1
Overview

TABLE OF CONTENTS

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

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

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

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

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

§1.1. WHERE THIS MATERIAL FITS

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

$$\text{Mechanics} \left\{ \begin{array}{l} \textit{Theoretical} \\ \textit{Applied} \\ \textit{Computational} \end{array} \right. \tag{1.1}$$

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

REMARK 1.1

Paraphrasing an old joke about mathematicians, one may define a computational mechanician as a person who searches for solutions to given problems, an applied mechanician as a person who searches for problems that fit given solutions, and a theoretical mechanician as a person who can prove the existence of problems and solutions.

§1.1.1. Computational Mechanics

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

$$\text{Computational Mechanics} \left\{ \begin{array}{l} \textit{Nanomechanics and micromechanics} \\ \textit{Continuum mechanics} \left\{ \begin{array}{l} \text{Solids and Structures} \\ \text{Fluids} \\ \text{Multiphysics} \end{array} \right. \\ \textit{Systems} \end{array} \right. \tag{1.2}$$

Nanomechanics deals with phenomena at the molecular and atomic levels of matter. As such it is closely interrelated with particle physics and chemistry. Micromechanics looks primarily at the crystallographic and granular levels of matter. Its main technological application is the design and fabrication of materials and microdevices.

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

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

Multiphysics is a more recent newcomer. This area is meant to include mechanical systems that transcend the classical boundaries of solid and fluid mechanics, as in interacting fluids and structures. Phase change problems such as ice melting and metal solidification fit into this category, as do the interaction of control, mechanical and electromagnetic systems.

Finally, *system* identifies mechanical objects, whether natural or artificial, that perform a distinguishable function. Examples of man-made systems are airplanes, buildings, bridges, engines, cars, microchips, radio telescopes, robots, roller skates and garden sprinklers. Biological systems, such as a whale, amoeba or pine tree are included if studied from the viewpoint of biomechanics. Ecological, astronomical and cosmological entities also form systems.[1]

In this progression of (1.2) the *system* is the most general concept. A system is studied by *decomposition*: its behavior is that of its components plus the interaction between the components. Components are broken down into subcomponents and so on. As this hierarchical process continues the individual components become simple enough to be treated by individual disciplines, but their interactions may get more complex. Consequently there is a tradeoff art in deciding where to stop.[2]

§1.1.2. Statics vs. Dynamics

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

$$\text{Continuum mechanics} \left\{ \begin{array}{l} \textit{Statics} \\ \textit{Dynamics} \end{array} \right. \tag{1.3}$$

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

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

[1] Except that their function may not be clear to us. "The usual approach of science of constructing a mathematical model cannot answer the questions of why there should be a universe for the model to describe. Why does the universe go to all the bother of existing?" (Stephen Hawking).

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

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

§1.1.3. Linear vs. Nonlinear

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

$$\text{Statics} \left\{ \begin{array}{l} \textit{Linear} \\ \textit{Nonlinear} \end{array} \right.$$

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

§1.1.4. Discretization methods

A final classification of CSM static analysis is based on the discretization method by which the continuum mathematical model is *discretized* in space, *i.e.*, converted to a discrete model of finite number of degrees of freedom:

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

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

Classical finite difference methods in solid and structural mechanics have virtually disappeared from practical use. This statement is not true, however, for fluid mechanics, where finite difference discretization methods are still important. Finite-volume methods, which address finite volume method conservation laws, are important in highly nonlinear problems of fluid mechanics. Spectral methods are based on transforms that map space and/or time dimensions to spaces where the problem is easier to solve.

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

§1.1.5. FEM Variants

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

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

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

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

§1.2. WHAT DOES A FINITE ELEMENT LOOK LIKE?

The subject of this book is FEM. But what is a finite element? The concept will be partly illustrated through a truly ancient problem: find the perimeter L of a circle of diameter d. Since $L = \pi\, d$, this is equivalent to obtaining a numerical value for π.

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

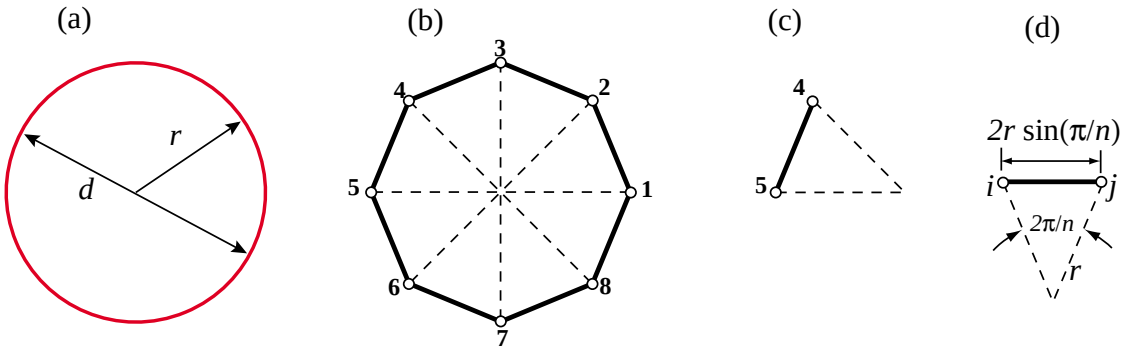


Figure 1.1. The "find π" problem treated with FEM concepts: (a) continuum object, (b) a discrete approximation (inscribed regular polygon), (c) disconnected element, (d), generic element.

Values of π_n obtained for $n = 1, 2, 4, \ldots 256$ are listed in the second column of Table 1.1. As can be seen the convergence to π is fairly slow. However, the sequence can be transformed by Wynn's ϵ algorithm[3] into that shown in the third column. The last value displays 15-place accuracy.

Some of the key ideas behind the FEM can be identified in this simple example. The circle, viewed as a *source mathematical object*, is replaced by polygons. These are *discrete approximations* to the circle. The sides, renamed as *elements*, are specified by their end *nodes*. Elements can be separated by disconnecting the nodes, a process called *disassembly* in the FEM. Upon disassembly

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

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

n	$\pi_n = n\sin(\pi/n)$	Extrapolated by Wynn-ϵ	Exact π to 16 places
1	0.000000000000000		
2	2.000000000000000		
4	2.828427124746190	3.414213562373096	
8	3.061467458920718		
16	3.121445152258052	3.141418327933211	
32	3.136548490545939		
64	3.140331156954753	3.141592658918053	
128	3.141277250932773		
256	3.141513801144301	3.141592653589786	3.141592653589793

a *generic element* can be defined, *independently of the original circle*, by the segment that connects two nodes i and j. The relevant element property: length L_{ij}, can be computed in the generic element independently of the others, a property called *local support* in the FEM. Finally, the desired property: the polygon perimeter, is obtained by reconnecting n elements and adding up their length; the corresponding steps in the FEM being *assembly* and *solution*, respectively. There is of course nothing magic about the circle; the same technique can be be used to rectify any smooth plane curve.[4]

This example has been offered in the FEM literature to aduce that finite element ideas can be traced to Egyptian mathematicians from *circa* 1800 B.C., as well as Archimedes' famous studies on circle rectification by 250 B.C. But comparison with the modern FEM, as covered in Chapters 2–3, shows this to be a stretch. The example does not illustrate the concept of degrees of freedom, conjugate quantities and local-global coordinates. It is guilty of circular reasoning: the compact formula $\pi = \lim_{n\to\infty} n\sin(\pi/n)$ uses the unknown π in the right hand side.[5] Reasonable people would argue that a circle is a simpler object than, say, a 128-sided polygon. Despite these flaws the example is useful in one respect: showing a fielder's choice in the replacement of one mathematical object by another. This is at the root of the simulation process described in the next section.

§1.3. THE FEM ANALYSIS PROCESS

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

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

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

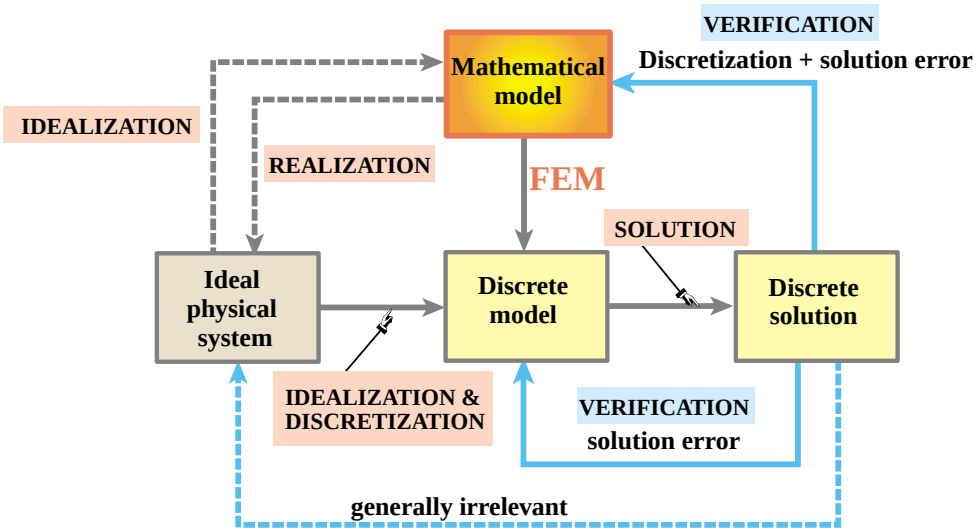


Figure 1.2. The Mathematical FEM. The mathematical model (top) is the source of the simulation process. Discrete model and solution follow from it. The ideal physical system (should one go to the trouble of exhibiting it) is inessential.

§1.3.1. The Mathematical FEM

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

On the left Figure 1.2 shows an *ideal physical system*. This may be presented as a *realization* of the mathematical model; conversely, the mathematical model is said to be an *idealization* of this system. For example, if the mathematical model is the Poisson's equation, realizations may be a heat conduction or a electrostatic charge distribution problem. This step is inessential and may be left out. Indeed FEM discretizations may be constructed without any reference to physics.

The concept of *error* arises when the discrete solution is substituted in the "model" boxes. This replacement is generically called *verification*. The *solution error* is the amount by which the discrete solution fails to satisfy the discrete equations. This error is relatively unimportant when using computers, and in particular direct linear equation solvers, for the solution step. More relevant is the *discretization error*, which is the amount by which the discrete solution fails to satisfy the mathematical model.[7] Replacing into the ideal physical system would in principle quantify modeling errors. In the mathematical FEM this is largely irrelevant, however, because the ideal physical system is merely that: a figment of the imagination.

---

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

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

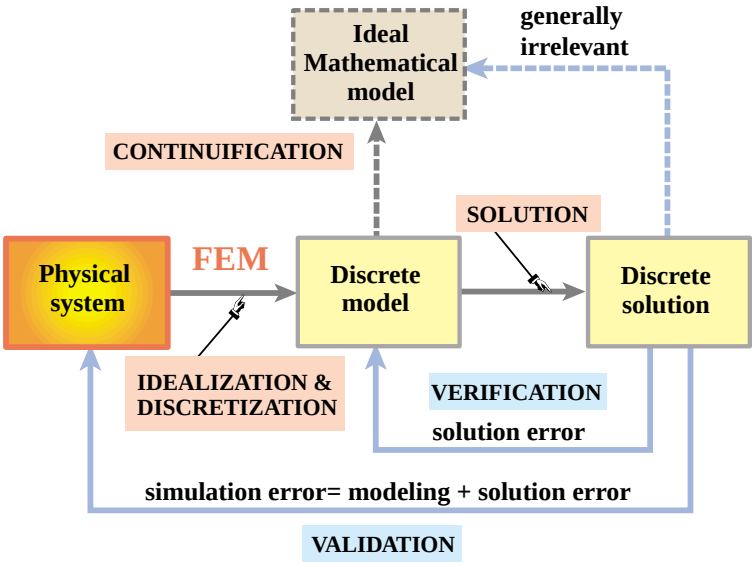


Figure 1.3. The Physical FEM. The physical system (left) is the source of the simulation process. The ideal mathematical model (should one go to the trouble of constructing it) is inessential.

§1.3.2. The Physical FEM

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

Just like Figure 1.2 shows an ideal physical system, 1.3 depicts an *ideal mathematical model*. This may be presented as a *continuum limit* or "continuification" of the discrete model. For some physical systems, notably those well modeled by continuum fields, this step is useful. For others, such as complex engineering systems, it makes no sense. Indeed FEM discretizations may be constructed and adjusted without reference to mathematical models, simply from experimental measurements.

The concept of *error* arises in the Physical FEM in two ways, known as *verification* and *validation*, respectively. Verification is the same as in the Mathematical FEM: the discrete solution is replaced into the discrete model to get the solution error. As noted above this error is not generally important. Substitution in the ideal mathematical model in principle provides the discretization error. This is rarely useful in complex engineering systems, however, because there is no reason to expect that the mathematical model exists, and if it does, that it is more physically relevant than the discrete model. Validation tries to compare the discrete solution against observation by computing the *simulation error*, which combines modeling and solution errors. As the latter is typically insignificant, the simulation error in practice can be identified with the modeling error.

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

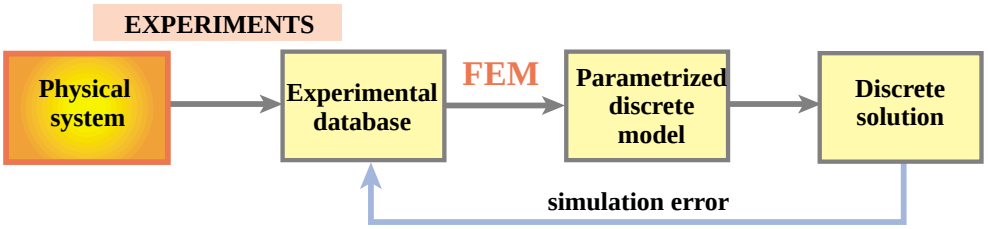


Figure 1.4. Model updating process in the Physical FEM.

§1.3.3. Synergy of Physical and Mathematical FEM

The foregoing physical and mathematical sequences are not exclusive but complementary. This synergy[8] is one of the reasons behind the power and acceptance of the method. Historically the Physical FEM was the first one to be developed to model very complex systems such as aircraft, as narrated in Appendix H. The Mathematical FEM came later and, among other things, provided the necessary theoretical underpinnings to extend FEM beyond structural analysis.

A glance at the schematics of a commercial jet aircraft makes obvious the reasons behind the physical FEM. There is no differential equation that captures, at a continuum mechanics level,[9] the structure, avionics, fuel, propulsion, cargo, and passengers eating dinner.

There is no reason for despair, however. The time honored *divide and conquer* strategy, coupled with *abstraction*, comes to the rescue. First, separate the structure and view the rest as masses and forces, most of which are time-varying and nondeterministic. Second, consider the aircraft structure as built of *substructures*:[10] wings, fuselage, stabilizers, engines, landing gears, and so on. Take each substructure, and continue to decompose it into *components*: rings, ribs, spars, cover plates, actuators, etc, continuing through as many levels as necessary. Eventually those components become sufficiently simple in geometry and connectivity that they can be reasonably well described by the continuum mathematical models provided, for instance, by Mechanics of Materials or the Theory of Elasticity. At that point, *stop*. The component level discrete equations are obtained from a FEM library based on the mathematical model. The system model is obtained by going through the reverse process: from component equations to substructure equations, and from those to the equations of the complete aircraft. This *system assembly* process is governed by the classical principles of Newtonian mechanics expressed in conservation form.

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

---

[8] This interplay is not exactly a new idea: "The men of experiment are like the ant, they only collect and use; the reasoners resemble spiders, who make cobwebs out of their own substance. But the bee takes the middle course: it gathers its material from the flowers of the garden and field, but transforms and digests it by a power of its own." (Francis Bacon, 1620).

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

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

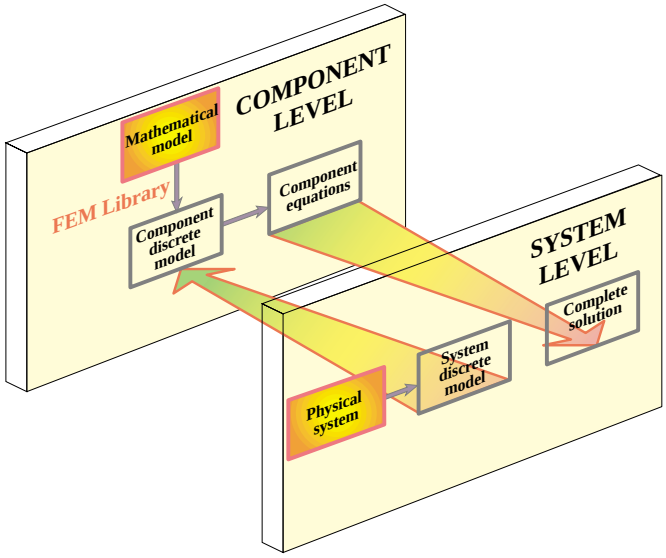


Figure 1.5. Combining physical and mathematical modeling through multilevel FEM. Only two levels (system and component) are shown for simplicity; intermediate substructure levels are omitted.

REMARK 1.2

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

REMARK 1.3

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

For sufficiently simple structures, passing to a discrete model is carried out in a single *idealization and discretization* step, as illustrated for the truss roof structure shown in Figure 1.6. Multiple levels are unnecessary here. Of course the truss may be viewed as a substructure of the roof, and the roof as a a substructure of a building.

§1.4. INTERPRETATIONS OF THE FINITE ELEMENT METHOD

Just like there are two complementary ways of using the FEM, there are two complementary interpretations for teaching it. One interpretation stresses the physical significance and is aligned with the Physical FEM. The other focuses on the mathematical context, and is aligned with the Mathematical FEM.

§1.4.1. Physical Interpretation

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

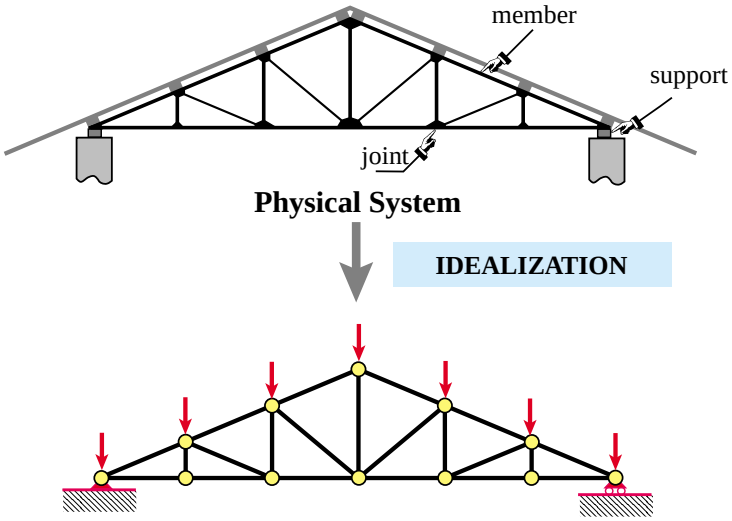


Figure 1.6. The idealization process for a simple structure. The physical system, here a roof truss, is directly idealized by the mathematical model: a pin-jointed bar assembly. For this particular structure, the idealization coalesces with the discrete model.

is reflected in the use of structural terms such as "stiffness matrix", "force vector" and "degrees of freedom." This terminology carries over to non-structural applications.

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

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

As discussed in §1.3, the underlying theme is *divide and conquer*. If the behavior of a system is too complex, the recipe is to divide it into more manageable subsystems. If these subsystems are still too complex the subdivision process is continued until the behavior of each subsystem is simple enough to fit a mathematical model that represents well the knowledge level the analyst is interested in. In the finite element method such "primitive pieces" are called *elements*. The behavior of the total system is that of the individual elements plus their interaction. A key factor in the initial acceptance of the FEM was that the element interaction can be physically interpreted and understood in terms that were eminently familiar to structural engineers.

§1.4.2. Mathematical Interpretation

This interpretation is closely aligned with the configuration of Figure 1.2. The FEM is viewed as a procedure for obtaining numerical approximations to the solution of boundary value problems

(BVPs) posed over a domain Ω. This domain is replaced by the union $\cup$ of disjoint subdomains $\Omega^{(e)}$ called finite elements. In general the geometry of Ω is only approximated by that of $\cup\Omega^{(e)}$.

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

REMARK 1.4

In the mathematical interpretation the emphasis is on the concept of *local (piecewise) approximation*. The concept of element-by-element breakdown and assembly, while convenient in the computer implementation, is not theoretically necessary. The mathematical interpretation permits a general approach to the questions of convergence, error bounds, trial and shape function requirements, etc., which the physical approach leaves unanswered. It also facilitates the application of FEM to classes of problems that are not so readily amenable to physical visualization as structures; for example electromagnetics and thermal conduction.

REMARK 1.5

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

§1.5. KEEPING THE COURSE

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

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

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

In Chapters 2 through 6 the time is frozen at about 1965, and the DSM presented as an aerospace engineer of that time would have understood it. This is not done for sentimental reasons, although that happens to be the year in which the writer began his thesis work on FEM under Ray Clough. Virtually all finite element codes are now based on the DSM and the computer implementation has not essentially changed since the late 1960s.

§1.6. *WHAT IS NOT COVERED

The following topics are not covered in this book:

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

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

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

Homework Exercises for Chapter 1
Overview

EXERCISE 1.1

[A:15] Do Archimedes' problem using a circumscribed regular polygon, with $n = 1, 2, \ldots 256$. Does the sequence converge any faster?

EXERCISE 1.2

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

2

The Direct Stiffness Method: Breakdown

TABLE OF CONTENTS

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

The exposition is done by following the DSM steps applied to a simple plane truss structure.

§2.1. WHY A PLANE TRUSS?

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

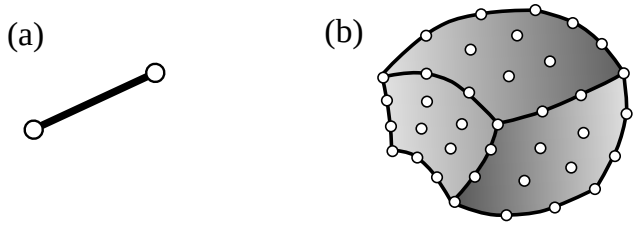


Figure 2.1. From the simplest to a highly complex structural finite element: (a) 2-node bar element for trusses, (b) 64-node tricubic, curved "brick" element for three-dimensional solid analysis.

Yet the remarkable fact is that, in the DSM, the simplest and most complex elements are treated alike! To illustrate the basic steps of this democratic method, it makes educational sense to keep it simple and use a structure composed of bar elements. A simple yet nontrivial structure is the *pin-jointed plane truss*.[1]

Using a plane truss to teach the stiffness method offers two additional advantages:

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

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

§2.2. TRUSS STRUCTURES

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

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

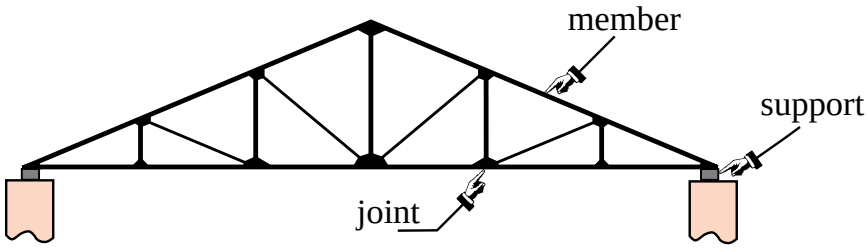


Figure 2.2. An actual plane truss structure. That shown is typical of a roof truss used in residential building construction.

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

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

In this and the following Chapter we will go over the basic steps of the DSM in a "hand-computer" calculation mode. This means that although the steps are done by hand, whenever there is a procedural choice we shall either adopt the way which is better suited towards the computer implementation, or explain the difference between hand and computer computations. The actual computer implementation using a high-level programming language is presented in Chapter 5.

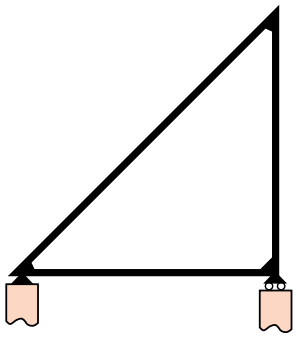


Figure 2.3. The example plane truss structure, called "example truss" in the sequel. It has three members and three joints.

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

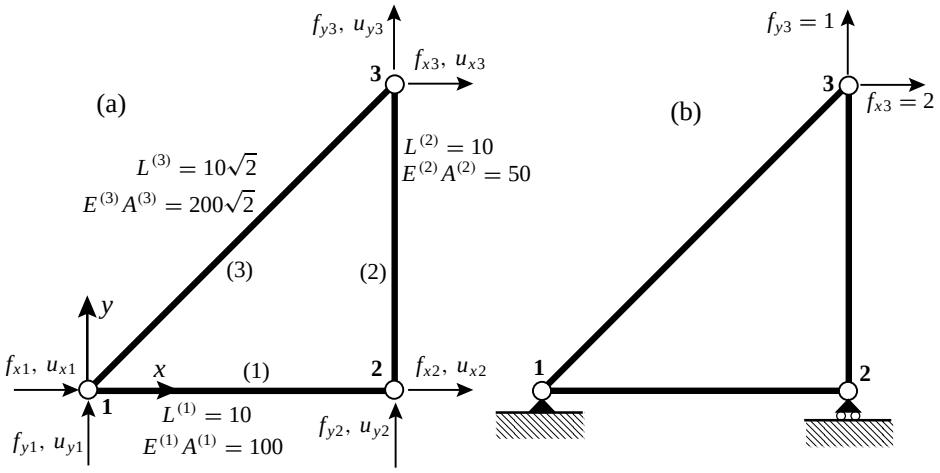


Figure 2.4. Pin-jointed idealization of example truss: (a) geometric and elastic properties, (b) support conditions and applied loads.

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

§2.3. IDEALIZATION

Although the pin-jointed assemblage of bars (as depicted in Figure 2.4) is sometimes presented as a real problem, it actually represents an *idealization* of a true truss structure. The axially-carrying members and frictionless pins of this structure are only an approximation of a real truss. For example, building and bridge trusses usually have members joined to each other through the use of gusset plates, which are attached by nails, bolts, rivets or welds; see Figure 2.2. Consequently members will carry some bending as well as direct axial loading.

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

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

§2.4. JOINT FORCES AND DISPLACEMENTS

The example truss shown in Figure 2.3 has three *joints*, which are labeled 1, 2 and 3, and three *members*, which are labeled (1), (2) and (3). These members connect joints 1–2, 2–3, and 1–3, respectively. The member lengths are denoted by $L^{(1)}$, $L^{(2)}$ and $L^{(3)}$, their elastic moduli by $E^{(1)}$, $E^{(2)}$ and $E^{(3)}$, and their cross-sectional areas by $A^{(1)}$, $A^{(2)}$ and $A^{(3)}$. Both E and A are assumed to be constant along each member.

Members are generically identified by index e (because of their close relation to finite elements, see below), which is usually enclosed in parentheses to avoid confusion with exponents. For example,

the cross-section area of a generic member is $A^{(e)}$. Joints are generically identified by indices such as i, j or n. In the general FEM, the name "joint" and "member" is replaced by *node* and *element*, respectively. The dual nomenclature is used in the initial Chapters to stress the physical interpretation of the FEM.

The geometry of the structure is referred to a common Cartesian coordinate system $\{x, y\}$, which is called the *global coordinate system*. Other names for it in the literature are *structure coordinate system* and *overall coordinate system*.

The key ingredients of the stiffness method of analysis are the *forces* and *displacements* at the joints.

In a idealized pin-jointed truss, externally applied forces as well as reactions *can act only at the joints*. All member axial forces can be characterized by the x and y components of these forces, which we call f_x and f_y, respectively. The components at joint i will be denoted as f_{xi} and f_{yi}, respectively. The set of all joint forces can be arranged as a 6-component column vector:

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

The other key ingredient is the displacement field. Classical structural mechanics tells us that the displacements of the truss *are completely defined by the displacements of the joints*. This statement is a particular case of the more general finite element theory.

The x and y displacement components will be denoted by u_x and u_y, respectively. The *values* of u_x and u_y at joint i will be called u_{xi} and u_{yi} and, like the joint forces, they are arranged into a 6-component vector:

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

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

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

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

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

§2.5. THE MASTER STIFFNESS EQUATIONS

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

Because the assumed behavior of the truss is linear, these equations must be linear relations that connect the components of the two vectors. Furthermore it will be assumed that if all displacements vanish, so do the forces.[3]

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

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

In matrix notation:

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

Here **K** is the *master stiffness matrix*, also called *global stiffness matrix*, *assembled stiffness matrix*, or *overall stiffness matrix*. It is a 6×6 square matrix that happens to be symmetric, although this attribute has not been emphasized in the written-out form (2.3). The entries of the stiffness matrix are often called *stiffness coefficients* and have a physical interpretation discussed below.

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

REMARK 2.1

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

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

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

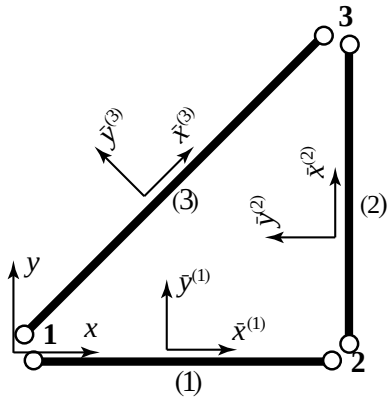


Figure 2.5. Breakdown of example truss into individual members (1), (2) and (3), and selection of local coordinate systems.

Thus K_{y1x2}, say, represents the y-force at joint 1 that would arise on prescribing a unit x-displacement at joint 2, while all other displacements vanish.

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

§2.6. BREAKDOWN

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

§2.6.1. Disconnection

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

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

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

§2.6.2. Localization

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

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

§2.6.3. Computation of Member Stiffness Equations

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

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

which written out in full is

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

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

There are several ways to construct the stiffness matrix $\bar{\mathbf{K}}$ in terms of the element properties L, E and A. The most straightforward technique relies on the Mechanics of Materials approach covered in undergraduate courses. Think of the truss member in Figure 2.6(a) as a linear spring of equivalent stiffness k_s, an interpretation depicted in Figure 2.7(b). If the member properties are *uniform* along its length, Mechanics of Materials bar theory tells us that[4]

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

Consequently the force-displacement equation is

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

where F is the internal axial force and d the relative axial displacement, which physically is the bar elongation.

The axial force and elongation can be immediately expressed in terms of the joint forces and displacements as

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

which express force equilibrium[5] and kinematic compatibility, respectively.

Combining (2.9) and (2.10) we obtain the matrix relation[6]

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

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

[5] 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 this on joint j yields $F = \bar{f}_{xj}$.

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

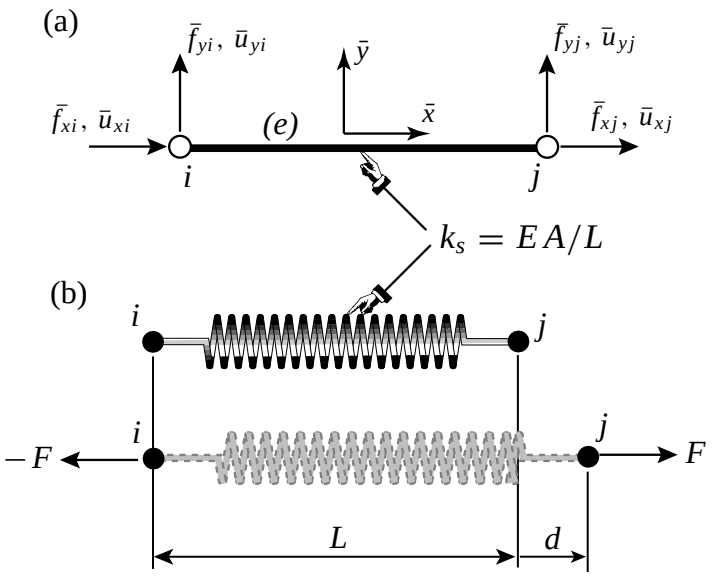


Figure 2.6. Generic truss member referred to its local coordinate system $\{\bar{x}, \bar{y}\}$: (a) idealization as bar element, (b) interpretation as equivalent spring.

Hence

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

This is the truss stiffness matrix in local coordinates.

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

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

Homework Exercises for Chapter 2
The Direct Stiffness Method: Breakdown

EXERCISE 2.1

[D:5] Explain why *arbitrarily oriented* mechanical loads on an *idealized* pin-jointed truss structure must be applied at the joints. [Hint: idealized truss members have no bending resistance.] How about actual trusses: can they take loads applied between joints?

EXERCISE 2.2

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

EXERCISE 2.3

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

EXERCISE 2.4

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

EXERCISE 2.5

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

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

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

EXERCISE 2.6

[A:20] Derive the equivalent spring formula $F = (EA/L)\, d$ of (2.9) by the Theory of Elasticity relations $e = d\bar{u}(\bar{x})/d\bar{x}$ (strain-displacement equation), $\sigma = Ee$ (Hooke's law) and $F = A\sigma$ (axial force definition). Here e is the axial strain (independent of $\bar{x}$) and σ the axial stress (also independent of $\bar{x}$). Finally, $\bar{u}(\bar{x})$ denotes the axial displacement of the cross section at a distance $\bar{x}$ from node i, which is linearly interpolated as

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

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

EXERCISE 2.7

[A:20] Derive the equivalent spring formula $F = (EA/L)\, d$ of (2.9) by the principle of Minimum Potential Energy (MPE). In Mechanics of Materials it is shown that the total potential energy of the axially loaded bar is

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

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

3

The Direct Stiffness Method: Assembly and Solution

TABLE OF CONTENTS

§3.1. INTRODUCTION

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

Assembly involves two substeps: *globalization*, through which the member stiffness equations are transformed back to the global coordinate system, and *merging* of those equations into the global stiffness equations. On the computer these steps are done concurrently, member by member. After all members are processed we have the *free-free master stiffness equations*.

Next comes the *solution*. This process also embodies two substeps: *application of boundary conditions* and *solution* for the unknown joint displacements. First, the free-free master stiffness equations are modified by taking into account which components of the joint displacements and forces are given and which are unknown. Then the modified equations are submitted to a linear equation solver, which returns the unknown joint displacements. On some equation solvers, both operations are done concurrently.

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

§3.2. ASSEMBLY

§3.2.1. Coordinate Transformations

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

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

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

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

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

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

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

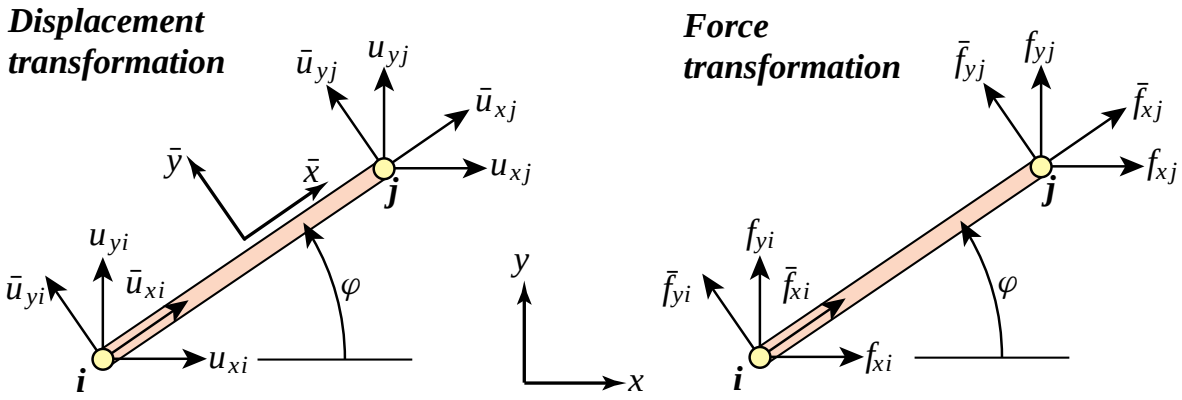


Figure 3.1. The transformation of node displacement and force components from the local system $\{\bar{x}, \bar{y}\}$ to the global system $\{x, y\}$.

The 4×4 matrix that appears above is called a *force transformation matrix*. A comparison of (3.2) and (3.3) reveals that the force transformation matrix is the *transpose* $\mathbf{T}^T$ of the displacement transformation matrix $\mathbf{T}$. This relation is not accidental and can be proved to hold generally.[2]

REMARK 3.1

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

REMARK 3.2

For this particular structural element $\mathbf{T}$ is square and orthogonal, that is, $\mathbf{T}^T = \mathbf{T}^{-1}$. But this property does not extend to more general elements. Furthermore in the general case $\mathbf{T}$ is not even a square matrix, and does not possess an ordinary inverse. However the congruential transformation relations (3.4)-(3.6) hold generally.

§3.2.2. Globalization

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

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

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

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

Inserting these matrix expressions into (2.6) and comparing with (3.4) we find that the member stiffness in the global system $\{x, y\}$ can be computed from the member stiffness $\bar{\mathbf{K}}^{(e)}$ in the local

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

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

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

Carrying out the matrix multiplications we get

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

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

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

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

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

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

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

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

§3.2.3. Assembly Rules

The key operation of the assembly process is the "placement" of the contribution of each member to the master stiffness equations. The process is technically called *merging* of individual members. The merge operation can be physically interpreted as reconnecting that member in the process of fabricating the complete structure. For a truss structure, reconnection means inserting the pins back into the joints. This is mathematically governed by two rules of structural mechanics:

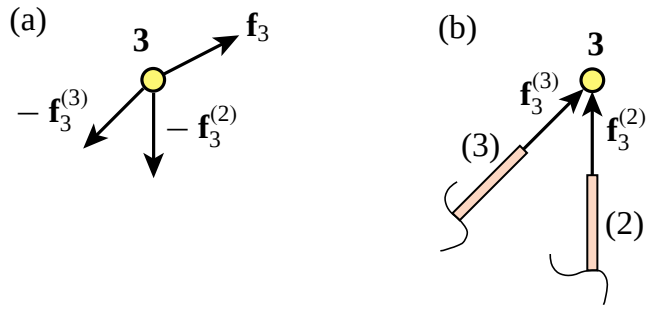


Figure 3.2. The force equilibrium of joint 3 of the example truss, depicted as a free body diagram in (a). Here $\mathbf{f}_3$ is the known external joint force applied on the joint. Joint forces $\mathbf{f}_3^{(2)}$ and $\mathbf{f}_3^{(3)}$ are applied *by the joint on the members*, as illustrated in (b). Consequently the forces applied *by the members on the joint* are $-\mathbf{f}_3^{(2)}$ and $-\mathbf{f}_3^{(3)}$. These forces would act in the directions shown if both members (2) and (3) were in tension. The free-body equilibrium statement is $\mathbf{f}_3 - \mathbf{f}_3^{(2)} - \mathbf{f}_3^{(3)} = \mathbf{0}$ or $\mathbf{f}_3 = \mathbf{f}_3^{(2)} + \mathbf{f}_3^{(3)}$. This translates into the two component equations: $f_{x3} = f_{x3}^{(2)} + f_{x3}^{(3)}$ and $f_{y3} = f_{y3}^{(2)} + f_{y3}^{(3)}$, of (3.11).

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

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

$$\begin{aligned} &\text{Rule 1:} \qquad u_{x3}^{(2)} = u_{x3}^{(3)}, \quad u_{y3}^{(2)} = u_{y3}^{(3)}. \\ &\text{Rule 2:} \quad f_{x3} = f_{x3}^{(2)} + f_{x3}^{(3)} = f_{x3}^{(1)} + f_{x3}^{(2)} + f_{x3}^{(3)}, \quad f_{y3} = f_{y3}^{(2)} + f_{y3}^{(3)} = f_{y3}^{(1)} + f_{y3}^{(2)} + f_{y3}^{(3)}. \end{aligned} \tag{3.11}$$

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

§3.2.4. Hand Assembly by Augmentation and Merge

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

For member (1):

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

For member (2):

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

For member (3):

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

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

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

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

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

These three equations can be represented in direct matrix notation as

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

According to the second rule

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

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

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

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

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

§3.3. SOLUTION

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

§3.3.1. Applying Displacement BCs by Reduction

If one attempts to solve the system (3.20) numerically for the displacements, surprise! The solution "blows up" because the coefficient matrix (the master stiffness matrix) is singular. The mathematical interpretation of this behavior is that rows and columns of **K** are linear combinations of each other (see Remark 3.4 below). The physical interpretation of singularity is that there are still unsuppressed *rigid body motions*: the truss "floats" in the $\{x, y\}$ plane.

