\ 0 & 30 & -18 & 78 & 0 & -69 & 18 & -39 \\ -21 & -18 & -15 & 0 & 42 & 18 & -6 & 0 \\ -18 & -39 & 0 & -69 & 18 & 78 & 0 & 30 \\ -15 & 0 & -21 & 18 & -6 & 0 & 42 & -18 \\ 0 & -69 & 18 & -39 & 0 & 30 & -18 & 78 \end{bmatrix} \tag{23.6}$$

Note that the rectangle dimension a does not appear in (23.6). This is a general property: *the stiffness matrix of plane stress elements is independent of inplane dimension scalings*. This follows from the fact that entries of the strain-displacement matrix $\mathbf{B}$ have dimensions $1/L$, where L denotes a 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 (23.6). 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 (23.6) has the eigenvalues

$$[\,223.64 \quad 90 \quad 78 \quad 46.3603 \quad 42 \quad 0 \quad 0 \quad 0\,] \tag{23.7}$$

which verifies that $\mathbf{K}^{(e)}$ has the correct rank of five (8 freedoms minus 3 rigid body modes).

§23.3.2. Test of Trapezoidal Geometry

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.

```
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,0,0},{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 computations are driven with the script shown in Figure 23.8. The value of `p` is changed in a loop. The flag `numer` 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:

$$\mathbf{K}^{(e)}_{1\times1} = \begin{bmatrix} 1840293 & 1051596 & -262899 & -262899 & -1840293 & -1051596 & 262899 & 262899 \\ 1051596 & 3417687 & -262899 & 1314495 & -1051596 & -3417687 & 262899 & -1314495 \\ -262899 & -262899 & 1051596 & -525798 & 262899 & 262899 & -1051596 & 525798 \\ -262899 & 1314495 & -525798 & 1051596 & 262899 & -1314495 & 525798 & -1051596 \\ -1840293 & -1051596 & 262899 & 262899 & 1840293 & 1051596 & -262899 & -262899 \\ -1051596 & -3417687 & 262899 & -1314495 & 1051596 & 3417687 & -262899 & 1314495 \\ 262899 & 262899 & -1051596 & 525798 & -262899 & -262899 & 1051596 & -525798 \\ 262899 & -1314495 & 525798 & -1051596 & -262899 & 1314495 & -525798 & 1051596 \end{bmatrix} \tag{23.8}$$

$$\mathbf{K}^{(e)}_{2\times2} = \begin{bmatrix} 2062746 & 1092042 & -485352 & -303345 & -1395387 & -970704 & -182007 & 182007 \\ 1092042 & 3761478 & -303345 & 970704 & -970704 & -2730105 & 182007 & -2002077 \\ -485352 & -303345 & 1274049 & -485352 & -182007 & 182007 & -606690 & 606690 \\ -303345 & 970704 & -485352 & 1395387 & 182007 & -2002077 & 606690 & -364014 \\ -1395387 & -970704 & -182007 & 182007 & 2730105 & 1213380 & -1152711 & -424683 \\ -970704 & -2730105 & 182007 & -2002077 & 1213380 & 4792851 & -424683 & -60669 \\ -182007 & 182007 & -606690 & 606690 & -1152711 & -424683 & 1941408 & -364014 \\ 182007 & -2002077 & 606690 & -364014 & -424683 & -60669 & -364014 & 2426760 \end{bmatrix} \tag{23.9}$$

$$\mathbf{K}^{(e)}_{3\times3} = \begin{bmatrix} 2067026 & 1093326 & -489632 & -304629 & -1386827 & -968136 & -190567 & 179439 \\ 1093326 & 3764046 & -304629 & 968136 & -968136 & -2724969 & 179439 & -2007213 \\ -489632 & -304629 & 1278329 & -484068 & -190567 & 179439 & -598130 & 609258 \\ -304629 & 968136 & -484068 & 1397955 & 179439 & -2007213 & 609258 & -358878 \\ -1386827 & -968136 & -190567 & 179439 & 2747225 & 1218516 & -1169831 & -429819 \\ -968136 & -2724969 & 179439 & -2007213 & 1218516 & 4803123 & -429819 & -70941 \\ -190567 & 179439 & -598130 & 609258 & -1169831 & -429819 & 1958528 & -358878 \\ 179439 & -2007213 & 609258 & -358878 & -429819 & -70941 & -358878 & 2437032 \end{bmatrix} \tag{23.10}$$

$$\mathbf{K}^{(e)}_{4\times4} = \begin{bmatrix} 2067156 & 1093365 & -489762 & -304668 & -1386567 & -968058 & -190827 & 179361 \\ 1093365 & 3764124 & -304668 & 968058 & -968058 & -2724813 & 179361 & -2007369 \\ -489762 & -304668 & 1278459 & -484029 & -190827 & 179361 & -597870 & 609336 \\ -304668 & 968058 & -484029 & 1398033 & 179361 & -2007369 & 609336 & -358722 \\ -1386567 & -968058 & -190827 & 179361 & 2747745 & 1218672 & -1170351 & -429975 \\ -968058 & -2724813 & 179361 & -2007369 & 1218672 & 4803435 & -429975 & -71253 \\ -190827 & 179361 & -597870 & 609336 & -1170351 & -429975 & 1959048 & -358722 \\ 179361 & -2007369 & 609336 & -358722 & -429975 & -71253 & -358722 & 2437344 \end{bmatrix} \tag{23.11}$$

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

(23.12)

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 stiffens the element.

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 high 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.

```
Quad4IsoPMembraneBodyForces[ncoor_,mprop_,fprop_,options_,bfor_]:=
  Module[{i,k,p=2,numer=False,Emat,th=1,h,
    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[fprop]>0, th=fprop[[1]]];
  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];

       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.

§23.4. CONSISTENT NODE FORCES FOR BODY FORCE FIELD

The module `Quad4IsoPMembraneBodyForces` listed in Figure 23.8 computes the consistent force associated with a body force field $\vec{\mathbf{b}} = \{b_x, b_y\}$ given over a four-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 are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

`mprop`	Not used; retained as placeholder.
`bfor`	Body forces per unit volume. Specified as a two-item one-dimensional list: { `bx,by` }, or as a four-entry two-dimensional list: { `bx1,by1` },{ `bx2,by2` }, { `bx3,by3` },{ `bx4,by4` }. 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 { `fx1,fy1,fx2,fy2,fx3,fy3, fx4,fy4` } to represent the vector

$$\mathbf{f}^{(e)} = [\ f_{x1} \quad f_{y1} \quad f_{x2} \quad f_{y2} \quad f_{x3} \quad f_{y3} \quad f_{x4} \quad f_{y4}\ ]^T . \tag{23.13}$$

```
Quad4IsoPMembraneStresses[ncoor_,mprop_,fprop_,options_,udis_]:=
  Module[{i,k,numer=False,Emat,th=1,h,qcoor,Nf,
    dNx,dNy,Jdet,B,qctab,ue=udis,sige=Table[0,{4},{3}]},
  qctab={{-1,-1},{1,-1},{1,1},{-1,1}};
  Emat=mprop[[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];
        B={ 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.(B.ue);
      ]; Return[sige]
   ];
```

Figure 23.10. Module for calculation of corner stresses.

§23.5. RECOVERY OF CORNER STRESSES

Akthough the subject of stress recovery is treated in further detail in a later chapter, for completeness a stress computation module for the 4-node quad is shown in Figure 23.10.

The arguments of the module are are exactly the same as for `Quad4IsoPMembraneStiffness` except for the following differences.

`fprop`	Not used; retained as placeholder.
`udis`	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

$$\sigma^{(e)} = \begin{bmatrix} \sigma_{xx1} & \sigma_{xx2} & \sigma_{xx3} & \sigma_{xx4} \\ \sigma_{yy1} & \sigma_{yy2} & \sigma_{yy3} & \sigma_{yy4} \\ \sigma_{xy1} & \sigma_{xy2} & \sigma_{xy3} & \sigma_{xy4} \end{bmatrix} \tag{23.14}$$

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.

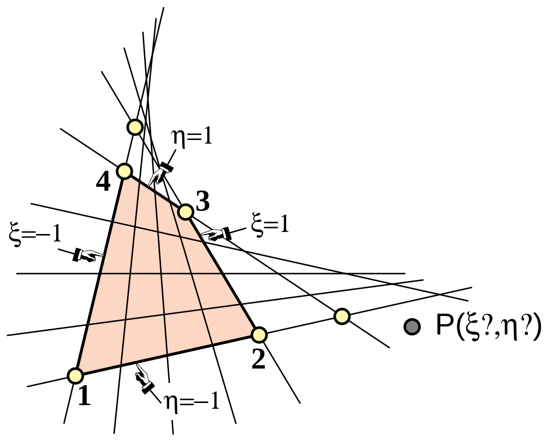


Figure 23.11. Quadrilateral coordinates can be extended outside the element to answer the problem posed in §23.4. In this figure the six yellow-fill circles identify the four corners 1-2-3-4 plus the two points where opposite sides intersect. This six-point set defines the so-called complete quadrilateral, which is important in projective geometry. The evolute of the coordinate lines is a parabola.

§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 $x_P = x_1N_1 + x_2N_2 + x_3N_3 + x_4N_4$ and $y_P = y_1N_1 + y_2N_2 + y_3N_3 + y_4N_4$, 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.

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.[1] $\eta_P^{min} = b/(b + \sqrt{b^2 - 4ac})$, 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

[1] See Press et al. *Numerical Recipes: The Art of Scientific Computing*, 2nd ed., Cambridge Univ. Press, 1992, §5.6.

$$\boxed{\begin{array}{l}
x_b = x_1 - x_2 + x_3 - x_4, \quad y_b = y_1 - y_2 + y_3 - y_4, \quad x_{cx} = x_1 + x_2 - x_3 - x_4, \quad y_{cx} = y_1 + y_2 - y_3 - y_4,\\
x_{ce} = x_1 - x_2 - x_3 + x_4, \quad y_{ce} = y_1 - y_2 - y_3 + y_4, \quad A = \tfrac{1}{2}((x_3 - x_1)(y_4 - y_2) - (x_4 - x_2)(y_3 - y_1)),\\
J_1 = (x_3 - x_4)(y_1 - y_2) - (x_1 - x_2)(y_3 - y_4), \quad J_2 = (x_2 - x_3)(y_1 - y_4) - (x_1 - x_4)(y_2 - y_3),\\
x_0 = \tfrac{1}{4}(x_1 + x_2 + x_3 + x_4), \quad y_0 = \tfrac{1}{4}(y_1 + y_2 + y_3 + y_4), \quad x_{P0} = x_P - x_0, \quad y_{P0} = y_P - y_0,\\
b_\xi = A - x_{P0}\, y_b + y_{P0}\, x_b, \quad b_\eta = -A - x_{P0}\, y_b + y_{P0} x_b, \quad c_\xi = x_{P0}\, y_{cx} - y_{P0}\, x_{cx},\\
c_\eta = x_{P0}\, y_{ce} - y_{P0}\, x_{ce}, \quad \xi_P = \dfrac{2c_\xi}{-\sqrt{b_\xi^2 - 2J_1 c_\xi} - b_\xi}, \quad \eta_P = \dfrac{2c_\eta}{\sqrt{b_\eta^2 + 2J_2 c_\eta} - b_\eta}.
\end{array}} \tag{23.15}$$

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 locations[2] to an existing FEM mesh.

A *Mathematica* module that implements (23.15) is listed in Figure 23.12.

```
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 23.11. A *Mathematica* module implementing the algorith (23.15).

[2] 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.

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: `{ { x1,y1 },{ x2,y2 },{ x3,y3 },{ x4,y4 },{ x5,y5 } }`, and that a variable plate thickness in `fprop` (one of the 3 possible formats) is specified as `{ h1,h2,h3,h4,h5 }`.

```
  Quad5IsoPMembraneStiffness[ncoor_,mprop_,fprop_,options_]:=
    Module[{i,j,k,p=2,numer=False,Emat,th=1,h,qcoor,c,w,Nf,
     dNx,dNy,Jdet,B,Ke=Table[0,{10},{10}]},
    Emat=mprop[[1]];
    If [Length[options]==2, {numer,p}=options, {numer}=options];
    If [Length[fprop]>0, th=fprop[[1]]];
    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, 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; J21=dNξ.y; J12=dNη.x; J22=dNη.y;
   Jdet=Simplify[J11*J22-J12*J21];
   dNx= ( J22*dNξ-J21*dNη)/Jdet;  dNx=Simplify[dNx];
   dNy= (-J12*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.

```
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.

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 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 `Ke` 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 `Ke5` using `Quad5IsoPMembraneStiffness` with $p = 2$ and `numer=False`. Insert this `Ke5` into `Quad5IsoPMembraneCondStiffness`, which returns a 8×8 stiffness `Ke`. Stick this `Ke` into equations (E17.6) and (E17.7) to get U_{quad}. Show that the energy ratio is

$$r = \frac{U_{quad}}{U_{beam}} = \frac{\gamma^2(1+\nu)\big(2+\gamma^2(1-\nu)\big)}{(1+\gamma^2)^2}. \tag{23.16}$$

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.[3]

EXERCISE 23.4

[C:25] Implement the 9-node biquadratic 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.

[3] 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.

24
Implementation of Iso-P Triangular Elements

TABLE OF CONTENTS

§24.1. INTRODUCTION

This Chapter continues with the subject of 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 each of these two differences in turn.

§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:

$$\boxed{\text{If the sample point } (\zeta_1, \zeta_2, \zeta_3) \text{ is present in a Gauss integration rule with weight } w\text{, then all other points obtainable by permuting the three triangular coordinates arbitrarily must appear in that rule, and have the same weight.}} \qquad (24.1)$$

This rule 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, condition (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 the condition 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:

$$\boxed{\text{All sample points must be inside the triangle (or on the triangle boundary) and all weights must be positive.}} \qquad (24.2)$$

These are called *numerical stability* conditions.

REMARK 24.1

The reasons for the stability conditions (24.2) cannot be covered in an elementary course. They are automatically satisfied by all Gauss product rules for quadrilaterals, and so it was not necessary to call attention to them. On the other hand, for triangles there are Gauss rules with as few as 4 and 6 points that violate those conditions.

[1] It would disconcerting to users, to say the least, to have the FEM solution depend on how nodes are numbered.

[2] As a consequence, the number of sample points in triangle Gauss quadrature rules 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.

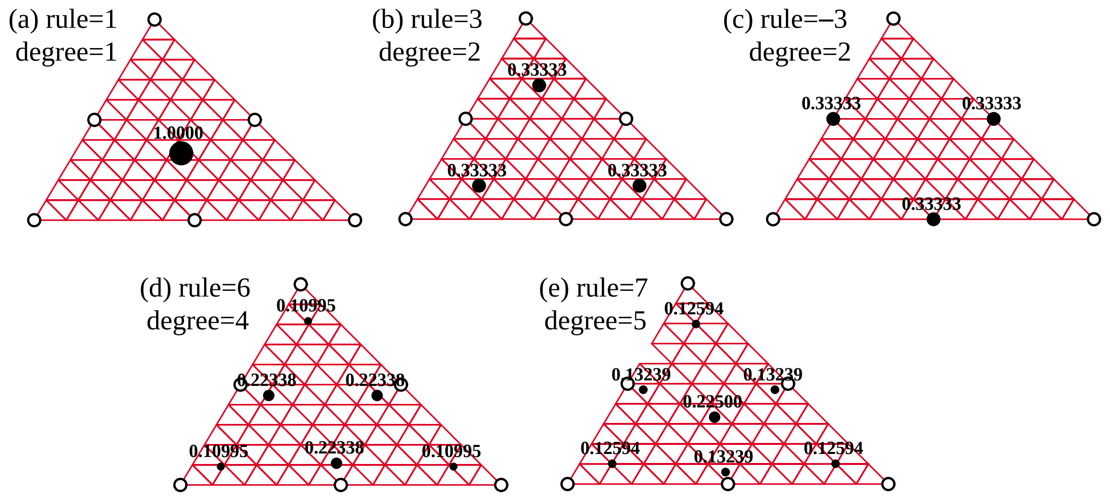


Figure 24.1. Location of sample points (dark circles) of five Gauss quadrature rules for straight sided (superparametric) 6-node triangles. Weight written to 5 places near each sample point; sample-point circle areas are proportional to weight.

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.

§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 in §24.3 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 quadratic 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,

$$\frac{1}{A}\int_{\Omega^{(e)}} F(\zeta_1,\zeta_2,\zeta_3)\,d\Omega^{(e)} \approx F(\tfrac{1}{3},\tfrac{1}{3},\tfrac{1}{3}), \tag{24.3}$$

where A is the triangle area:

$$A=\int_{\Omega^{(e)}} d\Omega^{(e)}=\tfrac{1}{2}\det\begin{bmatrix}1&1&1\\x_1&x_2&x_3\\y_1&y_2&y_3\end{bmatrix}=\tfrac{1}{2}\big[(x_2y_3-x_3y_2)+(x_3y_1-x_1y_3)+(x_1y_2-x_2y_1)\big]. \tag{24.4}$$

This rule is depicted in Figure 24.1(a). It has degree 1, meaning that it integrates exactly up to linear polynomials in triangular coordinates. For example, $F=4-\zeta_1+2\zeta_2-\zeta_3$ is exactly integrated by (24.3).

Three Point Rules. The next two rules in order of simplicity contain *three* sample points:

$$\frac{1}{A}\int_{\Omega^{(e)}} F(\zeta_1,\zeta_2,\zeta_3)\,d\Omega^{(e)} \approx \tfrac{1}{3}F(\tfrac{2}{3},\tfrac{1}{6},\tfrac{1}{6})+\tfrac{1}{3}F(\tfrac{1}{6},\tfrac{2}{3},\tfrac{1}{6})+\tfrac{1}{3}F(\tfrac{1}{6},\tfrac{1}{6},\tfrac{2}{3}). \tag{24.5}$$

$$\frac{1}{A}\int_{\Omega^{(e)}} F(\zeta_1,\zeta_2,\zeta_3)\,d\Omega^{(e)} \approx \tfrac{1}{3}F(\tfrac{1}{2},\tfrac{1}{2},0)+\tfrac{1}{3}F(0,\tfrac{1}{2},\tfrac{1}{2})+\tfrac{1}{3}F(\tfrac{1}{2},0,\tfrac{1}{2}). \tag{24.6}$$

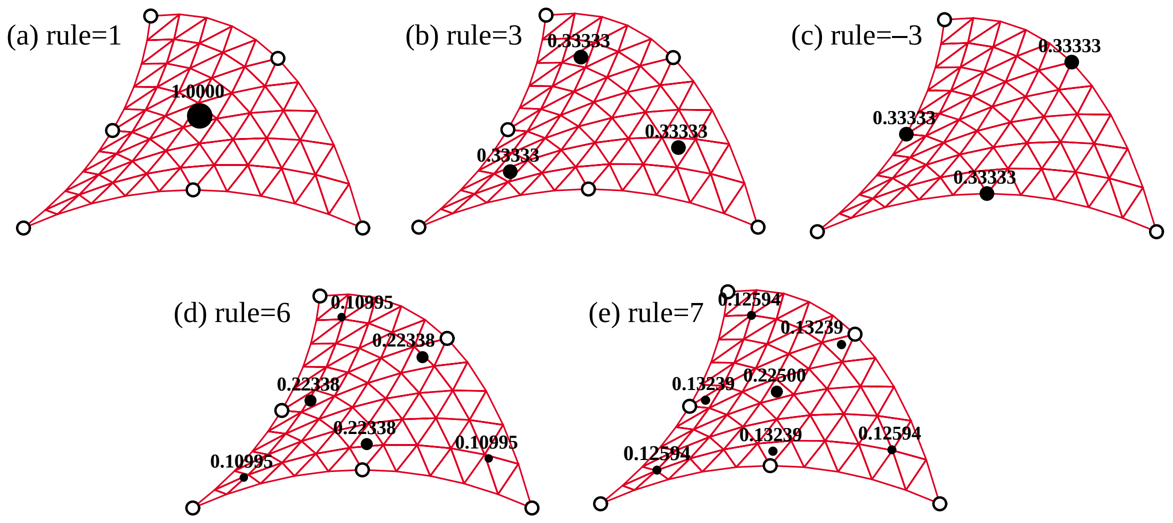


Figure 24.2. Location of sample points (dark circles) of five Gauss quadrature rules for curved sided 6-node triangles. Weight written to 5 places near each sample point; sample-point circle areas are proportional to weight.

These are depicted in Figures 24.1(b) and (c), respectively. Both rules are of degree 2; that is, exact 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.

REMARK 24.2

In certain applications, such as the axisymmetric solid structures considered in advanced FEM, the internal rule is preferred to avoid 0/0 singularities if an element edge falls on the axis of revolution.

Six and Seven Point Rules. There is a 4-point rule of degree 3 but it has a negative weight, which violates the stability condition (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, respectively. The 7-point rule includes the centroid as sample point. The abcissas and weights are expressable as rational combinations of square roots of integers and fractions. The expressions are listed in the *Mathematica* implementation shown in Figure 24.3. The rules is depicted in Figures 24.1(d) and (e).

§24.2.3. Arbitrary Iso-P Triangles

If the triangle has variable metric, as in the curved-side 6-node triangle geometries pictured in Figure 24.2, the foregoing formulas need adjustment because the element of area $d\Omega^{(e)}$ 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

$$d\Omega^{(e)} = J\, d\zeta_1\, d\zeta_2\, d\zeta_3, \qquad J = \tfrac{1}{2}\det \begin{bmatrix} 1 & 1 & 1 \\ \sum_{i=1}^{n} x_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum_{i=1}^{n} x_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum_{i=1}^{n} x_i \dfrac{\partial N_i}{\partial \zeta_3} \\ \sum_{i=1}^{n} y_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum_{i=1}^{n} y_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum_{i=1}^{n} y_i \dfrac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \tag{24.7}$$

```
TrigGaussRuleInfo[{rule_,numer_},point_]:= Module[
 {zeta,p=rule,i=point,g1,g2,info=Null},
  If [p== 1, info={{1/3,1/3,1/3},1}];
  If [p==-3, zeta={1/2,1/2,1/2}; zeta[[i]]=0; info={zeta,1/3}];
  If [p== 3, zeta={1/6,1/6,1/6}; zeta[[i]]=2/3; info={zeta,1/3}];
  If [p== 6,
     If [i<=3, g1=(8-Sqrt[10]+Sqrt[38-44*Sqrt[2/5]])/18;
         zeta={g1,g1,g1}; zeta[[i]]=1-2*g1;
         info={zeta,(620+Sqrt[213125-53320*Sqrt[10]])/3720}];
     If [i>3,  g2=(8-Sqrt[10]-Sqrt[38-44*Sqrt[2/5]])/18;
         zeta={g2,g2,g2}; zeta[[i-3]]=1-2*g2;
         info={zeta,(620-Sqrt[213125-53320*Sqrt[10]])/3720}]];
  If [p== 7,
     If [i==1,info={{1/3,1/3,1/3},9/40} ];
     If [i>1&&i<=4,zeta=Table[(6-Sqrt[15])/21,{3}];
         zeta[[i-1]]=(9+2*Sqrt[15])/21;
         info={zeta,(155-Sqrt[15])/1200}];
     If [i>4,  zeta=Table[(6+Sqrt[15])/21,{3}];
         zeta[[i-4]]=(9-2*Sqrt[15])/21;
         info={zeta,(155+Sqrt[15])/1200}]];
  If [numer, 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 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

$$\int_{\Omega^{(e)}} F(\zeta_1, \zeta_2, \zeta_3)\, d\Omega^{(e)} \approx J(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3})\, F(\tfrac{1}{3}, \tfrac{1}{3}, \tfrac{1}{3}). \tag{24.8}$$

whereas the midpoint rule becomes

$$\int_{\Omega^{(e)}} F(\zeta_1, \zeta_2, \zeta_3)\, d\Omega^{(e)} \approx \tfrac{1}{3} J(\tfrac{1}{2}, \tfrac{1}{2}, 0)\, F(\tfrac{1}{2}, \tfrac{1}{2}, 0) + \tfrac{1}{3} J(0, \tfrac{1}{2}, \tfrac{1}{2})\, F(0, \tfrac{1}{2}, \tfrac{1}{2}) + \tfrac{1}{3} J(\tfrac{1}{2}, 0, \tfrac{1}{2})\, F(\tfrac{1}{2}, 0, \tfrac{1}{2}). \tag{24.9}$$

These can be rewritten more compactly by saying that the Gauss integration rule is applied to JF.

§24.2.4. Implementation in Mathematica

The five rules of Figures 24.1–2 are implemented in a *Mathematica* module called `TrigGaussRuleInfo`, which is listed in Figure 24.3. It is called as

```
{{zeta1,zeta2,zeta3},w} = TrigGaussRuleInfo[{rule,numer},point]          (24.10)
```

The module has 3 arguments: `rule`, `numer` 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 `numer` 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]`.

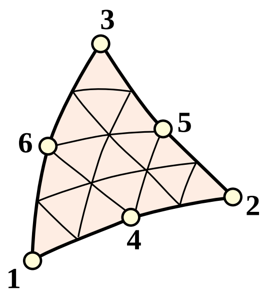


Figure 24.4. The 6-node quadratic iso-P triangle.

The module returns the two-level 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, the call `TrigGaussRuleInfo[{ 3,False },1]` returns `{ { 2/3,1/6,1/6 },1/3 }`. If `rule` is not 1, 3, -3, 6 or 7, the module returns `Null`.

§24.3. PARTIAL DERIVATIVE COMPUTATION

The calculation of Cartesian partial derivatives will be illustrated in this section for the 6-node quadratic isoparametric triangle shown in Figure 24.4. However the results are applicable to arbitrary iso-P triangles with any number of nodes.

The element geometry is defined by the corner coordinates $\{x_i, y_i\}$, with $i = 1, 2, \ldots 6$. 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 $\{u_{xi}, u_{yi}\}$, $i = 1, 2, \ldots, 6$, as per the iso-P quadratic interpolation formula (16.10–11). That formula is repeated here for convenience:

$$\begin{bmatrix} 1 \\ x \\ y \\ u_x \\ u_y \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 & x_6 \\ y_1 & y_2 & y_3 & y_4 & y_5 & y_6 \\ u_{x1} & u_{x2} & u_{x3} & u_{x4} & u_{x5} & u_{x6} \\ u_{y1} & u_{y2} & u_{y3} & u_{y4} & u_{y5} & u_{y6} \end{bmatrix} \mathbf{N}, \tag{24.11}$$

in which the shape functions and their natural derivatives are

$$\mathbf{N} = \begin{bmatrix} N_1^{(e)} \\ N_2^{(e)} \\ N_3^{(e)} \\ N_4^{(e)} \\ N_5^{(e)} \\ N_6^{(e)} \end{bmatrix} = \begin{bmatrix} \zeta_1(2\zeta_1 - 1) \\ \zeta_2(2\zeta_2 - 1) \\ \zeta_3(2\zeta_3 - 1) \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_1} = \begin{bmatrix} 4\zeta_1 - 1 \\ 0 \\ 0 \\ 4\zeta_2 \\ 0 \\ 4\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_2} = \begin{bmatrix} 0 \\ 4\zeta_2 - 1 \\ 0 \\ 4\zeta_1 \\ 4\zeta_3 \\ 0 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_3} = \begin{bmatrix} 0 \\ 0 \\ 4\zeta_3 - 1 \\ 0 \\ 4\zeta_2 \\ 4\zeta_1 \end{bmatrix}. \tag{24.12}$$

§24.3.1. Triangle Coordinate Partials

The bulk of the shape function subroutine 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: $w(\zeta_1, \zeta_2, \zeta_3)$, which is quadratically interpolated over the triangle by

$$w = [\, w_1 \quad w_2 \quad w_3 \quad w_4 \quad w_5 \quad w_6 \,] \begin{bmatrix} \zeta_1(2\zeta_1 - 1) \\ \zeta_2(2\zeta_2 - 1) \\ \zeta_3(2\zeta_3 - 1) \\ 4\zeta_1\zeta_2 \\ 4\zeta_2\zeta_3 \\ 4\zeta_3\zeta_1 \end{bmatrix}. \tag{24.13}$$

Symbol w may represent 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 with respect to x and y and applying the chain rule twice yields

$$\begin{aligned} \frac{\partial w}{\partial x} &= \sum w_i \frac{\partial N_i}{\partial x} = \sum w_i \left(\frac{\partial N_i}{\partial \zeta_1}\frac{\partial \zeta_1}{\partial x} + \frac{\partial N_i}{\partial \zeta_2}\frac{\partial \zeta_2}{\partial x} + \frac{\partial N_i}{\partial \zeta_3}\frac{\partial \zeta_3}{\partial x} \right), \\ \frac{\partial w}{\partial y} &= \sum w_i \frac{\partial N_i}{\partial y} = \sum w_i \left(\frac{\partial N_i}{\partial \zeta_1}\frac{\partial \zeta_1}{\partial y} + \frac{\partial N_i}{\partial \zeta_2}\frac{\partial \zeta_2}{\partial y} + \frac{\partial N_i}{\partial \zeta_3}\frac{\partial \zeta_3}{\partial y} \right), \end{aligned} \tag{24.14}$$

where all sums are understood to run from $i = 1$ through 6. In matrix form:

$$\begin{bmatrix} \dfrac{\partial w}{\partial x} \\ \dfrac{\partial w}{\partial y} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial \zeta_1}{\partial x} & \dfrac{\partial \zeta_2}{\partial x} & \dfrac{\partial \zeta_3}{\partial x} \\ \dfrac{\partial \zeta_1}{\partial y} & \dfrac{\partial \zeta_2}{\partial y} & \dfrac{\partial \zeta_3}{\partial y} \end{bmatrix} \begin{bmatrix} \sum w_i \dfrac{\partial N_i}{\partial \zeta_1} \\ \sum w_i \dfrac{\partial N_i}{\partial \zeta_2} \\ \sum w_i \dfrac{\partial N_i}{\partial \zeta_3} \end{bmatrix}. \tag{24.15}$$

Transposing both sides of (24.15) while exchanging left and right sides:

$$\begin{bmatrix} \sum w_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum w_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum w_i \dfrac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \zeta_1}{\partial x} & \dfrac{\partial \zeta_1}{\partial y} \\ \dfrac{\partial \zeta_2}{\partial x} & \dfrac{\partial \zeta_2}{\partial y} \\ \dfrac{\partial \zeta_3}{\partial x} & \dfrac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial w}{\partial x} & \dfrac{\partial w}{\partial y} \end{bmatrix}. \tag{24.16}$$

Now make $w \equiv 1, x, y$ and stack the results row-wise:

$$\begin{bmatrix} \sum \dfrac{\partial N_i}{\partial \zeta_1} & \sum \dfrac{\partial N_i}{\partial \zeta_2} & \sum \dfrac{\partial N_i}{\partial \zeta_3} \\ \sum x_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum x_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum x_i \dfrac{\partial N_i}{\partial \zeta_3} \\ \sum y_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum y_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum y_i \dfrac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \zeta_1}{\partial x} & \dfrac{\partial \zeta_1}{\partial y} \\ \dfrac{\partial \zeta_2}{\partial x} & \dfrac{\partial \zeta_2}{\partial y} \\ \dfrac{\partial \zeta_3}{\partial x} & \dfrac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial 1}{\partial x} & \dfrac{\partial 1}{\partial y} \\ \dfrac{\partial x}{\partial x} & \dfrac{\partial x}{\partial y} \\ \dfrac{\partial y}{\partial x} & \dfrac{\partial y}{\partial y} \end{bmatrix}. \tag{24.17}$$

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 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:

$$\begin{bmatrix} 1 & 1 & 1 \\ \sum x_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum x_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum x_i \dfrac{\partial N_i}{\partial \zeta_3} \\ \sum y_i \dfrac{\partial N_i}{\partial \zeta_1} & \sum y_i \dfrac{\partial N_i}{\partial \zeta_2} & \sum y_i \dfrac{\partial N_i}{\partial \zeta_3} \end{bmatrix} \begin{bmatrix} \dfrac{\partial \zeta_1}{\partial x} & \dfrac{\partial \zeta_1}{\partial y} \\ \dfrac{\partial \zeta_2}{\partial x} & \dfrac{\partial \zeta_2}{\partial y} \\ \dfrac{\partial \zeta_3}{\partial x} & \dfrac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{24.18}$$

§24.3.2. Solving the Jacobian System

By analogy with the quadrilateral isoparametric elements, the coefficient matrix of system (24.18) is called the *Jacobian matrix* and is 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

$$\mathbf{J}\,\mathbf{P} = \begin{bmatrix} 1 & 1 & 1 \\ J_{x1} & J_{x2} & J_{x3} \\ J_{y1} & J_{y2} & J_{y3} \end{bmatrix} \begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{24.19}$$

Solving this system gives

$$\begin{bmatrix} \frac{\partial \zeta_1}{\partial x} & \frac{\partial \zeta_1}{\partial y} \\ \frac{\partial \zeta_2}{\partial x} & \frac{\partial \zeta_2}{\partial y} \\ \frac{\partial \zeta_3}{\partial x} & \frac{\partial \zeta_3}{\partial y} \end{bmatrix} = \frac{1}{2J} \begin{bmatrix} J_{y23} & J_{x32} \\ J_{y31} & J_{x13} \\ J_{y12} & J_{x21} \end{bmatrix} = \mathbf{P}, \tag{24.20}$$

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

$$\begin{aligned} \frac{\partial w}{\partial x} &= \sum w_i \frac{\partial N_i}{\partial x} = \sum \frac{w_i}{2J}\left(\frac{\partial N_i}{\partial \zeta_1}J_{y23} + \frac{\partial N_i}{\partial \zeta_2}J_{y31} + \frac{\partial N_i}{\partial \zeta_3}J_{y12}\right), \\ \frac{\partial w}{\partial y} &= \sum w_i \frac{\partial N_i}{\partial y} = \sum \frac{w_i}{2J}\left(\frac{\partial N_i}{\partial \zeta_1}J_{x32} + \frac{\partial N_i}{\partial \zeta_2}J_{x13} + \frac{\partial N_i}{\partial \zeta_3}J_{x21}\right). \end{aligned} \tag{24.21}$$

In particular, the partials of the shape functions are

$$\begin{aligned} \frac{\partial N_i}{\partial x} &= \frac{1}{2J}\left(\frac{\partial N_i}{\partial \zeta_1}J_{y23} + \frac{\partial N_i}{\partial \zeta_2}J_{y31} + \frac{\partial N_i}{\partial \zeta_3}J_{y12}\right), \\ \frac{\partial N_i}{\partial y} &= \frac{1}{2J}\left(\frac{\partial N_i}{\partial \zeta_1}J_{x32} + \frac{\partial N_i}{\partial \zeta_2}J_{x13} + \frac{\partial N_i}{\partial \zeta_3}J_{x21}\right). \end{aligned} \tag{24.22}$$

In matrix form:

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = \mathbf{P}^T \begin{bmatrix} \frac{\partial N_i}{\partial \zeta_1} & \frac{\partial N_i}{\partial \zeta_2} & \frac{\partial N_i}{\partial \zeta_3} \end{bmatrix}^T, \tag{24.23}$$

where $\mathbf{P}$ is the 3×2 matrix of triangular coordinate partials defined in (24.20).

