INTRODUCTION to
FINITE ELEMENT METHODS

CARLOS A. FELIPPA

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

Last updated Fall 2004

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

Preface

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

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

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

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

Book Objectives

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

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

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

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

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

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

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

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

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

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

Book Organization

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

Part I: The Direct Stiffness Method. This part comprises Chapters 1 through 11. It covers major aspects of the Direct Stiffness Method (DSM). This is the most important realization of FEM, and the one implemented in general-purpose commercial finite element codes used by practicing engineers. Following 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 chapters the physical interpretation is emphasized for pedagogical convenience, as unifying vision of this "horizontal" framework.

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

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

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

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

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

Exercises

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

`[type:rating]`

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

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

5 A simple question that can be answered in seconds, or is already answered in the text if the student has read and understood the material.

10 A straightforward question that can be answered in minutes.

15 A relatively simple question that requires some thinking, and may take on the order of half to one hour to answer.

20 Either a problem of moderate difficulty, or a straightforward one requiring lengthy computations or some programming, normally taking one to six hours of work.

25 A scaled up version of the above, estimated to require six hours to one day of work.

30 A problem of moderate difficulty that normally requires on the order of one or two days of work. Arriving at the answer may involve a combination of techniques, some background or reference material, or 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.

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

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

Acknowledgements

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

Chapter Contents

Section

Appendices

1
Overview

TABLE OF CONTENTS

§1.1. Book Scope

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

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

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

§1.2. Where the Material Fits

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

§1.2.1. Top Level Classification

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

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

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

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

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

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

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

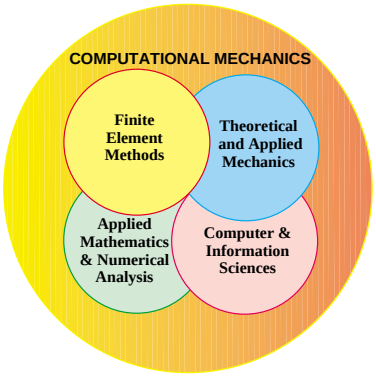


FIGURE 1.1. The "pizza slide:" Computational Mechanics integrates aspects of four disciplines.

§1.2.2. Computational Mechanics

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

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

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

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

Continuum mechanics studies bodies at the macroscopic level, using continuum models in which the microstructure is homogenized by phenomenological averaging. The two traditional areas of application are *solid* and *fluid mechanics*. *Structural mechanics* is a conjoint branch of solid mechanics, since structures, for obvious reasons, are fabricated with solids. Computational solid mechanics favors 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 subareas are hydrodynamics, aerodynamics, atmospheric physics, propulsion, and combustion.

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

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

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

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

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

§1.2.3. Statics versus Dynamics

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

$$\textbf{Continuum mechanics} \left\{ \begin{array}{l} \textbf{Statics} \left\{ \begin{array}{l} \textbf{Time Invariant} \\ \textit{Quasi-static} \end{array} \right. \\ \textit{Dynamics} \end{array} \right. \tag{1.3}$$

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

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

§1.2.4. Linear versus Nonlinear

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

$$\textbf{Statics} \left\{ \begin{array}{l} \textbf{Linear} \\ \textit{Nonlinear} \end{array} \right. \tag{1.4}$$

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

---

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

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

§1.2.5. Discretization Methods

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

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

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

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

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

§1.2.6. FEM Formulation Levels

The term *Finite Element Method* actually identifies a broad spectrum of techniques that share common features. Since its emergence in the framework of the Direct Stiffness Method (DSM) over 1956–1964, [765,768] FEM has expanded like a tsunami, surging from its origins in aerospace structures to cover a wide range of nonstructural applications, notably thermomechanics, fluid dynaamics, and electromagnetics. The continuously expanding range makes taxology difficult. Restricting ourselves to applications in computational solid and structural mechanics (CSSM), one classificaation of particular relevance to this book is

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

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

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

The next two levels are covered in the Advanced Finite Element Methods book [255]. The third one requires a deeper exposure to variational methods in mechanics, notably multifield and hybrid principles. The last level (templates) is the pinnacle "where the rivers of our wisdom flow into one another." Reaching it requires both mastery of advanced variational principles, as well as the confidence and fortitude to discard them along the way to the top.

§1.2.7. FEM Choices

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

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

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

§1.2.8. Finally: What The Book Is About

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

> The model-based simulation of linear static structures discretized by FEM, formulated at the two lowest levels of (1.6). (1.8)

Of the FEM variants listed in (1.7) emphasis will be placed on the *displacement* PUV choice and *stiffness* solution, just like in [257]. This particular combination is called the *Direct Stiffness Method* or DSM.

---

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

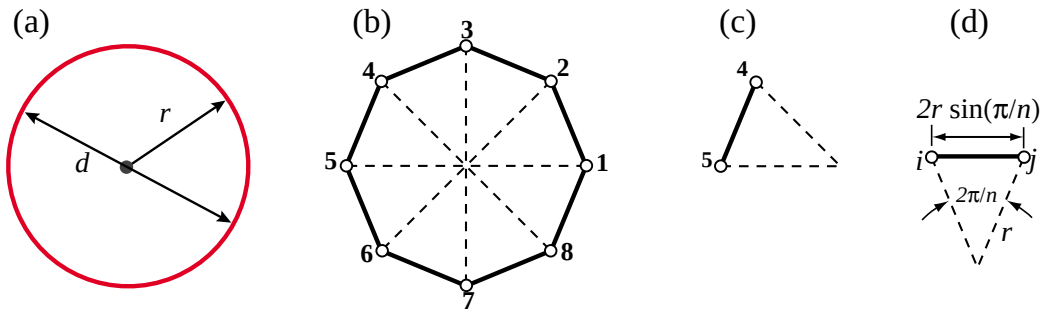


FIGURE 1.2. The "find π" problem treated with FEM concepts: (a) continuum object, (b) a discrete approximation by inscribed regular polygons, (c) disconnected element, (d) generic element.

§1.3. What Does a Finite Element Look Like?

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

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

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

n	$\pi_n = n \sin(\pi/n)$	Extrapolated by Wynn-ϵ	Exact π to 16 places
1	0.000000000000000		
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

Values of π_n obtained for $n = 1, 2, 4, \ldots 256$ and $r = 1$ are listed in the second column of Table 1.1. As can be seen the convergence to π is fairly slow. However, the sequence can be transformed by Wynn's ϵ algorithm[6] into that shown in the third column. The last value displays 15-place accuracy.

[6] A widely used lozenge extrapolation algorithm that speeds up the convergence of many sequences. See, e.g, [812].

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

This example has been offered in the FEM literature, e.g. in [476], to aduce that finite element ideas can be traced to Egyptian mathematicians from *circa* 1800 B.C., as well as Archimedes' famous studies on circle rectification by 250 B.C. But comparison with the modern FEM, as covered in following Chapters, shows this to be a stretch. The example does not illustrate the concept of degrees of freedom, conjugate quantities and local-global coordinates. It is guilty of circular reasoning: the compact formula $\pi = \lim_{n\to\infty} n \sin(\pi/n)$ uses the unknown π in the right hand side.[8] Reasonable people would argue that a circle is a simpler object than, say, a 128-sided polygon. Despite these flaws the example is useful in one respect: showing a fielder's choice in the replacement of one mathematical object by another. This is at the root of the simulation process described next.

§1.4. The FEM Analysis Process

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

§1.4.1. The Physical FEM

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

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

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

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

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

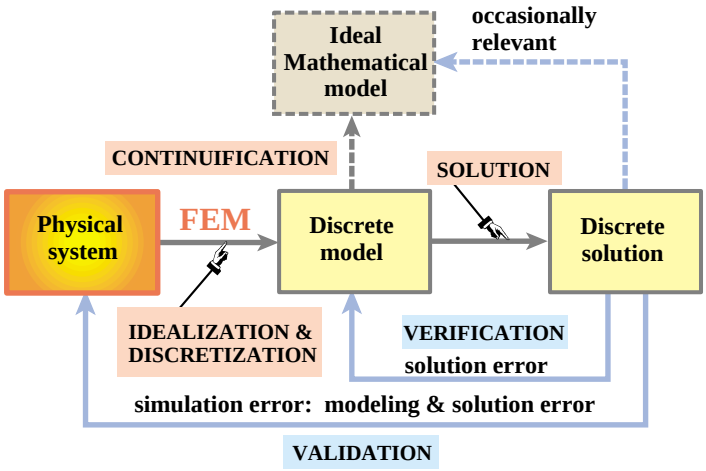


FIGURE 1.3. The Physical FEM. The physical system (left box) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

The concept of *error* arises in the Physical FEM in two ways. These are known as *verification* and *validation*, respectively.[9] Verification is done by replacing the discrete solution into the discrete model to get the solution error. This error is not generally important. Substitution in the ideal mathematical model in principle provides the *discretization error*. This step is rarely useful in complex engineering systems, however, because there is no reason to expect that the continuum model exists, and even if it does, that it is more physically relevant than the discrete model.

Validation tries to compare the discrete solution against observation by computing the *simulation error*, which combines modeling and solution errors. As the latter is typically unimportant, the simulation error in practice can be identified with the modeling error. In real-life applications this error overwhelms the others.[10]

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

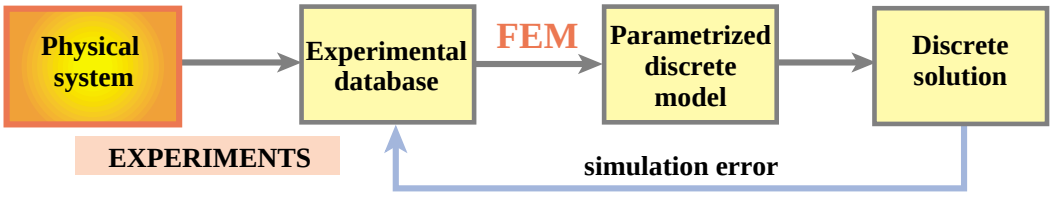


FIGURE 1.4. Model updating process in the Physical FEM.

[9] Programming analogs: static and dynamic testing are called *verification* and *validation*, respectively. Static testing is carried at the source level (e.g., code walkthroughs, compilation) whereas dynamic testing is done by running the code.

[10] "All models are wrong; some are useful" (George Box)

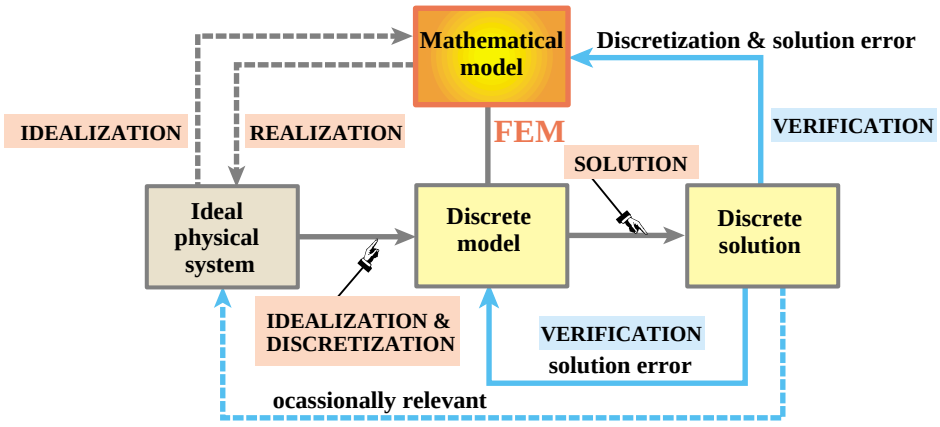


FIGURE 1.5. The Physical FEM. The physical system (left box) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

§1.4.2. The Mathematical FEM

The other canonical way of using FEM focuses on the mathematics. The process steps are illustrated in Figure 1.5. The spotlight now falls on the *mathematical model*. This is often an ordinary differential equation (ODE), or a partial differential equation (PDE) in space and time. A discrete finite element model is generated from a variational or weak form of the mathematical model.[11] This is the *discretization* step. The FEM equations are solved as described for the Physical FEM.

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

The concept of *error* arises when the discrete solution is substituted in the "model" boxes. This replacement is generically called *verification*. As in the Physical FEM, the *solution error* is the amount by which the discrete solution fails to satisfy the discrete equations. This error is relatively unimportant when using computers, and in particular direct linear equation solvers, for the solution step. More relevant is the *discretization error*, which is the amount by which the discrete solution fails to satisfy the mathematical model.[12] Replacing into the ideal physical system would in principle quantify modeling errors. In the Mathematical FEM this is largely irrelevant, however, because the ideal physical system is merely that: a figment of the imagination.

§1.4.3. Synergy of Physical and Mathematical FEM

The foregoing canonical sequences are not exclusive but complementary. This synergy[13] is one of the reasons behind the power and acceptance of the method. Historically the Physical FEM was the

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

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

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

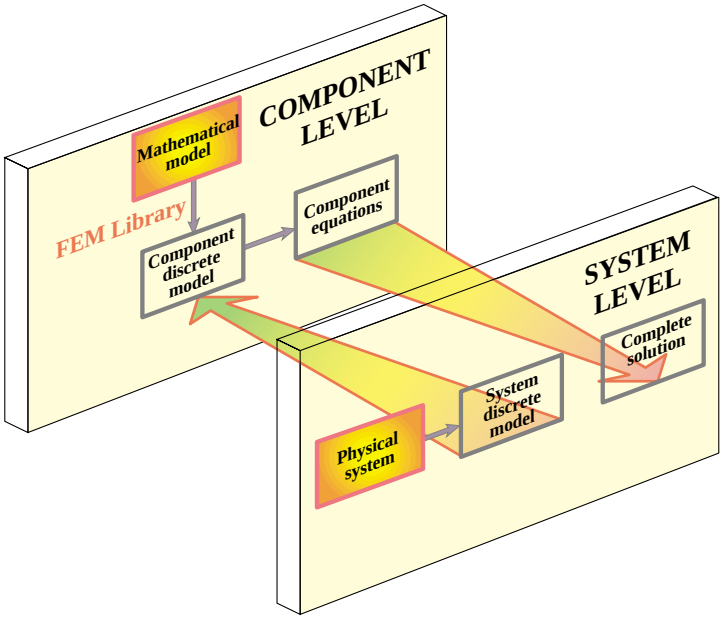


FIGURE 1.6. Combining physical and mathematical modeling through multilevel FEM. Only two levels (system and component) are shown for simplicity.

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

A glance at the schematics of a commercial jet aircraft makes obvious the reasons behind the Physical FEM. There is no simple differential equation that captures, at a continuum mechanics level,[14] the structure, avionics, fuel, propulsion, cargo, and passengers eating dinner. There is no reason for despair, however. The time honored *divide and conquer* strategy, coupled with *abstraction*, comes to the rescue.

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

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

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

[14] Of course at the (sub)atomic level quantum mechanics works for everything, from landing gears to passengers. But it would be slightly impractical to represent the aircraft by, say, 10^{36} interacting particles modeled by the Schrödinger equations. More seriously, Truesdell and Toupin correctly note that "*Newtonian mechanics, while not appropriate to the corpuscles making up a body, agrees with experience when applied to the body as a whole*, except for certain phenomena of astronomical scale" [759, p. 228].

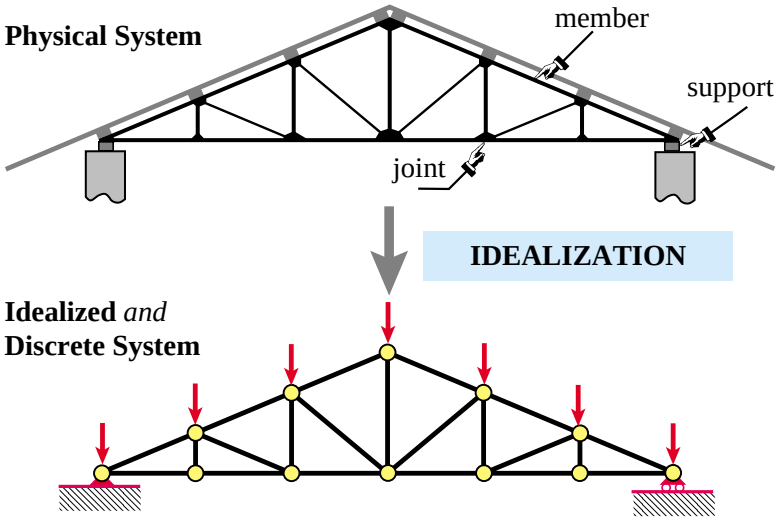


FIGURE 1.7. The idealization process for a simple structure. The physical system — here a conventional roof truss — is directly idealized by the mathematical model: a pin-jointed bar assembly. For this particular structure idealized and discrete models coalesce.

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

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

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

§1.4.4. Streamlined Idealization and Discretization

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

§1.5. Method Interpretations

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

§1.5.1. Physical Interpretation

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

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

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

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

§1.5.2. Mathematical Interpretation

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

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

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

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

Remark 1.5. It is interesting to note some similarities in the development of Heaviside's operational methods, Dirac's delta-function calculus, and the FEM. These three methods appeared as ad-hoc computational devices created by engineers and physicists to deal with problems posed by new science and technology (electricity, quantum mechanics, and delta-wing aircraft, respectively) with little help from the mathematical establishment.[15] Only some time after the success of the new techniques became apparent were new branches of mathematics (operational calculus, distribution theory and piecewise-approximation theory, respectively) constructed to justify that success. In the case of the finite element method, the development of a formal mathematical theory started in the late 1960s, and much of it is still in the making.

§1.6. Keeping the Course

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

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

In Chapters 2 and 3 the time is frozen at about 1965, and the DSM presented as an aerospace engineer of that time would have understood it. This is not done for sentimental reasons, although that happens to be the year in which the writer began thesis work on FEM under Ray Clough. Virtually all FEM commercial codes are now based on the DSM and the computer implementation has not essentially changed since the late 1960s.[16] What has greatly improved since is "marketing sugar": user interaction and visualization.

§1.7. *What is Not Covered

The following topics are not covered in this book:

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

---

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

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

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

Topics 1–5 belong to what may be called "Advanced Linear FEM", which is covered in the book [255]. Topics 6–7 pertain to "Nonlinear FEM", which is covered in the book [258]. Topics 8–10 fall into advanced applications, covrede in other books in preparation, whereas 11 is an interdisciplinary topic that interweaves with computer science.

§1.8. The Origins of the Finite Element Method

This section moved to Appendix O to facilitate further expansion.

§1.9. Recommended Books for Linear FEM

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

Basic level (reference): Zienkiewicz and Taylor [837]. This two-volume set is a comprehensive upgrade of the previous edition [835]. Primarily an encyclopœdic reference work that gives a panoramic coverage of FEM applications, as well as a comprehensive list of references. Not a textbook or monograph. Prior editions suffered from loose mathematics, largely fixed in this one. A three-volume fifth edition has appeared recently.

Basic level (textbook): Cook, Malkus and Plesha [149]. The third edition is comprehensive in scope although the coverage is more superficial than Zienkiewicz and Taylor. A fourth edition has appeared recently.

Intermediate level: Hughes [389]. It requires substantial mathematical expertise on the part of the reader. Recently (2000) reprinted as Dover edition.

Mathematically oriented: Strang and Fix [705]. Still the most readable mathematical treatment for engineers, although outdated in several subjects. Out of print.

Best value for the $$$: Przemieniecki's Dover edition [603], list price $15.95 (2003). A reprint of a 1966 McGraw-Hill book. Although woefully outdated in many respects (the word "finite element" does not appear except in post-1960 references), it is a valuable reference for programming simple elements. Contains a fairly detailed coverage of substructuring, a practical topic missing from the other books. Comprehensive bibliography in Matrix Structural Analysis up to 1966.

Most fun (if you appreciate British "humor"): Irons and Ahmad [401]. Out of print.

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

§1.9.1. Hasta la Vista, Fortran

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

Chemistry departments because of large amounts of legacy code.) So one end of the pipeline is drying up. Low-level scientific programming[17] is moving to C and C++, mid-level to Java, Perl and Python, high-level to Matlab, Mathematica and their free-source Linux equivalents. How attractive can a book teaching in a dead language be?

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

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

Notes and Bibliography

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

> " My involvement with the FEM began when I was employed by the Boeing Airplane Company in Seattle during summer 1952 as a member of their summer faculty program. When I had joined the civil engineering faculty at Berkeley in 1949, I decided to take advantage of my MIT structural dynamics background by taking up the field of Earthquake Engineering. So because the Boeing summer faculty program offered positions with their structural dynamics unit, I seized on that as the best means of advancing my preparation for the earthquake engineering field. I was particularly fortunate in this choice of summer work at Boeing because the head of their structural dynamics unit was Mr. M. J. Turner — a very capable man in dealing with problems of structural vibrations and flutter.
>
> When I arrived for the summer of 1952, Jon Turner asked me to work on the vibration analysis of a delta wing structure. Because of its triangular plan form, this problem could not be solved by procedures based on standard beam theory; so I spent the summer of 1952 trying to formulate a delta wing model built up as an assemblage of one-dimensional beams and struts. However, the results of deflection analyses based on this type of mathematical model were in very poor agreement with data obtained from laboratory tests of a scale model of a delta wing. My final conclusion was that my summer's work was a total failure—however, at least I learned what did not work.

[17] "A programming language is low level when its programs require attention to the irrelevant" (Alan Perlis).

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

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

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

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

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

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 1
Overview

EXERCISE 1.1 [A:15] Work out Archimedes' problem using a circumscribed regular polygon, with $n = 1, 2, 4, \ldots 256$. 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.

EXERCISE 1.3 [D:30] In one of the earliest articles on the FEM, Clough [139] writes:

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

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

2

The Direct Stiffness Method I

TABLE OF CONTENTS

§2.1. Foreword

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

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

§2.2. Why A Plane Truss?

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

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

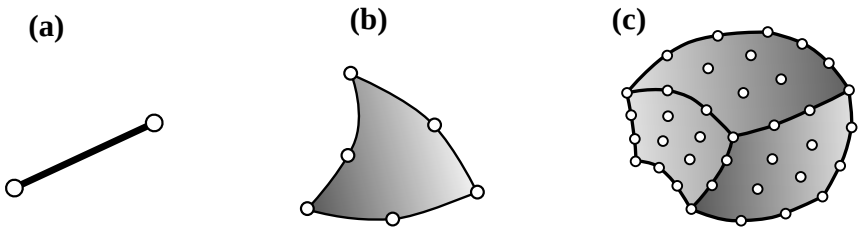


FIGURE 2.1. From the simplest through progressively more complex structural finite elements: (a) two-node bar element for trusses, (b) six-node triangle for thin plates, (b) 64-node tricubic, "brick" element for three-dimensional solid analysis.

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

(a) Computations can be entirely done by hand as long as the structure contains just a few elements. This allows various steps of the solution procedure to be carefully examined and understood (learning by doing) before passing to the computer implementation. Doing hand computations on more complex finite element systems rapidly becomes impossible.

(b) The computer implementation on any programming language is relatively simple and can be assigned as preparatory computer homework before reaching Part III.

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

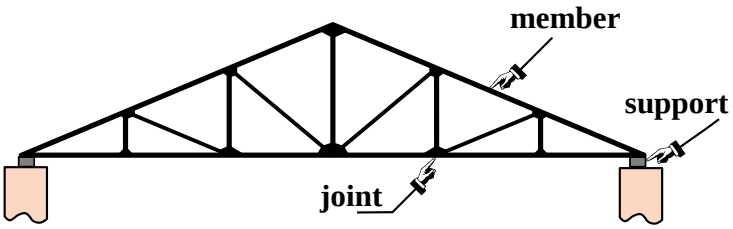


FIGURE 2.2. An actual plane truss structure. That shown is typical of a roof truss used in building construction for rather wide spans, say, over 10 meters. For shorter spans, as in residential buildings, trusses are simpler, with fewer bays.

§2.3. Truss Structures

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

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

§2.4. Idealization

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

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

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

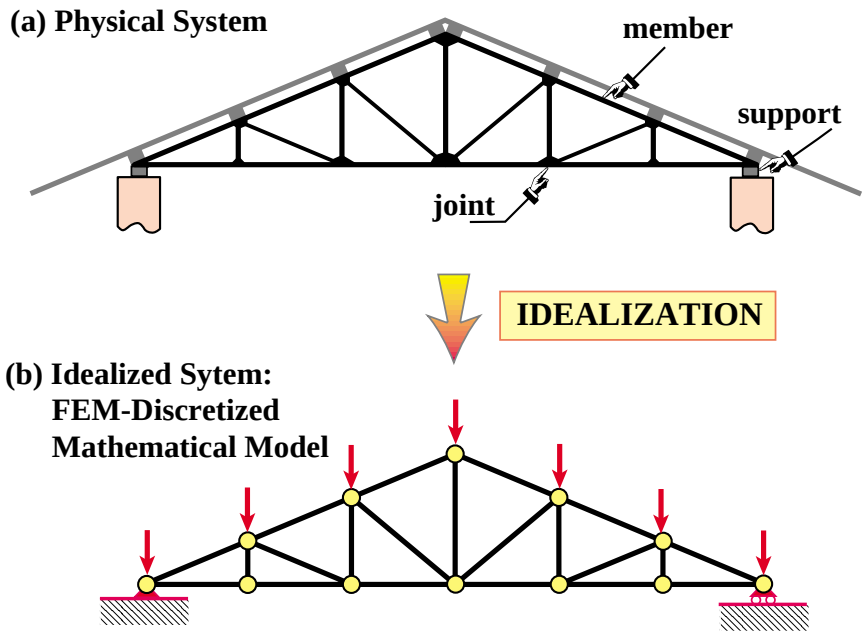


FIGURE 2.3. Idealization of roof truss: (a) physical system, (b) idealization as FEM discretized mathematical model.