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

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

whereas the known applied forces are

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

When solving the stiffness equations by hand, the simplest way to account for support conditions is to *remove* equations associated with known joint displacements from the master system. To apply (3.21) we have to remove equations 1, 2 and 4. This can be systematically accomplished by *deleting* or "striking out" rows and columns number 1, 2 and 4 from **K** and the corresponding components from **f** and **u**. The reduced three-equation system is

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

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

REMARK 3.3

In mathematical terms, the free-free master stiffness matrix **K** in (3.20) has order $N = 6$, rank $r = 3$ and a rank deficiency of $d = N - r = 6 - 3 = 3$ (these concepts are summarized in Appendix C.) The dimension of the null space of **K** is $d = 3$. This space is spanned by three independent rigid body motions: the two rigid translations along x and y and the rigid rotation about z.

REMARK 3.4

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

§3.3.2. Solving for Displacements

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

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

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

(3.21):

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

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

§3.4. POSTPROCESSING

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

§3.4.1. Recovery of Reaction Forces

Premultiplying the complete displacement solution (3.25) by **K** we get

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

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

§3.4.2. Recovery of Internal Forces and Stresses

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

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

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

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

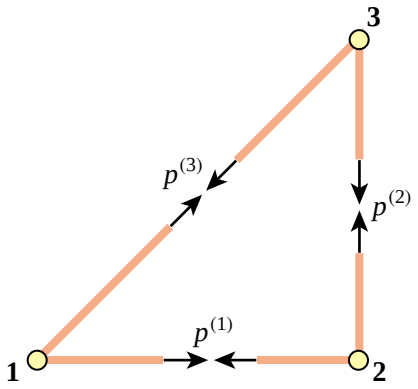


Figure 3.3. The internal forces in the example truss are the axial forces $p^{(1)}$, $p^{(2)}$ and $p^{(3)}$ in the members. Signs shown for these forces correspond to tension.

An alternative interpretation of (3.27) is to regard $e^{(e)} = d^{(e)}/L^{(e)}$ as the (average) member axial strain, $\sigma^{(e)} = E^{(e)}e^{(e)}$ as (average) axial stress, and $p^{(e)} = A^{(e)}\sigma^{(e)}$ as the axial force. This is more in tune with the Theory of Elasticity viewpoint discussed in Exercise 2.6.

§3.5. *COMPUTER ORIENTED ASSEMBLY AND SOLUTION

§3.5.1. *Assembly by Freedom Pointers

The practical computer implementation of the DSM assembly process departs significantly from the "augment and add" technique described in §3.2.4. There are two major differences:

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

(II) The master stiffness matrix **K** is stored using a special format that takes advantage of symmetry and sparseness.

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

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

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

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

Physically these tables map local freedom indices to global ones. For example, freedom number 3 of member (2) is u_{x3}, which is number 5 in the master equations; consequently $\text{EFT}^{(2)}(3) = 5$. Note that (3.28) involves 3 nested loops: over e (outermost), over i, and over j. The ordering of the last two is irrelevant. Advantage may be taken of the symmetry of $\mathbf{K}^{(e)}$ and **K** to roughly halve the number of additions. Exercise 3.6 follows the process (3.28) by hand.

§3.5.2. *Applying Displacement BCs by Modification

In §3.3.1 the support conditions (3.21) were applied by reducing (3.20) to (3.23). Reduction is convenient for hand computations because it cuts down on the number of equations to solve. But it has a serious flaw for computer implementation: the equations must be rearranged. It was previously noted that on the computer the number of equations is not the only important consideration. Rearrangement can be as or more expensive than solving the equations, particularly if the coefficient matrix is stored in sparse form on secondary storage.

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

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

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

REMARK 3.5

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

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

EXERCISE 3.1

[A:20] Derive (3.6) from $\bar{\mathbf{K}}^{(e)}\bar{\mathbf{u}}^{(e)} = \bar{\mathbf{f}}^{(e)}$, (3.4) and (3.5). (*Hint*: premultiply both sides of $\bar{\mathbf{K}}^{(e)}\bar{\mathbf{u}}^{(e)} = \bar{\mathbf{f}}^{(e)}$ by an appropriate matrix). Then check by hand that using that formula you get (3.7). Use Falk's scheme for the multiplications.[3]

EXERCISE 3.2

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

EXERCISE 3.3

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

EXERCISE 3.4

[A:20] Describe an alternative method that recovers the axial member forces of the example truss from consideration of joint equilibrium, without going through the computation of member deformations.

EXERCISE 3.5

[A:20] Suppose that the third support condition in (3.21) is $u_{x2} = 0$ instead of $u_{y2} = 0$. Rederive the reduced system (3.23) for this case. Verify that this system cannot be solved for the joint displacements u_{y2}, u_{x3} and u_{y3} because the reduced stiffness matrix is singular.[4] Offer a physical interpretation of this failure.

EXERCISE 3.6

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

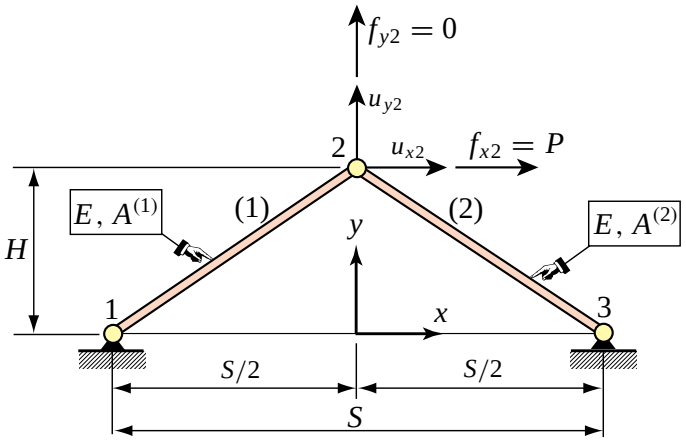


Figure E3.1. Truss structure for Exercise 3.7.

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

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

EXERCISE 3.7

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

(a) Assemble the master stiffness equations. Any method: 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 $p^{(1)} = -p^{(2)} = 15/2$.

4

The Direct Stiffness Method: Additional Topics

TABLE OF CONTENTS

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

§4.1. PRESCRIBED NONZERO DISPLACEMENTS

The support conditions considered in the example truss of Chapters 2–3 resulted in the specification of zero displacement components; for example $u_{y2} = 0$. There are cases, however, where the known value is nonzero. This happens, for example, in the study of settlement of foundations of ground structures such as buildings and bridges, and in the analysis of 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.

§4.1.1. Application of DBCs by Reduction

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

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

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

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

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

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

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

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

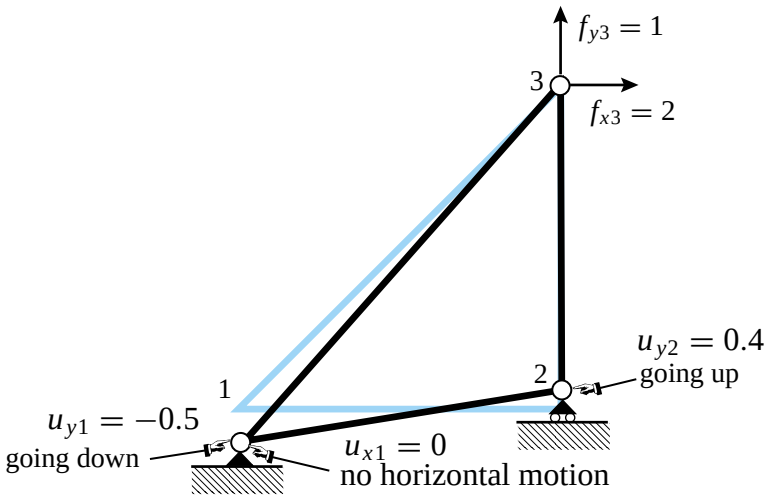


Figure 4.1. The example truss with prescribed nonzero vertical displacements at joints 1 and 2.

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

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

The matrix equation (4.5) is the *reduced system*. Note that its coefficient matrix is exactly the same as in the reduced system (3.23) for prescribed zero displacements. The right hand side, however, is different. It consists of the applied joint forces *modified by the effect of known nonzero displacements*. Solving the reduced system yields

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

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

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

Going through the postprocessing steps discussed in §3.3 with (4.7), we can find that the reaction forces and the internal member forces do not change. This is a consequence of the fact that the example truss is *statically determinate*. The force systems (internal and external) in such structures are insensitive to movements such as foundation settlements.

§4.1.2. *Application of DBCs by Modification

The computer-oriented modifification approach follows the same idea outlined in §3.5.2. As there, the 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 support conditions (4.2):

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

The solution of this system recovers (4.2) by construction (for example, the fourth equation is simply $1 \times u_{y2} = 0.4$). In the next stage, columns 1, 2 and 4 of the coefficient matrix are cleared by transferring all known terms to the right hand side, following the same procedure explained in (4.5). We thus arrive at

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

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

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

REMARK 4.1

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

§4.1.3. *Matrix Forms of DBC Application Methods

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

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

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

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

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

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

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

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

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

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

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

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

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

For the computer implemenmtation it is important to note that the partitioned form (4.10) is used only to facilitate the use of matrix notation. The equations are not explicitly rearranged and retain their original numbers. For instance, in the example truss

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

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

§4.2. THERMOMECHANICAL EFFECTS

The assumptions invoked in Chapters 2-3 for the example truss result in zero external forces under zero displacements. This is implicit in the linear-homogeneous expression of the master stiffness equation $\mathbf{f} = \mathbf{K}\mathbf{u}$. If $\mathbf{u}$ vanishes, so does $\mathbf{f}$. This behavior does not apply, however, if there are *initial force effects*.[1] If those effects are present, there can be displacements without external forces, and internal forces without displacements.

A common source of initial force effects are temperature changes. Imagine that a plane truss structure is *unloaded* (that is, not subjected to external forces) and is held at a *uniform reference temperature*. External displacements are measured from this environment, which is technically called a *reference state*. Now suppose that the temperature of some members changes with respect to the reference temperature while the applied external forces remain zero. Because the length of members changes on account of thermal expansion or contraction, the joints will displace. If the structure is statically indeterminate those displacements will induce strains and stresses and thus internal forces. These are distinguished from mechanical effects by the qualifier "thermal."

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

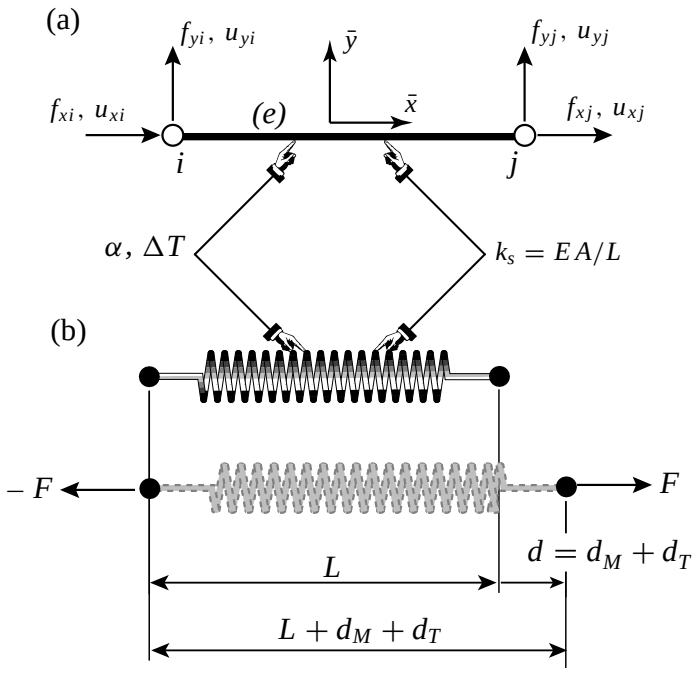


Figure 4.2. Generic truss member subjected to mechanical and thermal effects: (a) idealization as bar, (b) idealization as equivalent linear spring.

For many structures, particularly in aerospace and mechanical engineering, such effects have to be considered in the analysis and design.

There are other physical sources of initial force effects, such as moisture (hygrosteric) effects,[2] member prestress, residual stresses, or lack of fit. For linear structural models *all* such sources may be algebraically treated in the same way as thermal effects. The treatment results in an *initial force* vector that has to be added to the applied mechanical forces. This subject is outlined in §4.3 from a general perspective. However, to describe the main features of the matrix analysis procedure it is sufficient to consider the case of temperature changes.

In this Section we go over the analysis of a plane truss structure whose members undergo temperature changes from a reference state. It is assumed that the disconnection and localization steps of the DSM have been carried out. Therefore we begin with the derivation of the matrix stiffness equations of a generic truss member.

§4.2.1. Thermomechanical Behavior

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

We introduce the concept of *reference temperature* T_{ref}. This is conventionally chosen to be the temperature throughout the structure at which the displacements, strains and stresses are zero if

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

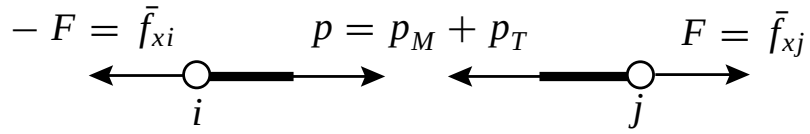


Figure 4.3. Equilibrium of truss member under thermomechanical forces.

no mechanical forces are applied. In structures such as buildings and bridges T_{ref} is often taken to be the mean temperature during the construction period. Those zero displacements, strains and stresses, together with T_{ref}, define the *thermomechanical reference state* for the structure.

The member temperature variation from that reference state is $\Delta T = T - T_{ref}$. This may be positive or negative. If the member is *disassembled* or *disconnected*, under this variation the member length is free to change from L to $L + d_T$. If the thermoelastic constitutive behavior is linear[3] then d_T is proportional to L and ΔT:

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

Here α is the coefficient of thermal expansion, which has physical units of one over temperature. This coefficient will be assumed to be uniform over the generic member. It may, however, vary from member to member. The *thermal strain* is defined as

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

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

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

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

At the other extreme, suppose that the member is completely blocked against axial elongation; that is, $d = 0$ but $\Delta T \neq 0$. Then $e = 0$ and $e_M = -e_T$. If $\alpha > 0$ and $\Delta T > 0$ the blocked member goes in *compression* because $\sigma = E e_M = -E e_T = -E\alpha\, \Delta T < 0$. This *thermal stress* is further discussed in Remark 4.2.

§4.2.2. Thermomechanical Stiffness Equations

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

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

[3] An assumption justified if the temperature changes are small enough so that α is approximately constant through the range of interest, and no material phase change effects occur.

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

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

Here $p_M = A\sigma$ denotes the mechanical axial force, and $p_T = EA\,\alpha\,\Delta T$, which has the dimension of a force, is called (not surprisingly) the *internal thermal force*. The sum $p = p_M + p_T$ is called the *effective internal force*. The last relation in (4.20), $F = p = p_M + p_T$ follows from free-body member equilibrium; see Figure 4.3. Passing to matrix form:

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

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

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

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

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

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

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

This is called the vector of *thermal joint forces* in local coordinates. It is an instance of an *initial force* vector at the element level.

REMARK 4.2

A useful physical interpretation of (4.23) is as follows. Suppose that the member is completely blocked against joint motions so that $\bar{\mathbf{u}} = \mathbf{0}$. Then $\bar{\mathbf{f}}_M + \bar{\mathbf{f}}_T = \mathbf{0}$ or $\bar{\mathbf{f}}_M = -\bar{\mathbf{f}}_T$. It follows that $\mathbf{f}_T$ contains the negated joint forces (internal forces) that develop in a heated or cooled bar if joint motions are precluded. Because for most materials $\alpha > 0$, rising the temperature of a blocked bar — that is, $\Delta T > 0$ — produces an internal compressive thermal force $p_T = A\sigma_T = -EA\alpha T$, in accordance with the expected physics. The quantity $\sigma_T = -E\alpha\,\Delta T$ is the *thermal stress*. This stress can cause buckling or cracking in severely heated structural members that are not allowed to expand or contract. This motivates the use of expansion joints in pavements, buildings and rails, and roller supports in long bridges.

§4.2.3. Globalization

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

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

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

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

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

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

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

§4.2.4. Merge

The merge process is based on the same assembly rules stated in §3.1.3 with only one difference: thermal forces are added to the right hand side. The member by member merge is carried out much as described as in §3.1.4, the main difference being that the thermal force vectors $\mathbf{f}_T^{(e)}$ are also merged into a master thermal force vector.[4] Upon completion of the assembly process we arrive at the free-free master stiffness equations

$$\boxed{\mathbf{Ku} = \mathbf{f}_M + \mathbf{f}_T = \mathbf{f}.} \tag{4.28}$$

§4.2.5. Solution

The master system (4.28) has formally the same configuration as the master stiffness equations (2.3). The only difference is that the *effective* joint force vector **f** contains a superposition of mechanical and thermal forces.

Displacement boundary conditions can be applied by reduction or modification of these equations, simply by using effective joint forces in the descriptions of §3.2.1, §3.4.1 and §4.1. Processing the reduced or modified system by a linear equation solver yields the displacement solution **u**.

§4.2.6. Postprocessing

The postprocessing steps described in §3.4 require some modifications because the derived quantities of interest to the structural engineer are *mechanical* reaction forces and internal forces. Effective forces by themselves are of little use in design.

Mechanical joint forces including reactions are recovered from

$$\mathbf{f}_M = \mathbf{Ku} - \mathbf{f}_T \tag{4.29}$$

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

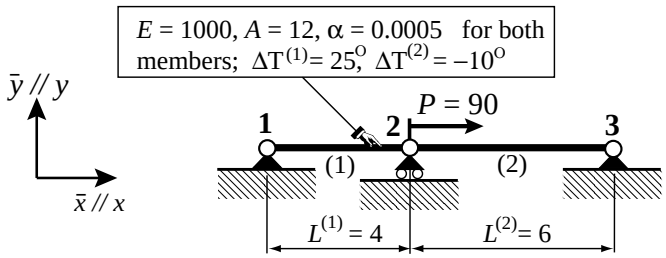


Figure 4.4. Structure for worked-out Example 1.

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

$$p_M^{(e)} = p^{(e)} - E^{(e)}A^{(e)}\alpha^{(e)}\,\Delta T^{(e)}. \tag{4.30}$$

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

§4.2.7. Worked-Out Example 1

The first problem is defined in Figure 4.4. Two truss members are connected in series as shown and fixed at the ends. Properties $E = 1000$, $A = 5$ and $\alpha = 0.0004$ are common to both members. The member lengths are 4 and 6. A mechanical load $P = 90$ acts on the roller node. The temperature of member (1) increases by $\Delta T^{(1)} = 25^\circ$ while that of member (2) drops by $\Delta T^{(2)} = -10^\circ$. Find the stress in the members.

To reduce clutter note that all y motions are suppressed so only the x freedoms are kept: $u_{x1} = u_1$, $u_{x2} = u_2$ and $u_{x3} = u_3$. The corresponding node forces are denoted $f_{x1} = f_1$, $f_{x2} = f_2$ and $f_{x3} = f_3$. The thermal force vectors, stripped to their $\bar{x} \equiv x$ components, are

$$\bar{\mathbf{f}}_T^{(1)} = \begin{bmatrix} \bar{f}_{T1}^{(1)} \\ \bar{f}_{T2}^{(1)} \end{bmatrix} = E^{(1)}A^{(1)}\alpha^{(1)}\Delta T^{(1)}\begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} -150 \\ 150 \end{bmatrix},\ \bar{\mathbf{f}}_T^{(2)} = \begin{bmatrix} \bar{f}_{T2}^{(2)} \\ \bar{f}_{T3}^{(2)} \end{bmatrix} = E^{(2)}A^{(2)}\alpha^{(2)}\Delta T^{(2)}\begin{bmatrix} -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 60 \\ -60 \end{bmatrix}. \tag{4.31}$$

The element stiffness equations are:

$$3000\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} \bar{u}_1^{(1)} \\ \bar{u}_2^{(1)} \end{bmatrix} = \begin{bmatrix} \bar{f}_{M1}^{(1)} \\ \bar{f}_{M2}^{(1)} \end{bmatrix} + \begin{bmatrix} -150 \\ 150 \end{bmatrix},\quad 2000\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} \bar{u}_2^{(2)} \\ \bar{u}_3^{(2)} \end{bmatrix} = \begin{bmatrix} \bar{f}_{M2}^{(2)} \\ \bar{f}_{M3}^{(2)} \end{bmatrix} + \begin{bmatrix} 60 \\ -60 \end{bmatrix}, \tag{4.32}$$

No globalization is needed because the equations are already in the global system, and thus we get rid of the local symbols: $\bar{f} \to f$, $\bar{u} \to u$. Assembling:

$$1000\begin{bmatrix} 3 & -3 & 0 \\ -3 & 5 & -2 \\ 0 & -2 & 2 \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} f_{M1} \\ f_{M2} \\ f_{M3} \end{bmatrix} + \begin{bmatrix} -150 \\ 150+60 \\ -60 \end{bmatrix} = \begin{bmatrix} f_{M1} \\ f_{M2} \\ f_{M3} \end{bmatrix} + \begin{bmatrix} -150 \\ 210 \\ -60 \end{bmatrix}. \tag{4.33}$$

The displacement boundary conditions are $u_1 = u_3 = 0$. The mechanical force boundary condition is $f_{M2} = 90$. On removing the first and third equations, the reduced system is $5000\,u_2 = f_{M2} + 210 = 90 + 210 = 300$, which yields $u_2 = 300/5000 = +0.06$. The internal forces in the members are recovered from

$$\begin{aligned} p_M^{(1)} &= \frac{E^{(1)}A^{(1)}}{L^{(1)}}(u_2 - u_1) - E^{(1)}A^{(1)}\alpha^{(1)}\Delta T^{(1)} = 3000\times 0.06 - 12000\times 0.0004\times 25 = 60, \\ p_M^{(2)} &= \frac{E^{(2)}A^{(2)}}{L^{(2)}}(u_3 - u_2) - E^{(2)}A^{(2)}\alpha^{(2)}\Delta T^{(2)} = 2000\times(-0.06) - 12000\times 0.0004\times(-10) = -72, \end{aligned} \tag{4.34}$$

whence $\sigma^{(1)} = 60/12 = 5$ and $\sigma^{(2)} = -72/12 = -6$. Member (1) is in tension and member (2) in compression.

§4.2.8. Worked-Out Example 2

The second example concerns the example truss of Chapters 2-3. The truss is mecanically *unloaded*, that is, $f_{Mx2} = f_{Mx3} = f_{My3} = 0$. However the temperature of members (1) (2) and (3) changes by ΔT, $-\Delta T$ and $3\Delta T$, respectively, with respect to T_{ref}. The thermal expansion coefficient of all three members is assumed to be α. We will perform the analysis keeping α and ΔT as variables.

The thermal forces for each member in global coordinates are obtained by using (4.25) and the third of (4.27):

$$\begin{aligned}
\mathbf{f}_T^{(1)} &= E^{(1)}A^{(1)}\alpha^{(1)}\,\Delta T^{(1)}\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{bmatrix}\begin{bmatrix}-1\\0\\1\\0\end{bmatrix} = 100\alpha\,\Delta T\begin{bmatrix}-1\\0\\1\\0\end{bmatrix},\\
\mathbf{f}_T^{(2)} &= E^{(2)}A^{(2)}\alpha^{(2)}\,\Delta T^{(2)}\begin{bmatrix}0&-1&0&0\\1&0&0&0\\0&0&0&-1\\0&0&1&0\end{bmatrix}\begin{bmatrix}-1\\0\\1\\0\end{bmatrix} = 50\alpha\,\Delta T\begin{bmatrix}0\\-1\\0\\1\end{bmatrix},\\
\mathbf{f}_T^{(3)} &= E^{(3)}A^{(3)}\alpha^{(3)}\,\Delta T^{(3)}\frac{1}{\sqrt{2}}\begin{bmatrix}1&-1&0&0\\1&1&0&0\\0&0&1&-1\\0&0&1&1\end{bmatrix}\begin{bmatrix}-1\\0\\1\\0\end{bmatrix} = 200\alpha\,\Delta T\begin{bmatrix}-1\\-1\\1\\1\end{bmatrix}.
\end{aligned} \tag{4.35}$$

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

$$\mathbf{f}_T = \alpha\,\Delta T\begin{bmatrix}-100+0-200\\0+0-200\\100+0+0\\0-50+0\\0+0+200\\0+50+200\end{bmatrix} = \alpha\,\Delta T\begin{bmatrix}-300\\-200\\100\\-50\\200\\250\end{bmatrix} \tag{4.36}$$

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

$$\begin{bmatrix}20&10&-10&0&-10&-10\\10&10&0&0&-10&-10\\-10&0&10&0&0&0\\0&0&0&5&0&-5\\-10&-10&0&0&10&10\\-10&-10&0&-5&10&15\end{bmatrix}\begin{bmatrix}u_{x1}=0\\u_{y1}=0\\u_{x2}\\u_{y2}=0\\u_{x3}\\u_{y3}\end{bmatrix} = \begin{bmatrix}f_{Mx1}\\f_{My1}\\f_{Mx2}=0\\f_{My2}\\f_{Mx3}=0\\f_{My3}=0\end{bmatrix} + \alpha\,\Delta T\begin{bmatrix}-300\\-200\\100\\-50\\200\\250\end{bmatrix} \tag{4.37}$$

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

$$\begin{bmatrix}10&0&0\\0&10&10\\0&10&15\end{bmatrix}\begin{bmatrix}u_{x2}\\u_{x3}\\u_{y3}\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix} + \alpha\,\Delta T\begin{bmatrix}100\\200\\250\end{bmatrix} = \alpha\,\Delta T\begin{bmatrix}100\\200\\250\end{bmatrix}. \tag{4.38}$$

Solving (4.38) gives $u_{x2} = u_{x3} = u_{y3} = 10\alpha\,\Delta T$. Completing **u** with the prescribed zero displacements and premultiplying by **K** gives the complete effective force vector:

$$\mathbf{f} = \mathbf{K}\mathbf{u} = \begin{bmatrix}20&10&-10&0&-10&-10\\10&10&0&0&-10&-10\\-10&0&10&0&0&0\\0&0&0&5&0&-5\\-10&-10&0&0&10&10\\-10&-10&0&-5&10&15\end{bmatrix}\begin{bmatrix}0\\0\\10\\0\\10\\10\end{bmatrix}\alpha\,\Delta T = \alpha\,\Delta T\begin{bmatrix}-300\\-200\\100\\-50\\200\\250\end{bmatrix}. \tag{4.39}$$

But the effective force vector is exactly $\mathbf{f}_T$. Consequently

$$\mathbf{f}_M = \mathbf{K}\mathbf{u} - \mathbf{f}_T = \mathbf{0}. \tag{4.40}$$

All mechanical joint forces, including reactions, vanish, and so do the internal mechanical forces. This is a consequence of the example frame being statically determinate.[5] Such structures *do not develop thermal stresses under any combination of temperature changes*.

§4.3. INITIAL FORCE EFFECTS

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

$$\boxed{\bar{\mathbf{K}}^{(e)}\bar{\mathbf{u}}^{(e)} = \bar{\mathbf{f}}_M^{(e)} + \bar{\mathbf{f}}_I^{(e)} = \bar{\mathbf{f}}^{(e)},} \tag{4.41}$$

and at the global (assembled structure) level:

$$\boxed{\mathbf{K}\mathbf{u} = \mathbf{f}_M + \mathbf{f}_I = \mathbf{f}.} \tag{4.42}$$

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

A physical interpretation of (4.42) can be obtained by considering that the structure is blocked against all motions: $\mathbf{u} = \mathbf{0}$. Then $\mathbf{f}_M = -\mathbf{f}_I$, and the undeformed structure experiences mechanical forces. These translate into internal forces and stresses. Engineers also call these *prestresses*.

Local effects that lead to initial forces at the member level are: temperature changes (studied in §4.2, in which $\mathbf{f}_I \equiv \mathbf{f}_T$), moisture diffusion, residual stresses, lack of fit in fabrication, and in-member prestressing. Global effects include prescribed nonzero joint displacements (studied in §4.1) and multimember prestressing (for example, by cable pretensioning of concrete structures).

As can be seen there is a wide variety of physical effects, whether natural or artificial, that lead to nonzero initial forces. The good news is that once the member equations (4.41) are formulated, the remaining DSM steps (globalization, merge and solution) are identical. This nice property extends to the general Finite Element Method.

§4.4. PSEUDOTHERMAL INPUTS

Some commercial FEM programs do not have a way to handle directly effects such as moisture, lack of fit, or prestress. But all of them can handle temperature variation inputs. Since in linear analysis all such effects can be treated as initial forces, it is possible (at least for bar elements) to model them as fictitious thermomechanical effects, by inputting phony temperature changes. The following example indicate that this is done for a bar element.

Suppose that a prestress force F_P is present in a bar. The total elongation is $d = d_M + d_P$ where $d_P = F_P L/(E A)$ is due to prestress. Equate to a thermal elongation: $d_T = \alpha \Delta T_P\, L$ and solve for

[5] For the definition of static determinacy, see any textbook on Mechanics of Materials.

$\Delta T_P = F_P/(EA\alpha)$. This is input to the program as a fictitious temperature change. If in addition there is a real temperature change ΔT one would of course specify $\Delta T + \Delta T_P$.

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

Homework Exercises for Chapter 4

The Direct Stiffness Method: Additional Topics

EXERCISE 4.1

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

EXERCISE 4.2

[N:20] Use the same data of Exercise 3.7 except that $P = 0$ and hence there are no applied mechanical forces. Both members have the same dilatation coefficient $\alpha = 10^{-6}$ 1/°F. Find the crown displacements u_{x2} and u_{y2} and the member stresses $\sigma^{(1)}$ and $\sigma^{(2)}$ if the temperature of member (1) rises by $\Delta T = 120\,^{\circ}$F above T_{ref}, whereas member (2) stays at T_{ref}.

Shortcut: the element stiffnesses and master stiffness matrix are the same as in Exercise 3.7, so if that Exercise has been previously assigned no stiffness recomputations are necessary.

EXERCISE 4.3

[A:15] Consider the generic truss member of Figure 2.6. The disconnected member was supposed to have length L, but because of lack of quality control it was fabricated with length $L + \delta$, where δ is called the "lack of fit." Determine the initial force vector $\bar{\mathbf{f}}_I$ to be used in (4.41). *Hint*: find the mechanical forces that would compensate for δ and restore the desired length.

EXERCISE 4.4

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

EXERCISE 4.5

[A:40]. Prove that statically determinate truss structures are free of thermal stresses.

5

Analysis of Example Truss by a CAS

TABLE OF CONTENTS

§5.1. COMPUTER ALGEBRA SYSTEMS

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

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

§5.1.1. Why Mathematica?

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

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

One common disadvantage of CAS, and *Mathematica* is not exception, is computational inefficiency in numerical calculations compared with a low-level implementation in, for instance, C or Fortran. The relative penalty can reach several orders of magnitude. For instructional use, however, the penalty is acceptable when compared to *human efficiency*. This means the ability to get FEM programs up and running in very short time, with capabilities for symbolic manipulation and graphics as a bonus.

§5.1.2. Programming Style and Prerequisites

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

Practice with the program until you reach the level of writing functions, modules and scripts with relative ease. With the Notebook interface and a good primer it takes only a few hours.

---

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

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

[3] This is also available on CDROM from MathWare, Ltd, P. O. Box 3025, Urbana, IL 61208, e-mail: `info@mathware.com`. The CDROM is a hyperlinked version of the book that can be installed on the same directory as *Mathematica*.

When approaching that level you may notice that functions in *Mathematica* display many aspects similar to C.[4] You can exploit this similarity if you are proficient in that language. But *Mathematica* functions do have some unique aspects, such as matching arguments by pattern, and the fact that internal variables are global unless otherwise made local.

Although function arguments can be modified, in practice this should be avoided because it may be difficult to trace side effects. The programming style enforced here outlaws output arguments and a function can only return its name. But since the name can be a list of arbitrary objects the restriction is not serious.[5]

Our objective is to develop a symbolic program written in *Mathematica* that solves the example plane truss as well as some symbolic versions thereof. The program will rely heavily on the development and use of *functions* implemented using the `Module` construct of *Mathematica*. Thus the style will be one of procedural programming.[6] The program will not be particularly modular (in the computer science sense) because *Mathematica* is not suitable for that programming style.[7]

The code presented in Sections 5.2–5.7 uses a few language constructs that may be deemed as advanced, and these are briefly noted in the text so that appropriate reference to the *Mathematica* reference manual can be made.

§5.1.3. Class Demo Scripts

The cell scripts shown in Figures 5.1 and 5.2 will be used to illustrate the organization of a Notebook file and the "look and feel" of some basic *Mathematica* commands. These scripts will be demonstrated in class from a laptop.

§5.2. THE ELEMENT STIFFNESS MODULE

As our first FEM code segment, the top box of Figure 5.3 shows a module that evaluates and returns the 4×4 stiffness matrix of a plane truss member (two-node bar) in global coordinates. The text in that box of that figure is supposed to be placed on a Notebook cell. Executing the cell, by clicking on it and hitting an appropriate key (`<Enter>` on a Mac), gives the output shown in the bottom box. The contents of the figure is described in further detail below.

§5.2.1. Module Description

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

---

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

[5] Such restrictions on arguments and function returns are closer in spirit to `C` than `Fortran` although you can of course modify C-function arguments using pointers.

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

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

Integration example

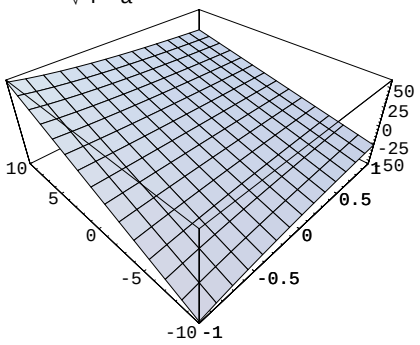
```
f[x_,α_,β_]:=(1+β*x^2)/(1+α*x+x^2);
F=Integrate[f[x,-1,2],{x,0,5}];
F=Simplify[F];
Print[F]; Print[N[F]];
F=NIntegrate[f[x,-1,2],{x,0,5}];
Print["F=",F//InputForm];
```

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

Figure 5.1. Example cell for class demo.

```
Fa=Integrate[f[z,a,b],{z,0,5}];
Fa=Simplify[Fa]; Print[Fa];
Plot3D[Fa,{a,-1,1},{b,-10,10},ViewPoint->{-1,-1,1}];
Print["Fa=",Fa//InputForm]
```

$$\frac{1}{2\sqrt{4-a^2}}\left(10\sqrt{4-a^2}\,b-a\sqrt{4-a^2}\,b\,\text{Log}[26+5a]-i\,(2+(-2+a^2)\,b)\,\text{Log}\Big[1-\frac{i\,a}{\sqrt{4-a^2}}\Big]+2\,i\,\text{Log}\Big[1+\frac{i\,a}{\sqrt{4-a^2}}\Big]-2\,i\,b\,\text{Log}\Big[1+\frac{i\,a}{\sqrt{4-a^2}}\Big]+i\,a^2\,b\,\text{Log}\Big[1+\frac{i\,a}{\sqrt{4-a^2}}\Big]+2\,i\,\text{Log}\Big[\frac{-10\,i-i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]-2\,i\,b\,\text{Log}\Big[\frac{-10\,i-i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]+i\,a^2\,b\,\text{Log}\Big[\frac{-10\,i-i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]-2\,i\,\text{Log}\Big[\frac{10\,i+i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]+2\,i\,b\,\text{Log}\Big[\frac{10\,i+i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]-i\,a^2\,b\,\text{Log}\Big[\frac{10\,i+i\,a+\sqrt{4-a^2}}{\sqrt{4-a^2}}\Big]\right)$$




```
Fa=(10*Sqrt[4 - a^2]*b - a*Sqrt[4 - a^2]*b*Log[26 + 5*a] -
    I*(2 + (-2 + a^2)*b)*Log[1 - (I*a)/Sqrt[4 - a^2]] + (2*I)*Log[1 + (I*a)/Sqrt[4 - a^2]] -
    (2*I)*b*Log[1 + (I*a)/Sqrt[4 - a^2]] + I*a^2*b*Log[1 + (I*a)/Sqrt[4 - a^2]] +
    (2*I)*Log[(-10*I - I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]] -
    (2*I)*b*Log[(-10*I - I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]] +
    I*a^2*b*Log[(-10*I - I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]] -
    (2*I)*Log[(10*I + I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]] +
    (2*I)*b*Log[(10*I + I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]] -
    I*a^2*b*Log[(10*I + I*a + Sqrt[4 - a^2])/Sqrt[4 - a^2]])/(2*Sqrt[4 - a^2])
```

Figure 5.1. Another example cell for class demo.

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

```
Numerical elem stiff matrix:

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

Symbolic elem stiff matrix:

 A Em          A Em          A Em          A Em
 ----------    ----------    ----------    ----------
 2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L

 A Em          A Em          A Em          A Em
 ----------    ----------    ----------    ----------
 2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L

-A Em          -A Em         A Em          A Em
 ----------    ----------    ----------    ----------
 2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L

-A Em          -A Em         A Em          A Em
 ----------    ----------    ----------    ----------
 2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L   2 Sqrt[2] L
```

Figure 5.3. Module `ElemStiff2DTwoNodeBar` to form the element stiffness of a 2D bar element in global coordinates, test program and its output.

The module takes two arguments:

`{{x1,y1},{y1,y2}}` A two-level list[8] containing the $\{x, y\}$ coordinates of the bar end nodes labelled as 1 and 2.[9]

`{Em,A}` A level-one list containing the bar elastic modulus, E and the member cross section area, A. See §5.2.3 as to why name `E` cannot be used.

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

The module name returns the 4×4 member stiffness matrix internally called `Ke`. The logic that leads to the formation of that matrix is straightforward and need not be explained in detail. Note, however, the elegant direct declaration of the matrix `Ke` as a level-two list, which eliminates the fiddling around with array indices typical of standard programming languages. The format in fact closely matches the mathematical expression (3.4).

---

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

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

§5.2.2. Programming Remarks

The function in Figure 5.3 uses several intermediate variables with short names: `dx`, `dy`, `s`, `c` and `L`. It is strongly advisable to make these symbols *local* to avoid potential names clashes somewhere else.[10] In the `Module[ ...]` construct this is done by listing those names in a list immediately after the opening bracket. Local variables may be *initialized* when they are constants or simple functions of the argument items; for example on entry to the module `dx=x2-x1` initializes variable `dx` to be the difference of x node coordinates, namely $\Delta x = x_2 - x_1$.

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

§5.2.3. Case Sensitivity

Mathematica, like most recent computer languages, is case sensitive so that for instance `E` is not the same as `e`. This is fine. But the language designer decided that names of system-defined objects such as built-in functions and constants must begin with a capital letter. Consequently the liberal use of names beginning with a capital letter may run into clashes. If, for example, you cannot use `E` because of its built-in meaning as the base of natural logariths.[11]

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

§5.2.4. Testing the Member Stiffness Module

Following the definition of `ElemStiff2DTwoNodeBar` in Figure 5.3 there are several statements that constitute the *module test program* that call the module and print the returned results. Two cases are tested. First, the stiffness of member (3) of the example truss, using all-numerical values. Next, some of the input arguments for the same member are given symbolic names so they stand for variables; for example the elastic module is given as `Em` instead of 100 as in the foregoing test. The print output of the test is shown in the lower portion of Figure 5.3.