REMARK 24.3

Here is the proof that each of the entries of the first row of (24.17) is a 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

$$S = \sum N_i = 1 = c_1 Z + c_2 Z^2 + \ldots c_n Z^n, \qquad c_1 + c_2 + \ldots + c_n = 1. \tag{24.24}$$

where the c_i are element dependent scalar coefficients. Differentiating S with respect to the ζ_i's and setting $Z = 1$ yields

$$\begin{aligned} C = \sum \frac{\partial N_i}{\partial \zeta_1} = \sum \frac{\partial N_i}{\partial \zeta_2} = \sum \frac{\partial N_i}{\partial \zeta_3} &= c_1 + 2c_2 Z + c_3 Z^3 + \ldots (n-1)Z^{n-1} = c_1 + 2c_2 + 3c_3 + \ldots + c_{n-1} \\ &= 1 + c_2 + 2c_3 + \ldots + (n-2)c_n, \end{aligned} \tag{24.25}$$

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.

```
Trig6IsoPShapeFunDer[ncoor_,tcoor_]:= Module[
  {ξ1,ξ2,ξ3,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}, {ξ1,ξ2,ξ3}=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={ξ1*(2*ξ1-1),ξ2*(2*ξ2-1),ξ3*(2*ξ3-1),4*ξ1*ξ2,4*ξ2*ξ3,4*ξ3*ξ1};
  Jx21= x2-x1+4*(dx4*(ξ1-ξ2)+(dx5-dx6)*ξ3);
  Jx32= x3-x2+4*(dx5*(ξ2-ξ3)+(dx6-dx4)*ξ1);
  Jx13= x1-x3+4*(dx6*(ξ3-ξ1)+(dx4-dx5)*ξ2);
  Jy12= y1-y2+4*(dy4*(ξ2-ξ1)+(dy6-dy5)*ξ3);
  Jy23= y2-y3+4*(dy5*(ξ3-ξ2)+(dy4-dy6)*ξ1);
  Jy31= y3-y1+4*(dy6*(ξ1-ξ3)+(dy5-dy4)*ξ2);
  Jdet = Jx21*Jy31-Jy12*Jx13;
  dNx= {(4*ξ1-1)*Jy23,(4*ξ2-1)*Jy31,(4*ξ3-1)*Jy12,4*(ξ2*Jy23+ξ1*Jy31),
         4*(ξ3*Jy31+ξ2*Jy12),4*(ξ1*Jy12+ξ3*Jy23)}/Jdet;
  dNy= {(4*ξ1-1)*Jx32,(4*ξ2-1)*Jx13,(4*ξ3-1)*Jx21,4*(ξ2*Jx32+ξ1*Jx13),
         4*(ξ3*Jx13+ξ2*Jx21),4*(ξ1*Jx21+ξ3*Jx32)}/Jdet;
  Return[Simplify[{Nf,dNx,dNy,Jdet}]]
];
```

Figure 24.5. Shape function module for 6-node quadratic triangle.

§24.4. THE QUADRATIC TRIANGLE

§24.4.1. Shape Function Module

We specialize now the results of §24.3 to work out 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 (24.19):

$$\begin{aligned} &J_{x1} = x_1(4\zeta_1-1)+4(x_4\zeta_2+x_6\zeta_3),\;\; J_{x2} = x_2(4\zeta_2-1)+4(x_5\zeta_3+x_4\zeta_1),\;\; J_{x3} = x_3(4\zeta_3-1)+4(x_6\zeta_1+x_5\zeta_2),\\ &J_{y1} = y_1(4\zeta_1-1)+4(y_4\zeta_2+y_6\zeta_3),\;\; J_{y2} = y_2(4\zeta_2-1)+4(y_5\zeta_3+y_4\zeta_1),\;\; J_{y3} = y_3(4\zeta_3-1)+4(y_6\zeta_1+y_5\zeta_2), \end{aligned} \tag{24.26}$$

from which the matrix $\mathbf{J}$ can be computed, and $\partial N_i/\partial x$ and $\partial N_i/\partial y$ obtained from (24.22). Somewhat simpler expressions, however, can be obtained by using the following "hierarchical" side node coordinates:

$$\begin{aligned} \Delta x_4 &= x_4 - \tfrac{1}{2}(x_1+x_2), \quad \Delta x_5 = x_5 - \tfrac{1}{2}(x_2+x_3), \quad \Delta x_6 = x_6 - \tfrac{1}{2}(x_3+x_1),\\ \Delta y_4 &= y_4 - \tfrac{1}{2}(y_1+y_2), \quad \Delta y_5 = y_5 - \tfrac{1}{2}(y_2+y_3), \quad \Delta y_6 = y_6 - \tfrac{1}{2}(y_3+y_1). \end{aligned} \tag{24.27}$$

Geometrically these represent the deviations from the midpoint positions; thus for a superparametric element $\Delta x4 = \Delta x5 = \Delta x6 = \Delta y4 = \Delta y5 = \Delta y6 = 0$. The Jacobian coefficients become

$$\begin{aligned} J_{x21} &= x_{21} + 4(\Delta x_4(\zeta_1-\zeta_2)+(\Delta x_5-\Delta x_6)\zeta_3), \quad J_{x32} = x_{32} + 4(\Delta x_5(\zeta_2-\zeta_3)+(\Delta x_6-\Delta x_4)\zeta_1),\\ J_{x13} &= x_{13} + 4(\Delta x_6(\zeta_3-\zeta_1)+(\Delta x_4-\Delta x_5)\zeta_2), \quad J_{y12} = y_{12} + 4(\Delta y_4(\zeta_2-\zeta_1)+(\Delta y_6-\Delta y_5)\zeta_3),\\ J_{y23} &= y_{23} + 4(\Delta y_5(\zeta_3-\zeta_2)+(\Delta y_4-\Delta y_6)\zeta_1), \quad J_{y31} = y_{31} + 4(\Delta y_6(\zeta_1-\zeta_3)+(\Delta y_5-\Delta y_4)\zeta_2). \end{aligned} \tag{24.28}$$

From this one gets $J = \frac{1}{2}\det\mathbf{J} = \frac{1}{2}(J_{x21}J_{y31} - J_{y12}J_{x13})$ and

$$\mathbf{P}^T = \frac{1}{2J}\begin{bmatrix} y_{23}+4(\Delta y_5(\zeta_3-\zeta_2)+(\Delta y_4-\Delta y_6)\zeta_1) & x_{32}+4(\Delta x_5(\zeta_2-\zeta_3)+(\Delta x_6-\Delta x_4)\zeta_1)\\ y_{31}+4(\Delta y_6(\zeta_1-\zeta_3)+(\Delta y_5-\Delta y_4)\zeta_2) & x_{13}+4(\Delta x_6(\zeta_3-\zeta_1)+(\Delta x_4-\Delta x_5)\zeta_2)\\ y_{12}+4(\Delta y_4(\zeta_2-\zeta_1)+(\Delta y_6-\Delta y_5)\zeta_3) & x_{21}+4(\Delta x_4(\zeta_1-\zeta_2)+(\Delta x_5-\Delta x_6)\zeta_3) \end{bmatrix}. \tag{24.29}$$

```
Trig6IsoPMembraneStiffness[ncoor_,mprop_,fprop_,opt_]:=
 Module[{i,k,l,p=3,numer=False,Emat,th={fprop},h,tcoor,w,c,
  Nf,dNx,dNy,Jdet,B,Ke=Table[0,{12},{12}]},
  Emat=mprop[[1]]; If [Length[fprop]>0, th=fprop[[1]]];
  If [Length[opt]>0, numer=opt[[1]]];
  If [Length[opt]>1, p=  opt[[2]]];
  If [p!=-3&&p!=1&&p!=3&&p!=7, 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 [Length[th]==0, h=th, h=th.Nf]; c=w*Jdet*h/2;
      B= {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[B].(Emat.B);
       ]; If[!numer,Ke=Simplify[Ke]]; Return[Ke]
 ];
```

Figure 24.6. Stiffness matrix module for 6-node plane stress triangle.

Specifically, the Cartesian derivatives of the shape functions are given by

$$\frac{\partial \mathbf{N}}{\partial x} = \frac{1}{2J}\begin{bmatrix} (4\zeta_1-1)\,J_{y23} \\ (4\zeta_2-1)\,J_{y31} \\ (4\zeta_3-1)\,J_{y12} \\ 4(\zeta_2 J_{y23}+\zeta_1 J_{y31}) \\ 4(\zeta_3 J_{y31}+\zeta_2 J_{y12}) \\ 4(\zeta_1 J_{y12}+\zeta_3 J_{y23}) \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial y} = \frac{1}{2J}\begin{bmatrix} (4\zeta_1-1)\,J_{x32} \\ (4\zeta_2-1)\,J_{x13} \\ (4\zeta_3-1)\,J_{x21} \\ 4(\zeta_2 J_{x32}+\zeta_1 J_{x13}) \\ 4(\zeta_3 J_{x13}+\zeta_2 J_{x21}) \\ 4(\zeta_1 J_{x21}+\zeta_3 J_{x32}) \end{bmatrix}. \tag{24.30}$$

Using these expressions, a *Mathematica* implementation of the shape functions is programmed as module `Trig6IsoPShapeFunDer`, which is shown in Figure 24.5. This module receives two arguments: `ncoor` and `tcoor`. The first one is the list of $\{x_i, y_i\}$ coordinates of the six nodes: `{ { x1,y1 },{ x2,y2 }, ... { x6,y6 } }`. 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` the shape function x derivative values,`dNy` the shape function y derivative values, and `Jdet` is the determinant of matrix $\mathbf{J}$, equal to $2J$ in the notation used here.

§24.4.2. Stiffness Module

The integration formula for the triangle stiffness may be stated as

$$\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\mathbf{B}^T\mathbf{E}\mathbf{B}\,d\Omega^{(e)} \approx \sum_{i=1}^{p} w_i \mathbf{F}(\zeta_{1i}, \zeta_{2i}, \zeta_{3i}), \qquad \text{where} \quad \mathbf{F}(\zeta_1, \zeta_2, \zeta_3) = h\mathbf{B}^T\mathbf{E}\mathbf{B}\,J. \tag{24.31}$$

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, ζ_{1i}, ζ_{2i}, ζ_{3i} are the sample point triangular coordinates and $J = \frac{1}{2}\det\mathbf{J}$. The last four numbers are returned by `TrigGaussRuleInfo` as explained in the previous section.

Module `Trig6IsoPMembraneStiffness`, listed in Figure 24.6, implements the computation of the element stiffness matrix of a quadratic plane stress triangle. The arguments of the module are

`ncoor`	Node coordinates arranged in two-dimensional list form: `{ { x1,y1 },{ x2,y2 },{ x3,y3 },{ x4,y4 },{ x5,y5 },{ x6,y6 } }`.

`mprop`	Material properties supplied as the list { `Emat`,`rho`,`alpha` }. `Emat` is a two-dimensional list storing the 3×3 plane stress matrix of elastic moduli: $$\mathbf{E} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \qquad (24.32)$$ The other two items in `mprop` are not used in this module so zeros may be inserted as placeholders.
`fprop`	Fabrication properties. The plate thickness specified as a one-entry list: { `h` }, as a six-entry list: { `h1`,`h2`,`h3`,`h4`,`h5`,`h6` }, or as an empty list: { }. 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. If an empty list appears the module assumes a uniform unit thickness.
`options`	Processing options. This list may contain two items: { `numer`,`rule` } or one: { `numer` }. `numer` is a flag with value `True` or `False`. If `True`, the computations are forced to go 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 `Ke` as an 12×12 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4} \quad u_{x5} \quad u_{y5} \quad u_{x6} \quad u_{y6} \,]^T . \qquad (24.33)$$

§24.4.3. Test Elements

The stiffness module is tested on the two triangle geometries shown in Figure 24.7. Both elements have unit thickness and isotropic material. The left one has the corner nodes placed at (0, 0), (4, 2) and 6, 4) with side nodes 4,5,6 at the midpoints of the sides. The right one has the 3 corners forming an equilateral triangle: $-1/2, 0,\ 1/2, 0,\ 0, \sqrt{3}/2$ whereas the side nodes are placed at $0, -1/(2\sqrt{3}),\ 1/2, 1/\sqrt{3},\ -1/2, 1/\sqrt{3}$ so that the six nodes lie on a circle of radius $1/\sqrt{3}$. These geometries will be used to illustrate the effect of the numerical integration rule.

The following test statements form $\mathbf{K}^{(e)}$ for the left triangle of Figure 24.7 with $E = 288$, $\nu = 1/3$ and $h = 1$:

```
  ClearAll[Em,nu,h];   h=1; Em=288; nu=1/3;
  ncoor={{0,0},{6,2},{4,4},{3,1},{5,3},{2,2}};
  Emat=Em/(1-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
  Ke=Trig6IsoPMembraneStiffness[ncoor,{Emat,0,0},{h},{False,-3}];
  Ke=Simplify[Ke];  Print[Chop[Ke]//MatrixForm];
  Print["eigs of Ke=",Chop[Eigenvalues[N[Ke]]]];
```

The returned stiffness matrix for integration `rule=3`, `rule=-3`, and `rule=7` is the same since those are exact for this element if the side nodes are at the midpoints, which is the case here. That stiffness is

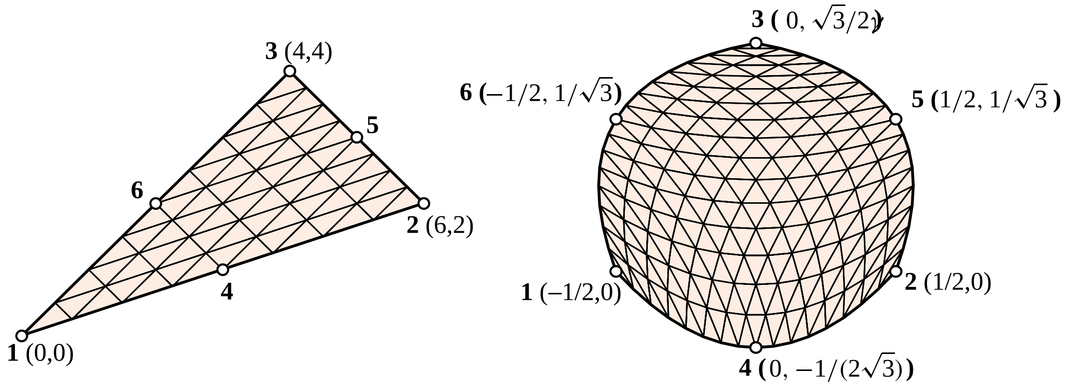


Figure 24.7. Test 6-node triangle element geometries.

$$\begin{bmatrix} 54 & 27 & 18 & 0 & 0 & 9 & -72 & 0 & 0 & 0 & 0 & -36 \\ 27 & 54 & 0 & -18 & 9 & 36 & 0 & 72 & 0 & 0 & -36 & -144 \\ 18 & 0 & 216 & -108 & 54 & -36 & -72 & 0 & -216 & 144 & 0 & 0 \\ 0 & -18 & -108 & 216 & -36 & 90 & 0 & 72 & 144 & -360 & 0 & 0 \\ 0 & 9 & 54 & -36 & 162 & -81 & 0 & 0 & -216 & 144 & 0 & -36 \\ 9 & 36 & -36 & 90 & -81 & 378 & 0 & 0 & 144 & -360 & -36 & -144 \\ -72 & 0 & -72 & 0 & 0 & 0 & 576 & -216 & 0 & -72 & -432 & 288 \\ 0 & 72 & 0 & 72 & 0 & 0 & -216 & 864 & -72 & -288 & 288 & -720 \\ 0 & 0 & -216 & 144 & -216 & 144 & 0 & -72 & 576 & -216 & -144 & 0 \\ 0 & 0 & 144 & -360 & 144 & -360 & -72 & -288 & -216 & 864 & 0 & 144 \\ 0 & -36 & 0 & 0 & 0 & -36 & -432 & 288 & -144 & 0 & 576 & -216 \\ -36 & -144 & 0 & 0 & -36 & -144 & 288 & -720 & 0 & 144 & -216 & 864 \end{bmatrix} \tag{24.34}$$

The eigenvalues are:

$$[\,1971.66 \quad 1416.75 \quad 694.82 \quad 545.72 \quad 367.7 \quad 175.23 \quad 157.68 \quad 57.54 \quad 12.899 \quad 0 \quad 0 \quad 0\,] \tag{24.35}$$

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.

For the "circle" geometry the results for $E = 504$, $\nu = 0$, $h = 1$ and the rank-sufficient rules `3,-3,6,7` are obtained through the following script:

```
ClearAll[Em,nu,h];  Em=7*72; nu=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-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
For [i=2,i<=5,i++, p={1,-3,3,6,7}[[i]];
    Ke=Trig6IsoPMembraneStiffness[ncoor,{Emat,0,0},{h},{True,p}];
    Ke=Chop[Simplify[Ke]];
    Print["Ke=",SetPrecision[Ke,4]//MatrixForm];
    Print["Eigenvalues of Ke=",Chop[Eigenvalues[N[Ke]],.0000001]]
];
```

This is straightforward except perhaps for the `SetPrecision[Ke,4]` statement in the `Print`. This specifies that the stiffness matrix entries be printed to only 4 places to save space (the default is otherwise 6 places).

For `rule=3`:

$$\begin{bmatrix}
344.7 & 75.00 & -91.80 & 21.00 & -86.60 & -24.00 & -124.7 & -72.00 & -20.78 & -36.00 & -20.78 & 36.00 \\
75.00 & 258.1 & -21.00 & -84.87 & 18.00 & -90.07 & 96.00 & 0 & -36.00 & 20.78 & -132.0 & -103.9 \\
-91.80 & -21.00 & 344.7 & -75.00 & -86.60 & 24.00 & -124.7 & 72.00 & -20.78 & -36.00 & -20.78 & 36.00 \\
21.00 & -84.87 & -75.00 & 258.1 & -18.00 & -90.07 & -96.00 & 0 & 132.0 & -103.9 & 36.00 & 20.78 \\
-86.60 & 18.00 & -86.60 & -18.00 & 214.8 & 0 & 41.57 & 0 & -41.57 & 144.0 & -41.57 & -144.0 \\
-24.00 & -90.07 & 24.00 & -90.07 & 0 & 388.0 & 0 & -41.57 & -24.00 & -83.14 & 24.00 & -83.14 \\
-124.7 & 96.00 & -124.7 & -96.00 & 41.57 & 0 & 374.1 & 0 & -83.14 & -72.00 & -83.14 & 72.00 \\
-72.00 & 0 & 72.00 & 0 & 0 & -41.57 & 0 & 374.1 & -72.00 & -166.3 & 72.00 & -166.3 \\
-20.78 & -36.00 & -20.78 & 132.0 & -41.57 & -24.00 & -83.14 & -72.00 & 374.1 & 0 & -207.8 & 0 \\
-36.00 & 20.78 & -36.00 & -103.9 & 144.0 & -83.14 & -72.00 & -166.3 & 0 & 374.1 & 0 & -41.57 \\
-20.78 & -132.0 & -20.78 & 36.00 & -41.57 & 24.00 & -83.14 & 72.00 & -207.8 & 0 & 374.1 & 0 \\
36.00 & -103.9 & 36.00 & 20.78 & -144.0 & -83.14 & 72.00 & -166.3 & 0 & -41.57 & 0 & 374.1
\end{bmatrix} \tag{24.36}$$

For `rule=-3`:

$$\begin{bmatrix}
566.4 & 139.0 & 129.9 & 21.00 & 79.67 & 8.000 & -364.9 & -104.0 & -205.5 & -36.00 & -205.5 & -28.00 \\
139.0 & 405.9 & -21.00 & 62.93 & 50.00 & 113.2 & 64.00 & -129.3 & -36.00 & -164.0 & -196.0 & -288.7 \\
129.9 & -21.00 & 566.4 & -139.0 & 79.67 & -8.000 & -364.9 & 104.0 & -205.5 & 28.00 & -205.5 & 36.00 \\
21.00 & 62.93 & -139.0 & 405.9 & -50.00 & 113.2 & -64.00 & -129.3 & 196.0 & -288.7 & 36.00 & -164.0 \\
79.67 & 50.00 & 79.67 & -50.00 & 325.6 & 0 & -143.2 & 0 & -170.9 & 176.0 & -170.9 & -176.0 \\
8.000 & 113.2 & -8.000 & 113.2 & 0 & 646.6 & 0 & -226.3 & 8.000 & -323.3 & -8.000 & -323.3 \\
-364.9 & 64.00 & -364.9 & -64.00 & -143.2 & 0 & 632.8 & 0 & 120.1 & -104.0 & 120.1 & 104.0 \\
-104.0 & -129.3 & 104.0 & -129.3 & 0 & -226.3 & 0 & 485.0 & -104.0 & 0 & 104.0 & 0 \\
-205.5 & -36.00 & -205.5 & 196.0 & -170.9 & 8.000 & 120.1 & -104.0 & 521.9 & -64.00 & -60.04 & 0 \\
-36.00 & -164.0 & 28.00 & -288.7 & 176.0 & -323.3 & -104.0 & 0 & -64.00 & 595.8 & 0 & 180.1 \\
-205.5 & -196.0 & -205.5 & 36.00 & -170.9 & -8.000 & 120.1 & 104.0 & -60.04 & 0 & 521.9 & 64.00 \\
-28.00 & -288.7 & 36.00 & -164.0 & -176.0 & -323.3 & 104.0 & 0 & 0 & 180.1 & 64.00 & 595.8
\end{bmatrix} \tag{24.37}$$

The stiffness for `rule=6` is omitted to save space. For `rule=7`:

$$\begin{bmatrix}
661.9 & 158.5 & 141.7 & 21.00 & 92.53 & 7.407 & -432.1 & -117.2 & -190.2 & -29.10 & -273.8 & -40.61 \\
158.5 & 478.8 & -21.00 & 76.13 & 49.41 & 125.3 & 50.79 & -182.7 & -29.10 & -156.6 & -208.6 & -341.0 \\
141.7 & -21.00 & 661.9 & -158.5 & 92.53 & -7.407 & -432.1 & 117.2 & -273.8 & 40.61 & -190.2 & 29.10 \\
21.00 & 76.13 & -158.5 & 478.8 & -49.41 & 125.3 & -50.79 & -182.7 & 208.6 & -341.0 & 29.10 & -156.6 \\
92.53 & 49.41 & 92.53 & -49.41 & 387.3 & 0 & -139.8 & 0 & -216.3 & 175.4 & -216.3 & -175.4 \\
7.407 & 125.3 & -7.407 & 125.3 & 0 & 753.4 & 0 & -207.0 & 7.407 & -398.5 & -7.407 & -398.5 \\
-432.1 & 50.79 & -432.1 & -50.79 & -139.8 & 0 & 723.6 & 0 & 140.2 & -117.2 & 140.2 & 117.2 \\
-117.2 & -182.7 & 117.2 & -182.7 & 0 & -207.0 & 0 & 562.6 & -117.2 & 4.884 & 117.2 & 4.884 \\
-190.2 & -29.10 & -273.8 & 208.6 & -216.3 & 7.407 & 140.2 & -117.2 & 602.8 & -69.71 & -62.78 & 0 \\
-29.10 & -156.6 & 40.61 & -341.0 & 175.4 & -398.5 & -117.2 & 4.884 & -69.71 & 683.3 & 0 & 207.9 \\
-273.8 & -208.6 & -190.2 & 29.10 & -216.3 & -7.407 & 140.2 & 117.2 & -62.78 & 0 & 602.8 & 69.71 \\
-40.61 & -341.0 & 29.10 & -156.6 & -175.4 & -398.5 & 117.2 & 4.884 & 0 & 207.9 & 69.71 & 683.3
\end{bmatrix} \tag{24.38}$$

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

(24.39)

Since the metric of this element is very distorted near its boundary, the stiffness matrix entries and eigenvalues change substantially as the integration formulas are advanced from 3 to 6 and 7 points. However as can be seen the matrix remains rank-sufficient.

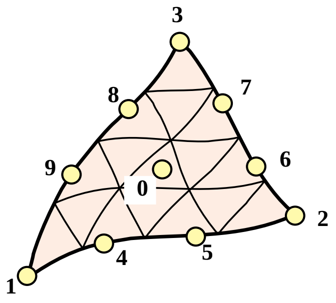


Figure 24.8. The 10-node cubic triangle.

§24.5. *THE CUBIC TRIANGLE

The 10-node cubic triangle, depicted in Figure 24.8, is rarely used in practice as such 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. One change in node labeling should be noted: the interior node is relabeled 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 were obtained in Chapter 18. They are reproduced here along with their natural derivatives:

$$\mathbf{N} = \begin{bmatrix} \frac{1}{2}\zeta_1(3\zeta_1-1)(3\zeta_1-2) \\ \frac{1}{2}\zeta_2(3\zeta_2-1)(3\zeta_2-2) \\ \frac{1}{2}\zeta_3(3\zeta_3-1)(3\zeta_3-2) \\ \frac{9}{2}\zeta_1\zeta_2(3\zeta_1-1) \\ \frac{9}{2}\zeta_1\zeta_2(3\zeta_2-1) \\ \frac{9}{2}\zeta_2\zeta_3(3\zeta_2-1) \\ \frac{9}{2}\zeta_2\zeta_3(3\zeta_3-1) \\ \frac{9}{2}\zeta_3\zeta_1(3\zeta_3-1) \\ \frac{9}{2}\zeta_3\zeta_1(3\zeta_1-1) \\ 27\zeta_1\zeta_2\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_1} = \frac{9}{2} \begin{bmatrix} \frac{2}{9}-2\zeta_1+3\zeta_1^2 \\ 0 \\ 0 \\ \zeta_2(6\zeta_1-1) \\ \zeta_2(3\zeta_2-1) \\ 0 \\ 0 \\ \zeta_3(3\zeta_3-1) \\ \zeta_3(6\zeta_1-1) \\ 6\zeta_2\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_2} = \frac{9}{2} \begin{bmatrix} 0 \\ \frac{2}{9}-2\zeta_2+3\zeta_2^2 \\ 0 \\ \zeta_1(3\zeta_1-1) \\ \zeta_1(6\zeta_2-1) \\ \zeta_3(6\zeta_2-1) \\ \zeta_3(3\zeta_3-1) \\ 0 \\ 0 \\ 6\zeta_1\zeta_3 \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial \zeta_3} = \frac{9}{2} \begin{bmatrix} 0 \\ 0 \\ \frac{2}{9}-2\zeta_3+3\zeta_3^2 \\ 0 \\ 0 \\ \zeta_2(3\zeta_2-1) \\ \zeta_2(6\zeta_3-1) \\ \zeta_1(6\zeta_3-1) \\ \zeta_1(3\zeta_1-1) \\ 6\zeta_1\zeta_2 \end{bmatrix}. \tag{24.40}$$

As in the case of the 6-node triangle it is convenient to introduce the deviations from thirdpoints and centroid: $\Delta x_4 = x_4-\frac{1}{3}(2x_1+x_2)$, $\Delta x_5 = x_5-\frac{1}{3}(x_1+2x_2)$, $\Delta x_6 = x_6-\frac{1}{3}(2x_2+x_3)$, $\Delta x_7 = x_7-\frac{1}{3}(x_2+2x_3)$, $\Delta x_8 = x_8-\frac{1}{3}(2x_3+x_1)$, $\Delta x_9 = x_9-\frac{1}{3}(x_3+2x_1)$, $\Delta x_0 = x_{10}-\frac{1}{3}(x_1+x_2+x_3)$, $\Delta y_4 = y_4-\frac{1}{3}(2y_1+y_2)$, $\Delta y_5 = y_5-\frac{1}{3}(y_1+2y_2)$, $\Delta y_6 = y_6-\frac{1}{3}(2y_2+y_3)$, $\Delta y_7 = y_7-\frac{1}{3}(y_2+2y_3)$, $\Delta y_8 = y_8-\frac{1}{3}(2y_3+y_1)$, $\Delta y_9 = y_9-\frac{1}{3}(y_3+2y_1)$, and $\Delta y_0 = y_{10}-\frac{1}{3}(y_1+y_2+y_3)$. Using *Mathematica* the expressions of the Jacobian coefficients are found to be

$$\begin{aligned} J_{x21} &= x_{21} + \tfrac{9}{2}\Big[\Delta x_4\big(\zeta_1(3\zeta_1-6\zeta_2-1)+\zeta_2\big) + \Delta x_5\big(\zeta_2(1-3\zeta_2+6\zeta_1)-\zeta_1\big) + \Delta x_6\zeta_3(6\zeta_2-1) \\ &\qquad + \Delta x_7\zeta_3(3\zeta_3-1) + \Delta x_8\zeta_3(1-3\zeta_3) + \Delta x_9\zeta_3(1-6\zeta_1) + 6\Delta x_0\zeta_3(\zeta_1-\zeta_2)\Big], \\ J_{x32} &= x_{32} + \tfrac{9}{2}\Big[\Delta x_4\zeta_1(1-3\zeta_1) + \Delta x_5\zeta_1(1-6\zeta_2) + \Delta x_6\big(\zeta_2(3\zeta_2-6\zeta_3-1)+\zeta_3\big) \\ &\qquad + \Delta x_7\big(\zeta_3(1-3\zeta_3+6\zeta_2)-\zeta_2\big) + \Delta x_8\zeta_1(6\zeta_3-1) + \Delta x_9\zeta_1(3\zeta_1-1) + 6\Delta x_0\zeta_1(\zeta_2-\zeta_3)\Big], \\ J_{x13} &= x_{13} + \tfrac{9}{2}\Big[\Delta x_4(6\zeta_1-1)\zeta_2 + \Delta x_5\zeta_2(3\zeta_2-1) + \Delta x_6\zeta_2(1-3\zeta_2) + \Delta x_7\zeta_2(1-6\zeta_3) \\ &\qquad + \Delta x_8\big(\zeta_3(3\zeta_3-6\zeta_1-1)+\zeta_1\big) + \Delta x_9\big(\zeta_1(1-3\zeta_1+6\zeta_3)-\zeta_3\big) + 6\Delta x_0\zeta_2(\zeta_3-\zeta_1)\Big], \end{aligned} \tag{24.41}$$

```
Trig10IsoPShapeFunDer[ncoor_,tcoor_]:= Module[
  {ξ1,ξ2,ξ3,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;  {ξ1,ξ2,ξ3}=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={ξ1*(3*ξ1-1)*(3*ξ1-2),ξ2*(3*ξ2-1)*(3*ξ2-2),ξ3*(3*ξ3-1)*(3*ξ3-2),
      9*ξ1*ξ2*(3*ξ1-1),9*ξ1*ξ2*(3*ξ2-1),9*ξ2*ξ3*(3*ξ2-1),
      9*ξ2*ξ3*(3*ξ3-1),9*ξ3*ξ1*(3*ξ3-1),9*ξ3*ξ1*(3*ξ1-1),54*ξ1*ξ2*ξ3}/2;
  Jx21=x2-x1+(9/2)*(dx4*(ξ1*(3*ξ1-6*ξ2-1)+ξ2)+
    dx5*(ξ2*(1-3*ξ2+6*ξ1)-ξ1)+dx6*ξ3*(6*ξ2-1)+dx7*ξ3*(3*ξ3-1)+
    dx8*ξ3*(1-3*ξ3)+dx9*ξ3*(1-6*ξ1)+6*dx0*ξ3*(ξ1-ξ2));
  Jx32=x3-x2+(9/2)*(dx4*ξ1*(1-3*ξ1)+dx5*ξ1*(1-6*ξ2)+
    dx6*(ξ2*(3*ξ2-6*ξ3-1)+ξ3)+dx7*(ξ3*(1-3*ξ3+6*ξ2)-ξ2)+
    dx8*ξ1*(6*ξ3-1)+dx9*ξ1*(3*ξ1-1)+6*dx0*ξ1*(ξ2-ξ3)) ;
  Jx13=x1-x3+(9/2)*(dx4*(6*ξ1-1)*ξ2+dx5*ξ2*(3*ξ2-1)+
    dx6*ξ2*(1-3*ξ2)+dx7*ξ2*(1-6*ξ3)+dx8*(ξ3*(3*ξ3-6*ξ1-1)+ξ1)+
    dx9*(ξ1*(1-3*ξ1+6*ξ3)-ξ3)+6*dx0*ξ2*(ξ3-ξ1));
  Jy12=y1-y2-(9/2)*(dy4*(ξ1*(3*ξ1-6*ξ2-1)+ξ2)+
    dy5*(ξ2*(1-3*ξ2+6*ξ1)-ξ1)+dy6*ξ3*(6*ξ2-1)+dy7*ξ3*(3*ξ3-1)+
    dy8*ξ3*(1-3*ξ3)+dy9*ξ3*(1-6*ξ1)+6*dy0*ξ3*(ξ1-ξ2));
  Jy23=y2-y3-(9/2)*(dy4*ξ1*(1-3*ξ1)+dy5*ξ1*(1-6*ξ2)+
    dy6*(ξ2*(3*ξ2-6*ξ3-1)+ξ3)+dy7*(ξ3*(1-3*ξ3+6*ξ2)-ξ2)+
    dy8*ξ1*(6*ξ3-1)+dy9*ξ1*(3*ξ1-1)+6*dy0*ξ1*(ξ2-ξ3)) ;
  Jy31=y3-y1-(9/2)*(dy4*(6*ξ1-1)*ξ2+dy5*ξ2*(3*ξ2-1)+
    dy6*ξ2*(1-3*ξ2)+dy7*ξ2*(1-6*ξ3)+dy8*(ξ3*(3*ξ3-6*ξ1-1)+ξ1)+
    dy9*(ξ1*(1-3*ξ1+6*ξ3)-ξ3)+6*dy0*ξ2*(ξ3-ξ1));
  Jdet = Jx21*Jy31-Jy12*Jx13;
  dNx={Jy23*(2/9-2*ξ1+3*ξ1^2),Jy31*(2/9-2*ξ2+3*ξ2^2),Jy12*(2/9-2*ξ3+3*ξ3^2),
       Jy31*ξ1*(3*ξ1-1)+Jy23*ξ2*(6*ξ1-1),Jy23*ξ2*(3*ξ2-1)+Jy31*ξ1*(6*ξ2-1),
       Jy12*ξ2*(3*ξ2-1)+Jy31*ξ3*(6*ξ2-1),Jy31*ξ3*(3*ξ3-1)+Jy12*ξ2*(6*ξ3-1),
       Jy23*ξ3*(3*ξ3-1)+Jy12*ξ1*(6*ξ3-1),Jy12*ξ1*(3*ξ1-1)+Jy23*ξ3*(6*ξ1-1),
       6*(Jy12*ξ1*ξ2+Jy31*ξ1*ξ3+Jy23*ξ2*ξ3)}/(2*Jdet/9);
  dNy={Jx32*(2/9-2*ξ1+3*ξ1^2),Jx13*(2/9-2*ξ2+3*ξ2^2),Jx21*(2/9-2*ξ3+3*ξ3^2),
       Jx13*ξ1*(3*ξ1-1)+Jx32*ξ2*(6*ξ1-1),Jx32*ξ2*(3*ξ2-1)+Jx13*ξ1*(6*ξ2-1),
       Jx21*ξ2*(3*ξ2-1)+Jx13*ξ3*(6*ξ2-1),Jx13*ξ3*(3*ξ3-1)+Jx21*ξ2*(6*ξ3-1),
       Jx32*ξ3*(3*ξ3-1)+Jx21*ξ1*(6*ξ3-1),Jx21*ξ1*(3*ξ1-1)+Jx32*ξ3*(6*ξ1-1),
       6*(Jx21*ξ1*ξ2+Jx13*ξ1*ξ3+Jx32*ξ2*ξ3)}/(2*Jdet/9);
  Return[Simplify[{Nf,dNx,dNy,Jdet}]]
];
```