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

§2.5. The Example Truss

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

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

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

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

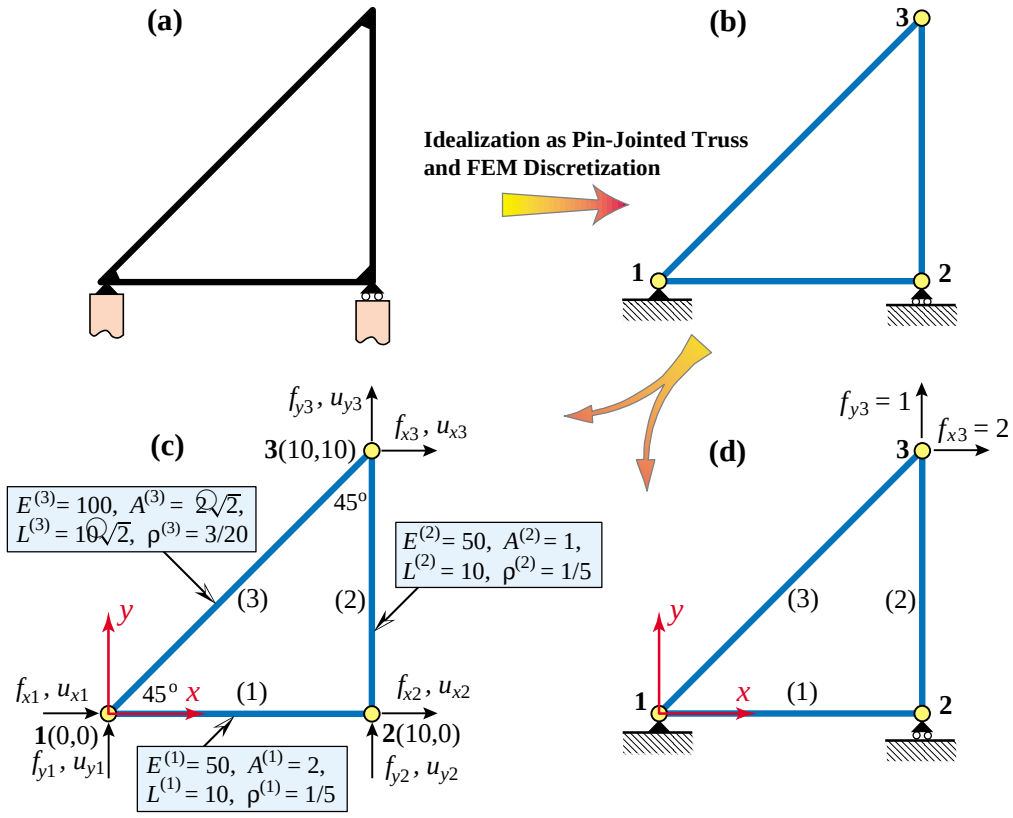


FIGURE 2.4. The three-member example truss: (a) physical structure; (b) idealization as a pin-jointed bar assemblage; (c) geometric, material and fabrication properties; (d) support conditions and applied loads.

§2.6. Members, Joints, Forces and Displacements

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

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

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

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

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

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

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

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

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

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

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

§2.7. The Master Stiffness Equations

The *master stiffness equations* relate the joint forces **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.[4] If both assumptions hold the relation must be homogeneous and

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

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

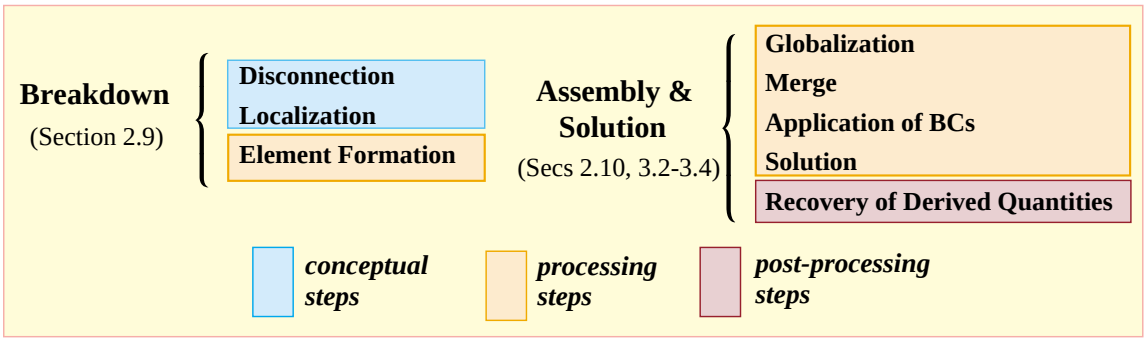


FIGURE 2.5. The Direct Stiffness Method steps.

expressable in component form as

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

In matrix notation:

$$\boxed{\mathbf{f} = \mathbf{K}\,\mathbf{u}.} \tag{2.3}$$

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 (2.2). The entries of the stiffness matrix are often called *stiffness coefficients* and have a physical interpretation discussed below.

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

Remark 2.2. *Interpretation of Stiffness Coefficients*. The following interpretation of the entries of **K** is 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,

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

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

§2.8. The DSM Steps

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

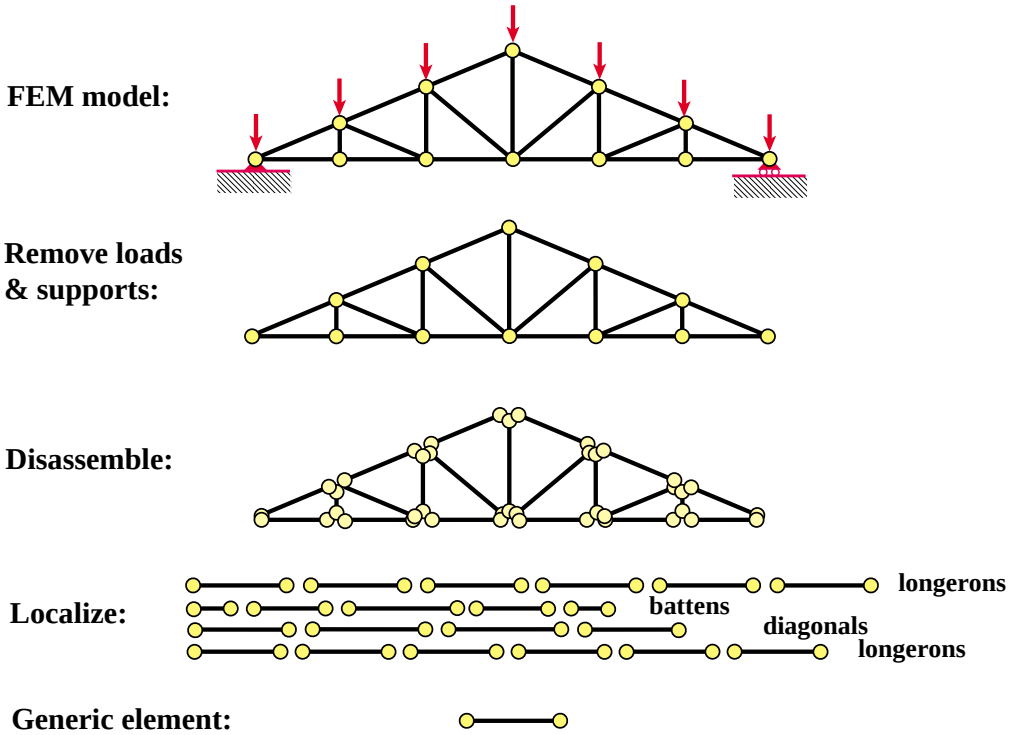


FIGURE 2.6. Visualization of the DSM breakdown stage.

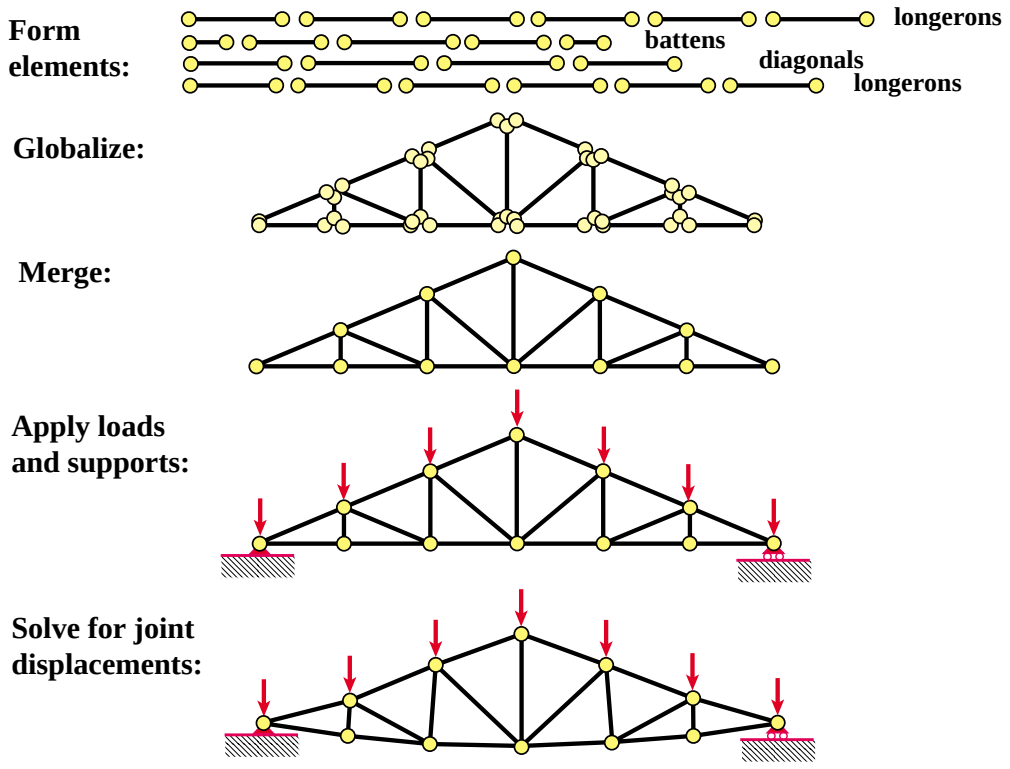


FIGURE 2.7. Visualization of the DSM assembly-and-solution stage.

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

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

§2.9. Breakdown Stage

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

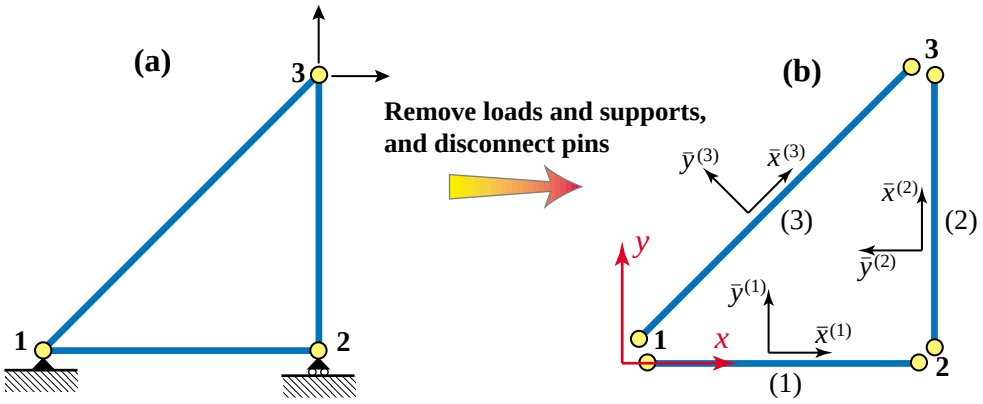


FIGURE 2.8. Disconnection step: (a) idealized example truss; (b) removal of loads and support, disconnection into members (1), (2) and (3), and selection of local coordinate systems. The latter are drawn offset from member axes for visualization convenience.

§2.9.1. Disconnection

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

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

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

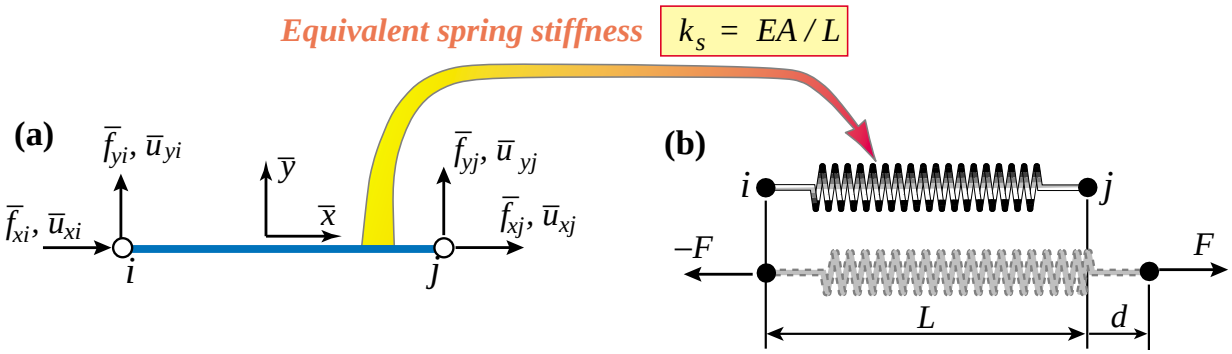


FIGURE 2.9. Generic truss member referred to its local coordinate system $\{\bar{x}, \bar{y}\}$: (a) idealization as 2-node bar element, (b) interpretation as equivalent spring. Element identification number e dropped to reduce clutter.

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

§2.9.2. Localization

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

§2.9.3. Member Stiffness Equations

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

$$\bar{\mathbf{f}} = \bar{\mathbf{K}}\,\bar{\mathbf{u}}, \tag{2.5}$$

which written out in full become

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

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

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

interpretation illustrated in Figure 2.9(b). If the member properties are *uniform* along its length, Mechanics of Materials bar theory tells us that[6]

$$k_s = \frac{EA}{L}, \tag{2.7}$$

Consequently the force-displacement equation is

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

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

$$F = \bar{f}_{xj} = -\bar{f}_{xi}, \qquad d = \bar{u}_{xj} - \bar{u}_{xi}, \tag{2.9}$$

which express force equilibrium[7] and kinematic compatibility, respectively. Combining (2.8) and (2.9) we obtain the matrix relation[8]

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

Hence

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

This is the truss stiffness matrix in local coordinates.

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

§2.10. Assembly and Solution Stage: Globalization

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

---

[6] See for example, Chapter 2 of [68].

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

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

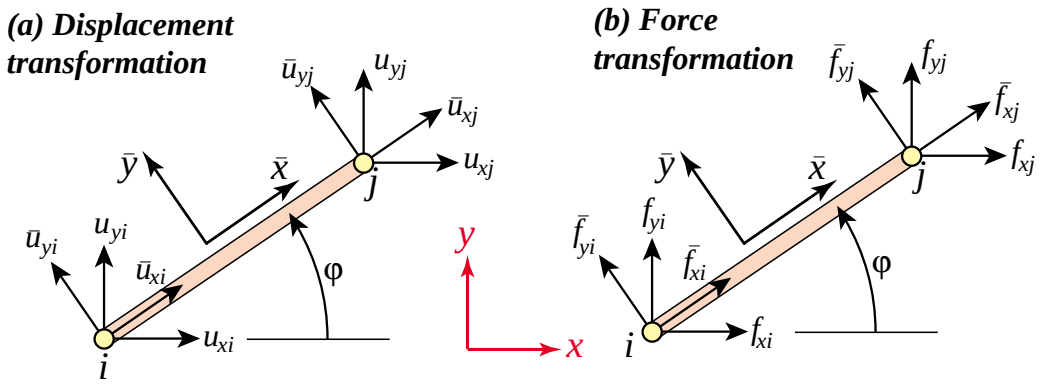


FIGURE 2.10. The transformation of node displacement and force components from the local system $\{\bar{x}, \bar{y}\}$ to the global system $\{x, y\}$.

§2.10.1. Displacement and Force Transformations

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

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

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

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

The 4×4 matrix that appears above is called a *displacement transformation matrix* and is denoted[9] by $\mathbf{T}$. The node forces transform as $f_{xi} = \bar{f}_{xi}\, c - \bar{f}_{yi}\, s$, etc., which in matrix form become

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

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

---

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

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

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

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

§2.10.2. Global Member Stiffness Equations

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

$$\mathbf{f}^e = \mathbf{K}^e \mathbf{u}^e. \tag{2.15}$$

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

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

Inserting these matrix expressions into $\bar{\mathbf{f}}^e = \overline{\mathbf{K}}^e \bar{\mathbf{u}}^e$ and comparing with (2.15) we find that the member stiffness in the global system $\{x, y\}$ can be computed from the member stiffness $\bar{\mathbf{K}}^e$ in the local system $\{\bar{x}, \bar{y}\}$ through the congruent transformation[11]

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

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

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

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

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

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

$$\begin{bmatrix} f^{(1)}_{x1} \\ f^{(1)}_{y1} \\ f^{(1)}_{x2} \\ f^{(1)}_{y2} \end{bmatrix} = 10 \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^{(1)}_{x1} \\ u^{(1)}_{y1} \\ u^{(1)}_{x2} \\ u^{(1)}_{y2} \end{bmatrix}. \tag{2.19}$$

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

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

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

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

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

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

Notes and Bibliography

The Direct Stiffness Method has been the dominant FEM version since the mid-1960s, and is the procedure followed by all major commercial codes in current use. The general DSM was developed at Boeing in the mid and late 1950s, through the leadership of Jon Turner [766,768], and had defeated its main competitor, the Force Method, by 1970 [240].

All applications-oriented FEM books cover the DSM, although the procedural steps are sometimes not clearly delineated. In particular, the textbooks recommended in §1.9 offer adequate expositions.

Trusses, also called bar assemblies, are usually the first structures treated in Mechanics of Materials books written for undergraduate courses in Aerospace, Civil and Mechanical Engineering. Two widely used textbooks at this level are [68] and [595].

Steps in the derivation of stiffness matrices for truss elements are well covered in a number of early treatment of finite element books, of which Chapter 5 of Przemieniecki [603] is a good example.

Force and displacement transformation matrices for structural analysis were introduced by G. Kron [428].

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 2
The Direct Stiffness Method I

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

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

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

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

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

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

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 can be extended to general systems)

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

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

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 (2.8) by the principle of Minimum Potential Energy (MPE). In Mechanics of Materials it is shown that the total potential energy of the axially loaded bar is

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

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

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

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

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

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

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

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

3

The Direct Stiffness Method II

TABLE OF CONTENTS

§3.1. The Remaining DSM Steps

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

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

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

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

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

§3.2. Assembly: Merge

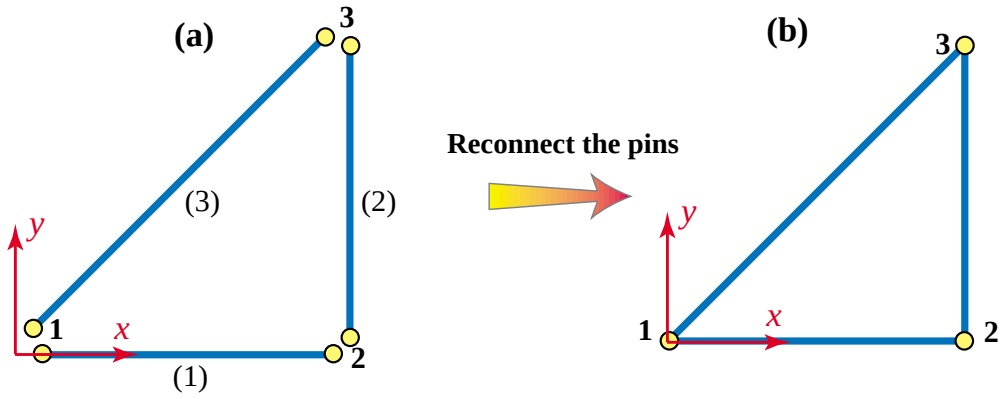


FIGURE 3.1. Physical meaning of merge operation: (a) disconnected example truss after globalization; (b) reconnected truss with the pins put back into the joints.

§3.2.1. Governing Rules

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

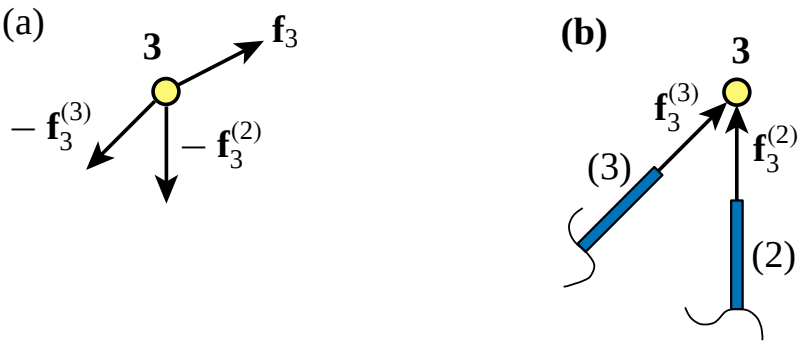


FIGURE 3.2. The force equilibrium of joint 3 of the example truss, depicted as an FBD in (a). Here $\mathbf{f}_3$ is the known external joint force applied on the joint. Internal forces $\mathbf{f}_3^{(2)}$ and $\mathbf{f}_3^{(3)}$ are applied *by the joint on the members*, as illustrated in (b). Thus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

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

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

1. *Compatibility of displacements*: The displacements of all members that meet at a joint are the same.
2. *Force equilibrium*: The sum of internal forces exerted by all members that meet at a joint balances the external force applied to that joint. (3.1)

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

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

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

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

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

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

For member (1):

$$\begin{bmatrix} f^{(1)}_{x1} \\ f^{(1)}_{y1} \\ f^{(1)}_{x2} \\ f^{(1)}_{y2} \\ f^{(1)}_{x3} \\ f^{(1)}_{y3} \end{bmatrix} = \begin{bmatrix} 10 & 0 & -10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u^{(1)}_{x1} \\ u^{(1)}_{y1} \\ u^{(1)}_{x2} \\ u^{(1)}_{y2} \\ u^{(1)}_{x3} \\ u^{(1)}_{y3} \end{bmatrix}. \tag{3.4}$$

For member (2):

$$\begin{bmatrix} f^{(2)}_{x1} \\ f^{(2)}_{y1} \\ f^{(2)}_{x2} \\ f^{(2)}_{y2} \\ f^{(2)}_{x3} \\ f^{(2)}_{y3} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & -5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -5 & 0 & 5 \end{bmatrix} \begin{bmatrix} u^{(2)}_{x1} \\ u^{(2)}_{y1} \\ u^{(2)}_{x2} \\ u^{(2)}_{y2} \\ u^{(2)}_{x3} \\ u^{(2)}_{y3} \end{bmatrix}. \tag{3.5}$$

For member (3):

$$\begin{bmatrix} f^{(3)}_{x1} \\ f^{(3)}_{y1} \\ f^{(3)}_{x2} \\ f^{(3)}_{y2} \\ f^{(3)}_{x3} \\ f^{(3)}_{y3} \end{bmatrix} = \begin{bmatrix} 10 & 10 & 0 & 0 & -10 & -10 \\ 10 & 10 & 0 & 0 & -10 & -10 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & -10 & 0 & 0 & 10 & 10 \\ -10 & -10 & 0 & 0 & 10 & 10 \end{bmatrix} \begin{bmatrix} u^{(3)}_{x1} \\ u^{(3)}_{y1} \\ u^{(3)}_{x2} \\ u^{(3)}_{y2} \\ u^{(3)}_{x3} \\ u^{(3)}_{y3} \end{bmatrix}. \tag{3.6}$$

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

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

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

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

These three equations can be represented in compact matrix notation as

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

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

$$\boxed{\mathbf{f} = \mathbf{f}^{(1)} + \mathbf{f}^{(2)} + \mathbf{f}^{(3)} = \left(\mathbf{K}^{(1)} + \mathbf{K}^{(2)} + \mathbf{K}^{(3)}\right)\mathbf{u} = \mathbf{K}\,\mathbf{u},} \tag{3.11}$$

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

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

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

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

§3.3. Solution

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

§3.3.1. Applying Boundary Conditions by Reduction

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

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

$$u_{x1} = u_{y1} = u_{y2} = 0, \tag{3.13}$$

whereas the known applied forces are

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

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

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

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

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

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

§3.3.2. Solving for Displacements

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

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

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

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

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

§3.4. PostProcessing

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

§3.4.1. Recovery of Reaction Forces

Premultiplying the complete displacement solution (3.17) by **K** we get

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

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

§3.4.2. Recovery of Internal Forces and Stresses

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

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

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

$$d^e = \bar{u}^e_{xj} - \bar{u}^e_{xi}, \qquad F^e = \frac{E^e A^e}{L^e} d^e. \tag{3.19}$$

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

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

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

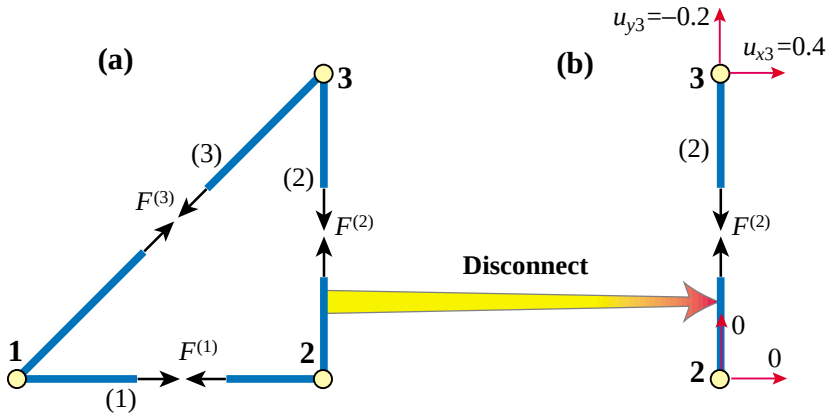


FIGURE 3.3. Internal force recovery for example truss: (a) member axial forces $F^{(1)}$, $F^{(2)}$ and $F^{(3)}$, with arrow directions pertaining to tension; (b) details of computation for member (2).

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

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

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

§3.4.3. *Reaction Recovery: General Case

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

$$\boxed{\mathbf{K}\,\mathbf{u} = \mathbf{f} = \mathbf{f}^a + \mathbf{f}^r.} \tag{3.21}$$

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

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

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

The fixed-free prismatic bar pictured in Figure 3.4(a) is subjected to a uniformly line load q per unit length. The bar has length L, elastic modulus E and cross-section area A. It is discretized by two equal-size elements as shown in Figure 3.4(b). The three x node displacements $u_1 = u_{x1}$, $u_2 = u_{x2}$ and $u_3 = u_{x3}$ are taken as

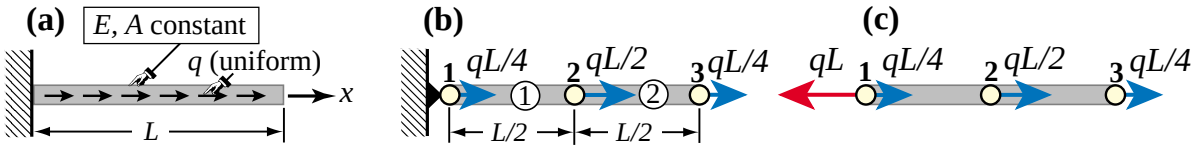


FIGURE 3.4. A simple problem to illustrate reaction recovery of support reactions when nonzero applied loads act on supports. (a) bar under distributed load; (b) two-element FEM idealization; (c) free body diagram showing applied node forces in blue and support reaction in red.

degrees of freedom. The line load is converted to node forces $f_1^a = \frac{1}{4}qL$, $f_2^a = \frac{1}{2}qL$ and $f_3^a = \frac{1}{4}qL$ at nodes 1, 2 and 3, respectively, using the EbE method discussed in Chapter 7. The master stiffness equations configured as per (3.21) are

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

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

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

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

§3.5. *Computer Oriented Assembly and Solution

§3.5.1. *Assembly by Freedom Pointers

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

(I) Member stiffness matrices are *not* expanded. Their entries are directly merged into those of **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

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

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

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

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

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

§3.5.2. *Applying DBC by Modification

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

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

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

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

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

§3.6. Prescribed Nonzero Displacements

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

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

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

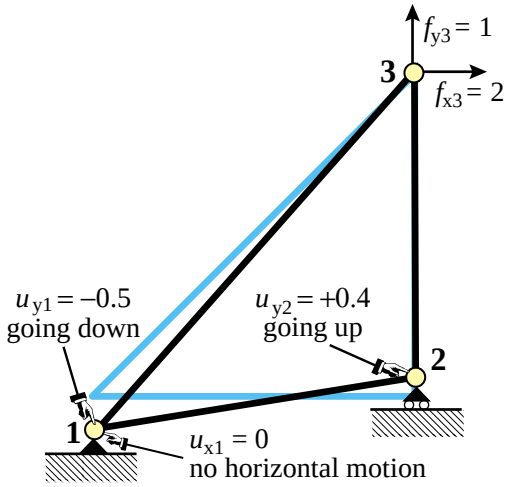


FIGURE 3.5. The example truss with prescribed nonzero vertical displacements at joints 1 and 2.