The first test returns the member stiffness matrix (3.10) as may be expected. The second test returns a symbolic form in which three symbols appear: the coordinates of end node 2, which is taken to be located at `{L,L}` instead of $\{10, 10\}$, `A`, which is the cross-section area and `Em`, which is the elastic modulus. Note that the returning matrix `Ke` is subject to a `Simplify` step before printing it, which is the subject of an Exercise. The ability to carry along variables is of course a fundamental capability of any CAS and the main reason for which such programs are used.

---

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

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

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

```
Initialized master stiffness matrix:

⎛0  0  0  0  0  0⎞
⎜0  0  0  0  0  0⎟
⎜0  0  0  0  0  0⎟
⎜0  0  0  0  0  0⎟
⎜0  0  0  0  0  0⎟
⎝0  0  0  0  0  0⎠

Member stiffness matrix:

⎛ 10   10  -10  -10⎞
⎜ 10   10  -10  -10⎟
⎜-10  -10   10   10⎟
⎝-10  -10   10   10⎠

Master stiffness after member merge:

⎛ 10   10  0  0  -10  -10⎞
⎜ 10   10  0  0  -10  -10⎟
⎜  0    0  0  0    0    0⎟
⎜  0    0  0  0    0    0⎟
⎜-10  -10  0  0   10   10⎟
⎝-10  -10  0  0   10   10⎠
```

Figure 5.4. Module `MergeElemIntoMasterStiff` to merge a 4×4 bar element stiffness into the master stiffness matrix, test program and its output.

§5.3. MERGING A MEMBER INTO THE MASTER STIFFNESS

The next fragment of *Mathematica* code, listed in Figure 5.4, is used in the assembly step of the DSM. Module `MergeElemIntoMasterStiff` receives the 4×4 element stiffness matrix formed by `FormElemStiff2DNodeBar` and "merges" it into the master stiffness matrix. The module takes three arguments:

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

`eftab` The column of the Element Freedom Table, defined in §3.4.1, appropriate to the member being merged; cf. (3.29). Recall that the EFT lists the global equation numbers for the four member degrees of freedom. This is a level-one list consisting of 4 integers.

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

`MergeElemIntoMasterStiff` returns, as module name, the updated master stiffness matrix internally called `K` with the member stiffness merged in. Thus we encounter here a novelty: an input-output

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

```
Master stiffness of example truss:

⎛  20   10  -10   0  -10  -10 ⎞
⎜  10   10    0   0  -10  -10 ⎟
⎜ -10    0   10   0    0    0 ⎟
⎜   0    0    0   5    0   -5 ⎟
⎜ -10  -10    0   0   10   10 ⎟
⎝ -10  -10    0  -5   10   15 ⎠
```

Figure 5.5. Module `MasterStiffOfExampleTruss` that forms the 6×6 master stiffness matrix of the example truss, test program and its output.

argument. Because a formal argument cannot be modified, the situation is handled by copying the incoming `Kin` into `K` on entry. It is the copy which is updated and returned via the `Return` statement. The implementation has a strong C flavor with two nested `For` loops. Because the iterators are very simple, nested `Do` loops could have been used as well.

The statements after the module provide a simple test. Before the first call to this function, the master stiffness matrix must be initialized to a zero 6×6 array. This is done in the first test statement using the `Table` function. The test member stiffness matrix is that of member (3) of the example truss, and is obtained by calling `ElemStiff2DTwoNodeBar`. The EFT is `{1,2,5,6}` since element freedoms 1,2,3,4 map into global freedoms 1,2,5,6. Running the test statements yields the listing given in Figure 5.4. The result is as expected.

§5.4. ASSEMBLING THE MASTER STIFFNESS

The module `MasterStiffOfExampleTruss`, listed in the top box of Figure 5.5, makes use of the foregoing two modules: `ElemStiff2DTwoNodeBar` and `MergeElemIntoMasterStiff`, to form the master stiffness matrix of the example truss. The initialization of the stiffness matrix array in `K` to zero is done by the `Table` function of `Mathematica`, which is handy for initializing lists. The remaining statements are self explanatory. The module is similar in style to argumentless Fortran or C functions. It takes no arguments. All the example truss data is "wired in."

The output from the test program in is shown in the lower box of Figure 5.5. The output stiffness matches that in Equation (3.20), as can be expected if all fragments used so far work correctly.

§5.5. MODIFYING THE MASTER SYSTEM

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

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

```
Assembled master stiffness:
⎛ Kij[1, 1]  Kij[1, 2]  Kij[1, 3]  Kij[1, 4]  Kij[1, 5]  Kij[1, 6] ⎞
⎜ Kij[2, 1]  Kij[2, 2]  Kij[2, 3]  Kij[2, 4]  Kij[2, 5]  Kij[2, 6] ⎟
⎜ Kij[3, 1]  Kij[3, 2]  Kij[3, 3]  Kij[3, 4]  Kij[3, 5]  Kij[3, 6] ⎟
⎜ Kij[4, 1]  Kij[4, 2]  Kij[4, 3]  Kij[4, 4]  Kij[4, 5]  Kij[4, 6] ⎟
⎜ Kij[5, 1]  Kij[5, 2]  Kij[5, 3]  Kij[5, 4]  Kij[5, 5]  Kij[5, 6] ⎟
⎝ Kij[6, 1]  Kij[6, 2]  Kij[6, 3]  Kij[6, 4]  Kij[6, 5]  Kij[6, 6] ⎠

Master stiffness modified for displacement B.C.:
⎛ 1  0     0     0     0         0        ⎞
⎜ 0  1     0     0     0         0        ⎟
⎜ 0  0  Kij[3, 3]  0  Kij[3, 5]  Kij[3, 6] ⎟
⎜ 0  0     0     1     0         0        ⎟
⎜ 0  0  Kij[5, 3]  0  Kij[5, 5]  Kij[5, 6] ⎟
⎝ 0  0  Kij[6, 3]  0  Kij[6, 5]  Kij[6, 6] ⎠

Force vector:
{fi[1], fi[2], fi[3], fi[4], fi[5], fi[6]}

Force vector modified for displacement B.C.:
{0, 0, fi[3], 0, fi[5], fi[6]}
```

Figure 5.6. Modules `ModifiedMasterStiff` and `ModifiedMasterForce` that modify the master stiffness matrix and force vector of a truss to impose displacement BCs.

Module `ModifiedMasterStiffForDBC` carries out this process for the master stiffness matrix **K**, whereas `ModifiedMasterForcesForDBC` does this for the nodal force vector **f**. These two modules are listed in the top box of Figure 5.6, along with test statements. The logic of both functions, but especially that of `ModifiedMasterForcesForBC`, is considerably simplified by assuming that *all prescribed displacements are zero*, that is, the BCs are homogeneous. The more general case of nonzero prescribed values is treated in Chapter 21.

Function `ModifiedMasterStiffnessForDBC` has two arguments:

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

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

The function clears appropriate rows and columns of **K**, places ones on the diagonal, and returns the modified **K** as function value. The only slightly fancy thing in this module is the use of the *Mathematica* function `Length` to extract the number of prescribed displacement components: `Length[pdof]` here

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

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

Figure 5.7. Module `IntForce2DTwoNodeBar` for computing the internal force in a bar element.

will return the value 3, which is the length of the list `pdof`. Similarly `nk=Length[K]` assigns 6 to `nk`, which is the order of matrix **K**. Although for the example truss these values are known *a priori*, the use of `Length` serves to illustrate a technique that is heavily used in more general code.

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

The test statements are purposedly chosen to illustrate another feature of *Mathematica*: the use of the `Array` function to generate subscripted symbolic arrays of one and two dimensions. The test output is shown in the bottom box of Figure 5.6, which should be self explanatory. The force vector and its modified form are printed as row vectors to save space.

§5.6. RECOVERING INTERNAL FORCES

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

Function `IntForce2DTwoNodeBar` listed in the top box of Figure 5.7 computes the internal force in an individual bar element. It is somewhat similar in argument sequence and logic to `ElemStiff2DTwoNodeBar` (Figure 5.3). The first two arguments are identical. Argument `eftab` provides the Element Foreedom Table array for the element. The last argument, `u`, is the vector of computed node displacements.

The logic of `IntForce2DTwoNodeBar` is straightforward and follows the method outlined in §3.2.1. Member joint displacements $\bar{\mathbf{u}}^{(e)}$ in local coordinates $\{\bar{x}, \bar{y}\}$ are recovered in array `ubar`, then the longitudinal strain $e = (\bar{u}_{xj} - \bar{u}_{xi})/L$ and the internal (axial) force $p = EAe$ is returned as function value. As coded the function contains redundant operations because entries 2 and 4 of `ubar` (that

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

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

Figure 5.8. Module `IntForceOfExampleTruss` that computes internal forces in the 3 members of the example truss.

is, components $\bar{u}_{yi}$ and u_{yj}) are not actually needed to get p, but were kept to illustrate the general backtransformation of global to local displacements.

Running this function with the test statements shown after the module produces the output shown in the bottom box of Figure 5.7. The first test is for member (3) of the example truss using the actual nodal displacements (3.24). It also illustrates the use of the *Mathematica* built in function `N` to produce output in floating-point form. The second test does a symbolic calculation in which several argument values are fed in variable form.

The top box of Figure 5.8 lists a higher-level function, `IntForcesOfExampleTruss`, which has a single argument: `u`, which is the complete vector of joint displacements **u**. This function calls `IntForce2DTwoNodeBar` three times, once for each member of the example truss, and returns the three member internal forces thus computed as a 3-component list.

The test statements listed after `IntForcesOfExampleTruss` feed the actual node displacements (3.24) to `IntForcesOfExampleTruss`. Running the functions with the test statements produces the output shown in the bottom box of Figure 5.8. The internal forces are $p^{(1)} = 0$, $p^{(2)} = -1$ and $p^{(3)} = 2\sqrt{2} = 2.82843$.

§5.7. PUTTING THE PIECES TOGETHER

After all this development and testing effort documented in Figures 5.3 through 5.8 we are ready to make use of all these bits and pieces of code to analyze the example plane truss. This is actually done with the logic shown in Figure 5.9. This *driver program* uses the previously described modules

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

These functions must have been defined ("compiled") at the time the driver programs described below are run. A simple way to making sure that all of them are defined is to put all these functions in the

same Notebook file and to mark them as *initialization cells*. These cells may be executed by picking up Kernel → Initialize → Execute Initialization. (An even simpler procedure would to group them all in one cell, but that would make placing separate test statements difficult.)

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

```
Computed nodal displacements:
{0, 0, 0, 0, 2/5, -1/5}
External node forces including reactions:
{-2, -2, 0, 1, 2, 1}
Internal member forces:
{0, -1, 2 Sqrt[2]}
```

Figure 5.9. Driver program for numerical analysis of example truss and its output.

The program listed in the top box of Figure 5.9 first assembles the master stiffness matrix through `MasterStiffOfExampleTruss`. Next, it applies the displacement boundary conditions through `ModifiedMasterStiffForDBC` and `ModifiedMasterForcesForDBC`. Note that the modified stiffness matrix is placed into `Kmod` rather than `K` to save the original form of the master stiffness for the reaction force recovery later. The complete displacement vector is obtained by the matrix calculation

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

which takes advantage of two built-in *Mathematica* functions. `Inverse` returns the inverse of its matrix argument[12] The dot operator signifies matrix multiply (here, matrix-vector multiply.) The enclosing `Simplify` function is placed to simplify the expression of vector `u` in case of symbolic calculations; it is actually redundant if all computations are numerical as in Figure 5.9.

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

Running the program of the top box of Figure 5.9 produces the output shown in the bottom box of that figure. The results confirm the hand calculations of Chapter 3.

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

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

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

```
Computed nodal displacements:
{0, 0, 0, 0, 1/10 (3 fx3 - 2 fy3), 1/5 (-fx3 + fy3)}
External node forces including reactions:
{-fx3, -fx3, 0, fx3 - fy3, fx3, fy3}
Internal member forces:
{0, -fx3 + fy3, Sqrt[2] fx3}
```

Figure 5.10. Driver program for symbolic analysis of example truss and its output.

problems, however, *Mathematica* would certainly beat hand-plus-calculator computations, the cross-over typically appearing for 10-20 equations. For up to about 500 equations and using floating-point arithmetic, *Mathematica* gives answers within minutes on a fast PC or Mac with sufficient memory but eventually runs out of steam at about 1000 equations. For a range of 1000 to about 10000 equations, *Matlab* would be the best compromise between human and computer flow time. Beyond 10000 equations a program in a low-level language, such as C or Fortran, would be most efficient in terms of computer time.

One distinct advantage of computer algebra systems appear when you need to *parametrize* a small problem by leaving one or more problem quantities as variables. For example suppose that the applied forces on node 3 are to be left as f_{x3} and f_{y3}. You replace the last two components of array p as shown in the top box of Figure 5.10, execute the cell and shortly get the symbolic answer shown in the bottom box of Figure 5.10. This is the answer to an infinite number of numerical problems. Although one may try to undertake such studies by hand, the likelyhood of errors grows rapidly with the complexity of the system. Symbolic manipulation systems can amplify human abilities in this regard, as long as the algebra "does not explode" because of combinatorial complexity. Examples of such nontrivial calculations will appear throughout the following Chapters.

REMARK 5.1

The "combinatorial explosion" danger of symbolic computations should be always kept in mind. For example, the numerical inversion of a $N \times N$ matrix is a $O(N^3)$ process, whereas symbolic inversion goes as $O(N!)$. For $N = 48$ the floating-point numerical inverse will be typically done in a fraction of a second. But the symbolic adjoint will have $48! = 124139155925360726708622890473733750385214863546777600000000000$ terms, or $O(10^{61})$. There may be enough electrons in this Universe to store that, but barely ...

Homework Exercises for Chapter 5

Analysis of Example Truss by a CAS

Before doing any of these Exercises, download the *Mathematica* Notebook file `ExampleTruss.nb` from the course web site. (Go to Chapter 5 Index and click on the link). Open this Notebook file using version 4.0 or a later one. The first eight cells contain the modules and test statements listed in the top boxes of Figures 5.3–10. The first six of these are marked as *initialization cells*. Before running driver programs, they should be executed by picking up Kernel → Initialize → Execute Initialization. Verify that the output of those six cells agrees with that shown in the bottom boxes of Figures 5.3–6. Then execute the driver programs in Cells 7–8 by clicking on the cells and hitting <Enter>, and compare the output with that shown in Figures 5.9–10. If the output checks out, you may proceed to the Exercises.

EXERCISE 5.1

[C:10] Explain why the `Simplify` command in the test statements of Figure 5.3 says `L>0`. (One way to figure this out is to just say `Ke=Simplify[Ke]` and look at the output. Related question: why does `Mathematica` refuse to simplify `Sqrt[L^2]` to `L` unless one specifies the sign of `L` in the `Simplify` command?

EXERCISE 5.2

[C:10] Explain the logic of the `For` loops in the merge function `MergeElemIntoMasterStiff` of Figure 5.4. What does the operator `+=` do?

EXERCISE 5.3

[C:10] Explain the reason behind the use of `Length` in the modules of Figure 5.6. Why not simply set `nk` and `np` to 6 and 3, respectively?

EXERCISE 5.4

[C:15] Of the seven modules listed in Figures 5.3 through 5.8, two can be used only for the example truss, three can be used for any plane truss, and two can be used for other structures analyzed by the DSM. Identify which ones and briefly state the reasons for your classification.

EXERCISE 5.5

[C:20] Modify the modules `MasterStiffOfExampleTruss`, `IntForcesOfExampleTruss` and the driver program of Figure 5.9 to solve numerically the three-node, two-member truss of Exercise 3.7. Verify that the output reproduces the solution given for that problem.

EXERCISE 5.6

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

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

Figure E5.1. Mystery program for Exercise 5.7.

EXERCISE 5.7

[C:20] Explain what the program of Figure E5.1 does, and the logic behind what it does. (You may want to put it in a cell and execute it.) What modifications would be needed so it can be used for any plane struss?

6

Constructing MoM Members

TABLE OF CONTENTS

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

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

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

§6.1. FORMULATION OF MOM MEMBERS

§6.1.1. What They Look Like

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

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

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

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

In this Chapter we study generic structural members that fit the preceding class. An individual member is identified by (e) but this superscript will be usually suppressed in the equations below

[1] Mechanics of Materials was called Strength of Materials in older texts. The scope of this subject includes bars, beams, shafts, arches, thin plates and shells, but only one-dimensional models are covered in basic undergraduate courses. MoM involves *ab initio* kinematic assumptions such as "plane sections remain plane." An identical acronym is used in Electrical Engineering for something completely different: the Method of Moments.

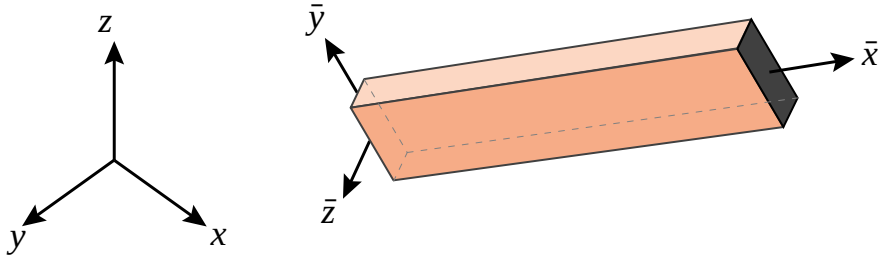


Figure 6.1. A Mechanics of Materials (MoM) member is a structural element one of whose dimensions (the longitudinal dimension) is significantly larger than the other two. Local axes $\{\bar{x}, \bar{y}, \bar{z}\}$ are chosen as indicated. Although the depicted member is prismatic, some applications utilize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

to reduce clutter. The local axes are denoted by $\{\bar{x}, \bar{y}, \bar{z}\}$, with $\bar{x}$ along the longitudinal direction. See Figure 6.1.

The mathematical model of a MoM member is obtained by an idealization process. The model represents the member as a line segment that connects the two end joints, as depicted in Figure 6.2.

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

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

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

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

§6.1.3. Internal Quantities

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

Internal member forces form a finite set of stress-resultant quantities collected in an array $\mathbf{p}$. This set globally characterizes the forces resisted by the material. They are also called generalized stresses in structural theory. Stresses at any point in the member may be recovered if $\mathbf{p}$ is known.

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

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

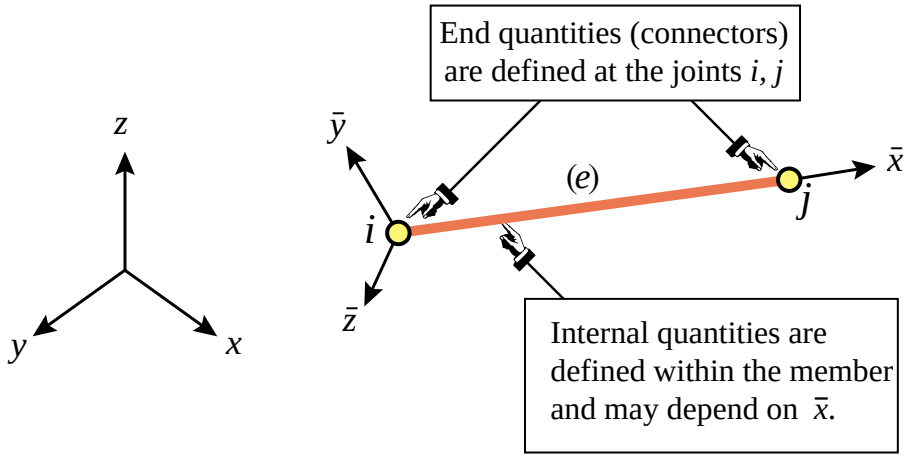


Figure 6.2. The FE mathematical idealization of a MoM member. The model is one-dimensional in $\bar{x}$. The two end joints are the site of end quantities: joint forces and displacements, that interconnect members. The internal quantities characterize the stresses and deformations in the member.

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

§6.1.4. Discrete Field Equations, Tonti Diagram

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

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

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

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

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

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

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

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

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

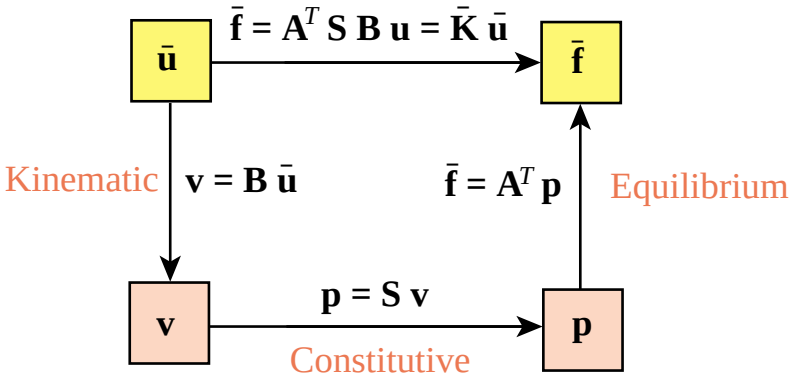


Figure 6.3. Tonti diagram of the three discrete field equations (6.1)–(6.3) and the stiffness equation (6.4) for a *simplex* MoM element. Internal and end quantities appear inside the orange and yellow boxes, respectively.

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

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

- **A** Equilibrium
- **S** Rigidity, material, constitutive[3]
- **B** Compatibility, deformation-displacement, strain-displacement

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

§6.2. SIMPLEX MOM MEMBERS

In this section we assume that the *internal quantities are constant over the member length.* Such members are called *simplex elements*. If so the matrices **A**, **B** and **S** are independent of member cross section, and the derivation of the element stiffness equations is particularly simple.

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

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

Hence the element stiffness matrix is

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

The four preceding matrix equations are diagrammed in Figure 6.3.

[3] The name *rigidity matrix* for **S** is preferable. It is a member integrated version of the cross section constitutive equations. The latter are usually denoted by symbol **R**, as in §6.3.

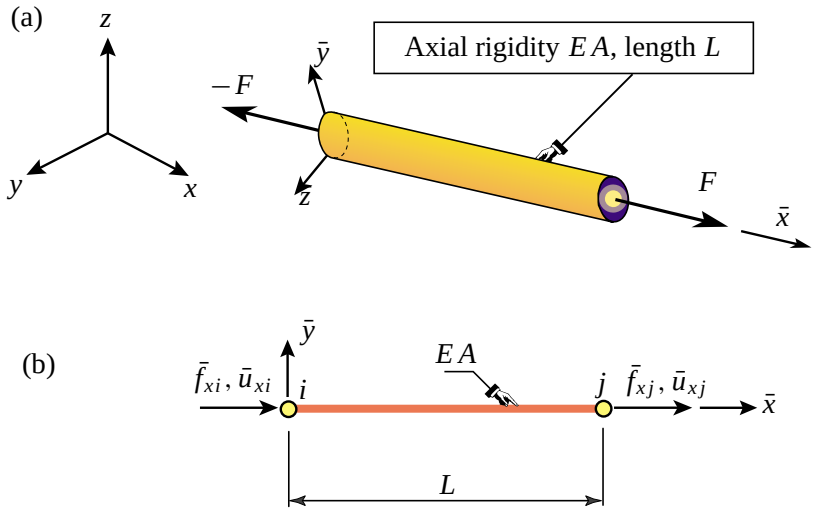


Figure 6.4. The prismatic truss (also called bar) member: (a) individual member in 3D space, (b) idealization as generic member.

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

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

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

REMARK 6.1

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

§6.2.1. The Truss Element Revisited

The simplest example of a MoM element is the prismatic truss (bar) element already derived in Chapter 2. See Figure 6.4. This qualifies as a simplex MoM element since all internal quantities are constant. One minor difference in the derivation below is that the joint displacements and forces in the $\bar{y}$ direction are omitted in the generic element since they contribute nothing to the stiffness equations. In the FEM terminology, freedoms associated with zero stiffness are called *inactive*.

Three choices for internal deformation and force variables are studied below. They illustrate that the resulting element stiffness equations coalesce, as can be expected, since the external quantities are the same.

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

constructed:

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

Hence

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

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

Derivation Using Mean Axial Strain and Axial Force. Instead of d we may use the mean axial strain $\bar{e} = d/L$ as deformation measure whereas F is kept as internal force measure. The only change is that $\mathbf{B}$ becomes $[\,-1 \quad 1\,]/L$ whereas S becomes EA. Matrix $\mathbf{A}$ does not change. The product $\mathbf{A}^T S \mathbf{B}$ gives the same $\bar{\mathbf{K}}$ as in (6.8), as can be expected. Now $\mathbf{A}^T$ is not equal to $\mathbf{B}$ because F and $\bar{e}$ are not conjugate, but they differ only by a factor $1/L$.

Derivation Using Mean Axial Strain and Axial Stress. We keep the mean axial strain $\bar{e} = d/L$ as deformation measure but the mean axial stress $\bar{\sigma} = F/A$, (which is *not* conjugate to $\bar{e}$) is taken as internal force measure. Now $\mathbf{B} = [\,-1 \quad 1\,]/L$, $S = E$ and $\mathbf{A}^T = A\,[\,-1 \quad 1\,]$. The product $\mathbf{A}^T S \mathbf{B}$ gives again the same $\bar{\mathbf{K}}$ shown in (6.8).

§6.2.2. The Spar Element

The *spar* or *shear-web* member has two joints, i and j. It can only resist and transmit a *constant shear force* V in the plane of the web, which is chosen to be the $\{\bar{x}, \bar{y}\}$ plane. See Figure 6.5. This element is often used in modeling high-aspect aircraft wing structures, as illustrated in Figure 6.6.

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

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

Therefore the member stiffness equations are

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

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

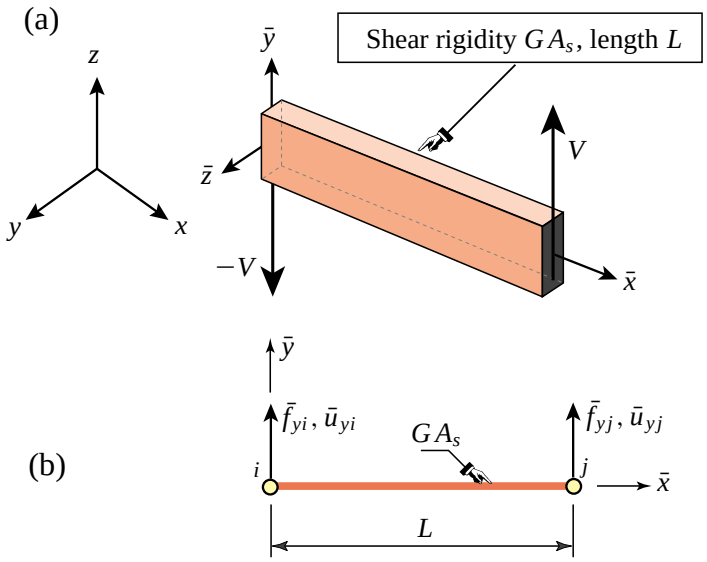


Figure 6.5. The prismatic spar (also called shear-web) member: (a) individual member in 3D space, (b) idealization as generic member.

Note that $\mathbf{A} \neq \mathbf{B}$ as V and γ are not conjugate. However, the difference is easily adjusted for. See Exercise 6.2.

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

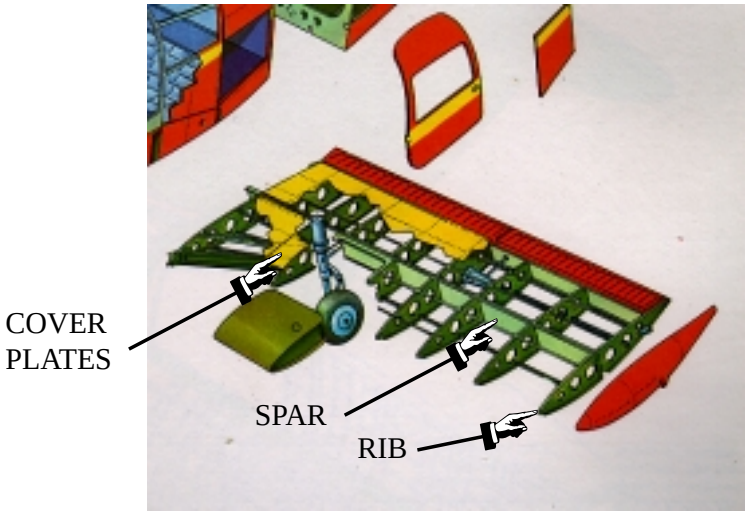


Figure 6.6. Spar members in aircraft wing (Piper Cherokee). For more impressive aircraft structures see CATECS slides.

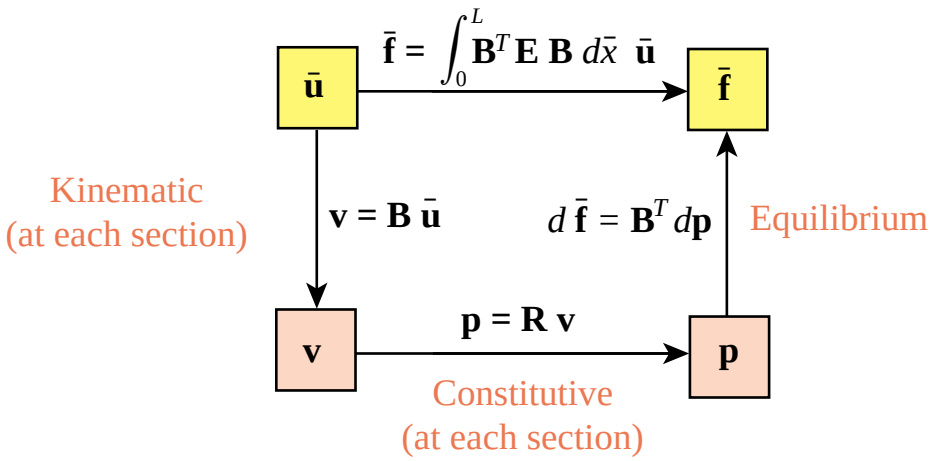


Figure 6.7. Diagram of the discrete field equations for a non-simplex MoM element.

§6.3. NON-SIMPLEX MOM MEMBERS

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

If the element falls under this category, the straightforward matrix multiplication recipe (6.5) cannot be used to construct the element stiffness matrix $\bar{\mathbf{K}}$. This fact can be grasped by observing that $\mathbf{A}(\bar{x})^T\mathbf{S}(\bar{x})\mathbf{B}(\bar{x})$ would depend on $\bar{x}$. On the other hand, $\mathbf{K}$ must be independent of $\bar{x}$ because it relates the end quantities $\bar{\mathbf{u}}$ and $\bar{\mathbf{f}}$.

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

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

Rule 2. From the PVW it may be shown[5] that the force equilibrium equation exists only in a differential sense: $\mathbf{B}^T d\mathbf{p} = d\bar{\mathbf{f}}$. Here d denotes differentiation with respect to $\bar{x}$.[6]

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

Relations that emanate from these rules are diagrammed in Figure 6.7. Internal quantities are now

---

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

[6] The meaning of $d\mathbf{p}$ is simply $\mathbf{p}(\bar{x})\,d\bar{x}$. That is, the differential of internal forces as one passes from cross-section $\bar{x}$ to a neighboring one $\bar{x} + d\bar{x}$. The interpretation of $d\bar{\mathbf{f}}$ is less immediate because $\bar{\mathbf{f}}$ is not a function of $\bar{x}$. It actually means the contribution of that member slice to the building of the node force vector $\bar{\mathbf{f}}$. See Equation (6.12).

[7] Note that symbol $\mathbf{R}$ replaces the $\mathbf{S}$ of the previous section. $\mathbf{R}$ applies to a specific cross section, $\mathbf{S}$ to the entire member. See Exercise 6.9.

eliminated using the differential equilibrium relation:

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

Integrating both sides over the member length L yields

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

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

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

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

Homework Exercises for Chapter 6

Constructing MoM Members

EXERCISE 6.1

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

The result should be

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

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

EXERCISE 6.2

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

EXERCISE 6.3

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

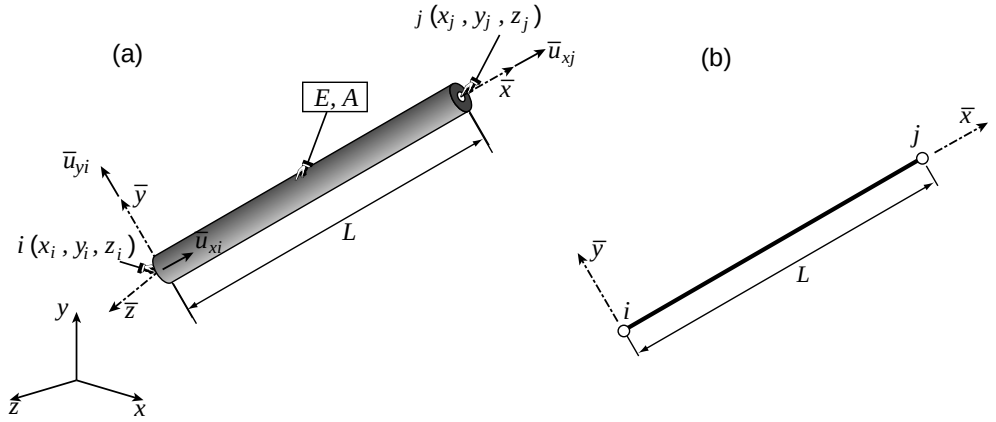


Figure E6.1. Bar element in 3D for Exercise 6.4.

EXERCISE 6.4

[A+N:15] A bar element moving in three dimensional space is defined by the global coordinates $\{x_i, y_i, z_i\}$, $\{x_j, y_j, z_j\}$ of its end nodes i and j, as illustrated in Figure E6.1. The 2×6 displacement transformation matrix $\mathbf{T}$ relates $\bar{\mathbf{u}}^{(e)} = \mathbf{T}\mathbf{u}^{(e)}$. Here $\bar{\mathbf{u}}^{(e)}$ contains the local axial displacements $\bar{u}_{xi}$ and $\bar{u}_{xj}$ whereas $\mathbf{u}^{(e)}$ contains the global displacements $u_{xi}, u_{yi}, u_{zi}, u_{xj}, u_{yj}, u_{zj}$. Show that

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

in which L is the element length, $x_{ji} = x_j - x_i$, etc., and $c_{xji} = x_{ji}/L$, etc., are the direction cosines of the vector going from i to j. Evaluate $\mathbf{T}$ for a bar going from node i at $\{1, 2, 3\}$ to node j at $\{3, 8, 6\}$.

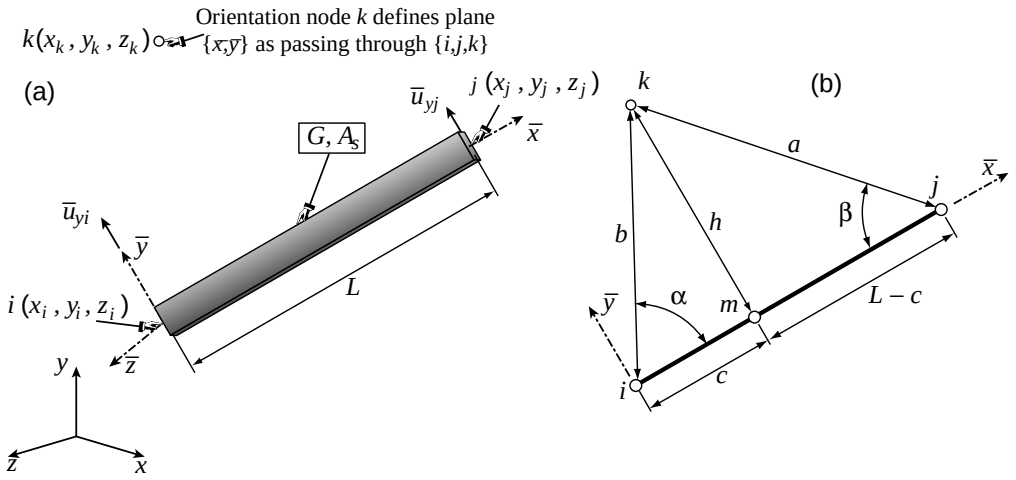


Figure E6.2. Spar element in 3D for Exercise 6.5.

EXERCISE 6.5

[A+N:25 (10+15)] A spar element in three dimensional space is only partially defined by the global coordinates $\{x_i, y_i, z_i\}$, $\{x_j, y_j, z_j\}$ of its end nodes i and j, as illustrated in Figure E6.2. The problem is that axis $\bar{y}$, which defines the direction of shear force transmission, is not uniquely defined by i and j.[8] Most FEM programs use the *orientation node* method to complete the definition. A third node k, not colinear with i and j, is provided by the user. Nodes $\{i, j, k\}$ define the $\{\bar{x}, \bar{y}\}$ plane and consequently $\bar{z}$. The projection of k on line ij is point m and the distance $h > 0$ from m to k is called h as shown in Figure E6.2. The 2×6 displacement transformation matrix $\mathbf{T}$ relates $\bar{\mathbf{u}}^{(e)} = \mathbf{T}\mathbf{u}^{(e)}$. Here $\bar{\mathbf{u}}^{(e)}$ contains the local transverse displacements $\bar{u}_{yi}$ and $\bar{u}_{yj}$ whereas $\mathbf{u}^{(e)}$ contains the global displacements $u_{xi}, u_{yi}, u_{zi}, u_{xj}, u_{yj}, u_{zj}$.

(a) Show that

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

in which $x_{km} = x_k - x_m$, etc., and $c_{xkm} = x_{km}/h$, etc., are the direction cosines of the vector going from m to k.

(b) Work out the formulas to compute the coordinates of point m in terms of the coordinates of $\{i, j, k\}$. Using the notation of Figure E6.2(b) and elementary trigonometry, show that $h = 2A/L$, where $A = \sqrt{p(p-a)(p-b)(p-L)}$ with $p = \frac{1}{2}(L + a + b)$ (Heron's formula), $\cos\alpha = (L^2 + b^2 - a^2)/(2bL)$, $\cos\beta = (L^2 + a^2 - b^2)/(2aL)$, $c = b\cos\alpha$, $L - c = a\cos\beta$, $x_m = x_i(L - c)/L + x_j c/L$, etc. Evaluate $\mathbf{T}$ for a spar member going from node i at $\{1, 2, 3\}$ to node j at $\{3, 8, 6\}$. with k at $\{4, 5, 6\}$.

EXERCISE 6.6

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

EXERCISE 6.7

[A:25] Explain in detail the quick derivation of footnote 5. (Knowledge of the Principle of Virtual Work is required to do this exercise.)

---

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

EXERCISE 6.8

[A:20] Explain how thermal effects can be generally incorporated in the constitutive equation (6.2) to produce an initial force vector.

EXERCISE 6.9

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

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

Here $\bar{\mathbf{R}}$ is an element-averaged constitutive matrix. Apply this result to form $\bar{\mathbf{K}}$ for a truss member of tapered cross section area $A(\bar{x})$. Assume that the area is defined by the linear law $A = A_i(1-\bar{x}/L) + A_j\bar{x}/L$, where A_i and A_j are the end areas at joints i and j, respectively. Take $\mathbf{B}$ to be the same as in the prismatic case discussed in §6.2.1.