Figure 24.9. Shape function module for 10-node cubic triangle.

$$\begin{aligned}
J_{y12} &= y_{12} - \tfrac{9}{2}\Big[\Delta y_4\big(\zeta_1(3\zeta_1-6\zeta_2-1)+\zeta_2\big) + \Delta y_5(\zeta_2(1-3\zeta_2+6\zeta_1)-\zeta_1) + \Delta y_6\zeta_3(6\zeta_2-1)\\
&\qquad + \Delta y_7\zeta_3(3\zeta_3-1) + \Delta y_8\zeta_3(1-3\zeta_3) + \Delta y_9\zeta_3(1-6\zeta_1) + 6\Delta y_0\zeta_3(\zeta_1-\zeta_2)\Big],\\
J_{y23} &= y_{23} - \tfrac{9}{2}\Big[\Delta y_4\zeta_1(1-3\zeta_1) + \Delta y_5\zeta_1(1-6\zeta_2) + \Delta y_6\big(\zeta_2(3\zeta_2-6\zeta_3-1)+\zeta_3\big)\\
&\qquad + \Delta y_7\big(\zeta_3(1-3\zeta_3+6\zeta_2)-\zeta_2\big) + \Delta y_8\zeta_1(6\zeta_3-1) + \Delta y_9\zeta_1(3\zeta_1-1) + 6\Delta y_0\zeta_1(\zeta_2-\zeta_3)\Big],\\
J_{y31} &= y_{31} - \tfrac{9}{2}\Big[\Delta y_4(6\zeta_1-1)\zeta_2 + \Delta y_5\zeta_2(3\zeta_2-1) + \Delta y_6\zeta_2(1-3\zeta_2) + \Delta y_7\zeta_2(1-6\zeta_3)\\
&\qquad + \Delta y_8\big(\zeta_3(3\zeta_3-6\zeta_1-1)+\zeta_1\big) + \Delta y_9\big(\zeta_1(1-3\zeta_1+6\zeta_3)-\zeta_3\big) + 6\Delta y_0\zeta_2(\zeta_3-\zeta_1)\Big].
\end{aligned} \tag{24.42}$$

```
Trig10IsoPMembraneStiffness[ncoor_,mprop_,fprop_,opt_]:=
 Module[{i,k,l,p=6,numer=False,Emat,th={fprop},h,tcoor,w,c,
  Nf,dNx,dNy,Jdet,B,Ke=Table[0,{20},{20}]},
  Emat=mprop[[1]]; If [Length[fprop]>0, th=fprop[[1]]];
  If [Length[opt]>0, numer=opt[[1]]];
  If [Length[opt]>1, p=  opt[[2]]];
  If [p!=1&&p!=-3&&p!=3&&p!=6&&p!=7, 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 [Length[th]==0, h=th, h=th.Nf]; c=w*Jdet*h/2;
       B= {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[B].(Emat.B);
       ]; If[!numer,Ke=Simplify[Ke]]; Return[Ke]
 ];
```

Figure 24.10. Stiffness module for 10-node cubic triangle.

The Jacobian determinant is $J = \frac{1}{2}(J_{x21}J_{y31} - J_{y12}J_{x13})$. Using (24.23) the shape function partial derivatives can be expressed as

$$\frac{\partial \mathbf{N}}{\partial x} = \frac{9}{4J}\begin{bmatrix} J_{y23}(\frac{2}{9} - 2\zeta_1 + 3\zeta_1^2/2) \\ J_{y31}(\frac{2}{9} - 2\zeta_2 + 3\zeta_2^2/2) \\ J_{y12}(\frac{2}{9} - 2\zeta_3 + 3\zeta_3^2/2) \\ J_{y31}\zeta_1(3\zeta_1 - 1) + J_{y23}\zeta_2(6\zeta_1 - 1) \\ J_{y23}\zeta_2(3\zeta_2 - 1) + J_{y31}\zeta_1(6\zeta_2 - 1) \\ J_{y12}\zeta_2(3\zeta_2 - 1) + J_{y31}\zeta_3(6\zeta_2 - 1) \\ J_{y31}\zeta_3(3\zeta_3 - 1) + J_{y12}\zeta_2(6\zeta_3 - 1) \\ J_{y23}\zeta_3(3\zeta_3 - 1) + J_{y12}\zeta_1(6\zeta_3 - 1) \\ J_{y12}\zeta_1(3\zeta_1 - 1) + J_{y23}\zeta_3(6\zeta_1 - 1) \\ 6(J_{y12}\zeta_1\zeta_2 + J_{y31}\zeta_1\zeta_3 + J_{y23}\zeta_2\zeta_3) \end{bmatrix}, \quad \frac{\partial \mathbf{N}}{\partial y} = \frac{9}{4J}\begin{bmatrix} J_{x32}(\frac{2}{9} - 2\zeta_1 + 3\zeta_1^2/2) \\ J_{x13}(\frac{2}{9} - 2\zeta_2 + 3\zeta_2^2/2) \\ J_{x21}(\frac{2}{9} - 2\zeta_3 + 3\zeta_3^2/2) \\ J_{x13}\zeta_1(3\zeta_1 - 1) + J_{x32}\zeta_2(6\zeta_1 - 1) \\ J_{x32}\zeta_2(3\zeta_2 - 1) + J_{x13}\zeta_1(6\zeta_2 - 1) \\ J_{x21}\zeta_2(3\zeta_2 - 1) + J_{x13}\zeta_3(6\zeta_2 - 1) \\ J_{x13}\zeta_3(3\zeta_3 - 1) + J_{x21}\zeta_2(6\zeta_3 - 1) \\ J_{x32}\zeta_3(3\zeta_3 - 1) + J_{x21}\zeta_1(6\zeta_3 - 1) \\ J_{x21}\zeta_1(3\zeta_1 - 1) + J_{x32}\zeta_3(6\zeta_1 - 1) \\ 6(J_{x21}\zeta_1\zeta_2 + J_{x13}\zeta_1\zeta_3 + J_{x32}\zeta_2\zeta_3) \end{bmatrix}. \qquad (24.43)$$

The shape function module `Trig10IsoPShapeFunDer` that implements these expressions is listed in Figure 24.9. This has the same arguments as `Trig6IsoPShapeFunDer`. As there `Jdet` denotes $2J$.

§24.5.2. *Stiffness Module

Module `Trig10IsoPMembraneStiffness`, listed in Figure 24.10, implements the computation of the element stiffness matrix of the 10-node cubic plane stress triangle. The arguments of the module are

`ncoor`	Node coordinates arranged in two-dimensional list form: `{ { x1,y1 },{ x2,y2 },{ x3,y3 }, ...,{ x9,y9 },{ x0,y0 } }`.
`mprop`	Same as for `Trig6IsoPMembraneStiffness`.
`fprop`	Fabrication properties. The plate thickness specified as a one-entry list: `{ h }`, as a ten-entry list: `{ h1,h2,h3,h4,...,h9,h0 }`, or as an empty list: `{ }`. The one-entry form specifies uniform thickness `h`. The ten-entry form is used to specify an element of variable thickness, in which case the entries are the ten nodal thicknesses and h is interpolated cubically. If an empty list appears the module assumes a uniform unit thickness.
`options`	Processing options. This list may contain two items: `{ numer,rule }` or one: `{ numer }`. `numer` is a flag with value `True` or `False`. If `True`, the computations are forced to go 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 this element both the 6-point and 7-point rules are sufficient to produce the correct rank. If omitted `rule` = 6 is assumed. See Remark below.

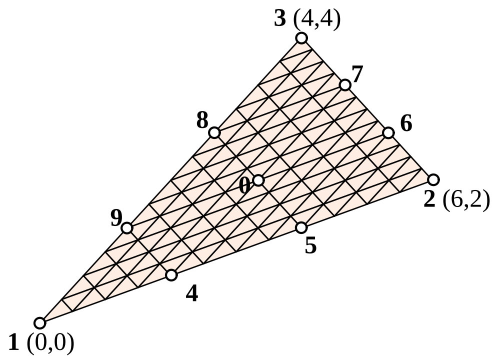


Figure 24.11. Test 10-node triangle element geometry.

The module returns Ke as an 20×20 symmetric matrix pertaining to the following arrangement of nodal displacements:

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad u_{y2} \quad u_{x3} \quad u_{y3} \quad u_{x4} \quad u_{y4} \quad \ldots \quad u_{x9} \quad u_{y9} \quad u_{x0} \quad u_{y0}\,]^T . \tag{24.44}$$

REMARK 24.4

For symbolic work with this element the 7-point rule should be preferred because the exact expressions of the abcissas and weigths at sample points are much simpler than for the 6-point rule, as can be observed in Figure 24.3. This greatly 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 24.11, already used for the quadratic triangle. The corner nodes are at (0, 0), (4, 2) and 6, 4), with the side nodes at the thirdpoints of the sides and the interior node at the centroid. The following test statements form $\mathbf{K}^{(e)}$ for isotropic material with $E = 1920$, $\nu = 0$ and $h = 1$, using exact arithmetic:

```
ClearAll[Em,nu,a,b,e,h]; h=1; Em=1920; nu=0;
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-nu^2)*{{1,nu,0},{nu,1,0},{0,0,(1-nu)/2}};
Ke=Trig10IsoPMembraneStiffness[ncoor,{Emat,0,0},{h},{False,7}];
Ke=Simplify[Ke];  Print[Ke//MatrixForm]; ev=Chop[Eigenvalues[N[Ke]]];
Print["eigs of Ke=",SetPrecision[ev,5]]
```

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)} = \left[\begin{array}{rrrrrrrrrrrrrrrrrrrr}
204 & 102 & -42 & 0 & 0 & -21 & -324 & 9 & 162 & 9 & -18 & -9 & -18 & -9 & 18 & 81 & 18 & -162 & 0 & 0 \\
102 & 204 & 0 & 42 & -21 & -84 & 9 & 360 & 9 & -126 & -9 & -18 & -9 & -18 & 81 & 306 & -162 & -666 & 0 & 0 \\
-42 & 0 & 816 & -408 & -126 & 84 & 216 & -36 & -270 & -36 & -954 & 648 & 504 & -324 & -72 & 36 & -72 & 36 & 0 & 0 \\
0 & 42 & -408 & 816 & 84 & -210 & -36 & -72 & -36 & 414 & 648 & -1638 & -324 & 792 & 36 & -72 & 36 & -72 & 0 & 0 \\
0 & -21 & -126 & 84 & 612 & -306 & -54 & 27 & -54 & 27 & 486 & -315 & -972 & 657 & 54 & -198 & 54 & 45 & 0 & 0 \\
-21 & -84 & 84 & -210 & -306 & 1428 & 27 & -126 & 27 & -126 & -315 & 846 & 657 & -1584 & -198 & -558 & 45 & 414 & 0 & 0 \\
-324 & 9 & 216 & -36 & -54 & 27 & 3240 & -1215 & -1134 & 243 & 0 & 81 & 0 & 81 & 486 & -324 & -2430 & 1620 & 0 & -486 \\
9 & 360 & -36 & -72 & 27 & -126 & -1215 & 4860 & 243 & -486 & 81 & 324 & 81 & 324 & -324 & 810 & 1620 & -4050 & -486 & -1944 \\
162 & 9 & -270 & -36 & -54 & 27 & -1134 & 243 & 3240 & -1215 & 0 & -405 & 0 & 81 & 486 & -324 & 486 & -324 & -2916 & 1944 \\
9 & -126 & -36 & 414 & 27 & -126 & 243 & -486 & -1215 & 4860 & -405 & -1620 & 81 & 324 & -324 & 810 & -324 & 810 & 1944 & -4860 \\
-18 & -9 & -954 & 648 & 486 & -315 & 0 & 81 & 0 & -405 & 3240 & -1215 & -2106 & 1215 & 162 & 0 & 162 & 0 & -972 & 0 \\
-9 & -18 & 648 & -1638 & -315 & 846 & 81 & 324 & -405 & -1620 & -1215 & 4860 & 1215 & -3402 & 0 & -162 & 0 & -162 & 0 & 972 \\
-18 & -9 & 504 & -324 & -972 & 657 & 0 & 81 & 0 & 81 & -2106 & 1215 & 3240 & -1215 & -810 & 0 & 162 & 0 & 0 & -486 \\
-9 & -18 & -324 & 792 & 657 & -1584 & 81 & 324 & 81 & 324 & 1215 & -3402 & -1215 & 4860 & 0 & 810 & 0 & -162 & -486 & -1944 \\
18 & 81 & -72 & 36 & 54 & -198 & 486 & -324 & 486 & -324 & 162 & 0 & -810 & 0 & 3240 & -1215 & -648 & 0 & -2916 & 1944 \\
81 & 306 & 36 & -72 & -198 & -558 & -324 & 810 & -324 & 810 & 0 & -162 & 0 & 810 & -1215 & 4860 & 0 & -1944 & 1944 & -4860 \\
18 & -162 & -72 & 36 & 54 & 45 & -2430 & 1620 & 486 & -324 & 162 & 0 & 162 & 0 & -648 & 0 & 3240 & -1215 & -972 & 0 \\
-162 & -666 & 36 & -72 & 45 & 414 & 1620 & -4050 & -324 & 810 & 0 & -162 & 0 & -162 & 0 & -1944 & -1215 & 4860 & 0 & 972 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -486 & -2916 & 1944 & -972 & 0 & 0 & -486 & -2916 & 1944 & -972 & 0 & 7776 & -2916 \\
0 & 0 & 0 & 0 & 0 & 0 & -486 & -1944 & 1944 & -4860 & 0 & 972 & -486 & -1944 & 1944 & -4860 & 0 & 972 & -2916 & 11664
\end{array}\right] \tag{24.45}
$$

The eigenvalues are:

$$
\begin{array}{l}
[\,26397. \quad 16597. \quad 14937. \quad 12285. \quad 8900.7 \quad 7626.2 \quad 5417.8 \quad 4088.8 \quad 3466.8 \quad 3046.4 \\
\;\; 1751.3 \quad 1721.4 \quad 797.70 \quad 551.82 \quad 313.22 \quad 254.00 \quad 28.019 \quad 0 \quad 0 \quad 0\,]
\end{array} \tag{24.46}
$$

The 3 zero eigenvalues pertain to the three independent rigid-body modes. The 17 other ones are positive. Consequently the computed $\mathbf{K}^{(e)}$ has the correct rank of 17.

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 3 interior point integration `rule=3`. Test the element for the geometry of the triangle depicted on the left of 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: $K_{11} = 1980$, $K_{18} = 1440$.

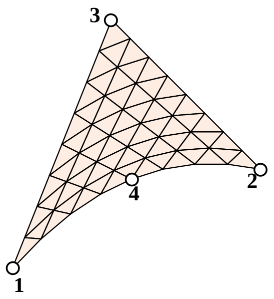


Figure E24.1. The 4-node transition triangle for Exercise 24.1.

EXERCISE 24.2

[A+C:20] 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 (24.35), 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 mistakingly 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

$$2J = 2A = \det \mathbf{J} = x_3 y_{12} + x_1 y_{23} + x_2 y_{31}, \qquad \mathbf{P} = \frac{1}{2J} \begin{bmatrix} y_{23} & y_{32} \\ y_{31} & x_{13} \\ y_{12} & x_{21} \end{bmatrix} \quad \text{(E24.3)}$$

are the same for both (E24.1) and (E24.2). Thus the mistake has no effect on the computation of derivatives.

EXERCISE 24.3

[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:[3]

$$\mathbf{K}^{(e)} = \tfrac{1}{3}Ah\left(\mathbf{B}_1^T\mathbf{E}\mathbf{B}_1 + \mathbf{B}_2^T\mathbf{E}\mathbf{B}_2 + \mathbf{B}_3^T\mathbf{E}\mathbf{B}_3\right) \tag{E24.4}$$

in which A is the triangle area and

$$\begin{aligned}
\mathbf{B}_1 &= \frac{1}{2A}\begin{bmatrix} y_{32} & 0 & y_{31} & 0 & y_{12} & 0 & 2y_{23} & 0 & 2y_{32} & 0 & 2y_{23} & 0 \\ 0 & x_{23} & 0 & x_{13} & 0 & x_{21} & 0 & 2x_{32} & 0 & 2x_{23} & 0 & 2x_{32} \\ x_{23} & y_{32} & x_{13} & y_{31} & x_{21} & y_{12} & 2x_{32} & 2y_{23} & 2x_{23} & 2y_{32} & 2x_{32} & 2y_{23} \end{bmatrix}\\
\mathbf{B}_2 &= \frac{1}{2A}\begin{bmatrix} y_{23} & 0 & y_{13} & 0 & y_{12} & 0 & 2y_{31} & 0 & 2y_{31} & 0 & 2y_{13} & 0 \\ 0 & x_{32} & 0 & x_{31} & 0 & x_{21} & 0 & 2x_{13} & 0 & 2x_{13} & 0 & 2x_{31} \\ x_{32} & y_{23} & x_{31} & y_{13} & x_{21} & y_{12} & 2x_{13} & 2y_{31} & 2x_{13} & 2y_{31} & 2x_{31} & 2y_{13} \end{bmatrix}\\
\mathbf{B}_3 &= \frac{1}{2A}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{21} & 0 & 2y_{21} & 0 & 2y_{12} & 0 & 2y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{12} & 0 & 2x_{12} & 0 & 2x_{21} & 0 & 2x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x_{12} & y_{21} & 2x_{12} & 2y_{21} & 2x_{21} & 2y_{12} & 2x_{21} & 2y_{12} \end{bmatrix}
\end{aligned} \tag{E24.5}$$

Using next the 3-interior-point quadrature rule, show that the element stiffness can be expressed again as (E24.4) but with

$$\begin{aligned}
\mathbf{B}_1 &= \frac{1}{6A}\begin{bmatrix} 5y_{23} & 0 & y_{13} & 0 & y_{21} & 0 & 2y_{21}+6y_{31} & 0 & 2y_{32} & 0 & 6y_{12}+2y_{13} & 0 \\ 0 & 5x_{32} & 0 & x_{31} & 0 & x_{12} & 0 & 2x_{12}+6x_{13} & 0 & 2x_{23} & 0 & 6x_{21}+2x_{31} \\ 5x_{32} & 5y_{23} & x_{31} & y_{13} & x_{12} & y_{21} & 2x_{12}+6x_{13} & 2y_{21}+6y_{31} & 2x_{23} & 2y_{32} & 6x_{21}+2x_{31} & 6y_{12}+2y_{13} \end{bmatrix}\\
\mathbf{B}_2 &= \frac{1}{6A}\begin{bmatrix} y_{32} & 0 & 5y_{31} & 0 & y_{21} & 0 & 2y_{21}+6y_{23} & 0 & 6y_{12}+2y_{32} & 0 & 2y_{13} & 0 \\ 0 & x_{23} & 0 & 5x_{13} & 0 & x_{12} & 0 & 2x_{12}+6x_{32} & 0 & 6x_{21}+2x_{23} & 0 & 2x_{31} \\ x_{23} & y_{32} & 5x_{13} & 5y_{31} & x_{12} & y_{21} & 2x_{12}+6x_{32} & 2y_{21}+6y_{23} & 6x_{21}+2x_{23} & 6y_{12}+2y_{32} & 2x_{31} & 2y_{13} \end{bmatrix}\\
\mathbf{B}_3 &= \frac{1}{6A}\begin{bmatrix} y_{32} & 0 & y_{13} & 0 & 5y_{12} & 0 & 2y_{21} & 0 & 6y_{31}+2y_{32} & 0 & 2y_{13}+6y_{23} & 0 \\ 0 & x_{23} & 0 & x_{31} & 0 & 5x_{21} & 0 & 2x_{12} & 0 & 6x_{13}+2x_{23} & 0 & 2x_{31}+6x_{32} \\ x_{23} & y_{32} & x_{31} & y_{13} & 5x_{21} & 5y_{12} & 2x_{12} & 2y_{21} & 6x_{13}+2x_{23} & 6y_{31}+2y_{32} & 2x_{31}+6x_{32} & 2y_{13}+6y_{23} \end{bmatrix}
\end{aligned} \tag{E24.6}$$

The fact that two very different expressions yield the same $\mathbf{K}^{(e)}$ explain why sometimes authors rediscover the same element derived with different methods.

Yet another set that produces the correct stiffness is

$$\begin{aligned}
\mathbf{B}_1 &= \frac{1}{6A}\begin{bmatrix} y_{23} & 0 & y_{31} & 0 & y_{21} & 0 & 2y_{21} & 0 & 2y_{12} & 0 & 2y_{12} & 0 \\ 0 & x_{32} & 0 & x_{13} & 0 & x_{12} & 0 & 2x_{12} & 0 & 2x_{21} & 0 & 2x_{21} \\ x_{32} & y_{23} & x_{13} & y_{31} & x12 & y_{21} & 2x_{12} & 2y_{21} & 2x_{21} & 2y_{12} & 2x_{21} & 2y_{12} \end{bmatrix}\\
\mathbf{B}_2 &= \frac{1}{6A}\begin{bmatrix} y_{32} & 0 & y_{31} & 0 & y_{12} & 0 & 2y_{23} & 0 & 2y_{32} & 0 & 2y_{23} & 0 \\ 0 & x_{23} & 0 & x_{13} & 0 & x_{21} & 0 & 2x_{32} & 0 & 2x_{23} & 0 & 2x_{32} \\ x_{23} & y_{32} & x_{13} & y_{31} & x_{21} & y_{12} & 2x_{32} & 2y_{23} & 2x_{23} & 2y_{32} & 2x_{32} & 2y_{23} \end{bmatrix}\\
\mathbf{B}_3 &= \frac{1}{6A}\begin{bmatrix} y_{23} & 0 & y_{13} & 0 & y_{12} & 0 & 2y_{31} & 0 & 2y_{31} & 0 & 2y_{13} & 0 \\ 0 & x_{32} & 0 & x_{31} & 0 & x_{21} & 0 & 2x_{13} & 0 & 2x_{13} & 0 & 2x_{31} \\ x_{32} & y_{23} & x_{31} & y_{13} & x_{21} & y_{12} & 2x_{13} & 2y_{31} & 2x_{13} & 2y_{31} & 2x_{31} & 2y_{13} \end{bmatrix}
\end{aligned} \tag{E24.7}$$

EXERCISE 24.4

[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.)

[3] C. A. Felippa, Refined Finite Element Analysis of Linear and Nonlinear Two-Dimensional Structures, Ph.D. Thesis and SESM Report 66–2, Department of Civil Engineering, University of California at Berkeley, Berkeley, CA, 1966. With this form $\mathbf{K}^{(e)}$ can be computed in approximately 1000 floating-point operations.

25 The Assembly Process

TABLE OF CONTENTS

§25.1. INTRODUCTION

Previous Chapters have explained operations for forming element level entities such as stiffness matrices. Sandwiched between element processing and solution there is the *assembly* process, in which the master stiffness equations are constructed.

The position of the assembler in a DSM-based code for static analysis is sketched in Figure 25.1. In practice the assembler is implemented as a *driver* of the element library. That is, instead of forming all elements first and then assembling, the assembler constructs one element at a time, and immediately merges it into the master equations.

Assembly is the most complex stage of a production finite element program in terms of data flow. The complexity is not due to the mathematical operations, which merely involve matrix addition, but to the combined effect of formation of individual elements and merge. Forming an element requires access to node, element, constitutive and fabrication data. Merge requires access to the freedom and connection data, as well as knowledge of the sparse matrix format in which **K** is stored. As illustrated in Figure 25.1, the assembler sits at the crossroads of this process.

The coverage of the assembly process for a general FEM implementation is beyond the scope of an introductory course. Instead this Chapter uses assumptions that lead to drastic simplifications, which in turn allows the basic aspects to be covered with a few examples.

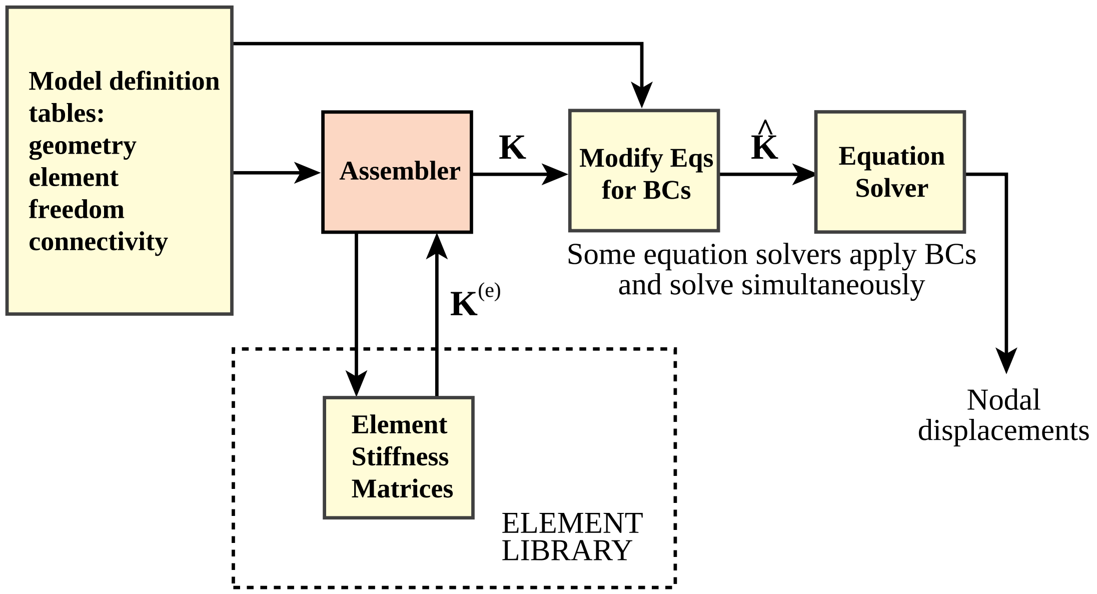

Model definition tables: geometry element freedom connectivity
Assembler
$\mathbf{K}$
Modify Eqs for BCs
$\hat{\mathbf{K}}$
Equation Solver
Some equation solvers apply BCs and solve simultaneously
$\mathbf{K}^{(e)}$
Element Stiffness Matrices
ELEMENT LIBRARY
Nodal displacements

Figure 25.1. Role of assembler in FEM program.

§25.2. SIMPLIFIED ASSEMBLY PROCESS

The assembly process is considerably simplified if the finite element implementation has the following properties:

1. All elements are of the same type. For example: all elements are plane stress elements.
2. The number and configuration of degrees of freedom at each node is the same.
3. There are no “gaps” in the node numbering sequence.
4. There are no multifreedom constraints (MFCs) treated by master-slave or Lagrange multiplier methods.
5. The master stiffness matrix is stored as a full symmetric matrix.

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 merge process.

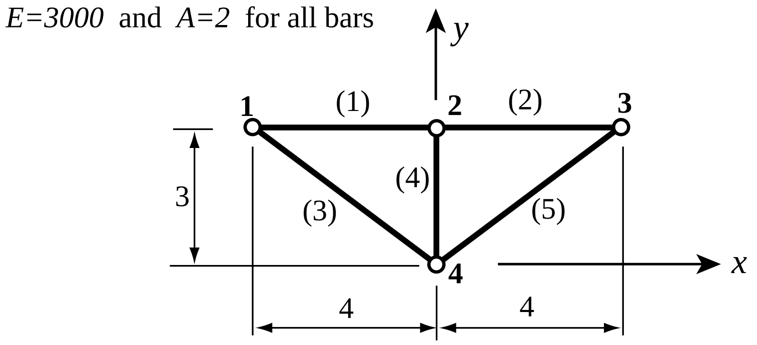


Figure 25.2. Sample structure to illustrate simplified assembly process.

§25.2.1. A Plane Truss Example Structure

The plane truss structure shown in Figure 25.2 will be used to illustrate the details of the assembly process. The structure has 5 elements, 4 nodes and 8 degrees of freedom.

Begin by clearing all entries of the 8×8 matrix $\mathbf{K}$ to zero, so that we effectively start with the null matrix:

$$\mathbf{K} = \begin{bmatrix} 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0&0 \end{bmatrix} \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \mathit{7} \\ \mathit{8} \end{array} \tag{25.1}$$

The numbers written after each row of $\mathbf{K}$ are the *global freedom numbers*.