§3.6.1. Application of Nonzero-DBCs by Reduction

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

§3.6.2. *Application of Nonzero-DBCs by Modification

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

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

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

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

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

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

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

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

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

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

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

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

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

$$\mathbf{K}_{11}\mathbf{u}_1 = \mathbf{f}_1. \tag{3.39}$$

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

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

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

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

$$\widehat{\mathbf{K}}\mathbf{u} = \hat{\mathbf{f}}. \tag{3.41}$$

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

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

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

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

Notes and Bibliography

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

(1) Some treatments apply support conditions *during* merge, explicitly eliminating known displacement freedoms as the elements are processed and merged into **K**. The output of the assembly process is what is called here a reduced stiffness matrix.[2]

(2) In the *frontal solution method* of Irons [399,401], assembly and solution are done concurrently. More precisely, as elements are formed and merged, displacement boundary conditions are applied, and Gauss elimination and reduction of the right hand side starts once the assembler senses (by tracking an "element wavefront") that no more elements contribute to a certain node.

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

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

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

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

The augment-and-add procedure for hand assembly of the master stiffness matrix is due to H. Martin [474].

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

References

Referenced items have been moved to Appendix R.

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

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

Homework Exercises for Chapter 3
The Direct Stiffness Method II

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

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

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

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

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

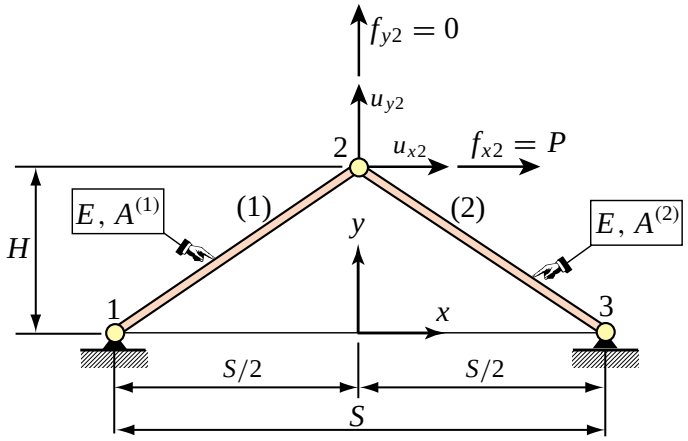


FIGURE E3.1. Truss structure for Exercises 3.6 and 3.7.

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

(a) Assemble the master stiffness equations. Any method: augment-and-add, 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 node 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 $F^{(1)} = -F^{(2)} = 15/2$.

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

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

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

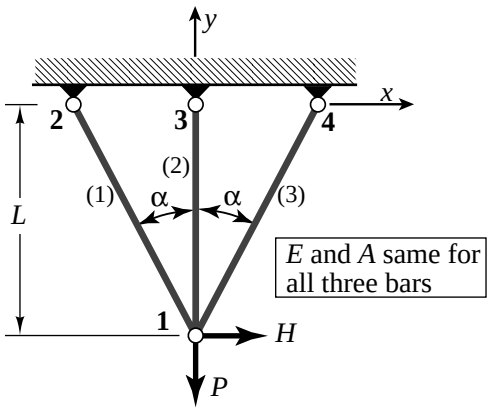


FIGURE E3.2. Truss structure for Exercise 3.8.

(a) Show that the master stiffness equations are

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

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

(b) Apply the BCs and show the 2-equation modified stiffness system.

(c) Solve for the displacements u_{x1} and u_{y1}. Check that the solution makes physical sense for the limit cases $\alpha \to 0$ and $\alpha \to \pi/2$. Why does u_{x1} "blow up" if $H \neq 0$ and $\alpha \to 0$?

(d) Recover the axial forces in the three members. Partial answer: $F^{(3)} = -H/(2s) + Pc^2/(1+2c^3)$. Why do $F^{(1)}$ and $F^{(3)}$ "blow up" if $H \neq 0$ and $\alpha \to 0$?

6

FEM Modeling: Introduction

TABLE OF CONTENTS

§6.1. Introduction

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

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

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

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

§6.2. FEM Terminology

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

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

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

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

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

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

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

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

Table 6.1. Significance of **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

stiffness equations:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}.} \tag{6.1}$$

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

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

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

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I\,.} \tag{6.2}$$

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

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

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

§6.3. Idealization

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

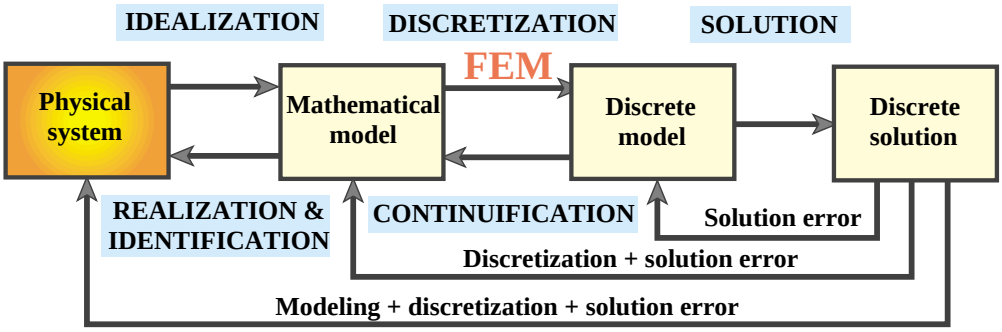


FIGURE 6.1. A simplified view of the physical simulation process, reproduced from Chapter 2 to illustrate modeling terminology.

§6.3.1. Models

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

> A model is a symbolic device built to simulate and predict aspects of behavior of a system. (6.3)

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

§6.3.2. Mathematical Models

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

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

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

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

[4] "All models are wrong, some are useful." (George Box)

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

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

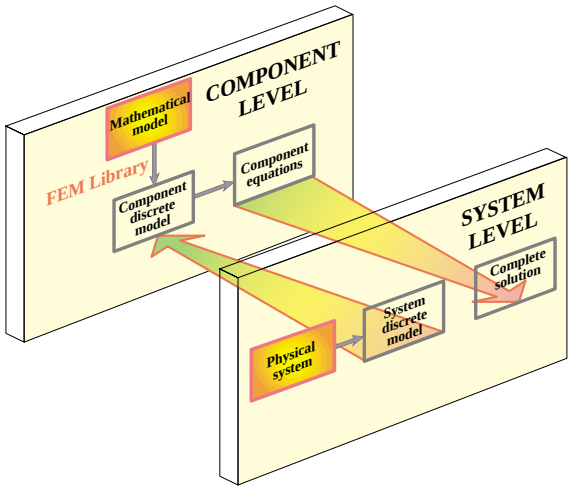


FIGURE 6.2. A reproduction of Figure 1.5 with some relabeling. Illustrates implicit modeling: picking elements from an existing FEM code consents to an idealization. This has professional as well as legal implications.

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

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

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

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

§6.3.3. Implicit vs. Explicit Modeling

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

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

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

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

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

§6.4. Discretization

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

§6.4.1. Analytical or Numerical?

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

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

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

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

---

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

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

§6.4.2. Error Sources and Approximation

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

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

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

§6.4.3. Other Discretization Methods

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

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

§6.5. The Finite Element Method

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

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

---

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

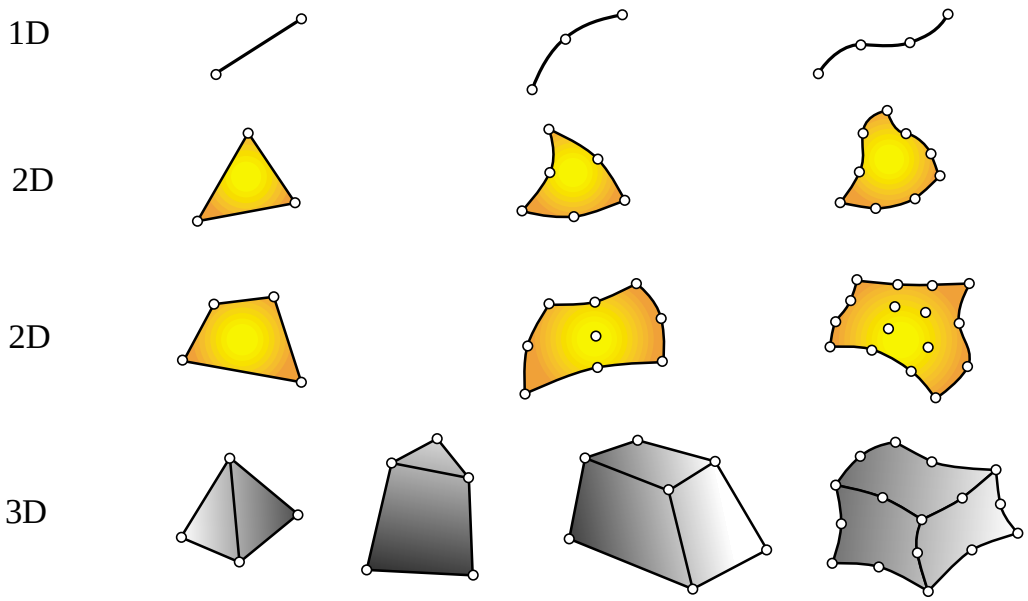


FIGURE 6.3. Typical finite element geometries in one through three dimensions.

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

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

§6.6. Element Attributes

Just like members in the truss example, one can take finite elements of any kind one at a time. Their local properties can be developed by considering them in isolation, as individual entities. This is the key to the modular programming of element libraries.

In the Direct Stiffness Method, elements are isolated by the disconnection and localization steps, which were described for the truss example in Chapter 2. The procedure involves the separation of elements from their neighbors by disconnecting the nodes, followed by referral of the element to a convenient local coordinate system.[10] After that we can consider *generic* elements: a bar element, a beam element, and so on. From the standpoint of the computer implementation, it means that you can write one subroutine or module that constructs, by suitable parametrization, all elements of one type, instead of writing a new one for each element instance.

Following is a summary of the data associated with an individual finite element. This data is used in finite element programs to carry out element level calculations.

§6.6.1. Element Dimensionality

[10] Both steps are only carried out in the modeler's mind. They are placed as part of the DSM for instructional convenience. In practice, processing begins directly at the element level.

Elements can have intrinsic dimensionality of one, two or three space dimensions.[11] There are also special elements with zero dimensionality, such as lumped springs or point masses. The intrinsic dimensionality can be expanded as necessary by use of kinematic transformations. For example a 1D element such as a bar, spar or beam may be used to build a model in 2D or 3D space.

§6.6.2. Element Nodes

Each element possesses a set of distinguishing points called *nodal points* or *nodes* for short. Nodes serve a dual purpose: definition of element geometry, and home for degrees of freedom. When a distinction is necessary we call the former *geometric nodes* and the latter *connection nodes*. For most elements studied here, geometric and connector nodes coalesce.

Nodes are usually located at the corners or end points of elements, as illustrated in Figure 6.3. In the so-called refined or higher-order elements nodes are also placed on sides or faces, as well as possibly the interior of the element.

Remark 6.1. In some elements geometric and connection nodes may be at different locations. This is illustrated by the Veubeke equilibrium triangle described in Chapter 15. Some elements have purely geometric nodes, also called *orientation nodes* to complete the definition of certain geometric attributes. An example is the spar element in 3D shown in Figure E5.2, in which a third geometric node is used to define a local plane.

§6.6.3. Element Geometry

The geometry of the element is defined by the placement of the geometric nodal points. Most elements used in practice have fairly simple geometries. In one-dimension, elements are usually straight lines or curved segments. In two dimensions they are of triangular or quadrilateral shape. In three dimensions the most common shapes are tetrahedra, pentahedra (also called wedges or prisms), and hexahedra (also called cuboids or "bricks"). See Figure 6.3.

§6.6.4. Element Degrees of Freedom

The element degrees of freedom (DOF) specify the *state* of the element. They also function as "handles" through which adjacent elements are connected. DOFs are defined as the values (and possibly derivatives) of a primary field variable at connector node points. The actual selection depends on criteria studied at length in Part II. Here we simply note that the key factor is the way in which the primary variable appears in the mathematical model. For mechanical elements, the primary variable is the displacement field and the DOF for many (but not all) elements are the displacement components at the nodes.

§6.6.5. Nodal Forces

There is always a set of nodal forces in a one-to-one correspondence with degrees of freedom. In mechanical elements the correspondence is established through energy arguments.

§6.6.6. Element Constitutive Properties

For a mechanical element these are relations that specify the material behavior. For example, in a linear elastic bar element it is sufficient to specify the elastic modulus E and the thermal coefficient of expansion α.

[11] In dynamic analysis, time may appear as an additional dimension.

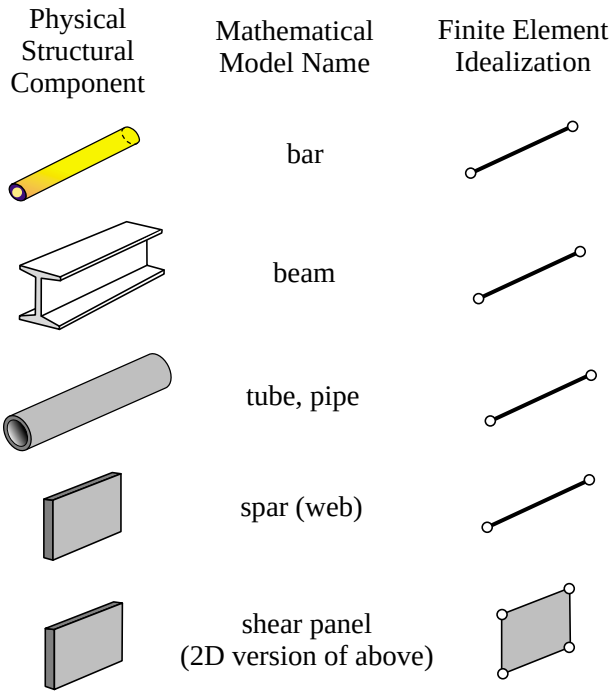


FIGURE 6.4. Examples of primitive structural elements.

§6.6.7. Element Fabrication Properties

For mechanical elements these are fabrication properties which have been integrated out from the element dimensionality. Examples are cross sectional properties of MoM elements such as bars, beams and shafts, as well as the thickness of a plate or shell element.

For computer implementation the foregoing data sets are organized into data structures. These are used by element generation modules to compute element stiffness relations in the local system.

§6.7. Classification of Mechanical Elements

The following classification of finite elements in structural mechanics is loosely based on the "closeness" of the element with respect to the original physical structure. It is given here because it clarifies points that recur in subsequent sections, as well as providing insight into advanced modeling techniques such as hierarchical breakdown and global-local analysis.

§6.7.1. Primitive Structural Elements

These resemble fabricated structural components. They are often drawn as such; see Figure 6.4. The qualifier *primitive* distinguishes them from macroelements, which is another element class described below. Primitive means that they are not decomposable into simpler elements.

These elements are usually derived from Mechanics-of-Materials simplified theories and are better understood from a physical, rather than mathematical, standpoint. Examples are the elements discussed in Chapter 5: bars, cables, beams, shafts, spars.

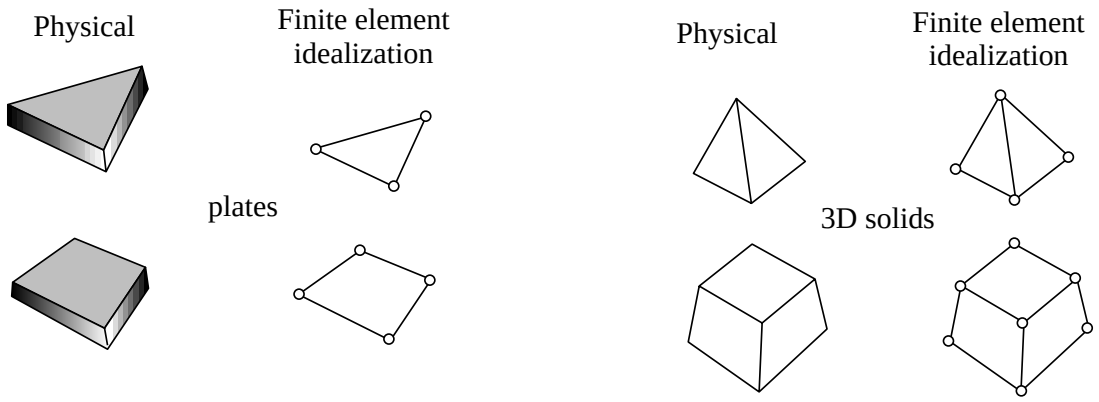


FIGURE 6.5. Continuum element examples.

§6.7.2. Continuum Elements

These do not resemble fabricated structural components at all. They result from the subdivision of "blobs" of continua, or of structural components viewed as continua.

Unlike structural elements, continuum elements are better understood in terms of their mathematical interpretation. Examples: plates, slices, shells, axisymmetric solids, general solids. See Figure 6.5.

§6.7.3. Special Elements

Special elements partake of the characteristics of structural and continuum elements. They are derived from a continuum mechanics standpoint but include features closely related to the physics of the problem. Examples: crack elements for fracture mechanics applications, shear panels, infinite and semi-infinite elements, contact and penalty elements, rigid-body elements. See Figure 6.6.

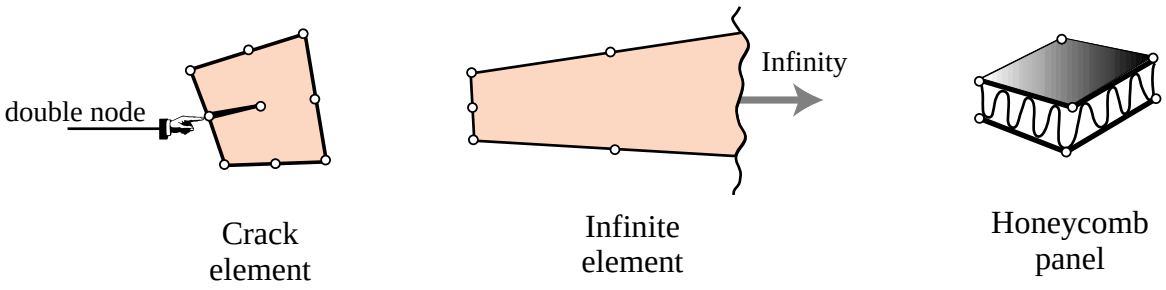


FIGURE 6.6. Special element examples.

§6.7.4. Macroelements

Macroelements are also called mesh units and superelements, although the latter term overlaps with substructures (defined below). These often resemble structural components, but are fabricated with simpler elements. See Figure 6.7.

The main reason for introducing macroelements is to simplify preprocessing tasks. For example, it may be simpler to define a regular 2D mesh using quadrilaterals rather than triangles. The fact that, behind the scene, the quadrilateral is actually a macroelement may not be important to most users.

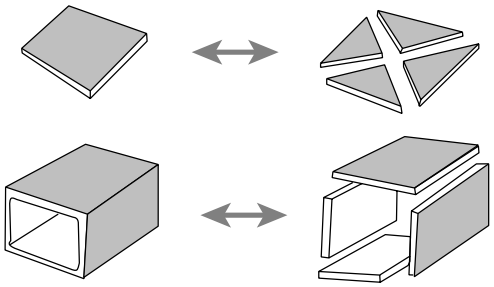


FIGURE 6.7. Macroelement examples.

Similarly a box macroelement can save modeling times for structures that are built by such components; for example box-girder bridges

§6.7.5. Substructures

Also called structural modules and superelements. These are sets of elements with a well defined structural function, typically obtained by cutting the complete structure into functional components. Examples: the wings and fuselage of an airplane; the towers, deck and cables of a suspension bridge.

The distinction between substructures and macroelements is not clear-cut. The main conceptual distinction is that substructures are defined "top down" as parts of a complete structure, whereas macroelements are built "bottom up" from primitive elements. The term *superelement* is often used in a collective sense to embrace element groupings that range from macroelements to substructures. This topic is further covered in Chapter 10.

§6.8. Assembly

The assembly procedure of the Direct Stiffness Method for a general finite element model follows rules identical in principle to those discussed for the truss example. As in that case the processs involves two basic steps:

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

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

The hand calculations for the example truss conceal, however, the implementation complexity. The master stiffness equations in practical applications may involve thousands or even millions of freedoms, and programming can become involved. The topic is elaborated upon in Chapter 25.

§6.9. Boundary Conditions

A key strength of the FEM is the ease and elegance with which it handles arbitrary boundary and interface conditions. This power, however, has a down side. A big hurdle faced by FEM newcomers is the understanding and proper handling of boundary conditions. Below is a simple recipe for treating boundary conditions. The following Chapter provides more specific rules and examples.

§6.9.1. Essential and Natural B.C.

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

Essential BCs directly affect DOFs, and are imposed on the left-hand side vector **u**.

Natural BCs do not directly affect DOFs and are imposed on the right-hand side vector **f**.

The mathematical justification for this distinction requires use of concepts from variational calculus, and is consequently relegated to Part II. For the moment, the basic recipe is:

1. If a boundary condition involves one or more degrees of freedom in a *direct* way, it is essential. An example is a prescribed node displacement.
2. Otherwise it is natural.

The term "direct" is meant to exclude derivatives of the primary function, unless those derivatives also appear as degrees of freedom, such as rotations in beams and plates.

§6.9.2. B.C. in Structural Problems

Essential boundary conditions in mechanical problems involve *displacements* (but not strain-type displacement derivatives). Support conditions for a building or bridge problem furnish a particularly simple example. But there are more general boundary conditions that occur in practice. A structural engineer must be familiar with displacement B.C. of the following types.

Ground or support constraints. Directly restraint the structure against rigid body motions.

Symmetry conditions. To impose symmetry or antisymmetry restraints at certain points, lines or planes of structural symmetry. This allows the discretization to proceed only over part of the structure with a consequent savings in modeling effort and number of equations to be solved.

Ignorable freedoms. To suppress displacements that are irrelevant to the problem.[12] Even experienced users of finite element programs are sometimes baffled by this kind. An example are rotational degrees of freedom normal to smooth shell surfaces.

Connection constraints. To provide connectivity to adjoining structures or substructures, or to specify relations between degrees of freedom. Many conditions of this type can be subsumed under the label *multipoint constraints* or *multifreedom constraints*. These can be notoriously difficult to handle from a numerical standpoint, and are covered in Chapters 8–9.

[12] In classical dynamics these are called *ignorable coordinates*.

Notes and Bibliography

Most FEM textbooks do not provide a systematic treatment of modeling. This is no accident: few academic authors have experience with complex engineering systems. Good engineers are too busy (and in demand) to have time for writing books. This gap has been particularly acute since FEM came on the scene because of generational gaps: "real engineers" tend to mistrust the computer, and often for good reason. The notion of explicit versus implicit modeling, which has legal and professional implications, is rarely mentioned.

FEM terminology is by now standard, and so is a majority of the notation. But that is not so in early publications. E.g. **K** is universally used[13] for stiffness matrix in virtually all post-1960 books. There are a few exceptions: the often cited book of Przemieniecki [575] uses **S**. There is less unanimity on **u** and **f** for node displacement and force vectors, respectively; some books such as Zienkiewicz and Taylor [794] still use different symbols.

The element classification given here attempts to systematize dispersed references. In particular, the distinction between macroelements, substructures and superelements is an ongoing source of confusion, particularly since massively parallel computation popularized the notion of "domain decomposition" in the computer science community. The all-encompassing term "superelement" emerged in Norway by 1968 as part of the implementation of the computer program SESAM. Additional historical details are provided in Chapter 10.

The topic of BC classification and handling is a crucial one in practice. More modeling mistakes are done in this aspect of FEM application than anywhere else.

References

Referenced items have been moved to Appendix R.

[13] A symbol derived from the "spring constant" k that measures the stiffness of a mechanical spring.

7

FEM Modeling: Mesh, Loads and BCs

TABLE OF CONTENTS

§7.1. Introduction

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

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

§7.2. General Recommendations

The general rules that should guide you in the use of commercial or public FEM packages, are[1]

- Use the *simplest* type of finite element that will do the job.
- *Never, never, never* mess around with complicated or special elements, unless you are *absolutely sure* of what you are doing.
- Use the *coarsest mesh* you think will capture the dominant physical behavior of the physical system, particularly in *design* applications.

Three word summary: *keep it simple*. Initial FE models may have to be substantially revised to accommodate design changes. There is little point in using complicated models that will not survive design iterations. The time for refinement is when the design has stabilized and you have a better picture of the underlying physics, possibly reinforced by experiments or observation.

§7.3. Guidelines on Element Layout

The following guidelines are stated for structural applications. As noted above, they will be often illustrated for two-dimensional meshes of continuum elements for ease of visualization.

§7.3.1. Mesh Refinement

Use a relatively fine (coarse) discretization in regions where you expect a high (low) *gradient* of strains and/or stresses.[2] Regions to watch out for high gradients are:

- Near entrant corners (see Remark below) or sharply curved edges.
- In the vicinity of concentrated (point) loads, concentrated reactions, cracks and cutouts.
- In the interior of structures with abrupt changes in thickness, material properties or cross sectional areas.

The examples in Figure 7.1 illustrate some of these "danger regions." Away from such regions one can use a fairly coarse discretization within constraints imposed by the need of representing the structural geometry, loading and support conditions reasonably well.

[1] Paraphrasing the Bellman in The Hunting of the Snark: "what I say three times is true."

[2] *Gradient* is the key word. High gradient means rapid variation. A high value by itself means nothing in this context.

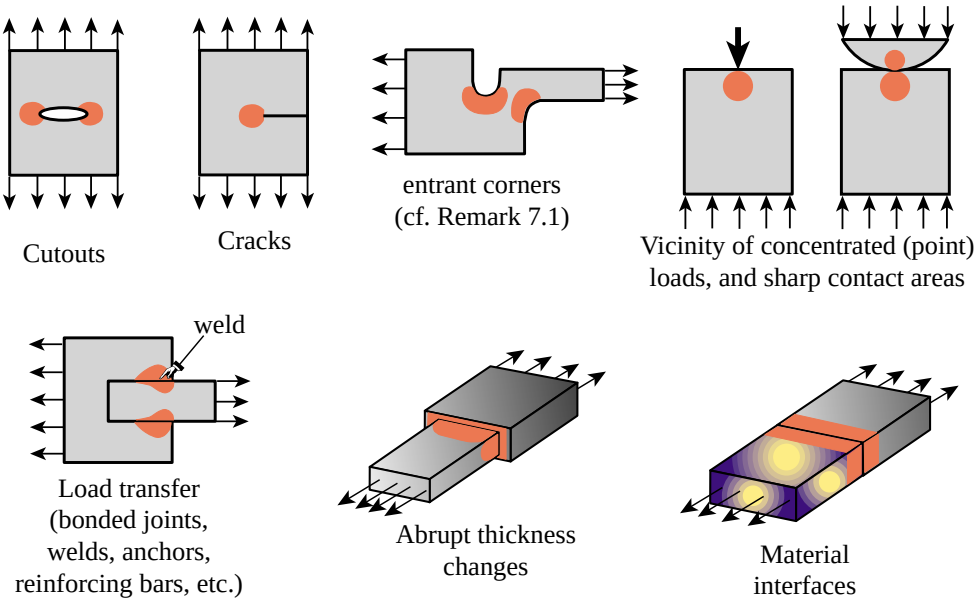


FIGURE 7.1. Some situations where a locally refined finite element discretization (in the red-shaded areas) is recommended.