EXERCISE 6.10

[A/C+N:30] A prismatic bar element in 3D space is referred to a global coordinate system $\{x, y, z\}$, as in Figure E6.1. The end nodes are located at $\{x_1, y_1, z_1\}$ and $\{x_2, y_2, z_2\}$.[9] The elastic modulus E and the cross section area A are constant along the length. Denote $x_{21} = x_2 - x_1$, $y_{21} = y_2 - y_1$, $z_{21} = z_2 - z_1$ and $L = \sqrt{x_{21}^2 + y_{21}^2 + z_{21}^2}$. Show that the element stiffness matrix in *global* coordinates can be compactly written[10]

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

where

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

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

```
Stiffness3DBar[ncoor_,mprop_,fprop_,opt_]:= Module[
 {x1,x2,y1,y2,z1,z2,x21,y21,z21,Em,Gm,rho,alpha,A,
  num,L,LL,LLL,B,Ke},  {{x1,y1,z1},{x2,y2,z2}}=ncoor;
  {x21,y21,z21}={x2-x1,y2-y1,z2-z1};
```

---

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

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

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

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

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

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

```
    {Em,Gm,rho,alpha}=mprop; {A}=fprop; {num}=opt;
    If [num,{x21,y21,z21,Em,A}=N[{x21,y21,z21,Em,A}]];
    LL=x21^2+y21^2+z21^2; L=PowerExpand[Sqrt[LL]];
    LLL=Simplify[LL*L]; B={{-x21,-y21,-z21,x21,y21,z21}};
    Ke=(Em*A/LLL)*Transpose[B].B;
  Return[Ke]];

  ClearAll[Em,A]; Em=343; A=1;
  ncoor={{0,0,0},{2,6,3}}; mprop={Em,0,0,0}; fprop={A}; opt={False};
  Ke=Stiffness3DBar[ncoor,mprop,fprop,opt];
  Print["Stiffness of 3D Bar Element:"];
  Print[Ke//MatrixForm];
  Print["eigs of Ke: ",Eigenvalues[Ke]];
```

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

7

FEM Modeling: Introduction

TABLE OF CONTENTS

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

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

§7.1. FEM TERMINOLOGY

The ubiquitous term "degrees of freedom," often abbreviated to DOF, has figured prominently in the preceding Chapters. This term, as well as "stiffness matrix" and "force vector," originated in structural mechanics, which is the application for which 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 and developed by Hamilton and Jacobi as a systematic formulation of Newtonian mechanics. Its objects of attention are models of mechanical systems ranging from particles composed of sufficiently large of molecules, through airplanes, to the Solar System.[1] The spatial configuration of any such system is described by its *degrees of freedom*. These are also called *generalized coordinates*. The terms *state variables* and *primary variables* are also used, particularly in mathematically oriented treatments.

If the number of degrees of freedom of the model is finite, the model is called *discrete*, and *continuous* otherwise.

Because FEM is a discretization method, the number of degrees of freedom of a FEM model is necessarily finite. The freedoms are collected in a column vector called **u**. This vector is generally called the *DOF vector* or *state vector*. The term *nodal displacement vector* for **u** is reserved to mechanical applications.

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

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

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

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

---

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

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

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

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

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

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

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

Here $\mathbf{f}_I$ is the initial node force vector introduced in Chapter 4 for effects such as temperature changes, and $\mathbf{f}_M$ is the vector of mechanical forces.

The basic steps of FEM are discussed below in more generality. Although attention is focused on structural problems, most of the steps translate to other applications problems as noted above. The role of FEM in numerical simulation is schematized in Figure 7.1, which is a merged simplification of Figures 1.2 and 1.3. Although this diagram oversimplifies the way FEM is actually used, it serves to illsutrates terminology. It shows the three key simulation steps are: *idealization*, *discretization* and *solution*,and indicates that each step is a source of errors. For example, the discretization error is the discrepancy that appears when the discrete solution is substituted in the mathematical model. The reverse steps: continuification and realization, are far more difficult and ill-posed problems.

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

§7.2. IDEALIZATION

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

§7.2.1. Models

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

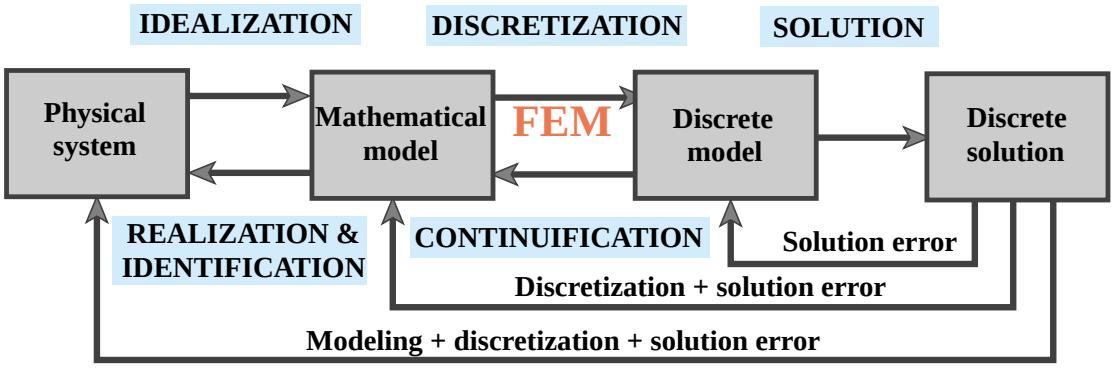


Figure 7.1. A simplified view of the physical simulation process, primarily useful to illustrate modeling terminology.

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

Note the distinction made between *behavior* and *aspects of behavior*. To predict everything, in all physical scales, you must deal with the actual system. A model *abstracts* aspects of interest to the modeler. The qualifier *symbolic* means that a model represents a system in terms of the symbols and language of another discipline. For example, engineering systems may be (and are) modeled with the symbols of mathematics and/or computer sciences.[3]

§7.2.2. Mathematical Models

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

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

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

1. A *very thin* plate model based on Von Karman's coupled membrane-bending theory.
2. A *thin* plate model, such as the classical Kirchhoff's plate theory.
3. A *moderately thick* plate model, for example Mindlin-Reissner plate theory.
4. A *very thick* plate model based on three-dimensional elasticity.

The person responsible for this kind of decision is supposed to be familiar with the advantages,

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

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

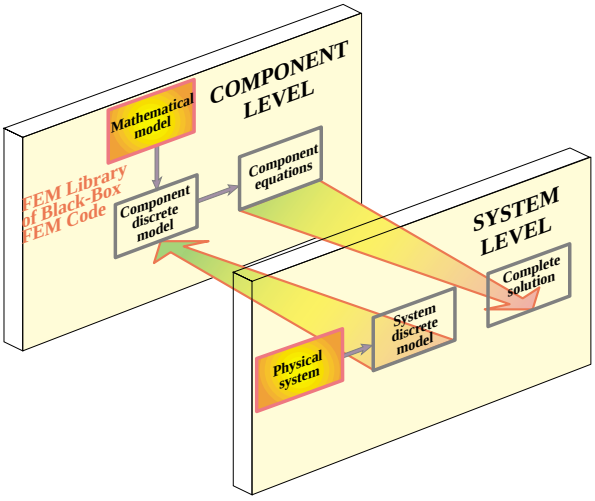


Figure 7.2. A reproduction of Figure 1.5 with some relabeling. Illustrates implicit modeling: picking elements from an existing FEM code consents to an idealization.

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

§7.2.3. Implicit vs. Explicit Modeling

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

A common scenario in industry is: you have to analyze a structure or a substructure, and at your disposal is a "black box" general-purpose finite element program. Those programs offer a *catalog* of element types; for example, bars, beams, plates, shells, axisymmetric solids, general 3D solids, and so on. The moment you choose specific elements from the catalog you automatically accept the mathematical models on which the elements are based. This is *implicit modeling*. Ideally you should be fully aware of the implications of your choice. Providing such "finite element literacy" is one of the objective of this book. Unfortunately many users of commercial programs are unaware of the implied-consent aspect of implicit modeling.

The other extreme happens when you select a mathematical model of the physical problem with

your eyes wide open and *then* either shop around for a finite element program that implements that model, or write the program yourself. This is *explicit modeling*. It requires far more technical expertise, resources, experience and maturity than implicit modeling. But for problems that fall out of the ordinary it may be the right thing to do.

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

§7.3. DISCRETIZATION

§7.3.1. Purpose

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

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

Most problems faced by the engineer either do not yield to analytical treatment or doing so would require a disproportionate amount of effort.[5] The practical way out is numerical simulation. Here is where finite element methods and the digital computer enter the scene.

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

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

§7.3.2. Error Sources and Approximation

Figure 7.1 tries to convey graphically that each simulation step introduces a source of error. In engineering practice modeling errors are by far the most important. But they are difficult and expensive to evaluate, because such *model verification and validation* requires access to and comparison with experimental results. These may be either scarce, or unavailable in the case of a new product.

---

[5] This statement has to be tempered in two respects. First, the wider availability and growing power of computer algebra systems, discussed in Chapter 5, has widened the realm of analytical solutions than can be obtained within a practical time frame. Second, a combination of analytical and numerical techniques is often effective to reduce the dimensionality of the problem and to facilitate parameter studies. Important examples are provided by Fourier analysis, perturbation and boundary-element methods.

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

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

§7.3.3. Other Discretization Methods

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

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

§7.4. THE FINITE ELEMENT METHOD

§7.4.1. Interpretation

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

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

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

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*.

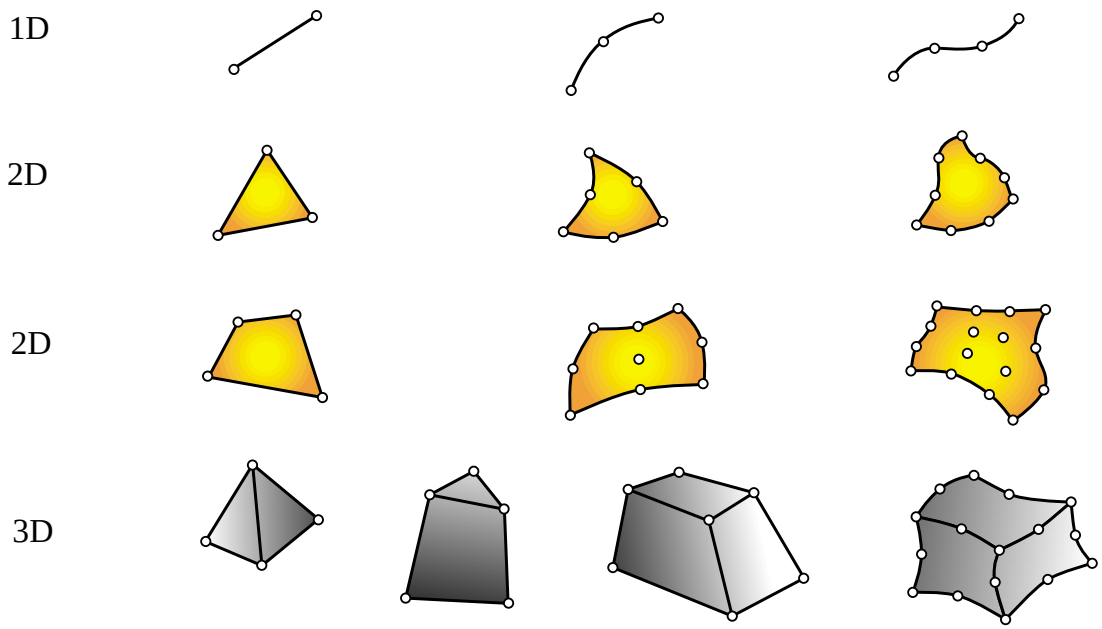


Figure 7.3. Typical finite element geometries in one through three dimensions.

§7.4.2. Element Attributes

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

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

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

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

Nodal points. Each element possesses a set of distinguishing points called *nodal points* or *nodes* for short. Nodes serve two purposes: definition of element geometry, and home for degrees of freedom. They are usually located at the corners or end points of elements, as illustrated in Figure 7.3; in the so-called refined or higher-order elements nodes are also placed on sides or faces, as well as the interior of the element.

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

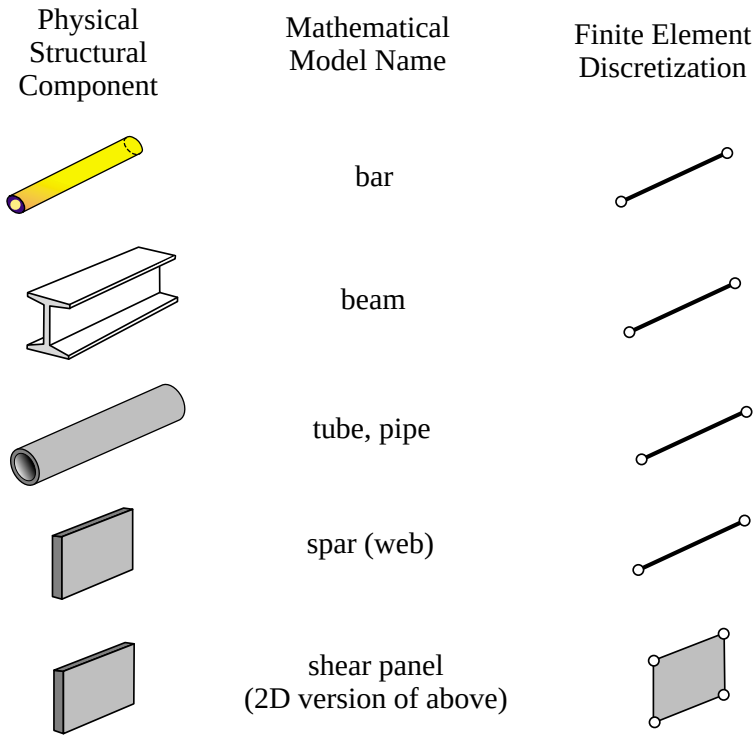


Figure 7.4. Examples of primitive structural elements.

Geometry. The geometry of the element is defined by the placement of the nodal points. Most elements used in practice have fairly simple geometries. In one-dimension, elements are usually straight lines or curved segments. In two dimensions they are of triangular or quadrilateral shape. In three dimensions the three common shapes are tetrahedra, pentahedra (also called wedges or prisms), and hexahedra (also called cuboids or "bricks"). See Figure 7.3.

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

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

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

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

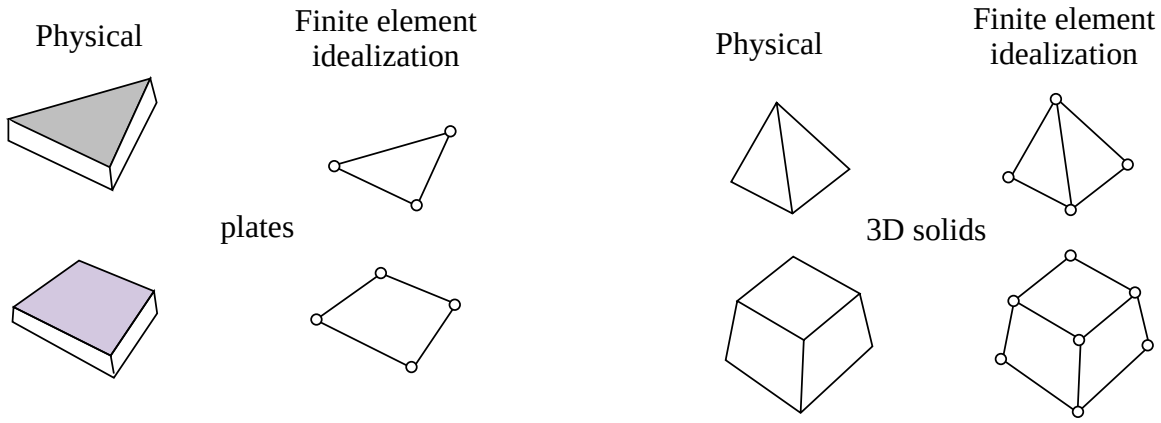


Figure 7.5. Continuum element examples.

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

§7.5. CLASSIFICATION OF MECHANICAL ELEMENTS

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

§7.5.1. Primitive Structural Elements

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

§7.5.2. Continuum Elements

These do not resemble fabricated structural components at all. They result from the subdivision of "blobs" of continua, or of structural components viewed as continua. Unlike structural elements, continuum elements are better understood in terms of their mathematical interpretation. Examples: plates, slices, shells, axisymmetric solids, general solids. See Figure 7.5.

§7.5.3. Special Elements

Special elements partake of the characteristics of structural and continuum elements. They are derived from a continuum mechanics standpoint but include features closely related to the physics of the problem. Examples: crack elements for fracture mechanics applications, shear panels, infinite and semi-infinite elements, contact and penalty elements, rigid-body elements. See Figure 7.6.

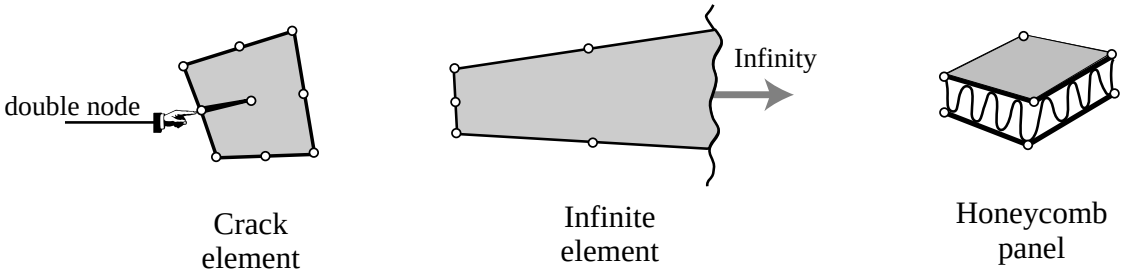


Figure 7.6. Special element examples.

§7.5.4. Macroelements

Macroelements are also called mesh units and superelements, although the latter term overlaps with substructures (defined below). These often resemble structural components, but are fabricated with simpler elements. See Figure 7.7.

§7.5.5. Substructures

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

§7.6. ASSEMBLY

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

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

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

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

§7.7. BOUNDARY CONDITIONS

A key strength of the FEM is the ease and elegance with which it handles arbitrary boundary and interface conditions. This power, however, has a down side. One of the biggest hurdles a FEM newcomer faces is the understanding and proper handling of boundary conditions. In the present Section we summarize some basic rules for treating boundary conditions. The following Chapter provides specific rules and examples.

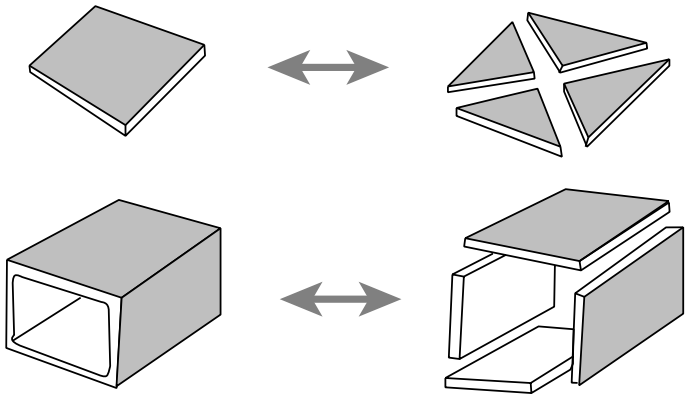


Figure 7.7. Macroelement examples.

§7.7.1. Essential and Natural B.C.

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

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

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

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

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

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

§7.7.2. Boundary Conditions in Structural Problems

In mechanical problems, essential boundary conditions are those that involve *displacements* (but not strain-type displacement derivatives). The support conditions for the truss problem furnish a particularly simple example. But there are more general boundary conditions that occur in practice. A structural engineer must be familiar with displacement B.C. of the following types.

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

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

Ignorable freedoms. To suppress displacements that are irrelevant to the problem. (In classical dynamics these are called *ignorable coordinates*.) Even experienced users of finite element programs are sometimes baffled by this kind. An example are rotational degrees of freedom normal to shell surfaces.

Connection constraints. To provide connectivity to adjoining structures or substructures, or to specify relations between degrees of freedom. Many conditions of this type can be subsumed under the label *multipoint constraints* or *multifreedom constraints*, which can be notoriously difficult to handle from a numerical standpoint. These will be covered in Chapters 9 and 10.

8

FEM Modeling: Mesh, Loads and BCs

TABLE OF CONTENTS

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

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

§8.1. GENERAL RECOMMENDATIONS

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

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

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

§8.2. GUIDELINES ON ELEMENT LAYOUT

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

§8.2.1. Mesh Refinement

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

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

The examples in Figure 8.1 illustrate some of these "danger regions." Away from such regions one can use a fairly coarse discretization within constraints imposed by the need of representing the structural geometry, loading and support conditions reasonably well.

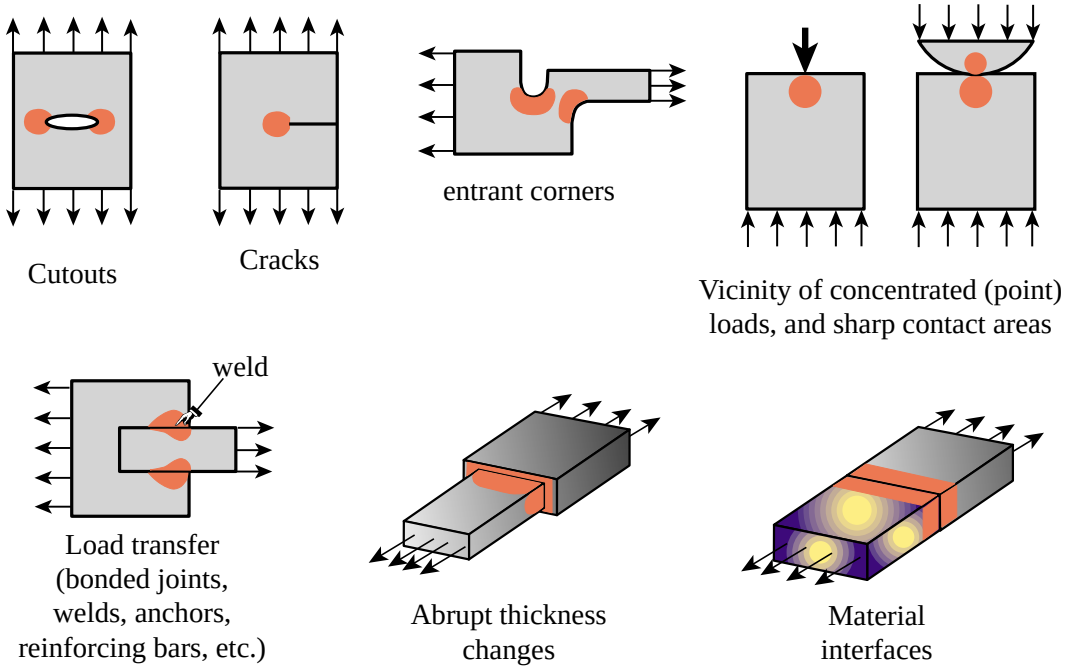


Figure 8.1. Some situations where a locally refined finite element discretization (in the red-colored areas) is recommended.

§8.2.2. Element Aspect Ratios

When discretizing two and three dimensional problems, try to avoid finite elements of high aspect ratios: elongated or "skinny" elements, as the ones illustrated in Figure 8.2.[1] As a rough guideline, elements with aspect ratios exceeding 3 should be viewed with caution and those exceeding 10 with alarm. Such elements will not necessarily produce bad results — that depends on the loading and boundary conditions of the problem — but do introduce the potential for trouble.

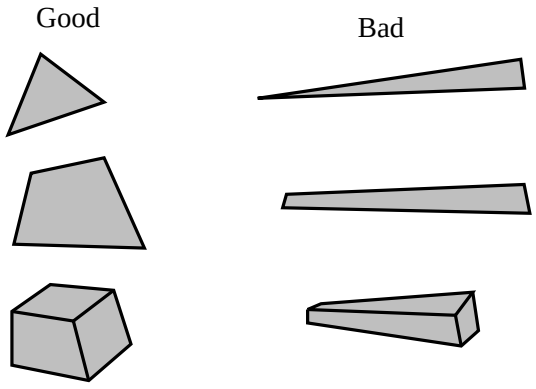


Figure 8.2.Elements of good and bad aspect ratios.

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

REMARK 8.1

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

§8.2.3. Physical Interfaces

A physical interface, resulting from example from a change in material, should also be an interelement boundary. That is, *elements must not cross interfaces*. See Figure 8.3.

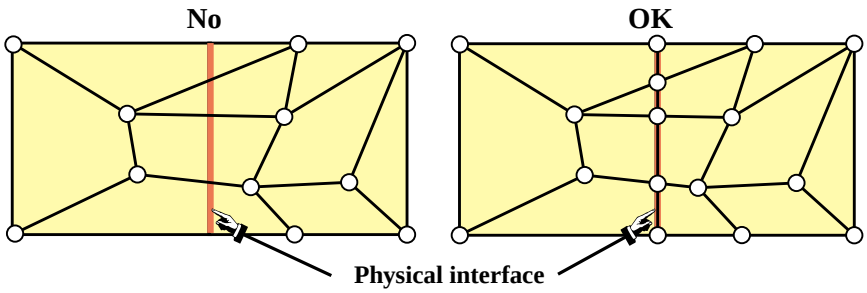


Figure 8.3. Illustration of the rule that elements should not cross material interfaces.

§8.2.4. Preferred Shapes

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

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

§8.3. DIRECT LUMPING OF DISTRIBUTED LOADS

In practical structural problems, distributed loads are more common than concentrated (point) loads.[2] Distributed loads may be of surface or volume type.

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

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

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

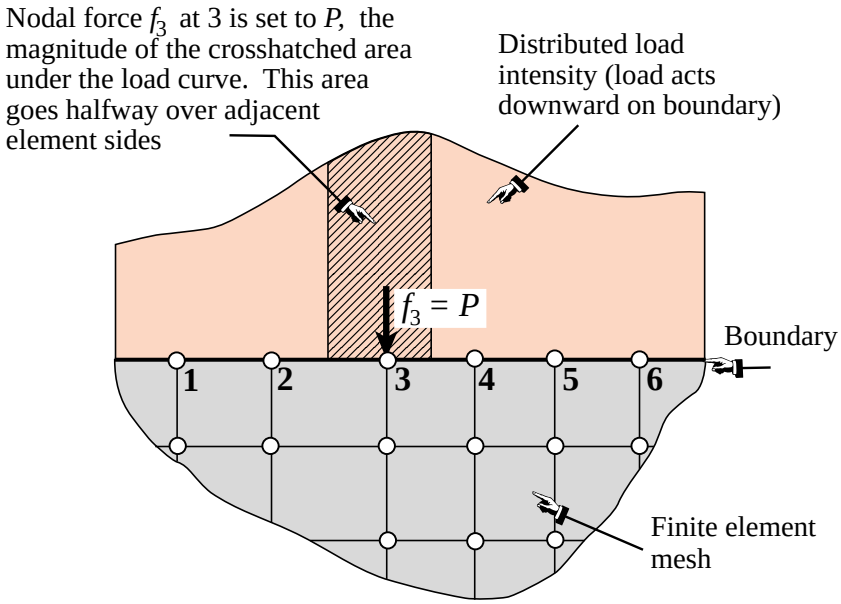


Figure 8.4. NbN direct lumping of distributed load, illustrated for a 2D problem.

A derived type: line loads, result from the integration of surface loads along one transverse direction, or of volume loads along two transverse directions. Line loads are measured in force per unit length.

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

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

§8.3.1. Node by Node (NbN) Lumping

The node by node (NbN) lumping method is graphically explained in Figure 8.4. This example shows a distributed surface loading acting on the straight boundary of a two-dimensional FE mesh. (The load is assumed to have been integrated through the thickness normal to the figure, so it is actually a line load measured as force per unit length.)

The procedure is also called *tributary region* or *contributing region* method. For the example of Figure 8.4, each boundary node is assigned a *tributary region* around it that extends halfway to the adjacent nodes. The force contribution P of the cross-hatched area is directly assigned to node 3.

This method has the advantage of not requiring the computation of centroids, as required in the EbE technique discussed in the next subsection. For this reason it is often preferred in hand computations. It can be extended to three-dimensional meshes as well as volume loads.[3] It should

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

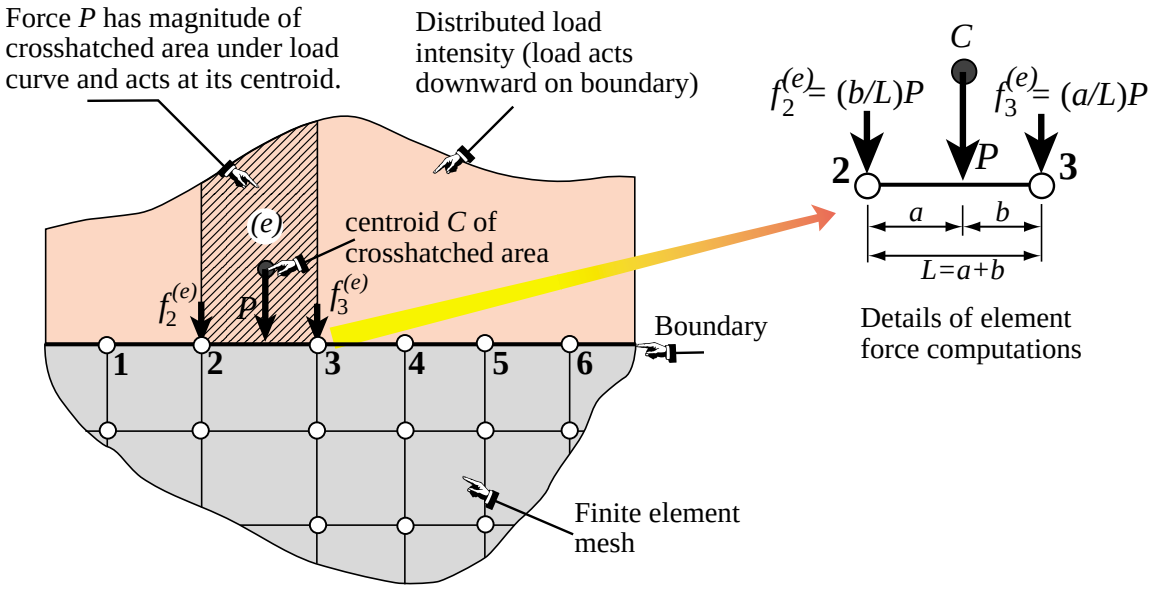


Figure 8.5. EbE direct lumping of distributed load, illustrated for a 2D problem.

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

§8.3.2. Element by Element (EbE) Lumping

In this variant the distributed loads are divided over element domains. The resultant load is assigned to the centroid of the load diagram, and apportioned to the element nodes by statics. A node force is obtained by adding the contributions from all elements meeting at that node. The procedure is illustrated in Figure 8.5, which shows details of the computation over segment 2-3. The total force at node 3, for instance, would that contributed by segments 2-3 and 3-4.

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

The procedure is not applicable if the centroidal resultant load cannot be apportioned by statics. This happens if the element has midside faces or internal nodes in addition to corner nodes, or if it has rotational degrees of freedom. For those elements the variational approach is preferable.

§8.4. BOUNDARY CONDITIONS

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

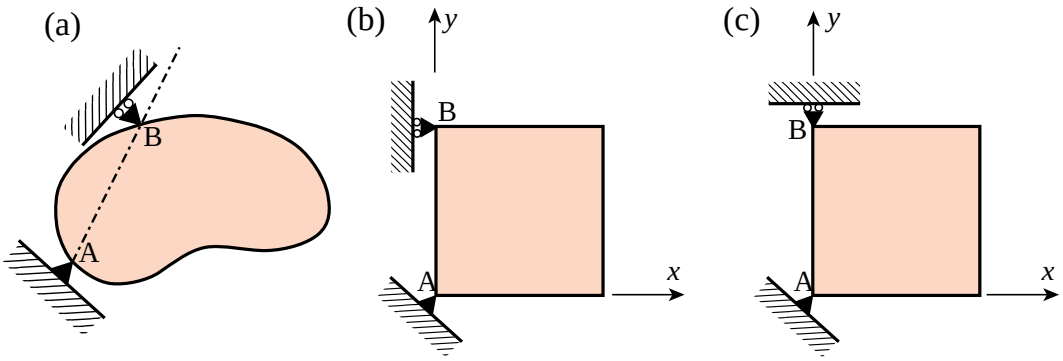


Figure 8.6. Examples of restraining a body against two-dimensional rigid body motions.

In structural problems formulated by the DSM, the recipe of §7.7.1 that distinguishes between essential and natural BC is: if it directly involves the nodal freedoms, such as displacements or rotations, it is essential. Otherwise it is natural. Conditions involving applied loads are natural. Essential BCs take precedence over natural BCs.

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

§8.5. SUPPORT CONDITIONS

Supports are used to restrain structures against relative rigid body motions. This is done by attaching them to Earth ground (through foundations, anchors or similar devices), or to a "ground structure" which is viewed as the external environment.[4] The resulting boundary conditions are often called *motion constraints*. In what follows we analyze two- and three-dimensional motions separately.

§8.5.1. Supporting Two Dimensional Bodies

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

In Figure 8.4, support A provides *translational* restraint, whereas support B, together with A, provides *rotational* restraint. In finite element terminology, we say that we *delete* (fix, remove, preclude) all translational displacements at point A, and that we delete the translational degree of freedom directed along the normal to the AB direction at point B. This body is free to distort in any manner without the supports imposing any displacement constraints.

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

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

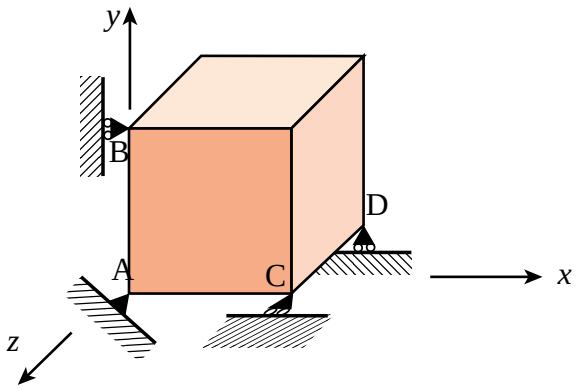


Figure 8.7. Suppressing RBM freedoms in a three-dimensional body.

Figure 8.6(b) shows a simplified version of Figure 8.6(a). Here the line AB is parallel to the global y axis. We simply delete the x and y translations at point A, and the x translation at point B. If the roller support at B is modified as in Figure 6(c), it becomes ineffective in constraining the infinitesimal rotational motion about point A because the rolling direction is normal to AB. The configuration of Figure 6(c) is called a *kinematic mechanism*, and can be flagged by a singular modified stiffness matrix.

§8.5.2. Supporting Three Dimensional Bodies

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

§8.6. SYMMETRY AND ANTISYMMETRY CONDITIONS

Engineers doing finite element analysis should be on the lookout for conditions of *symmetry* or *antisymmetry*. Judicious use of these conditions allows only a portion of the structure to be analyzed, with a consequent saving in data preparation and computer processing time.[5]

§8.6.1. Visualization

Recognition of symmetry and antisymmetry conditions can be done by either visualization of the displacement field, or by imagining certain rotational ot reflection motions. Both techniques are illustrated for the two-dimensional case.

A *symmetry line* in two-dimensional motion can be recognized by remembering the "mirror" displacement pattern shown in Figure 8.8(a). Alternatively, a 180° rotation of the body about the symmetry line reproduces exactly the original problem.

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

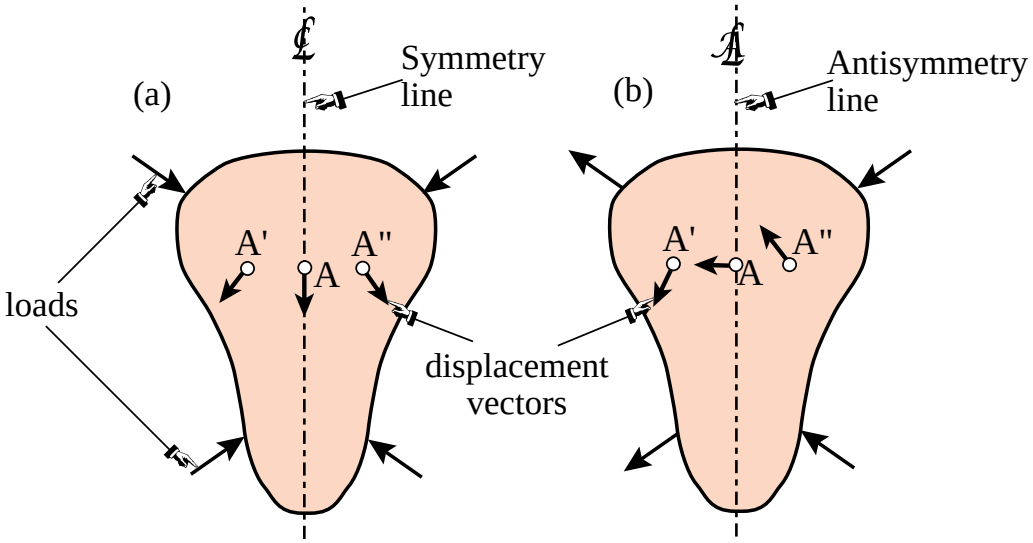


Figure 8.8. Visualizing symmetry and antisymmetry lines.

An *antisymmetry line* can be recognized by the displacement pattern illustrated in Figure 8.8(b). Alternatively, a 180° rotation of the body about the antisymmetry line reproduces exactly the original problem except that all applied loads are reversed.

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

§8.6.2. Effect of Loading Patterns

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

Consider the plate structure shown in Figure 8.9(a). This structure is symmetrically loaded on the x-y plane. Applying the recognition patterns stated above one concludes that the structure is *doubly symmetric* in both geometry and loading. It is evident that no displacements in the x-direction are possible for any point on the y-axis, and that no y displacements are possible for points on the x axis. A finite element model of this structure may look like that shown in Figure 8.8(b).

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

REMARK 8.2

For the antisymmetric loading case, one node point has to be constrained against vertical motion. The choice is arbitrary and amounts only to an adjustment on the overall (rigid-body) vertical motion. In Figure 8.10(b) the center point C has been chosen to be that vertically-constrained node. But any other node could be selected as well; for example A or D. The important thing is not to overconstrain the structure by applying more than one y constraint.

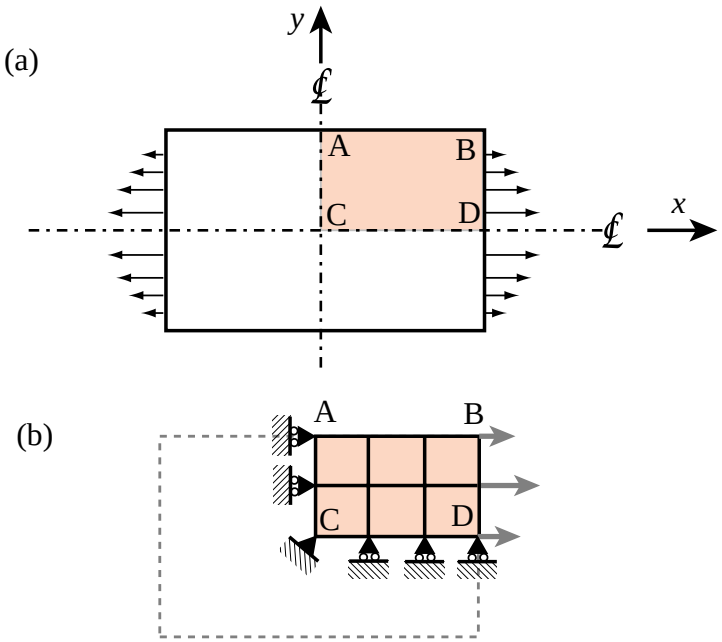


Figure 8.9. A doubly symmetric structure under symmetric loading.

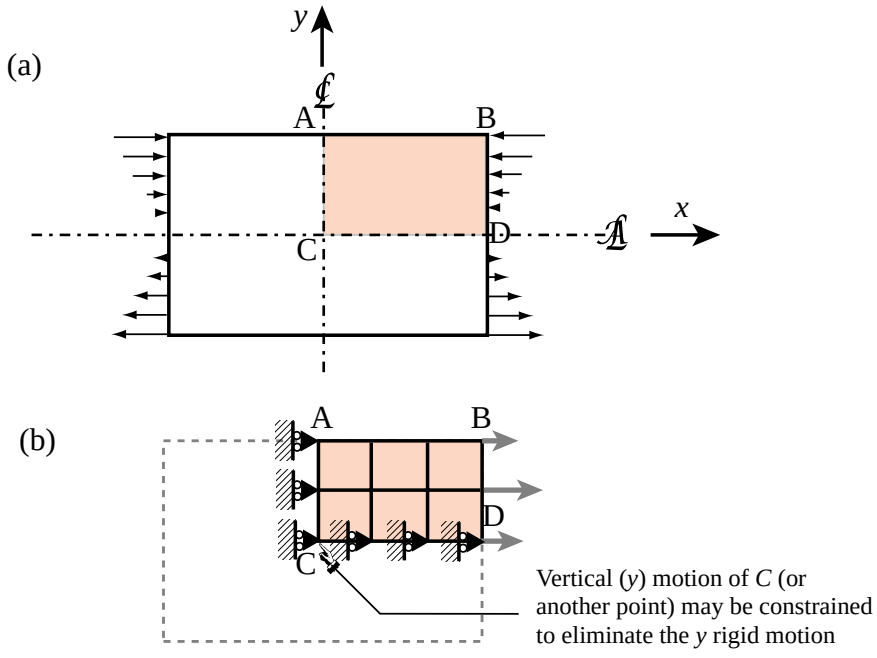


Figure 8.10. A doubly symmetric structure under antisymmetric loading.