Element (1) goes from node 1 to 2. The degrees of freedom of these nodes have global numbers $2 \times 1 - 1 = 1$, $2 \times 1 = 1$, $2 \times 2 - 1 = 3$ and $2 \times 2 = 4$. These 4 numbers are collected into an array

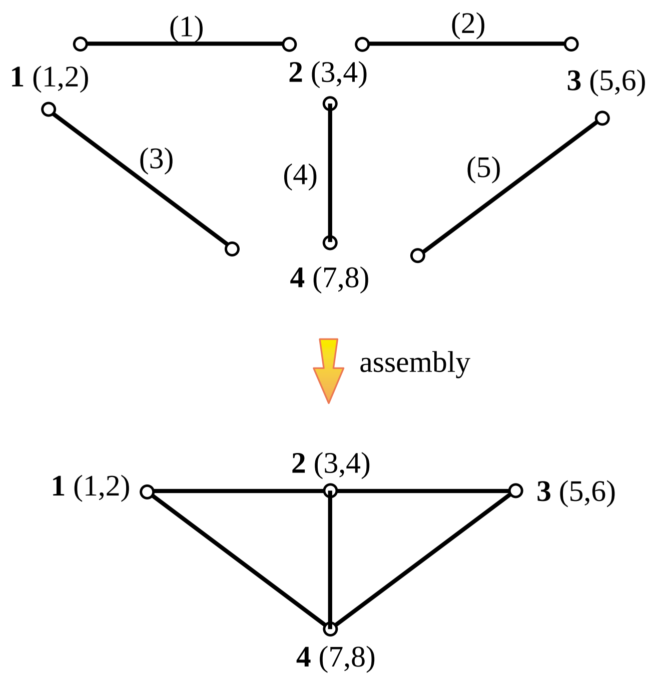


Figure 25.3. Illustration of disassembly/assembly process for plane truss example structure. Numbers in parentheses written after node numbers are the global freedom numbers.

called the *element freedom table*, or EFT for short. The element stiffness matrix is listed below with the EFT entries listed after the matrix rows:

$$\mathbf{K}^{(1)} = \begin{array}{c} \left[\begin{array}{cccc} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \end{array} \begin{array}{c} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \end{array} \tag{25.2}$$

Merging the element into (25.1) gives

$$\mathbf{K} = \left[\begin{array}{cccccccc} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{c} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \\ \\ \\ \\ \end{array} \tag{25.3}$$

Element (2) goes from node 2 to 3. The degrees of freedom at these nodes have global numbers $2 \times 2 - 1 = 3$, $2 \times 2 = 4$, $2 \times 3 - 1 = 5$ and $2 \times 3 = 6$. Hence the EFT is 3,4,5,6, and

$$\mathbf{K}^{(2)} = \left[\begin{array}{cccc} 1500 & 0 & -1500 & 0 \\ 0 & 0 & 0 & 0 \\ -1500 & 0 & 1500 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{c} \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \end{array} \tag{25.4}$$

Merging into (25.3) yields:

$$\mathbf{K} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccccccc} 1500 & 0 & -1500 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] & \begin{array}{c} \\ \\ 3 \\ 4 \\ 5 \\ 6 \\ \\ \\ \end{array}\end{array} \tag{25.5}$$

Element (3) goes from node 1 to 4. Its EFT is 1,2,7,8, and its stiffness is

$$\mathbf{K}^{(3)} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccc} 768 & -576 & -768 & 576 \\ -576 & 432 & 576 & -432 \\ -768 & 576 & 768 & -576 \\ 576 & -432 & -576 & 432 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ 7 \\ 8 \end{array}\end{array} \tag{25.6}$$

Merging this into (25.5) yields:

$$\mathbf{K} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccccccc} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & 0 & 0 & 0 & -576 & 432 \end{array}\right] & \begin{array}{c} 1 \\ 2 \\ \\ \\ \\ \\ 7 \\ 8 \end{array}\end{array} \tag{25.7}$$

Element (4) goes from node 2 to 4. Its EFT is 3,4,7,8, and its stiffness is

$$\mathbf{K}^{(4)} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 2000 & 0 & -2000 \\ 0 & 0 & 0 & 0 \\ 0 & -2000 & 0 & 2000 \end{array}\right] & \begin{array}{c} 3 \\ 4 \\ 7 \\ 8 \end{array}\end{array} \tag{25.8}$$

Merging this into (25.7) yields

$$\mathbf{K} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccccccc} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -768 & 576 & 0 & 0 & 0 & 0 & 768 & -576 \\ 576 & -432 & 0 & -2000 & 0 & 0 & -576 & 2432 \end{array}\right] & \begin{array}{c} \\ \\ 3 \\ 4 \\ \\ \\ 7 \\ 8 \end{array}\end{array} \tag{25.9}$$

Finally, element (5) goes from node 3 to 4. Its EFT is 5,6,7,8, and its element stiffness is

$$\mathbf{K}^{(5)} = \begin{array}{@{}c@{}c@{}}\left[\begin{array}{cccc} 768 & 576 & -768 & -576 \\ 576 & 432 & -576 & -432 \\ -768 & -576 & 768 & 576 \\ -576 & -432 & 576 & 432 \end{array}\right] & \begin{array}{c} 5 \\ 6 \\ 7 \\ 8 \end{array}\end{array} \tag{25.10}$$

Merging this into (25.9) yields

$$\mathbf{K} = \begin{array}{c}\left[\begin{array}{cccccccc} 2268 & -576 & -1500 & 0 & 0 & 0 & -768 & 576 \\ -576 & 432 & 0 & 0 & 0 & 0 & 576 & -432 \\ -1500 & 0 & 3000 & 0 & -1500 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2000 & 0 & 0 & 0 & -2000 \\ 0 & 0 & -1500 & 0 & 2268 & 576 & -768 & -576 \\ 0 & 0 & 0 & 0 & 576 & 432 & -576 & -432 \\ -768 & 576 & 0 & 0 & -768 & -576 & 1536 & 0 \\ 576 & -432 & 0 & -2000 & -576 & -432 & 0 & 2864 \end{array}\right]\end{array}\begin{array}{c} \\ \\ \\ \\ 5 \\ 6 \\ 7 \\ 8 \end{array} \tag{25.11}$$

Since all elements have been processed, (25.11) is the complete free-free master stiffness.

§25.2.2. A Mathematica Implementation

See Cell 25.1 for a *Mathematica* implementation of the assembly process just described. This is a reproduction of the assembler module used in Chapter 22 for a plane truss structure. The listing of the stiffness formation module `Stiffness2DBar`, which is the same as that used in Chapter 22, is omitted to save space.

Running the test statements shown at the end of the cell produces the output shown in Cell 25.2. This reproduces the master stiffness (25.11). The eigenvalue analysis verifies that the assembled stiffness has the correct rank of $8 - 3 = 5$.

§25.3. COMPLICATIONS

We now proceed to follow the assembly of a more complicated structure in which two of the simplifications listed previously do not apply:

1. The structure contains different element types: bars and beam-columns.
2. The nodal freedom configuration varies.

Under these conditions it is not possible to compute the global freedom number directly from the node number. It is necessary to build a Node-to-Freedom Map Table, or NFMT, first. The code to generate an NFMT is the topic of Exercise 25.3.

§25.3.1. Frame-Truss Example Structure

We illustrate the assembly process with the structure shown in Figure 25.4. The finite element discretization, disassembly into elements and assembly are illustrated in Figure 25.5.

The structure is a frame-truss, with two element types: bars and beams (more precisely, beam-columns). It has 11 degrees of freedom, 3 at nodes 1, 2 and 3, and 2 at node 3. They are ordered as follows:

GDOF #:	1	2	3	4	5	6	7	8	9	10	11
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	θ_{z2}	u_{x3}	u_{y3}	θ_{z3}	u_{x4}	u_{y4}

(25.12)

The position of each freedom, as identified by the numbers in the first row, is called the *global freedom number*.

Cell 25.1 - A *Mathematica* Assembler for the Truss Example of Figure 25.2

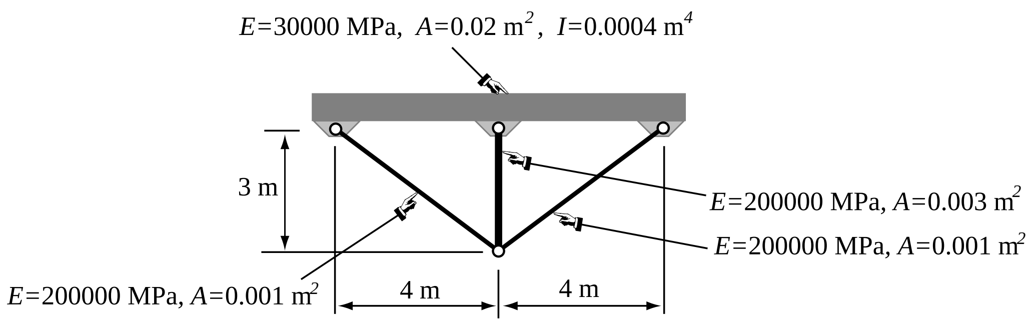
```
AssembleMasterStiffnessPlaneTruss[nodcoor_,elenod_,
  elemat_,elefab_,eleopt_]:=Module[
  {numele=Length[elenod],numnod=Length[nodcoor],
  e,eNL,eftab,ni,nj,i,j,ncoor,mprop,fprop,opt,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];
  For [e=1, e<=numele, e++,
     eNL=elenod[[e]]; {ni,nj}=eNL;
     eftab={2*ni-1,2*ni,2*nj-1,2*nj};
     ncoor={nodcoor[[ni]],nodcoor[[nj]]};
     mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
     Ke=Stiffness2DBar[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]];

(* Stiffness2DBar listing omitted to save space *)

nodcoor={{-4,3},{0,3},{4,3},{0,0}};
elenod= {{1,2},{2,3},{1,4},{2,4},{3,4}};
elemat= Table[{3000,0,0,0},{5}];
elefab= Table[{2},{5}];
eleopt= {True};
K=AssembleMasterStiffnessPlaneTruss[nodcoor,elenod,elemat,elefab,eleopt];
Print["Master Stiffness of Example Truss:"];
Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];
```

Cell 25.2 - Output from the Program of Cell 25.1

```
Master Stiffness of Example Truss:
 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.
Eigs of K={5007.22, 4743.46, 2356.84, 2228.78, 463.703, 0, 0, 0}
```



E=30000 MPa, A=0.02 m^2, I=0.0004 m^4
3 m
E=200000 MPa, A=0.003 m^2
E=200000 MPa, A=0.001 m^2
E=200000 MPa, A=0.001 m^2
4 m
4 m

Figure 25.4. A frame-truss structure that illustrates assembly with multiple element types (beams and bars) and different nodal freedom configurations.

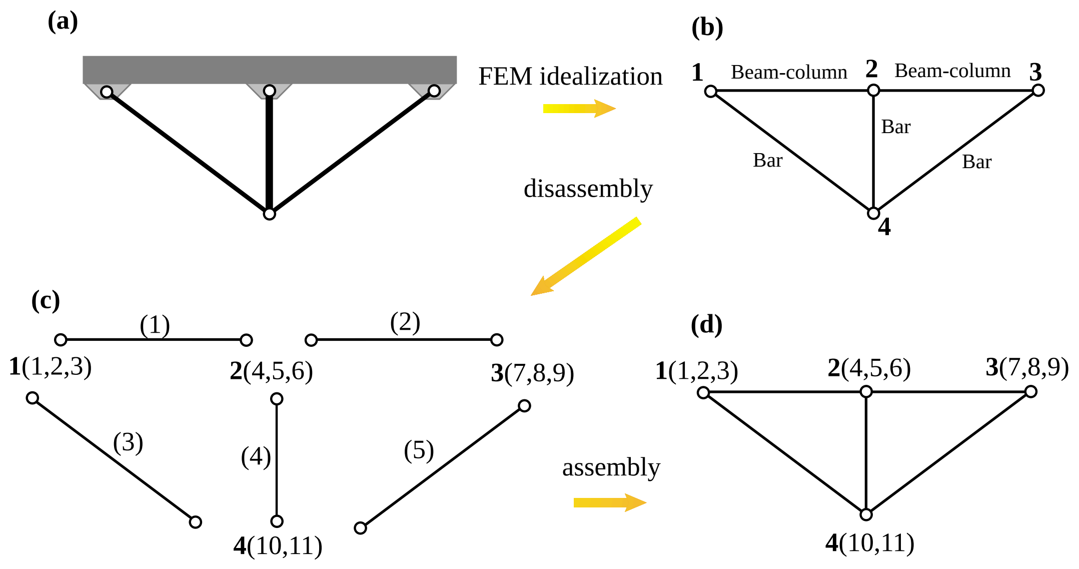


Figure 25.5. Illustration of disassembly/assembly process for frame-truss example structure. Numbers written in parentheses after the node numbers are the global freedom numbers.

We now proceed to go over the assembly process by hand. The master stiffness **K** is displayed as a fully stored matrix for visualization convenience. The process beging by initialization of $\mathbf{K} = \mathbf{0}$.

Element (1): This is a plane beam-column element with freedoms

$$\boxed{\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ u_{x1} & u_{y1} & \theta_{z1} & u_{x2} & u_{y2} & \theta_{z2} \end{array}} \qquad (25.13)$$

The mapping between the element and global freedoms is specified by the EFT

$$\boxed{\begin{array}{lcccccc} \text{element:} & 1 & 2 & 3 & 4 & 5 & 6 \\ \text{assembly:} & 1 & 2 & 3 & 4 & 5 & 6 \end{array}} \qquad (25.14)$$

The element stiffness is

$$\mathbf{K}^{(1)} = \left[\begin{array}{rrrrrr} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \end{array} \tag{25.15}$$

Upon merging this element the master stiffness matrix becomes

$$\mathbf{K} = \left[\begin{array}{cccccccccccc} 150. & 0. & 0. & -150. & 0. & 0. & 0 & 0 & 0 & 0 & 0 \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 0 & 0 \\ 0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0 \\ -150. & 0. & 0. & 150. & 0. & 0. & 0 & 0 & 0 & 0 & 0 \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. & 0 & 0 & 0 & 0 & 0 \\ 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right] \begin{array}{l} \mathit{1} \\ \mathit{2} \\ \mathit{3} \\ \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \\ \\ \\ \\ \\ \end{array} \tag{25.16}$$

Element (2): This is a plane beam-column element with freedoms

1	2	3	4	5	6
u_{x2}	u_{y2}	θ_{z2}	u_{x3}	u_{y3}	θ_{z3}

(25.17)

The mapping between the element and global freedoms is

element:	1	2	3	4	5	6
assembly:	4	5	6	7	8	9

(25.18)

The element stiffness matrix $\mathbf{K}^{(2)}$ is identical to $\mathbf{K}^{(1)}$:

$$\mathbf{K}^{(2)} = \left[\begin{array}{rrrrrr} 150. & 0. & 0. & -150. & 0. & 0. \\ 0. & 22.5 & 45. & 0. & -22.5 & 45. \\ 0. & 45. & 120. & 0. & -45. & 60. \\ -150. & 0. & 0. & 150. & 0. & 0. \\ 0. & -22.5 & -45. & 0. & 22.5 & -45. \\ 0. & 45. & 60. & 0. & -45. & 120. \end{array}\right] \begin{array}{l} \mathit{4} \\ \mathit{5} \\ \mathit{6} \\ \mathit{7} \\ \mathit{8} \\ \mathit{9} \end{array} \tag{25.19}$$

Upon merge the master stiffness matrix becomes

$$\mathbf{K} = \left[\begin{array}{ccccccccccc|c}
150. & 0. & 0. & -150. & 0. & 0. & 0 & 0 & 0 & 0 & 0 & \\
0. & 22.5 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 0 & 0 & \\
0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0 & \\
-150. & 0. & 0. & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0 & \mathit{4} \\
0. & -22.5 & -45. & 0. & 45. & 0. & 0. & -22.5 & 45. & 0 & 0 & \mathit{5} \\
0. & 45. & 60. & 0. & 0. & 240. & 0. & -45. & 60. & 0 & 0 & \mathit{6} \\
0 & 0 & 0 & -150. & 0. & 0. & 150. & 0. & 0. & 0 & 0 & \mathit{7} \\
0 & 0 & 0 & 0. & -22.5 & -45. & 0. & 22.5 & -45. & 0 & 0 & \mathit{8} \\
0 & 0 & 0 & 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0 & \mathit{9} \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 &
\end{array}\right] \tag{25.20}$$

Element (3): This is a plane bar element with freedoms

$$\boxed{\begin{array}{cccc} 1 & 2 & 3 & 4 \\ u_{x1} & u_{y1} & u_{x4} & u_{y4} \end{array}} \tag{25.21}$$

The mapping between the element and global freedoms is

$$\boxed{\begin{array}{lcccc} \text{element:} & 1 & 2 & 3 & 4 \\ \text{assembly:} & 1 & 2 & 10 & 11 \end{array}} \tag{25.22}$$

The element stiffness matrix is

$$\mathbf{K}^{(3)} = \left[\begin{array}{cccc|c}
25.6 & -19.2 & -25.6 & 19.2 & \mathit{1} \\
-19.2 & 14.4 & 19.2 & -14.4 & \mathit{2} \\
-25.6 & 19.2 & 25.6 & -19.2 & \mathit{10} \\
19.2 & -14.4 & -19.2 & 14.4 & \mathit{11}
\end{array}\right] \tag{25.23}$$

Upon merge the master stiffness matrix becomes

$$\mathbf{K} = \left[\begin{array}{ccccccccccc|c}
175.6 & -19.2 & 0. & -150. & 0. & 0. & 0 & 0 & 0 & -25.6 & 19.2 & \mathit{1} \\
-19.2 & 36.9 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 19.2 & -14.4 & \mathit{2} \\
0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0 & \\
-150. & 0. & 0. & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0 & \\
0. & -22.5 & -45. & 0. & 45. & 0. & 0. & -22.5 & 45. & 0 & 0 & \\
0. & 45. & 60. & 0. & 0. & 240. & 0. & -45. & 60. & 0 & 0 & \\
0 & 0 & 0 & -150. & 0. & 0. & 150. & 0. & 0 & 0 & 0 & \\
0 & 0 & 0 & 0. & -22.5 & -45. & 0. & 22.5 & -45. & 0 & 0 & \\
0 & 0 & 0 & 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0 & \\
-25.6 & 19.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25.6 & -19.2 & \mathit{10} \\
19.2 & -14.4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -19.2 & 14.4 & \mathit{11}
\end{array}\right] \tag{25.24}$$

Element (4): This is a plane bar element with freedoms

$$\boxed{\begin{array}{cccc} 1 & 2 & 3 & 4 \\ u_{x2} & u_{y2} & u_{x4} & u_{y4} \end{array}} \tag{25.25}$$

The mapping between the element and global freedoms is

element:	1	2	3	4
assembly:	4	5	10	11

(25.26)

The element stiffness matrix is

$$\mathbf{K}^{(4)} = \left[\begin{array}{cccc} 0 & 0 & 0 & 0 \\ 0 & 200. & 0 & -200. \\ 0 & 0 & 0 & 0 \\ 0 & -200. & 0 & 200. \end{array}\right] \begin{array}{l} \mathit{4} \\ \mathit{5} \\ \mathit{10} \\ \mathit{11} \end{array} \tag{25.27}$$

Upon merge the master stiffness matrix becomes

$$\mathbf{K} = \left[\begin{array}{ccccccccccc} 175.6 & -19.2 & 0. & -150. & 0. & 0. & 0 & 0 & 0 & -25.6 & 19.2 \\ -19.2 & 36.9 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 19.2 & -14.4 \\ 0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0 \\ -150. & 0. & 0. & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0 \\ 0. & -22.5 & -45. & 0. & 245. & 0. & 0. & -22.5 & 45. & 0 & -200. \\ 0. & 45. & 60. & 0. & 0. & 240. & 0. & -45. & 60. & 0 & 0 \\ 0 & 0 & 0 & -150. & 0. & 0. & 150. & 0. & 0 & 0 & 0 \\ 0 & 0 & 0 & 0. & -22.5 & -45. & 0. & 22.5 & -45. & 0 & 0 \\ 0 & 0 & 0 & 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0 \\ -25.6 & 19.2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 25.6 & -19.2 \\ 19.2 & -14.4 & 0 & 0 & -200. & 0 & 0 & 0 & 0 & -19.2 & 214.4 \end{array}\right] \begin{array}{l} \\ \\ \\ \mathit{4} \\ \mathit{5} \\ \\ \\ \\ \\ \mathit{10} \\ \mathit{11} \end{array} \tag{25.28}$$

Element (5): This is a plane bar element with freedoms

1	2	3	4
u_{x2}	u_{y2}	u_{x4}	u_{y4}

(25.29)

The mapping between the element and global freedoms is

element:	1	2	3	4
assembly:	7	8	10	11

(25.30)

The element stiffness matrix is

$$\mathbf{K}^{(5)} = \left[\begin{array}{cccc} 25.6 & 19.2 & -25.6 & -19.2 \\ 19.2 & 14.4 & -19.2 & -14.4 \\ -25.6 & -19.2 & 25.6 & 19.2 \\ -19.2 & -14.4 & 19.2 & 14.4 \end{array}\right] \begin{array}{l} \mathit{7} \\ \mathit{8} \\ \mathit{10} \\ \mathit{11} \end{array} \tag{25.31}$$

Upon merge the master stiffness matrix becomes

$$\mathbf{K}=\left[\begin{array}{ccccccccccc}
175.6 & -19.2 & 0. & -150. & 0. & 0. & 0 & 0 & 0 & -25.6 & 19.2\\
-19.2 & 36.9 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 19.2 & -14.4\\
0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0\\
-150. & 0. & 0. & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0\\
0. & -22.5 & -45. & 0. & 245. & 0. & 0. & -22.5 & 45. & 0 & -200.\\
0. & 45. & 60. & 0. & 0. & 240. & 0. & -45. & 60. & 0 & 0\\
0 & 0 & 0 & -150. & 0. & 0. & 175.6 & 19.2 & 0 & -25.6 & -19.2\\
0 & 0 & 0 & 0. & -22.5 & -45. & 19.2 & 36.9 & -45. & -19.2 & -14.4\\
0 & 0 & 0 & 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0\\
-25.6 & 19.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 & 51.2 & 0\\
19.2 & -14.4 & 0 & 0 & -200. & 0 & -19.2 & -14.4 & 0 & 0 & 228.8
\end{array}\right]\begin{array}{l}
\\ \\ \\ \\ \\ \\ \mathit{7}\\ \mathit{8}\\ \\ \mathit{10}\\ \mathit{11}
\end{array}\tag{25.32}$$

Since all elements have been processed, (25.32) is the master stiffness matrix of the example structure of Figure 25.4. That it has the proper rank of $11 - 3 = 8$ may be verified by an eigenvalue analysis.

REMARK 25.1

For storage as a skyline matrix the connection between nodes 1 and 3 would be out of the skyline envelope. Omitting the lower triangle the skyline template for (25.32) looks like

$$\mathbf{K}=\left[\begin{array}{ccccccccccc}
175.6 & -19.2 & 0. & -150. & 0. & 0. & & & & -25.6 & 19.2\\
 & 36.9 & 45. & 0. & -22.5 & 45. & & & & 19.2 & -14.4\\
 & & 120. & 0. & -45. & 60. & & & & 0 & 0\\
 & & & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0\\
 & & & & 245. & 0. & 0. & -22.5 & 45. & 0 & -200.\\
 & & & & & 240. & 0. & -45. & 60. & 0 & 0\\
 & & & & & & 175.6 & 19.2 & 0. & -25.6 & -19.2\\
 & & & & & & & 36.9 & -45. & -19.2 & -14.4\\
 & & & & & & & & 120. & 0 & 0\\
 & & & & & & & & & 51.2 & 0.\\
\mathit{symm} & & & & & & & & & & 228.8
\end{array}\right]\tag{25.33}$$

The diagonal location pointers of (25.33) are defined by the table

```
DLT = { 0,1,3,6,10,15,21,25,30,36,46,57 }
```
(25.34)

Examination of the skyline template (25.33) reveals that some additional zero entries could be removed from the template; for example K_{13}. The fact that those entries are exactly 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.

§25.3.2. A Mathematica Implementation

Cell 25.3 gives a *Mathematica* implementation of the assembly process for the frame-truss structure just described. This is done by module `AssembleMasterStiffnessPlaneFrameTruss`. The two subordinate modules `Stiffness2DBar` and `Stiffness2DBeamColumn`, which implement the stiffness matrix calculation for a plane bar and beam-column, respectively, are omitted to save space. These are the same used in Chapter 21.

Cell 25.3 - A Mathematica Assembler for the Frame-Truss Example of Figure 25.4

```
AssembleMasterStiffnessPlaneFrameTruss[nodcoor_,eletyp_,
   elenod_,elemat_,elefab_,eleopt_,NFCT_]:=Module[
 {numele=Length[elenod],numnod=Length[nodcoor],
  numdof,e,eNL,eftab,n,ni,nj,i,j,k,m,mi,mj,
  ncoor,mprop,fprop,opt,NFMT,Ke,K},  NFMT=Table[0,{numnod}];
  m=0; For [n=1,n<=numnod,n++, k=NFCT[[n]];
       If [k>0, NFMT[[n]]=m; m+=k, NFMT[[n]]=-1]]; numdof=m;
  K=Table[0,{numdof},{numdof}];
  For [e=1, e<=numele, e++,
     eNL=elenod[[e]]; {ni,nj}=eNL; mi=NFMT[[ni]]; mj=NFMT[[nj]];
     ncoor={nodcoor[[ni]],nodcoor[[nj]]};
     mprop=elemat[[e]]; fprop=elefab[[e]]; opt=eleopt;
     If [eletyp[[e]]=="Bar",
         {ni,nj}=eNL; mi=NFMT[[ni]]; mj=NFMT[[nj]];
         ncoor={nodcoor[[ni]],nodcoor[[nj]]};
         eftab={mi+1,mi+2,mj+1,mj+2};
         Ke=Stiffness2DBar[ncoor,mprop,fprop,opt]];
     If [eletyp[[e]]=="BeamColumn",
        {ni,nj}=eNL; mi=NFMT[[ni]]; mj=NFMT[[nj]];
         ncoor={nodcoor[[ni]],nodcoor[[nj]]};
         eftab={mi+1,mi+2,mi+3,mj+1,mj+2,mj+3};
         Ke=Stiffness2DBeamColumn[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]
  ];

(* Stiffness2DBar and Stiffness2DBeamColumn listings omitted to save space *)

nodcoor={{-4,3},{0,3},{4,3},{0,0}};
eletyp= {"BeamColumn","BeamColumn","Bar","Bar","Bar"};
elenod= {{1,2},{2,3},{1,4},{2,4},{3,4}};
elemat= Join[Table[{30000,0,0,0},{2}],Table[{200000,0,0,0},{3}]];
elefab= {{0.02,0.004},{0.02,0.004},{0.001},{0.003},{0.001}};
eleopt= {True}; NFCT= {3,3,3,2};
K=AssembleMasterStiffnessPlaneFrameTruss[nodcoor,eletyp,
       elenod,elemat,elefab,eleopt,NFCT]; K=Chop[K];
Print["Master Stiffness of Example Frame-Truss:"];
Print[K//MatrixForm];
Print["Eigs of K=",Chop[Eigenvalues[N[K]]]];
```

Comparing the logic of `AssembleMasterStiffnessPlaneFrameTruss` to that of Cell 25.1, an increase in complexity is evident. This assembler has two additional arguments:

Cell 25.4 - Output from the Program of Cell 25.3

```
Master Stiffness of Example Frame-Truss:
 175.6  -19.2    0.  -150.    0.     0.    0       0      0  -25.6   19.2
 -19.2   36.9   45.     0.  -22.5   45.    0       0      0   19.2  -14.4
   0.    45.   120.     0.  -45.    60.    0       0      0   0      0
-150.     0.     0.   300.    0.     0. -150.     0.     0.  0      0
   0.   -22.5  -45.     0.  245.     0.    0.   -22.5   45.   0   -200.
   0.    45.    60.     0.    0.   240.    0.   -45.    60.   0      0
   0      0      0   -150.    0.     0.  175.6   19.2    0. -25.6  -19.2
   0      0      0      0.  -22.5  -45.   19.2  36.9   -45. -19.2  -14.4
   0      0      0      0.   45.    60.    0.  -45.    120.   0      0
 -25.6   19.2    0      0     0      0    -25.6 -19.2     0   51.2    0.
  19.2  -14.4    0      0  -200.     0    -19.2 -14.4     0    0.   228.8
Eigs of K={460.456, 445.431, 306.321, 188.661, 146.761, 82.7415, 74.181,
           25.4476, 0, 0, 0}
```

`eletyp`	A list of element type identifiers provided as character strings: `"Bar"` for a bar element or `"BeamColumn"` for a beam-column element. For the example frame-truss structure `eletyp` is { `"BeamColumn","BeamColumn","Bar","Bar","Bar"` }.
`NFCT`	The Node Freedom Count Table (NFCT) is an integer array with one entry per node. Entry `NFCT[[n]]=k` if there are `k`$\geq$ 0 degrees of freedom defined at node `n`. If `k` is zero, node `n` has not been defined. The NFCT for the example structure of Figure 25.4 is { `3,3,3,2` }. In Cell 25.3 this array is directly set by the test statements. The construction of this table from more primitive node freedom data is the subject of Exercise 25.2.

The element fabrication list `elefab` has minor modifications. A plane beam-column element requires two properties: { `A,Izz` }, 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 frame-truss structure `elefab` is { { `0.02,0.004` },{ `0.02,0.004` },{ `0.001` },{ `0.003` }, { `0.001` } } in accordance with Figure 25.4. The element material property list `elemat` is modified on the account that the elastic moduli for the beam and bars is different. The other arguments are the same as in Cell 25.1.

Upon entry the assembler proceeds to build the Node Freedom Map Table or NFMT from NFCT. This is an array with one entry per node. Entry `NFT[[n]]=i` , `i`$\geq$0, if the global freedoms associated with node `n` are `i+1, ... ,i+k`, where `k=NFCT[[n]]`. If node `n` is undefined, `NFMT[[n]]=-1`. For example, the NFMT for the frame-truss example structure is { `0,3,6,9` }. The total number of degrees of freedoms, `numdof`, is obtained as a byproduct, and used to initialize the master stiffness array.

The assembler then loops over the elements and unpacks data. It calls the appropriate element stiffness module according to element type. The NFMT is used to construct the Element Freedom Table (EFT) array. The merging logic (the two `For` loops at the end of the element loop) is common to both elements.

Running the test statements shown at the end of Cell 25.3 produces the output shown in Cell 25.4. This reproduces the master stiffness (25.32). The eigenvalue analysis verifies that the assembled stiffness has the correct rank of $11 - 3 = 8$.

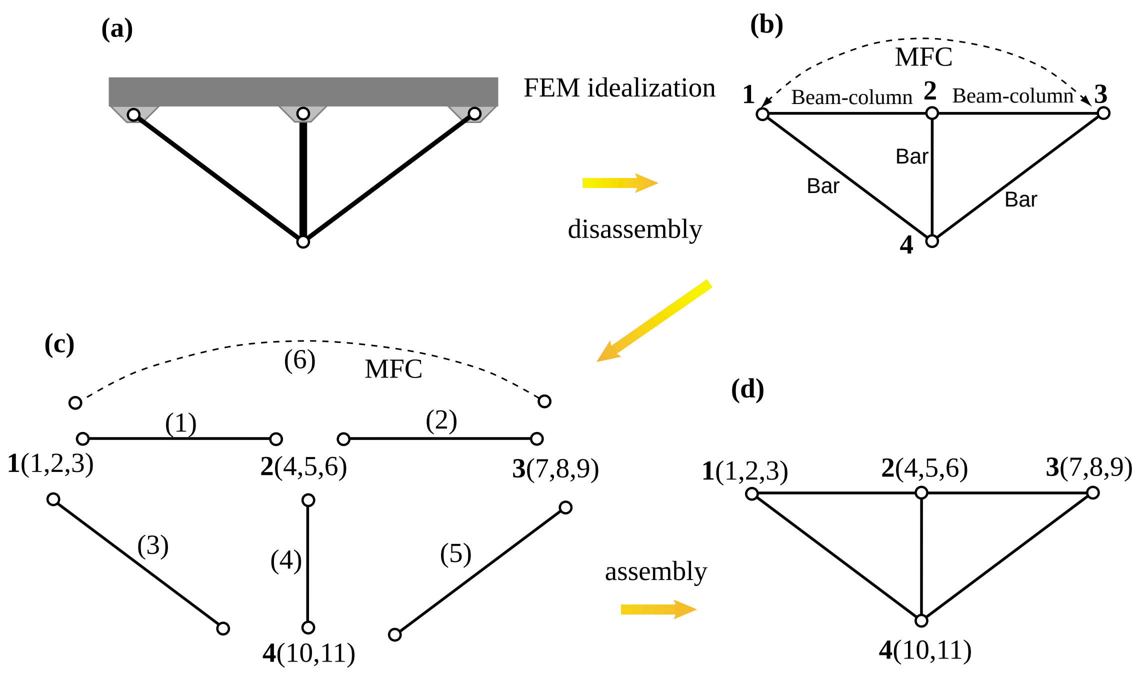


Figure 25.6. Finite element discretization, disassembly and assembly of frame-truss example structure with MFC $u_{y1} = u_{y3}$.

§25.4. *KINEMATIC RELEASES

Occassionally it is necessary to account for *kinematic releases* that preclude freedoms at a node to be the same. This occurs when modeling devices such as hinges and expansion joints. The treatment of these cases by freedom injection or node duplication is left to future version of these Notes.