Remark 7.1. The first bullet above mentions "entrant corner." That is a region where isostatics (principal stress trajectories) "bunch up." For a two-dimensional problem mathematically posed on a singly-connected interior domain, they can be recognized as follows. Traverse the boundary CCW so the body or structure is on your left. When hitting a sharp or rounded corner, look at the angle (positive CCW) formed by the *exterior* normals (the normal going toward your right) before and after, measured *from* before *to* after, and always taking the *positive* value.

If the angle exceeds 180°, it is an entrant corner. [If it is close to 360°, it is a crack tip.] For exterior problems or multiple-connected domains, the definition must be appropriately adjusted.

§7.3.2. Element Aspect Ratios

When discretizing two and three dimensional problems, try to avoid finite elements of high aspect ratios: elongated or "skinny" elements, such as the ones illustrated on the right of Figure 7.2. (The aspect ratio of a two- or three-dimensional element is the ratio between its largest and smallest dimension.)

As a rough guideline, elements with aspect ratios exceeding 3 should be viewed with caution and those exceeding 10 with alarm. Such elements will not necessarily produce bad results — that depends on the loading and boundary conditions of the problem — but do introduce the potential for trouble.

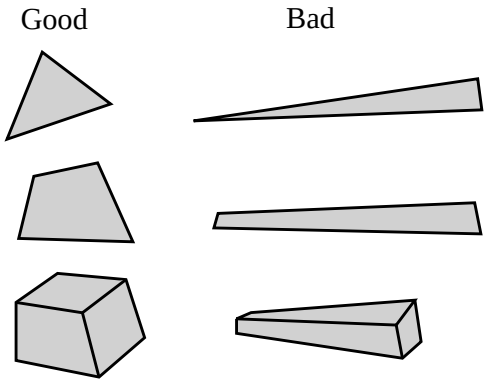


FIGURE 7.2. Elements with good and bad aspect ratios.

Remark 7.2. In many "thin" structures modeled as continuous bodies the appearance of "skinny" elements is inevitable on account of computational economy reasons. An example is provided by the three-dimensional modeling of layered composites in aerospace and mechanical engineering problems.

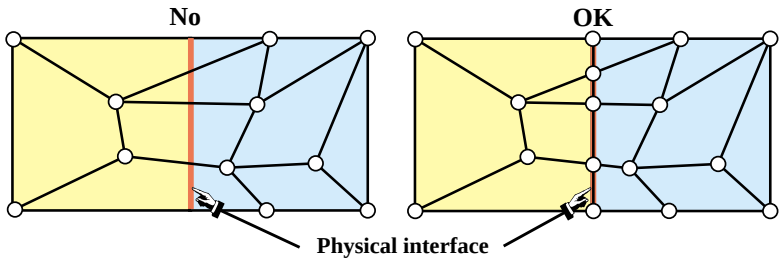


FIGURE 7.3. llustration of the rule that elements should not cross material interfaces.

§7.3.3. Physical Interfaces

A physical interface, resulting from example from a change in material, should also be an interelement boundary. That is, *elements must not cross interfaces*. See Figure 7.3.

§7.3.4. Preferred Shapes

In 2D FE modeling, if you have a choice between triangles and quadrilaterals with similar nodal arrangement, prefer quadrilaterals. Triangles are quite convenient for mesh generation, mesh transitions, rounding up corners, and the like. But sometimes triangles can be avoided altogether with some thought. One of the homework exercises is oriented along these lines.

In 3D FE modeling, prefer strongly bricks over wedges, and wedges over tetrahedra. The latter should be used only if there is no viable alternative.[3] The main problem with tetrahedra and wedges is that they can produce wrong stress results even if the displacement solution looks reasonable.

§7.4. Direct Lumping of Distributed Loads

In practical structural problems, distributed loads are more common than concentrated (point) loads.[4] Distributed loads may be of surface or volume type.

Distributed surface loads (called surface tractions in continuum mechanics) are associated with actions such as wind or water pressure, snow weight on roofs, lift in airplanes, live loads on bridges, and the like. They are measured in force per unit area.

Volume loads (called body forces in continuum mechanics) are associated with own weight (gravity), inertial, centrifugal, thermal, prestress or electromagnetic effects. They are measured in force per unit volume.

A derived type: line loads, result from the integration of surface loads along one transverse direction, such as a beam or plate thickness, or of volume loads along two transverse directions, such as a bar or beam area. Line loads are measured in force per unit length.

Whatever their nature or source, distributed loads *must be converted to consistent nodal forces* for FEM analysis. These forces end up in the right-hand side of the master stiffness equations.

---

[3] Unfortunately, many existing space-filling automatic mesh generators in three dimensions produce tetrahedral meshes. There are generators that try to produce bricks, but these often fail in geometrically complicated regions.

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

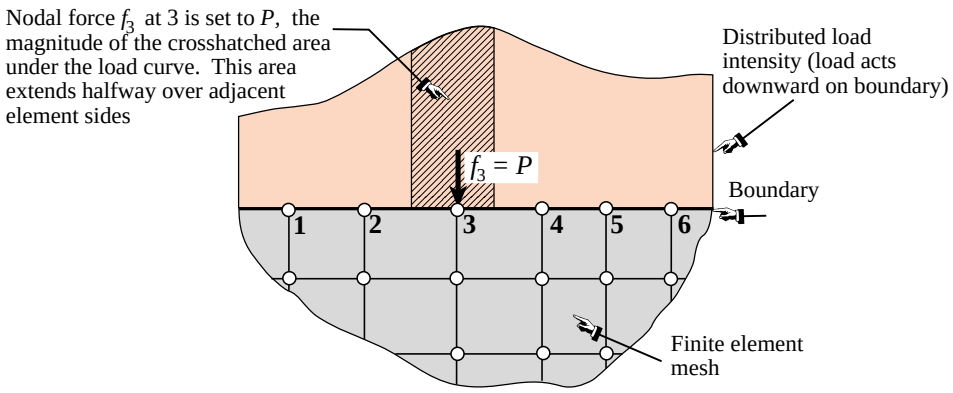


FIGURE 7.4. NbN direct lumping of distributed line load, illustrated for a 2D problem.

The meaning of "consistent" can be made precise through variational arguments, by requiring that the distributed loads and the nodal forces produce the same external work. Since this requires the introduction of external work functionals, the topic is deferred to Part II. However, a simpler approach called *direct load lumping*, or simply *load lumping*, is often used by structural engineers in lieu of the mathematically impeccable but complicated variational approach. Two variants of this technique are described below for distributed surface loads.

§7.4.1. Node by Node Lumping

The node by node (NbN) lumping method is graphically explained in Figure 7.4. This example shows a distributed surface loading acting normal to the straight boundary of a two-dimensional FE mesh. (The load is assumed to have been integrated through the thickness normal to the figure, so it is actually a *line load* measured as force per unit length.)

The procedure is also called *tributary region* or *contributing region* method. For the example of Figure 7.4, each boundary node is assigned a *tributary region* around it that extends halfway to adjacent nodes. The force contribution P of the cross-hatched area is directly assigned to node 3.

This method has the advantage of not requiring the error-prone computation of centroids, as needed in the EbE technique discussed below. For this reason it is often preferred in hand computations.[5] It can be extended to three-dimensional meshes as well as volume loads.[6] It should be avoided, however, when the applied forces vary rapidly (within element length scales) or act only over portions of the tributary regions.

§7.4.2. Element by Element Lumping

In this variant the distributed loads are divided over element domains. The resultant load is assigned to the centroid of the load diagram, and apportioned to the element nodes by statics. A node force is obtained by adding the contributions from all elements meeting at that node. The procedure is illustrated in Figure 7.5, which shows details of the computation over 2–3. The total force at node 3, for instance, would be that contributed by segments 2–3 and 3–4. For the frequent case in which

[5] It has been extensively used in the aircraft industry for smooth-varying pressure loads computations.

[6] The computation of tributary areas and volumes for general 2D and 3D regions can be done through the so-called Voronoi diagrams. This is an advanced topic in computational geometry (see, e.g., [194]) and thus not treated here.

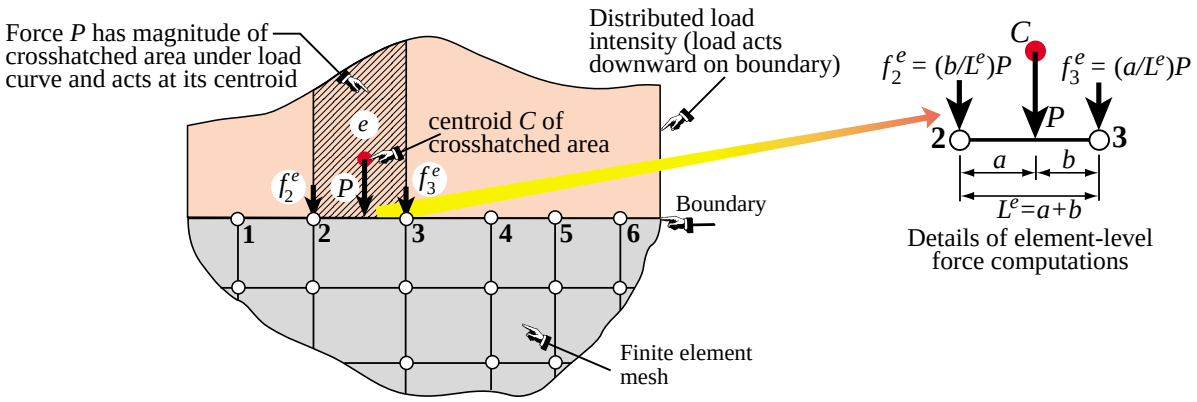


FIGURE 7.5. EbE direct lumping of distributed line load, illustrated for a 2D problem.

the variation of the load over the element is linear (so the area under the load is a trapezoid) the node forces can be computed directly by the formulas given in Figure 7.6.

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

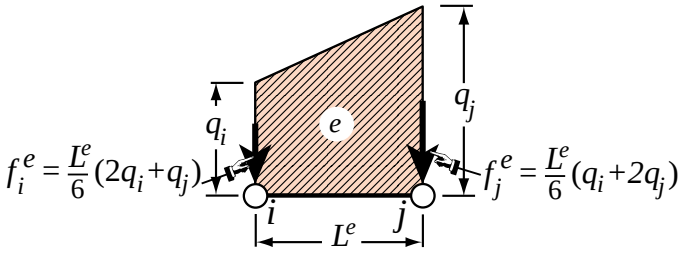


FIGURE 7.6. EbE lumping of linearly varying line load over element.

The EbE procedure is not applicable if the centroidal resultant load cannot be apportioned by statics. This happens if the element has midside faces or internal nodes in addition to corner nodes, or if it has rotational degrees of freedom. For those elements the variational-based consistent approach covered in Part II and briefly outlined in §7.3.3, is preferable.

Example 7.1. Figures 7.7(a,b) show web-downloaded pictures of the Norfork Dam, a 220-ft high, concrete-built gravity dam. Its typical cross section is shown in 7.7(c). The section is discretized by triangular elements as illustrated in Figure 7.7(d). The dam has a length of 2624 ft and was constructed over the White River in Arkansas over 1941–44.[7] The structure is assumed to be in plane strain. Accordingly the FEM model shown in 7.7(d) is a typical 1-ft slice of the dam and near-field soil.

In the analysis of dam and marine structures, a "wet node" of a FEM discretization is one in contact with the water. The effect of hydrostatic pressure is applied to the structure through nodal forces on wet nodes. The wet nodes for a water head of 180 ft over the riverbed are shown in Figure 7.8(b). Nodes are numbered 1 through 9 for convenience. The pressure in psf (lbs per sq-ft) is $p = 62.4\,d$ where d is the depth in ft. Compute the horizontal hydrostatic nodal forces f_{x1} and f_{x2} using NbN and EbE, assuming that the wet face AB is vertical for simplicity.

Node by Node. For node 1 go halfway to 2, a distance of $(180 - 136)/2 = 22$ ft. The tributary load area is a triangle extending 22 ft along y with bottom pressure $62.4 \times 22 = 1372.8$ psf and unit width normal to

[7] As discussed in **Notes and Bibliography**, this example has historical significance, as the first realistic Civil Engineering structure modeled ca. 1960 by the Finite Element Method, which until then had been largely confined to aerospace.

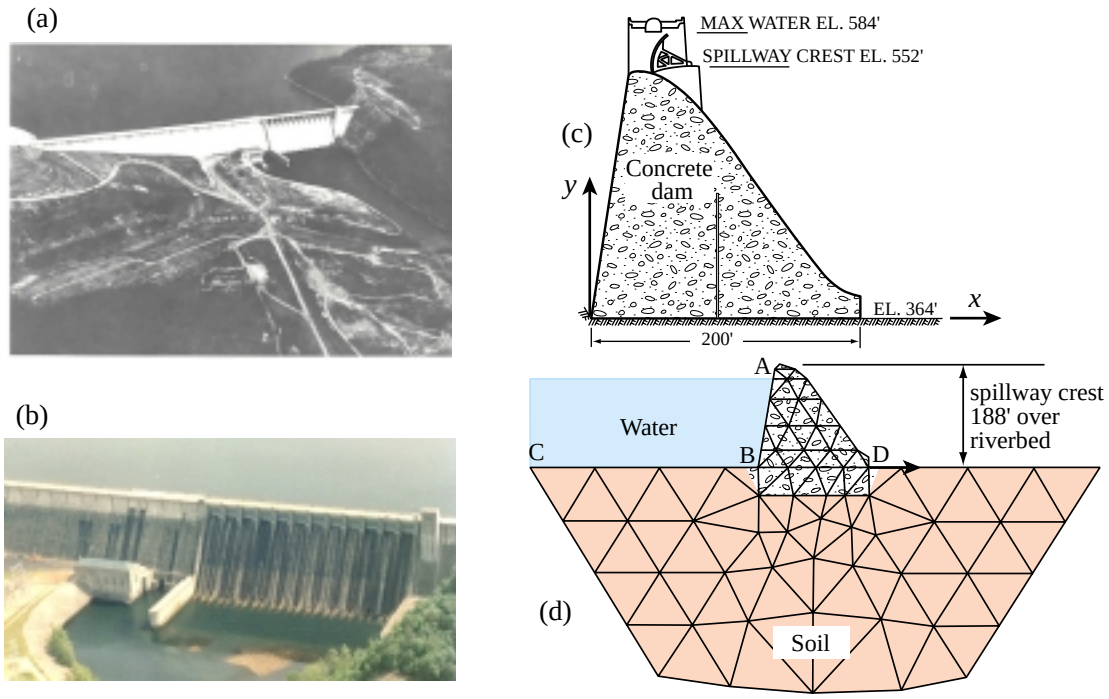


FIGURE 7.7. Norfork Dam: (a,b) pictures; (c) cross section of dam above foundation (line inside dam is a thermally induced crack considered in the 1960 study); (d) coarse mesh including foundation and soil but not crack [137].

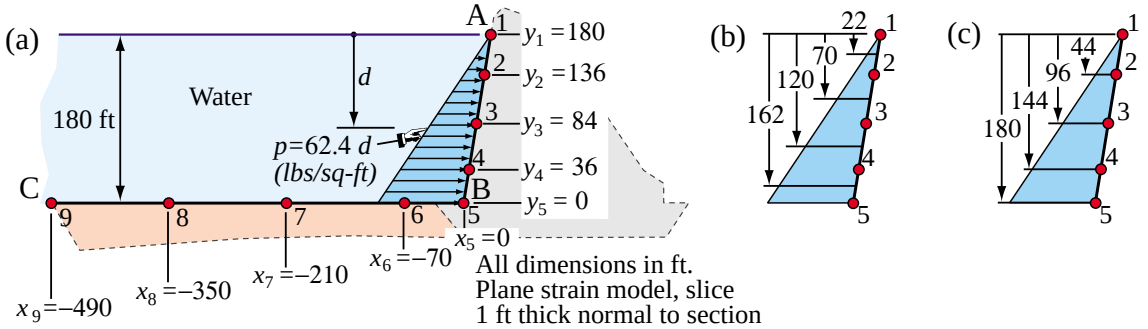


FIGURE 7.8. Norfork Dam example: (a) computation of wet node forces due to hydrostatic pressure; (b) NbN tributary regions; (c) EbE regions.

paper, giving $f_{x1} = \frac{1}{2}22 \times 1372.8 = 15101$ lbs. For node 2 go up 22 ft and down $(136-84)/2 = 26$ ft. The tributary load area is a trapezoid extending $22+26 = 48$ ft vertically, with pressures 1372.8 psf at the top and $62.4 \times 70 = 4368.0$ psf at the bottom. This gives $f_{x2} = 48 \times (1372.8 + 4368.0)/2 = 137779$ lbs.

Element by Element. It is convenient to pre-compute hydro pressures at wet node levels: $p_1 = 0$, $p_2 = 62.4 \times 44 = 2745.6$ psf, $p_3 = 62.4 \times 96 = 5990.4$. Because the variation of p is linear, the formulas of Figure 7.6 can be applied directly: $f_{x1}^{(1)} = (44/6)(2 \times 0 + 2745.6) = 20134$ lbs, $f_{x2}^{(1)} = (44/6)(0 + 2 \times 2745.6) = 40269$ lbs and $f_{x2}^{(2)} = (52/6)(2 \times 2745.6 + 5990.4) = 99507$ lbs. Adding contributions to nodes 1 and 2: $f_{x1} = f_{x1}^{(1)} = 20134$ lbs and $f_{x2} = f_{x2}^{(1)} + f_{x2}^{(2)} = 40269 + 99507 = 139776$ lbs.

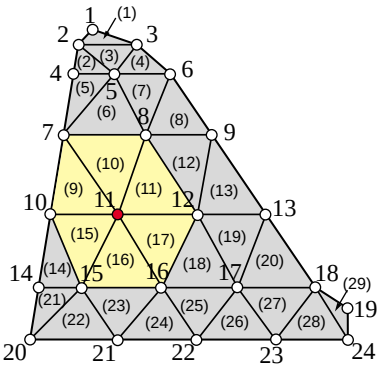
The computations for wet nodes 3 through 9 are left as an exercise.

Example 7.2. Figure 7.9 shows the mesh for the $y > 0$ portion of the gravity dam of the previous example.

(The node numbering is different). The specific weight for concrete of $\gamma = 200$ pcf (pounds per cubic foot). Compute the node force f_{y11} due to own weight.

For this calculation NbN is unwieldy because computation of the nodal tributary region requires construction of a Voronoi diagram. (Furthermore for constant specific weight it gives the same answer as EbE.) To apply EbE, select the elements that contribute to 11: the six triangles (9), (10), (11), (15), (16) and (17).

The area of a triangle with corners at $\{x_1, y_1\}$, $\{x_2, y_2\}$ and $\{x_3, y_3\}\}$ is given by $A = (x_2y_3 - x_3y_2) + (x_3y_1 - x_1y_3) + (x_1y_2 - x_2y_1)$. Applying this to the geometry of the figure one finds that the areas are $A^{(9)} =$, $A^{(10)} =$, $A^{(11)} =$, $A^{(15)} =$, $A^{(16)} =$, and $A^{(17)} =$. The weight forces on each element are $W^{(9)} = \gamma h$, $W^{(10)} = \gamma h$, $W^{(11)} = \gamma h$, $A^{(15)} = \gamma h$, $A^{(16)} = \gamma h$, and $A^{(17)} = \gamma h$, where h is the thickness normal to the figure (1 ft here).



Node	x	y
7	*.00	152.00
8	*.00	152.00
9	*.00	152.00
10	*.00	94.00
11	*.00	94.00
12	*.00	94.00
13	*.00	94.00
14	*.00	40.00
15	*.00	40.00
16	*.00	40.00
17	*.00	40.00
18	*.00	40.00

FIGURE 7.9. Computation of weight force at node 11 of gravity dam example of Figure 7.7(d).

For uniform element thickness and specific weight, one third of each force goes to each element corner. Thus node 11 receives

$$f_{y11} = \tfrac{1}{3}(W^{(9)} + W^{(10)} + W^{(11)} + W^{(17)} + W^{(18)}) = \tag{7.1}$$

(example incomplete, TBF)

§7.4.3. *Weighted Lumping

The NbN and EbE methods are restricted to simple elements, specifically those with corner nodes only. We outline here a general method that works for more complicated models. The mathematical justification requires energy theorems covered in Part II. Thus at this stage the technique is merely presented as a recipe.

To fix the ideas consider again the 2D situation depicted in Figures 7.4 and 7.5. Denote the distributed load by $q(x)$. We want to find the lumped force f_n at an interior node n of coordinate x_n. Let the adjacent nodes be $n-1$ and $n+1$, with coordinates x_{n-1} and x_{n+1}, respectively. Introduce a *weight function* $W_n(x)$ with properties to be specified below. The lumped force is given by

$$f_n = \int_{x_{n-1}}^{x_{n+1}} W_n(x)\, q(x)\, dx \tag{7.2}$$

For this formula to make sense, the weight function must satisfy several properties:

1. Unit value at node n: $W_n(x_n) = 1$.
2. Vanishes over any element not pertaining to n: $W_n = 0$ if $x \le x_{n-1}$ or $x \ge x_{n+1}$
3. Gives the same results as NbN or EbE for constant q over elements with corner nodes only.

Both NbN and EbE are special cases of (7.2). For NbN pick

$$W_n(x) = 1 \quad \text{if} \quad \tfrac{1}{2}(x_{n-1} + x_n) \le x \le \tfrac{1}{2}(x_n + x_{n+1}), \quad w_n(x) = 0 \ \text{ otherwise.} \tag{7.3}$$

For EbE pick

$$W_n(x) = \begin{cases} 1 - \dfrac{x - x_{n-1}}{x_n - x_{n-1}} & \text{if } x_{n-1} \le x \le x_n, \\ 1 - \dfrac{x - x_n}{x_{n+1} - x_n} & \text{if } x_n \le x \le x_{n+1}, \\ 0 & \text{otherwise.} \end{cases} \tag{7.4}$$

These particular weight functions are depicted in Figure 7.10.

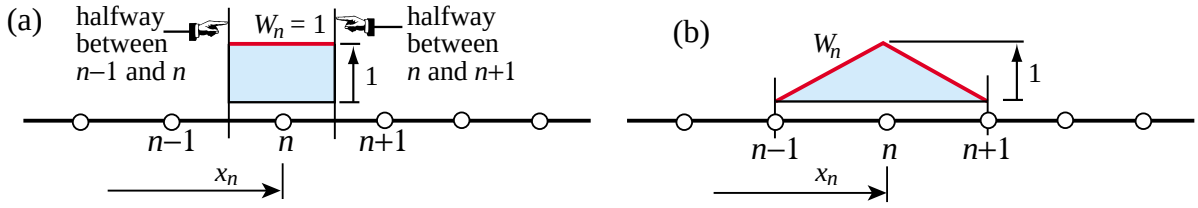


FIGURE 7.10. Weight functions corresponding to: (a) NbN lumping, and (b) EbE lumping.

§7.4.4. *Energy Consistent Lumping

The rule (7.2) can be justified from the standpoint of the Principle of Virtual Work. Let δu_n be a virtual node displacement paired to f_n. Take $\delta u_n W_n(x)$ to be the associated displacement variation. The external virtual work of $q(x)$ is $\int q(x) W_n(x)\, \delta u_n\, dx$ extended over the portion where $W_n \neq 0$. Equating this to $f_n\, \delta u_n$ and cancelling δu_n from both sides yields (7.2).

If W_n is the *trial displacement function* actually used for the development of element equations the lumping is called *energy consistent*, or *consistent* for short.

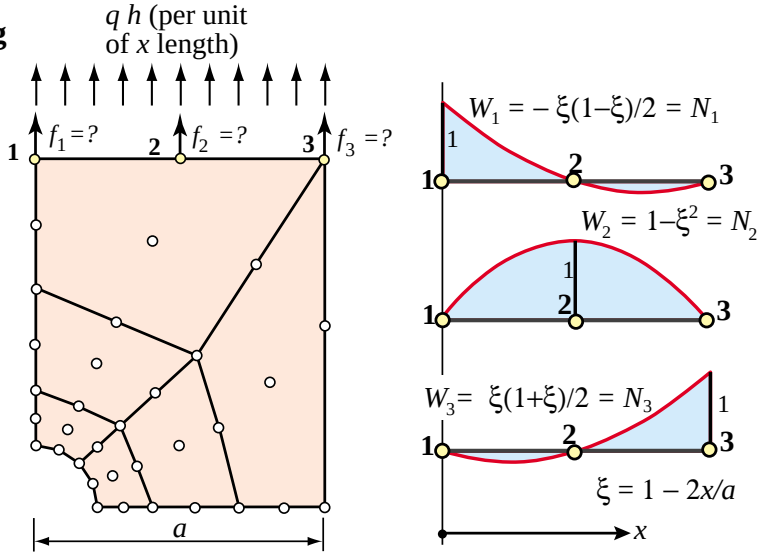


FIGURE 7.11. Example of consistent load lumping.

(As will be seen later, trial functions are the union of shape functions over the patch of all elements connected to node n.) This important technique is studied in Part II of the course after energy methods are introduced.

Example 7.3. Conside the mesh of 9-node quadrilaterals shown in Figure 7.11. (This is later used as a benchmark problem in Chapter 27.) The upper plate edge 1–3 is subject to a uniform normal load qh per unit length, where h is the plate thickness. The problem is to compute the node forces f_1, f_2 and f_3. If 1–2 and 2–3 were on two different elements, both NbN and EbE would give $f_1 = f_3 = \frac{1}{4}qha$ and $f_2 = \frac{1}{2}qha$. But this lumping is wrong for an element with midside nodes. Instead, pick the weight functions W_i $(i = 1, 2, 3)$ shown on the right of Figure 7.11.

Since there is only one element on the loaded edge, the W_i are actually the quadratic shape functions N_i for a 3-node line element, developed in later Chapters. The dimensionless variable ξ is called an *isoparametric natural coordinate*. Applying the rule (7.2) we get

$$f_1 = \int_0^a W_1(x)\, q\, h\, dx = \int_{-1}^{1} W_1(\xi)\, q\, h\, \frac{dx}{d\xi}\, d\xi = \int_{-1}^{1} -\tfrac{1}{2}\xi(1-\xi) q\, h\, (\tfrac{1}{2}a)\, d\xi = \tfrac{1}{6}qha. \qquad (7.5)$$

Similarly $f_2 = \frac{2}{3}qha$ and $f_3 = f_1$. As a check, $f_1 + f_2 + f_3 = qha$, which is the total load acting on the plate edge.