Homework Exercises for Chapters 7 and 8

FEM Modeling

EXERCISE 8.1

[D:10] The plate structure shown in Figure E8.1 is loaded and deforms in the plane of the figure. The applied load at D and the supports at I and N extend over a fairly narrow area. Give a list of what you think are the likely "trouble spots" that would require a locally finer finite element mesh to capture high stress gradients.

Identify those spots by its letter and a reason. For example, D: vicinity of point load.

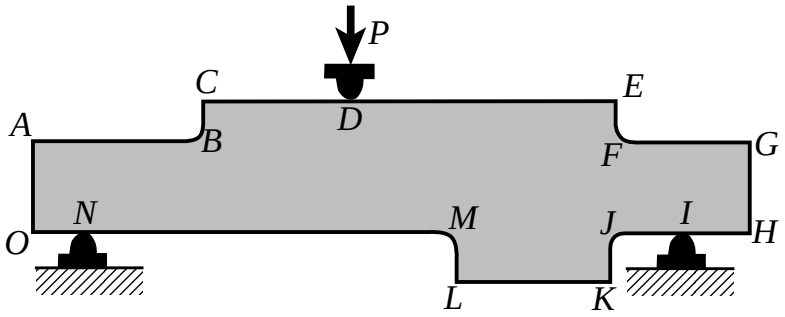


Figure E8.1. Plate structure for Exercise 8.1.

EXERCISE 8.2

[D:15] Part of a two-dimensional FE mesh has been set up as indicated in Figure E8.2. Region $ABCD$ is still unmeshed. Draw a *transition mesh* within that region that correctly merges with the regular grids shown, uses 4-node quadrilateral elements (quadrilaterals with corner nodes only), and *avoids triangles*. *Note*: There are several (equally acceptable) solutions.

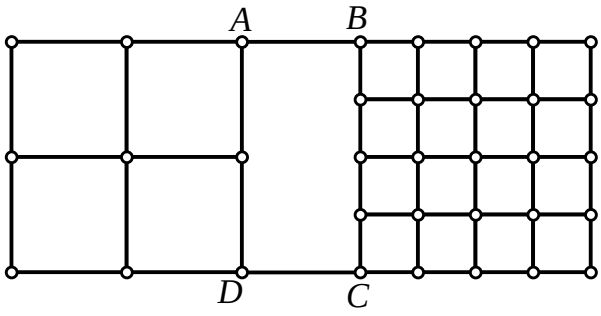


Figure E8.2. Plate structure for Exercise 8.2.

EXERCISE 8.3

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

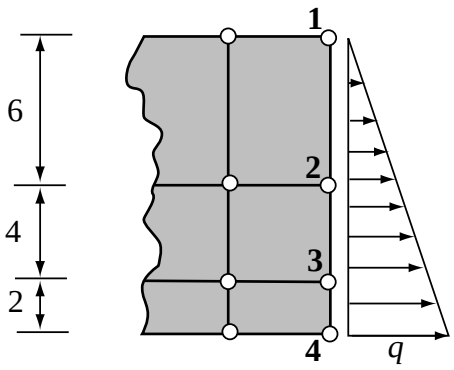


Figure E8.3. Mesh layout for Exercise 8.3.

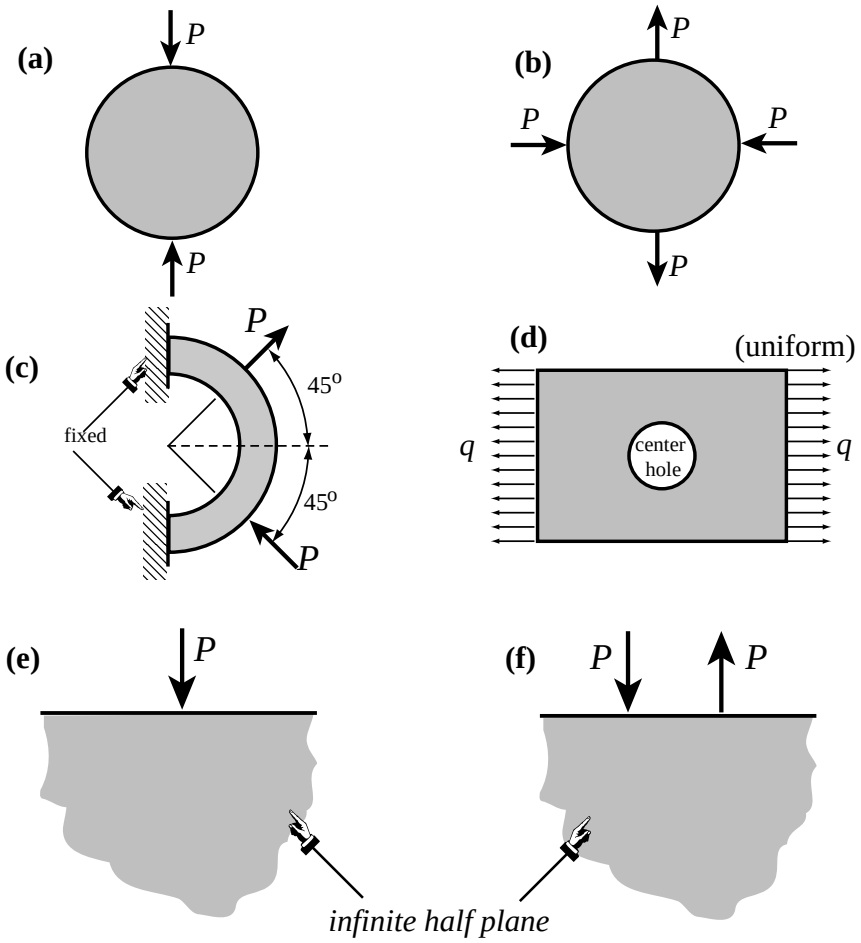


Figure E8.4. Problems for Exercise 8.4.

EXERCISE 8.4

[D:15] Identify the symmetry and antisymmetry lines in the two-dimensional problems illustrated in Figure E8.4. They are: (a) a circular disk under two diametrically opposite point forces (the famous "Brazilian test" for concrete); (b) the same disk under two diametrically opposite force pairs; (c) a clamped semiannulus under

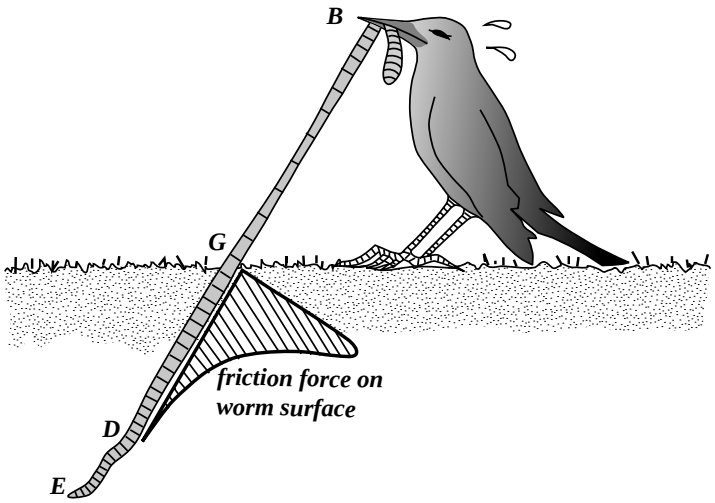


Figure E8.5. The hungry bird.

a force pair oriented as shown; (d) a stretched rectangular plate with a central circular hole. Finally (e) and (f) are half-planes under concentrated loads.[6]

Having identified those symmetry/antisymmetry lines, state whether it is possible to cut the complete structure to one half or one quarter before laying out a finite element mesh. Then draw a coarse FE mesh indicating, with rollers or fixed supports, which kind of displacement BCs you would specify on the symmetry or antisymmetry lines. *Note: Do all sketches on your paper, not on the printed figures*.

EXERCISE 8.5

[D:20] You (a finite element guru) pass away and come back to the next life as an intelligent but hungry bird. Looking around, you notice a succulent big worm taking a peek at the weather. You grab one end and pull for dinner; see Figure E8.5.

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

(a) The worm is simply modeled as a string of one-dimensional (bar) elements. The "worm axial force" is of course constant from the beak B to ground level G, then decreases rapidly because of soil friction (which varies roughly as plotted in the figure above) and drops to nearly zero over DE. Sketch how a good "worm-element mesh" should look like to capture the axial force well.

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

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

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

Briefly explain your decisions. Dont write equations.

---

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

EXERCISE 8.6

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

EXERCISE 8.7

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

EXERCISE 8.8

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

EXERCISE 8.9

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

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

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

(c) A symmetry plane and an antisymmetry plane planes intersect. State the kinematic properties of the intersection line. Can the angle between the planes be arbitrary?

(d) Can two antisymmetry planes intersect?

(e) Three symmetry planes intersect. State the kinematic properties of the intersection point.

EXERCISE 8.10

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

EXERCISE 8.11

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

EXERCISE 8.12

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

9

MultiFreedom Constraints I

TABLE OF CONTENTS

§9.1. CLASSIFICATION OF CONSTRAINT CONDITIONS

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

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

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

For example:

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

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

§9.1.1. MultiFreedom Constraints

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

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

where function F vanishes if all its nodal displacement arguments do. Equation (9.3), where all displacement components are in the left-hand side, is called the *canonical form* of the constraint.

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

The constraint is called *homogeneous* if, upon transfering all terms that depend on displacement components to the left-hand side, the right-hand side — the "prescribed value" in (9.3) — is zero. It is called *non-homogeneous* otherwise.

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

REMARK 9.1

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.

§9.1.2. MFC Examples

Here are three examples of MFCs:

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

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

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

The last one is multipoint, nonlinear and homogeneous. Geometrically it expresses that the distance between nodes 3 and 5 in two-dimensional motions 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 course.

§9.1.3. *MFC Matrix Forms

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

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

or, in direct matrix notation:

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

in which index i ($i = 1, 2, \ldots$) identifies the constraint, $\bar{\mathbf{a}}_i$ is a row vector, $\bar{\mathbf{u}}_i$ collects the set of degrees of freedom that participate in the constraint, and g_i is the right hand side scalar (0.25 above). The bar over **a** and **u** distinguishes (9.6) from the expanded form (9.8) discussed below.

For method description and general proofs it is often convenient to expand matrix forms so that they embody *all* degrees of freedom. For example, if (9.5) is part of a two-dimensional finite element model with 12 freedoms: u_{x1}, u_{y1}, ... u_{y6}, the left-hand side row vector may be expanded with nine zeros as follows

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

in which case the matrix notation

$$\mathbf{a}_i\mathbf{u} = g_i \tag{9.8}$$

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

$$\mathbf{A}\mathbf{u} = \mathbf{g}, \tag{9.9}$$

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

§9.2. METHODS FOR IMPOSING MULTIFREEDOM CONSTRAINTS

Accounting for multifreedom constraints is done — at least conceptually — by changing the assembled master stiffness equations to produce a modified system of equations. The modification process is also called *constraint application* or *constraint imposition*. The modified system is the one submitted to the equation solver.

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

1. *Master-Slave Elimination*. The degrees of freedom involved in each MFC are separated into master and slave freedoms. The slave freedoms are then explicitly eliminated. The modified equations do not contain the slave freedoms.
2. *Penalty Augmentation*. Also called the *penalty function method*. Each MFC is viewed as the presence of a fictitious elastic structural element called *penalty element* that enforces it approximately. This element is parametrized by a numerical *weight*. The exact constraint is recovered if the weight goes to infinity. The MFCs are imposed by augmenting the finite element model with the penalty elements.
3. *Lagrange Multiplier Adjunction*. For each MFC an additional unknown is adjoined to the master stiffness equations. Physically this set of unknowns represent *constraint forces* that would enforce the constraints exactly should 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 Chapters 3 and 4 for single-freedom constraints. The master stiffness equations are assembled ignoring all constraints. Then the MFCs are imposed by appropriate modification of those equations. There are, however, two important practical differences:

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

REMARK 9.2

The three methods are also applicable, as can be expected, to the simpler case of a single-freedom constraint such as (9.2). For most situations, however, the generality afforded by the penalty function and Lagrange multiplier methods are not warranted. The hand-oriented reduction process discussed in Chapters 3 and 4 is in fact a special case of the master-slave elimination method in which "there is no master."

REMARK 9.3

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

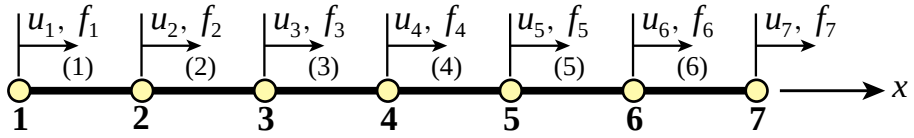


Figure 9.1. A one-dimensional problem discretized with six bar finite elements. The seven nodes may move only along the x direction. Subscript x is omitted from the u's and f's for simplicity.

§9.3. THE EXAMPLE STRUCTURE

The one-dimensional finite element discretization shown in Figure 9.1 will be used throughout to illustrate the three MFC application methods. This structure consists of six bar elements connected by seven nodes that can only displace in the x direction.

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

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

or

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

The nonzero stiffness coefficients K_{ij} in (9.10) depend on the bar rigidity properties. For example, if $E^{(e)}A^{(e)}/L^{(e)} = 100$ for each element $e = 1, \ldots, 6$, then $K_{11} = K_{77} = 100$, $K_{22} = \ldots = K_{66} = 200$, $K_{12} = K_{23} = \ldots = K_{67} = -100$. However, for the purposes of the following treatment the coefficients may be kept arbitrary. The component index x in the nodal displacements u and nodal forces f has been omitted for brevity.

Now let us specify a multifreedom constraint that states that nodes 2 and 6 are to move by the same amount:

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

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

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

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

[1] This physical interpretation is exploited in the penalty method described in the next Chapter. In two and three dimensions rigid link constraints are more complicated.

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

§9.4. THE MASTER-SLAVE METHOD

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

§9.4.1. A One-Constraint Example

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

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

This is the required transformation relation. In compact form:

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

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

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

Carrying out the indicated matrix multiplications yields

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

Equation (9.17) is a new linear system containing 6 equations in the remaining 6 unknowns: u_1, u_2, u_3 , u_4, u_5 and u_7. Upon solving it, u_6 is recovered from the constraint (9.12).

REMARK 9.4

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

REMARK 9.5

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

REMARK 9.6

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

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

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

§9.4.2. Several Homogeneous MFCs

The matrix equation (9.16) in fact holds for the general case of multiple homogeneous linear constraints. Direct establishment of the transformation equation, however, is more complicated if slave freedoms in one constraint appear as masters in another. To illustrate this point, suppose that for the example system we have three homogeneous multifreedom constraints:

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

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

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

Observe that solving for u_3 from the third constraint brings u_4 to the right-hand side. But because u_4 is also a slave freedom (it was chosen as such for the second constraint) it is replaced in favor of u_1 using $u_4 = -\frac{1}{4}u_1$. The matrix form of the transformation (9.20) is

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

The modified system is now formed through the congruential transformation (9.16). Note that the slave freedoms selected from each constraint must be distinct; for example the choice u_6, u_4, u_4 would be inadmissible as long as the constraints are independent. This rule is easy to enforce when slave freedoms are chosen by hand, but can lead to implementation and numerical difficulties when it is programmed as an automated procedure, as further discussed later.

REMARK 9.7

The three MFCs (9.20) with u_6, u_4 and u_2 chosen as slaves and u_1, u_2 and u_5 chosen as masters, may be presented in the partitioned matrix form:

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

This may be compactly written $\mathbf{A}_s\mathbf{u}_s + \mathbf{A}_m\mathbf{u}_m = \mathbf{0}$. Solving for the slave freedoms gives $\mathbf{u}_s = -\mathbf{A}_s^{-1}\mathbf{A}_m\mathbf{u}_m$. Expanding with zeros to fill out $\mathbf{u}$ and $\hat{\mathbf{u}}$ produces (9.21). This general matrix form is considered in §9.4.4. Note that non-singularity of $\mathbf{A}_s$ is essential for this method to work.

§9.4.3. Nonhomogeneous MFCs

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

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

Nonzero RHS values such as 0.2 in (9.23) may be often interpreted physically as "gaps" (thus the use of the symbol $\mathbf{g}$ in the matrix form). Chose u_6 again as slave: $u_6 = u_2 - 0.2$, and build the transformation

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

In compact matrix notation,

$$\boxed{\mathbf{u} = \mathbf{T}\,\hat{\mathbf{u}} + \mathbf{g}.} \tag{9.25}$$

Here the constraint gap vector $\mathbf{g}$ is nonzero and $\mathbf{T}$ is the same as before. To get the modified system applying the shortcut rule of Remark 9.4, premultiply both sides of (9.25) by $\mathbf{T}^T\mathbf{K}$, replace $\mathbf{Ku}$ by $\mathbf{f}$, and pass the data to the RHS:

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

Upon solving (9.26) for $\hat{\mathbf{u}}$, the complete displacement vector is recovered from (9.25).

For the MFC (9.22) this technique gives the system

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

See Exercise 9.2 for multiple non-homogeneous MFCs.

§9.4.4. *The General Case

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

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

The MFCs may be written in matrix form as

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

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

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

Inserting into the partitioned stiffness matrix and symmetrizing

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

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

§9.4.5. *Retaining the Original Freedoms

A potential disadvantage of the master-slave method in computer work is that it requires a rearrangement of the original stiffness equations because $\hat{\mathbf{u}}$ is a subset of $\mathbf{u}$. 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 trickiness it is possible to maintain the original freedom ordering. Let us display it for the example problem under (9.13). Instead of (9.14), use the *square* transformation

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

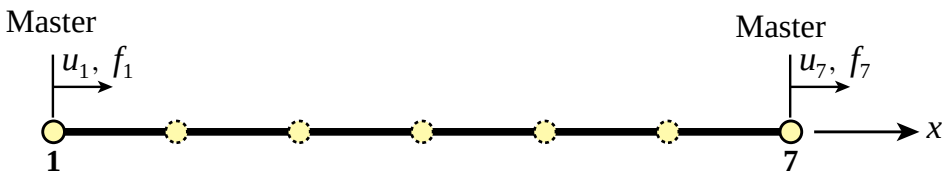


Figure 9.2. Model reduction of the example structure of Figure 9.1 to the end freedoms.

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

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

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

§9.4.6. Model Reduction by Kinematic Constraints

The congruential transformation equations (9.16) and (9.26) have additional applications beyond the master-slave method. An important one is *model reduction by kinematic constraints*. Through this procedure the number of DOF of a static or dynamic FEM model is reduced by a significant number, typically to 1% – 10% of the original number. This is done by taking a lot of slaves and a few masters. Only the masters are left after the transformation. The reduced model is commonly used in subsequent calculations as component of a larger system, particularly during design or in parameter identification.

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

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

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

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

```
(* Model Reduction Example *)
ClearAll[K11,K12,K22,K23,K33,K34,K44,K45,K55,K56,K66,
         f1,f2,f3,f4,f5,f6];
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:"];
Print["Khat(1,1)=",Khat[[1,1]],"\nKhat(1,2)=",Khat[[1,2]],
    "\nKhat(2,2)=",Khat[[2,2]] ];
Print["Modified Force:"];
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:

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

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

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

Modified Force:

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

Figure 9.3. *Mathematica* script for the model reduction example of Figure 9.2.

where

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

This reduces the order of the FEM model from 7 to 2. A *Mathematica* script to do the reduction is shown in Figure 9.3. The key feature is that the masters are picked *a priori*, as the freedoms to be retained in the model for further use.

REMARK 9.8

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

§9.4.7. Assessment of the Master-Slave Method

What are the good and bad points of this constraint imposition method? It enjoys the advantage of being exact (except for inevitable solution errors) and of reducing the number of unknowns. The concept is also easy to explain. The main implementation drawback is the complexity of the general case as can be seen by studying (9.28) through (9.31). The complexity is due to three factors:

1. The equations may have to be rearranged because of the disappearance of the slave freedoms. This drawback can be alleviated, however, through the placeholder trick outlined in §9.4.5.
2. An auxiliary linear system, namely (9.30), has to be assembled and solved to produce the transformation matrix **T** and vector **g**.
3. The transformation process may generate many additional matrix terms. If a sparse matrix storage scheme is used for **K**, the logic for allocating memory and storing these entries can be difficult and expensive.

The level of complexity depends on the generality allowed as well as on programming decisions. At one extreme, if **K** is stored as full matrix and slave freedom coupling in the MFCs is disallowed the logic is simple.[2] At the other extreme, if arbitrary couplings are permitted and **K** is placed in secondary (disk) storage according to some sparse scheme, the complexity can become overwhelming.

Another, more subtle, drawback of this method is that it requires decisions as to which degrees of freedom are to be treated as slaves. This can lead to implementation and numerical stability problems. Although for disjointed constraints the process can be programmmed in reliable form, in more general cases of coupled constraint equations it can lead to incorrect decisions. For example, suppose that in the example problem you have the following two MFCs:

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

For numerical stability reasons it is usually better to pick as slaves the freedoms with larger coefficients. If this is done, the program would select u_6 as slave freedoms from both constraints. This leads to a contradiction because having two constraints we must eliminate two slave degrees of freedom, not just one. The resulting modified system would in fact be inconsistent. Although this defect can be easily fixed by the program logic in this case, one can imagine the complexity burden if faced with hundreds or thousands of MFCs.

Serious numerical problems can arise if the MFCs are not independent. For example:

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

[2] This is the case in model reduction, since each slave freedom appears in one and only one MFC.

The last constraint is an exact linear combination of the first two. If the program blindly choses u_2, u_3 and u_7 as slaves, the modified system is incorrect because we eliminate three equations when in fact there are only two independent constraints.

Exact linear dependence, as in (9.38), can be recognized by a rank analysis of the $\mathbf{A}_s$ matrix defined in (9.29). In inexact floating-point arithmetic, however, such detection may fail.[3]

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

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

In conclusion, the master-slave method is useful when a few simple linear constraints are imposed by hand. As a general purpose technique for finite element analysis it suffers from complexity and lack of robustness. It is worth learning this method, however, because of its great importance of the congruential transformation technique in *model reduction* for static and dynamic problems.

[3] The safest technique to identify dependencies is to do a singular-value decomposition (SVD) of $\mathbf{A}_s$. This can be, however, prohibitively expensive if one is dealing with hundreds or thousands of constraints.

Homework Exercises for Chapter 9
MultiFreedom Constraints I

EXERCISE 9.1

[C+N:20] The example structure of Figure 9.1 has $E^{(e)}A^{(e)}/L^{(e)} = 100$ for each element $e = 1, \ldots, 6$. Consequently $K_{11} = K_{77} = 100$, $K_{22} = \ldots = K_{66} = 200$, $K_{12} = K_{23} = \ldots = K_{67} = -100$. The applied node forces are taken to be $f_1 = 1$, $f_2 = 2$, $f_3 = 3$, $f_4 = 4$, $f_5 = 5$, $f_6 = 6$ and $f_7 = 7$, which are easy to remember. The structure is subjected to one support condition: $u_1 = 0$ (a fixed left end), and to one MFC: $u_2 - u_6 = 1/5$.

Solve this problem using the master-slave method to process the MFC, taking u_6 as slave. Upon forming the modified system (9.30) apply the left-end support $u_1 = 0$ using the placeholder method of §3.4. Solve the equations and verify that the displacement solution and the recovered node forces including reactions are

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

Use *Mathematica* or *Matlab* to do the algebra is recommended. For example, the following *Mathematica* script solves this Exercise:

```
(* Exercise 9.1 - 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]];
```

EXERCISE 9.2

[C+N:25] As in the previous Exercise but applying the following three MFCs, two of which are non-homogeneous:

$$u_2 - u_6 = 1/5, \qquad u_3 + 2u_4 = -2/3, \qquad 2u_3 - u_4 + u_5 = 0. \tag{E9.2}$$

Hints. Chose u_4, u_5 and u_6 as slaves. Much of the script shown for Exercise 9.1 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 saying

```
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]
```

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

$$\begin{aligned} \mathbf{u} &= [\,0. \quad 0.043072 \quad -0.075033 \quad -0.29582 \quad -0.14575 \quad -0.15693 \quad -0.086928\,]^T\,, \\ \mathbf{Ku} &= [\,-4.3072 \quad 16.118 \quad 10.268 \quad -37.085 \quad 16.124 \quad -8.1176 \quad 7.\,]^T\,. \end{aligned} \tag{E9.3}$$

EXERCISE 9.3

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

EXERCISE 9.4

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

EXERCISE 9.5

[A:25] Work out the general technique by which the unknowns need not be rearranged, that is, **u** and $\hat{\mathbf{u}}$ are the same. Use "placeholders" for the slave freedoms. (Hint: use ideas of §3.4).

EXERCISE 9.6

[A/C:35] Is it possible to establish a slave selection strategy that makes $\mathbf{A}_s$ diagonal or triangular? (This requires knowledge of matrix techniques such as pivoting.)

EXERCISE 9.7

[A/C:40] Work out a strategy that produces a well conditioned $\mathbf{A}_s$ by selecting new slaves as linear combinations of finite element freedoms if necessary. (Requires background in numerical analysis).

10

MultiFreedom Constraints II

TABLE OF CONTENTS

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

§10.1. THE PENALTY METHOD

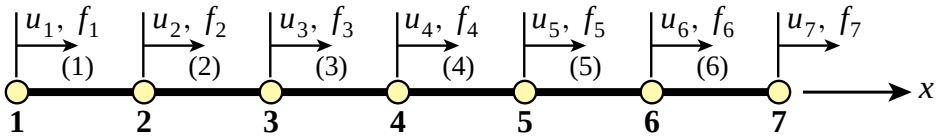


Figure 10.1. The example structure of Chapter 9, repeated for convenience.

§10.1.1. Physical Interpretation

The penalty method will be first presented using a physical interpretation, leaving the mathematical formulation to a subsequent section. Consider again the example structure of Chapter 9, which is reproduced in Figure 10.1 for convenience. To impose $u_2 = u_6$ imagine that nodes 2 and 6 are connected with a "fat" bar of axial stiffness w, labeled with element number 7, as shown in Figure 10.2. This bar is called a *penalty element* and w is its *penalty weight*.

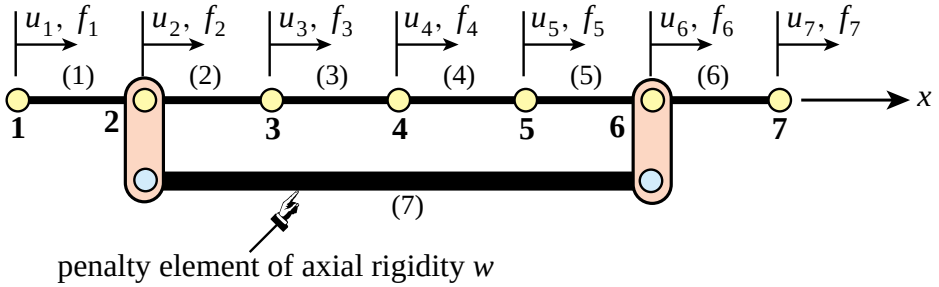


Figure 10.2. Adjunction of a fictitious penalty bar of axial stiffness w, identified as element 7,to enforce the MFC $u_2 = u_6$.

Such an element, albeit fictitious, can be treated exactly like another bar element insofar as continuing the assembly of the master stiffness equations. The penalty element stiffness equations, $\mathbf{K}^{(7)}\mathbf{u}^{(7)} = \mathbf{f}^{(7)}$, are[1]

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

[1] The general method to get these equations is described in §10.1.4.

Because there is one freedom per node, the two local element freedoms map into global freedoms 2 and 6, respectively. Using the assembly rules of Chapter 3 we obtain the following modified master stiffness equations: $\widehat{\mathbf{K}}\hat{\mathbf{u}} = \hat{\mathbf{f}}$, which shown in detail are

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

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

§10.1.2. Choosing the Penalty Weight

What happens when (10.2) is solved numerically? If a *finite* weight w is chosen the constraint $u_2 = u_6$ is approximately satisfied in the sense that one gets $u_2 - u_6 = e_g$, where $e_g \neq 0$. The "gap error" e_g is called the *constraint violation*. The magnitude $|e_g|$ of this violation depends on w: the larger w, the smaller the violation. More precisely, it can be shown that $|e_g|$ becomes proportional to $1/w$ as w gets to be sufficiently large (see Exercises). For example, raising w from, say, 10^6 to 10^7 can be expected to cut the constraint violation by roughly 10 if the physical stiffnesses are small compared to w.

Consequently, it seems as if the proper strategy should be: try to make w as large as possible while respecting computer overflow limits. However, this is misleading. As the penalty weight w tends to ∞ the modified stiffness matrix in (10.2) becomes more and more *ill-conditioned with respect to inversion*.

To make this point clear, suppose for definiteness that the rigidities $E^{(e)}A^{(e)}/L^{(e)}$ of the actual bars $e = 1, \ldots 6$ are unity, that $w >> 1$, and that the computer solving the stiffness equations has a floating-point precision of 16 decimal places. Numerical analysts characterize such precision by saying that $\epsilon_f = O(10^{-16})$, where $|\epsilon_f|$ is the smallest power of 10 that perceptibly adds to 1 in floating-point arithmetic.[2] The modified stiffness matrix of (10.2) becomes

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

Clearly as $w \to \infty$ rows 2 and 6, as well as columns 2 and 6, tend to become linearly dependent; in fact the negative of each other. Linear dependence means singularity; hence $\widehat{\mathbf{K}}$ approaches

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

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.

§10.1.3. The Square Root Rule

Obviously we have two effects at odds with each other. Making w larger reduces the constraint violation error but increases the solution error. The best w is that which makes both errors roughly equal in absolute value. This tradeoff value is difficult to find aside of systematically running numerical experiments. In practice the heuristic *square root rule* is often followed.

This rule can be stated as follows. Suppose that the largest stiffness coefficient, before adding penalty elements, is of the order of 10^k and that the working machine precision is p digits.[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 more 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 the choice of penalty weight by this rule involves knowledge of both stiffness magnitudes and floating-point hardware properties of the computer used as well as the precision selected by the program.

§10.1.4. Penalty Elements for General MFCs

For the constraint $u_2 = u_6$ the physical interpretation of the penalty element is clear. Points 2 and 6 must move in lockstep along x, which can be approximately enforced by the heavy bar device shown in Figure 10.2. But how about $3u_3 + u_5 - 4u_6 = 1$? Or just $u_2 = -u_6$?

The treatment of more general constraints is linked to the theory of *Courant penalty functions*, which in turn is a topic in variational calculus. Because the necessary theory has not yet been introduced (it is given in the next subsection), the procedure used for constructing a penalty element is stated here as a recipe. Consider the homogeneous constraint

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

Rewrite this equation in matrix form

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

[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.

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

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

Here $\bar{\mathbf{K}}^{(e)}$ is the *unscaled* stiffness matrix of the penalty element. This is now multiplied by the penalty weight w and assembled into the master stiffness matrix following the usual rules. For the example problem, augmenting (10.1) with the scaled penalty element (10.6) yields

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

If the constraint is nonhomogeneous the force vector is also modified. To illustrate this effect, consider the MFC: $3u_3 + u_5 - 4u_6 = 1$. Rewrite in matrix form as

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

Premultiply both sides by the transpose of the coefficient matrix:

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

Scaling by w and assembling yields

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

§10.1.5. *The Theory Behind the Recipe

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

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

where $\mathbf{u}$ contains all degrees of freedom and each $\mathbf{a}_p$ is a row vector with same length as $\mathbf{u}$. To incorporate the MFCs into the FEM model one selects a weight $w_p > 0$ for each constraints and constructs the so-called Courant quadratic penalty function or "penalty energy"

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

where we have called $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and $\mathbf{f}^{(p)} = w_p \mathbf{a}^T b_i$. P is added to the potential energy function $\Pi = \frac{1}{2}\mathbf{u}^T \mathbf{K}\mathbf{u} - \mathbf{u}^T \mathbf{f}$ to form the *augmented potential energy* $\Pi_a = \Pi + P$. Minimization of Π_a with respect to $\mathbf{u}$ yields

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

Each term of the sum on p, which derives from term P_p in (10.12), may be viewed as contributed by a penalty element with globalized stiffness matrix $\mathbf{K}^{(p)} = w_p \mathbf{a}_p^T \mathbf{a}_p$ and globalized added force term $\mathbf{f}^{(p)} = w_p \mathbf{a}_p^T b_p$.

To use a even more compact form we may write the set of multifreedom constraints as $\mathbf{A}\mathbf{u} = \mathbf{b}$. Then the penalty augmented system can be written compactly as

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

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

§10.1.6. Assessment of the Penalty Method

The main advantage of the penalty function method is its straightforward computer implementation. Looking at modifified systems such as (10.7) and (10.10) it is obvious that the equations need not be re-arranged. That is, $\mathbf{u}$ and $\hat{\mathbf{u}}$ are the same. Constraints may be programmed as "penalty elements," and stiffness and force contributions of these elements implemented through the standard assembler. In fact using this method there is no need to distinguish between unconstrained and constrained equations! Once all elements — regular and penalty — are assembled, the system (upon processsing for single-freedom constraints if necessary) can be passed to the equation solver.

An important advantage with respect to the master-slave (elimination) method is its lack of sensitivity with respect to whether constraints are linearly dependent. To give a simplistic example, suppose that the constraint $u_2 = u_6$ appears twice. Then two penalty bar elements connecting 2 and 6 will be inserted, doubling the intended weight but otherwise not causing undue harm.

An advantage with respect to the Lagrange multiplier method described in §10.2 is that positive definiteness is not lost. Such loss can affect the performance of certain numerical processes. Finally, it is worth noting that the the penalty method is easily extendible to nonlinear constraints although such extension falls outside the scope of this book.

The main disadvantage, however, is a serious one: the choice of weight values that balance solution accuracy with the violation of constraint conditions. For simple cases the square root rule mentioned previously often works, although 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

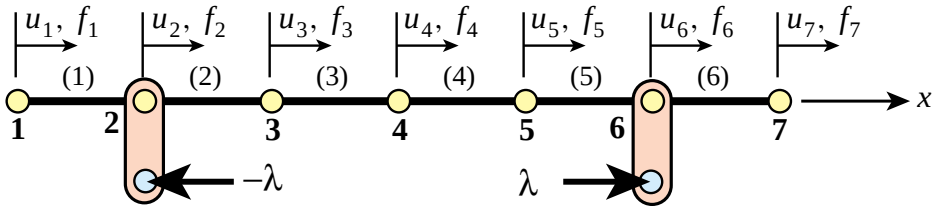


Figure 10.3. Physical interpretation of Lagrange multiplier adjunction to enforce the MFC $u_2 = u_6$.

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 a serious side effects in some solution procedures such as eigenvalue extraction or iterative solvers. Finally, even if optimal weights are selected, the combined solution error cannot be lowered beyond a treshold value.

From these comments it is evident that penalty augmentation, although superior to the master-slave method from the standpoint of generality and ease of implementation, is no panacea.

§10.2. LAGRANGE MULTIPLIER ADJUNCTION

§10.2.1. Physical Interpretation

As in the case of the penalty function method, the method of Lagrange multipliers can be given a rigorous justification within the framework of variational calculus. But in the same spirit it will be introduced for the example structure from a physical standpoint that is particularly illuminating.

Consider again the constraint $u_2 = u_6$. Borrowing some ideas from the penalty method, imagine that nodes 2 and 6 are connected now by a *rigid* link rather than a flexible one. Thus the constraint is imposed exactly. But of course the penalty method with an infinite weight would "blow up."

We may remove the link if it is replaced by an appropriate reaction force pair $(-\lambda, +\lambda)$, as illustrated in Figure 10.3. These are called the *constraint forces*. Incorporating these forces into the original stiffness equations (9.10) we get

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

This λ is called a *Lagrange multiplier*. Because λ is an unknown, let us transfer it to the *left hand*

side by *appending* it to the vector of unknowns:

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

But now we have 7 equations in 8 unknowns. To render the system determinate, the constraint condition $u_2 - u_6 = 0$ is appended as eighth equation:

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

This is called the *multiplier-augmented* system. Its coefficient matrix, which is symmetric, is called the *bordered stiffness matrix*. The process by which λ is appended to the vector of original unknowns is called *adjunction*. Solving this system provides the desired solution for the degrees of freedom while also characterizing the constraint forces through λ.

§10.2.2. Lagrange Multipliers for General MFCs

The general procedure will be stated first as a recipe. Suppose that we want to solve the example structure subjected to three MFCs

$$u_2 - u_6 = 0, \qquad 5u_2 - 8u_7 = 3, \qquad 3u_3 + u_5 - 4u_6 = 1, \tag{10.18}$$

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

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

Three Lagrange multipliers: λ_1, λ_2 and λ_3, are required to take care of three MFCs. Adjoin those unknowns to the nodal displacement vector. Symmetrize the coefficient matrix by adjoining 3 columns that are the transpose of the 3 last rows, and filling the bottom right-hand corner with zeros:

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

§10.2.3. *The Theory Behind the Recipe

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

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

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

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

The master stiffness matrix $\mathbf{K}$ in (10.22) is said to be *bordered* with $\mathbf{A}$ and $\mathbf{A}^T$. Solving this system provides $\mathbf{u}$ and $\boldsymbol{\lambda}$. The latter can be interpreted as forces of constraint in the following sense: a removed constraint can be replaced by a system of forces characterized by $\boldsymbol{\lambda}$ multiplied by the constraint coefficients. More precisely, the constraint forces are $-\mathbf{A}^T\boldsymbol{\lambda}$.

§10.2.4. Assessment of the Lagrange Multiplier Method

In contrast to the penalty method, the method of Lagrange multipliers has the advantage of being exact (aside from computation errors). It provides directly the constraint forces, which are of interest in many applications. It does not require any guesses as regards weights. As the penalty method, it can be extended without difficulty to nonlinear constraints.

It is not free of disadvantages. It introduces additional unknowns, requiring expansion of the original stiffness method. It renders the augmented stiffness matrix indefinite, a property that may cause grief with some linear equation solving methods such as Cholesky factorization or conjugate gradients. Finally, as the master-slave method, it is sensitive to the degree of linear independence of the constraints: if the constraint $u_2 = u_6$ is specified twice, the bordered stiffness is obviously singular.

On the whole the Lagrangian multiplier method appear to be the most elegant for a general-purpose finite element program that is supposed to work as a "black box" by minimizing guesses and choices from its users. Its implementation, however, is not simple. Special care must be exercised to detect singularities due to constraint dependency and to account for the effect of loss of positive definiteness of the bordered stiffness on equation solvers.

§10.3. *THE AUGMENTED LAGRANGIAN METHOD

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

Because the lower diagonal block of the bordered stiffness matrix in (10.22) is null, it is not possible to directly eliminate $\boldsymbol{\lambda}$. To allow that, replace this block by $\epsilon\mathbf{S}^{-1}$, where $\mathbf{S}$ is a constraint-scaling diagonal matrix of appropriate order and ϵ is a small number. The reciprocal of ϵ is a large number called $w = 1/\epsilon$. To maintain exactness of the second equation, $\epsilon\mathbf{S}^{-1}\boldsymbol{\lambda}$ is added to the right-hand side:

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

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

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

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

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

On taking $\mathbf{W} = w\mathbf{S}$, the general matrix equation (10.14) of the penalty method is recovered.

This relation suggests the construction of *iterative procedures* in which one tries to *improve the accuracy of the penalty function method while w is kept constant*.[5] This strategy circumvents the aforementioned ill-conditioning problems when the weight w is gradually increased. One such method is easily constructed by inspecting (10.24). Using superscript k as an iteration index and keeping w fixed, solve equations (10.24) in tandem as follows:

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

for $k = 0, 1, \ldots$, beginning with $\boldsymbol{\lambda}^0 = \mathbf{0}$.[6] Then $\mathbf{u}^0$ is the penalty solution. If the process converges one recovers the exact Lagrangian solution without having to solve the Lagrangian system (10.23) directly.

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

---

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

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

	Master-Slave Elimination	Penalty Function	Lagrange Multiplier
Generality	**fair**	**excellent**	**excellent**
Ease of implementation	**poor to fair**	**good**	**fair**
Sensitivity to user decisions	**high**	**high**	**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 10.4. Assessment summary of MFC application methods.

§10.4. SUMMARY

The treatment of linear MFCs in finite element systems can be carried out by several methods. Three of these: the master-slave elimination, penalty augmentation and Lagrange multiplier adjunction, have been discussed. It is emphasized that no method is uniformly satisfactory in terms of generality, numerical behavior and simplicity of implementation. See Figure 10.4 for a summary.

For a general purpose program that tries to approach "black box" behavior (that is, minimal decisions on the part of users) the method of Lagrange multipliers has the edge. This edge is unfortunately blunted by a fairly complex computer implementation and by the loss of positive definiteness in the bordered stiffness matrix.

Homework Exercises for Chapter 10
MultiFreedom Constraints II

EXERCISE 10.1

[C+N:20] This is identical to Exercise 9.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the penalty function method. Take the weight w to be 10^k, in which k varies as $k = 3, 4, 5, \ldots 16$. 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 (E9.1). 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 (§10.1.3) if the computations carry 16 digits of precision?

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