§25.5. *IMPOSING A MULTIFREEDOM CONSTRAINT

To see the effect of imposing an MFC through the Lagrange multiplier method on the configuration of the master stiffness matrix, suppose that the frame-truss example structure of the previous section is subjected to the constraint that nodes 1 and 3 must move vertically by the same amount. That is,

$$u_{y1} = u_{y3} \qquad \text{or} \qquad u_{y1} - u_{y3} = 0. \tag{25.35}$$

For assembly purposes (25.35) may be viewed as an element labeled as (6). See Figure 25.6.

The degrees of freedom of the assembled structure increase by one to 12. They are ordered as follows:

GDOF #:	1	2	3	4	5	6	7	8	9	10	11	12
DOF:	u_{x1}	u_{y1}	θ_{z1}	u_{x2}	u_{y2}	θ_{z2}	u_{x3}	u_{y3}	θ_{z3}	u_{x4}	u_{y4}	$\lambda^{(6)}$

(25.36)

The assembly of the first five elements proceeds as explained before, and produces the same master stiffness as

(25.32), except for an extra zero row and column. Processing the MFC element (6) yields

$$\mathbf{K} = \begin{bmatrix} 175.6 & -19.2 & 0. & -150. & 0. & 0. & 0 & 0 & 0 & -25.6 & 19.2 & 0 \\ -19.2 & 36.9 & 45. & 0. & -22.5 & 45. & 0 & 0 & 0 & 19.2 & -14.4 & 1. \\ 0. & 45. & 120. & 0. & -45. & 60. & 0 & 0 & 0 & 0 & 0 & 0 \\ -150. & 0. & 0. & 300. & 0. & 0. & -150. & 0. & 0. & 0 & 0 & 0 \\ 0. & -22.5 & -45. & 0. & 245. & 0. & 0. & -22.5 & 45. & 0 & -200. & 0 \\ 0. & 45. & 60. & 0. & 0. & 240. & 0. & -45. & 60. & 0 & 0 & 0 \\ 0 & 0 & 0 & -150. & 0. & 0. & 175.6 & 19.2 & 0 & -25.6 & -19.2 & 0 \\ 0 & 0 & 0 & 0. & -22.5 & -45. & 19.2 & 36.9 & -45. & -19.2 & -14.4 & -1. \\ 0 & 0 & 0 & 0. & 45. & 60. & 0. & -45. & 120. & 0 & 0 & 0 \\ -25.6 & 19.2 & 0 & 0 & 0 & 0 & -25.6 & -19.2 & 0 & 51.2 & 0 & 0 \\ 19.2 & -14.4 & 0 & 0 & -200. & 0 & -19.2 & -14.4 & 0 & 0 & 228.8 & 0 \\ 0 & 1. & 0 & 0 & 0 & 0 & 0 & -1. & 0 & 0 & 0 & 0 \end{bmatrix} \tag{25.37}$$

in which the coefficients 1 and -1 associated with the MFC (25.35) end up in the last row and column.

Homework Exercises for Chapter 25: The Assembly Process

EXERCISE 25.1

[A+C:5+10] The Freedom Activity Table, or FAT, is a two-dimensional data structure that marks which degrees of freedoms (DOF) are specified at each node. Entry `FAT[[n]]` is a one-dimensional list of freedom activity tags. For example, suppose that[1] up to six DOF: three translations and three rotations ordered as

$$u_{xn}, u_{yn}, u_{zn}, \theta_{xn}, \theta_{yn}, \theta_{zn} \tag{E25.1}$$

can be specified at each node $n = 1, 2, \ldots$. Then each entry of the FAT If the j^{th} freedom is defined it is identified by a one, and if undefined by a zero. The configuration of the FAT for the plane truss example structure of Figure 25.2 is

```
FAT = {{1,1,0,0,0,0}, {1,1,0,0,0,0}, {1,1,0,0,0,0}, {1,1,0,0,0,0}}
```

which says that only u_{xn} and u_{yn} are defined at each of the four nodes.

(a) Show how the FAT of the frame-truss structure of Figure 25.4 looks like if this marking scheme is used. (Note: θ_n is the same as θ_{zn}.)

(b) Suppose that the nodes of the frame-truss structure of Figure 25.4 are numbered 1, 2, 5 and 6 instead of 1, 2, 3 and 4. In other words, there is a "node gap" in that nodes 3 and 4 are undefined. Show how the FAT would then look like.

EXERCISE 25.2

[A+C:15] The Node Freedom Count Table, or NFCT, is an array with one entry per node. Entry `NFT[[n]]=k` if there are `k`$\geq$ 0 defined DOFs at node `n`. If `k` is zero, node `n` has not been defined. For example, the NFCT for the frame-truss structure of Figure 25.4 is

$$\texttt{NFCT = \{3,3,3,2\}} \tag{E25.2}$$

Write a `Mathematica` module `NodeFreedomCountTable` that receives the FAT as only argument and returns NFCT. Test the module for the frame-truss example structure. Hint: a one-liner implementation is

```
NodeFreedomCountTable[FAT_]:=Table[Sum[FAT[[n,i]],{i,1,6}],{n,1,Length[FAT]}];
```

but this is difficult to figure out. Try a more readable form.

EXERCISE 25.3

[A+C:15] The Node Freedom Map Table, or NFMT, is an array with one entry per node. Entry `NFMT[[n]]=i` with `i`$\geq$`0`, if the global freedoms associated with node `n` are `i+1, ... ,i+k`, where `k=NFCT[[n]]`. If node `n` is undefined, `NFMT[[n]]=-1`.[2] For example, the NFMT for the frame-truss structure of Figure 25.4 is

$$\texttt{NFMT = \{0,3,6,9\}} \tag{E25.3}$$

Write a `Mathematica` module `NodeFreedomMapTable` that receives the NFCT as only argument and returns NFMT. Test the module for this example structure. Hint: the logic can be copied from code in Chapter 25 if you know where to look.

[1] This is actually the scheme implemented in the NASTRAN commercial FEM code

[2] Other schemes are used in FEM codes; this is just one of several possible choices.

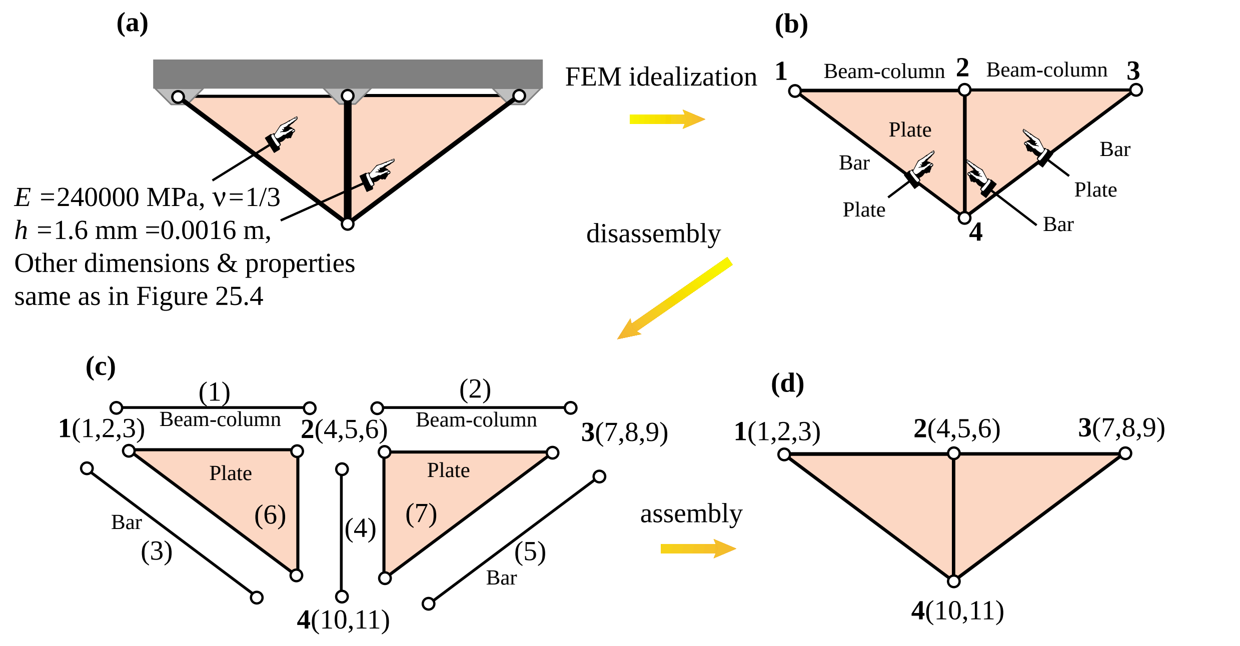


Figure E25.1. Plate-reinforced frame-truss structure for Exercise 25.4.

EXERCISE 25.4

[A/C:25] The frame-truss structure of Figure 25.4 is reinforced with two triangular steel plates attached as shown in Figure E25.1(a). The plate thickness is 1.6 mm= 0.0016 m; the material is isotropic with $E = 240000$ MPa and $\nu = 1/3$. The reinforcing plates are modeled with two plane stress 3-node linear triangles numbered (6) and (7), as illustrated in Figure E25.1(b,c). Compute the master stiffness matrix **K** of the structure.[3]

This exercise may be done through *Mathematica*. For this download Notebook `ExampleAssemblers.nb` from this Chapter index, and complete Cell 3 by writing the assembler.[4] The plate element identifier is `"Trig3"`. Debugging hint: check that matrix (25.32) is obtained if the plate thickness h_{plate} is (temporarily) set to zero.

Target:

$$\mathbf{K} = \begin{bmatrix} 337.6 & -19.2 & 0 & -312. & -72. & 0 & 0 & 0 & 0 & -25.6 & 91.2 \\ -19.2 & 90.9 & 45. & -72. & -76.5 & 45. & 0 & 0 & 0 & 91.2 & -14.4 \\ 0 & 45. & 120. & 0 & -45. & 60. & 0 & 0 & 0 & 0 & 0 \\ -312. & -72. & 0 & 816. & 0 & 0 & -312. & 72. & 0 & -192. & 0 \\ -72. & -76.5 & -45. & 0 & 929. & 0 & 72. & -76.5 & 45. & 0 & -776. \\ 0 & 45. & 60. & 0 & 0 & 240. & 0 & -45. & 60. & 0 & 0 \\ 0 & 0 & 0 & -312. & 72. & 0 & 337.6 & 19.2 & 0 & -25.6 & -91.2 \\ 0 & 0 & 0 & 72. & -76.5 & -45. & 19.2 & 90.9 & -45. & -91.2 & -14.4 \\ 0 & 0 & 0 & 0 & 45. & 60. & 0 & -45. & 120. & 0 & 0 \\ -25.6 & 91.2 & 0 & -192. & 0 & 0 & -25.6 & -91.2 & 0 & 243.2 & 0 \\ 91.2 & -14.4 & 0 & 0 & -776. & 0 & -91.2 & -14.4 & 0 & 0 & 804.8 \end{bmatrix} \qquad \text{(E25.4)}$$

[3] This structure would violate the compatibility requirements stated in Chapter 19 unless the beams are allow to deflect laterally independently of the plates. This is the fabrication sketched in Figure E25.1(a).

[4] Cells 1 and 2 contain working assemblers for plane truss and plane frames, respectively, which may be used as templates.

26
Solving FEM Equations

TABLE OF CONTENTS

§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 basic rules that govern the assembly process are 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

$$\mathbf{K}\mathbf{u} = \mathbf{f} \tag{26.1}$$

where **K** is the master stiffness matrix, **f** the vector of node forces and **u** the vector or 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 suoercomputers. Presently the record is about 50 million.

In *linear* FEM analysis the cost of solving this system of equations rapidly overwhelms other computational phases. Much attention has therefore 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. Handling the master stiffness matrix, however, presents computational difficulties.

§26.1.1. The Curse of Fullness

If **K** is stored and processed as if it were a *full* matrix, the storage and processing time resources rapidly becomes 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.

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 **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, whereas a supercomputer may be able to sustain 10^9 or more. These times assume that the entire matrix is kept in high-speed memory; for otherwise the elapsed time may increase by factors of 10 or more due to I/O transfer operations. The elaspsed timed estimated given in Table 26.1 illustrate that for present computer resources, orders above 10^4 would pose significant computational difficulties.

Table 26.1 Storage & Solution Time for a Fully-Stored Stiffness Matrix

Matrix order N	Storage (double prec)	Factor op.units	Factor time workstation	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	Factor time workstation	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	8000 MB	$10^{12}/2$	15 hrs	8 min

§26.1.2. The Advantages of Sparsity

Fortunately a very high percentage of the entries of the master stiffness matrix **K** are zero. Such matrices are call *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 use 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 bve 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.

In preparation for assembling **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 **K** is best illustrated through a simple example. Consider the 6×6 stiffness matrix

$$\mathbf{K} = \begin{bmatrix} K_{11} & 0 & K_{13} & 0 & 0 & K_{16} \\ & K_{22} & 0 & K_{24} & 0 & 0 \\ & & K_{33} & K_{34} & 0 & 0 \\ & & & K_{44} & 0 & K_{46} \\ & & & & K_{55} & K_{56} \\ symm & & & & & K_{66} \end{bmatrix} \tag{26.2}$$

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 **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:

$$\mathbf{K} = \begin{bmatrix} K_{11} & & K_{13} & & & K_{16} \\ & K_{22} & 0 & K_{24} & & 0 \\ & & K_{33} & K_{34} & & 0 \\ & & & K_{44} & & K_{46} \\ & & & & K_{55} & K_{56} \\ symm & & & & & K_{66} \end{bmatrix} \tag{26.3}$$

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*.

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*:

$$\texttt{s}: \quad [\,K_{11},\ K_{22},\ K_{13},\ 0,\ K_{33},\ K_{24},\ K_{34},\ K_{44},\ K_{55},\ K_{16},\ 0,\ 0,\ K_{46},\ K_{56},\ K_{66}\,] \tag{26.4}$$

This array is complemented by a $(N+1)$ integer array `p` that contains addresses of *diagonal locations*. The array has $N+1$ entries. The $(i+1)^{th}$ entry of `p` has the location of the i^{th} diagonal entry of **K** in `s`. For the example matrix:

$$\texttt{p}: \quad [\,0,\ 1,\ 2,\ 5,\ 8,\ 9,\ 15\,] \tag{26.5}$$

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

$$\mathtt{p}: \quad [\,0,\ 1,\ 2,\ -5,\ 8,\ -9,\ 15\,] \tag{26.6}$$

REMARK 26.1

In Fortran it is convenient to dimension the diagonal location array as `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

$$\mathbf{K} = \mathbf{LDU} = \mathbf{LDL}^T = \mathbf{U}^T\mathbf{DU}, \tag{26.7}$$

where **L** is a unit lower triangular matrix, **D** is a nonsingular diagonal matrix, and **U** and **L** are the transpose of each other. The original matrix is overwritten by the entries of $\mathbf{D}^{-1}$ and **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 **K** has been factored, the solution **u** for a given right hand side **f** is obtained by carrying out three stages:

$$\textit{Forward reduction}: \quad \mathbf{Lz} = \mathbf{f}, \tag{26.8}$$

$$\textit{Diagonal scaling}: \quad \mathbf{Dy} = \mathbf{z}, \tag{26.9}$$

$$\textit{Back substitution}: \quad \mathbf{Uu} = \mathbf{y}, \tag{26.10}$$

where **y** and **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 the present implementation of `MathFET`, MultiFreedom Constraints (MFCs) are treated with Lagrange multiplier. There is one multiplier for each constraint. The multipliers are placed at the end of the solution vector.

Specifically, let the `nummul>0` MFCs be represented in matrix form as $\mathbf{Cu} = \mathbf{g}$, where **C** and **g** are given, and let the `nummul` multipliers be collected in a vector $\boldsymbol{\lambda}$. The multiplier-augmented master stiffness equations are

$$\begin{bmatrix} \mathbf{K} & \mathbf{C}^T \\ \mathbf{C} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{g} \end{bmatrix} \tag{26.11}$$

or

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{26.12}$$

where the symmetric matrix **A**, called a stiffness-bordered matrix, is of order `numdof+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), **A** can no longer be positive definite. In principle pivoting should be used during the factorization of **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 **K** is positive definite, and the constraints are linearly independent (that is, **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* for *MathFET*. 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 `S`. Such a representation consists of a list of two numeric objects:

$$\texttt{S = \{ p, s \}} \tag{26.13}$$

Here `p=GSDLT` is the Global Skyline Diagonal Location Table introduced in §11.6, and `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 factions, or even symbolic entries.

For example, suppose that the numerical entries of the 6×6 skymatrix (26.10) are actually

$$\mathbf{K} = \begin{bmatrix} 11 & & 13 & & & 16 \\ & 22 & 0 & 24 & & 0 \\ & & 33 & 34 & & 0 \\ & & & 44 & & 46 \\ & & & & 55 & 56 \\ symm & & & & & 66 \end{bmatrix} \tag{26.14}$$

Its *Mathematica* representation, using the symbols (26.13) is

```
p= { 0,1,2,5,8,9,15};
s= { 11,22,13,0,33,24,34,44,55,16,0,0,46,56,66};          (26.15)
S= { p, s};
```

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]];
];
```

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}};
```
(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.]

§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 } = SymmSkyMatrixFactor[S,tol]
```

The input arguments are

`S`	The skymatrix to be factored, stored as the two-object list `{p,s}`; see previous subsection.

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];
```

`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× or 10× 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 }, where `du` stores the entries of $\mathbf{D}^{-1}$ in the diagonal locations, and of **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.

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.

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     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     1.0000
Computed x={{1., 3., -4.}, {2., 3., 3.}, {3., 3., -2.}, {4., 3., 1.},

   {5., 3., 0.}}
```

§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

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]
];
```

```
x = SymmSkyMatrixVectorSolve[Sf,b]
```

The input arguments are

`Sf`	The factored matrix returned by `SymmSkyMatrixFactor`.
`b`	The right-hand side vector to be solved for, stored as a single-level (one dimensional) list. If the `i-th` entry of `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 `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 **B**, module `SymmSkyMatrixColBlockSolve` may be invoked as

```
X = SymmSkyMatrixVectorSolve[Sf,B]
```

to provide the solution **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 `B` and `X` are matrices (two-dimensional lists) with the righ-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.

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];*)
```

§26.3.4. *Matrix-Vector Multiply

For various applications it is necessary to form the matrix-vector product

$$\mathbf{b} = \mathbf{S}\mathbf{x} \tag{26.17}$$

where **S** is a symmetric skymatrix and **x** is given.

This is done by module `SymmSkyMatrixVectorMultiply`, which is listed in Cell 26.6. Its arguments are the skymatrix `S` and the vector `x`. The function returns **Sx** in `b`.

Module `SymmSkyMatrixColBlockMultiply` implements the multiplication by a block of vectors stored as columns of a rectangular matrix **X**:

$$\mathbf{B} = \mathbf{S}\mathbf{X} \tag{26.18}$$

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.

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]
 ];
```

§26.3.5. *Printing and Mapping

Module `SymmSkyMatrixUpperTrianglePrint`, listed in Cell 26.8, prints a symmetric skymatrix in upper triangle form. Is 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.[1]

[1] 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,k,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];
```

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

```
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:

$$\mathbf{S} = \mathbf{LDU}, \qquad \text{or} \quad \mathbf{S} = \mathbf{LD}^{-1}\mathbf{U} \tag{26.19}$$

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.

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,k,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 Exercise for Chapter 26

Solving FEM Equations

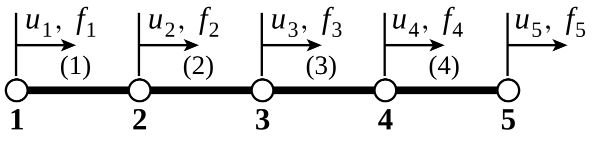


Figure 26.1. Structure for Exercise 26.1

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

$$\mathbf{K}^{(e)} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{E26.1}$$

(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 `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.[2] Show that the entries of $\mathbf{D}$ are 1, 1, 1, 1 and 0, which mean that $\mathbf{K}$ is singular. Why?

[2] 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

§27.1. INTRODUCTION

This Chapter describes a complete finite element plane stress program. Unlike the previous chpaters the description is top down, i.e. starts from the main driver programs.

§27.2. 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. These are quite similar to those steps outlined in Chapter 22 for frame/truss 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 Application of displacement BC. This is done by the equation modification method.

II.3 Solution of the modified equations for displacements. The built in *Mathematica* function `LinearSolve` is used.

Upon executing these three processing steps, the displacements are available The following post-processing steps follow:

III.1 Recovery of forces including reactions, done through the built-in matrix multiplication.

III.2 Computation of element stresses and interelement averaging to get nodal stresses.

III.3 Plotting of displacement and stress fields.

§27.3. MODEL DEFINITION

The input data to a finite element program consists generally of three sets, which are associated with three phases of the program execution:

$$\boxed{\text{Input data}\left\{\begin{array}{l}\text{Preprocessing phase: problem definition data}\\ \text{Processing phase: problem solving data}\\ \text{Postprocessing phase: result display data}\end{array}\right.}$$

§27.4. PROBLEM DEFINITION

The problem-definition data may be broken down into three sets, which are listed below by order of appearance:

$$\boxed{\text{Model definition}\left\{\begin{array}{l}\text{Geometry data: node coordinates}\\ \text{Element data: type, connectivity, material and fabrication}\\ \text{Degree of freedom data: forces and support BCs}\end{array}\right.}$$

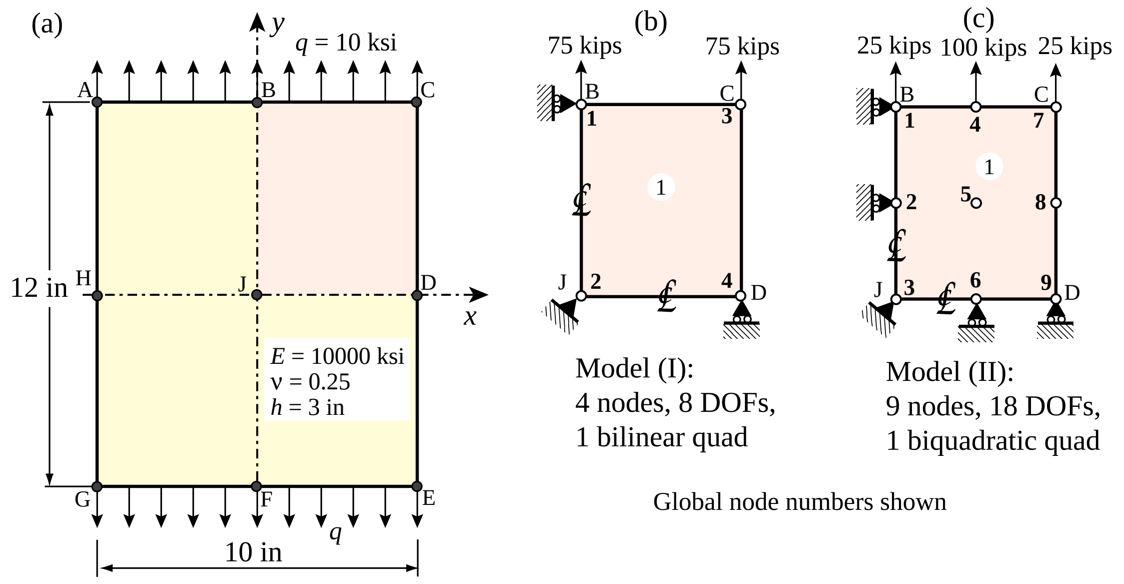


Figure 27.2. Rectangular plate under uniaxial loading, and two one-element FEM discretizations of quadrant BCDJ.

Note that the element data is broken down into four subsets: type, connnectivity, material and fabrication, each of which has its own data structure. The degree of freedom data is broken into two subsets: tag and value. In addition there are miscellaneous process options, which 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*, `ElemNodeLists`, `ElemMaterial`, `ElemFabrication`, `FreedomTags`, `FreedomValues` and `ProcessingOptions`

These data sets are described in the following subsections with reference to the problems and discretizations of Figures 27.1–3.

§27.4.1. Illustrative Problems

Figure 27.1 is a rectangular plate in plane stress under uniform uniaxial loading. Its analytical solution is $\sigma_{yy} = q$, $\sigma_{xx} = \sigma_{xy} = 0$, $u_y = qy/E$, $u_x = -\nu q x/E$. This problem should be solved exactly by *any* finite element mesh. In particular, the two one-element models shown on the right of that figure.

A similar but more complicated problem is shown in Figure 27.2: the axially-loaded rectangular plate of Figure 27.2 now with a central circular hole. This problem is defined in Figure Q3.2. A FE solution is to be obtained using the two quadrilateral element models (4-node and 9-node) depicted in Figure Q3.3. The main result sought is the stress concentration factor on the hole boundary.

§27.4.2. Node Coordinates

The geometry data is specified through `NodeCoordinates`. This is a list of node coordinates configured as

$$\texttt{NodeCoordinates = } \{\{x_1, y_1\},\{x_2, y_2\}, \ \ldots \ \{x_N, y_N\}\} \tag{27.1}$$

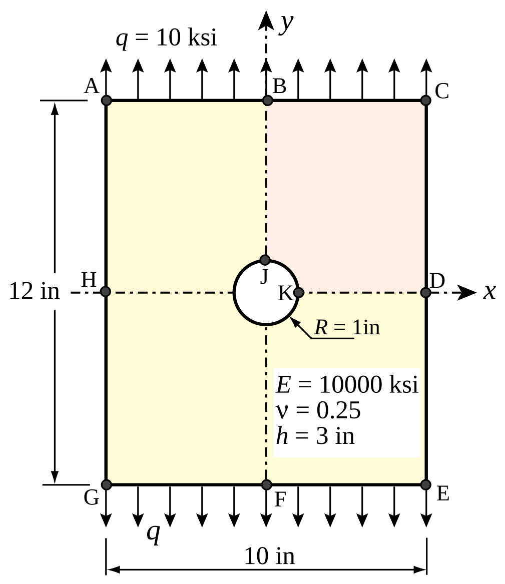


Figure 27.2. Rectangular plate with central circular hole.

where N is the number of nodes. Coordinate values should be floating point numbers; use the `N` function to insure that property if necessary. Nodes must be numbered consecutively and no gaps are permitted.

Example 1. For Model (I) of Figure 27.1:

```
   NodeCoordinates = N[{{0,6},{0,0},{5,6},{5,0}}];
```

Example 2. For Model (II) of Figure 27.1:

```
   NodeCoordinates = N[{{0,6},{0,3},{0,0},{5/2,6},{5/2,3},
                     {5/2,0},{5,6},{5,3},{5,0}}];
```

Example 3. For Model (I) of Figure 27.3, using a bit of coordinate generation along lines:

```
  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}];
```

The result of this generation is that some of the interior nodes are not in the same positions as sketched in Figure 27.3, but that is inconsequential.

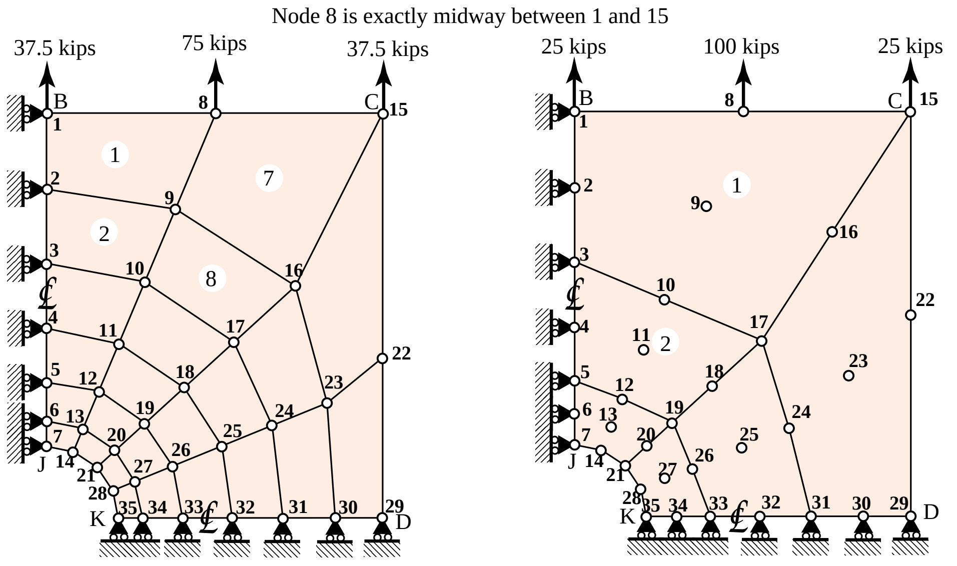

Node 8 is exactly midway between 1 and 15
37.5 kips
75 kips
37.5 kips
25 kips
100 kips
25 kips
B
C
D
K
J
Model (I): 35 nodes, 70 DOFs, 24 bilinear quads
Model (II): 35 nodes, 70 DOFs, 6 biquadratic quads

Figure 27.3. Two FEM discretizations of quadrant BCDKJ of the plate of Figure 27.2. Only a few element numbers are shown to reduce clutter.

§27.4.3. Element Type

Element type is a label that specifies the type of element to be used.

$$\texttt{ElemTypes = \{\{etyp}^{(1)}\texttt{\}, \{etyp}^{(2)}\texttt{\}, ... \{etyp}^{(N_e)}\texttt{\}\}} \tag{27.2}$$

Here etyp(e) is the type descriptor of the e-th element specified as a character string. Allowable types are listed in Table 27.1.

For example, for Model (I) in Figure 27.3:

```
ElemTypes=Table[{"Quad4"},{numele}];
```

and for Model (II):

```
ElemTypes=Table[{"Quad9"},{numele}];
```

Here numele is the number of elements. This can be extracted as numele=Length[ElemNodeLists], where ElemNodeLists is defined below.

§27.4.4. Element Connectivity

Element connectivity information specifies how the elements are connected.[1] This information is stored in ElemNodeLists, which is a list of element nodelists:

$$\texttt{ElemNodeLists = \{\{eNL}^{(1)}\texttt{\}, \{eNL}^{(2)}\texttt{\}, ... \{enL}^{(N_e)}\texttt{\}\}} \tag{27.3}$$

[1] Some FEM programs call this the “topology” data.

Table 27.1 Element Type Descriptors

Identifier	Plane Stress Model
`"Trig3"`	3-node linear triangle
`"Trig6"`	6-node quadratic triangle
`"Trig10"`	10-node cubic triangle
`"Quad4"`	4-node blinear quad
`"Quad9"`	9-node biquadratic quad

where `eNL`$^{(e)}$ denotes the lists of nodes of the element e (given by nglobal node numbers) and N_e is the total number of elements.

Element boundaries must be traversed counterclockwise but you can start at any corner. Numbering elements with midnodes requires more care: first list corners counterclockwise, followed by midpoints (first midpoint is the one that follows first corner when going counterclockwise). When elements have an interior node, as in the 9-node biquadratic quadrilateral, that node goes last.

Example 1. For Model (I) of Figure 27.1, which has only one element:

```
ElemNodeLists=  {{1,2,4,3}};
```

Example 2. For Model (II) of Figure 27.1, which has only one element:

```
ElemNodeLists=  {{1,3,9,7,2,6,8,4,5}};
```

Example 3. For Model (I) of Figure 27.3, numbering the elements from top to bottom and from left to right:

```
ElemNodeLists=Table[{0,0,0,0},{24}];
ElemNodeLists[[1]]={1,2,9,8};
Do [ElemNodeLists[[e]]=ElemNodeLists[[e-1]]+{1,1,1,1},{e,2,6}];
ElemNodeLists[[7]]=ElemNodeLists[[6]]+{2,2,2,2};
Do [ElemNodeLists[[e]]=ElemNodeLists[[e-1]]+{1,1,1,1},{e,8,12}];
ElemNodeLists[[13]]=ElemNodeLists[[12]]+{2,2,2,2};
Do [ElemNodeLists[[e]]=ElemNodeLists[[e-1]]+{1,1,1,1},{e,14,18}];
ElemNodeLists[[19]]=ElemNodeLists[[18]]+{2,2,2,2};
Do [ElemNodeLists[[e]]=ElemNodeLists[[e-1]]+{1,1,1,1},{e,20,24}];
```

Example 4. For Model (II) of Figure 27.3, numbering the elements from top to bottom and from left to right:

```
ElemNodeLists=Table[{0,0,0,0,0,0,0,0,0},{6}];
ElemNodeLists[[1]]={1,3,17,15,2,10,16,8,9};
Do [ElemNodeLists[[e]]=ElemNodeLists[[e-1]]+{2,2,2,2,2,2,2,2,2},{e,2,3}];
ElemNodeLists[[4]]=ElemNodeLists[[3]]+{10,10,10,10,10,10,10,10,10};
Do [ElemNodeLists[[e]]=ElemNodeLists[[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.

§27.4.5. Material Properties

Data structure `ElemMaterial` lists the elastic modulus E and the Poisson's ratio ν for each element:

$$\texttt{ElemMaterial = \{\{} E^{(1)}, \nu^{(1)} \texttt{\}, \{} E^{(2)}, \nu^{(2)} \texttt{\}, ... \{} E^{(N_e)}, \nu^{(N_e)} \texttt{\}\}} \tag{27.4}$$

In the common case in which all elements have the same E and ν, this list can be easily generated by a `Table` instruction.

for all models shown in Figures 27.1 and 27.3,

```
ElemMaterial=Table[{Em,nu},{numele}],
```

in which `numele=Length[ElemeNodeLists]` is the number of elements (24 there) and the values of `Em` and `nu` are typically declared separately.

§27.4.6. Fabrication Properties

`ElemMaterial` lists the plate thickness for each element:

$$\texttt{ElemFabrication = \{\{} h(1) \texttt{\}, \{} h^{(2)} \texttt{\}, ... \{} h^{(N_e)} \texttt{\}\}} \tag{27.5}$$

where N_e is the number of elements.

If all elements have the same thickness, this list can be easily generated by a `Table` instruction. For example, for the model of Figure 27.3, `ElemFabrication=Table[th,{numele}]`, Here `numele` is the number of elements (24 in that case; this can be extracted as the `Length` of `ElemNodeList`) and the value of `th=3` is set at the start of the input cell.

§27.4.7. Freedom Tags

`FreedomTags` labels each node degree of freedom as to whether the load or the displacement is specified. The configuration of this list is similar to that of `NodeCoordinates`:

$$\texttt{FreedomTags=\{\{tag}_{x1}\texttt{, tag}_{y1}\texttt{\},\{tag}_{x2}\texttt{, tag}_{y2}\texttt{\}, ... \{tag}_{xN}\texttt{, tag}_{yN}\texttt{\}\}}$$

The tag value is 0 if the force is specified, else 1. When there are a lot of nodes, the quickest way to specify this list is to start from all zeros, and then insert the boundary conditions appropriately. For example, for the Model (I) of Figure 27.3:

```
numnod=Length[NodeCoordinates];
FreedomTags=Table[{0,0},{numnod}];     (* initialize and reserve space *)
Do[FreedomTags[[n]]={1,0},{n,1,7}];    (* vroller @ nodes 1 through 7 *)
Do[FreedomTags[[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.