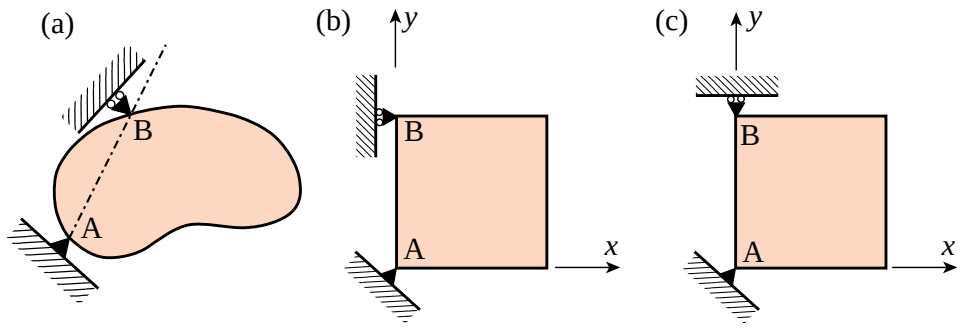


FIGURE 7.12. Suppressing two-dimensional rigid body motions.

§7.5. Boundary Conditions

The key distinction between *essential* and *natural* boundary conditions (BC) was introduced in the previous Chapter. The distinction is explained in Part II from a variational standpoint. In this section we discuss the simplest *essential* boundary conditions in structural mechanics from a physical standpoint. This makes them relevant to problems with which a structural engineer is familiar. Because of the informal setting, the ensuing discussion relies heavily on examples.

In structural problems formulated by the DSM, the recipe of §6.7.1 that distinguishes between essential and natural BC is: if it directly involves the nodal freedoms, such as displacements or rotations, it is essential. Otherwise it is natural. Conditions involving applied loads are natural. Essential BCs take precedence over natural BCs.

The simplest essential boundary conditions are support and symmetry conditions. These appear in many practical problems. More exotic types, such as multifreedom constraints, require more advanced mathematical tools and are covered in the next two Chapters.

§7.6. Support Conditions

Supports are used to restrain structures against relative rigid body motions. This is done by attaching them to Earth ground (through foundations, anchors or similar devices), or to a "ground structure" which is viewed as the external environment.[8] The resulting boundary conditions are often called *motion constraints*. In what follows we analyze two- and three-dimensional motions separately.

§7.6.1. Supporting Two Dimensional Bodies

Figure 7.12 shows two-dimensional bodies that move in the plane of the paper. If a body is not restrained, an applied load will cause infinite displacements. Regardless of loading conditions, the body must be restrained against two translations along x and y, and one rotation about z. Thus the minimum number of constraints that has to be imposed in two dimensions is *three*.

In Figure 7.12, support A provides *translational* restraint, whereas support B, together with A, provides *rotational* restraint. In finite element terminology, we say that we *delete* (fix, remove, preclude) all translational displacements at point A, and that we delete the translational degree of

[8] For example, the engine of a car is attached to the vehicle frame through mounts. The car frame becomes the "ground structure," which moves with respect to Earth ground, as Earth rotates and moves through space, etc.

freedom directed along the normal to the AB direction at point B. This body is free to distort in any manner without the supports imposing any deformation constraints.

Engineers call A and B *reaction-to-ground points*. This means that if the supports are conceptually removed, applied loads are automatically balanced by reactive forces at A and B, in accordance with Newton's third law. Additional freedoms may be precluded to model greater restraint by the environment. However, Figure 7.12(a) does illustrate the *minimal* number of constraints.

Figure 7.12(b) is a simplification of Figure 7.12(a). Here the line AB is parallel to the global y axis. We simply delete the x and y translations at point A, and the x translation at point B. If the roller support at B is modified as in Figure 7.12(c), however, it becomes ineffective in constraining the infinitesimal rotational motion about point A because the rolling direction is normal to AB. The configuration of Figure 7.12(c) is called a *kinematic mechanism*, and will result in a singular modified stiffness matrix.

§7.6.2. Supporting Three Dimensional Bodies

Figure 7.13 illustrates the extension of the freedom-restraining concept to three dimensions. The minimal number of freedoms that have to be constrained is now *six* and many combinations are possible. In the example of Figure 7.13, all three degrees of freedom at point A have been fixed. This prevents all rigid body translations, and leaves three rotations to be taken care of. The x displacement component at point B is deleted to prevent rotation about z, the z component is deleted at point C to prevent rotation about y, and the y component is deleted at point D to prevent rotation about x.

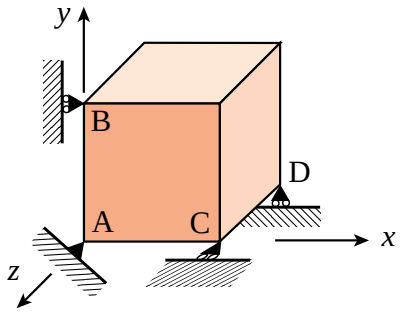


FIGURE 7.13. Suppressing rigid body motions in a three-dimensional body.

§7.7. Symmetry and Antisymmetry Conditions

Engineers doing finite element analysis should be on the lookout for conditions of *symmetry* or *antisymmetry*. Judicious use of these conditions allows only a portion of the structure to be analyzed, with a consequent saving in data preparation and computer processing time.[9]

§7.7.1. Symmetry and Antisymmetry Visualization

Recognition of symmetry and antisymmetry conditions can be done by either visualization of the displacement field, or by imagining certain rotational 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 7.14(a). Alternatively, a 180° rotation of the body about the symmetry line reproduces exactly the original problem.

An *antisymmetry line* can be recognized by the displacement pattern illustrated in Figure 7.14(b).

[9] Even if symmetry or antisymmetry are not explicitly applied through boundary conditions, they provide valuable checks on the computed solution.

Alternatively, a 180° rotation of the body about the antisymmetry line reproduces exactly the original problem except that all applied loads are reversed.

Similar recognition patterns can be drawn in three dimensions to help visualization of *planes* of symmetry or antisymmetry. More complex regular patterns associated with *sectorial* symmetry (also called *harmonic* symmetry) as well as *rotational symmetry* can be treated in a similar manner, but will not be discussed here.

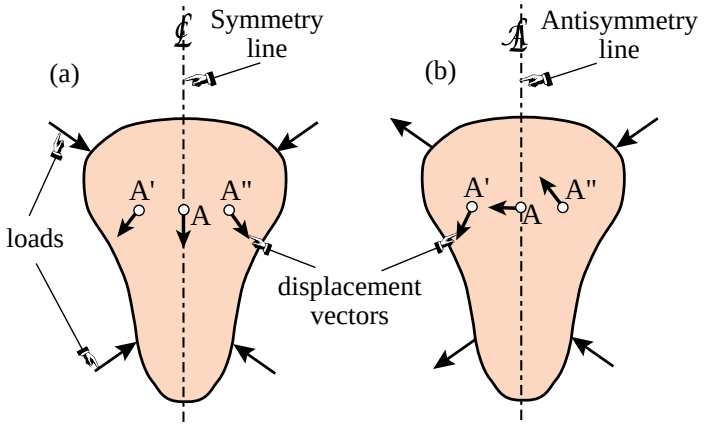


FIGURE 7.14. Visualizing symmetry and antisymmetry lines.

§7.7.2. Effect of Loading Patterns

Although the structure may look symmetric in shape, it must be kept in mind that model reduction can be used only if the loading conditions are also symmetric or antisymmetric.

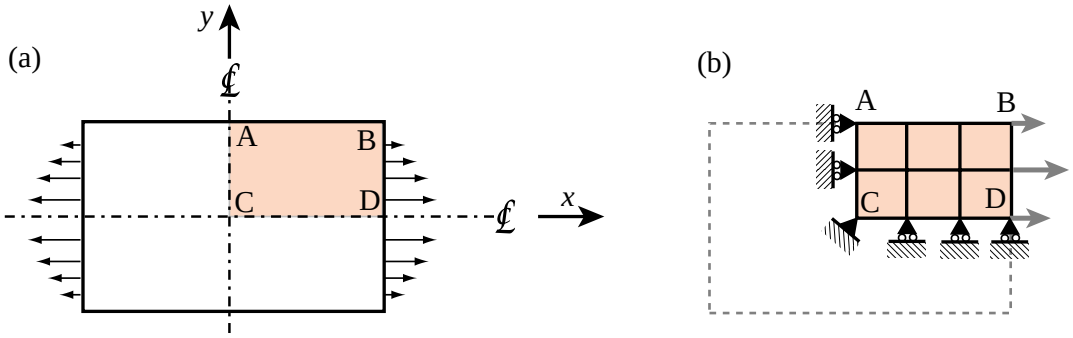


FIGURE 7.15. A doubly symmetric structure under symmetric loading.

Consider the plate structure shown in Figure 7.15(a). This structure is symmetrically loaded on the x-y plane. Applying the recognition patterns stated above one concludes that the structure is *doubly symmetric* in both geometry and loading. It is evident that no displacements in the x-direction are possible for any point on the y-axis, and that no y displacements are possible for points on the x axis. A finite element model of this structure may look like that shown in Figure 7.15(b).

On the other hand if the loading is *antisymmetric*, as illustrated in Figure 7.16(a), the x axis becomes an *antisymmetry line* as none of the $y = 0$ points can move along the x direction. The boundary conditions to be imposed on the FE model are also different, as shown in Figure 7.16(b).

Remark 7.3. For the case shown in Figure 7.16(b) note that all rollers slide in the same direction. Thus the vertical rigid body motion along y is not precluded. To do that, one node has to be constrained in the y direction. If there are no actual physical supports, the choice is arbitrary and amounts only to an adjustment on the overall (rigid-body) vertical motion. In Figure 7.16(b) the center point C has been so chosen. But any other node could be selected as well; for example A or D. The important thing is *not to overconstrain* the structure by applying more than one y constraint.

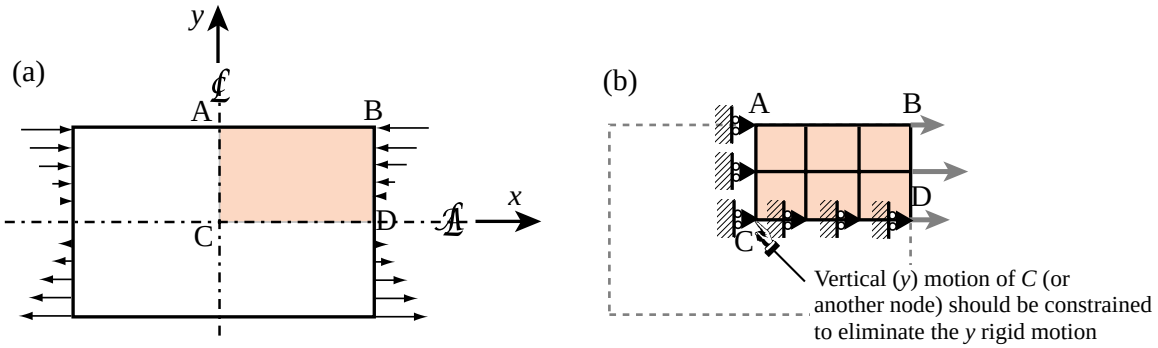


FIGURE 7.16. A doubly symmetric structure under antisymmetric loading.

Remark 7.4. Point loads acting at nodes located on symmetry or antisymmetry lines require special care. For example, consider the doubly symmetric plate structure of Figure 7.15 under the two point loads of magnitude P, as pictured in Figure 7.17(a). If the structure is broken down into 4 quadrants as in Figure 7.17(b), P must be halved as indicated in Figure 7.17(c). The same idea applies to point loads on antisymmetry lines, but there the process is trickier, as illustrated in Figure 7.18. The load must not be applied if the node is fixed against motion, since then the node force will appear as a reaction.

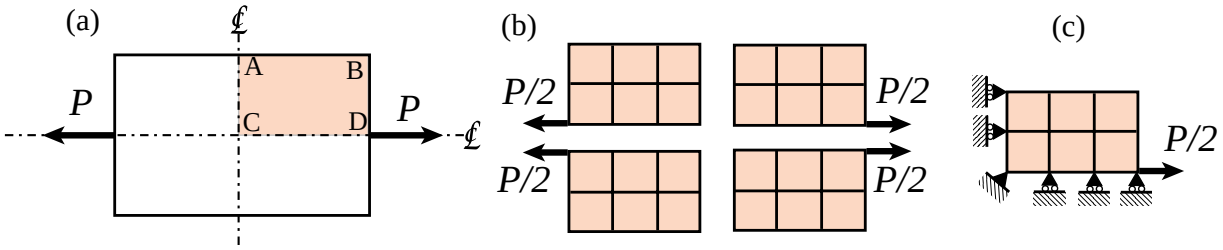


FIGURE 7.17. Breaking up a point load acting on a symmetry line node.

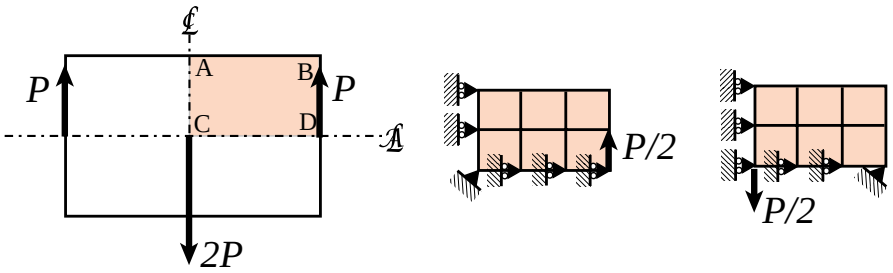


FIGURE 7.18. Breaking up a point load acting on a antisymmetry line node.

Distributed loads should not be divided when the structure is broken down into pieces, since the lumping process will take care of the necessary apportionment to nodes.

Notes and Bibliography

FEM modeling rules in most textbooks are often diffuse, if given at all. In those that focus on the mathematical interpretation of FEM they are altogether lacking; the emphasis being on academic boundary value problems. The rule collection at the start of this Chapter attempts to collect important recommendations in one place.

The treatment of boundary conditions, particularly symmetry and antisymmetry, tends to be also flaky. A notable exception is [397], which is understandable since Irons worked in industry (at Rolls-Royce Aerospace Division) before moving to academia.

The Norfork Dam used in Examples 7.1 and 7.2 (and two Exercises) for hydrostatic load-lumping calculations was the first realistic Civil Engineering structure analyzed by FEM. It greatly contributed to the acceptance of the method beyond the aerospace industry where it had originated. How this seminal event came to pass is narrated by Wilson in [803], from which the following fragment is taken. Annotations are inserted in squared brackets.

> "On the recommendation from Dr. Roy Carlson, a consultant to the Little Rock District of the Corps of Engineers, Clough [then a Professor at UC Berkeley] submitted [in 1960] a proposal to perform a finite element analysis of Norfork Dam, a gravity dam that had a temperature induced vertical crack near the center of the section. The proposal contained a coarse mesh solution that was produced by the new program [a matrix code developed by E. L. Wilson, then a doctoral student under Clough's supervision; the mesh is that shown in Figure 7.7(d)] and clearly indicated the ability of the new method to model structures of arbitrary geometry with different orthotropic properties within the dam and foundation. The finite element proposal was accepted by the Corps over an analog computer proposal submitted by Professor Richard MacNeal of CalTech [who later directed the development of NASTRAN under a NASA contract in the late 1960s], which at that time was considered as the state-of-the-art method for solving such problems.
>
> The Norfork Dam project provided an opportunity to improve the numerical methods used within the program and to extend the finite element method to the nonlinear solution of the crack closing due to hydrostatic loading. Wilson and a new graduate student, Ian King, conducted the detailed analyses that were required by the study. The significant engineering results of the project indicated that the cracked dam was safe [since the crack would be closed as the reservoir was filled]."

References

Referenced items moved to Appendix R.

Homework Exercises for Chapters 6 and 7

FEM Modeling: Mesh, Loads and BCs

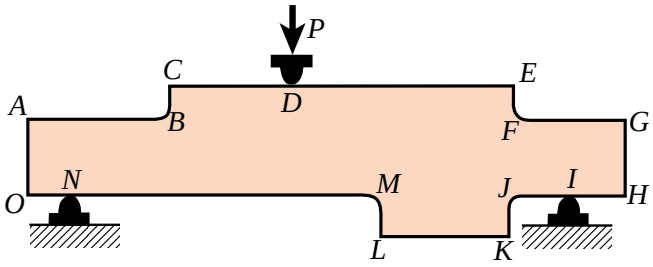


FIGURE E7.1. Inplane bent plate for Exercise 7.1.

EXERCISE 7.1 [D:10] The plate structure shown in Figure E7.1 is loaded and deforms in the plane of the paper. The applied load at D and the supports at I and N extend over a fairly narrow area. List what you think are the likely "trouble spots" that would require a locally finer finite element mesh to capture high stress gradients. Identify those spots by its letter and a reason. For example, D: vicinity of point load.

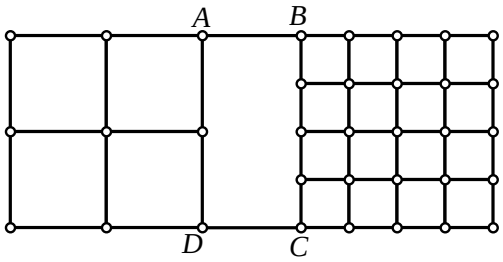


FIGURE E7.2. Transition meshing for Exercise 7.2.

EXERCISE 7.2 [D:15] Part of a two-dimensional FE mesh has been set up as indicated in Figure E7.2. Region $ABCD$ is still unmeshed. Draw a *transition mesh* within that region that correctly merges with the regular grids shown, uses 4-node quadrilateral elements (quadrilaterals with corner nodes only), and *avoids triangles*. *Note*: There are several (equally acceptable) solutions.

EXERCISE 7.3 [A:15] A rectangular plate of constant thickness h and inplane dimensions $8a$ and $6a$ is meshed with 8 rectangular elements as shown in Figure E7.3(a). The plate specific weight is γ and acts along the $-y$ axis direction.

(a) Compute the node forces due to plate weight at nodes 1 through 15, using the NbN method. Obtain the node-tributary regions as sketched in Figure E7.3(b), which shows each element divided by the medians drawn as dashed lines (the tributary region of node 7 is shown in yellow). Partial answer: $f_{y1} = -2a^2\gamma h$. Check that adding up all y forces at the 15 nodes one gets $W = -48a^2\gamma h$.

(b) Repeat the computations using the EbE method. For this, take the total weight force on each element, and assign one quarter to each corner node. (This agrees with consistent energy lumping for 4-node rectangular elements.) Do the results agree with NbN lumping?

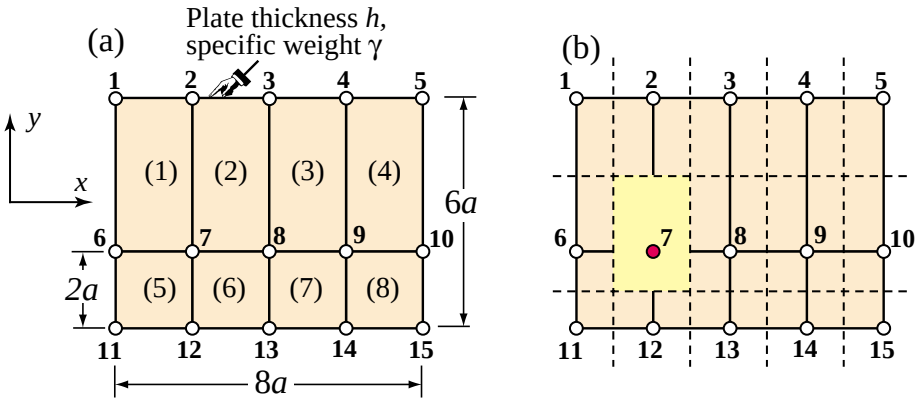


FIGURE E7.3. (a) Mesh layout for Exercise 7.3. (b) shows tributary area for node 7.

EXERCISE 7.4 [N/C:20] Complete the computation of hydrostatic node forces on the Norfork Dam under a water head of 180 ft, initiated in Example 7.1, using the data of Figure 7.8, and either the NbN or EbE method (pick one). Assume face AB is vertical. Do two checks: sum of horizontal (x) forces on nodes 1 through 5 is $\frac{1}{2}180^2 \times 62.4$ lbs, and sum of vertical (y) forces on nodes 5 through 9 is $-180 \times 62.4 \times 490$ lbs.

EXERCISE 7.5 [N/C:25] Complete the computation of own weight nodal forces of the coarse mesh of the Norfork dam proper, initiated in Example 7.2, reusing the data of Figure 7.9. Use the EbE method. Recover coordinates, and write a computer program to compute node forces. Compute the total weight of the dam slice in lbs.

EXERCISE 7.6 [A:10] Figure E7.4 depicts two instances of a pull test. In (a) a stiffer material (steel rod) is pulled out of a softer one (concrete block); in this case the load transfer diagram shows a rapid variation near the inner end of the rod. In (b) a softer material (plastic rod) is pulled out of a stiffer one (concrete rod), and the load transfer diagram is reversed.

If the test is to be simulated by a finite element model, indicate for (a) and (b) where a finer mesh would be desirable. Explain.

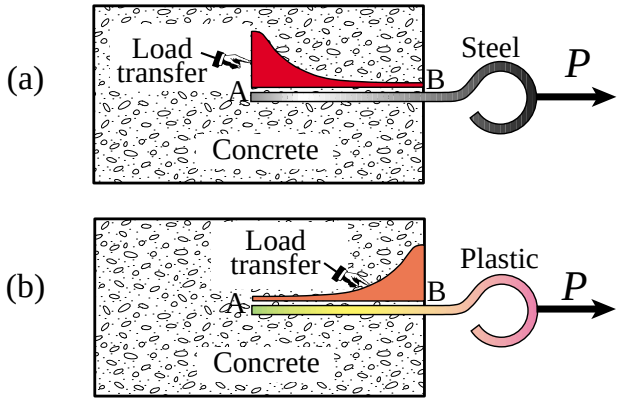


FIGURE E7.4. Pull tests for Exercise 7.6.

EXERCISE 7.7 [D:20] Identify the symmetry and antisymmetry lines in the two-dimensional problems illustrated in Figure E7.5. They are: (a) a circular disk under two diametrically opposite point forces (the famous "Brazilian test" for concrete); (b) the same disk under two diametrically opposite force pairs; (c) a clamped semiannulus under a force pair oriented as shown; (d) a stretched rectangular plate with a central circular hole. Finally (e) and (f) are half-planes under concentrated loads.[10]

Having identified those symmetry/antisymmetry lines, state whether it is possible to cut the complete structure to one half or one quarter before laying out a finite element mesh. Then draw a coarse FE mesh indicating, with

[10] Note that (e) is the famous Flamant's problem, which is important in the 2D design of foundations of civil structures. The analytical solution of (e) and (f) may be found, for instance, in Timoshenko-Goodier's *Theory of Elasticity*, 2nd Edition, page 85ff.

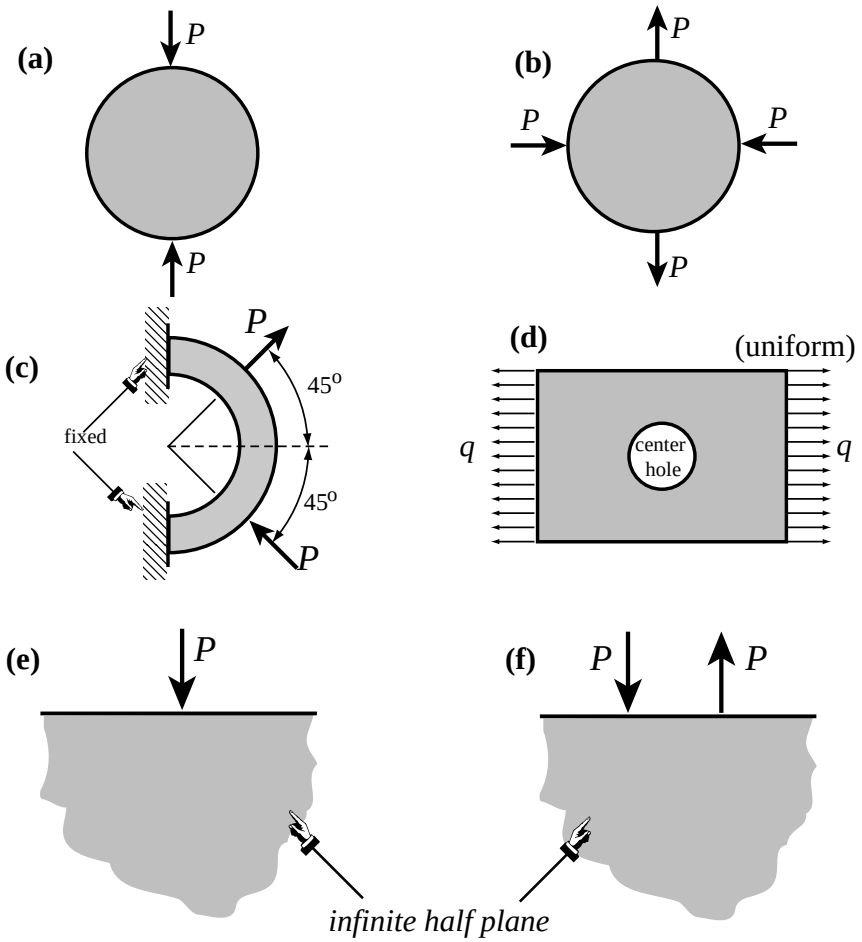


FIGURE E7.5. Problems for Exercise 7.7.

rollers or fixed supports, which kind of displacement BCs you would specify on the symmetry or antisymmetry lines. *Note: Do all sketches on your paper, not on the printed figures*.

EXERCISE 7.8 [D:20] You (a finite element guru) pass away and come back to the next life as an intelligent but hungry bird. Looking around, you notice a succulent big worm taking a peek at the weather. You grab one end and pull for dinner; see Figure E7.6.

After a long struggle, however, the worm wins. While hungrily looking for a smaller one your thoughts wonder to FEM and how the worm extraction process might be modeled so you can pull it out more efficiently. Then you wake up to face this homework question. Try your hand at the following "worm modeling" points.

(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.

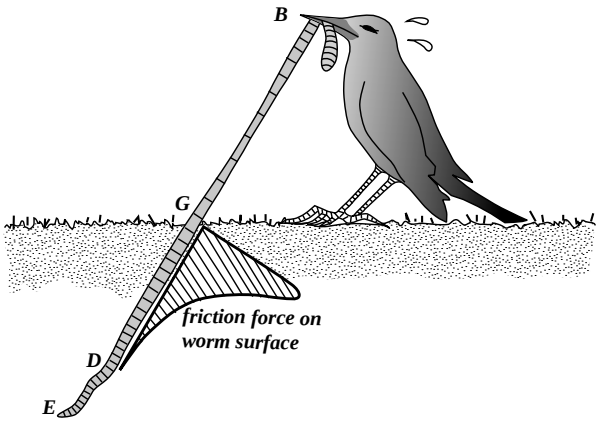


FIGURE E7.6. The hungry bird.