```
(* Exercise 10.1 - Penalty Method  *)
(* MFC: u2-u6=1/5  variable w *)
K=MasterStiffnessOfSixElementBar[100];
Print["Stiffness K=",K//MatrixForm];
f={1,2,3,4,5,6,7}; Print["Applied forces=",f];
uexact= {0,0.27,0.275,0.25,0.185,0.07,0.14}; ew={};
For [w=100, w<=10^16, w=10*w; (* increase w by 10 every pass *)
     Khat=K; fhat=f;
     Khat[[2,2]]+=w; Khat[[6,6]]+=w; Khat[[6,2]]=Khat[[2,6]]-=w;
     fhat[[2]]+=(1/5)*w; fhat[[6]]-=(1/5)*w; (*insert penalty *)
     {Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat];
     u=LinearSolve[N[Kmod],N[fmod]];
     Print["Weight w=",N[w]//ScientificForm," u=",u//InputForm];
     e=Sqrt[(u-uexact).(u-uexact)];
     (*Print["L2 solution error=",e//ScientificForm]; *)
     AppendTo[ew,{Log[10,w],Log[10,e]}];
   ];
ListPlot[ew,AxesOrigin->{5,-8},Frame->True, PlotStyle->
  {AbsolutePointSize[4],AbsoluteThickness[2],RGBColor[1,0,0]},
   PlotJoined->True,AxesLabel->{"Log10(w)","Log10(u error)"}];
```

Here `MasterStiffnessOfSixElementBar` and `FixLeftEndOfSixElementBar` are the same modules listed in Exercise 9.1.

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

EXERCISE 10.2

[C+N:15] Again identical to Exercise 9.1, except that the MFC $u_2 - u_6 = 1/5$ is to be treated by the Lagrange multiplier method. The results for the computed **u** and the recovered force vector **Ku** should agree with (E9.1). Use *Mathematica*, *Matlab* (or similar) to do the algebra. For example, the following *Mathematica* script solves this Exercise:

```
(* Exercise 10.2 - Lagrange Multiplier Method  *)
(* MFC: u2-u6=1/5  *)
K=MasterStiffnessOfSixElementBar[100];
```

```
Khat=Table[0,{8},{8}]; f={1,2,3,4,5,6,7}; fhat=AppendTo[f,0];
For [i=1,i<=7,i++, For[j=1,j<=7,j++, Khat[[i,j]]=K[[i,j]] ]];
{Kmod,fmod}=FixLeftEndOfSixElementBar[Khat,fhat];
Kmod[[2,8]]=Kmod[[8,2]]= 1;
Kmod[[6,8]]=Kmod[[8,6]]=-1; fmod[[8]]=1/5;
Print["Kmod=",Kmod//MatrixForm];
Print["fmod=",fmod];
umod=LinearSolve[N[Kmod],N[fmod]]; u=Take[umod,7];
Print["Solution u=",u ,",   lambda=",umod[[8]]];
Print["Recovered node forces=",K.u];
```

Here `MasterStiffnessOfSixElementBar` and `FixLeftEndOfSixElementBar` are the same modules listed in Exercise 9.1.

Does the computed solution agree with (E9.1)?

EXERCISE 10.3

[A:10] For the example structure, show which penalty elements would implement the following MFCs:

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

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

EXERCISE 10.4

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

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

This system is to be solved subject to the multipoint constraint

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

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

(b) Impose the constraint (E10.3) by the penalty function method, leaving the weight w as a free parameter. Solve the equations by hand or CAS (Cramer's rule is recommended) and verify analytically that as $w \to \infty$ the solution approaches that found in (a). Tabulate the values of u_1, u_2, u_3 for $w = 0,\ 1,\ 10,\ 100$. *Hint 1*: the value of u_2 should not change. *Hint 2*: the solution for u_1 should be $(6w + 5)/(4w + 4)$.

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

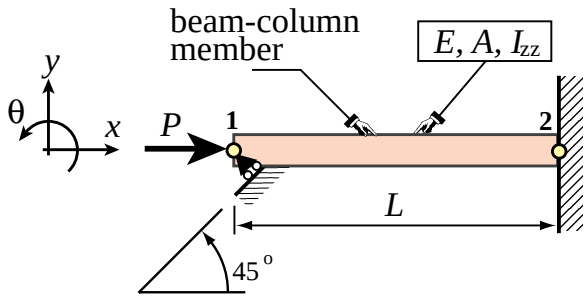


Figure E10.1. Beam-colum member for Exercise 10.5.

EXERCISE 10.5

[A/C:10+15+10] The beam-column member shown in Figure E10.1 rests on a skew roller that forms a 45° angle with the horizontal axis x, and is loaded axially by a force P.

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

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

where E, A, and I_{zz} are member properties, θ_1 is the left end rotation, and L is the member length.[7] To simplify the calculations set $P = \alpha EA$, and $I_{zz} = \beta AL^2$, 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 by the penalty method by adjoining a penalty truss member of axial stiffness $k = wEA$ normal to the roller, and compute u_{x1} (Cramer's rule is recommended if solved by hand). Verify that as $w \to \infty$ the answer obtained in (a) is recovered. Partial solution: $u_{x1} = \alpha L(3\beta + wL)/(3\beta + wL(1+3\beta))$.

(c) Apply the skew roller constraint by Lagrangian multiplier adjunction, and solve the resulting 4×4 system of equations using a CAS. Verify that you get the same solution as in (a).

EXERCISE 10.6

[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

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

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

EXERCISE 10.7

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

---

[7] The stiffness equations for a beam column are derived in Part III of this book. For now consider (E10.4) as a recipe.

EXERCISE 10.8

[A:40] (Advanced, close to a research problem) Show that the master-slave transformation method $\mathbf{u} = \mathbf{T}\hat{\mathbf{u}}$ can be expressed as a limit of the penalty function method as the weights go to infinity. Start from the augmented functional

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

Write down the matrix stationarity conditions with respect to to $\mathbf{u}$ $\hat{\mathbf{u}}$ and eliminate $\mathbf{u}$. *Hint*: using Woodbury's formula (Appendix C, §C.5.2)

$$(\mathbf{K} + w\mathbf{T}^T\mathbf{S}\mathbf{T})^{-1} = \mathbf{K}^{-1} - \mathbf{K}^{-1}\mathbf{T}^T(\overline{\mathbf{K}} + w^{-1}\mathbf{S}^{-1})^{-1}\mathbf{T}\,\mathbf{K}^{-1}. \tag{E10.7}$$

show that

$$\overline{\mathbf{K}} = \mathbf{T}\mathbf{K}^{-1}\mathbf{T}^T \tag{E10.8}$$

11

Superelements and Global-Local Analysis

TABLE OF CONTENTS

§11.1. SUPERELEMENT CONCEPT

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

A random assortment of elements does not necessarily make up a superelement. To be considered as such, an element grouping must meet certain conditions. Informally we can say that the grouping must form a structural component on its own. This imposes certain conditions stated mathematically in §11.1.3. Inasmuch as these conditions involve advanced concepts such as rank sufficiency, which are introduced in later Chapters, the restrictions are not dwelled upon here.

As noted in Chapter 7, superelements may originate from two overlapping contexts: "bottom up" or "top down." In a bottom up context one thinks of superelements as built from simpler elements. In the top-down context, superelements may be thought as being large pieces of a complete structure. This dual viewpoint motivates the following classification:

Macroelements. These are superelements assembled with a few primitive elements. Also called *mesh units* when they are presented to program users as individual elements.

Substructures. Complex assemblies of elements that result on breaking up a structure into distinguishable portions.

When does a substructure becomes a macroelement or vice-versa? There are no precise rules. In fact the generic term *superelement* was coined in the 1970s to take up the entire spectrum, ranging from *individual elements* to *complete structures*. This universality is helped by the common features noted below.

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

§11.1.1. Where Does the Idea Comes From?

Substructuring was invented by aerospace engineers in the early 1960s[1] to carry out a first-level breakdown of complex systems such as a complete airplane, as depicted in Figures 11.1 and 11.2. The decomposition may continue hierarchically through additional levels as illustrated in Figure 11.3. The concept is also natural for space vehicles operating in stages, such as the Apollo short stack depicted in Figure 11.4.

One obvious advantage of this idea results if the structure is built of several identical units. For example, the wing substructures S_2 and S_3 are largely identical except for a reflection about the fuselage midplane, and so are the stabilizers S_4 and S_5. Even if the loading is not symmetric, taking account of the structural symmetry reduces mesh preparation time.

---

[1] For a bibliography of early work, see J. S. Przeminiecki, *Theory of Matrix Structural Analysis*, McGraw-Hill, New York, 1968 (also in Dover ed).

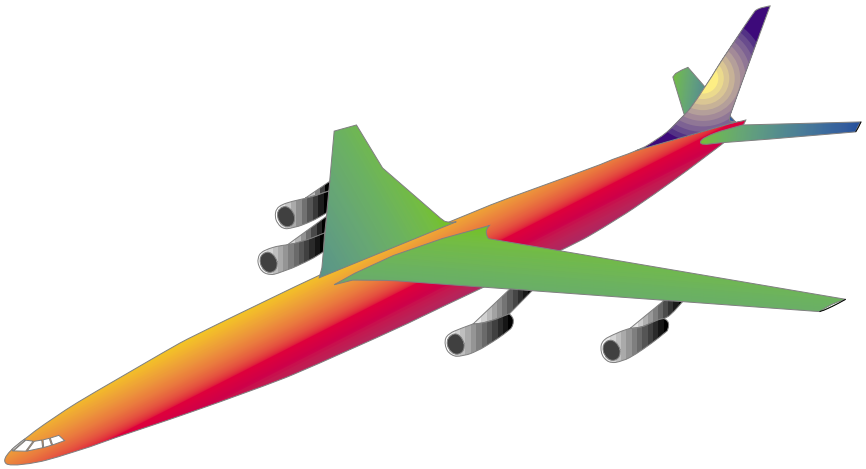


Figure 11.1. A complete airplane.

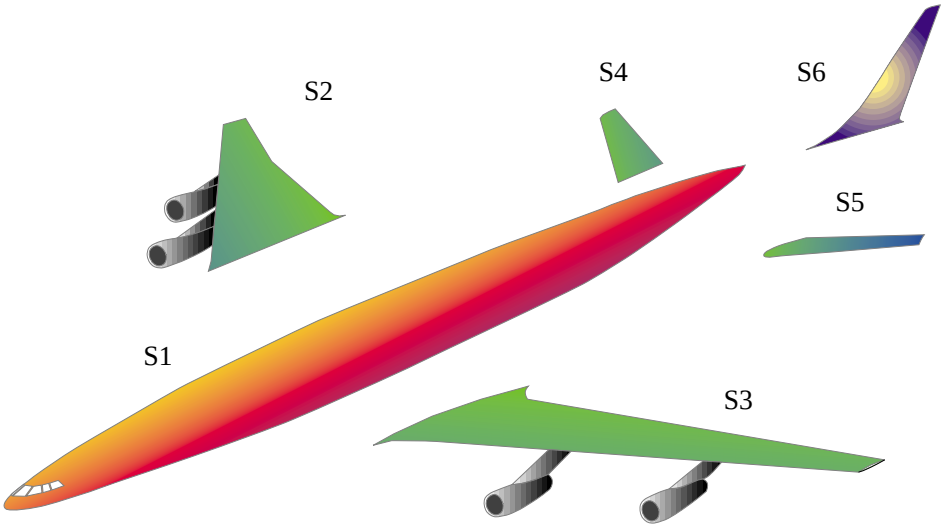


Figure 11.2. Airplane broken down into six level one substructures identified as S_1 through S_6.

The concept was then picked up and developed extesively by the offshore and shipbuilding industries, the products of which tend to be very modular and repetitive to reduce fabrication costs. As noted above, repetition of structural components favors the use of substructuring techniques.

At the other extreme, mesh units appeared in the early days of finite element methods. They were motivated by user convenience. For example, in hand preparation of finite element models, quadrilateral and bricks involve less human labor than triangles and tetrahedra, respectively. It was therefore natural to combine the latter to assemble the former.

§11.1.2. Subdomains

Applied mathematicians working on solution procedures for parallel computation have developed the concept of *subdomains*. These are groupings of finite elements that are entirely motivated by computational considerations. They are subdivisions of the finite element model done more or less

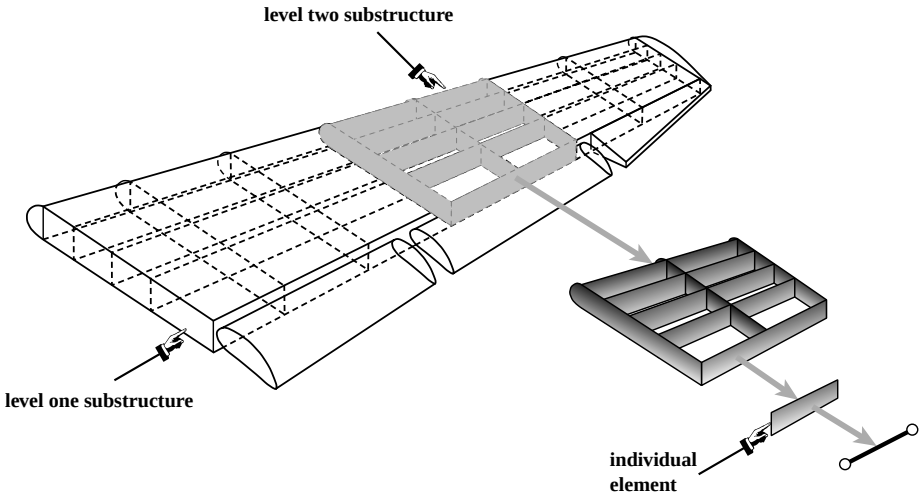


Figure 11.3. Further breakdown of wing structure. The decomposition process may continue down to the individual element level.

automatically by a program called *domain decomposer*.

Although the concepts of substructures and subdomains overlap in many respects, it is better to keep the two separate. The common underlying theme is divide and conquer but the motivation is different.

§11.1.3. *Mathematical Requirements

A superelement is said to be *rank-sufficient* if its only zero-energy modes are rigid-body modes. Equivalently, the superelement does not possess spurious kinematic mechanisms.

Verification of the rank-sufficient condition guarantees that the static condensation procedure described below will work properly.

§11.2. STATIC CONDENSATION

Degrees of freedom of a superelement are classified into two groups:

Internal Freedoms. Those that are not connected to the freedoms of another superelement. Node whose freedoms are internal are called *internal nodes*.

Boundary Freedoms. These are connected to at least another superelement. They usually reside at *boundary nodes* placed on the periphery of the superelement. See Figure 11.5.

The objective is to get rid of all displacement degrees of freedom associated with *internal freedoms*. This elimination process is called *static condensation*, or simply *condensation*.

Condensation may be presented in terms of explicit matrix operations, as shown in the next subsection. A more practical technique based on symmetric Gauss elimination is discussed later.

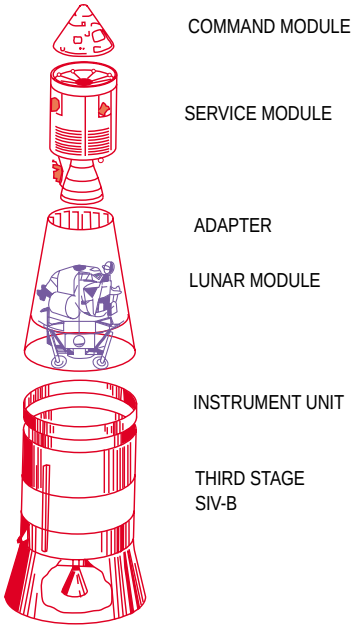


Figure 11.4. The Apollo short stack.

§11.2.1. Condensation by Explicit Matrix Operations

To carry out the condensation process, the assembled stiffness equations of the superelement are partitioned as follows:

$$\begin{bmatrix} \mathbf{K}_{bb} & \mathbf{K}_{bi} \\ \mathbf{K}_{ib} & \mathbf{K}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{f}_b \\ \mathbf{f}_i \end{bmatrix}. \tag{11.1}$$

where subvectors $\mathbf{u}_b$ and $\mathbf{u}_i$ collect *boundary* and *interior* degrees of freedom, respectively. Take the second matrix equation:

$$\mathbf{K}_{ib}\mathbf{u}_b + \mathbf{K}_{ii}\mathbf{u}_i = \mathbf{f}_i, \tag{11.2}$$

If $\mathbf{K}_{ii}$ is nonsingular we can solve for the interior freedoms:

$$\mathbf{u}_i = \mathbf{K}_{ii}^{-1}(\mathbf{f}_i - \mathbf{K}_{ib}\mathbf{u}_b), \tag{11.3}$$

Replacing into the first matrix equation of (11.2) yields the *condensed stiffness equations*

$$\tilde{\mathbf{K}}_{bb}\mathbf{u}_b = \tilde{\mathbf{f}}_b. \tag{11.4}$$

In this equation,

$$\tilde{\mathbf{K}}_{bb} = \mathbf{K}_{bb} - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{K}_{ib}, \qquad \tilde{\mathbf{f}}_b = \mathbf{f}_b - \mathbf{K}_{bi}\mathbf{K}_{ii}^{-1}\mathbf{f}_i, \tag{11.5}$$

are called the *condensed* stiffness matrix and force vector, respectively, of the substructure.

From this point onward, the condensed superelement may be viewed, from the standpoint of further operations, as an *individual element* whose element stiffness matrix and nodal force vector are $\tilde{\mathbf{K}}_{bb}$ and $\tilde{\mathbf{f}}_b$, respectively.

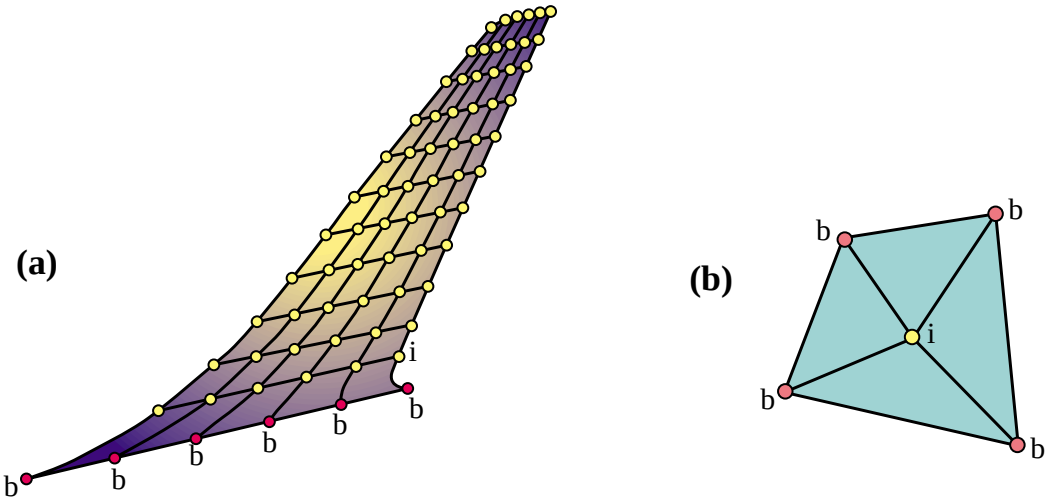


Figure 11.5. Classification of superelement freedoms into boundary and internal. (a) shows the vertical stabilizer substructure S_6 of Figure 11.2. (The FE mesh is depicted as two-dimensional for illustrative purposes; for an actual aircraft it will be three dimensional with an internal structure similar to the one displayed in Figure 11.3.) Boundary freedoms are those associated to the boundary nodes labeled *b* (shown in red), which are connected to the fuselage substructure. (b) shows a quadrilateral macroelement mesh-unit fabricated with 4 triangles: it has one interior and four boundary nodes.

REMARK 11.1

The feasibility of the condensation process (11.5) hinges on the non-singularity of $\mathbf{K}_{ii}$. This matrix is nonsingular if the superelement is rank-sufficient in the sense stated in §11.1.3, and if fixing the boundary freedoms precludes all rigid body motions. If the former condition is verified but not the latter, the superelement is called *floating*. Processing floating superelements demands more advanced computational techniques, among which we cite the concepts of projectors and generalized inverses.

§11.2.2. Condensation by Symmetric Gauss Elimination

In the computer implementation of the the static condensation process, calculations are not carried out as outlined above. There are two major differences: the equations of the substructure are not actually rearranged, and the explicit calculation of the inverse of $\mathbf{K}_{ii}$ is avoided. The procedure is in fact coded as a variant of symmetric Gauss elimination. To convey the flavor of this technique, consider the following stiffness equations of a superelement:

$$\begin{bmatrix} 6 & -2 & -1 & -3 \\ -2 & 5 & -2 & -1 \\ -1 & -2 & 7 & -4 \\ -3 & -1 & -4 & 8 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \\ 0 \end{bmatrix}. \tag{11.6}$$

Suppose that the last two displacement freedoms: u_3 and u_4, are classified as interior and are to be statically condensed out. To eliminate u_4, perform symmetric Gauss elimination of the fourth row

Cell 11.1 Program to Condense Out the Last Freedom from Ku = f

```
CondenseLastFreedom[K_,f_]:=Module[{n=Length[K],c,pivot,Kc,fc},
   If [n<=0,Return[{K,f}]];
   Kc=Table[0,{n-1},{n-1}]; fc=Table[0,{n-1},{1}];
   pivot=K[[n,n]]; If [pivot==0,Print["Singular Matrix"]; Return[{K,f}]];
   Do[ c=K[[i,n]]/pivot; fc[[i,1]]=f[[i,1]]-c*f[[n,1]];
      Do[  Kc[[j,i]]=Kc[[i,j]]=K[[i,j]]-c*K[[n,j]],
     {j,1,i}],
   {i,1,n-1}];
   Return[{Kc,fc}]
];