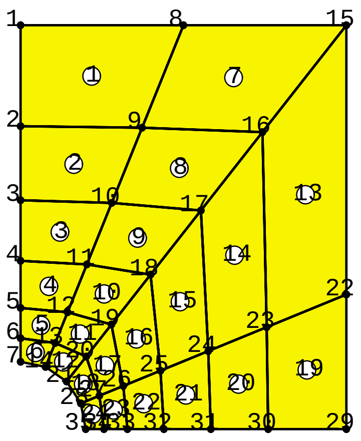


Figure 27.4. Mesh plot showing node and element numbers for Model (I) of Figure chapdot3.

§27.4.8. Freedom Values

`FreedomValues` has the same node by node configuration as `FreedomTags`. It lists the specified values of the applied node force component if the corresponding tag is zero, and of the prescribed displacement component if the tag is one. Typically most of the list entries are zero. For example, in the model (I) of Figure 27.3 only 3 values (for the *y* forces on nodes 1, 8 and 15) will be nonzero:

```
numnod=Length[NodeCoordinates];
FreedomValues=Table[{0,0},{numnod}]; (* initialize and reserve space *)
FreedomValues[[1]]=FreedomValues[[15]]={0,37.5};
FreedomValues[[8]]={0,75};            (* y nodal loads *)
```

§27.4.9. Processing Options

Array `ProcessingOptions` should normally set as follows:

```
ProcessingOptions={True};
```

This specifies floating point numerical computations.

§27.4.10. Mesh Display

Nodes and elements of Model (I) of Figure 27.3 may be plotted by the following statement:

```
aspect=6/5;
Plot2DElementsAndNodes[NodeCoordinates,ElemNodeLists,aspect,
  "Plate with circular hole - 4-node quad model",True,True];
```

Here `aspect` is the plot frame aspect ratio (*y* dimension over *x* dimension), and the last two `True` argument values specify node labels and element labels, respectively. The output is shown in Figure 27.4.

```
MembraneSolution[nodcoor_,eletyp_,elenod_,elemat_,
  elefab_,eleopt_,doftag_,dofval_]:= Module[{K,Kmod,u,f,sig,j,n,ns,
  supdof,supval,numnod=Length[nodcoor],numele=Length[elenod]},
  u=f=sig={};
  K=MembraneMasterStiffness[nodcoor,
          eletyp,elenod,elemat,elefab,eleopt];  K=N[K];
  ns=0; Do [Do [If[doftag[[n,j]]>0,ns++],{j,1,2}],{n,1,numnod}];
  supdof=supval=Table[0,{ns}];
  k=0; Do [Do [If[doftag[[n,j]]>0,k++;supdof[[k]]=2*(n-1)+j;
         supval[[k]]=dofval[[n,j]]]],{j,1,2}],{n,1,numnod}];
  f=ModifiedNodeForces[supdof,supval,K,Flatten[dofval]];
  Kmod=ModifiedMasterStiffness[supdof,K];
  u=Simplify[Inverse[Kmod].f]; u=Chop[u];
  f=Simplify[K.u]; f=Chop[f];
  sig=MembraneNodalStresses[nodcoor,
        eletyp,elenod,elemat,elefab,eleopt,u];
  sig=Chop[sig];
  Return[{u,f,sig}];
];
```

Figure 27.5. Module to carry out the analysis of the plane stress problem.

§27.5. PROCESSING

The static solution is carried out by calling module `LinearSolutionOfPlaneStressModel` shown in Figure 27.5. The function call is

```
{u,f,sig}=MembraneSolution[NodeCoordinates,ElemTypes,
          ElemNodeLists,ElemMaterial,ElemFabrication,
          ProcessingOptions,FreedomTags,FreedomValues];
```

The function begins by assembling the free-free master stiffness matrix `K` by calling the module `MembraneMasterStiffness`. this module is listed in Figure 27.5. As a study of its code reveals, it can handle the five element types described in the previous section. The modules that compute the element stiffness matrices have been studied in previous Chapters and need not be listed here.

The displacement boundary conditions are applied by modules `ModifiedmasterStiffness` and `ModifiedNodeForces`, which return the modified stiffness matrix $\hat{\mathbf{K}}$ and the modified force vector $\hat{\mathbf{f}}$ in `Khat` and `fnat`, respectively. The logic of these modules has been explained in Chapter 22 and need not be described again here.

The unknown node displacements `u` are then obtained through the built in `LinearSolve` function, as `u=LinearSolve[Khat,fhat]`. This solution strategy is of course restricted to very small systems, but it has the advantages of simplicity.

The function returns arrays `u`, `f` and `p`, which are lists of length 12, 12 and 13, respectively. Array `u` contains the computed node displacements ordered $u_{x1} < u_{y1}, u_{x2}, \ldots u_{y8}$. Array `f` contains the node forces recovered from $\mathbf{f} = \mathbf{Ku}$; this includes the reactions f_{x1}, f_{y1} and f_{y8}.

Finally, array `sig` contains the nodal stresses $\sigma_{xx}, \sigma_{yy}, \sigma_{xy}$ at each node, recovered from the displacement solution. This computation is driven by module `MembraneNodeStresses`, which is

```
MembraneMasterStiffness[nodcoor_,eletyp_,elenod_,
  elemat_,elefab_,eleopt_]:=
  Module[{numele=Length[elenod],numnod=Length[nodcoor],numer,
   ne,eNL,eftab,neldof,i,n,Em,ν,Emat,th,ncoor,Ke,K},
  K=Table[0,{2*numnod},{2*numnod}];  numer=eleopt[[1]];
  For [ne=1,ne<=numele,ne++,
       {type}=eletyp[[ne]];
       If [type!="Trig3"&&type!="Trig6"&&type!="Trig10"&&
           type!="Quad4"&&type!="Quad9",
           Print["Illegal element type,ne=",ne]; Return[Null]];
       eNL=elenod[[ne]]; n=Length[eNL];
       eftab=Flatten[Table[{2*eNL[[i]]-1,2*eNL[[i]]},{i,n}]];
       ncoor=Table[nodcoor[[eNL[[i]]]],{i,n}];
       {Em,ν}=elemat[[ne]];
       Emat=Em/(1-ν^2)*{{1,ν,0},{ν,1,0},{0,0,(1-ν)/2}};
       {th}=elefab[[ne]];
       If [type=="Trig3", Ke=Trig3IsoPMembraneStiffness[ncoor,
          {Emat,0,0},{th},{numer}] ];
       If [type=="Quad4", Ke=Quad4IsoPMembraneStiffness[ncoor,
          {Emat,0,0},{th},{numer,2}] ];
       If [type=="Trig6", Ke=Trig6IsoPMembraneStiffness[ncoor,
          {Emat,0,0},{th},{numer,3}] ];
       If [type=="Quad9", Ke=Quad9IsoPMembraneStiffness[ncoor,
          {Emat,0,0},{th},{numer,3}] ];
       If [type=="Trig10",Ke=Trig10IsoPMembraneStiffness[ncoor,
          {Emat,0,0},{th},{numer,3}] ];
       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 27.6. Module to assemble the master stiffness matrix.

listed in Figure 27.7. 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.

§27.6. POSTPROCESSING

Postprocessing are activities undertaken on return from `membraneSoution`. They include optional printout of `u`, `f` and `p`, plus some simple graphic displays described below.

§27.6.1. Displacement Field Contour Plots

Contour plots of the displacement components u_x and u_y interpolated over elements from the computed node displacements cand be obtained by the following script:

```
aspect=6/5; Nsub=4; ux=uy=Table[0,{numnod}];
Do[ux[[n]]=u[[2*n-1]]; uy[[n]]=u[[2*n]], {n,1,numnod}];
uxmax=uymax=0;
```

```
MembraneNodalStresses[nodcoor_,eletyp_,elenod_,
  elemat_,elefab_,eleopt_,u_]:= Module[{numele=Length[elenod],
  numnod=Length[nodcoor],numer,type,ne,eNL,eftab,i,ii,j,k,n,
  Em,ν,Emat,th,ncoor,ue,ncount,esig,sige,sig},
  ncount=Table[0,{numnod}]; {numer}=eleopt;
  sige=  Table[0,{numele}]; sig=Table[{0,0,0},{numnod}];
  For [ne=1,ne<=numele,ne++,
       {type}=eletyp[[ne]];
       eNL=elenod[[ne]]; n=Length[eNL]; ue=Table[0,{2*n}];
       eftab=Flatten[Table[{2*eNL[[i]]-1,2*eNL[[i]]},{i,1,n}]];
       ncoor=Table[nodcoor[[eNL[[i]]]],{i,n}];
       Do [ii=eftab[[i]];ue[[i]]=u[[ii]],{i,1,2*n}];
       {Em,ν}=elemat[[ne]];
       Emat=Em/(1-ν^2)*{{1,ν,0},{ν,1,0},{0,0,(1-ν)/2}};
       th=elefab[[ne]];
       If [type=="Trig3", esig=Trig3IsoPMembraneStresses[ncoor,
          {Emat,0,0},{th},{numer},ue] ];
       If [type=="Quad4", esig=Quad4IsoPMembraneStresses[ncoor,
          {Emat,0,0},{th},{numer,2},ue] ];
       If [type=="Trig6", esig=Trig6IsoPMembraneStresses[ncoor,
          {Emat,0,0},{th},{numer,3},ue] ];
       If [type=="Quad9", esig=Quad9IsoPMembraneStresses[ncoor,
          {Emat,0,0},{th},{numer,3},ue] ];
       If [type=="Trig10",esig=Trig10IsoPMembraneStresses[ncoor,
          {Emat,0,0},{th},{numer,3},ue] ];
       esig=Chop[esig]; sige[[ne]]=esig;
       For [i=1,i<=n,i++,k=eNL[[i]]; ncount[[k]]++;
            Do[ sig[[k,j]]+=esig[[i,j]],{j,3}] ];
      ];
  For [n=1,n<=numnod,n++,k=ncount[[n]];If [k>1,sig[[n]]/=k]
      ];
  Return[sig];
];
```

Figure 27.7. Module to compute averaged nodal stresses.

```
Do[uxmax=Max[Abs[ux[[n]]],uxmax]; uymax=Max[Abs[uy[[n]]],uymax],
     {n,1,numnod}];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodeLists,ux,
     uxmax,Nsub,aspect,"Displacement component ux"];
ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodeLists,uy,
     uymax,Nsub,aspect,"Displacement component uy"];
```

§27.6.2. Stress Field Contour Plots

Contour plots of the stress components σ_{xx}, σ_{yy} and σ_{xy} returned by `MembraneSolution` can be produced by the following script:

```
aspect=6/5; Nsub=4; sxx=syy=sxy=Table[0,{numnod}];
Do[{sxx[[n]],syy[[n]],sxy[[n]]}=sig[[n]],{n,1,numnod}];
sxxmax=syymax=sxymax=0;
Do[sxxmax=Max[Abs[sxx[[n]]],sxxmax];
 syymax=Max[Abs[syy[[n]]],syymax];
```

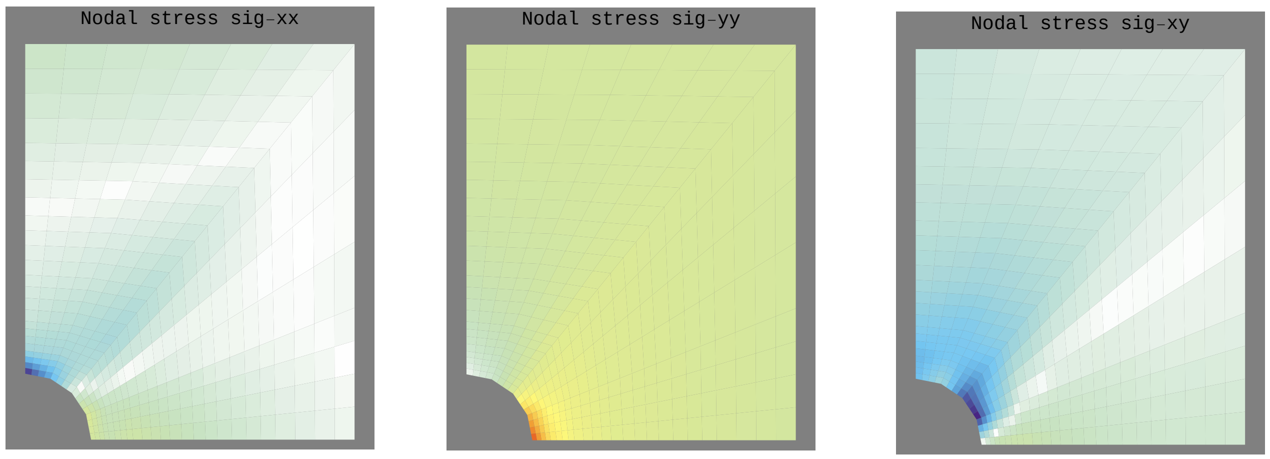


Figure 27.8. Stress contour plots for Model (I) of Figure 27.3.