Briefly explain your decisions. Dont write equations.

EXERCISE 7.9 [D:15, 5 each] Prove from kinematics:

(a) Two symmetry lines in 2D cannot cross at a finite point, unless that point is fixed.

(b) Two antisymmetry lines in 2D cannot cross at a finite point, unless that point is fixed.

(c) A symmetry line and an antisymmetry line must cross at right angles, unless the cross point is fixed.

Note: proofs of (a,b,c) are very similar; just draw vectors at alleged intersections.

EXERCISE 7.10 [A/D:15] A 2D body has $n > 1$ symmetry lines passing through a point C and spanning an angle π/n from each other. This is called *sectorial symmetry* if $n \geq 3$. Draw a picture for $n = 5$, say for a car wheel. Explain why point C is fixed.

EXERCISE 7.11 [A/D:25, 5 each] A body is in 3D space. The analogs of symmetry and antisymmetry lines are symmetry and antisymmetry planes, respectively. The former are also called mirror planes.

(a) State the kinematic properties of symmetry and antisymmetric planes, and how they can be identified.

(b) Two symmetry planes intersect. State the kinematic properties of the intersection line.

(c) A symmetry plane and an antisymmetry plane 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 7.12 [A:25] A 2D problem is called *periodic* in the x direction if all fields, in particular displacements, repeat upon moving over a distance $a > 0$: $u_x(x+a, y) = u_x(x, y)$ and $u_y(x+a, y) = u_y(x, y)$. Can this situation be treated by symmetry and/or antisymmetry lines?

EXERCISE 7.13 [A:25] Extend the previous exercise to *antiperiodicity*, in which $u_x(x + a, y) = u_x(x, y)$ and $u_y(x + a, y) = -u_y(x, y)$.

EXERCISE 7.14 [A:20] Prove that EbE and energy consistent lumping agree if the element shape functions are piecewise linear.

EXERCISE 7.15 [A:40] If the world were spatially n-dimensional (meaning it has elliptic metric), how many independent rigid body modes would a body have? (Prove by induction)

8

MultiFreedom Constraints I

TABLE OF CONTENTS

§8.1. Classification of Constraint Conditions

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

$$\boxed{\text{nodal displacement component} = \text{prescribed value.}} \tag{8.1}$$

These are called *single-freedom constraints*. Chapter 3 explains how to incorporate constraints of this form into the master stiffness equations, using hand- or computer-oriented techniques. The displacement boundary conditions studied in Chapter 7, which include modeling of symmetry and antisymmetry, lead to constraints of this form.

For example:

$$u_{x4} = 0, \qquad u_{y9} = 0.6. \tag{8.2}$$

These are two single-freedom constraints. The first one is homogeneous while the second one is non-homogeneous. These attributes are defined below.

§8.1.1. MultiFreedom Constraints

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

$$\boxed{F(\text{nodal displacement components}) = \text{prescribed value},} \tag{8.3}$$

where function F vanishes if all its nodal displacement arguments do. Equation (8.3), in which all displacement components are in the left-hand side, is called the *canonical form* of the constraint.

An MFC of this form is called *multipoint* or *multinode* if it involves displacement components at different nodes. The constraint is called *linear* if all displacement components appear linearly on the left-hand-side, and *nonlinear* otherwise.

The constraint is called *homogeneous* if, upon transfering all terms that depend on displacement components to the left-hand side, the right-hand side — the "prescribed value" in (8.3) —is zero. It is called *non-homogeneous* otherwise.

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

Remark 8.1. The most general constraint class is that of inequality constraints, such as $u_{y5} - 2u_{x2} \geq 0.5$. These constraints are relatively infrequent in linear structural analysis, except in problems that involve contact conditions. They are of paramount importance, however, in other fields such as optimization and control.

Example 8.1. Here are three instances of MFCs:

$$u_{x2} = \tfrac{1}{2}u_{y2}, \quad u_{x2} - 2u_{x4} + u_{x6} = \tfrac{1}{4}, \quad (x_5 + u_{x5} - x_3 - u_{x3})^2 + (y_5 + u_{y5} - y_3 - u_{y3})^2 = 0. \tag{8.4}$$

The first one is linear and homogeneous. It is not a multipoint constraint because it involves the displacement components of one node: 2.

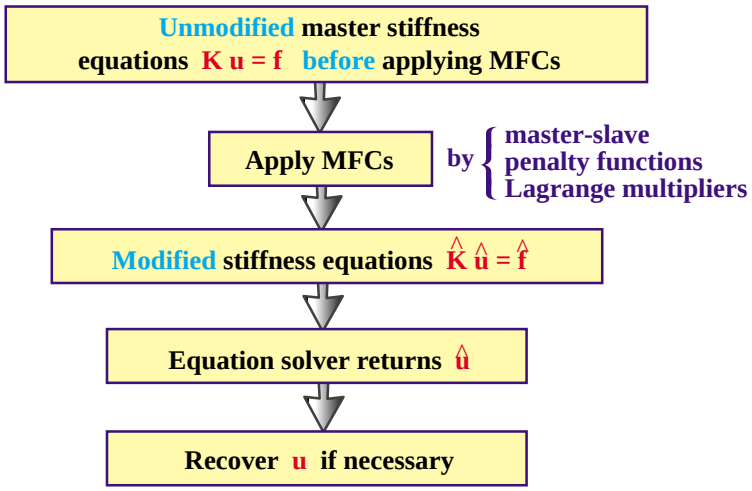


FIGURE 8.1. Schematics of MFC application.

The second one is multipoint because it involves three nodes: 2, 4 and 6. It is linear and non-homogeneous.

The last one is multipoint, nonlinear and homogeneous. Geometrically it expresses that the distance between nodes 3 and 5 in two-dimensional motions on the $\{x, y\}$ plane remains constant. This kind of constraint appears in geometrically nonlinear analysis of structures, which is a topic beyond the scope of this book.

§8.1.2. Methods for Imposing Multifreedom Constraints

Accounting for multifreedom constraints is done — at least conceptually — by changing the assembled master stiffness equations to produce a *modified* system of equations:

$$\mathbf{K}\,\mathbf{u} = \mathbf{f} \quad \overset{\text{MFC}}{\Rightarrow} \quad \hat{\mathbf{K}}\,\hat{\mathbf{u}} = \hat{\mathbf{f}}. \tag{8.5}$$

The modification process (8.5) is also called *constraint application* or *constraint imposition*. The modified system is that submitted to the equation solver, which returns $\hat{\mathbf{u}}$.

The procedure is flowcharted in Figure 8.1. The sequence of operations sketched therein applies to all methods outlined below.

Three methods for treating MFCs are discussed in this and the next Chapter:

1. *Master-Slave Elimination*. The degrees of freedom involved in each MFC are separated into master and slave freedoms. The slave freedoms are then explicitly eliminated. The modified equations do not contain the slave freedoms.
2. *Penalty Augmentation*. Also called the *penalty function method*. Each MFC is viewed as the presence of a fictitious elastic structural element called *penalty element* that enforces it approximately. This element is parametrized by a numerical *weight*. The exact constraint is recovered if the weight goes to infinity. The MFCs are imposed by augmenting the finite element model with the penalty elements.
3. *Lagrange Multiplier Adjunction*. For each MFC an additional unknown is adjoined to the master stiffness equations. Physically this set of unknowns represent *constraint forces* that would enforce the constraints exactly should they be applied to the unconstrained system.

For each method the exposition tries to give first the basic flavor by working out the same example for each method. The general technique is subsequently presented in matrix form for completeness but is considered an advanced topic.

Conceptually, imposing MFCs is not different from the procedure discussed in Chapter 3 for single-freedom constraints. The master stiffness equations are assembled ignoring all constraints. Then the MFCs are imposed by appropriate modification of those equations. There are, however, two important practical differences:

1. The modification process is not unique because there are alternative constraint imposition methods, namely those listed above. These methods offer tradeoffs in generality, programming implementation complexity, computational effort, numerical accuracy and stability.
2. In the implementation of some of these methods — notably penalty augmentation — constraint imposition and assembly are carried out simultaneously. In that case the framework "first assemble, then modify," is not strictly respected in the actual implementation.

Remark 8.2. The three methods are also applicable, as can be expected, to the simpler case of a single-freedom constraint such as (8.2). For most situations, however, the generality afforded by the penalty function and Lagrange multiplier methods are not warranted. The hand-oriented reduction process discussed in Chapters 3 and 4 is in fact a special case of the master-slave elimination method in which "there is no master."

Remark 8.3. Often both multifreedom and single-freedom constraints are prescribed. The modification process then involves two stages: apply multifreedom constraints and apply single freedom constraints. The order in which these are carried out is implementation dependent. Most implementations do the MFCs first, either after the assembly is completed or during the assembly process. The reason is practical: single-freedom constraints are often automatically taken care of by the equation solver itself.

§8.1.3. *MFC Matrix Forms

Matrix forms of linear MFCs are often convenient for compact notation. An individual constraint such as the second one in (8.4) may be written

$$\begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} u_{x2} \\ u_{x4} \\ u_{x6} \end{bmatrix} = 0.25. \tag{8.6}$$

In direct matrix notation:

$$\bar{\mathbf{a}}_i \bar{\mathbf{u}}_i = g_i, \qquad \text{(no sum on } i\text{)} \tag{8.7}$$

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 g_i is the right hand side scalar (0.25 in the foregoinf example). The bars over **a** and **u** distinguishes (8.7) from the expanded form (8.9) discussed below.

For method description and general proofs it is often convenient to expand matrix forms so that they embody *all* degrees of freedom. For example, if (8.6) is part of a two-dimensional finite element model with 12 freedoms: u_{x1}, u_{y1}, ... u_{y6}, the left-hand side row vector may be expanded with nine zeros as follows

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & -2 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ u_{x2} \\ \vdots \\ u_{y6} \end{bmatrix} = 0.25, \tag{8.8}$$

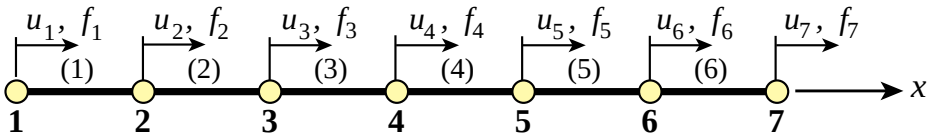


FIGURE 8.2. A one-dimensional problem discretized with six bar finite elements. The seven nodes may move only along the x direction. Subscript x is omitted from the u's and f's to reduce clutter.

in which case the matrix notation

$$\mathbf{a}_i\,\mathbf{u} = g_i \tag{8.9}$$

is used. Finally, all multifreedom constraints expressed as (8.9) may be collected into a single matrix relation:

$$\mathbf{A}\,\mathbf{u} = \mathbf{g}, \tag{8.10}$$

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

§8.2. The Example Structure

The one-dimensional finite element discretization shown in Figure 8.2 will be used throughout Chapters 8 and 9 to illustrate the three MFC application methods. This structure consists of six bar elements connected by seven nodes that can only displace in the x direction.

Before imposing various multifreedom constraints discussed below, the master stiffness equations for this problem are assumed to be

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}, \tag{8.11}$$

or

$$\mathbf{K}\mathbf{u} = \mathbf{f}. \tag{8.12}$$

The nonzero stiffness coefficients K_{ij} in (8.11) depend on the bar rigidity properties. For example, if $E^e A^e/L^e = 100$ for each element $e = 1, \ldots, 6$, 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 must move by the same amount:

$$u_2 = u_6. \tag{8.13}$$

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

$$\boxed{u_2 - u_6 = 0.} \tag{8.14}$$

Constraint conditions of this type are sometimes called *rigid links* because they can be mechanically interpreted as forcing node points 2 and 6 to move together as if they were tied by a rigid member.[1]

We now study the imposition of constraint (8.14) on the master equations (8.11) by the methods mentioned above. In this Chapter the master-slave method is treated. The other two methods: penalty augmentation and Lagrange multiplier adjunction, are discussed in the following Chapter.

§8.3. The Master-Slave Method

To apply this method *by hand*, the MFCs are taken one at a time. For each constraint a *slave* degree of freedom is chosen. The freedoms remaining in that constraint are labeled *master*. A new set of degrees of freedom $\hat{\mathbf{u}}$ is established by removing all slave freedoms from $\mathbf{u}$. This new vector contains master freedoms as well as those that do not appear in the MFCs. A matrix transformation equation that relates $\mathbf{u}$ to $\hat{\mathbf{u}}$ is generated. This equation is used to apply a congruent transformation to the master stiffness equations. This procedure yields a set of modified stiffness equations that are expressed in terms of the new freedom set $\hat{\mathbf{u}}$. Because the modified system does not contain the slave freedoms, these have been effectively eliminated.

§8.3.1. A One-Constraint Example

The mechanics of the process is best seen by going through an example. To impose (8.14) pick u_6 as slave and u_2 as master. Relate the original unknowns $u_1, \ldots u_7$ to the new set in which u_6 is missing:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix}, \tag{8.15}$$

This is the required transformation relation. In compact form:

$$\boxed{\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}}. \tag{8.16}$$

Replacing (8.15) into (8.12) and premultiplying by $\mathbf{T}^T$ yields the modified system

$$\boxed{\hat{\mathbf{K}}\,\hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T\,\mathbf{K}\,\mathbf{T}, \qquad \hat{\mathbf{f}} = \mathbf{T}^T\,\mathbf{f}.} \tag{8.17}$$

Carrying out the indicated matrix multiplications yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2+f_6 \\ f_3 \\ f_4 \\ f_5 \\ f_7 \end{bmatrix}, \tag{8.18}$$

[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.

Equation (8.18) is a new linear system containing 6 equations in the remaining 6 unknowns: u_1, u_2, u_3 , u_4, u_5 and u_7. Upon solving it, u_6 is recovered from the constraint (8.13).

Remark 8.4. The form of modified system (8.17) can be remembered by a simple mnemonic rule: premultiply both sides of $\mathbf{T}\,\hat{\mathbf{u}} = \mathbf{u}$ by $\mathbf{T}^T\,\mathbf{K}$, and replace $\mathbf{K}\,\mathbf{u}$ by $\mathbf{f}$ on the right hand side.

Remark 8.5. For a simple freedom constraint such as $u_4 = 0$ the only possible choice of slave is of course u_4 and there is no master. The congruent transformation is then nothing more than the elimination of u_4 by striking out rows and columns from the master stiffness equations.

Remark 8.6. For a simple MFC such as $u_2 = u_6$, it does not matter which degree of freedom is chosen as master or unknown. Choosing u_2 as slave produces a system of equations in which now u_2 is missing:

$$\begin{bmatrix} K_{11} & 0 & 0 & 0 & K_{12} & 0 \\ 0 & K_{33} & K_{34} & 0 & K_{23} & 0 \\ 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ K_{12} & K_{23} & 0 & K_{56} & K_{22}+K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_3 \\ f_4 \\ f_5 \\ f_2+f_6 \\ f_7 \end{bmatrix}. \tag{8.19}$$

Although (8.18) and (8.19) are algebraically equivalent, the latter would be processed faster if a skyline solver (Part III of course) is used for the modified equations.

§8.3.2. Multiple Homogeneous MFCs

The matrix equation (8.17) in fact holds for the general case of multiple homogeneous linear constraints. Direct establishment of the transformation equation, however, is more complicated if slave freedoms in one constraint appear as masters in another. To illustrate this point, suppose that for the example system we have three homogeneous multifreedom constraints:

$$u_2 - u_6 = 0, \qquad u_1 + 4u_4 = 0, \qquad 2u_3 + u_4 + u_5 = 0, \tag{8.20}$$

Picking as slave freedoms u_6, u_4 and u_3 from the first, second and third constraint, respectively, we can solve for them as

$$u_6 = u_2, \qquad u_4 = -\tfrac{1}{4}u_1, \qquad u_3 = -\tfrac{1}{2}(u_4 + u_5) = \tfrac{1}{8}u_1 - \tfrac{1}{2}u_5. \tag{8.21}$$

Observe that solving for u_3 from the third constraint brings u_4 to the right-hand side. But because u_4 is also a slave freedom (it was chosen as such for the second constraint) it is replaced in favor of u_1 using $u_4 = -\frac{1}{4}u_1$. The matrix form of the transformation (8.21) is

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \frac{1}{8} & 0 & -\frac{1}{2} & 0 \\ -\frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \\ u_7 \end{bmatrix}, \tag{8.22}$$

The modified system is now formed through the congruent transformation (8.17). Note that the slave freedoms selected from each constraint must be distinct; for example the choice u_6, u_4, u_4

would be inadmissible as long as the constraints are independent. This rule is easy to enforce when slave freedoms are chosen by hand, but can lead to implementation and numerical difficulties when it is programmed as an automated procedure, as further discussed later.

Remark 8.7. The three MFCs (8.20) with u_6, u_4 and u_2 chosen as slaves and u_1, u_2 and u_5 chosen as masters, may be presented in the partitioned matrix form:

$$\begin{bmatrix} 0 & 0 & 1 \\ 0 & 4 & 0 \\ 2 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_3 \\ u_4 \\ u_6 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_5 \end{bmatrix} \tag{8.23}$$

This may be compactly written $\mathbf{A}_s\mathbf{u}_s + \mathbf{A}_m\mathbf{u}_m = \mathbf{0}$. Solving for the slave freedoms gives $\mathbf{u}_s = -\mathbf{A}_s^{-1}\mathbf{A}_m\mathbf{u}_m$. Expanding with zeros to fill out $\mathbf{u}$ and $\hat{\mathbf{u}}$ produces (8.22). This general matrix form is considered in §8.4.4. Note that non-singularity of $\mathbf{A}_s$ is essential for this method to work.

§8.3.3. Nonhomogeneous MFCs

Extension to non-homogeneous constraints is immediate. In this case he transformation equation becomes non-homogeneous. For example suppose that (8.14) has a nonzero prescribed value:

$$\boxed{u_2 - u_6 = 0.2} \tag{8.24}$$

Nonzero RHS values such as 0.2 in (8.24) may be often interpreted physically as "gaps" (thus the use of the symbol $\mathbf{g}$ in the matrix form). Chose u_6 again as slave: $u_6 = u_2 - 0.2$, and build the transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -0.2 \\ 0 \end{bmatrix}. \tag{8.25}$$

In compact matrix notation,

$$\boxed{\mathbf{u} = \mathbf{T}\,\hat{\mathbf{u}} + \mathbf{g}.} \tag{8.26}$$

Here the constraint gap vector $\mathbf{g}$ is nonzero and $\mathbf{T}$ is the same as before. To get the modified system applying the shortcut rule of Remark 8.4, premultiply both sides of (8.26) by $\mathbf{T}^T\mathbf{K}$, replace $\mathbf{Ku}$ by $\mathbf{f}$, and pass the data to the RHS:

$$\boxed{\hat{\mathbf{K}}\,\hat{\mathbf{u}} = \hat{\mathbf{f}}, \quad \text{in which} \quad \hat{\mathbf{K}} = \mathbf{T}^T\,\mathbf{K}\,\mathbf{T}, \quad \hat{\mathbf{f}} = \mathbf{T}^T(\mathbf{f} - \mathbf{K}\,\mathbf{g}).} \tag{8.27}$$

Upon solving (8.27) for $\hat{\mathbf{u}}$, the complete displacement vector is recovered from (8.26). For the MFC (8.24) this technique gives the system

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & K_{56} & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 \\ 0 & K_{67} & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 - 0.2K_{66} \\ f_3 \\ f_4 \\ f_5 - 0.2K_{56} \\ f_7 - 0.2K_{67} \end{bmatrix}. \tag{8.28}$$

See Exercise 8.2 for multiple non-homogeneous MFCs.

§8.3.4. *The General Case

For implementation in general-purpose programs the master-slave method can be described as follows. The degrees of freedoms in $\mathbf{u}$ are classified into three types: independent or unconstrained, masters and slaves. (The unconstrained freedoms are those that do not appear in any MFC.) Label these sets as $\mathbf{u}_u$, $\mathbf{u}_m$ and $\mathbf{u}_s$, respectively, and partition the stiffness equations accordingly:

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um} & \mathbf{K}_{us} \\ \mathbf{K}_{um}^T & \mathbf{K}_{mm} & \mathbf{K}_{ms} \\ \mathbf{K}_{us}^T & \mathbf{K}_{ms}^T & \mathbf{K}_{ss} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \\ \mathbf{u}_s \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u \\ \mathbf{f}_m \\ \mathbf{f}_s \end{bmatrix} \tag{8.29}$$

The MFCs may be written in matrix form as

$$\mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s \mathbf{u}_s = \mathbf{g}_A, \tag{8.30}$$

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

$$\mathbf{u}_s = -\mathbf{A}_s^{-1} \mathbf{A}_m \mathbf{u}_m + \mathbf{A}_s^{-1} \mathbf{g}_A \stackrel{\text{def}}{=} \mathbf{T} \mathbf{u}_m + \mathbf{g}, \tag{8.31}$$

Inserting into the partitioned stiffness equations (8.30) and symmetrizing yields

$$\begin{bmatrix} \mathbf{K}_{uu} & \mathbf{K}_{um} + \mathbf{K}_{us}\mathbf{T} \\ \text{symm} & \mathbf{K}_{mm} + \mathbf{T}^T\mathbf{K}_{ms}^T + \mathbf{K}_{ms}\mathbf{T} + \mathbf{T}^T\mathbf{K}_{ss}\mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{u}_u \\ \mathbf{u}_m \end{bmatrix} = \begin{bmatrix} \mathbf{f}_u - \mathbf{K}_{us}\mathbf{g} \\ \mathbf{f}_m - \mathbf{K}_{ms}\mathbf{g} \end{bmatrix} \tag{8.32}$$

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

§8.3.5. *Retaining the Original Freedoms

A potential disadvantage of the master-slave method in computer work is that it requires a rearrangement of the original stiffness equations because $\hat{\mathbf{u}}$ is a subset of $\mathbf{u}$. The disadvantage can be annoying when sparse matrix storage schemes are used for the stiffness matrix, and becomes intolerable if secondary storage is used for that purpose.

With a bit of trickery it is possible to maintain the original freedom ordering. Let us display it for the example problem under (8.14). Instead of (8.15), use the *square* transformation

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix}, \tag{8.33}$$

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

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+K_{66} & K_{23} & 0 & 0 & K_{56} & 0 & K_{67} \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & K_{56} & 0 & K_{45} & K_{55} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & K_{67} & 0 & 0 & 0 & 0 & 0 & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ \tilde{u}_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 + f_6 \\ f_3 \\ f_4 \\ f_5 \\ 0 \\ f_7 \end{bmatrix}, \tag{8.34}$$

which are submitted to the equation solver. If the solver is not trained to skip zero rows and columns, a one should be placed in the diagonal entry for the $\tilde{u}_6$ (sixth) equation. The solver will return $\tilde{u}_6 = 0$, and this placeholder value is replaced by u_2. Note several points in common with the computer-oriented placeholder technique described in §3.4 to handle single-freedom constraints.

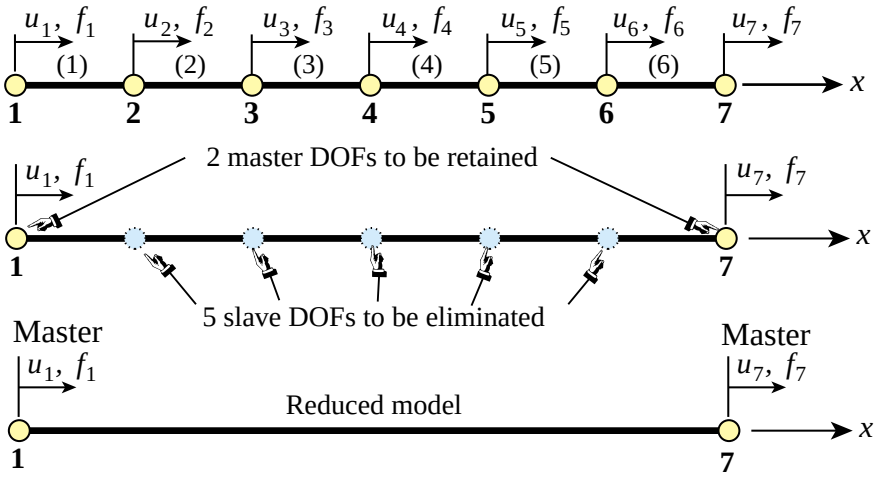


FIGURE 8.3. Model reduction of the example structure of Figure 8.2 to the end freedoms u_1 and u_7.

§8.4. Model Reduction by Kinematic Constraints

The congruent transformation equations (8.17) and (8.27) have additional applications beyond the master-slave method. An important one is *model reduction by kinematic constraints*. Through this procedure the number of DOF of a static or dynamic FEM model is reduced by a significant number, typically to 1% – 10% of the original number. This is done by taking a lot of slaves and a few masters. Only the masters are left after the transformation. The reduced model is commonly used in subsequent calculations as component of a larger system, particularly during design or in parameter identification.