K={{6,-2,-1,-3},{ -2,5,-2,-1},{ -1,-2,7,-4},{-3,-1,-4,8}};
f={{3},{6},{4},{0}}; Print["K=",K, "  f=",f];
{K,f}=CondenseLastFreedom[K,f]; Print["Upon condensing freedom 4: ",K,f];
{K,f}=CondenseLastFreedom[K,f]; Print["Upon condensing freedom 3: ",K,f];
```

and column:

$$\begin{bmatrix} 6-\frac{(-3)\times(-3)}{8} & -2-\frac{(-1)\times(-3)}{8} & -1-\frac{(-4)\times(-3)}{8} \\ -2-\frac{(-3)\times(-1)}{8} & 5-\frac{(-1)\times(-1)}{8} & -2-\frac{(-4)\times(-1)}{8} \\ -1-\frac{(-3)\times(-4)}{8} & -2-\frac{(-1)\times(-4)}{8} & 7-\frac{(-4)\times(-4)}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3-\frac{0\times(-3)}{8} \\ 6-\frac{0\times(-1)}{8} \\ 4-\frac{0\times(-4)}{8} \end{bmatrix}, \tag{11.7}$$

or

$$\begin{bmatrix} \frac{39}{8} & -\frac{19}{8} & -\frac{5}{2} \\ -\frac{19}{8} & \frac{39}{8} & -\frac{5}{2} \\ -\frac{5}{2} & -\frac{5}{2} & 5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 6 \\ 4 \end{bmatrix}. \tag{11.8}$$

Repeat the process for the third row and column to eliminate u_3:

$$\begin{bmatrix} \frac{39}{8}-\frac{(-5/2)\times(-5/2)}{5} & -\frac{19}{8}-\frac{(-5/2)\times(-5/2)}{5} \\ -\frac{19}{8}-\frac{(-5/2)\times(-5/2)}{5} & \frac{39}{8}-\frac{(-5/2)\times(-5/2)}{5} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 3-\frac{4\times(-5/2)}{5} \\ 6-\frac{4\times(-5/2)}{5} \end{bmatrix}, \tag{11.9}$$

or

$$\begin{bmatrix} \frac{29}{8} & -\frac{29}{8} \\ -\frac{29}{8} & \frac{29}{8} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 5 \\ 8 \end{bmatrix}. \tag{11.10}$$

These are the condensed stiffness equations. Cell 11.1 shows a *Mathematica* program that executes the foregoing steps.

Obviously this procedure is much simpler than going through the explicit matrix inverse. Another important advantage of Gauss elimination is that equation rearrangement is not required even if the condensed degrees of freedom do not appear in any particular order. For example, suppose that the assembled substructure contains originally eight degrees of freedom and that the freedoms to

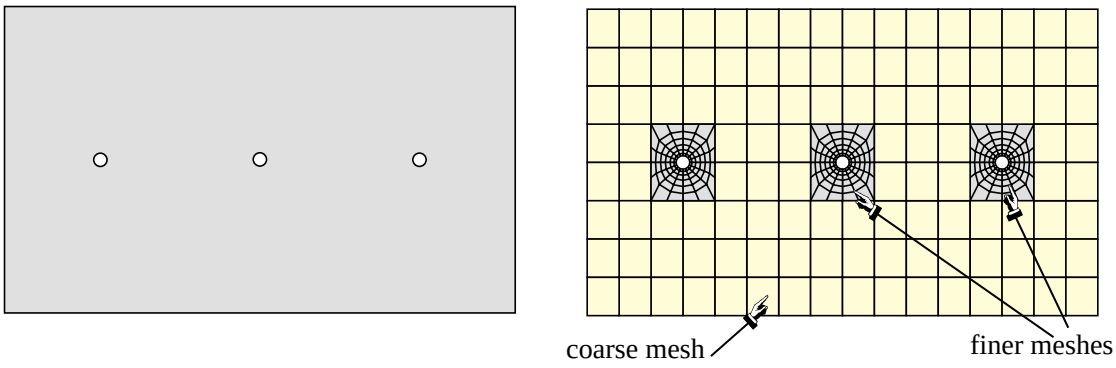


Figure 11.6. Left: example panel structure for global-local analysis.
Right: a FEM mesh for a one-shot analysis.

be condensed out are numbered 1, 4, 5, 6 and 8. Then Gauss elimination is carried out over those equations only, and the condensed (3×3) stiffness and (3×1) force vector extracted from rows and columns 2, 3 and 7.

REMARK 11.2

The symmetric Gauss elimination procedure, as illustrated in steps (11.7) through (11.10), is primarily useful for macroelements and mesh units, since the number of stiffness equations for those typically does not exceed a few hundreds. This permits the use of full matrix storage. For substructures containing thousands or millions of degrees of freedom — such as in the airplane example — the elimination is carried out using more sophisticated sparse matrix algorithms.

REMARK 11.3

The static condensation process is a matrix operation called "partial inversion" or "partial elimination" that appears in many disciplines. Here is the general form. Suppose the linear system $\mathbf{A}\mathbf{x} = \mathbf{y}$, where $\mathbf{A}$ is $n \times n$ square and $\mathbf{x}$ and $\mathbf{y}$ are n-vectors, is partitioned as

$$\begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix}. \tag{11.11}$$

Assuming the appropriate inverses to exist, then the following are easily verified matrix identities:

$$\begin{bmatrix} \mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \\ \mathbf{A}_{21}\mathbf{A}_{11}^{-1} & \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{12}\mathbf{A}_{22}^{-1} \\ -\mathbf{A}_{22}^{-1}\mathbf{A}_{21} & \mathbf{A}_{22}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{x}_2 \end{bmatrix}. \tag{11.12}$$

We say that $\mathbf{x}_1$ has been eliminated or "condensed out" in the left identity and $\mathbf{x}_2$ in the right one. In FEM applications, it is conventional to condense out the bottom vector $\mathbf{x}_2$, so the right identity is relevant. If $\mathbf{A}$ is symmetric, to retain symmetry in (11.12) it is necessary to change the sign of one of the subvectors.

§11.3. GLOBAL-LOCAL ANALYSIS

As noted in the first Chapter, complex engineering systems are often modeled in a *multilevel* fashion following the divide and conquer approach. The superelement technique is a practical realization of that approach.

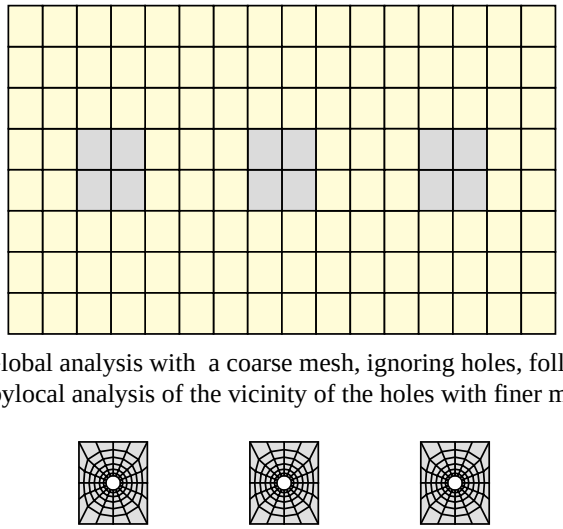


Figure 11.7. Global-local analysis of problem of Figure 11.5.

A related, but not identical, technique is *multiscale* analysis. The whole system is first analyzed as a global entity, discarding or passing over details deemed not to affect its overall behavior. Local details are then analyzed using the results of the global analysis as boundary conditions. The process can be continued into the analysis of further details of local models. And so on. When this procedure is restricted to two stages and applied in the context of finite element analysis, it is called *global-local* analysis in the FEM literature.

In the global stage the behavior of the entire structure is simulated with a finite element model that necessarily ignores details such as cutouts or joints. These details do not affect the overall behavior of the structure, but may have a bearing on safety. Such details are incorporated in a series of local analyses.

The gist of the global-local approach is explained in the example illustrated in Figures 11.6 and 11.7. Although the structure is admittedly two simple to merit the application of global-local analysis, it serves to illustrate the basic ideas. Suppose one is faced with the analysis of the rectangular panel shown on the top of Figure 11.6, which contains three small holes. The bottom of that figure shows a standard (one-stage) FEM treatment using a largely regular mesh that is refined near the holes. Connecting the coarse and fine meshes usually involves using multifreedom constraints because the nodes at mesh boundaries do not match, as depicted in that figure.

Figure 11.6 illustrates the global-local analysis procedure. The global analysis is done with a coarse but regular FEM mesh which *ignores the effect of the holes*. This is followed by local analysis of the region near the holes using refined finite element meshes. The key ingredient for the local analyses is the application of boundary conditions (BCs) on the finer mesh boundaries. These BCs may be of displacement (essential) or of force (natural) type. If the former, the applied boundary displacements are interpolated from the global mesh solution. If the latter, the internal forces or stresses obtained from the global calculation are converted to nodal forces on the fine meshes through a lumping process.

The BC choice noted above gives rise to two basic variations of the global-local approach. Expe-

rience accumulated over several decades[2] has shown that the stress-BC approach generally gives more reliable answers.

The global-local technique can be extended to more than two levels, in which case it receives the more encompassing name *multiscale analysis*. Although this generalization is still largely in the realm of research, it is receiving increasing attention from various science and engineering communities for complex products such as the thermomechanical analysis of microelectronic components.

[2] Particularly in the aerospace industry, in which the global-local technique has been used since the mid 1960s.

Homework Exercises for Chapter 11

Superelements and Global-Local Analysis

EXERCISE 11.1

[N:15] Suppose that the assembled stiffness equations for a one-dimensional superelement are

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}. \tag{E11.1}$$

Eliminate u_2 from the unconstrained stiffness equations (E10.1) by static condensation, and show (but do not solve) the condensed equation system. Use either the explicit inverse formulas (11.5) or the symmetric Gauss elimination process explained in §11.2.2. Hint: regardless of method the result should be

$$\begin{bmatrix} 1.5 & -0.5 \\ -0.5 & 1.5 \end{bmatrix} \begin{bmatrix} u_1 \\ u_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}, \tag{E11.2}$$

EXERCISE 11.2

[C+N:20] Generalize the program of **Cell 11.1** to a module `CondenseFreedom[K,f,i]` that is able to condense the `i`th degree of freedom from `K` and `f`, which is not necessarily the last one. That is, `i` may range from 1 to `n`, where `n` is the dimension of $\mathbf{Ku} = \mathbf{f}$. Apply that program to solve Exercise 11.1.

EXERCISE 11.3

[D:15] Explain the similarities and differences between superelement analysis and global-local FEM analysis.

EXERCISE 11.4

[A:25] Show that the static condensation process can be viewed as a master-slave transformation method (Chapter 9) in which the interior freedoms $\mathbf{u}_i$ are taken as slaves linked by the transformation relation

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_b \\ \mathbf{u}_i \end{bmatrix} = \begin{bmatrix} \mathbf{I} \\ -\mathbf{K}_{22}^{-1}\mathbf{K}_{21} \end{bmatrix} [\,\mathbf{u}_b\,] - \begin{bmatrix} \mathbf{0} \\ -\mathbf{K}_{22}^{-1}\mathbf{f}_i \end{bmatrix} = \mathbf{T}\mathbf{u}_b - \mathbf{g}. \tag{E11.3}$$

Hint: apply (9.26) in which $\hat{\mathbf{u}} \equiv \mathbf{u}_b$ are the masters and compare the result to (11.6)-(11.7).

EXERCISE 11.5

[D:30] (Requires thinking) Explain the conceptual and operational differences between standard (one-stage) FEM analysis and global-local analysis of a problem such as that illustrated in Figures 11.5-6. Are the answers the same? What is gained, if any, by the global-local approach over the one-stage FEM analysis?

EXERCISE 11.6

[A:20] Two beam elements: 1–2 and 2–3, each of length L and rigidity EI are connected at node 2. Eliminate node 2 by condensation. Does the condensed stiffness equals the stiffness of a beam element of length $2L$ and rigidity EI?

(For expressions of the beam stiffness matrices, see Chapter 13.)

12

Variational Formulation of Bar Element

TABLE OF CONTENTS

§12.1. A NEW BEGINNING

This Chapter begins Part II of the course. This Part focuses on the construction of structural and continuum finite elements using a *variational formulation* based on the Total Potential Energy.

Why only elements? Because the other synthesis steps of the DSM: assembly and solution, remain the same as in Part I. These operations are not element dependent.

Part II constructs *individual elements* beginning with the simplest ones and progressing to more complicated ones. The formulation of two-dimensional finite elements from a variational standpoint is discussed in Chapters 14 and following. Although the scope of that formulation is broad, exceeding structural mechanics, it is better understood by going through specific elements first.

The simplest finite elements from a geometrical standpoint are one-dimensional or *line elements*. The term means that the *intrinsic dimensionality* is one, although they may be used in one, two or three space dimensions upon transformation to global coordinates as appropriate. The simplest one-dimensional structural element is the *two-node bar element*, which we have already encountered in Chapters 2-3 as the truss member.

In this Chapter the stiffness equations of that bar element are rederived using the variational formulation. For uniform properties the resulting equations are the same as those found previously using the physical or Mechanics of Materials approach. The variational method has the advantage of being readily extendible to more complicated situations, such as variable cross section or more than two nodes.

§12.2. DEFINITION OF BAR MEMBER

In structural mechanics a *bar* is a structural component characterized by two properties:

(1) One bar dimension: the *longitudinal dimension* or *axial dimension* is much larger that the other two dimensions, which are collectively known as *transverse dimensions*. The intersection of a plane normal to the longitudinal dimension and the bar defines the *cross sections*. The longitudinal dimension defines the *longitudinal axis*. See Figure 12.1.

(2) The bar resists an internal axial force along its longitudinal dimension.

In addition to trusses, bar elements are used to model cables, chains and ropes. They are also used as fictitious elements in penalty function methods, as discussed in Chapter 10.

We will consider here only *straight bars*, although their cross section may vary. The one-dimensional mathematical model assumes that the bar material is linearly elastic, obeying Hooke's law, and that displacements and strains are infinitesimal. Figure 12.2 pictures the relevant quantities for a fixed-free bar. Table 12.1 collects the necessary terminology for the governing equations.

Figure 12.3 displays the governing equations of the bar in a graphic format called a *Tonti diagram*. The formal similarity with the diagrams used in Chapter 6 to explain MoM elements should be noted, although the diagram of Figure 12.3 pertains to the continuum model rather than to the discrete one.

Table 12.1 Nomenclature for Mathematical Model of Axially Loaded Bar

Quantity	*Meaning*
x	Longitudinal bar axis*
$(.)'$	$d(.)/dx$
$u(x)$	Axial displacement
$q(x)$	Distributed axial force, given per unit of bar length
L	Total bar length
E	Elastic modulus
A	Cross section area; may vary with x
EA	Axial rigidity
$e = du/dx = u'$	Infinitesimal axial strain
$\sigma = Ee = Eu'$	Axial stress
$p = A\sigma = EAe = EAu'$	Internal axial force
P	Prescribed end load

* x is used in this Chapter instead of $\bar{x}$ (as in Chapters 2–3) to simplify the notation.

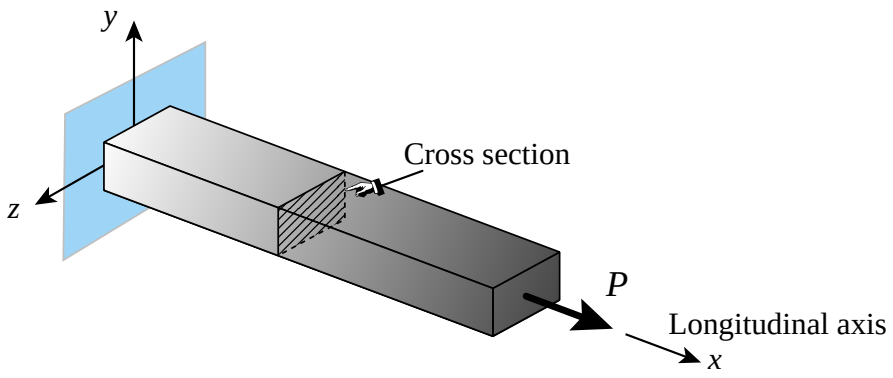


Figure 12.1. A fixed-free bar member.

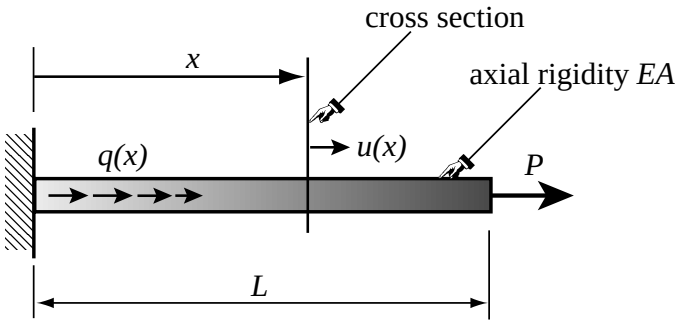


Figure 12.2. Quantities that appear in the mathematical model of a bar member.

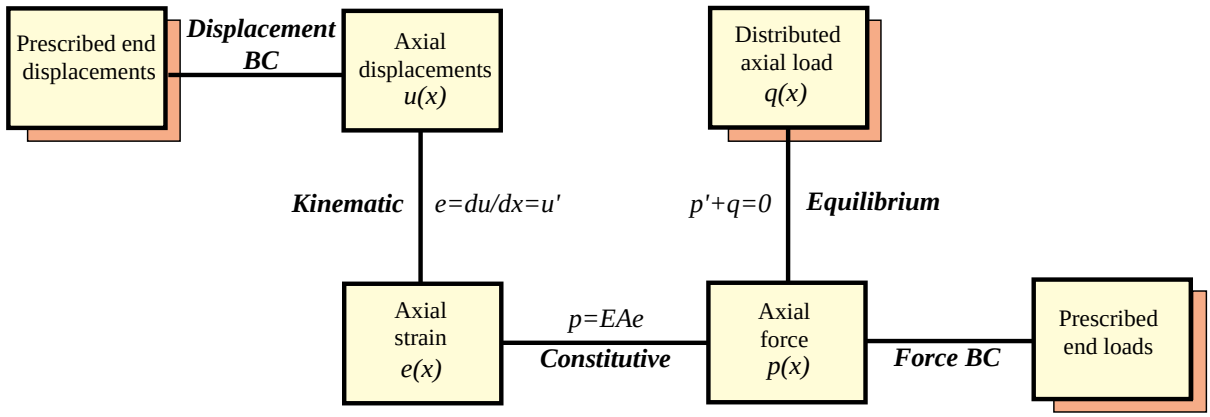


Figure 12.3. Tonti diagram for the mathematical model of a bar member. The diagram displays the field equations and boundary conditions as lines connecting the boxes. Boxes shown in relief contain known (given) quantities.

§12.3. VARIATIONAL FORMULATION

To illustrate the variational formulation, the finite element equations of the bar will be derived from the Minimum Potential Energy principle.

§12.3.1. The Total Potential Energy Functional

In Mechanics of Materials it is shown that the *internal energy density* at a point of a linear-elastic material subjected to a one-dimensional state of stress σ and strain e is $\mathcal{U} = \frac{1}{2}\sigma(x)e(x)$, where σ is to be regarded as linked to the displacement u through Hooke's law $\sigma = Ee$ and the strain-displacement relation $e = u' = du/dx$. This $\mathcal{U}$ is also called the *strain energy density*. Integration over the volume of the bar gives the total internal energy

$$U = \tfrac{1}{2}\int_0^L pe\,dx = \tfrac{1}{2}\int_0^L (EAu')u'\,dx = \tfrac{1}{2}\int_0^L u'EA\,u'\,dx, \tag{12.1}$$

in which all integrand quantities may depend on x.

The *external energy* due to applied mechanical loads pools contributions from two sources:

1. The distributed load $q(x)$. This contributes a cross-section density of $q(x)u(x)$ because q is assumed to be already integrated over the section.
2. Any applied end load(s). For the fixed-free example of Figure 12.2 the end load P would contribute $P\,u(L)$.

The second source may be folded into the first by conventionally writing any point load P acting at a cross section $x = a$ as a contribution $P\,\delta(a)$ to $q(x)$, where $\delta(a)$ denotes the one-dimensional Dirac delta function at $x = a$. If this is done the external energy can be concisely expressed as

$$W = \int_0^L qu\,dx. \tag{12.2}$$

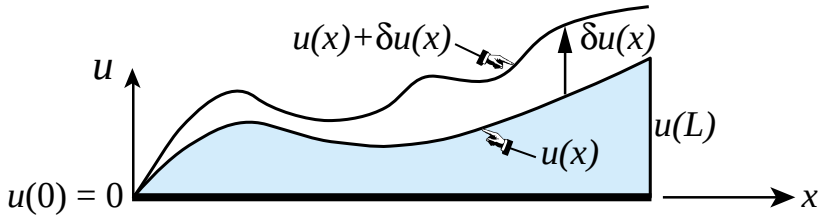


Figure 12.4. Concept of variation of the axial displacement functional $u(x)$. For convenience $u(x)$ is plotted normal to the bar longitudinal axis. Both the $u(x)$ and $u(x) + \delta u(x)$ shown in the figure are kinematically admissible, and so is the variation $\delta u(x)$.

The total potential energy of the bar is given by

$$\boxed{\Pi = U - W} \tag{12.3}$$

Mathematically this is a functional, called the *Total Potential Energy* functional or TPE. It depends only on the axial displacement $u(x)$. In variational calculus this is called the *primary variable* of the functional. When the dependence of Π on u needs to be emphasized we shall write $\Pi[u] = U[u]-W[u]$, with brackets enclosing the primary variable. To display both primary and independent variables we write, for example, $\Pi[u(x)] = U[u(x)] - W[u(x)]$.

§12.3.2. Variation of an Admissible Function

The concept of *admissible variation* is fundamental in both variational calculus and the variationally formulated FEM. *Only the primary variable(s) of a functional may be varied.* For the TPE functional (12.3) this is the axial displacement $u(x)$. Suppose that $u(x)$ is changed to $u(x) + \delta\, u(x)$.[1] This is illustrated in Figure 12.4, where for convenience $u(x)$ is plotted normal to x. The functional changes from Π to $\Pi + \delta\Pi$. The function $\delta\, u(x)$ and the scalar $\delta\Pi$ are called the *variations* of $u(x)$ and Π, respectively. The variation $\delta\, u(x)$ should not be confused with the ordinary differential $du(x) = u'(x)\, dx$ since on taking the variation the independent variable x is frozen; that is, $\delta x = 0$.

A displacement variation $\delta u(x)$ is said to be *admissible* when both $u(x)$ and $u(x) + \delta\, u(x)$ are *kinematically admissible* in the sense of the Principle of Virtual Work (PVW), which agrees with the conditions stated in the classic variational calculus. A kinematically admissible axial displacement $u(x)$ obeys two conditions:

(i) It is continuous over the bar length, that is, $u(x) \in \mathcal{C}_0$ in $x \in [0, L]$.

(ii) It satisfies exactly any displacement boundary condition, such as the fixed-end specification $u(0) = 0$ of Figure 12.2.

The variation $\delta\, u(x)$ depicted in Figure 12.4 is kinematically admissible because both $u(x)$ and $u(x) + \delta\, u(x)$ satisfy the foregoing conditions. The physical meaning of (i)–(ii) is the subject of Exercise 12.1.

[1] The symbol δ not immediately followed by a parenthesis is not a delta function but instead denotes variation with respect to the variable that follows.

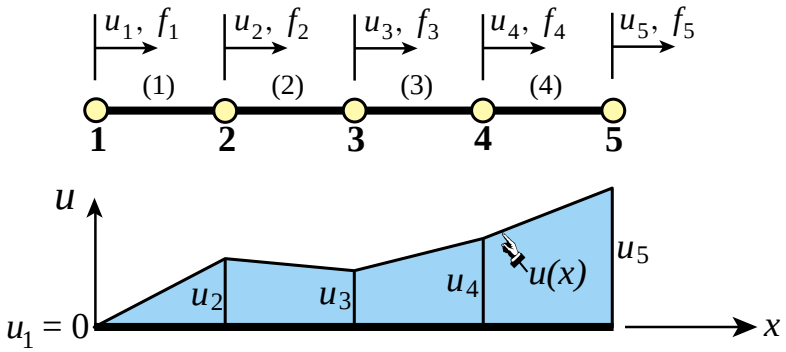


Figure 12.5. FEM discretization of bar member. A piecewise-linear admissible displacement trial function $u(x)$ is drawn below the mesh. It is assumed that the left end is fixed, hence $u_1 = 0$.

§12.3.3. The Minimum Potential Energy Principle

The Minimum Potential Energy (MPE) principle states[2] that the actual displacement solution $u^*(x)$ that satisfies the governing equations is that which renders Π stationary:

$$\boxed{\delta\Pi = \delta U - \delta W = 0 \quad \text{iff} \quad u = u^*} \tag{12.4}$$

with respect to *admissible* variations $u = u^* + \delta u$ of the exact displacement field $u^*(x)$. (The symbol "iff" in (12.4) is an abbreviation for "if and only if".)

REMARK 12.1

Using standard techniques of variational calculus[3] it can be shown that if $EA > 0$ the solution $u^*(x)$ of (12.4) exists, is unique, and renders $\Pi[u]$ a minimum over the class of kinematically admissible displacements. The last attribute explains the "mininum" in the name of the principle.

§12.3.4. TPE Discretization

To apply the TPE functional (12.3) to the derivation of finite element equations we replace the continuum mathematical model by a discrete one consisting of a union of bar elements. For example, Figure 12.5 illustrates the subdivision of a bar member into four two-node elements.

Functionals are scalars. Consequently, corresponding to a discretization such as that shown in Figure 12.5, the TPE functional (12.3) may be decomposed into a sum of contributions of individual elements:

$$\boxed{\Pi = \Pi^{(1)} + \Pi^{(2)} + \ldots + \Pi^{(N_e)}} \tag{12.5}$$

where N_e is the number of elements. The same decomposition applies to the internal and external energies, as well as to the stationarity condition (12.4):

$$\boxed{\delta\Pi = \delta\Pi^{(1)} + \delta\Pi^{(2)} + \ldots + \delta\Pi^{(N_e)} = 0.} \tag{12.6}$$

---

[2] The proof may be found in texts on variational methods in mechanics, e.g., H. L. Langhaar, *Energy Methods in Applied Mechanics*, McGraw-Hill, 1960. This is the most readable "old fashioned" treatment of the energy principles of structural mechanics, with a beautiful treatment of virtual work. Out of print but used copies may be found from web sites.

[3] See for example, I. M. Gelfand and S. V. Fomin, *Calculus of Variations*, Prentice-Hall, 1963.

Using the fundamental lemma of variational calculus,[4] it can be shown that (12.6) implies that for a generic element e we may write

$$\boxed{\delta\Pi^{(e)} = \delta U^{(e)} - \delta W^{(e)} = 0.} \tag{12.7}$$

This *variational equation*[5] is the basis for the derivation of element stiffness equations once the displacement field has been discretized over the element, as described next.

§12.3.5. Bar Element Discretization

Figure 12.5 depicts a generic bar element e. The element is referred to its local axis x. Note that there is no need to call it $\bar{x}$ because in one dimension local and global coordinates coalesce. The two degrees of freedom are u_i and u_j.

The mathematical concept of bar finite elements is based on *approximation* of the axial displacement $u(x)$ over the element. The exact displacement u^* is replaced by an approximate displacement

$$u^*(x) \approx u(x) \tag{12.8}$$

over the finite element mesh. The value of $u(x)$ over element e is denoted by $u^{(e)}(x)$. This approximate displacement, $u(x)$, taken over all elements, is called the *finite element trial expansion* or simply *trial function*. See Figure 12.5.

This FE trial expansion must belong to the class of kinematically admissible displacements defined in §12.3.2. Consequently, it must be C_0 continuous over and between elements.

§12.3.6. Shape Functions

For a two node bar element the only possible variation of the displacement $u^{(e)}$ which satisfies the interelement continuity requirement stated above is *linear*, and expressable by the interpolation formula

$$\boxed{u^{(e)}(x) = N_i^{(e)} u_i^{(e)} + N_j^{(e)} u_j^{(e)} = [\, N_i^{(e)} \quad N_j^{(e)} \,] \begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \mathbf{N}\mathbf{u}^{(e)}.} \tag{12.9}$$

The functions $N_i^{(e)}$ and $N_j^{(e)}$ that multiply the node displacements u_i and u_j are called *shape functions*. These functions *interpolate* the internal displacement $u^{(e)}$ directly from the node values. See Figure 12.6. For the present element the shape functions are linear:

$$N_i^{(e)} = 1 - \frac{x}{\ell} = 1 - \zeta, \qquad N_j^{(e)} = \frac{x}{\ell} = \zeta, \tag{12.10}$$

---

[4] See Gelfand and Fomin, *loc. cit.*, Chapter II.

[5] Mathematically called a *weak form* or *Galerkin form*. Equation (12.7) also states the Principle of Virtual Work for each element: $\delta U^{(e)} = \delta W^{(e)}$, which says that the virtual work of internal and external forces on admissible displacement variations is equal if the element is in equilibrium.

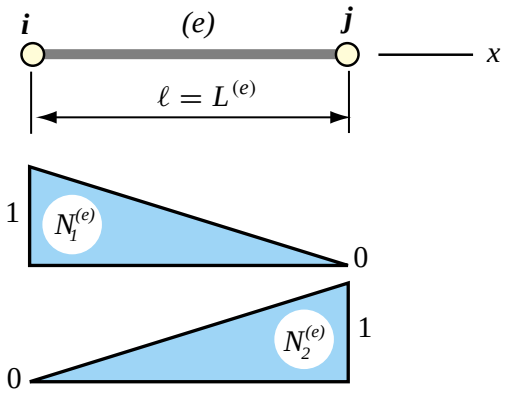


Figure 12.6. The generic two-node bar element and its two shape functions.

in which $\ell = L^{(e)}$ denotes the element length, and $\zeta = x/\ell$ is a dimensionless coordinate, also known as a *natural coordinate*.

Note that the shape function $N_i^{(e)}$ has the value 1 at node i and 0 at node j. Conversely, shape function $N_j^{(e)}$ has the value 0 at node i and 1 at node j. This is a general property of shape functions, required by the fact that that element displacement interpolations such as (12.9) are based on physical node values.

REMARK 12.2

In addition to continuity, shape functions must satisfy a *completeness* requirement with respect to the governing variational principle. This condition is stated and discussed in later Chapters. Suffices for now to say that the shape functions (12.10) do satisfy this requirement.

§12.3.7. The Strain-Displacement Equation

The axial strain over the element is

$$e = \frac{du^{(e)}}{dx} = (u^{(e)})' = \left[\begin{array}{cc} \dfrac{dN_i^{(e)}}{dx} & \dfrac{dN_j^{(e)}}{dx} \end{array}\right] \left[\begin{array}{c} u_i^{(e)} \\ u_j^{(e)} \end{array}\right] = \frac{1}{\ell}\left[\begin{array}{cc} -1 & 1 \end{array}\right] \left[\begin{array}{c} u_i^{(e)} \\ u_j^{(e)} \end{array}\right] = \mathbf{B}\mathbf{u}^{(e)}, \qquad (12.11)$$

where

$$\mathbf{B} = \frac{1}{\ell}\left[\begin{array}{cc} -1 & 1 \end{array}\right] \qquad (12.12)$$

is called the *strain-displacement* matrix.

§12.4. THE FINITE ELEMENT EQUATIONS

In linear finite element analysis, the discretization process for the TPE functional invariably leads to the following algebraic form

$$\Pi^{(e)} = U^{(e)} - W^{(e)}, \quad U^{(e)} = \tfrac{1}{2}(\mathbf{u}^{(e)})^T\mathbf{K}^{(e)}\mathbf{u}^{(e)}, \quad W^{(e)} = (\mathbf{u}^{(e)})^T\mathbf{f}^{(e)}, \qquad (12.13)$$

where $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ are called the *element stiffness matrix* and the *element consistent nodal force vector*, respectively. Note that in (12.13) the three energies are only function of the node displacements $\mathbf{u}^{(e)}$. $U^{(e)}$ and $W^{(e)}$ depend quadratically and linearly, respectively, on those displacements.

Taking the variation of the discretized TPE of (12.13) with respect to the node displacements gives[6]

$$\delta\Pi^{(e)} = \left(\delta\mathbf{u}^{(e)}\right)^T \frac{\partial\Pi^{(e)}}{\partial\mathbf{u}^{(e)}} = \left(\delta\mathbf{u}^{(e)}\right)^T \left[\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{f}^{(e)}\right] = 0. \tag{12.14}$$

Because the variations $\delta\mathbf{u}^{(e)}$ can be arbitrary, the bracketed quantity must vanish, which yields

$$\boxed{\mathbf{K}^{(e)}\mathbf{u}^{(e)} = \mathbf{f}^{(e)}} \tag{12.15}$$

These are the element stiffness equations. Hence the foregoing names given to $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ are justified *a posteriori*.

§12.4.1. The Stiffness Matrix

For the two-node bar element, the internal energy $U^{(e)}$ is

$$U^{(e)} = \tfrac{1}{2}\int_0^\ell e\,EA\,e\,dx, \tag{12.16}$$

where the strain e is related to the nodal displacements through (12.11). This form is symmetrically expanded by inserting $e = \mathbf{B}\mathbf{u}^{(e)}$ into the second e and $e = e^T = (\mathbf{u}^{(e)})^T\mathbf{B}^T$ into the first e:

$$U^{(e)} = \tfrac{1}{2}\int_0^\ell \begin{bmatrix} u_i^{(e)} & u_j^{(e)} \end{bmatrix} \frac{1}{\ell}\begin{bmatrix} -1 \\ 1 \end{bmatrix} EA\,\frac{1}{\ell}\begin{bmatrix} -1 & 1 \end{bmatrix}\begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} dx. \tag{12.17}$$

The nodal displacements can be moved out of the integral, giving

$$U^{(e)} = \tfrac{1}{2}\begin{bmatrix} u_i^{(e)} & u_j^{(e)} \end{bmatrix}\int_0^\ell \frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx \begin{bmatrix} u_i^{(e)} \\ u_j^{(e)} \end{bmatrix} = \tfrac{1}{2}\left(\mathbf{u}^{(e)}\right)^T\mathbf{K}^{(e)}\mathbf{u}^{(e)}. \tag{12.18}$$

in which

$$\boxed{\mathbf{K}^{(e)} = \int_0^\ell \frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} dx = \int_0^\ell EA\,\mathbf{B}^T\mathbf{B}\,dx,} \tag{12.19}$$

is the element stiffness matrix. If the rigidity EA is constant over the element,

$$\mathbf{K}^{(e)} = EA\,\mathbf{B}^T\mathbf{B}\int_0^\ell dx = \frac{EA}{\ell^2}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}\ell = \frac{EA}{\ell}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}. \tag{12.20}$$

This is the same element stiffness matrix of the prismatic truss member derived in Chapters 2 and 6 by Mechanics of Materials arguments, but now obtained through a variational method.

---

[6] The $\frac{1}{2}$ factor disappears on taking the variation because $U^{(e)}$ is quadratic in the node displacements. For a review on the calculus of discrete quadratic forms, see Appendix D.

§12.4.2. The Consistent Node Force Vector

The *consistent node force vector* $\mathbf{f}^{(e)}$ introduced in (12.13) comes from the element contribution to the external work potential W:

$$W^{(e)} = \int_0^\ell qu\,dx = \int_0^\ell q\,\mathbf{N}^T\mathbf{u}^{(e)}\,dx = \left(\mathbf{u}^{(e)}\right)^T \int_0^\ell q \begin{bmatrix} 1-\zeta \\ \zeta \end{bmatrix} dx = \left(\mathbf{u}^{(e)}\right)^T \mathbf{f}^{(e)}, \qquad (12.21)$$

in which $\zeta = x/\ell$. Consequently

$$\boxed{\mathbf{f}^{(e)} = \int_0^\ell q \begin{bmatrix} 1-\zeta \\ \zeta \end{bmatrix} dx} \qquad (12.22)$$

If the force q is constant over the element, one obtains the same results as with the EbE load-lumping method of Chapter 8. See Exercise 12.3.

§12.4.3. When are Two-Node Bar Elements Exact?

Suppose that the following three conditions are satisfied:

1. The bar properties are constant along the length (prismatic member).
2. The distributed force $q(x)$ is zero between nodes.
3. The only applied forces are point forces applied at the nodes.

If so, a linear axial displacement $u(x)$ as defined by (12.9) and (12.10) is the exact solution over each element because constant strain and stress satisfy, element by element, all of the governing equations listed in Figure 12.3.[7] It follows that if the foregoing conditions are verified the FEM solution is *exact*; that is, it agrees with the analytical solution of the mathematical model. Adding extra elements and nodes would not change the solution. That is the reason behind the truss discretizations used in Chapters 2–4: *one element per member is sufficient* if the members are prismatic and the only loads are applied at the joints.[8]

If any of the foregoing conditions is violated, the FEM solution will be approximate, but can be improved by adding more elements, nodes and degrees of freedom.

§12.5. GENERALIZATIONS

The foregoing development pertains to the simplest finite element possible: the linear two-node bar element. If we stay within the realm of one-dimensional elements the technique may be generalized in the following directions:

Refined bar elements. Adding internal nodes we can pass from linear to quadratic and cubic shape functions. These elements are rarely useful on their own right, but as accessories to 2D and 3D

[7] The internal equilibrium equation $p' + q = 0$ is trivially verified because $p' = q = 0$.

[8] In fact, adding more elements per member makes the stiffness equations singular in 2D and 3D because zero-energy mechanisms appear. Removing those mechanisms would require the application of MFCs at intermediate nodes.

high order continuum elements (for example, to model plate edge reinforcements.) For that reason they are not considered here. The 3-node bar element is developed in Exercise 16.5.

Two and three dimensional structures built with bar elements. The only required extra ingredient are the nodal-displacement transformation matrices discussed in Chapters 3 and 6.

Beam-type elements. These require the introduction of displacement-derivative nodal freedoms (interpretable as nodal rotations). Plane beam elements are treated in the next Chapter.

Curved elements. These are derivable using isoparametric mapping. This device will be introduced later when considering triangular and quadrilateral elements in Chapter 16.

Homework Exercises for Chapter 12
Variational Formulation of Bar Element

EXERCISE 12.1

[D:10] Explain the kinematic admissibility requirements stated in §12.3.2 in terms of physics, namely ruling out the possibility of gaps or interpenetration as the bar material deforms.

EXERCISE 12.2

[A/C:15] Using the expression (12.19), derive the stiffness matrix for a *tapered* bar element in which the cross section area varies linearly along the element length:

$$A = A_i(1-\zeta) + A_j\,\zeta \tag{E12.1}$$

where A_i and A_j are the areas at the end nodes, and $\zeta = x/\ell$ is the dimensionless coordinate defined in §12.3.6. Show that this yields the same answer as that of a stiffness of a constant-area bar with cross section $\frac{1}{2}(A_i + A_j)$. Note: the following *Mathematica* script may be used to solve this exercise:[9]

```
ClearAll[Le,x,Em,A,Ai,Aj];
Be={{-1,1}}/Le; ξ=x/Le; A=Ai*(1-ξ)+Aj*ξ;
Ke=Integrate[Em*A*Transpose[Be].Be,{x,0,Le}];
Ke=Simplify[Ke];
Print["Ke for varying cross section bar: ",Ke//MatrixForm];
```

In this and following scripts `Le` stands for ℓ.

EXERCISE 12.3

[A:10] Using the expression (12.22), find the consistent load vector $\mathbf{f}^{(e)}$ for a bar of constant area A subject to a uniform axial force $q = \rho g A$ per unit length along the element. Show that this vector is the same as that obtained with the element-by-element (EbE) "lumping" method of §8.4, which simply assigns half of the total load: $\frac{1}{2}\rho g A\ell$, to each node.

EXERCISE 12.4

[A/C:15] Repeat the previous calculation for the tapered bar element subject to a force $q = \rho g A$ per unit length, in which A varies according to (E12.1) whereas ρ and g are constant. Check that if $A_i = A_j$ one recovers $f_i = f_j = \frac{1}{2}\rho g A\ell$. Note: the following *Mathematica* script may be used to solve this exercise:[10]

```
ClearAll[q,A,Ai,Aj,ρ,g,Le,x];
ζ=x/Le; Ne={{1-ζ,ζ}}; A=Ai*(1-ζ)+Aj*ζ; q=ρ*g*A;
fe=Integrate[q*Ne,{x,0,Le}];
fe=Simplify[fe];
Print["fe for uniform load q: ",fe//MatrixForm];
ClearAll[A];
Print["fe check: ",Simplify[fe/.{Ai->A,Aj->A}]//MatrixForm];
```

[9] The `ClearAll[...]` at the start of the script is recommended programming practice to initialize variables and avoid "cell crosstalk." In a `Module` this is done by listing the local variables after the `Module` keyword.

[10] The `ClearAll[A]` before the last statement is essential; else `A` would retain the previous assignation.

EXERCISE 12.5

[A/C:20] A tapered bar element of length ℓ, end areas A_i and A_j with A interpolated as per (E12.1), and constant density ρ, rotates on a plane at uniform angular velocity ω (rad/sec) about node i. Taking axis x along the rotating bar with origin at node i, the centrifugal axial force is $q(x) = \rho A\omega^2 x$ along the length. Find the consistent node forces as functions of ρ, A_i, A_j, ω and ℓ, and specialize the result to the prismatic bar $A = A_i = A_j$. Partial result check: $f_j = \frac{1}{3}\rho\omega^2 A\ell^2$ for $A = A_i = A_j$.

EXERCISE 12.6

[A:15] (Requires knowledge of Dirac's delta function properties.) Find the consistent load vector $\mathbf{f}^{(e)}$ if the bar is subjected to a concentrated axial force Q at a distance $x = a$ from its left end. Use Equation (12.22), with $q(x) = Q\,\delta(a)$, in which $\delta(a)$ is the one-dimensional Dirac's delta function at $x = a$. Note: the following script does it by *Mathematica*, but it is overkill:

```
ClearAll[Le,q,Q,a,x];
ξ=x/Le; Ne={{1-ξ,ξ}}; q=Q*DiracDelta[x-a];
fe=Simplify[ Integrate[q*Ne,{x,-Infinity,Infinity}] ];
Print["fe for point load Q at x=a: ",fe//MatrixForm];
```

EXERCISE 12.7

[C+D:20] In a learned paper, Dr. I. M. Clueless proposes "improving" the result for the example truss by putting three extra nodes, 4, 5 and 6, at the midpoint of members 1–2, 2–3 and 1–3, respectively. His "reasoning" is that more is better. Try Dr. C.'s suggestion using the *Mathematica* implementation of Chapter 5 and verify that the solution "blows up" because the modified master stiffness is singular. Explain physically what happens.

13

Variational Formulation of Plane Beam Element

TABLE OF CONTENTS

§13.1. INTRODUCTION

The previous Chapter introduced the TPE-based variational formulation of finite elements, which was illustrated for the bar element. This Chapter applies that technique to a more complicated one-dimensional element: the plane beam described by the Bernoulli-Euler mathematical model.

Mathematically, the main difference of beams with respect to bars is the increased order of continuity required for the assumed transverse-displacement functions to be admissible. Not only must these functions be continuous but they must possess continuous x first derivatives. To meet this requirement both deflections *and* slopes are matched at nodal points. Slopes may be viewed as *rotational* degrees of freedom in the small-displacement assumptions used here.

§13.2. WHAT IS A BEAM?

Beams are the most common type of structural component. A *beam* is a bar-like structural member whose primary function is to support *transverse* loading and carry to the supports. See Figure 13.1. By "bar-like" it is meant that one of the dimensions is considerably larger than the other two. This dimension is called the *longitudinal dimension* or *beam axis*. The intersection of planes normal to the longitudinal dimension with the beam member are called *cross sections*. A *longitudinal plane* is one that passes through the beam axis.

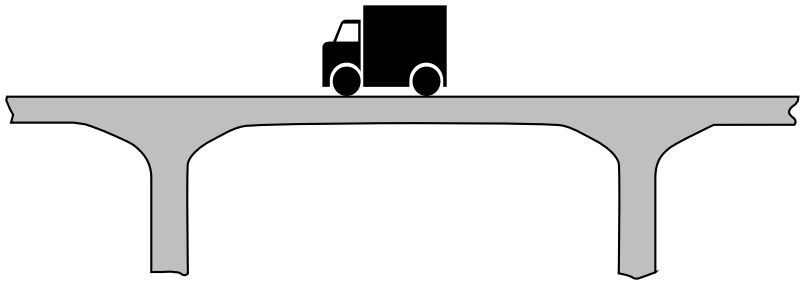


Figure 13.1. A beam is a structural member design to resist transverse loads.

A beam resists transverse loads mainly through *bending action*, as illustrated in Figure 13.2. Such bending produces compressive longitudinal stresses in one side of the beam and tensile stresses in the other. The two regions are separated by a *neutral surface* of zero stress.

The combination of tensile and compressive stresses produces an internal *bending moment*. This moment is the primary mechanism that transports loads to the supports.

§13.2.1. Terminology

A *general beam* is a bar-like member designed to resist a combination of loading actions such as biaxial bending, transverse shears, axial stretching or compression, and possibly torsion. If the internal axial force is compressive, the beam has also to be designed to resist buckling. If the beam is subject primarily to bending and axial forces, it is called a *beam-column*. If it is subjected primarily to bending forces, it is called simply a beam. A beam is *straight* if its longitudinal axis is straight. It is *prismatic* if its cross section is constant.

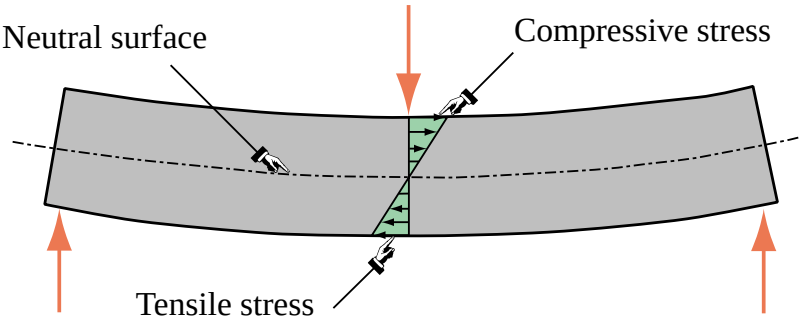


Figure 13.2. Beam transverse loads are primarily resisted by bending action.

A *spatial beam* supports transverse loads that can act on arbitrary directions along the cross section. A *plane beam* resists primarily transverse loading on a preferred longitudinal plane. This Chapter considers only plane beams.

§13.2.2. Mathematical Models

One-dimensional mathematical models of structural beams are constructed on the basis of *beam theories*. Because beams are actually three-dimensional bodies, all models necessarily involve some form of approximation to the underlying physics. The simplest and best known models for straight, prismatic beams are based on the *Bernoulli-Euler* theory, also called *classical beam theory* or *engineering beam theory*, and the *Timoshenko beam theory*. The Bernoulli-Euler theory is that taught in introductory Mechanics of Materials, and is the one used here. The Timoshenko beam theory, which assumes additional importance in dynamics and vibration, is studied in more advanced courses.

Both models can be used to formulate beam finite elements. The classical beam theory used here leads to the so-called *Hermitian* beam elements.[1] These elements neglect transverse shear deformations. Elements based on Timoshenko beam theory incorporate a first order correction for transverse shear effects.

§13.2.3. Assumptions of Classical Beam Theory

The classical (Bernoulli-Euler) theory for *plane beams* is based on the following assumptions:

1. *Planar symmetry*. The longitudinal axis is straight, and the cross section of the beam has a longitudinal plane of symmetry. The resultant of the transverse loads acting on each section lies on this plane.
2. *Cross section variation*. The cross section is either constant or varies smoothly.
3. *Normality*. Plane sections originally normal to the longitudinal axis of the beam remain plane and normal to the deformed longitudinal axis upon bending.

[1] The qualifier "Hermitian" relates to the use of a interpolation method studied by the French mathematician Hermite. The term has nothing to do with the beam model used.

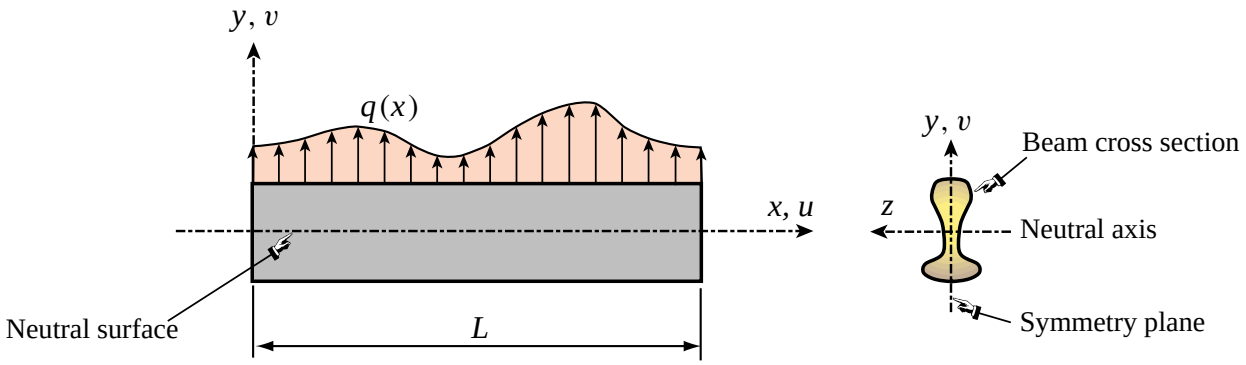


Figure 13.3. Terminology in Bernoulli-Euler model of plane beam.

4. *Strain energy*. The internal strain energy of the beam member accounts only for bending moment deformations. All other effects, notably transverse shear and axial force effects, are ignored.
5. *Linearization*. Transverse deflections, rotations and deformations are considered so small that the assumptions of infinitesimal deformations apply.
6. *Elastic behavior*. The beam is fabricated of material assumed to be elastic and isotropic. Heterogeneous beams fabricated with several materials, such as reinforced concrete, are not excluded.

§13.3. THE CLASSICAL BEAM THEORY

§13.3.1. The Neutral Axis

Under transverse loading one of the beam surfaces shortens while the other elongates; see Figure 13.2. Therefore a *neutral surface* exists between the top and the bottom that undergoes no elongation or contraction. The intersection of this surface with each cross section defines the *neutral axis* of that cross section. If the beam is fabricated of uniform material, the position of the neutral axis is only a function of the cross section geometry.

§13.3.2. Element Coordinate Systems

The Cartesian axes for plane beam analysis are chosen as follows:

1. x along the longitudinal beam axis, at neutral axis height.[2]
2. z along the neutral axis at the origin section.
3. y upwards forming a RHS system with x and z.

The origin of the x, y, z system is placed at the the leftmost section. The total length of the beam member is L. See Figure 13.3.

[2] If the beam is homogenous, the neutral axis passes through the centroid of the cross section. If the beam is fabricated of different materials — for example, a reinforced concrete beam — the neutral axes passes through the centroid of an "equivalent" cross section. This topic is covered in Mechanics of Materials textbooks.

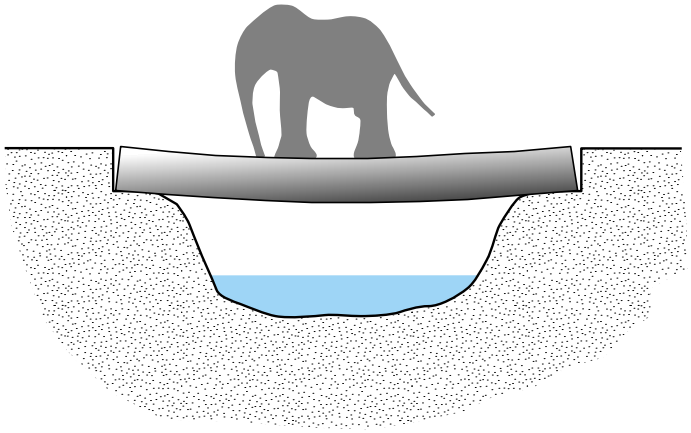


Figure 13.4. A simply supported beam has end supports that preclude transverse displacements but permit end rotations.

§13.3.3. Kinematics

The *motion* of a plane beam member in the x, y plane is described by the two dimensional displacement field

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix}, \tag{13.1}$$

where u and v are the axial and transverse displacement components, respectively, of an arbitrary beam material point. The motion in the z direction, which is primarity due to Poisson's ratio effects, is of no interest. The normality assumption can be represented mathematically as

$$\begin{bmatrix} u(x,y) \\ v(x,y) \end{bmatrix} = \begin{bmatrix} -y\dfrac{\partial v(x)}{\partial x} \\ v(x) \end{bmatrix} = \begin{bmatrix} -yv' \\ v(x) \end{bmatrix} = \begin{bmatrix} -y\theta \\ v(x) \end{bmatrix}. \tag{13.2}$$

Note that the slope $v' = \partial v/\partial x = dv/dx$ of the deflection curve has been identified with the *rotation* symbol θ. This is permissible because θ represents to first order, according to the kinematic assumptions of the Bernoulli-Euler model, the rotation of a cross section about z, positive counterclockwise.

§13.3.4. Loading

The transverse force *per unit length* that acts on the beam in the $+y$ direction is denoted by $q(x)$, as illustrated in Figure 13.3.

Concentrated loads and moments acting on isolated beam sections can be represented by the delta function and its derivative. For example, if a transverse point load F acts at $x = a$, it contributes $F\delta(a)$ to $q(x)$. If the concentrated moment C acts at $x = b$, positive counterclockwise, it contributes $C\delta'(b)$ to $q(x)$, where δ' denotes a doublet acting at $x = b$.

§13.3.5. Support Conditions

Support conditions for beams exhibit far more variety than for bar members. Two canonical cases often encountered in engineering practice: simple support and cantilever support. These are illustrated in Figures 13.4 and 13.5, respectively. Beams often appear as components of skeletal structures called frameworks, in which case the support conditions are of more complex type.

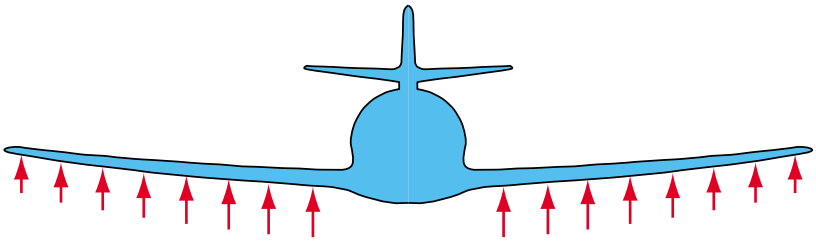


Figure 13.5. A cantilever beam is clamped at one end and free at the other. Airplane wings and stabilizers provide examples of this configuration.

§13.3.6. Strains, Stresses and Bending Moments

The classical (Bernoulli-Euler) model assumes that the internal energy of beam member is entirely due to bending strains and stresses. Bending produces axial stresses σ_{xx}, which will be abbreviated to σ, and axial strains e_{xx}, which will be abbreviated to e. The strains can be linked to the displacements by differentiating the axial displacement $u(x)$ of (13.2):

$$e = \frac{\partial u}{\partial x} = -y\frac{\partial^2 v}{\partial x^2} = -y\frac{d^2 v}{dx^2} = -y\kappa, \tag{13.3}$$

in which κ denotes the deformed beam axis curvature, which to first order is $\partial^2 v/\partial x^2 approx v''$. The bending stress $\sigma = \sigma_{xx}$ is linked to e through the one-dimensional Hooke's law

$$\sigma = Ee = -Ey\frac{d^2 v}{dx^2} = -Ey\kappa, \tag{13.4}$$

where E is the elastic modulus.

The most important stress resultant in classical beam theory is the *bending moment* M, which is defined as the cross section integral

$$M = \int_A -y\sigma\, dx = E\frac{d^2 v}{dx^2}\int_A y^2\, dA = EI\kappa, \tag{13.5}$$

in which I denotes the moment of inertia $\int_A y^2\, dA$ of the cross section with respect to the z (neutral) axis. (In general beams this moment of inertia is identified as I_{zz}.) The product EI is called the *bending rigidity* of the beam with respect to flexure about the z axis.

The governing equations of the Bernoulli-Euler beam model are summarized in the Tonti diagram of Figure 13.6.

§13.4. TOTAL POTENTIAL ENERGY FUNCTIONAL

The total potential energy of the beam is

$$\boxed{\Pi = U - W} \tag{13.6}$$

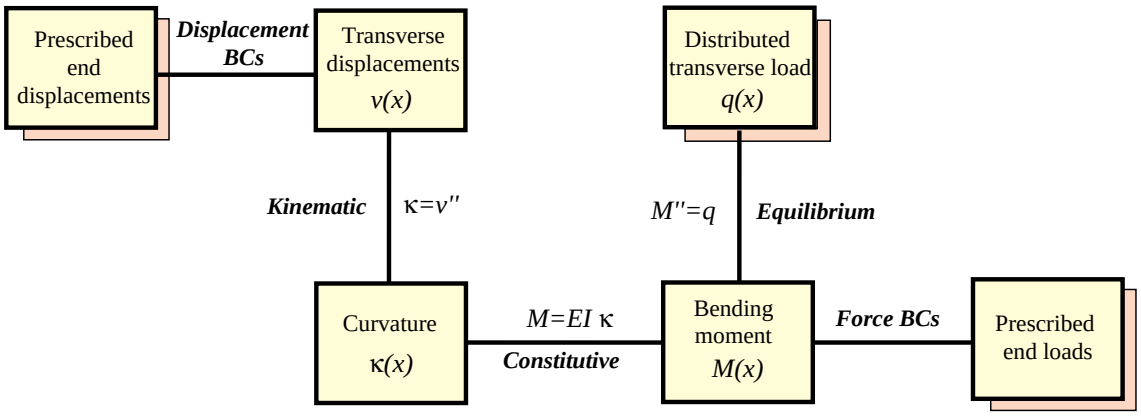


Figure 13.6. The Tonti diagram for the governing equations of the Bernoulli-Euler beam model.

where as usual U and W are the internal and external energies, respectively. As previously explained, in the Bernoulli-Euler model U includes only the bending energy:

$$\boxed{U = \tfrac{1}{2}\int_V \sigma e\, dV = \tfrac{1}{2}\int_0^L M\kappa\, dx = \tfrac{1}{2}\int_0^L EI\kappa^2\, dx = \tfrac{1}{2}\int_0^L EI\left(v''\right)^2\, dx = \tfrac{1}{2}\int_0^L v''EIv''\, dx.} \tag{13.7}$$

The external work W accounts for the applied transverse force:

$$\boxed{W = \int_0^L qv\, dx.} \tag{13.8}$$

The three functionals Π, U and W must be regarded as depending on the transverse displacement $v(x)$. When this dependence needs to be emphasized we write $\Pi[v]$, $U[v]$ and $W[v]$.

Observe that $\Pi[v]$ includes up to second derivatives in v, because $v'' = \kappa$ appears in U. This number (the order of the highest derivatives present in the functional) is called the *variational index*. Variational calculus tells us that since the variational index is 2, admissible displacements $v(x)$ must be continuous, have continuous first derivatives (slopes or rotations), and satisfy the displacement boundary conditions exactly.

This continuity requirement can be succintly stated by saying that admissible displacements must be C^1 continuous. This guides the construction of beam finite elements described below.

REMARK 13.1

If there is an applied distributed moment $m(x)$ per unit of beam length, the external energy (13.8) must be augmented with a $\int_0^L m(x)\theta(x)\, dx$ term. This is further elaborated in Exercises 13.4 and 13.5. Such kind of distributed loading is uncommon in practice although in framework analysis occassionally the need arises for treating a concentrated moment C between nodes.

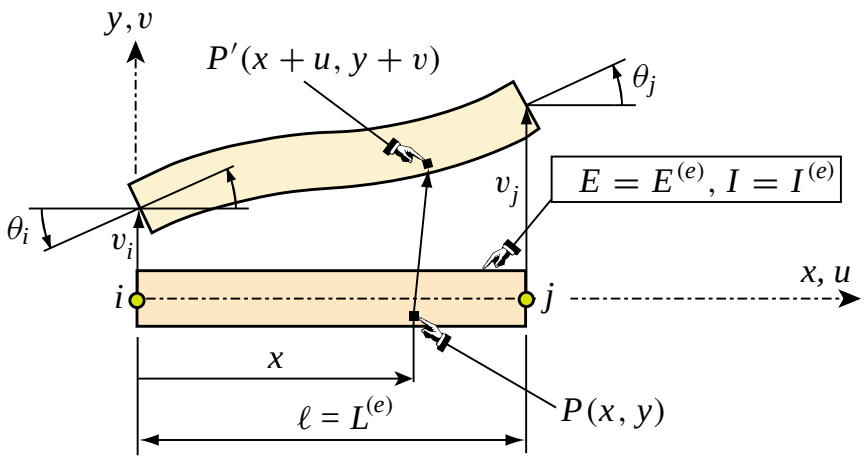


Figure 13.7. The two-node plane beam element with four degrees of freedom.

§13.5. BEAM FINITE ELEMENTS

Beam finite elements are obtained by subdividing beam members longitudinally. The simplest Bernoulli-Euler plane beam element, depicted in Figure 13.7, has two end nodes, i and j, and four degrees of freedom:

$$\mathbf{u}^{(e)} = \begin{bmatrix} v_i \\ \theta_i \\ v_j \\ \theta_j \end{bmatrix}. \tag{13.9}$$

§13.5.1. Finite Element Trial Functions

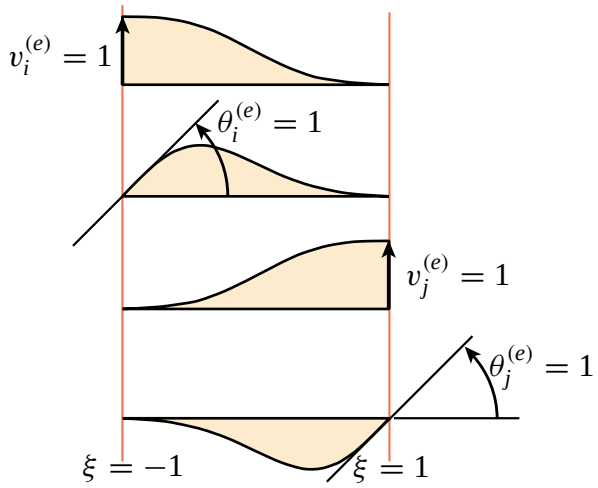
The freedoms (13.9) must be used to define uniquely the variation of the transverse displacement $v^{(e)}(x)$ over the element. The C^1 continuity requirement stated at the end of the previous Section says that both w and the slope $\theta = v'$ must be continuous over the entire beam member, and in particular between beam elements.

C^1 continuity can be trivially satisfied within each element by choosing polynomial interpolation functions as shown below. Matching the nodal displacements and rotations with adjacent beam elements enforces the necessary interelement continuity.

§13.5.2. Shape Functions

The simplest shape functions that meet the C^1 continuity requirements for the nodal freedoms (13.9) are called the *Hermitian cubic* shape functions. The interpolation formula based on these functions may be written as

$$v^{(e)} = \begin{bmatrix} N_{v_i}^{(e)} & N_{\theta_i}^{(e)} & N_{v_j}^{(e)} & N_{\theta_j}^{(e)} \end{bmatrix} \begin{bmatrix} v_i^{(e)} \\ \theta_i^{(e)} \\ v_j^{(e)} \\ \theta_j^{(e)} \end{bmatrix} = \mathbf{N}\mathbf{u}^{(e)}. \tag{13.10}$$



$$N_{vi}^{(e)}(\xi) = \tfrac{1}{4}(1-\xi)^2(2+\xi)$$

$$N_{\theta i}^{(e)}(\xi) = \tfrac{1}{8}\ell(1-\xi)^2(1+\xi)$$

$$N_{vj}^{(e)}(\xi) = \tfrac{1}{4}(1+\xi)^2(2-\xi)$$

$$N_{\theta j}^{(e)}(\xi) = -\tfrac{1}{8}\ell(1+\xi)^2(1-\xi)$$

Figure 13.8. Hermitian shape functions of plane beam element

These shape functions are conveniently written in terms of the dimensionless "natural" coordinate

$$\xi = \frac{2x}{\ell} - 1, \tag{13.11}$$

which varies from -1 at node i ($x = 0$) to $+1$ at node j ($x = \ell$); ℓ being the element length:

$$\boxed{\begin{aligned} N_{v_i}^{(e)} &= \tfrac{1}{4}(1-\xi)^2(2+\xi), \qquad & N_{\theta_i}^{(e)} &= \tfrac{1}{8}\ell(1-\xi)^2(1+\xi), \\ N_{v_j}^{(e)} &= \tfrac{1}{4}(1+\xi)^2(2-\xi), \qquad & N_{\theta_j}^{(e)} &= -\tfrac{1}{8}\ell(1+\xi)^2(1-\xi). \end{aligned}} \tag{13.12}$$

These functions are depicted in Figure 13.8.

The curvature κ that appears in U can be expressed in terms of the nodal displacements by differentiating twice with respect to x:

$$\kappa = \frac{d^2 v^{(e)}(x)}{dx^2} = \frac{4}{\ell^2}\frac{d^2 v^{(e)}(\xi)}{d\xi^2} = \frac{4}{\ell^2}\frac{d\mathbf{N}^{(e)}}{d\xi^2}\mathbf{u}^{(e)} = \mathbf{B}\mathbf{u}^{(e)} = \mathbf{N}''\mathbf{u}^{(e)}. \tag{13.13}$$

Here $\mathbf{B} = \mathbf{N}''$ is the 1×4 curvature-displacement matrix

$$\boxed{\mathbf{B} = \frac{1}{\ell}\left[\,6\frac{\xi}{\ell} \quad 3\xi - 1 \quad -6\frac{\xi}{\ell} \quad 3\xi + 1\,\right]}. \tag{13.14}$$

REMARK 13.2

The $4/\ell^2$ factor that shows up in (13.13) comes from the differentiation chain rule. If $f(x)$ is a function of x, and $\xi = 2x/\ell - 1$,

$$\begin{aligned} \frac{df(x)}{dx} &= \frac{df(\xi)}{d\xi}\frac{d\xi}{dx} = \frac{2}{\ell}\frac{df(\xi)}{d\xi}, \\ \frac{d^2 f(x)}{dx^2} &= \frac{d(2/\ell)}{dx}\frac{df(\xi)}{d\xi} + \frac{2}{\ell}\frac{d}{dx}\left(\frac{df(\xi)}{d\xi}\right) = \frac{4}{\ell^2}\frac{d^2 f(\xi)}{d\xi^2}, \end{aligned} \tag{13.15}$$

because $d(2/\ell)/dx = 0$.

§13.6. THE FINITE ELEMENT EQUATIONS

Insertion of (13.12) and (13.14) into the TPE functional specialized to the element, yields the quadratic form in the nodal displacements

$$\Pi^{(e)} = \tfrac{1}{2}\mathbf{u}^{(e)\,T}\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{u}^{(e)\,T}\mathbf{f}^{(e)}, \tag{13.16}$$

where

$$\boxed{\mathbf{K}^{(e)} = \int_0^\ell E\,I\;\mathbf{B}^T\mathbf{B}\,dx = \int_{-1}^{1} E\,I\;\mathbf{B}^T\mathbf{B}\;\tfrac{1}{2}\ell\,d\xi,} \tag{13.17}$$

is the element stiffness matrix and

$$\boxed{\mathbf{f}^{(e)} = \int_0^\ell \mathbf{N}^T q\,dx = \int_{-1}^{1} \mathbf{N}^T q\;\tfrac{1}{2}\ell\,d\xi,} \tag{13.18}$$

is the consistent element nodal force vector.

The calculation of the entries of $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ for prismatic beams and uniform load q is studied next. More complex cases are treated in the Exercises.

§13.6.1. The Stiffness Matrix of a Prismatic Beam

If the bending rigidity $E\,I$ is constant over the element it can be moved out of the ξ-integral in (13.17):

$$\mathbf{K}^{(e)} = \tfrac{1}{2}E\,I\,\ell\int_{-1}^{1}\mathbf{B}^T\mathbf{B}\,d\xi = \frac{E\,I}{2\ell}\int_{-1}^{1}\begin{bmatrix}\frac{6\xi}{\ell}\\ 3\xi-1\\ \frac{-6\xi}{\ell}\\ 3\xi+1\end{bmatrix}\begin{bmatrix}\frac{6\xi}{\ell} & 3\xi-1 & \frac{-6\xi}{\ell} & 3\xi+1\end{bmatrix}d\xi. \tag{13.19}$$

Expanding and integrating over the element yields

$$\begin{aligned}\mathbf{K}^{(e)} &= \frac{E\,I}{2\ell^3}\int_{-1}^{1}\begin{bmatrix}36\xi^2 & 6\xi(3\xi-1)\ell & -36\xi^2 & 6\xi(3\xi+1)\ell\\ & (3\xi-1)^2\ell^2 & -6\xi(3\xi-1)\ell & (9\xi^2-1)\ell^2\\ & & 36\xi^2 & -6\xi(3\xi+1)\ell\\ \textit{symm} & & & (3\xi+1)^2\ell^2\end{bmatrix}d\xi\\ &= \frac{E\,I}{\ell^3}\begin{bmatrix}12 & 6\ell & -12 & 6\ell\\ & 4\ell^2 & -6\ell & 2\ell^2\\ & & 12 & -6\ell\\ \textit{symm} & & & 4\ell^2\end{bmatrix}\end{aligned} \tag{13.20}$$

Although the foregoing integrals can be easily carried out by hand, it is equally expedient to use a CAS such as *Mathematica* or *Maple*. For example the *Mathematica* script listed in the top box of Figure 13.9 processes (13.20) using the `Integrate` function. The output, shown in the bottom box, corroborates the hand integration result.