```
   sxymax=Max[Abs[sxy[[n]]],sxymax], {n,1,numnod}];
  ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodeLists,
       sxx,sxxmax,Nsub,aspect,"Nodal stress sig-xx"];
  ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodeLists,
       syy,syymax,Nsub,aspect,"Nodal stress sig-yy"];
  ContourPlotNodeFuncOver2DMesh[NodeCoordinates,ElemNodeLists,
       sxy,sxymax,Nsub,aspect,"Nodal stress sig-xy"]
```

The stress plots are shown in Figure 27.8.

§27.6.3. Animation

Sometimes it is useful to animate results such as stress fields when a load changes as a function of a time-like parameter t. this can be done by performing a sequence of analysis in a 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 righmost 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.

28 Stress Recovery

TABLE OF CONTENTS

§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 $\boldsymbol{\sigma}$ are directly related to strains $\mathbf{e}$ at each point through the elastic constitutive equations $\boldsymbol{\sigma} = \mathbf{E}\mathbf{e}$. 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 strightforward.

Suppose that we have solved the master stiffness equations

$$\mathbf{K}\mathbf{u} = \mathbf{f}, \tag{28.1}$$

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

$$\mathbf{e} = \mathbf{B}\mathbf{u}^{(e)}, \tag{28.2}$$

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

$$\boldsymbol{\sigma} = \mathbf{E}\mathbf{e} = \mathbf{E}\mathbf{B}\mathbf{u} \tag{28.3}$$

[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 $\boldsymbol{\sigma}$ 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 $\boldsymbol{\sigma}$ 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.

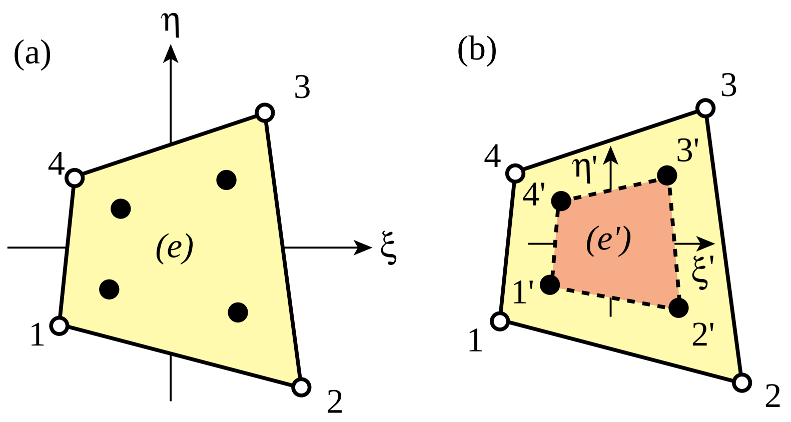


Figure 28.1. Extrapolation from 4-node quad Gauss points: (a) 2×2 rule, (b) Gauss element (e')

§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

$$\xi = \xi'/\sqrt{3}, \quad \eta = \eta'/\sqrt{3}, \qquad \xi' = \xi\sqrt{3}, \quad \eta' = \eta\sqrt{3}. \tag{28.4}$$

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':

$$w(\xi',\eta') = [\, w'_1 \quad w'_2 \quad w'_3 \quad w'_4 \,] \begin{bmatrix} N_1^{(e')} \\ N_2^{(e')} \\ N_3^{(e')} \\ N_4^{(e')} \end{bmatrix}, \tag{28.5}$$

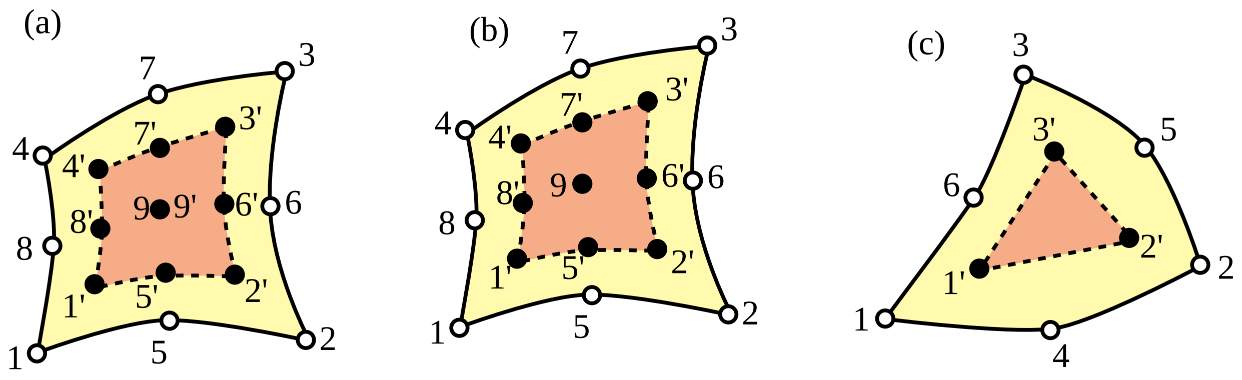


Figure 28.2. Gauss elements for higher order quadrilaterals and triangles: (a) 9-node element with 3 × 3 Gauss rule, (b) 8-node element with 3 × 3 Gauss rule, (c) 6-node element with 3-interior point rule.

where (cf. §15.6.2)

$$\begin{aligned} N_1^{(e')} &= \tfrac{1}{4}(1-\xi')(1-\eta'),\\ N_2^{(e')} &= \tfrac{1}{4}(1+\xi')(1-\eta'),\\ N_3^{(e')} &= \tfrac{1}{4}(1+\xi')(1+\eta'),\\ N_4^{(e')} &= \tfrac{1}{4}(1-\xi')(1+\eta'). \end{aligned} \tag{28.6}$$

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

$$\begin{bmatrix} w_1\\ w_2\\ w_3\\ w_4 \end{bmatrix} = \begin{bmatrix} 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2}\\ -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3}\\ 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} & -\frac{1}{2}\\ -\frac{1}{2} & 1-\frac{1}{2}\sqrt{3} & -\frac{1}{2} & 1+\frac{1}{2}\sqrt{3} \end{bmatrix} \begin{bmatrix} w'_1\\ w'_2\\ w'_3\\ w'_4 \end{bmatrix} \tag{28.7}$$

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.

A
Matrix Algebra: Vectors

§A.1 MOTIVATION

Matrix notation was invented[1] 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, ... x_n, and another set of m quantities, y_1, y_2, ..., y_m:

$$\begin{array}{ccccccccccccc}
a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1j}x_j & + & \cdots & + & a_{1n}x_n & = & y_1 \\
a_{21}x_2 & + & a_{22}x_2 & + & \cdots & + & a_{2j}x_j & + & \cdots & + & a_{2n}x_n & = & y_2 \\
\ldots & & \ldots & & \ldots & & \ldots & & \ldots & & \ldots & & \ldots \\
a_{i1}x_1 & + & a_{i2}x_2 & + & \cdots & + & a_{ij}x_j & + & \cdots & + & a_{in}x_n & = & y_i \\
\ldots & & \ldots & & \ldots & & \ldots & & \ldots & & \ldots & & \ldots \\
a_{m1}x_1 & + & a_{m2}x_2 & + & \cdots & + & a_{mj}x_j & + & \cdots & + & a_{mn}x_n & = & y_m
\end{array} \tag{A.1}$$

The subscripted a, x and y quantities that appear in this set of relations may be formally arranged as follows:

$$\begin{bmatrix}
a_{11} & a_{12} & \ldots & a_{1j} & \ldots & a_{1n} \\
a_{21} & a_{22} & \ldots & a_{2j} & \ldots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
a_{i1} & a_{i2} & \ldots & a_{ij} & \ldots & a_{in} \\
\vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \ldots & a_{mj} & \ldots & a_{mn}
\end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_j \\ \vdots \\ x_n \end{bmatrix}
=
\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_i \\ \vdots \\ y_m \end{bmatrix} \tag{A.2}$$

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.

Replacing the expressions in (A.2) by these symbols we obtain the *matrix form* of (A.1):

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{A.3}$$

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 programming languages that offer array data structures.

[1] By Arthur Cayley at Cambridge (UK) in 1858.

§A.2 VECTORS

We begin by defining a *vector*, a set of *n* numbers which we shall write in the form

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \tag{A.4}$$

A vector of this type 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 **x** is the name of the vector.

If *n* numbers are arranged in a horizontal array, as in

$$\mathbf{z} = [\, z_1 \quad z_2 \quad \ldots \quad z_n \,] \tag{A.5}$$

then **z** is called a *row vector*. If we use the term "vector" 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:

$$\mathbf{a}, \quad \mathbf{b}, \quad \mathbf{c}, \quad \mathbf{x}, \quad \mathbf{y}, \quad \mathbf{z}$$

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:

$$\underset{\sim}{a}$$

The subscripted quantities such as x_1 in (A.4) are called the *entries* or *components*[2] of **x**, while *n* is called the *order* of the vector **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, **z** in the example 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 this 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 it may lead to confusion with finite elements.

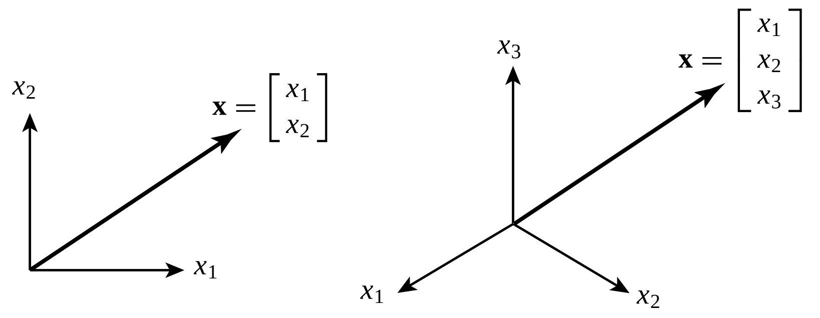

x_2
$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
x_1
x_3
$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$
x_1
x_2

Figure A.1. Standard visualization of 2-vectors and 3-vectors as position vectors in 2-space and 3-space, respectively.

EXAMPLE A.1

$$\mathbf{x} = \begin{bmatrix} 4 \\ -2 \\ 3 \\ 0 \end{bmatrix}$$

is real column vector of order 4, or, briefly, a 4-vector.

EXAMPLE A.2

$$\mathbf{q} = [\,1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1\,]$$

is a real row vector of order 6.

Occassionally we shall use the short-hand "component notation"

$$\mathbf{x} = [x_i] \tag{A.6}$$

for a generic vector. This comes handy when it is desirable to show the scheme of notation for the components.

§A.2.2 Visualization

To aid visualization a two-dimensional vector **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. 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. Drawing becomes a bit messier, however, although some knowledge of perspective and projective geometry helps.

The interpretation extends to all 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 $\mathbf{x}^T$:

$$\mathbf{x}^T = [\, x_1 \quad x_2 \quad \ldots \quad x_n \,]. \tag{A.7}$$

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

$$\mathbf{a} = \begin{bmatrix} 3 \\ -1 \\ 6 \end{bmatrix} \qquad \text{is} \qquad \mathbf{b} = \mathbf{a}^T = [\, 3 \quad -1 \quad 6 \,]$$

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 $i = 1, \ldots n$. 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

$$\mathbf{x} + \mathbf{y} \overset{\text{def}}{=} \begin{bmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{bmatrix}. \tag{A.8}$$

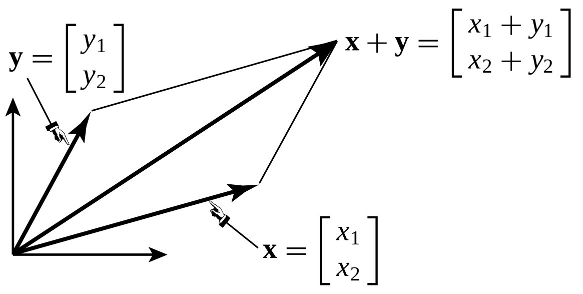

$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$
$\mathbf{x}+\mathbf{y} = \begin{bmatrix} x_1+y_1 \\ x_2+y_2 \end{bmatrix}$
$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

Figure A.2. In two and three-dimensional space, the vector addition operation is equivalent to the well known parallelogram composition law.

If **x** and **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 **x** and **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 paralellogram law; see Figure A.2. For $n = 1$ the usual scalar addition results.

For vector subtraction, replace $+$ by $-$ in the previous expressions.

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, Span

Multiplication of a vector **x** by a scalar c is defined by means of the relation

$$c\,\mathbf{x} \overset{\text{def}}{=} \begin{bmatrix} cx_1 \\ cx_2 \\ \vdots \\ cx_n \end{bmatrix} \tag{A.9}$$

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 paticular, 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

$$\frac{\mathbf{x}}{c} \equiv \mathbf{x}/c \overset{\text{def}}{=} (1/c)\mathbf{x} \tag{A.10}$$

The operation is not defined if $c = 0$.

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.

§A.3.5 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

$$(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y} = \mathbf{y}^T\mathbf{x} \overset{\text{def}}{=} \sum_{i=1}^{n} x_i y_i \overset{\text{sc}}{=} x_i y_i \tag{A.11}$$

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.11), 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.11) 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 dimension $n \geq 4$.

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:

$$\|\mathbf{x}\| \overset{\text{def}}{=} (\mathbf{x}, \mathbf{x}) = \sum_{i=1}^{n} x_i^2 \overset{\text{sc}}{=} x_i x_i \tag{A.12}$$

Because the norm is a sum of squares, it is zero only if **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:

$$|\mathbf{x}| \overset{\text{def}}{=} +\sqrt{\|\mathbf{x}\|}. \tag{A.13}$$

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 course.

Two important inequalities satisfied by vector norms (and not just the Euclidean norm) are the *Cauchy-Schwarz inequality*:

$$|(\mathbf{x}, \mathbf{y})| \leq |\mathbf{x}|\,|\mathbf{y}|, \tag{A.14}$$

and the *triangle inequality*:

$$|\mathbf{x} + \mathbf{y}| \leq |\mathbf{x}| + |\mathbf{y}|. \tag{A.15}$$

EXAMPLE A.6

The Euclidean norm of

$$\mathbf{x} = \begin{bmatrix} 4 \\ 3 \end{bmatrix}$$

is $\|x\| = 4^2 + 3^2 = 25$, and its length is $|x| = \sqrt{25} = 5$.

§A.3.6 Unit Vectors and Normalization

A vector of length one is called a *unit vector*. Any non-null vector **x** can be scaled to unit length by dividing all components by its original length:

$$\mathbf{x}/|\mathbf{x}| = \begin{bmatrix} x_1/|\mathbf{x}| \\ x_2/|\mathbf{x}| \\ \vdots \\ x_n/|\mathbf{x}| \end{bmatrix} \tag{A.16}$$

This particular scaling is called *normalization to unit length*. If **x** is the null vector, the normalization operation is undefined.

EXAMPLE A.7

The unit-vector normalization of

$$\mathbf{x} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \qquad \text{is} \qquad \mathbf{x}/|\mathbf{x}| = \mathbf{x}/5 = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}$$

§A.3.7 Angles, Orthonormality and Projections

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:

$$\cos\theta = (\mathbf{x}, \mathbf{y}) \tag{A.17}$$

If the vectors are not of unit length, they should be normalized to such, and so the general formula for two arbitrary vectors is

$$\cos\theta = (\frac{\mathbf{x}}{|\mathbf{x}|}, \frac{\mathbf{y}}{|\mathbf{y}|}) = \frac{(\mathbf{x}, \mathbf{y})}{|\mathbf{x}|\,|\mathbf{y}|} \tag{A.18}$$

The angle θ formed by a non-null vector with itself is zero. Note that this will always be the case for $n = 1$. If one of the vectors is null, the angle is undefined.

The definition (A.18) 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 *Span*($\mathbf{x}$) and *Span*($\mathbf{y}$) is the real number ϕ satisfying $0 \leq \phi \leq \pi/2$ and

$$\cos\phi = \frac{|(\mathbf{x}, \mathbf{y})|}{|\mathbf{x}|\,|\mathbf{y}|} \tag{A.19}$$

Two vectors $\mathbf{x}$ and $\mathbf{y}$ connected by the relation

$$(\mathbf{x}, \mathbf{y}) = 0 \tag{A.20}$$

are said to be *orthogonal*. The acute angle formed by two orthogonal vectors is $\pi/2$ radians or 90°.

The *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

$$(\mathbf{p}, \mathbf{y} - \mathbf{p}) = 0, \qquad \text{where} \quad \mathbf{p} = c\,\frac{\mathbf{x}}{|\mathbf{x}|} \tag{A.21}$$

and it is easily shown that

$$c = (\mathbf{y}, \frac{\mathbf{x}}{|\mathbf{x}|}) = |\mathbf{y}|\ \cos\theta \tag{A.22}$$

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.3.8 Orthogonal Bases, Subspaces

Let $\mathbf{b}^k$, $k = 1, \ldots m$ be a set of n-dimensional *unit* vectors which are *mutually orthogonal*. Then if a particular n-dimensional vector $\mathbf{x}$ admits the representation

$$\mathbf{x} = \sum_{k=1}^{m} c_k \mathbf{b}^k \tag{A.23}$$

the coefficients c_k are given by the inner products

$$c_k = (\mathbf{x}, \mathbf{b}^k) = \sum_{i=1}^{n} x_i b_i^k \overset{\text{sc}}{=} x_i b_i^k. \tag{A.24}$$

We also have *Parseval's equality*

$$(\mathbf{x}, \mathbf{x}) = \sum_{k=1}^{m} c_k^2 \overset{\text{sc}}{=} c_k c_k \tag{A.25}$$

If the representation (A.23) 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.24) forms a *subspace* of dimension m. The subspace is said to be *spanned* by the basis $\mathbf{b}^k$, and is called *Span*($\mathbf{b}^k$).[*] 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*

$$\mathbf{b}^1 = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \qquad \mathbf{b}^2 = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \qquad \cdots \qquad \mathbf{b}^n = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix} \tag{A.26}$$

In this case the coordinates of $\mathbf{x}$ are simply its components, that is, $c_k \equiv x_k$.

[*] Note that if $m = 1$ we have the span of a single vector, which as noted before is simply a line passing through the origin of coordinates.

Homework Exercises for Appendix A: Vectors

EXERCISE A.1

Given the four-dimensional vectors

$$\mathbf{x} = \begin{bmatrix} 2 \\ 4 \\ 4 \\ -8 \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} 1 \\ -5 \\ 7 \\ 5 \end{bmatrix} \tag{EA.1}$$

(a) compute the Euclidean norms and lengths of **x** and **y**;

(b) compute the inner product (**x**,**y**);

(c) verify that the inequalities (A.15) and (A.16) hold;

(d) normalize **x** and **y** to unit length;

(e) compute the angles $\theta = \angle(\mathbf{x}, \mathbf{y})$ and ϕ given by (A.19) and (A.20);

(f) compute the projection of **y** onto **x** and verify that the orthogonality condition (A.22) holds.

EXERCISE A.2

Given the base vectors

$$\mathbf{b}^1 = \tfrac{1}{10} \begin{bmatrix} 1 \\ -7 \\ 1 \\ 7 \end{bmatrix}, \qquad \mathbf{b}^2 = \tfrac{1}{10} \begin{bmatrix} 5 \\ 5 \\ -5 \\ 5 \end{bmatrix} \tag{EA.2}$$

(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:

$$\mathbf{z} = \begin{bmatrix} 8 \\ -16 \\ -2 \\ 26 \end{bmatrix} = c_1\mathbf{b}^1 + c_2\mathbf{b}^2 \tag{EA.3}$$

(c) Using the values computed in (b), verify that the coordinate-expansion formulas (A.24) and Parseval's equality (A.25) 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 **x** and the opposite vector $-\mathbf{x}$?

EXERCISE A.5

Show that

$$(\alpha\mathbf{x}, \beta\mathbf{y}) = \alpha\beta\,(\mathbf{x}, \mathbf{y}) \tag{EA.4}$$

where **x** and **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)

$$|(\mathbf{x}, \mathbf{y})| = 30 \le |\mathbf{x}|\,|\mathbf{y}| = 100$$
$$|\mathbf{x} + \mathbf{y}| = \sqrt{3^2 + (-1)^2 + (-11)^2 + (-3)^2} = \sqrt{140} \le |\mathbf{x}| + |\mathbf{y}| = 10 + 10 = 20$$

(d)

$$\mathbf{x}^u = \frac{\mathbf{x}}{10} = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ -0.8 \end{bmatrix}, \qquad \mathbf{y}^u = \frac{\mathbf{y}}{10} = \begin{bmatrix} 0.1 \\ -0.5 \\ 0.7 \\ 0.5 \end{bmatrix}$$

(e)

$$\cos\theta = (\mathbf{x}^u, \mathbf{y}^u) = -0.30, \qquad \theta = 107.46^\circ$$
$$\cos\phi = |(\mathbf{x}^u, \mathbf{y}^u)| = 0.30, \qquad \phi = 72.54^\circ$$

(f)

$$c = |\mathbf{y}| \cos\theta = -3$$
$$\mathbf{p} = c\,\mathbf{x}^u = -3 \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \\ -0.8 \end{bmatrix} = \begin{bmatrix} -0.6 \\ -1.2 \\ -1.2 \\ 2.4 \end{bmatrix}$$
$$\mathbf{y} - \mathbf{p} = \begin{bmatrix} 1.6 \\ -3.8 \\ 8.2 \\ 2.6 \end{bmatrix}$$
$$(\mathbf{p}, \mathbf{y} - \mathbf{p}) = -0.96 + 4.56 - 9.84 + 6.24 = 0$$

EXERCISE A.2

(a)

$$\|\mathbf{b}^1\| = 0.1^2 + (-0.7)^2 + 0.1^2 + 0.7^2 = 1, \qquad |\mathbf{b}^1| = 1$$
$$\|\mathbf{b}^2\| = 0.5^2 + 0.5^2 + (-0.5)^2 + 0.5^2 = 1, \qquad |\mathbf{b}^2| = 1$$
$$(\mathbf{b}^1, \mathbf{b}^2) = 0.05 - 0.35 - 0.05 + 0.35 = 0$$

(b)

$$c_1 = 30, \quad c_2 = 10$$

(by inspection, or solving a system of 2 linear equations)

(c)

$$c_1 = (\mathbf{z}, \mathbf{b}^1) = 0.8 + 11.2 - 0.2 + 18.2 = 30$$
$$c_2 = (\mathbf{z}, \mathbf{b}^2) = 4.0 - 8.0 + 1.0 + 13.0 = 10$$
$$c_1^2 + c_2^2 = 30^2 + 10^2 = 1000 = \|\mathbf{z}\| = 8^2 + (-16)^2 + (-2)^2 + 26^2 = 1000$$

EXERCISE A.3

See any linear algebra book, e.g. Strang's.

EXERCISE A.4

$$\theta = 180^\circ, \qquad \phi = 0^\circ$$

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

Matrix Algebra: Matrices

§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:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1j} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2j} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{i1} & a_{i2} & \dots & a_{ij} & \dots & a_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \dots & a_{mj} & \dots & a_{mn} \end{bmatrix}. \tag{B.1}$$

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. Occassionally we shall use the short-hand component notation

$$\mathbf{A} = [a_{ij}]. \tag{B.2}$$

EXAMPLE B.1

The following is a 2×3 numerical matrix:

$$\mathbf{B} = \begin{bmatrix} 2 & 6 & 3 \\ 4 & 9 & 1 \end{bmatrix} \tag{B.3}$$

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. For the example matrix (B.3) we would show

$$\underset{2\times 3}{\mathbf{B}} \tag{B.4}$$

REMARK B.1

Matrices should not be confused with determinants. A determinant is a number associated with square matrices ($m = n$), defined according to the rules stated in Appendix C.

§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

$$\mathbf{C} = \begin{bmatrix} 12 & 6 & 3 \\ 8 & 24 & 7 \\ 2 & 5 & 11 \end{bmatrix} \tag{B.5}$$

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:

$$\mathbf{S} = \begin{bmatrix} 11 & 6 & 1 \\ 6 & 3 & -1 \\ 1 & -1 & -6 \end{bmatrix}. \tag{B.6}$$

The following is an antisymmetric matrix of order 4:

$$\mathbf{W} = \begin{bmatrix} 0 & 3 & -1 & -5 \\ -3 & 0 & 7 & -2 \\ 1 & -7 & 0 & 0 \\ 5 & 2 & 0 & 0 \end{bmatrix}. \tag{B.7}$$

§B.1.5 Are Vectors a Special Case of Matrices?

Consider the 3-vector **x** and a 3 × 1 matrix **X** with the same components:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \qquad \mathbf{X} = \begin{bmatrix} x_{11} \\ x_{21} \\ x_{31} \end{bmatrix}. \tag{B.8}$$

in which $x_1 = x_{11}$, $x_2 = x_{22}$ and $x_3 = x_{33}$. Are **x** and **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 **0**, is the matrix all of whose components are zero.

EXAMPLE B.4

The null matrix of order 2 × 3 is

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{B.9}$$

The *identity matrix*, written **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

$$\mathbf{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{B.10}$$

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:

$$\mathbf{D} = \begin{bmatrix} 14 & 0 & 0 & 0 \\ 0 & -6 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 \end{bmatrix}. \tag{B.11}$$

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:

$$\mathbf{D} = \mathbf{diag}\,[\,14 \quad -6 \quad 0 \quad 3\,]. \tag{B.12}$$

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:

$$\mathbf{U} = \begin{bmatrix} 6 & 4 & 2 & 1 \\ 0 & 6 & 4 & 2 \\ 0 & 0 & 6 & 4 \\ 0 & 0 & 0 & 6 \end{bmatrix}, \qquad \mathbf{L} = \begin{bmatrix} 5 & 0 & 0 & 0 \\ 10 & 4 & 0 & 0 \\ -3 & 21 & 6 & 0 \\ -15 & -2 & 18 & 7 \end{bmatrix}. \tag{B.13}$$

§B.2 ELEMENTARY MATRIX OPERATIONS

§B.2.1 Equality

Two matrices **A** and **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 $i = 1, \ldots m,\ j = 1, \ldots n$. 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 **A** is another matrix denoted by $\mathbf{A}^T$ that has n rows and m columns

$$\mathbf{A}^T = [a_{ji}]. \tag{B.14}$$

The rows of $\mathbf{A}^T$ are the columns of **A**, and the rows of **A** are the columns of $\mathbf{A}^T$.

Obviously the transpose of $\mathbf{A}^T$ is again **A**, that is, $(\mathbf{A}^T)^T = \mathbf{A}$.

EXAMPLE B.8

$$\mathbf{A} = \begin{bmatrix} 5 & 7 & 0 \\ 1 & 0 & 4 \end{bmatrix}, \qquad \mathbf{A}^T = \begin{bmatrix} 5 & 1 \\ 7 & 0 \\ 0 & 4 \end{bmatrix}. \tag{B.15}$$

The transpose of a square matrix is also a square matrix. The transpose of a symmetric matrix **A** is equal to the original matrix, *i.e.*, $\mathbf{A} = \mathbf{A}^T$. The negated transpose of an antisymmetric matrix matrix **A** is equal to the original matrix, *i.e.* $\mathbf{A} = -\mathbf{A}^T$.

EXAMPLE B.9

$$\mathbf{A} = \begin{bmatrix} 4 & 7 & 0 \\ 7 & 1 & 2 \\ 0 & 2 & 3 \end{bmatrix} = \mathbf{A}^T, \qquad \mathbf{W} = \begin{bmatrix} 0 & 7 & 0 \\ -7 & 0 & -2 \\ 0 & 2 & 0 \end{bmatrix} = -\mathbf{W}^T \tag{B.16}$$

§B.2.3 Addition and Subtraction

The simplest operation acting on two matrices is *addition*. The sum of two matrices of the same order, **A** and **B**, is written $\mathbf{A} + \mathbf{B}$ and defined to be the matrix

$$\mathbf{A} + \mathbf{B} \stackrel{\text{def}}{=} [a_{ij} + b_{ij}]. \tag{B.17}$$

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 (B.17).

EXAMPLE B.10

The sum of

$$\mathbf{A} = \begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} 6 & 3 & -3 \\ 7 & -2 & 5 \end{bmatrix} \quad \text{is} \quad \mathbf{A}+\mathbf{B} = \begin{bmatrix} 7 & 0 & -3 \\ 11 & 0 & 4 \end{bmatrix}. \tag{B.18}$$

§B.2.4 Scalar Multiplication

Multiplication of a matrix **A** by a scalar c is defined by means of the relation

$$c\,\mathbf{A} \stackrel{\text{def}}{=} [c a_{ij}] \tag{B.19}$$

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

$$\text{If} \quad \mathbf{A} = \begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix}, \qquad 3\,\mathbf{A} = \begin{bmatrix} 3 & -9 & 0 \\ 12 & 6 & -3 \end{bmatrix}. \tag{B.20}$$

§B.3 MATRIX PRODUCTS

§B.3.1 Matrix by Vector

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

$$\mathbf{y} = \mathbf{A}\mathbf{x}, \tag{B.21}$$

to mean the linear transformation

$$y_i \overset{\text{def}}{=} \sum_{j=1}^{n} a_{ij} x_j \overset{\text{sc}}{=} a_{ij} x_j, \qquad i = 1, \ldots, m. \tag{B.22}$$

EXAMPLE B.12

The product of a 2×3 matrix and a vector of order 3 is a vector of order 2:

$$\begin{bmatrix} 1 & -3 & 0 \\ 4 & 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} -5 \\ 5 \end{bmatrix} \tag{B.23}$$

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 $\mathbf{xA}$ 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 (see §A.2), and the matrix-vector product reduces to the inner product defined by Equation (A.11). 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

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 $\mathbf{B}$:

$$\mathbf{C} = \mathbf{AB}. \tag{B.24}$$

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

$$c_{ik} \overset{\text{def}}{=} \sum_{j=1}^{n} a_{ij} b_{jk} \overset{\text{sc}}{=} a_{ij} b_{jk}, \quad i = 1, \ldots, m, \quad k = 1, \ldots, p. \tag{B.25}$$

We see that the $(i,k)^{th}$ entry of **C** is computed by taking the *inner product* of the i^{th} row of **A** with the k^{th} column of **B**. For this definition to work and the product be possible, *the column dimension of* **A** *must be the same as the row dimension of* **B**. Matrices that satisfy this rule are said to be *product-conforming*, or *conforming* for short. If two matrices do not conform, their product is undefined. The following mnemonic notation often helps in remembering this rule:

$$\underset{m\times p}{\mathbf{C}} = \underset{m\times n}{\mathbf{A}}\;\underset{n\times p}{\mathbf{B}} \tag{B.26}$$

For the matrix-by-vector case treated in the preceding subsection, $p = 1$.

Matrix **A** is called the pre-multiplicand and is said to *premultiply* **B**. Matrix **B** is called the post-multiplicand and is said to *postmultiply* **A**. This careful distinction on which matrix comes first is a consequence of the absence of commutativity: even if **BA** exists (it only does if $m = n$), it is not generally the same as **AB**.

For *hand* computations, the matrix product is most conveniently organized by the so-called Falk's scheme:

$$\begin{array}{cc} & \begin{bmatrix} b_{11} & \cdots & b_{ik} & \cdots & b_{1p} \\ \vdots & \ddots & \downarrow & \ddots & \vdots \\ b_{n1} & \cdots & b_{nk} & \cdots & b_{np} \end{bmatrix} \\ \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{i1} & \rightarrow & a_{in} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} & \begin{bmatrix} & & \vdots & & \\ & \cdots & c_{ik} & & \\ & & & & \end{bmatrix}. \end{array} \tag{B.27}$$

Each entry in row i of **A** is multiplied by the corresponding entry in column k of **B** (note the arrows), and the products are summed and stored in the $(i,k)^{th}$ entry of **C**.

EXAMPLE B.13

To illustrate Falk's scheme, let us form the product $\mathbf{C} = \mathbf{AB}$ of the following matrices

$$\mathbf{A} = \begin{bmatrix} 3 & 0 & 2 \\ 4 & -1 & 5 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 2 & 1 & 0 & -5 \\ 4 & 3 & -1 & 0 \\ 0 & 1 & -7 & 4 \end{bmatrix} \tag{B.28}$$

The matrices are conforming because the column dimension of **A** and the row dimension of **B** are the same (3). We arrange the computations as shown below:

$$\begin{array}{ccc} & \begin{bmatrix} 2 & 1 & 0 & -5 \\ 4 & 3 & -1 & 0 \\ 0 & 1 & -7 & 4 \end{bmatrix} & = \mathbf{B} \\ \mathbf{A} = \begin{bmatrix} 3 & 0 & 2 \\ 4 & -1 & 5 \end{bmatrix} & \begin{bmatrix} 6 & 5 & -14 & -7 \\ 4 & 6 & -34 & 0 \end{bmatrix} & = \mathbf{C} = \mathbf{AB} \end{array} \tag{B.29}$$

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 $\mathbf{AA}$ 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.

Similarly, $\mathbf{A}^3 = \mathbf{AAA} = \mathbf{A}^2\mathbf{A} = \mathbf{AA}^2$. 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 require the notion of eigensystem.

A square matrix $\mathbf{A}$ that satisfies $\mathbf{A} = \mathbf{A}^2$ is called *idempotent*. We shall see later that that equation 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 Properties of Matrix Products

Associativity. The associative law is verified:

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}. \tag{B.30}$$

Hence we may delete the parentheses and simply write $\mathbf{ABC}$.

Distributivity. The distributive law also holds: If $\mathbf{B}$ and $\mathbf{C}$ are matrices of the same order, then

$$\mathbf{A}\,(\mathbf{B}+\mathbf{C}) = \mathbf{AB}+\mathbf{AC}, \qquad \text{and} \qquad (\mathbf{B}+\mathbf{C})\,\mathbf{A} = \mathbf{BA}+\mathbf{CA}. \tag{B.31}$$

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 $\mathbf{AB}$ and $\mathbf{BA}$ are both possible but in general $\mathbf{AB} \neq \mathbf{BA}$.

If $\mathbf{AB} = \mathbf{BA}$, 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

$$\mathbf{A} = \begin{bmatrix} a & b \\ b & c \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} a-\beta & b \\ b & c-\beta \end{bmatrix}, \tag{B.32}$$

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:

$$(\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T. \tag{B.33}$$

The general transposition formula for an arbitrary product sequence is

$$(\mathbf{ABC}\ldots\mathbf{MN})^T = \mathbf{N}^T\mathbf{M}^T\ldots\mathbf{C}^T\mathbf{B}^T\mathbf{A}^T. \tag{B.34}$$

Congruential Transformation. If $\mathbf{B}$ is a *symmetric* matrix of order m and $\mathbf{A}$ is an arbitrary $m \times n$ matrix, then

$$\mathbf{S} = \mathbf{A}^T\mathbf{B}\mathbf{A}. \tag{B.35}$$

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 $\mathbf{B} \neq \mathbf{0}$. Similar, it is possible that $\mathbf{A} \neq \mathbf{0}$, $\mathbf{A}^2 \neq \mathbf{0}$, $\ldots$, 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:

$$s = \underset{1\times n}{\mathbf{y}^T}\ \underset{n\times n}{\mathbf{A}}\ \underset{n\times 1}{\mathbf{x}} \tag{B.36}$$

This is called a *bilinear form*.

Transposing both sides of (B.36) and noting that the transpose of a scalar does not change, we obtain the result

$$s = \mathbf{x}^T\mathbf{A}^T\mathbf{y}. \tag{B.37}$$

If $\mathbf{A}$ is symmetric and vectors $\mathbf{x}$ and $\mathbf{y}$ coalesce, *i.e.*

$$\mathbf{A}^T = \mathbf{A}, \qquad \mathbf{x} = \mathbf{y}, \tag{B.38}$$

the bilinear form becomes a *quadratic form*

$$s = \mathbf{x}^T\mathbf{A}\mathbf{x}. \tag{B.39}$$

Transposing both sides of a quadratic form reproduces the same equation.

EXAMPLE B.15

The kinetic energy of a system consisting of three point masses m_1, m_2, m_3 is

$$T = \tfrac{1}{2}(m_1 v_1^2 + m_2 v_2^2 + m_3 v_3^2). \tag{B.40}$$

This can be expressed as the quadratic form

$$T = \tfrac{1}{2}\mathbf{u}^T\mathbf{M}\mathbf{u} \tag{B.41}$$

where

$$\mathbf{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}, \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}. \tag{B.42}$$

Homework Exercises for Appendix B: Matrices

EXERCISE B.1

Given the three matrices

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 1 & 0 \\ -1 & 2 & 3 & 1 \\ 2 & 5 & -1 & 2 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 2 & -2 \\ 1 & 0 \\ 4 & 1 \\ -3 & 2 \end{bmatrix}, \quad \mathbf{C} = \begin{bmatrix} 1 & -3 & 2 \\ 2 & 0 & 2 \end{bmatrix} \tag{EB.1}$$

compute the product $\mathbf{D} = \mathbf{ABC}$ by hand using Falk's scheme. (*Hint:* do **BC** first, then premultiply that by **A**.)

EXERCISE B.2

Given the square matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 3 \\ -4 & 2 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 3 & 0 \\ 1 & -2 \end{bmatrix} \tag{EB.2}$$

verify by direct computation that $\mathbf{AB} \neq \mathbf{BA}$.

EXERCISE B.3

Given the matrices

$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ -1 & 2 \\ 2 & 0 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 3 & -1 & 4 \\ -1 & 2 & 0 \\ 4 & 0 & 0 \end{bmatrix} \tag{EB.3}$$

(note that **B** is symmetric) compute $\mathbf{S} = \mathbf{A}^T\mathbf{BA}$, and verify that **S** is symmetric.

EXERCISE B.4

Given the square matrices

$$\mathbf{A} = \begin{bmatrix} 3 & -1 & 2 \\ 1 & 0 & 3 \\ 3 & -2 & -5 \end{bmatrix}, \qquad \mathbf{B} = \begin{bmatrix} 3 & -6 & -3 \\ 7 & -14 & -7 \\ -1 & 2 & 1 \end{bmatrix} \tag{EB.4}$$

verify that $\mathbf{AB} = \mathbf{0}$ although $\mathbf{A} \neq \mathbf{0}$ and $\mathbf{B} \neq \mathbf{0}$. Is **BA** also null?

EXERCISE B.5

Given the square matrix

$$\mathbf{A} = \begin{bmatrix} 0 & a & b \\ 0 & 0 & c \\ 0 & 0 & 0 \end{bmatrix} \tag{EB.5}$$

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 (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 **C** and **D** agree.)

EXERCISE B.8

If **A** is an arbitrary $m \times n$ matrix, show: (a) both products $\mathbf{A}^T\mathbf{A}$ and $\mathbf{AA}^T$ are possible, and (b) both products are square and symmetric. (*Hint*: for (b) make use of the symmetry condition $\mathbf{S} = \mathbf{S}^T$ and of (B.31).)

EXERCISE B.9

Show that $\mathbf{A}^2$ only exists if and only if **A** is square.

EXERCISE B.10

If **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

$$\begin{array}{rr} & \begin{bmatrix} 1 & -3 & 2 \\ 2 & 0 & 2 \end{bmatrix} = \mathbf{C} \\ \mathbf{B} = \begin{bmatrix} 2 & -2 \\ 1 & 0 \\ 4 & 1 \\ -3 & 2 \end{bmatrix} & \begin{bmatrix} -2 & -6 & 0 \\ 1 & -3 & 2 \\ 6 & -12 & 10 \\ 1 & 9 & -2 \end{bmatrix} = \mathbf{BC} \\ \mathbf{A} = \begin{bmatrix} 2 & 4 & 1 & 0 \\ -1 & 2 & 3 & 1 \\ 2 & 5 & -1 & 2 \end{bmatrix} & \begin{bmatrix} 6 & -36 & 18 \\ 23 & -27 & 32 \\ -3 & 3 & -4 \end{bmatrix} = \mathbf{ABC} = \mathbf{D} \end{array}$$

EXERCISE B.2

$$\mathbf{AB} = \begin{bmatrix} 6 & -6 \\ -10 & -4 \end{bmatrix} \neq \mathbf{BA} = \begin{bmatrix} 3 & 9 \\ 9 & -1 \end{bmatrix}$$

EXERCISE B.3

$$\mathbf{S} = \mathbf{A}^T\mathbf{BA} = \begin{bmatrix} 23 & -6 \\ -6 & 8 \end{bmatrix}$$

which is symmetric, like **B**.

EXERCISE B.4

$$\begin{array}{rr} & \begin{bmatrix} 3 & -6 & -3 \\ 7 & -14 & -7 \\ -1 & 2 & 1 \end{bmatrix} = \mathbf{B} \\ \mathbf{A} = \begin{bmatrix} 3 & -1 & 2 \\ 1 & 0 & 3 \\ 3 & -2 & -5 \end{bmatrix} & \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{AB} = \mathbf{0} \end{array}$$

However,

$$\mathbf{BA} = \begin{bmatrix} -6 & 3 & 3 \\ -14 & 7 & 7 \\ 2 & -1 & -1 \end{bmatrix} \neq \mathbf{0}$$

EXERCISE B.5

$$\mathbf{A}^2 = \mathbf{AA} = \begin{bmatrix} 0 & 0 & ac \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \qquad \mathbf{A}^3 = \mathbf{AAA} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \mathbf{0}$$

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:

$$\underset{m\times n}{\mathbf{A}} = [a_{ij}], \qquad \underset{n\times m}{\mathbf{A}^T} = [a_{ji}]$$

$$\underset{n\times p}{\mathbf{B}} = [b_{jk}], \qquad \underset{p\times n}{\mathbf{B}^T} = [b_{kj}]$$

Indices i, j and k run over $1\ldots m$, $1\ldots n$ and $1\ldots p$, respectively. Now call

$$\underset{p\times m}{\mathbf{C}} = [c_{ki}] = (\mathbf{AB})^T$$

$$\underset{p\times m}{\mathbf{D}} = [d_{ki}] = \mathbf{B}^T\mathbf{A}^T$$

From the definition of matrix product,

$$c_{ki} = \sum_{j=1}^{n} a_{ij} b_{jk}$$

$$d_{ki} = \sum_{j=1}^{n} b_{jk} a_{ij} = \sum_{j=1}^{n} a_{ij} b_{jk} = c_{ki}$$

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:

$$\underset{n\times m}{\mathbf{A}^T}\ \underset{m\times n}{\mathbf{A}}, \qquad \underset{m\times n}{\mathbf{A}}\ \underset{n\times m}{\mathbf{A}^T}$$

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.

$$(\mathbf{A}^T\mathbf{A})^T = \mathbf{A}^T(\mathbf{A}^T)^T = \mathbf{A}^T\mathbf{A}$$

$$(\mathbf{A}\mathbf{A}^T)^T = (\mathbf{A}^T)^T\mathbf{A}^T = \mathbf{A}\mathbf{A}^T$$

Or, to do it more slowly, call $\mathbf{B} = \mathbf{A}^T$, $\mathbf{B}^T = \mathbf{A}$, $\mathbf{C} = \mathbf{AB}$, and let’s go over the first one again:

$$\mathbf{C}^T = (\mathbf{AB})^T = \mathbf{B}^T\mathbf{A}^T = \mathbf{A}\mathbf{A}^T = \mathbf{AB} = \mathbf{C}$$

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 $\mathbf{A}^2 = \mathbf{AA}$ 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):

$$\mathbf{A}^2 = \mathbf{AA} = -\mathbf{A}\mathbf{A}^T$$

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

Matrix Algebra: Determinants, Inverses, Eigenvalues

This Chapter discusses more specialized properties of matrices, such as determinants, eigenvalues and rank. These apply only to *square* matrices unless extension to rectangular matrices is explicitly stated.

§C.1 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:

$$|\mathbf{A}| = \pm \prod a_{1j_1}\, a_{2j_2}\, \ldots\, a_{nj_n}, \tag{C.1}$$

where the column indices $j_1,\, j_2, \ldots\, j_n$ are taken from the set $1, 2, \ldots, n$ with no repetitions allowed. The plus (minus) sign is taken if the permutation $(j_1\ j_2\ \ldots\ j_n)$ is even (odd).

EXAMPLE C.1

For a 2×2 matrix,

$$\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}. \tag{C.2}$$

EXAMPLE C.2

For a 3×3 matrix,

$$\begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{13}a_{22}a_{31} - a_{12}a_{21}a_{33} - a_{11}a_{23}a_{32}. \tag{C.3}$$

REMARK C.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 C.2

Inasmuch as the product (C.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 (C.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 C.3.

§C.1.1 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. That is

$$|\mathbf{A}| = |\mathbf{A}^T|. \tag{C.4}$$

II. If two rows (or columns) are interchanged the sign of the determinant is changed. For example:

$$\begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = - \begin{vmatrix} 1 & -2 \\ 3 & 4 \end{vmatrix}. \tag{C.5}$$

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:

$$\begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = \begin{vmatrix} 3+1 & 4-2 \\ 1 & -2 \end{vmatrix} = \begin{vmatrix} 4 & 2 \\ 1 & -2 \end{vmatrix} = -10. \tag{C.6}$$