Example 8.2. Consider the bar assembly of Figure 8.2. Assume that the only masters are the end motions u_1 and u_7, as illustrated in Figure 8.2, and interpolate all freedoms linearly:

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 5/6 & 1/6 \\ 4/6 & 2/6 \\ 3/6 & 3/6 \\ 2/6 & 4/6 \\ 1/6 & 5/6 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix}, \qquad \text{or} \quad \mathbf{u} = \mathbf{T}\,\hat{\mathbf{u}}. \tag{8.35}$$

The reduced-order-model (ROM) equations are $\hat{\mathbf{K}}\hat{\mathbf{u}} = \mathbf{T}^T\mathbf{K}\mathbf{T}\hat{\mathbf{u}} = \mathbf{T}^T\mathbf{f} = \hat{\mathbf{f}}$, or in detail

$$\begin{bmatrix} \hat{K}_{11} & \hat{K}_{12} \\ \hat{K}_{12} & \hat{K}_{22} \end{bmatrix} \begin{bmatrix} u_1 \\ u_7 \end{bmatrix} = \begin{bmatrix} \hat{f}_1 \\ \hat{f}_2 \end{bmatrix}, \tag{8.36}$$

in which

$$\begin{aligned} \hat{K}_{11} &= \tfrac{1}{36}(36K_{11}+60K_{12}+25K_{22}+40K_{23}+16K_{33}+24K_{34}+9K_{44}+12K_{45}+4K_{55}+4K_{56}+K_{66}), \\ \hat{K}_{12} &= \tfrac{1}{36}(6K_{12}+5K_{22}+14K_{23}+8K_{33}+18K_{34}+9K_{44}+18K_{45}+8K_{55}+14K_{56}+5K_{66}+6K_{67}), \\ \hat{K}_{22} &= \tfrac{1}{36}(K_{22}+4K_{23}+4K_{33}+12K_{34}+9K_{44}+24K_{45}+16K_{55}+40K_{56}+25K_{66}+60K_{67}+36K_{77}), \\ \hat{f}_1 &= \tfrac{1}{6}(6f_1+5f_2+4f_3+3f_4+2f_5+f_6), \quad \hat{f}_2 = \tfrac{1}{6}(f_2+2f_3+3f_4+4f_5+5f_6+6f_7). \end{aligned} \tag{8.37}$$

```
(* Model Reduction by Kinematic Constraints - Example *)
ClearAll[K11,K12,K22,K23,K33,K34,K44,K45,K55,K56,K66,K67,K77,
         f1,f2,f3,f4,f5,f6,f7];
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={{6,0},{5,1},{4,2},{3,3},{2,4},{1,5},{0,6}}/6;
Print[" T (transposed to save space)=",Transpose[T]//MatrixForm];
Khat=Simplify[Transpose[T].K.T];
fhat=Simplify[Transpose[T].f];
Print["Modified stiffness entries:"];
Print["Khat(1,1)=",Khat[[1,1]],"\nKhat(1,2)=",Khat[[1,2]],
      "\nKhat(2,2)=",Khat[[2,2]] ];
Print["Modified force entries:"];
Print["fhat(1)=",fhat[[1]]," fhat(2)=",fhat[[2]] ];
```

$$K=\begin{pmatrix} K11 & K12 & 0 & 0 & 0 & 0 & 0 \\ K12 & K22 & K23 & 0 & 0 & 0 & 0 \\ 0 & K23 & K33 & K34 & 0 & 0 & 0 \\ 0 & 0 & K34 & K44 & K45 & 0 & 0 \\ 0 & 0 & 0 & K45 & K55 & K56 & 0 \\ 0 & 0 & 0 & 0 & K56 & K66 & K67 \\ 0 & 0 & 0 & 0 & 0 & K67 & K77 \end{pmatrix}$$

f={f1,f2,f3,f4,f5,f6,f7};

$$T \text{ (transposed to save space)}= \begin{pmatrix} 1 & \frac{5}{6} & \frac{2}{3} & \frac{1}{2} & \frac{1}{3} & \frac{1}{6} & 0 \\ 0 & \frac{1}{6} & \frac{1}{3} & \frac{1}{2} & \frac{2}{3} & \frac{5}{6} & 1 \end{pmatrix}$$

Modified stiffness entries:

$$Kkat(1,1)= \frac{1}{36}(36\,K11 + 60\,K12 + 25\,K22 + 40\,K23 + 16\,K33 + 24\,K34 + 9\,K44 + 12\,K45 + 4\,K55 + 4K56 + K66)$$

$$Khat(1,2)= \frac{1}{36}(6\,K12 + 5\,K22 + 14\,K23 + 8\,K33 + 18\,K34 + 9\,K44 + 18\,K45 + 8\,K55 + 14\,K56 + 5\,K66 + 6\,K67)$$

$$Khat(2,2)= \frac{1}{36}(K22 + 4\,K23 + 4\,K33 + 12\,K34 + 9\,K44 + 24\,K45 + 16\,K55 + 40\,K56 + 25\,K66 + 60\,K67 + 36\,K77)$$

Modified force entries:

$$fhat(1)= \frac{1}{6}(6\,f1 + 5\,f2 + 4\,f3 + 3\,f4 + 2\,f5 + f6) \quad fhat(2)= \frac{1}{6}(f2 + 2\,f3 + 3\,f4 + 4\,f5 + 5\,f6 + 6\,f7)$$

FIGURE 8.4. *Mathematica* script for the model reduction example of Figure 8.3 and printed output.

This reduces the order of the FEM model from 7 to 2. A *Mathematica* script to do the reduction is shown in Figure 8.4. The key feature is that the masters are picked *a priori*, as the freedoms to be retained in the model for further use.

Remark 8.8. Model reduction can also be done by the static condensation method explained in Chapter 10. As its name indicates, condensation is restricted to static analysis. On the other hand, for such problems it is exact whereas model reduction by kinematic constraints generally introduces approximations.

§8.5. Assessment of the Master-Slave Method

What are the good and bad points of this constraint application method? It enjoys the advantage of being exact (except for inevitable solution errors) and of reducing the number of unknowns. The underlying ideas are easy to explain and learn. The main implementation drawback is the

complexity of the general case as can be seen by studying (8.29) through (8.32). The complexity is due to three factors:

1. The equations may have to be rearranged because of the disappearance of the slave freedoms. This drawback can be alleviated, however, through the placeholder trick outlined in §8.3.5
2. An auxiliary linear system, namely (8.31), may have 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 those entries can be difficult and expensive.

The level of complexity depends on the generality allowed as well as on programming decisions. If **K** is stored as full matrix and slave freedom coupling in the MFCs is disallowed the logic is simple.[2] On the other hand, if arbitrary couplings are permitted and **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 disjoint 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:

$$\tfrac{1}{6}u_2 + \tfrac{1}{2}u_4 = u_6, \qquad u_3 + 6u_6 = u_7. \tag{8.38}$$

For numerical stability reasons it is usually better to pick as slaves the freedoms with larger coefficients. If this is done, the program would select u_6 as slave freedoms from both constraints. This leads to a contradiction because having two constraints we must eliminate two slave degrees of freedom, not just one. The resulting modified system would in fact be inconsistent. Although this defect can be easily fixed by the program logic in this case, one can imagine the complexity burden if faced with hundreds or thousands of MFCs; e,g., in incompressible materials.

Serious numerical problems can arise if the MFCs are not independent. For example:

$$\tfrac{1}{6}u_2 = u_6, \qquad \tfrac{1}{5}u_3 + 6u_6 = u_7, \qquad u_2 + u_3 - u_7 = 0. \tag{8.39}$$

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 (8.39), can be recognized by a rank analysis of the $\mathbf{A}_s$ matrix defined in (8.30). In floating-point arithmetic, however, such detection may fail because that kind of computation is inexact by nature.[3]

The complexity of slave selection is in fact equivalent to that of automatically selecting kinematic redundancies in the Force Method of structural analysis. It has led implementors of programs that use this method while requiring masters and slaves to be identified in the input data, thus transfering the burden to users.

---

[2] This is the case in model reduction, since each slave freedom appears in one and only one MFC.

[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.

The method is not generally extendible to nonlinear constraints without case by case programming.

In conclusion, the master-slave method is useful when a few simple linear constraints are imposed by hand. As a general purpose technique for finite element analysis it suffers from complexity and lack of robustness. It is worth learning, however, because of the great importance of congruent transformations in *model reduction* for static and dynamic problems.

Notes and Bibliography

Multifreedom constraints are treated in several of the FEM books recommended in §1.7.5, notably Zienkiewicz and Taylor [862]. The master-slave method was incorporated to treat MFCs as part of the Direct Stiffness Method (DSM) developed at Boeing during the 1950s. It is first summarily described in the DSM-overview by Turner, Martin and Weikel [789, p. 212]. The implementation differs, however, from the one presented here because the relation of FEM to energy methods was not clear at the time.

The master-slave method became popular through its adoption by the general-purpose NASTRAN code developed in the late 1960s [21] and early assessments of its potential [773]. The implementation unfortunately relied on user inputs to identify slave DOFs. Through this serious blunder the method gained a reputation for unreliability that persists to the present day.

The application of master-slave to model reduction by kinematic constraints, which by itself justifies teaching the method, is rarely mentioned in FEM textbooks.

References

Referenced items have been moved to Appendix R.

Homework Exercises for Chapter 8
MultiFreedom Constraints I

EXERCISE 8.1 [C+N:20] The example structure of Figure 8.2 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 (8.27) apply the left-end support $u_1 = 0$ using the placeholder method of §3.4. Solve the equations and verify that the displacement solution and the recovered node forces including reactions are

$$\begin{aligned} \mathbf{u} &= [\,0 \quad 0.270 \quad 0.275 \quad 0.250 \quad 0.185 \quad 0.070 \quad 0.140\,]^T \\ \mathbf{Ku} &= [\,-27 \quad 26.5 \quad 3 \quad 4 \quad 5 \quad -18.5 \quad 7\,]^T \end{aligned} \tag{E8.1}$$

The solution procedure is outlined in the C-style pseudo-code in Figure E8.1. This can be readily translated to any high-level matrix language such as *Matlab*, *Mathematica*, or *Octave*. For example, an implementation in *Mathematica* is shown in Figure E8.2.

```
procedure MasterStiffnessOfSixElementBar(kbar);
   real K=nullmatrix(7,7)); K(1,1)=K(7,7)=kbar;
   for  i=2,i<=6,i++, K(i,i)=2*kbar; endfor;
   for  i=1,i<=6,i++, K(i,i+1)=K(i+1,i)=–kbar; endfor;
   return(K); endprocedure;
procedure FixLeftEndOfSixElementBar(Khat,fhat);
  Kmod=Khat; fmod=fhat;  fmod(1)=0; Kmod(1,1)=1;
  Kmod(1,2)=Kmod(1,2)=0; return((Kmod,fmod)); endprocedure;
// Main script follows
K=MasterStiffnessOfSixElementBar(100); print K; // construct master K
f=vector(1,2,3,4,5,6,7); print f; // construct force vector
T=matrix((1,0,0,0,0,0),(0,1,0,0,0,0),(0,0,1,0,0,0),
    (0,0,0,1,0,0),(0,0,0,0,1,0),(0,1,0,0,0,0),(0,0,0,0,0,1)); print T;
// construct transformation matrix T
g=vector(0,0,0,0,0,–1/5,0);  print g; // construct gap vector
Khat=T'.K.T; fhat=T'.(f–K.g); // modified stiffness and force
(Kmod,fmod)=FixLeftEndOfSixElementBar(Khat,fhat); // fix left end
print Kmod; print fmod;
umod=inverse(Kmod).fmod; print umod; // modified solution vector
u=T.umod+g; print u;   // complete solution
fu=K.u; print fu; //  recovered forces including reactions
```

FIGURE E8.1. Pseudo-code script for solving Exercise 8.1. Notes: reserved keywords appear in boldface, statements are terminated by semicolons, and `//` starts an inline comment a la `C++`.

```
(* Exercise 8.1 - Master-Slave Method  *)
(* MFC: u2-u6 = 1/5  - slave: u6  *)
MasterStiffnessOfSixElementBar[kbar_]:=Module[
  {K=Table[0,{7},{7}]}, K[[1,1]]=K[[7,7]]=kbar;
  For [i=2,i<=6,i++,K[[i,i]]=2*kbar];
  For [i=1,i<=6,i++,K[[i,i+1]]=K[[i+1,i]]=-kbar];
  Return[K]];
FixLeftEndOfSixElementBar[Khat_,fhat_]:=Module[
  {Kmod=Khat,fmod=fhat}, fmod[[1]]=0; Kmod[[1,1]]=1;
  Kmod[[1,2]]=Kmod[[2,1]]=0; Return[{Kmod,fmod}]];

K=MasterStiffnessOfSixElementBar[100];
Print["Stiffness K=",K//MatrixForm];
f={1,2,3,4,5,6,7}; Print["Applied forces=",f];
T={{1,0,0,0,0,0},{0,1,0,0,0,0},{0,0,1,0,0,0},
   {0,0,0,1,0,0},{0,0,0,0,1,0},{0,1,0,0,0,0},
   {0,0,0,0,0,1}};
Print["Transformation matrix T=",T//MatrixForm];
g={0,0,0,0,0,-1/5,0};
Print["Constraint gap vector g=",g];
Khat=Simplify[Transpose[T].K.T]; fhat=Simplify[Transpose[T].(f-K.g)];
{Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat]; (* fix left end *)
Print["Modified Stiffness upon fixing node 1:",Kmod//MatrixForm];
Print["Modified RHS upon fixing node 1:",fmod];
umod=LinearSolve[Kmod,fmod];
Print["Computed umod (lacks slave u6)=",umod];
u=T.umod+g; Print["Complete solution u=",u];
Print["Numerical u=",N[u]];
fu=K.u; Print["Recovered forces K.u with reactions=",fu];
Print["Numerical K.u=",N[fu]];
```

FIGURE E8.2. *Mathematica* script for solving Exercise 8.1.

EXERCISE 8.2 [C+N:25] As in the previous Exercise but applying the following three MFCs, two of which are non-homogeneous:

$$u_2 - u_6 = 1/5, \qquad u_3 + 2u_4 = -2/3, \qquad 2u_3 - u_4 + u_5 = 0. \tag{E8.2}$$

Hint. Chose u_4, u_5 and u_6 as slaves. Much of the script shown for Exercise 8.1 can be reused. The main changes are in the formation of **T** and **g**. If you are a *Mathematica* wizard (or willing to be one) those can be automatically formed by the statements listed in Figure E8.3.

```
sol=Simplify[Solve[{u2-u6==1/5, u3+2*u4==-2/3,2*u3-u4+u5==0},{u4,u5,u6}]];
ums={u1,u2,u3,u4,u5,u6,u7}/.sol[[1]]; um={u1,u2,u3,u7};
T=Table[Coefficient[ums[[i]],um[[j]]],{i,1,7},{j,1,4}];
g=ums/.{u1->0,u2->0,u3->0,u4->0,u5->0,u6->0,u7->0};
Print["Transformation matrix T=",T//MatrixForm];
Print["Gap vector g=",g];
```

FIGURE E8.3. Script for partially solving Exercise 8.2.

If you do this, explain what it does and why it works. Otherwise form and enter **T** and **g** by hand. The numerical results (shown to 5 places) should be

$$\begin{aligned} \mathbf{u} &= [\,0. \quad 0.043072 \quad -0.075033 \quad -0.29582 \quad -0.14575 \quad -0.15693 \quad -0.086928\,]^T\,, \\ \mathbf{Ku} &= [\,-4.3072 \quad 16.118 \quad 10.268 \quad -37.085 \quad 16.124 \quad -8.1176 \quad 7.\,]^T\,. \end{aligned} \tag{E8.3}$$

EXERCISE 8.3 [A:25] Can the MFCs be pre-processed to make sure that no slave freedom in a MFC appears as master in another?

EXERCISE 8.4 [A:25] In the general case discussed in §8.4.4, under which condition is the matrix $\mathbf{A}_s$ of (8.30) diagonal and thus trivially invertible?

EXERCISE 8.5 [A:25] Work out the general technique by which the unknowns need not be rearranged, that is, $\mathbf{u}$ and $\hat{\mathbf{u}}$ are the same. Use "placeholders" for the slave freedoms. (Hint: use ideas of §3.4).

EXERCISE 8.6 [A/C:35] Is it possible to establish a slave selection strategy that makes $\mathbf{A}_s$ diagonal or triangular? (This requires knowledge of matrix techniques such as pivoting.)

EXERCISE 8.7 [A/C:40] Work out a strategy that produces a well conditioned $\mathbf{A}_s$ by selecting new slaves as linear combinations of finite element freedoms if necessary. (Requires background in numerical analysis and advanced programming experience in matrix algebra).

9

MultiFreedom Constraints II

TABLE OF CONTENTS

§9.1. Introduction

In this Chapter we continue the discussion of methods to treat multifreedom constraints (MFCs). The master-slave method described previously was easy to explain, but exhibits serious shortcomings for treating arbitrary constraints (although the method has important applications to model reduction).

We now pass to the study of two other methods: penalty augmentation and Lagrange multiplier adjunction. Both techniques are better suited to general implementations of the Finite Element Method, whether linear or nonlinear.

§9.2. The Penalty Method

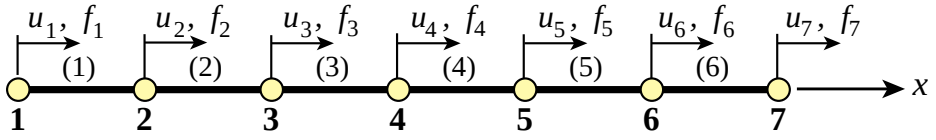


FIGURE 9.1. The example structure of Chapter 8, repeated for convenience.

§9.2.1. Physical Interpretation of Penalty Method

The penalty method will be first presented using a physical argument, leaving the mathematical formulation to a subsequent section. Consider again the example structure of Chapter 8, which is reproduced in Figure 9.1 for convenience. To impose $u_2 = u_6$ imagine that nodes 2 and 6 are connected with a "fat" bar of axial stiffness w, labeled with element number 7, as shown in Figure 9.2. This bar is called a *penalty element* and w is its *penalty weight*.

Such an element, albeit fictitious, can be treated exactly like another bar element insofar as continuing the assembly of the master stiffness equations. The penalty element stiffness equations, $\mathbf{K}^{(7)}\mathbf{u}^{(7)} = \mathbf{f}^{(7)}$, are[1]

$$w\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} u_2 \\ u_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{9.1}$$

Because there is one freedom per node, the two local element freedoms map into global freedoms 2 and 6, respectively. Using the assembly rules of Chapter 3 we obtain the following modified master stiffness equations: $\widehat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}$, which shown in detail are

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22}+w & K_{23} & 0 & 0 & -w & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & -w & 0 & 0 & K_{56} & K_{66}+w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \tag{9.2}$$

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

[1] The general method to construct these equations is described in §9.1.4.

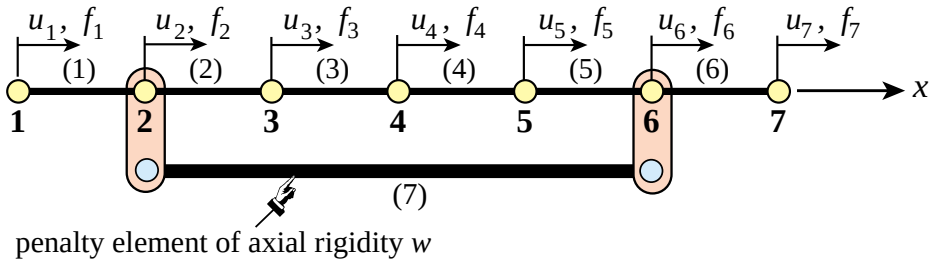


FIGURE 9.2. Adjunction of a fictitious penalty bar of axial stiffness w, identified as element 7,to enforce the MFC $u_2 = u_6$.

§9.2.2. Choosing the Penalty Weight

What happens when (9.2) is solved numerically? If a *finite* weight w is chosen the constraint $u_2 = u_6$ is approximately satisfied in the sense that one gets $u_2 - u_6 = e_g$, where $e_g \neq 0$. The "gap error" e_g is called the *constraint violation*. The magnitude $|e_g|$ of this violation depends on the weight: the larger w, the smaller the violation. More precisely, it can be shown that $|e_g|$ becomes proportional to $1/w$ as w gets to be sufficiently large (see Exercises). For example, raising w from, say, 10^6 to 10^7 can be expected to cut the constraint violation by roughly 10 if the physical stiffnesses are small compared to w.

Therefore it seems as if the proper strategy should be: try to make w as large as possible while respecting computer overflow limits. However, this is misleading. As the penalty weight w tends to ∞ the modified stiffness matrix in (9.2) becomes more and more *ill-conditioned with respect to inversion*.

To make this point clear, suppose for definiteness that the rigidities $E^e A^e/L^e$ of the actual bars $e = 1, \ldots 6$ 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.[2] The modified stiffness matrix of (9.2) becomes

$$\widehat{\mathbf{K}} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2+w & -1 & 0 & 0 & -w & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & -w & 0 & 0 & -1 & 2+w & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{9.3}$$

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. But *linear dependency means singularity*. Therefore $\widehat{\mathbf{K}}$ 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.

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

§9.2.3. The Square Root Rule

Plainly we have two effects at odds with each other. Making w larger reduces the constraint violation error but increases the solution error. The best w is that which makes both errors roughly equal in absolute value. This tradeoff value is difficult to find aside of systematically running numerical experiments. In practice the heuristic *square root rule* is often followed.

This rule can be stated as follows. Suppose that the largest stiffness coefficient, before adding penalty elements, is of the order of 10^k and that the working machine precision is p digits.[3] Then choose penalty weights to be of order $10^{k+p/2}$ with the proviso that such a choice would not cause arithmetic overflow.[4]

For the above example in which $k \approx 0$ and $p \approx 16$, the optimal w given by this rule would be $w \approx 10^8$. This w would yield a constraint violation and a solution error of order 10^{-8}. Note that there is no simple way to do better than this accuracy aside from using extended (e.g., quad) floating-point precision. This is not easy to do when using standard low-level programming languages.

The name "square root" arises because the recommended w is in fact $10^k\sqrt{10^p}$. It is seen that picking the weight by this rule requires knowledge of both stiffness magnitudes and floating-point hardware properties of the computer used, as well as the precision selected by the program.

§9.2.4. Penalty Elements for General MFCs

For the constraint $u_2 = u_6$ the physical interpretation of the penalty element is clear. Nodal points 2 and 6 must move in lockstep long x, which can be approximately enforced by the heavy bar device shown in Figure 9.2. But how about $3u_3 + u_5 - 4u_6 = 1$? Or just $u_2 = -u_6$?

The treatment of more general constraints is linked to the theory of *Courant penalty functions*, which in turn is a topic in variational calculus. Because the necessary theory given in §9.1.5 is viewed as an advanced topic, the procedure used for constructing a penalty element is stated here as a recipe. Consider the homogeneous constraint

$$3u_3 + u_5 - 4u_6 = 0. \tag{9.4}$$

Rewrite this equation in matrix form

$$\begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 0, \tag{9.5}$$

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

$$\begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix} \begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \bar{\mathbf{K}}^e \mathbf{u}^e = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}. \tag{9.6}$$

[3] Such order-of-magnitude estimates can be readily found by scanning the diagonal of $\mathbf{K}$ because the largest stiffness coefficient of the actual structure is usually a diagonal entry.

[4] If overflows occurs, the master stiffness should be scaled throughout or a better choice of physical units made.

Here $\bar{\mathbf{K}}^e$ is the *unscaled* stiffness matrix of the penalty element. This is now multiplied by the penalty weight w and assembled into the master stiffness matrix following the usual rules. For the example problem, augmenting (9.2) with the w-scaled penalty element (9.6) yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33}+9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55}+w & K_{56}-4w & 0 \\ 0 & 0 & -12w & 0 & K_{56}-4w & K_{66}+16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \tag{9.7}$$

If the constraint is nonhomogeneous the force vector is also modified. To illustrate this effect, consider the MFC: $3u_3 + u_5 - 4u_6 = 1$. Rewrite in matrix form as

$$\begin{bmatrix} 3 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = 1. \tag{9.8}$$

Premultiply both sides by the transpose of the coefficient matrix:

$$\begin{bmatrix} 9 & 3 & -12 \\ 3 & 1 & -4 \\ -12 & -4 & 16 \end{bmatrix} \begin{bmatrix} u_3 \\ u_5 \\ u_6 \end{bmatrix} = \begin{bmatrix} 3 \\ 1 \\ -4 \end{bmatrix}. \tag{9.9}$$

Scaling by w and assembling yields

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33}+9w & K_{34} & 3w & -12w & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 3w & K_{45} & K_{55}+w & K_{56}-4w & 0 \\ 0 & 0 & -12w & 0 & K_{56}-4w & K_{66}+16w & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3+3w \\ f_4 \\ f_5+w \\ f_6-4w \\ f_7 \end{bmatrix}. \tag{9.10}$$

§9.2.5. *The Theory Behind the Penalty Method

The rule comes from the following mathematical theory. Suppose we have a set of m linear MFCs. Using the matrix notation introduced in §8.1.3, these will be stated as

$$\mathbf{a}_p\mathbf{u} = b_p, \quad p = 1, \ldots m \tag{9.11}$$

where $\mathbf{u}$ contains all degrees of freedom and each $\mathbf{a}_p$ is a row vector with same length as $\mathbf{u}$. To incorporate the MFCs into the FEM model one selects a weight $w_p > 0$ for each constraints and constructs the so-called Courant quadratic penalty function or "penalty energy"

$$P = \sum_{p=1}^{m} P_p, \qquad \text{with} \quad P_p = \mathbf{u}^T \left(\tfrac{1}{2}\mathbf{a}_p^T\mathbf{a}_p\mathbf{u} - w_p\mathbf{a}_p^T b_p\right) = \tfrac{1}{2}\mathbf{u}^T\mathbf{K}^{(p)}\mathbf{u} - \mathbf{u}^T\mathbf{f}^{(p)}, \tag{9.12}$$

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

$$\left(\mathbf{K} + \sum_{p=1}^{m} \mathbf{K}^{(p)}\right) \mathbf{u} = \mathbf{f} + \sum_{p=1}^{m} \mathbf{f}^{(p)}. \tag{9.13}$$

Each term of the sum on p, which derives from term P_p in (9.12), may be viewed as contributed by a penalty element with globalized stiffness matrix $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and globalized added force term $\mathbf{f}^{(p)} = w_p \mathbf{a}_p^T b_p$.

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

$$(\mathbf{K} + \mathbf{A}^T\mathbf{W}\mathbf{A})\,\mathbf{u} = \mathbf{f} + \mathbf{W}\mathbf{A}^T\mathbf{b}, \tag{9.14}$$

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

§9.2.6. Assessment of the Penalty Method

The main advantage of the penalty function method is its straightforward computer implementation. Looking at modified systems such as (9.2), (9.7) or (9.10) it is obvious that the master equations need not be rearranged. That is, $\mathbf{u}$ and $\hat{\mathbf{u}}$ are the same. Constraints may be programmed as "penalty elements," and stiffness and force contributions of these elements merged through the standard assembler. In fact using this method there is no need to distinguish between unconstrained and constrained equations! Once all elements — regular and penalty — are assembled, the system can be passed to the equation solver.[5]

An important advantage with respect to the master-slave (elimination) method is its lack of sensitivity with respect to whether constraints are linearly dependent. To give a simplistic example, suppose that the constraint $u_2 = u_6$ appears twice. Then two penalty elements connecting 2 and 6 will be inserted, doubling the intended weight but not otherwise causing undue harm.

An advantage with respect to the Lagrange multiplier method described in §9.2 is that positive definiteness is not lost. Such loss can affect the performance of certain numerical processes.[6] Finally, it is worth noting that the penalty method is easily extendible to nonlinear constraints although such extension falls outside the scope of this book.

The main disadvantage, however, is a serious one: the choice of weight values that balance solution accuracy with the violation of constraint conditions. For simple cases the square root rule previously described often works, although its effective use calls for knowledge of the magnitude of stiffness coefficients. Such knowledge may be difficult to extract from a general purpose "black box" program. For difficult cases selection of appropriate weights may require extensive numerical experimentation, wasting the user time with numerical games that have no bearing on the actual objective, which is getting a solution.

The deterioration of the condition number of the penalty-augmented stiffness matrix can have serious side effects in some solution procedures such as eigenvalue extraction or iterative solvers.

---

[5] Single freedom constraints, such as those encountered in Chapter 3, are usually processed separately for efficiency.

[6] For example, solving the master stiffness equations by Cholesky factorization or conjugate-gradients.