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:

$$\begin{vmatrix} 6 & 8 \\ 1 & -2 \end{vmatrix} = 2\begin{vmatrix} 3 & 4 \\ 1 & -2 \end{vmatrix} = 2 \times (-10) = -20. \tag{C.7}$$

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

$$\begin{vmatrix} 3 & 2 & 1 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{vmatrix}. \tag{C.8}$$

This determinant is zero because the first column is a linear combination of the second and third columns:

$$\text{column 1} = \text{column 2} + \text{column 3} \tag{C.9}$$

Similarly there is a linear dependence between the rows which is given by the relation

$$\text{row 1} = \tfrac{7}{8}\,\text{row 2} + \tfrac{4}{5}\,\text{row 3} \tag{C.10}$$

VI. The determinant of an upper triangular or lower triangular matrix is the product of the main diagonal entries. For example,

$$\begin{vmatrix} 3 & 2 & 1 \\ 0 & 2 & -1 \\ 0 & 0 & 4 \end{vmatrix} = 3 \times 2 \times 4 = 24. \tag{C.11}$$

This rule is easily verified from the definition (C.1) because all terms vanish except $j_1 = 1$, $j_2 = 2, \ldots\ j_n = n$, 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:

$$|\mathbf{AB}| = |\mathbf{A}|\,|\mathbf{B}|. \tag{C.12}$$

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:

$$|\mathbf{A}^T| = |\mathbf{A}|. \tag{C.13}$$

This rule can be directly verified from the definition of determinant.

REMARK C.3

Rules VI and VII are the key to the practical evaluation of determinants. Any square nonsingular matrix **A** (where the qualifier "nonsingular" is explained in §C.3) can be decomposed as the product of two triangular factors

$$\mathbf{A} = \mathbf{L}\mathbf{U}, \tag{C.14}$$

where **L** is unit lower triangular and **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:

$$|\mathbf{A}| = |\mathbf{L}|\,|\mathbf{U}|. \tag{C.15}$$

But according to rule VI, $|\mathbf{L}| = 1$ and $|\mathbf{U}| = u_{11}u_{22}\ldots u_{nn}$. The last operation requires only $O(n)$ operations. Thus the evaluation of $|\mathbf{A}|$ is dominated by the effort involved in computing the factorization (C.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 (C.1), as noted in Remark C.1. Thus the LU-based method is roughly 30, 000 times faster for that modest matrix order, and the ratio increases exponentially for large n.

§C.1.2 Cramer's Rule

Cramer's rule provides a recipe for solving linear algebraic equations in terms of determinants. Let the simultaneous equations be as usual denoted as

$$\mathbf{A}\mathbf{x} = \mathbf{y}, \tag{C.16}$$

where **A** is a given $n \times n$ matrix, **y** is a given $n \times 1$ vector, and **x** is the $n \times 1$ vector of unknowns. The explicit form of (C.16) is Equation (A.1) of Appendix A, with $n = m$.

The explicit solution for the components $x_1, x_2 \ldots, x_n$ of **x** in terms of determinants is

$$x_1 = \frac{\begin{vmatrix} y_1 & a_{12} & a_{13} & \ldots & a_{1n} \\ y_2 & a_{22} & a_{23} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ y_n & a_{n2} & a_{n3} & \ldots & a_{nn} \end{vmatrix}}{|\mathbf{A}|}, \quad x_2 = \frac{\begin{vmatrix} a_{11} & y_1 & a_{13} & \ldots & a_{1n} \\ a_{21} & y_2 & a_{23} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots & \\ a_{n1} & y_n & a_{n3} & \ldots & a_{nn} \end{vmatrix}}{|\mathbf{A}|}, \ldots \tag{C.17}$$

The rule can be remembered as follows: in the numerator of the quotient for x_j, replace the j^{th} column of **A** by the right-hand side **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, this 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 C.3

Solve the 3×3 linear system

$$\begin{bmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 5 \\ 3 \end{bmatrix}, \tag{C.18}$$

by Cramer's rule:

$$x_1 = \frac{\begin{vmatrix} 8 & 2 & 1 \\ 5 & 2 & 0 \\ 3 & 0 & 2 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1, \quad x_2 = \frac{\begin{vmatrix} 5 & 8 & 1 \\ 3 & 5 & 0 \\ 1 & 3 & 2 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1, \quad x_3 = \frac{\begin{vmatrix} 5 & 2 & 8 \\ 3 & 2 & 5 \\ 1 & 0 & 3 \end{vmatrix}}{\begin{vmatrix} 5 & 2 & 1 \\ 3 & 2 & 0 \\ 1 & 0 & 2 \end{vmatrix}} = \frac{6}{6} = 1. \tag{C.19}$$

EXAMPLE C.4

Solve the 2×2 linear algebraic system

$$\begin{bmatrix} 2+\beta & -\beta \\ -\beta & 1+\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 0 \end{bmatrix} \tag{C.20}$$

by Cramer's rule:

$$x_1 = \frac{\begin{vmatrix} 5 & -\beta \\ 0 & 1+\beta \end{vmatrix}}{\begin{vmatrix} 2+\beta & -\beta \\ -\beta & 1+\beta \end{vmatrix}} = \frac{5+5\beta}{2+3\beta}, \quad x_2 = \frac{\begin{vmatrix} 2+\beta & 5 \\ -\beta & 0 \end{vmatrix}}{\begin{vmatrix} 2+\beta & -\beta \\ -\beta & 1+\beta \end{vmatrix}} = \frac{5\beta}{2+3\beta}. \tag{C.21}$$

§C.1.3 Homogeneous Systems

One immediate consequence of Cramer's rule is what happens if

$$y_1 = y_2 = \ldots = y_n = 0. \tag{C.22}$$

The linear equation systems with a null right hand side

$$\mathbf{A}\mathbf{x} = \mathbf{0}, \tag{C.23}$$

is called a *homogeneous system*. From the rule (C.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.

§C.2 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 C.5

The 3×3 matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 2 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{bmatrix}, \tag{C.24}$$

has rank $r = 3$ because $|\mathbf{A}| = -3 \neq 0$.

EXAMPLE C.6

The matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 2 & 1 \\ 1 & 2 & -1 \\ 2 & -1 & 3 \end{bmatrix}, \tag{C.25}$$

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 (C.9) and (C.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 (C.25) has rank $r = 2$.

§C.2.1 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 C.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 an undesirable property.

EXAMPLE C.8

An an illustration of the foregoing rule, consider the two-node, 4-DOF plane beam element stiffness derived in Chapter 13:

$$\mathbf{K} = \frac{EI}{L^3} \begin{bmatrix} 12 & 6L & -12 & 6L \\ & 4L^2 & -6L & 2L^2 \\ & & 12 & -6L \\ symm & & & 4L^2 \end{bmatrix} \tag{C.26}$$

where EI and L are nonzero scalars. It can 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.

§C.2.2 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,

$$Rank(\mathbf{AB}) \leq \min(r_a, r_b). \tag{C.27}$$

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,

$$Rank(\mathbf{A} + \mathbf{B}) \leq r_a + r_b. \tag{C.28}$$

§C.2.3 Singular Systems: Particular and Homegeneous Solutions

Having introduced the notion of rank we can now discuss what happens to the linear system (C.16) when the determinant of $\mathbf{A}$ vanishes, meaning that its rank is less than n. If so, the linear system (C.16) has either no solution or an infinite number of solution. 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 $\mathbf{z}_i$, $i = 1, \ldots d$ such that

$$\mathbf{A}\mathbf{z}_i = \mathbf{0}. \tag{C.29}$$

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 (C.13) is in the *range* of $\mathbf{A}$, that is, there exists an nonzero $\mathbf{x}_p$ such that $\mathbf{y} = \mathbf{A}\mathbf{x}_p$, its general solution is

$$\mathbf{x} = \mathbf{x}_p + \mathbf{x}_h = \mathbf{x}_p + \mathbf{Z}\mathbf{w}, \tag{C.30}$$

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* part, respectively, of the solution $\mathbf{x}$. If $\mathbf{y} = \mathbf{0}$ only the homogeneous part remains.

If $\mathbf{y}$ is not in the range of $\mathbf{A}$, system (C.13) 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.

§C.2.4 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 (C.27) and (C.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.

§C.3 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

$$\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}. \tag{C.31}$$

The most important application of the concept of inverse is the solution of linear systems. Suppose that, in the usual notation, we have

$$\mathbf{A}\mathbf{x} = \mathbf{y} \tag{C.32}$$

Premultiplying both sides by $\mathbf{A}^{-1}$ we get the inverse relationship

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \tag{C.33}$$

More generally, consider the matrix equation for multiple (m) right-hand sides:

$$\underset{n\times n}{\mathbf{A}}\ \underset{n\times m}{\mathbf{X}} = \underset{n\times m}{\mathbf{Y}}, \tag{C.34}$$

which reduces to (C.32) for $m = 1$. The inverse relation that gives $\mathbf{X}$ as function of $\mathbf{Y}$ is

$$\mathbf{X} = \mathbf{A}^{-1}\mathbf{Y}. \tag{C.35}$$

In particular, the solution of

$$\mathbf{A}\mathbf{X} = \mathbf{I}, \tag{C.36}$$

is $\mathbf{X} = \mathbf{A}^{-1}$. Practical methods for computing inverses are based on directly solving this equation; see Remark C.4.

§C.3.1 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 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 (C.36). Let $\mathbf{B} = [b_{ij}] = \mathbf{A}^{-1}$. Then

$$b_{ij} = \frac{A_{ji}}{|\mathbf{A}|}, \tag{C.37}$$

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: 2 or 3. In the examples below the inversion formulas for second and third order matrices are listed.

EXAMPLE C.9

For order $n = 2$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \qquad \mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|}\begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{22} \end{bmatrix}, \tag{C.38}$$

in which $|\mathbf{A}|$ is given by (C.2).

EXAMPLE C.10

For order $n = 3$:

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \qquad \mathbf{A}^{-1} = \frac{1}{|\mathbf{A}|} \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}, \tag{C.39}$$

where

$$\begin{aligned} b_{11} &= \begin{vmatrix} a_{22} & a_{23} \\ a_{32} & a_{33} \end{vmatrix}, & b_{21} &= -\begin{vmatrix} a_{12} & a_{13} \\ a_{32} & a_{33} \end{vmatrix}, & b_{31} &= \begin{vmatrix} a_{12} & a_{13} \\ a_{22} & a_{23} \end{vmatrix}, \\ b_{12} &= -\begin{vmatrix} a_{21} & a_{23} \\ a_{31} & a_{33} \end{vmatrix}, & b_{22} &= \begin{vmatrix} a_{11} & a_{13} \\ a_{31} & a_{33} \end{vmatrix}, & b_{32} &= -\begin{vmatrix} a_{11} & a_{13} \\ a_{21} & a_{23} \end{vmatrix}, \\ b_{13} &= \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}, & b_{23} &= -\begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}, & b_{33} &= \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \end{aligned} \tag{C.40}$$

in which $|\mathbf{A}|$ is given by (C.3).

EXAMPLE C.11

$$\mathbf{A} = \begin{bmatrix} 2 & 4 & 2 \\ 3 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}, \qquad \mathbf{A}^{-1} = -\frac{1}{8} \begin{bmatrix} 1 & -4 & 2 \\ -2 & 0 & 4 \\ -1 & 4 & -10 \end{bmatrix}. \tag{C.41}$$

If the order exceeds 3, the general inversion formula based on Cramer's rule becomes rapidly useless as it displays combinatorial complexity. For numerical work it is preferable to solve the system (C.38) after **A** is factored. Those techniques are described in detail in linear algebra books; see also Remark C.4.

§C.3.2 Some Properties of the Inverse

I. The inverse of the transpose is equal to the transpose of the inverse:

$$(\mathbf{A}^T)^{-1} = (\mathbf{A}^{-1})^T, \tag{C.42}$$

because

$$(\mathbf{A}\mathbf{A}^{-1}) = (\mathbf{A}\mathbf{A}^{-1})^T = (\mathbf{A}^{-1})^T\,\mathbf{A}^T = \mathbf{I}. \tag{C.43}$$

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:

$$(\mathbf{A}\mathbf{B})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \tag{C.44}$$

This is easily verified by substituting both sides of (C.39) into (C.31). This property generalizes to an arbitrary number of factors.

IV. For a diagonal matrix **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 **S** is a block diagonal matrix:

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_{11} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{22} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{S}_{nn} \end{bmatrix} = \mathrm{diag}\,[\,\mathbf{S}_{ii}\,], \tag{C.45}$$

then the inverse matrix is also block diagonal and is given by

$$\mathbf{S}^{-1} = \begin{bmatrix} \mathbf{S}_{11}^{-1} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{22}^{-1} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{S}_{33}^{-1} & \ldots & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ldots & \mathbf{S}_{nn}^{-1} \end{bmatrix} = \mathrm{diag}\,[\,\mathbf{S}_{ii}^{-1}\,]. \tag{C.46}$$

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 C.4

The practical numerical calculation of inverses is based on triangular factorization. Given a nonsingular $n \times n$ matrix **A**, calculate its LU factorization $\mathbf{A} = \mathbf{LU}$, which can be obtained in $O(n^3)$ operations. Then solve the linear triangular systems:

$$\mathbf{UY} = \mathbf{I}, \qquad \mathbf{LX} = \mathbf{Y}, \tag{C.47}$$

and the computed inverse $\mathbf{A}^{-1}$ appears in **X**. One can overwrite **I** with **Y** and **Y** with **X**. The whole process can be completed in $O(n^3)$ floating-point operations. For symmetric matrices the alternative decomposition $\mathbf{A} = \mathbf{LDL}^T$, where **L** is unit lower triangular and **D** is diagonal, is generally preferred to save computing time and storage.

§C.4 EIGENVALUES AND EIGENVECTORS

Consider the special form of the linear system (C.13) in which the right-hand side vector **y** is a multiple of the solution vector **x**:

$$\mathbf{Ax} = \lambda\mathbf{x}, \tag{C.48}$$

or, written in full,

$$\begin{array}{ccccccccc} a_{11}x_1 & + & a_{12}x_2 & + & \cdots & + & a_{1n}x_n & = & \lambda x_1 \\ a_{21}x_2 & + & a_{22}x_2 & + & \cdots & + & a_{2n}x_n & = & \lambda x_2 \\ \cdots & & \cdots & & \cdots & & \cdots & & \cdots \\ a_{n1}x_1 & + & a_{n2}x_2 & + & \cdots & + & a_{nn}x_n & = & \lambda x_n \end{array} \tag{C.49}$$

This is called the standard (or classical) *algebraic eigenproblem*. System (C.48) can be rearranged into the homogeneous form

$$(\mathbf{A} - \lambda\mathbf{I})\,\mathbf{x} = \mathbf{0}. \tag{C.50}$$

A nontrivial solution of this equation 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

$$|\mathbf{A} - \lambda\mathbf{I}| = \begin{vmatrix} a_{11}-\lambda & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22}-\lambda & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn}-\lambda \end{vmatrix} = 0. \tag{C.51}$$

When this determinant is expanded, we obtain an algebraic polynomial equation in λ of degree n:

$$P(\lambda) = \lambda^n + \alpha_1 \lambda^{n-1} + \dots + \alpha_n = 0. \tag{C.52}$$

This is known as the *characteristic equation* of the matrix $\mathbf{A}$. The left-hand side is called the *characteristic polynomial*. We known that a polynomial of degree n has n (generally complex) roots λ_1, λ_2, $\dots$, λ_n. These n numbers are called the *eigenvalues*, *eigenroots* or *characteristic values* of matrix $\mathbf{A}$.

With each eigenvalue λ_i there is an associated vector $\mathbf{x}_i$ that satisfies

$$\mathbf{A}\mathbf{x}_i = \lambda_i \mathbf{x}_i. \tag{C.53}$$

This $\mathbf{x}_i$ is called an *eigenvector* or *characteristic vector*. An eigenvector is unique only up to a scale factor since if $\mathbf{x}_i$ is an eigenvector, so is $\beta\mathbf{x}_i$ where β is an arbitrary nonzero number. Eigenvectors are often *normalized* so that their Euclidean length is 1, or their largest component is unity.

§C.4.1 Real Symmetric Matrices

Real symmetric matrices are of special importance in the finite element method. In linear algebra books dealing with the algebraic eigenproblem it is shown that:

(a) The n eigenvalues of a real symmetric matrix of order n are real.

(b) The eigenvectors corresponding to distinct eigenvalues are orthogonal. The eigenvectors corresponding to multiple roots may be orthogonalized with respect to each other.

(c) The n eigenvectors form a complete orthonormal basis for the Euclidean space E_n.

§C.4.2 Positivity

Let $\mathbf{A}$ be an $n \times n$ square *symmetric* matrix. $\mathbf{A}$ is said to be *positive definite* (p.d.) if

$$\mathbf{x}^T\mathbf{A}\mathbf{x} > 0, \qquad \mathbf{x} \neq 0 \tag{C.54}$$

A positive definite matrix has rank n. This property can be checked by computing the n eigenvalues λ_i of $\mathbf{A}\mathbf{z} = \lambda\mathbf{z}$. If all $\lambda_i > 0$, $\mathbf{A}$ is p.d.

$\mathbf{A}$ is said to be *nonnegative*[1] if zero equality is allowed in (C.54):

$$\mathbf{x}^T\mathbf{A}\mathbf{x} \leq 0, \qquad \mathbf{x} \neq 0 \tag{C.55}$$

A p.d. matrix is also nonnegative but the converse is not necessarily true. This property can be checked by computing the n eigenvalues λ_i of $\mathbf{A}\mathbf{z} = \lambda\mathbf{z}$. If r eigenvalues $\lambda_i > 0$ and $n - r$ eigenvalues are zero, $\mathbf{A}$ is nonnegative with rank r.

A symmetric square matrix $\mathbf{A}$ that has at least one negative eigenvalue is called *indefinite*.

[1] A property called *positive semi-definite* by some authors.

§C.4.3 Normal and Orthogonal Matrices

Let $\mathbf{A}$ be an $n \times n$ real square matrix. This matrix is called *normal* if

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T \tag{C.56}$$

A normal matrix is called *orthogonal* if

$$\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = \mathbf{I} \qquad \text{or} \quad \mathbf{A}^T = \mathbf{A}^{-1} \tag{C.57}$$

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.

§C.5 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 expressable as a low-rank correction to the original model.

§C.5.1 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

$$\left(\mathbf{A} + \beta\mathbf{u}\mathbf{v}^T\right)^{-1} = \mathbf{A}^{-1} - \frac{\beta}{\sigma}\,\mathbf{A}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{A}^{-1}. \tag{C.58}$$

This is called the Sherman-Morrison formula[2] when $\beta = 1$. Since any rank-one correction to $\mathbf{A}$ can be written as $\beta\mathbf{u}\mathbf{v}^T$, (C.58) gives the rank-one change to its inverse. The proof is by direct multiplication as in Exercise C.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 $-(\beta/\sigma)\mathbf{a}\mathbf{b}^T$.

§C.5.2 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 $\boldsymbol{\Sigma} = \mathbf{I}_k + \beta\mathbf{V}^T\mathbf{A}^{-1}\mathbf{U}$, in which $\mathbf{I}_k$ denotes the $k \times k$ identity matrix, is invertible. Then

$$\left(\mathbf{A} + \beta\mathbf{U}\mathbf{V}^T\right)^{-1} = \mathbf{A}^{-1} - \beta\,\mathbf{A}^{-1}\mathbf{U}\boldsymbol{\Sigma}^{-1}\mathbf{V}^T\mathbf{A}^{-1}. \tag{C.59}$$

This is called the Woodbury formula.[3] It reduces to (C.58) if $k = 1$, in which case $\boldsymbol{\Sigma} \equiv \sigma$ is a scalar. The proof is by direct multiplication.

[2] J. Sherman and W. J. Morrison, 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. Statist.*, **20**, (1949), 621. For a history of this remarkable formula and its extensions, which are quite important in many applications such as statistics, see the paper by Henderson and Searle in *SIAM Review*, **23**, 53–60.

[3] M.A.Woodbury, Inverting modified matrices, *Statist. Res. Group, Mem. Rep.* No. 42, Princeton Univ., Princeton, N.J., 1950

§C.5.3 Formulas for Modified Determinants

Let $\widetilde{\mathbf{A}}$ denote the adjoint of $\mathbf{A}$. Taking the determinants from both sides of $\mathbf{A} + \beta \mathbf{u}\mathbf{v}^T$ one obtains

$$|\mathbf{A} + \beta \mathbf{u}\mathbf{v}^T| = |\mathbf{A}| + \beta\, \mathbf{v}^T\, \widetilde{\mathbf{A}}\, \mathbf{u}. \tag{C.60}$$

If $\mathbf{A}$ is invertible, replacing $\widetilde{\mathbf{A}} = |\mathbf{A}|\, \mathbf{A}^{-1}$ this becomes

$$|\mathbf{A} + \beta \mathbf{u}\mathbf{v}^T| = |\mathbf{A}|\, (1 + \beta\, \mathbf{v}^T \mathbf{A}^{-1}\, \mathbf{u}). \tag{C.61}$$

Similarly, one can show that if $\mathbf{A}$ is invertible, and $\mathbf{U}$ and $\mathbf{V}$ are $n \times k$ matrices,

$$|\mathbf{A} + \beta \mathbf{U}\mathbf{V}^T| = |\mathbf{A}|\ |\mathbf{I}_k + \beta\, \mathbf{V}^T \mathbf{A}^{-1} \mathbf{U}|. \tag{C.62}$$

Exercises for Appendix C: Determinants, Inverses, Eigenvalues

EXERCISE C.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 C.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

$$\det(\mathbf{I} - \mathbf{u}\mathbf{v}^T) = 1 - \mathbf{v}^T\mathbf{u} \tag{EC.1}$$

EXERCISE C.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

$$\mathbf{H} = \mathbf{I} - 2\mathbf{u}\mathbf{u}^T \tag{EC.2}$$

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 C.4

The *trace* of a $n \times n$ square matrix $\mathbf{A}$, denoted $\textbf{trace}(\mathbf{A})$ is the sum $\sum_{i=1}^n a_{ii}$ of its diagonal coefficients. Show that if the entries of $\mathbf{A}$ are real,

$$\textbf{trace}(\mathbf{A}^T\mathbf{A}) = \sum_{i=1}^n \sum_{j=1}^n a_{ij}^2 \tag{EC.3}$$

EXERCISE C.5

Prove the Sherman-Morrison formula (C.59) by direct matrix multiplication.

EXERCISE C.6

Prove the Sherman-Morrison formula (C.59) for $\beta = 1$ by considering the following block bordered system

$$\begin{bmatrix} \mathbf{A} & \mathbf{U} \\ \mathbf{V}^T & \mathbf{I}_k \end{bmatrix} \begin{bmatrix} \mathbf{B} \\ \mathbf{C} \end{bmatrix} = \begin{bmatrix} \mathbf{I}_n \\ \mathbf{0} \end{bmatrix} \tag{EC.4}$$

in which $\mathbf{I}_k$ and $\mathbf{I}_n$ denote the identy matrices of orders k and n, respectively. Solve (C.62) 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 C.7

Show that the eigenvalues of a real symmetric square matrix are real, and that the eigenvectors are real vectors.

EXERCISE C.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 C.9

Show that if $\mathbf{A}$ is p.d., $\mathbf{A}\mathbf{x} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$.

EXERCISE C.10

Show that for any real $m \times n$ matrix $\mathbf{A}$, $\mathbf{A}^T\mathbf{A}$ exists and is nonnegative.

EXERCISE C.11

Show that a triangular matrix is normal if and only if it is diagonal.

EXERCISE C.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 C.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 C.14

(Tough) Let $\mathbf{A}$ be $m \times n$ and $\mathbf{B}$ be $n \times m$. Show that the nonzero eigenvalues of $\mathbf{AB}$ are the same as those of $\mathbf{BA}$ (Kahan).

EXERCISE C.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 C.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 C.17

Let $\mathbf{P}$ be a real square matrix that satisfies

$$\mathbf{P}^2 = \mathbf{P}. \tag{EC.5}$$

Such matrices are called *idempotent*, and also *orthogonal projectors*. Show that all eigenvalues of $\mathbf{P}$ are either zero or one.

EXERCISE C.18

The necessary and sufficient condition for two square matrices to commute is that they have the same eigenvectors.

EXERCISE C.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.

D
Matrix Calculus

In this Appendix we collect some useful formulas of matrix calculus that often appear in finite element derivations.

§D.1 THE DERIVATIVES OF VECTOR FUNCTIONS

Let **x** and **y** be vectors of orders n and m respectively:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \qquad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, \tag{D.1}$$

where each component y_i may be a function of all the x_j, a fact represented by saying that **y** is a function of **x**, or

$$\mathbf{y} = \mathbf{y}(\mathbf{x}). \tag{D.2}$$

If $n = 1$, **x** reduces to a scalar, which we call x. If $m = 1$, **y** reduces to a scalar, which we call y. Various applications are studied in the following subsections.

§D.1.1 Derivative of Vector with Respect to Vector

The derivative of the vector **y** with respect to vector **x** is the $n \times m$ matrix

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \overset{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} & \cdots & \frac{\partial y_m}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \frac{\partial y_2}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \tag{D.3}$$

§D.1.2 Derivative of a Scalar with Respect to Vector

If y is a scalar,

$$\frac{\partial y}{\partial \mathbf{x}} \overset{\text{def}}{=} \begin{bmatrix} \frac{\partial y}{\partial x_1} \\ \frac{\partial y}{\partial x_2} \\ \vdots \\ \frac{\partial y}{\partial x_n} \end{bmatrix}. \tag{D.4}$$

§D.1.3 Derivative of Vector with Respect to Scalar

If x is a scalar,

$$\frac{\partial \mathbf{y}}{\partial x} \overset{\text{def}}{=} \begin{bmatrix} \frac{\partial y_1}{\partial x} & \frac{\partial y_2}{\partial x} & \cdots & \frac{\partial y_m}{\partial x} \end{bmatrix} \tag{D.5}$$

REMARK D.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 D.1

Given

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{D.6}$$

and

$$\begin{aligned} y_1 &= x_1^2 - x_2 \\ y_2 &= x_3^2 + 3x_2 \end{aligned} \tag{D.7}$$

the partial derivative matrix $\partial\mathbf{y}/\partial\mathbf{x}$ is computed as follows:

$$\frac{\partial \mathbf{y}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_2}{\partial x_1} \\ \frac{\partial y_1}{\partial x_2} & \frac{\partial y_2}{\partial x_2} \\ \frac{\partial y_1}{\partial x_3} & \frac{\partial y_2}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 2x_1 & 0 \\ -1 & 3 \\ 0 & 2x_3 \end{bmatrix} \tag{D.8}$$

§D.1.4 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

$$J = \left| \frac{\partial \mathbf{x}}{\partial \mathbf{y}} \right| \tag{D.9}$$

is called the *Jacobian* of the transformation determined by $\mathbf{y} = \mathbf{y}(\mathbf{x})$. The inverse determinant is

$$J^{-1} = \left| \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \right|. \tag{D.10}$$

[1] One authors puts is 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 D.2

The transformation from spherical to Cartesian coordinates is defined by

$$x = r \sin\theta \cos\psi, \qquad y = r \sin\theta \sin\psi, \qquad z = r\cos\theta \tag{D.11}$$

where $r > 0$, $0 < \theta < \pi$ and $0 \le \psi < 2\pi$. To obtain the Jacobian of the transformation, let

$$\begin{matrix} x \equiv x_1, & y \equiv x_2, & z \equiv x_3 \\ r \equiv y_1, & \theta \equiv y_2, & \psi \equiv y_3 \end{matrix} \tag{D.12}$$

Then

$$\begin{aligned} J = \left|\frac{\partial \mathbf{x}}{\partial \mathbf{y}}\right| &= \begin{vmatrix} \sin y_2 \cos y_3 & \sin y_2 \sin y_3 & \cos y_2 \\ y_1 \cos y_2 \cos y_3 & y_1 \cos y_2 \sin y_3 & -y_1 \sin y_2 \\ -y_1 \sin y_2 \sin y_3 & y_1 \sin y_2 \cos y_3 & 0 \end{vmatrix} \\ &= y_1^2 \sin y_2 = r^2 \sin\theta. \end{aligned} \tag{D.13}$$

The foregoing definitions can be used to obtain derivatives to many frequently used expressions, including quadratic and bilinear forms.

EXAMPLE D.3

Consider the quadratic form

$$y = \mathbf{x}^T \mathbf{A} \mathbf{x} \tag{D.14}$$

where $\mathbf{A}$ is a square matrix of order n. Using the definition (D.3) one obtains

$$\frac{\partial y}{\partial \mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{A}^T \mathbf{x} \tag{D.15}$$

and if $\mathbf{A}$ is symmetric,

$$\frac{\partial y}{\partial \mathbf{x}} = 2\mathbf{A}\mathbf{x}. \tag{D.16}$$

We can of course continue the differentiation process:

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = \frac{\partial}{\partial \mathbf{x}}\left(\frac{\partial y}{\partial \mathbf{x}}\right) = \mathbf{A} + \mathbf{A}^T, \tag{D.17}$$

and if $\mathbf{A}$ is symmetric,

$$\frac{\partial^2 y}{\partial \mathbf{x}^2} = 2\mathbf{A}. \tag{D.18}$$

The following table collects several useful vector derivative formulas.

$\mathbf{y}$	$\dfrac{\partial \mathbf{y}}{\partial \mathbf{x}}$
$\mathbf{A}\mathbf{x}$	$\mathbf{A}^T$
$\mathbf{x}^T\mathbf{A}$	$\mathbf{A}$
$\mathbf{x}^T\mathbf{x}$	$2\mathbf{x}$
$\mathbf{x}^T\mathbf{A}\mathbf{x}$	$\mathbf{A}\mathbf{x} + \mathbf{A}^T\mathbf{x}$

§D.2 THE CHAIN RULE FOR VECTOR FUNCTIONS

Let

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \quad , \quad \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_r \end{bmatrix} \quad \text{and} \quad \mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \tag{D.19}$$

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

$$\left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^T = \begin{bmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_1}{\partial x_2} & \dots & \frac{\partial z_1}{\partial x_n} \\ \frac{\partial z_2}{\partial x_1} & \frac{\partial z_2}{\partial x_2} & \dots & \frac{\partial z_2}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial z_m}{\partial x_1} & \frac{\partial z_m}{\partial x_2} & \dots & \frac{\partial z_m}{\partial x_n} \end{bmatrix} \tag{D.20}$$

Each entry of this matrix may be expanded as

$$\frac{\partial z_i}{\partial x_j} = \sum_{q=1}^{r} \frac{\partial z_i}{\partial y_q} \frac{\partial y_q}{\partial x_j} \qquad \begin{cases} i = 1, 2, \ldots, m \\ j = 1, 2, \ldots, n. \end{cases} \tag{D.21}$$

Then

$$\begin{aligned} \left(\frac{\partial \mathbf{z}}{\partial \mathbf{x}}\right)^T &= \begin{bmatrix} \sum \frac{\partial z_1}{y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_1}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \dots & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \dots & \sum \frac{\partial z_2}{\partial y_q} \frac{\partial y_q}{\partial x_n} \\ \vdots & & & \\ \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_1} & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_2} & \dots & \sum \frac{\partial z_m}{\partial y_q} \frac{\partial y_q}{\partial x_n} \end{bmatrix} \\ &= \begin{bmatrix} \frac{\partial z_1}{\partial y_1} & \frac{\partial z_1}{\partial y_2} & \dots & \frac{\partial z_1}{\partial y_r} \\ \frac{\partial z_2}{\partial y_1} & \frac{\partial z_2}{\partial y_2} & \dots & \frac{\partial z_2}{\partial y_r} \\ \vdots & & & \\ \frac{\partial z_m}{\partial y_1} & \frac{\partial z_m}{\partial y_2} & \dots & \frac{\partial z_m}{\partial y_r} \end{bmatrix} \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \frac{\partial y_1}{\partial x_2} & \dots & \frac{\partial y_1}{\partial x_n} \\ \frac{\partial y_2}{\partial x_1} & \frac{\partial y_2}{\partial x_2} & \dots & \frac{\partial y_2}{\partial x_n} \\ \vdots & & & \\ \frac{\partial y_r}{\partial x_1} & \frac{\partial y_r}{\partial x_2} & \dots & \frac{\partial y_r}{\partial x_n} \end{bmatrix} \\ &= \left(\frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^T \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}}\right)^T = \left(\frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}}\right)^T. \end{aligned} \tag{D.22}$$

On transposing both sides, we finally obtain

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}}, \tag{D.23}$$

which is the *chain rule* for vectors. If all vectors reduce to scalars,

$$\frac{\partial z}{\partial x} = \frac{\partial y}{\partial x} \frac{\partial z}{\partial y} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}, \tag{D.24}$$

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 **w** is a function of **z**, which is a function of **y**, which is a function of **x**,

$$\frac{\partial \mathbf{w}}{\partial \mathbf{x}} = \frac{\partial \mathbf{y}}{\partial \mathbf{x}} \frac{\partial \mathbf{z}}{\partial \mathbf{y}} \frac{\partial \mathbf{w}}{\partial \mathbf{z}}. \tag{D.25}$$

On the other hand, in the ordinary chain rule one can indistictly build the product to the right or to the left because scalar multiplication is commutative.

§D.3 THE DERIVATIVE OF SCALAR FUNCTIONS OF A MATRIX

Let $\mathbf{X} = (x_{ij})$ be a matrix of order $(m \times n)$ and let

$$y = f(\mathbf{X}), \tag{D.26}$$

be a scalar function of **X**. The derivative of y with respect to **X**, denoted by

$$\frac{\partial y}{\partial \mathbf{X}}, \tag{D.27}$$

is defined as the following matrix of order $(m \times n)$:

$$\mathbf{G} = \frac{\partial y}{\partial \mathbf{X}} = \begin{bmatrix} \frac{\partial y}{\partial x_{11}} & \frac{\partial y}{\partial x_{12}} & \cdots & \frac{\partial y}{\partial x_{1n}} \\ \frac{\partial y}{\partial x_{21}} & \frac{\partial y}{\partial x_{22}} & \cdots & \frac{\partial y}{\partial x_{2n}} \\ \vdots & \vdots & & \vdots \\ \frac{\partial y}{\partial x_{m1}} & \frac{\partial y}{\partial x_{m2}} & \cdots & \frac{\partial y}{\partial x_{mn}} \end{bmatrix} = \left[\frac{\partial y}{\partial x_{ij}} \right] = \sum_{i,j} \mathbf{E}_{ij} \frac{\partial y}{\partial x_{ij}}, \tag{D.28}$$

where $\mathbf{E}_{ij}$ denotes the elementary matrix* of order $(m \times n)$. This matrix **G** is also known as a *gradient matrix*.

EXAMPLE D.4

Find the gradient matrix if y is the trace of a square matrix **X** of order n, that is

$$y = \operatorname{tr}(\mathbf{X}) = \sum_{i=1}^{n} x_{ii}. \tag{D.29}$$

Obviously all non-diagonal partials vanish whereas the diagonal partials equal one, thus

$$\mathbf{G} = \frac{\partial y}{\partial \mathbf{X}} = \mathbf{I}, \tag{D.30}$$

where **I** denotes the identity matrix of order n.

* The elementary matrix $\mathbf{E}_{ij}$ of order $m \times n$ has all zero entries except for the (i, j) entry, which is one.

§D.3.1 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 $y = |\mathbf{AX}|$. 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

$$\frac{\partial |\mathbf{Y}|}{\partial \mathbf{X}}. \tag{D.31}$$

Using the chain rule we can write

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \sum_i \sum_j \mathbf{Y}_{ij} \frac{\partial |\mathbf{Y}|}{\partial y_{ij}} \frac{\partial y_{ij}}{\partial x_{rs}}. \tag{D.32}$$

But

$$|\mathbf{Y}| = \sum_j y_{ij} \mathbf{Y}_{ij}, \tag{D.33}$$

where $\mathbf{Y}_{ij}$ is the *cofactor* of the element y_{ij} in $|\mathbf{Y}|$. Since the cofactors $\mathbf{Y}_{i1}, \mathbf{Y}_{i2}, \ldots$ are independent of the element y_{ij}, we have

$$\frac{\partial |\mathbf{Y}|}{\partial y_{ij}} = \mathbf{Y}_{ij}. \tag{D.34}$$

It follows that

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \sum_i \sum_j \mathbf{Y}_{ij} \frac{\partial y_{ij}}{\partial x_{rs}}. \tag{D.35}$$

There is an alternative form of this result which is ocassionally useful. Define

$$a_{ij} = \mathbf{Y}_{ij}, \quad \mathbf{A} = [a_{ij}], \qquad b_{ij} = \frac{\partial y_{ij}}{\partial x_{rs}}, \quad \mathbf{B} = [b_{ij}]. \tag{D.36}$$

Then it can be shown that

$$\frac{\partial |\mathbf{Y}|}{\partial x_{rs}} = \mathrm{tr}(\mathbf{AB}^T) = \mathrm{tr}(\mathbf{B}^T\mathbf{A}). \tag{D.37}$$

EXAMPLE D.5

If $\mathbf{X}$ is a nonsingular square matrix and $\mathbf{Z} = |\mathbf{X}|\mathbf{X}^{-1}$ its cofactor matrix,

$$\mathbf{G} = \frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = \mathbf{Z}^T. \tag{D.38}$$

If $\mathbf{X}$ is also symmetric,

$$\mathbf{G} = \frac{\partial |\mathbf{X}|}{\partial \mathbf{X}} = 2\mathbf{Z}^T - \mathrm{diag}(\mathbf{Z}^T). \tag{D.39}$$

§D.4 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

$$df = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}\, dx_i. \tag{D.40}$$

In harmony with this formula, we define the differential of an $m \times n$ matrix $\mathbf{X} = [x_{ij}]$ to be

$$d\mathbf{X} \overset{\text{def}}{=} \begin{bmatrix} dx_{11} & dx_{12} & \dots & dx_{1n} \\ dx_{21} & dx_{22} & \dots & dx_{2n} \\ \vdots & \vdots & & \vdots \\ dx_{m1} & dx_{m2} & \dots & dx_{mn} \end{bmatrix}. \tag{D.41}$$

This definition complies with the multiplicative and associative rules

$$d(\alpha\mathbf{X}) = \alpha\, d\mathbf{X}, \qquad d(\mathbf{X}+\mathbf{Y}) = d\mathbf{X} + d\mathbf{Y}. \tag{D.42}$$

If $\mathbf{X}$ and $\mathbf{Y}$ are product-conforming matrices, it can be verified that the differential of their product is

$$d(\mathbf{XY}) = (d\mathbf{X})\mathbf{Y} + \mathbf{X}(d\mathbf{Y}). \tag{D.43}$$

which is an extension of the well known rule $d(xy) = y\,dx + x\,dy$ for scalar functions.

EXAMPLE D.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 $\mathbf{X}$:

$$d|\mathbf{X}| = \sum_{i,j} \frac{\partial|\mathbf{X}|}{\partial x_{ij}} dx_{ij} = \sum_{i,j} \mathbf{X}_{ij}\, dx_{ij} = \text{tr}(|\mathbf{X}|\mathbf{X}^{-1})^T\, d\mathbf{X}) = \text{tr}(\mathbf{Z}^T\, d\mathbf{X}), \tag{D.44}$$

where X_{ij} denotes the cofactor of x_{ij} in $\mathbf{X}$.

EXAMPLE D.7

With the same assumptions as above, find $d(\mathbf{X}^{-1})$. The quickest derivation follows by differentiating both sides of the identity $\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$:

$$d(\mathbf{X}^{-1})\mathbf{X} + \mathbf{X}^{-1}\, d\mathbf{X} = \mathbf{0}, \tag{D.45}$$

from which

$$d(\mathbf{X}^{-1}) = -\mathbf{X}^{-1}\, d\mathbf{X}\, \mathbf{X}^{-1}. \tag{D.46}$$

If $\mathbf{X}$ reduces to the scalar x we have

$$d\left(\frac{1}{x}\right) = -\frac{dx}{x^2}. \tag{D.47}$$