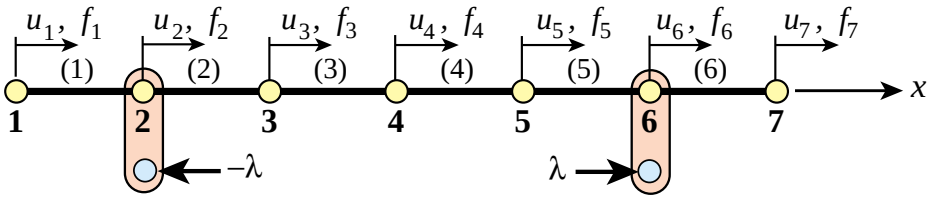


FIGURE 9.3. Physical interpretation of Lagrange multiplier adjunction to enforce the MFC $u_2 = u_6$.

Finally, even if optimal weights are selected, the combined solution error cannot be lowered beyond a threshold value.

From this assessment it is evident that penalty augmentation, although superior to the master-slave method from the standpoint of generality and ease of implementation, is no panacea.

§9.3. Lagrange Multiplier Adjunction

§9.3.1. Physical Interpretation

As in the case of the penalty function method, the method of Lagrange multipliers can be given a rigorous justification within the framework of variational calculus. But in the same spirit it will be introduced for the example structure from a physical standpoint that is particularly illuminating.

Consider again the constraint $u_2 = u_6$. Borrowing some ideas from the penalty method, imagine that nodes 2 and 6 are connected now by a *rigid* link rather than a flexible one. Thus the constraint is imposed exactly. But of course the penalty method with an infinite weight would "blow up."

We may remove the link if it is replaced by an appropriate reaction force pair $(-\lambda, +\lambda)$, as illustrated in Figure 9.3. These are called the *constraint forces*. Incorporating these forces into the original stiffness equations (8.10) we get

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 - \lambda \\ f_3 \\ f_4 \\ f_5 \\ f_6 + \lambda \\ f_7 \end{bmatrix}. \tag{9.15}$$

This λ is called a *Lagrange multiplier*. Because λ is an unknown, let us transfer it to the *left hand side* by *appending* it to the vector of unknowns:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \end{bmatrix}. \tag{9.16}$$

But now we have 7 equations in 8 unknowns. To render the system determinate, the constraint condition $u_2 - u_6 = 0$ is appended as eighth equation:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \end{bmatrix}, \tag{9.17}$$

This is called the *multiplier-augmented* system. Its coefficient matrix, which is symmetric, is called the *bordered stiffness matrix*. The process by which λ is appended to the vector of original unknowns is called *adjunction*. Solving this system provides the desired solution for the degrees of freedom while also characterizing the constraint forces through λ.

§9.3.2. Lagrange Multipliers for General MFCs

The general procedure will be stated first as a recipe. Suppose that we want to solve the example structure subjected to three MFCs

$$u_2 - u_6 = 0, \qquad 5u_2 - 8u_7 = 3, \qquad 3u_3 + u_5 - 4u_6 = 1, \tag{9.18}$$

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

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}, \tag{9.19}$$

Three Lagrange multipliers: λ_1, λ_2 and λ_3, are required to take care of three MFCs. Adjoin those unknowns to the nodal displacement vector. Symmetrize the coefficient matrix by appending 3 columns that are the transpose of the 3 last rows in (9.19), and filling the bottom right-hand corner

with a 3×3 zero matrix:

$$\begin{bmatrix} K_{11} & K_{12} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ K_{12} & K_{22} & K_{23} & 0 & 0 & 0 & 0 & 1 & 5 & 0 \\ 0 & K_{23} & K_{33} & K_{34} & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & K_{34} & K_{44} & K_{45} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & K_{45} & K_{55} & K_{56} & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & K_{56} & K_{66} & K_{67} & -1 & 0 & -4 \\ 0 & 0 & 0 & 0 & 0 & K_{67} & K_{77} & 0 & -8 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 & 0 & 0 & -8 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 1 & -4 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ f_4 \\ f_5 \\ f_6 \\ f_7 \\ 0 \\ 3 \\ 1 \end{bmatrix}. \tag{9.20}$$

§9.3.3. *The Theory Behind Lagrange Multipliers

The recipe illustrated by (9.20) comes from a well known technique of variational calculus. Using the matrix notation introduced in §8.1.3, compactly denote the set of m MFCs by $\mathbf{Au} = \mathbf{b}$, 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

$$L(\mathbf{u}, \boldsymbol{\lambda}) = \Pi + \boldsymbol{\lambda}^T(\mathbf{Au} - \mathbf{b}) = \tfrac{1}{2}\mathbf{u}^T\mathbf{Ku} - \mathbf{u}^T\mathbf{f} + \boldsymbol{\lambda}^T(\mathbf{Au} - \mathbf{b}). \tag{9.21}$$

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

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \mathbf{b} \end{bmatrix}. \tag{9.22}$$

The master stiffness matrix $\mathbf{K}$ in (9.22) is said to be *bordered* with $\mathbf{A}$ and $\mathbf{A}^T$. Solving this system provides $\mathbf{u}$ and $\boldsymbol{\lambda}$. The latter can be interpreted as forces of constraint in the following sense: a removed constraint can be replaced by a system of forces characterized by $\boldsymbol{\lambda}$ multiplied by the constraint coefficients. More precisely, the constraint forces are $-\mathbf{A}^T\boldsymbol{\lambda}$.

§9.3.4. Assessment of the Lagrange Multiplier Method

In contrast to the penalty method, the method of Lagrange multipliers has the advantage of being exact (aside from computational errors due to finite precision arithmetic). It provides directly the constraint forces, which are of interest in many applications. It does not require guesses as regards weights. As the penalty method, it can be extended without difficulty to nonlinear constraints.

It is not free of disadvantages. It introduces additional unknowns, requiring expansion of the original stiffness method, and more complicated storage allocation procedures. It renders the augmented stiffness matrix indefinite, an effect that may cause grief with some linear equation solving methods that rely on positive definiteness. Finally, as the master-slave method, it is sensitive to the degree of linear independence of the constraints: if the constraint $u_2 = u_6$ is specified twice, the bordered stiffness is obviously singular.

On the whole this method appears to be the most elegant one for a general-purpose finite element program that is supposed to work as a "black box" by minimizing guesses and choices from its users. Its implementation, however, is not simple. Special care must be exercised to detect singularities due to constraint dependency and to account for the effect of loss of positive definiteness of the bordered stiffness on equation solvers.

§9.4. *The Augmented Lagrangian Method

The general matrix forms of the penalty function and Lagrangian multiplier methods are given by expressions (9.13) and (9.22), respectively. A useful connection between these methods can be established as follows.

Because the lower diagonal block of the bordered stiffness matrix in (9.22) is null, it is not possible to directly eliminate $\boldsymbol{\lambda}$. To make this possible, replace this block by $\epsilon\mathbf{S}^{-1}$, where $\mathbf{S}$ is a constraint-scaling diagonal matrix of appropriate order and ϵ is a small number. The reciprocal of ϵ is a large number called $w = 1/\epsilon$. To maintain exactness of the second equation, $\epsilon\mathbf{S}^{-1}\boldsymbol{\lambda}$ is added to the right-hand side:

$$\begin{bmatrix} \mathbf{K} & \mathbf{A}^T \\ \mathbf{A} & \epsilon\mathbf{S}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \mathbf{f} \\ \epsilon\mathbf{S}^{-1}\boldsymbol{\lambda}^P \end{bmatrix} \tag{9.23}$$

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

$$\begin{aligned} (\mathbf{K} + w\mathbf{A}^T\mathbf{S}\mathbf{A})\,\mathbf{u} &= \mathbf{f} + w\mathbf{A}^T\mathbf{S}\mathbf{b} - \mathbf{A}^T\boldsymbol{\lambda}^P, \\ \boldsymbol{\lambda} &= \boldsymbol{\lambda}^P + w\mathbf{S}(\mathbf{b} - \mathbf{A}\mathbf{u}), \end{aligned} \tag{9.24}$$

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

$$(\mathbf{K} + w\mathbf{A}^T\mathbf{S}\mathbf{A})\,\mathbf{u} = \mathbf{f} + w\mathbf{A}^T\mathbf{S}\mathbf{b}. \tag{9.25}$$

On taking $\mathbf{W} = w\mathbf{S}$, the general matrix equation (9.13) of the penalty method is recovered.

This relation suggests the construction of *iterative procedures* in which one tries to *improve the accuracy of the penalty function method while w is kept constant* [218]. This strategy circumvents the aforementioned ill-conditioning problems when the weight w is gradually increased. One such method is easily constructed by inspecting (9.24). Using superscript k as an iteration index and keeping w fixed, solve equations (9.24) in tandem as follows:

$$\begin{aligned} (\mathbf{K} + \mathbf{A}^T\mathbf{W}\mathbf{A})\,\mathbf{u}^k &= \mathbf{f} + \mathbf{A}^T\mathbf{W}\mathbf{b} - \mathbf{A}^T\boldsymbol{\lambda}^k, \\ \boldsymbol{\lambda}^{k+1} &= \boldsymbol{\lambda}^k - \mathbf{W}(\mathbf{b} - \mathbf{A}\mathbf{u}^k), \end{aligned} \tag{9.26}$$

for $k = 0, 1, \ldots$, beginning with $\boldsymbol{\lambda}^0 = \mathbf{0}$. Then $\mathbf{u}^0$ is the penalty solution. If the process converges one recovers the exact Lagrangian solution without having to solve the Lagrangian system (9.23) directly.

The family of iterative procedures that may be precipitated from (9.24) collectively pertains to the class of *augmented Lagrangian methods*. See **Notes and Bibliography** for references.

§9.5. Summary

The treatment of linear MFCs in finite element systems can be carried out by several methods. Three of these: master-slave elimination, penalty augmentation and Lagrange multiplier adjunction, have been discussed. It is emphasized that no method is uniformly satisfactory in terms of generality, robustness, numerical behavior and simplicity of implementation.

Figure 9.4 gives an assessment of the three techniques in terms of seven attributes.

For a general purpose program that tries to attain "black box" behavior (that is, minimal decisions on the part of users) the method of Lagrange multipliers has the edge. This edge is unfortunately blunted by a fairly complex computer implementation and by the loss of positive definiteness in the bordered stiffness matrix.

	Master-Slave Elimination	Penalty Function	Lagrange Multipliers
Generality	**fair**	**excellent**	**excellent**
Ease of implementation	**poor to fair**	**good**	**fair**
Sensitivity to user decisions	**high**	**high**	**small to none**
Accuracy	**variable**	**mediocre**	**excellent**
Sensitivity as regards constraint dependence	**high**	**none**	**high**
Retains positive definiteness	**yes**	**yes**	**no**
Modifies unknown vector	**yes**	**no**	**yes**

FIGURE 9.4. Assessment summary of three MFC application methods.

Notes and Bibliography

A form of the penalty function method, quite close to that described in §9.1.5, was first proposed by Courant in the early 1940s [156]. It entered the FEM through the work of researchers in the 1960s. There is a good description in the book by Zienkiewicz and Taylor [862].

The Lagrange Multiplier method is much older. Multipliers (called initially "coefficients") were described by Lagrange in his famous *Mécanique Analytique* monograph [446], as part of the procedure for forming the function now called the Lagrangian.[7] Its use in FEM is more recent than that of penalty methods, and is not clear where (and by whom) it was first used. It received a boost in the 1990 with the invention of the FETI method for domain decomposition solution procedures suitable to parallel processing [207,208].

Augmented Lagrangian methods received much attention since the late 1960s, when they originated in the field of constrained optimization. The original papers are by Hestenes [369] and Powell [613]. The use of the Augmented Lagrangian Multiplier method for FEM kinematic constraints is first discussed in [218], wherein the iterative algorithm (9.26) for the master stiffness equations is derived.

References

Referenced items have been moved to Appendix R.

[7] A quantity that characterizes the state of a system in terms of energy of its constituent components, augmented with multiplier-treated constraint terms if any. For a mechanical system the Lagrangian is the kinetic energy minus the total potential energy, plus any contributions from constraints. This function plays an essential role in variational formulations of mathematical physics.

Homework Exercises for Chapter 9
MultiFreedom Constraints II

EXERCISE 9.1 [C+N:20] This is identical to Exercise 8.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the penalty function method. Take the weight w to be 10^k, in which k varies as $k = 3, 4, 5, \ldots 16$. For each sample w compute the Euclidean-norm solution error $e(w) = ||\mathbf{u}^p(w) - \mathbf{u}^{ex}||_2$, where $\mathbf{u}^p$ is the computed solution and $\mathbf{u}^{ex}$ is the exact solution listed in (E8.1). Plot $k = \log_{10} w$ versus $\log_{10} e$ and report for which weight e attains a minimum. (See Slide #5 for a check). Does it roughly agree with the square root rule (§9.1.3) if the computations carry 16 digits of precision?

Figure E9.1 shows an implementation in pseudo code that solves this exercise. This can can be readily translated to a higher order language, such as *Mathematica*, *Matlab* or *Octave*, but notice that producing the log-log plot is highly language dependent.

```
K=MasterStiffnessOfSixElementBar(100); print K; // Defined in  Ex 8.1
f=vector(1,2,3,4,5,6,7); print f; // Force vector
uexact=vector(0.,0.27,0.275,0.25,0.185,0.07.0.14);  // Exact solution
ew=();  Array to receive weight vs error in loop below
for  w=100,w<=10^16,w=10*w,  // Increase w by 100 every pass
     Khat=K; fhat=f; Khat(2,2)+=w; Khat(6,6)+=w; Khat(2,6)=Khat(6,2)–=w;
     fhat(2)+=w/5; fhat(6)–=w/5;  // Inject penalty terms in stiffness and force
     ((Kmod,fmod))=FixLeftEndOfSixElementBar(Khat,fhat);  // Defined in Ex 8.1
     u=linearsolve(Kmod,fmod); // Use built-in linear solver of your language
     e=sqrt((u-uexact).(u-uexact)) ; // Take Euclidean error norm of solution
     appendto(ew,log10(w,e)) ; //  Store decimal logs of (weight,error) pair in ew
endfor;  print ew;   // Print stored w vs e  data
listplot(ew);  // Do  log-log plot of w vs e - this is very language dependent
```

FIGURE E9.1. Pseudo code to solve Exercise 9.1. Notes: reserved keywords appear in boldface, statements are terminated by semicolons, and `//` starts an inline comment a la `C++`.

An implementation of the foregoing pseudo code in *Mathematica* is listed in Figure E9.2. This implementation should work in all versions $\geq$ 4.0. In this script `MasterStiffnessOfSixElementBar` and `FixLeftEndOfSixElementBar` are modules listed in Exercise 8.1 of the previous chapter.

Remark 9.1. If you run the script of Figure E9.2 you may get several ominous looking error messages from *Mathematica* as it is processing some of the systems with very large weights. Don't be alarmed: those are only warnings. The `LinearSolve` function is alerting you that the penalty-augmented stiffness matrices $\hat{\mathbf{K}}$ for weights of order 10^{12} or bigger are ill-conditioned.

EXERCISE 9.2 [C+N:15] Again identical to Exercise 8.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the Lagrange multiplier method. The results for the computed **u** and the recovered force vector **Ku** should agree with (E8.1). A pseudo code script that solves this Exercise is shown in Figure E9.3.

An implementation of the pseudo code of Figure E9.3 in *Mathematica* is listed in Figure E9.4. This implementation should work in all versions $\geq$ 4.0. Comment on whether the computed solution agrees with that of (E8.1).

EXERCISE 9.3 [A:10] For the six-element example structure of Figure 8.2, describe which penalty elements would implement the following two MFCs:

$$u_2 + u_6 = 0, \qquad u_2 - 3u_6 = 1/3. \tag{E9.1}$$

```
(* Exercise 9.1 - Penalty Method, MFC: u2-u6=1/5, with variable w *)
K=MasterStiffnessOfSixElementBar[100]; (* Ex 8.1 *) f={1,2,3,4,5,6,7};
Print["Master stiffness K=",K//MatrixForm]; Print["Applied forces=",f];
uexact= {0,0.27,0.275,0.25,0.185,0.07,0.14}; ew={};
For [w=100, w<=10^16, w=10*w; (* increase w by 10 every pass *)
     Khat=K; fhat=f;
     Khat[[2,2]]+=w; Khat[[6,6]]+=w; Khat[[6,2]]=Khat[[2,6]]-=w;
     fhat[[2]]+=(1/5)*w; fhat[[6]]-=(1/5)*w; (*insert penalty *)
     {Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat]; (* Ex 8.1 *)
     u=uhat=LinearSolve[N[Kmod],N[fmod]];
     Print["Weight w=",N[w]//ScientificForm];
    (*Print["Weight w=",N[w]//ScientificForm," u=",u//InputForm];*)
     e=Sqrt[(u-uexact).(u-uexact)];
     Print["L2 solution error=",e//ScientificForm];
     AppendTo[ew,{Log[10,w],Log[10,e]}];
    ];
If [$VersionNumber<6.0, dfun=$DisplayFunction; JoinOpt=PlotJoined,
     dfun=Print; JoinOpt=Joined];
ListPlot[ew,AxesOrigin->{5,-8},Frame->True, PlotStyle->
  {AbsolutePointSize[4],AbsoluteThickness[2],RGBColor[1,0,0]},
   ImageSize->350, JoinOpt->True,DisplayFunction->dfun,
   AxesLabel->{"Log10(w)","Log10(u error)"}];
```

FIGURE E9.2. *Mathematica* implementation of pseudo code of Figure E9.1

```
K=MasterStiffnessOfSixElementBar(100); // Defined in Ex 8.1
Khat=nullmatrix(8,8); f=vector(1,2,3,4,5,6,7); fhat=append(f,0);
for i=1,i<=7,i++, for j=1,j<=7,j++, Khat(i,j)=K(i,j); endfor; endfor;
(Kmod,fmod)=FixLeftEndOfSixElementBar(Khat,fhat); // Defined in  Ex 8.1
Kmod(2,8)=Kmod(8,2)=1; Kmod(6,8)=Kmod(8,6)=–1; fmod(8)=1/5;
print Kmod; print fmod;  // Multiplier-augmented system
umod=linearsolve(Kmod,fmod);  // Solve multiplier-augmented system
u=umod(1:7); λ=umod(8); //  Separate displacement solution & Lagrange multiplier
print u; print λ;  // and display them
```

FIGURE E9.3. Pseudo code to solve Exercise 9.2.

```
(* Exercise 9.2 - Lagrange Multiplier Method, MFC: u2-u6=1/5 *)
K=MasterStiffnessOfSixElementBar[100]; (* from Ex 8.1 *)
Khat=Table [0,{8},{8}]; f={1,2,3,4,5,6,7}; fhat=AppendTo[f,0];
For [i=1,i<=7,i++, For [j=1,j<=7,j++, Khat[[i,j]]=K[[i,j]] ]];
{Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat]; (* from Ex 8.1 *)
Kmod[[2,8]]=Kmod[[8,2]]=1; Kmod[[6,8]]=Kmod[[8,6]]=-1; fmod[[8]]=1/5;
Print["Kmod=",Kmod//MatrixForm]; Print["fmod=",fmod];
umod=LinearSolve[N[Kmod],N[fmod]]; u=Take[umod,7];
Print["Solution u=",u,", λ=",umod[[8]]];
Print["Recovered node forces=",K.u];
```

FIGURE E9.4. *Mathematica* implementation of pseudo code of Figure E9.3

As answer, show the stiffness equations of those two elements in a manner similar to (9.1).

EXERCISE 9.4 [A/C+N:15+15+10] Suppose that the assembled stiffness equations for a one-dimensional finite element model before imposing constraints are

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}. \tag{E9.2}$$

This system is to be solved subject to the multipoint constraint

$$u_1 = u_3. \tag{E9.3}$$

(a) Impose the constraint (E9.3) by the master-slave method taking u_1 as master, and solve the resulting 2×2 system of equations by hand.

(b) Impose the constraint (E9.3) by the penalty function method, leaving the weight w as a free parameter. Solve the equations by hand or CAS (Cramer's rule is recommended) and verify analytically that as $w \to \infty$ 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 (E9.3) by the Lagrange multiplier method. Show the 4×4 multiplier-augmented system of equations analogous to (9.13) and solve it by computer or calculator.

EXERCISE 9.5 [A/C:10+15+10] The left end of the cantilevered beam-column member illustrated in Figure E9.5 rests on a skew-roller that forms a 45° angle with the horizontal axis x. The member is loaded axially by a force P as shown. The finite element equations upon removing the fixed right end freedoms $\{u_{x2}, u_{y2}, \theta_2\}$, but *before* imposing the skew-roller MFC, are

$$\begin{bmatrix} EA/L & 0 & 0 \\ 0 & 12EI/L^3 & 6EI/L^2 \\ 0 & 6EI/L^2 & 4EI/L \end{bmatrix} \begin{bmatrix} u_{x1} \\ u_{y1} \\ \theta_1 \end{bmatrix} = \begin{bmatrix} P \\ 0 \\ 0 \end{bmatrix}, \tag{E9.4}$$

where E, A, and $I = I_{zz}$ are given member properties, θ_1 is the left end rotation, and L is the member length.[8]

To simplify the calculations set $P = \alpha EA$, and $I = \beta AL^2$, in which α and β are dimensionless parameters, and express the following solutions in terms of α and β.

(a) Apply the skew-roller constraint by the master-slave method (make u_{y1} slave) and solve for u_{x1} and θ_1 in terms of L, α and β. This may be done by hand or a CAS. Partial solution: $u_{x1} = \alpha L/(1 + 3\beta)$.

(b) Apply the skew-roller constraint with the penalty method by inserting a penalty element at node 1. Follow the rule of §9.1.4 to construct the 2×2 penalty stiffness. Compute u_{x1} from the modified equations (Cramer's rule is recommended if solved by hand). Verify that as $w \to \infty$ the answer obtained in (a) is recovered. Partial solution: $u_{x1} = \alpha L(3EA\beta + wL)/(3EA\beta + wL(1+3\beta))$. Can the penalty stiffness be physically interpreted in some way?

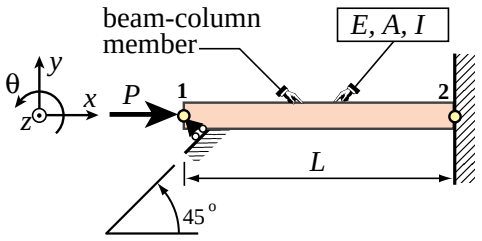


FIGURE E9.5. Cantilevered beam-column on skew-roller for Exercise 9.5.

(c) Apply the skew roller constraint by Lagrangian multiplier adjunction, and solve the resulting 4×4 system of equations using a CAS (by hand it will take long). Verify that you get the same solution as in (a).

[8] The stiffness equations for a beam column are derived in Part III of this book. For now consider (E9.4) as a recipe.

EXERCISE 9.6 [A:5+5+10+10+5] A cantiveler beam-column is to be joined to a plane stress plate mesh as depicted in Figure E9.6.[9] Both pieces move in the plane $\{x, y\}$. Plane stress elements have two degrees of freedom per node: two translations u_x and u_y along x and y, respectively, whereas a beam-column element has three: two translations u_x and u_y along x and y, and one rotation (positive CCW) θ_z about z. To connect the cantilever beam to the mesh, the following "gluing" conditions are applied:

(1) The horizontal (u_x) and vertical (u_y) displacements of the beam at their common node (2 of beam, 4 of plate) are the same.

(2) The beam end rotation θ_2 and the mean rotation of the plate edge 3–5 are the same. For infinitesimal displacements and rotations the latter is $\theta_{35}^{avg} = (u_{x5} - u_{x3})/H$.

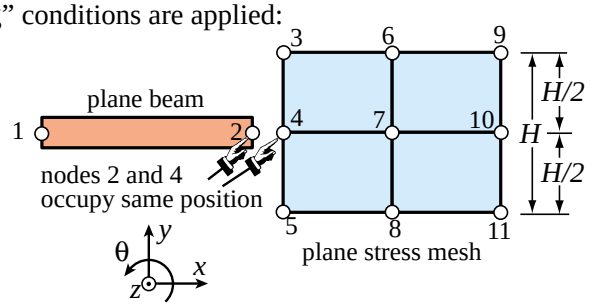


FIGURE E9.6. Beam linked to plate in plane stress for Exercise 9.6. Beam shown slightly separate from plate for visualization convenience: nodes 2 and 4 actually are at the same location.

Questions:

(a) Write down the three MFC conditions: two from (1) and one from (2), and state whether they are linear and homogeneous.

(b) Where does the above expression of θ_{35}^{avg} come from? (Geometric interpretation is OK.) Can it be made more accurate[10] by including u_{x4}?

(c) Write down the master-slave transformation matrix if $\{u_{x2}, u_{y2}, \theta_2\}$ are picked as slaves. It is sufficient to write down the transformation for the DOFs of nodes 2, 3, 4, and 5, which gives a $\mathbf{T}$ of order 9×6, since the transformations for the other freedoms are trivial.

(d) If the penalty method is used, write down the stiffness equations of the three penalty elements assuming the same weight w is used. Their stiffness matrices are of order 2×2, 2×2 and 3×3, respectively. (Do not proceed further)

(e) If Lagrange multiplier adjunction is used, how many Lagrange multipliers will you need to append? (Do not proceed further).

EXERCISE 9.7 [A:30] Show that the master-slave transformation method $\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}$ can be written down as a special form of the method of Lagrange multipliers. Start from the augmented functional

$$\Pi_{MS} = \tfrac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} + \boldsymbol{\lambda}^T(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \tag{E9.5}$$

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

EXERCISE 9.8 [A:35] Check the matrix equations (9.23) through (9.26) quoted for the Augmented Lagrangian method.

EXERCISE 9.9 [A:40] (Advanced, close to a research paper). Show that the master-slave transformation method $\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}$ can be expressed as a limit of the penalty function method as the weights go to infinity. Start from the augmented functional

$$\Pi_P = \tfrac{1}{2}\mathbf{u}^T\mathbf{K}\mathbf{u} - \mathbf{u}^T\mathbf{f} + \tfrac{1}{2}w(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}})^T(\mathbf{u} - \mathbf{T}\hat{\mathbf{u}}) \tag{E9.6}$$

[9] This is extracted from a question previously given in the Aerospace Ph. D. Preliminary Exam. Technically it is not difficult once the student understand what is being asked, but may cause panic in an eventful exam. Understanding can take some time, but a HW is more relaxed.

[10] To answer the second question, observe that the displacements along 3–4 and 4–5 vary linearly. Thus the angle of rotation about z is constant for each of them, and (for infinitesimal displacements) may be set equal to the tangent.

Write down the matrix stationarity conditions with respect to to $\mathbf{u}$ and $\hat{\mathbf{u}}$ and take the limit $w \to \infty$. *Hint*: using Woodbury's formula (Appendix C, §C.5.2)

$$(\mathbf{K} + w\mathbf{T}^T\mathbf{S}\mathbf{T})^{-1} = \mathbf{K}^{-1} - \mathbf{K}^{-1}\mathbf{T}^T(\overline{\mathbf{K}} + w^{-1}\mathbf{S}^{-1})^{-1}\mathbf{T}\,\mathbf{K}^{-1}. \tag{E9.7}$$

show that

$$\overline{\mathbf{K}}^{-1} = \mathbf{T}\mathbf{K}^{-1}\mathbf{T}^T. \tag{E9.8}$$