```
ClearAll[EI,l,ξ];
B={{6*ξ,(3*ξ-1)*l,-6*ξ,(3*ξ+1)*l}}/l^2;
Ke=(EI*l/2)*Integrate[Transpose[B].B,{ξ,-1,1}];
Ke=Simplify[Ke];
Print["Ke for prismatic beam:"];
Print[Ke//MatrixForm];
```

Ke for prismatic beam:

$$\begin{pmatrix} \frac{12\,EI}{l^3} & \frac{6\,EI}{l^2} & -\frac{12\,EI}{l^3} & \frac{6\,EI}{l^2} \\ \frac{6\,EI}{l^2} & \frac{4\,EI}{l} & -\frac{6\,EI}{l^2} & \frac{2\,EI}{l} \\ -\frac{12\,EI}{l^3} & -\frac{6\,EI}{l^2} & \frac{12\,EI}{l^3} & -\frac{6\,EI}{l^2} \\ \frac{6\,EI}{l^2} & \frac{2\,EI}{l} & -\frac{6\,EI}{l^2} & \frac{4\,EI}{l} \end{pmatrix}$$

Figure 13.9. Using *Mathematica* to form $\mathbf{K}^{(e)}$ for a prismatic beam element.

```
ClearAll[q,l,ξ]
Ne={{2*(1-ξ)^2*(2+ξ), (1-ξ)^2*(1+ξ)*l,
     2*(1+ξ)^2*(2-ξ),-(1+ξ)^2*(1-ξ)*l}}/8;
fe=(q*l/2)*Integrate[Ne,{ξ,-1,1}]; fe=Simplify[fe];
Print["fe^T for uniform load q:"];
Print[fe//MatrixForm];
```

fe^T for uniform load q:

$$\begin{pmatrix} \frac{l\,q}{2} & \frac{l^2\,q}{12} & \frac{l\,q}{2} & -\frac{l^2\,q}{12} \end{pmatrix}$$

Figure 13.10. Using *Mathematica* to form $\mathbf{f}^{(e)}$ for uniform transverse load q.

§13.6.2. Consistent Nodal Force Vector for Uniform Load

If q does not depend on x it can be moved out of (13.18), giving

$$\mathbf{f}^{(e)} = \tfrac{1}{2}q\ell \int_{-1}^{1} \mathbf{N}^T\, d\xi = \tfrac{1}{2}q\ell \int_{-1}^{1} \begin{bmatrix} \frac{1}{4}(1-\xi)^2(2+\xi) \\ \frac{1}{8}\ell(1-\xi)^2(1+\xi) \\ \frac{1}{4}(1+\xi)^2(2-\xi) \\ -\frac{1}{8}\ell(1+\xi)^2(1-\xi) \end{bmatrix} d\xi = q\ell \begin{bmatrix} \frac{1}{2} \\ \frac{1}{12}\ell \\ \frac{1}{2} \\ -\frac{1}{12}\ell \end{bmatrix}. \qquad (13.21)$$

This shows that a uniform load q over the beam element maps to two transverse node loads $q\ell/2$, as may be expected, plus two nodal moments $\pm q\ell^2/12$. The latter are called the *fixed-end moments* in the FEM literature.

The hand result (13.21) can be verified with the *Mathematica* script displayed in Figure 13.10, in which $\mathbf{f}^{(e)}$ is printed as a row vector to save space.

Homework Exercises for Chapter 13
Variational Formulation of Plane Beam Element

EXERCISE 13.1

[A/C:20] Use (13.17) to derive the element stiffness matrix $\mathbf{K}^{(e)}$ of a Hermitian beam element of variable bending rigidity given by the inertia law

$$I(x) = I_i(1 - \frac{x}{\ell}) + I_j \frac{x}{\ell} = I_i \tfrac{1}{2}(1-\xi) + I_j \tfrac{1}{2}(1+\xi). \tag{E13.1}$$

Use of *Mathematica* or similar CAS tool is recommended since the integrals are time consuming. *Mathematica* hint: write

```
EI = EIi*(1-ξ)/2 + EIj*(1+ξ)/2;
```
(E13.2)

and keep `EI` inside the argument of `Integrate`. Check whether you get back (13.20) if `EI=EIi=EIj`. If you use *Mathematica*, this check can be simply done after you got and printed the tapered beam `Ke`, by writing `ClearAll[EI]; Ke=Simplify[ Ke/.{EIi->EI,EIj->EI}];` and printing this matrix.[3]

EXERCISE 13.2

[A/C:20] Use (13.18) to derive the consistent nodal force vector $\mathbf{f}^{(e)}$ for a Hermitian beam element under linearly varying transverse load q defined by

$$q(x) = q_i(1 - \frac{x}{\ell}) + q_j \frac{x}{\ell} = q_i \tfrac{1}{2}(1-\xi) + q_j \tfrac{1}{2}(1+\xi). \tag{E13.3}$$

Again use of a CAS is recommended, particularly since the polynomials to be integrated are quartic in ξ, and hand computations are error prone. *Mathematica* hint: write

```
q = qi*(1-ξ)/2 + qj*(1+ξ)/2;
```
(E13.4)

and keep `q` inside the argument of `Integrate`. Check whether you get back (13.21) if $q_i = q_j = q$ (See previous Exercise for *Mathematica* procedure).

EXERCISE 13.3

[A:20] Obtain the consistent node force vector $\mathbf{f}^{(e)}$ of a Hermitian beam element subject to a transverse point load P at abscissa $x = a$ where $0 \le a \le \ell$. Use the Dirac's delta function expression $q(x) = P\,\delta(a)$ and the fact that for any continuous function $f(x)$, $\int_0^\ell f(x)\,\delta(a)\,dx = f(a)$ if $0 \le a \le \ell$.

EXERCISE 13.4

[A:25] Derive the consistent node force vector $\mathbf{f}^{(e)}$ of a Hermitian beam element subject to a uniformly distributed z-moment m per unit length. Use the fact that the external work per unit length is $m(x)\theta(x) = m(x)\,v'(x) = (\mathbf{u}^{(e)})^T (d\mathbf{N}/dx)^T\, m(x)$. For arbitrary $m(x)$ show that this gives

$$\mathbf{f}^{(e)} = \int_0^\ell \frac{\partial \mathbf{N}^T}{\partial x} m\,dx = \int_{-1}^{1} \frac{\partial \mathbf{N}^T}{\partial \xi} \frac{2}{\ell} m\,\tfrac{1}{2}\ell\,d\xi = \int_{-1}^{1} \mathbf{N}_\xi^T m\,d\xi, \tag{E13.5}$$

where $\mathbf{N}_\xi^T$ denote the column vectors of beam shape function derivatives with respect to ξ. Can you see a shortcut that avoids the integral for constant m?

---

[3] `ClearAll[EI]` discards the previous definition (E13.2) of `EI`; the same effect can be achieved by writing `EI=.` (dot).

EXERCISE 13.5

[A:20] Obtain the consistent node force vector $\mathbf{f}^{(e)}$ of a Hermitian beam element subject to a concentrated moment C applied at $x = a$. Use the expression (E13.5) in which $m(x) = C\,\delta(a)$, where $\delta(a)$ denotes the Dirac's delta function at $x = a$.

EXERCISE 13.6

[A/C:25] Consider the one-dimensional Gauss integration rules.

$$\text{One point :} \qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq 2\,f(0) \tag{E13.6}$$

$$\text{Two points:} \qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq f(-1/\sqrt{3}) + f(1/\sqrt{3}) \tag{E13.7}$$

$$\text{Three points:} \qquad \int_{-1}^{1} f(\xi)\,d\xi \doteq \frac{5}{9}\,f(-\sqrt{3/5}) + \frac{8}{9}\,f(0) + \frac{5}{9}\,f(\sqrt{3/5}) \tag{E13.8}$$

Try each rule on the monomial integrals

$$\int_{-1}^{1} d\xi, \qquad \int_{-1}^{1} \xi\,d\xi, \qquad \int_{-1}^{1} \xi^2\,d\xi, \quad \ldots \tag{E13.9}$$

until the rule fails. In this way verify that the rules (E13.6), (E13.7) and (E13.8) are exact for polynomials of degree up to 1, 3 and 5, respectively. (*Labor-saving hint*: for odd monomial degree no computations need to be done; why?).

EXERCISE 13.7

[A/C:25] Repeat the derivation of Exercise 13.1 using the two-point Gauss rule (E13.7) to evaluate integrals in ξ. A CAS is recommended. If using *Mathematica* you may use a function definition to save typing; for example to evaluate $\int_{-1}^{1} f(\xi)\,d\xi$ in which $f(\xi) = 6\xi^4 - 3\xi^2 + 7$, by the 3-point Gauss rule (E13.8), say

```
f[ξ_]:=6ξ^4-3ξ^2+7; int=Simplify[(5/9)*(f[-Sqrt[3/5]]+f[Sqrt[3/5]])+(8/9)*f[0]];
```

and print `int`. To form an element by Gauss integration define matrix functions in terms of ξ, for example `Be[ξ_]`, or use the substitution operator `/.`, whatever you prefer. Check whether one obtains the same answers as with analytical integration, and explain why there is agreement or disagreement. Hint for the explanation: consider the order of the ξ polynomials you are integrating over the element.

EXERCISE 13.8

[A/C:25] As above but for Exercise 13.2.

14

The Plane Stress Problem

TABLE OF CONTENTS

§14.1. INTRODUCTION

We now pass to the variational formulation of two-dimensional *continuum* finite elements. The problem of plane stress will serve as the vehicle for illustrating such formulations. As narrated in Appendix H, continuum finite elements were invented in the aircraft industry[1] to solve this kind of problem when it arose in the design and analysis of delta wing panels.

The problem is presented here within the framework of the linear theory of elasticity.

§14.1.1. Plate in Plane Stress

In structural mechanics, a flat thin sheet of material is called a *plate*.[2] The distance between the plate faces is called the *thickness* and denoted by h. The *midplane* lies halfway between the two faces. The direction normal to the midplane is called the *transverse* direction. Directions parallel to the midplane are called *in-plane* directions. The global axis z will be oriented along the transverse direction. Axes x and y are placed in the midplane, forming a right-handed Rectangular Cartesian Coordinate (RCC) system. Thus the midplane equation is $z = 0$. See Figure 14.1.

A plate loaded in its midplane is said to be in a state of *plane stress*, or a *membrane state*, if the following assumptions hold:

1. All loads applied to the plate act in the midplane direction, and are symmetric with respect to the midplane.
2. All support conditions are symmetric about the midplane.
3. In-plane displacements, strains and stresses can be taken to be uniform through the thickness.
4. The normal and shear stress components in the z direction are zero or negligible.

The last two assumptions are not necessarily consequences of the first two. For the latter to hold, the thickness h should be small, typically 10% or less, than the shortest in-plane dimension. If the plate thickness varies it should do so gradually. Finally, the plate fabrication must exhibit symmetry with respect to the midplane.

To these four assumptions we add the following restriction:

5. The plate is fabricated of the same material though the thickness. Such plates are called *transversely homogeneous* or *monocoque* plates.

The last assumption excludes wall constructions of importance in aerospace, in particular composite and honeycomb plates. The development of models for such configurations requires a more complicated integration over the thickness as well as the ability to handle coupled bending and stretching effects, and will not be considered here.

REMARK 14.1

Selective relaxation from assumption 4 lead to the so-called *generalized plane stress state*, in which z stresses are accepted. The *plane strain state* is obtained if strains in the z direction is precluded. Although the

[1] At Boeing over the period 1952-53.

[2] If it is relatively thick, as in concrete pavements, or cheese slices, the term *slab* is also used but not for plane stress conditions.

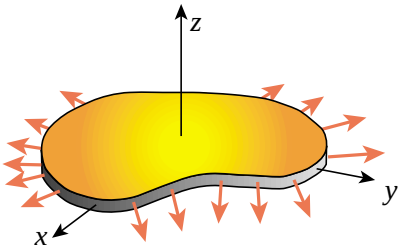


Figure 14.1. A plate structure in plane stress.

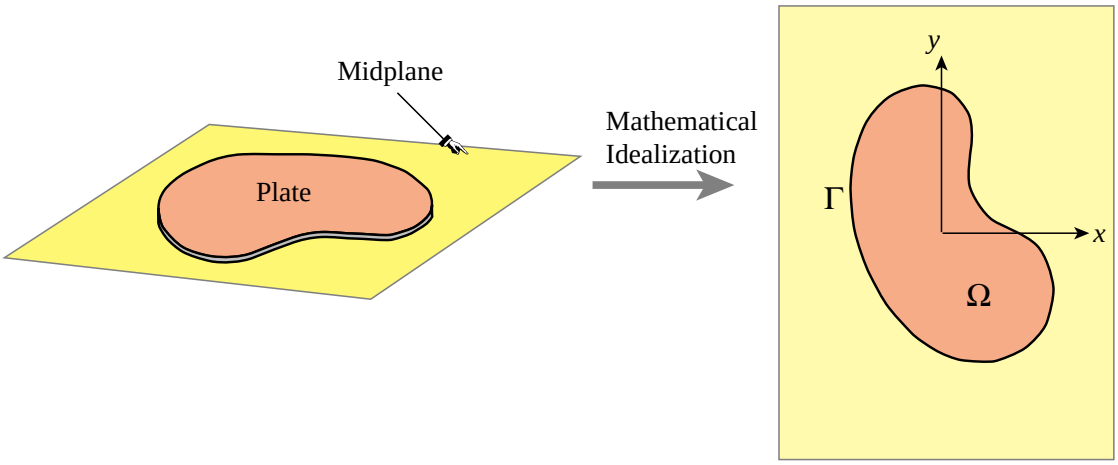


Figure 14.2. Mathematical model of plate in plane stress.

construction of finite element models for those states has many common points with plane stress, we shall not consider those models here. For isotropic materials the plane stress and plane strain problems can be mapped into each other through a fictitious-property technique; see Exercise 14.1.

REMARK 14.2

Transverse loading on a plate produces *plate bending*, which is associated with a more complex configuration of internal forces and deformations. This subject is studied in more advanced courses.

§14.1.2. Mathematical Model

The mathematical model of the plate in plane stress is a two-dimensional boundary value problem (BVP). This problem is posed over a plane domain Ω with a boundary Γ, as illustrated in Figure 14.2.

In this idealization the third dimension is represented as functions of x and y that are *integrated through the plate thickness*. Engineers often work with internal plate forces, which result from integrating the in-plane stresses through the thickness. See Figure 14.3.

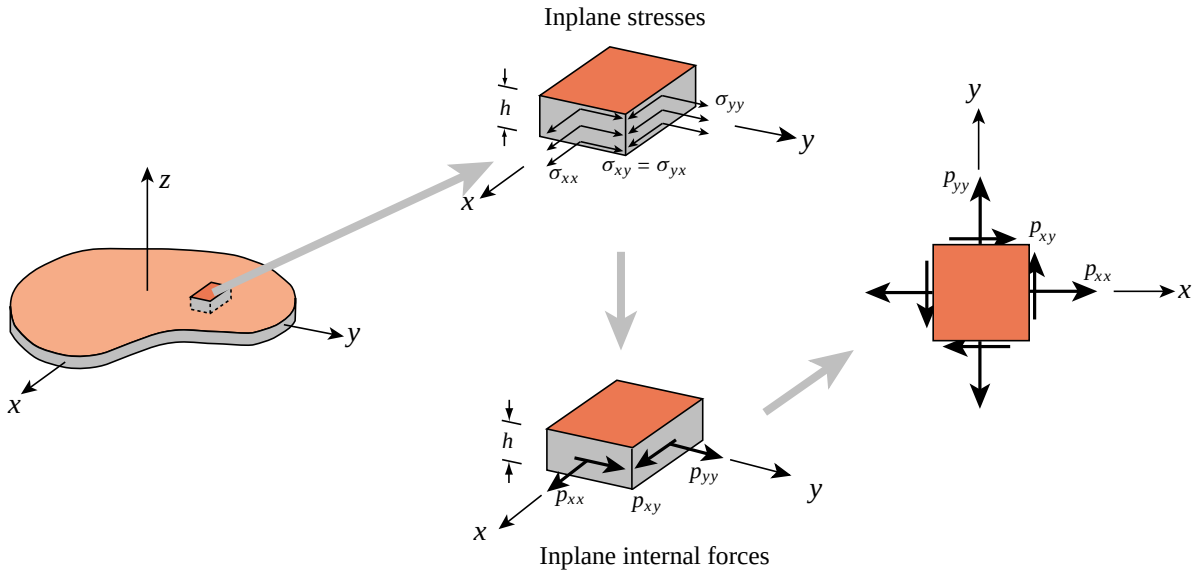


Figure 14.3. Internal stresses and plate forces.

§14.2. PLANE STRESS PROBLEM DESCRIPTION

§14.2.1. Problem Data

The given data includes:

Domain geometry. This is defined by the boundary Γ illustrated in Figure 14.2.

Thickness. Most plates used as structural components have constant thickness. If the thickness does vary, in which case $h = h(x, y)$, it should do so gradually to maintain the plane stress state.

Material data. This is defined by the constitutive equations. Here we shall assume that the plate material is linearly elastic but not necessarily isotropic.

Specified Interior Forces. These are known forces that act in the interior Ω of the plate. There are of two types. *Body forces* or *volume forces* are forces specified per unit of plate volume; for example the plate weight. *Face forces* act tangentially to the plate faces and are transported to the midplane. For example, the friction or drag force on an airplane skin is of this type if the skin is modeled to be in plane stress.

Specified Surface Forces. These are known forces that act on the boundary Γ of the plate. In elasticity these are called *surface tractions*. In actual applications it is important to know whether these forces are specified per unit of surface area or per unit length.

Displacement Boundary Conditions. These specify how the plate is supported. Points on the plate boundary may be fixed, allowed to move in one direction, or subject to multipoint constraints. In addition symmetry and antisymmetry lines may be identified as discussed in Chapter 8.

If no displacement boundary conditions are imposed, the plate structure is said to be *free-free*.

§14.2.2. Problem Unknowns

The three unknown fields are displacements, strains and stresses. Because of the assumed wall fabrication homogeneity the in-plane components are assumed to be *uniform through the plate thickness*. Consequently the dependence on z disappears and all such components become functions of x and y only.

Displacements. The in-plane displacement field is defined by two components:

$$\mathbf{u}(x,y) = \begin{bmatrix} u_x(x,y) \\ u_y(x,y) \end{bmatrix} \tag{14.1}$$

The transverse displacement component $u_z(x,y,z)$ component is generally nonzero because of Poisson's ratio effects, and depends on z. However, this displacement does not appear in the governing equations.

Strains. The in-plane strain field forms a tensor defined by three independent components: e_{xx}, e_{yy} and e_{xy}. To allow stating the FE equations in matrix form, these components are conventionally arranged to form a 3-component "strain vector"

$$\mathbf{e}(x,y) = \begin{bmatrix} e_{xx}(x,y) \\ e_{yy}(x,y) \\ 2e_{xy}(x,y) \end{bmatrix} \tag{14.2}$$

The factor of 2 in e_{xy} shortens strain energy expressions. The shear strain components e_{xz} and e_{yz} vanish. The transverse normal strain e_{zz} is generally nonzero because of Poisson effects. This strain does not enter the governing equations as unknown because the associated stress σ_{zz} is zero, which eliminates the contribution of $\sigma_{zz}e_{zz}$ to the internal energy.

Stresses. The in-plane stress field forms a tensor defined by three independent components: σ_{xx}, σ_{yy} and σ_{xy}. As in the case of strains, to allow stating the FE equations in matrix form, these components are conventionally arranged to form a 3-component "stress vector"

$$\boldsymbol{\sigma}(x,y) = \begin{bmatrix} \sigma_{xx}(x,y) \\ \sigma_{yy}(x,y) \\ \sigma_{xy}(x,y) \end{bmatrix} \tag{14.3}$$

The remaining three stress components: σ_{zz}, σ_{xz} and σ_{yz}, are assumed to vanish.

The *plate internal forces* are obtained on integrating the stresses through the thickness. Under the assumption of uniform stress distribution,

$$p_{xx} = \sigma_{xx}h, \quad p_{yy} = \sigma_{yy}h, \quad p_{xy} = \sigma_{xy}h. \tag{14.4}$$

These p's also form a tensor. They are often called *membrane forces* in the literature. See Figure 14.3.

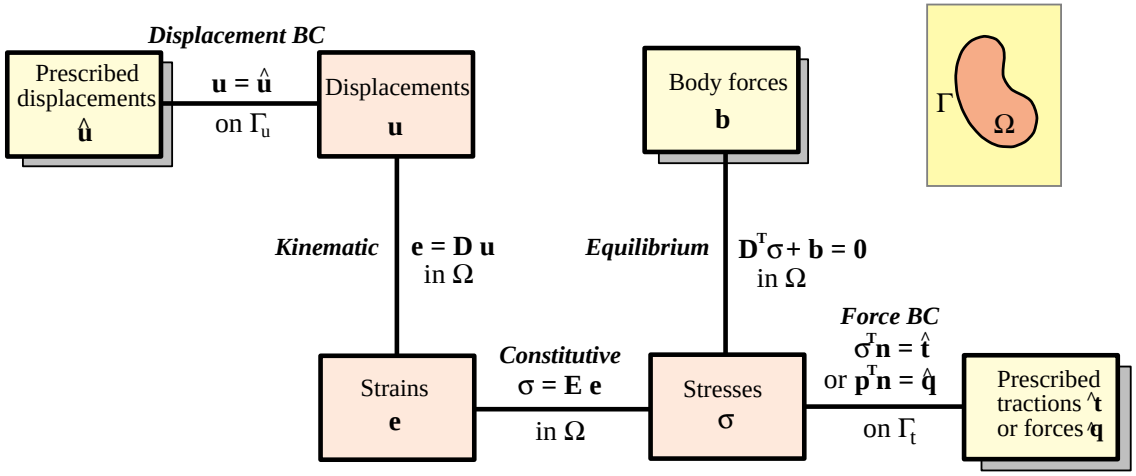


Figure 14.4. The Strong Form of the plane stress equations of linear elastostatics displayed as a Tonti diagram. Yellow boxes identify prescribed fields whereas orange boxes denote unknown fields. The distinction between Strong and Weak Forms is explained in §14.3.3.

§14.3. LINEAR ELASTICITY EQUATIONS

We shall develop plane stress finite elements in the framework of classical linear elasticity. The necessary governing equations are presented below. They are graphically represented in the Strong Form Tonti diagram of Figure 14.4.

§14.3.1. Governing Equations

The three internal fields: displacements, strains and stresses (14.2)-(14.4) are connected by three field equations: kinematic, constitutive and internal-equilibrium equations. If initial strain effects are ignored, these equations read

$$
\begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix} = \begin{bmatrix} \partial/\partial x & 0 \\ 0 & \partial/\partial y \\ \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} u_x \\ u_y \end{bmatrix},
$$
$$
\begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \begin{bmatrix} E_{11} & E_{12} & E_{13} \\ E_{12} & E_{22} & E_{23} \\ E_{13} & E_{23} & E_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \tag{14.5}
$$
$$
\begin{bmatrix} \partial/\partial x & 0 & \partial/\partial y \\ 0 & \partial/\partial y & \partial/\partial x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} + \begin{bmatrix} b_x \\ b_y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.
$$

In these equations, b_x and b_y are the components of the interior body force $\mathbf{b}$, $\mathbf{E}$ is the 3×3 stress-strain matrix of plane stress elastic moduli, $\mathbf{D}$ is the 3×2 symmetric-gradient operator and its transpose the 2×3 tensor-divergence operator. The dependence on (x, y) has been omitted to reduce clutter.

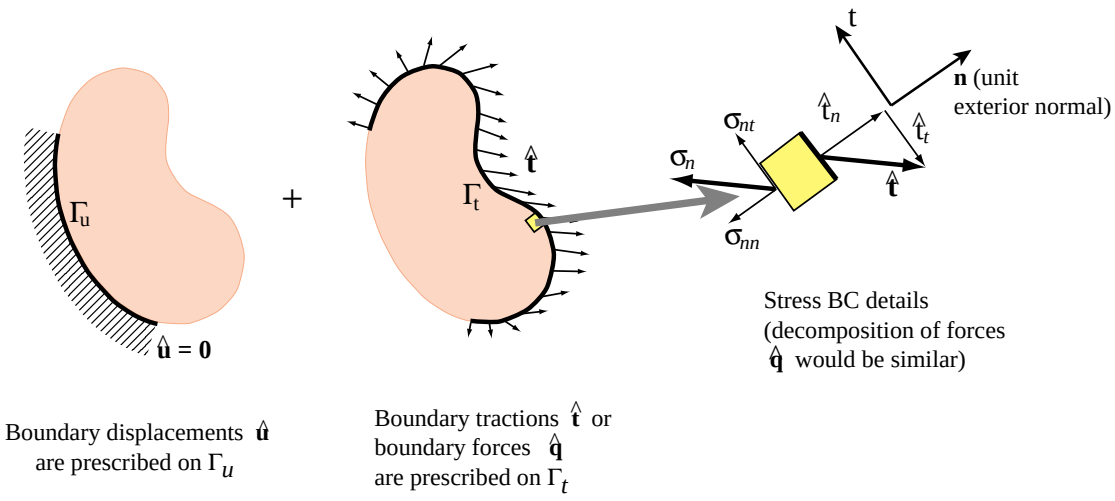


Figure 14.5. Displacement and force (stress, traction) boundary conditions for the plane stress problem.

The compact matrix version of (14.5) is

$$\boxed{\mathbf{e} = \mathbf{D}\mathbf{u}, \qquad \boldsymbol{\sigma} = \mathbf{E}\mathbf{e}, \qquad \mathbf{D}^T\boldsymbol{\sigma} + \mathbf{b} = \mathbf{0},} \tag{14.6}$$

in which $\mathbf{E} = \mathbf{E}^T$.

If the plate material is isotropic with elastic modulus E and Poisson's ratio ν, the moduli in the constitutive matrix $\mathbf{E}$ reduce to $E_{11} = E_{22} = E/(1-\nu^2)$, $E_{33} = \frac{1}{2}E/(1+\nu) = G$, $E_{12} = \nu E_{11}$ and $E_{13} = E_{23} = 0$. See also Exercise 14.1.

§14.3.2. Boundary Conditions

Boundary conditions prescribed on Γ may be of two types: displacement BC or force BC (also called stress BC or traction BC). To write down those conditions it is conceptually convenient to break up Γ into two subsets: Γ_u and Γ_t, over which displacements and force or stresses, respectively, are specified. See Figure 14.5.

Displacement boundary conditions are prescribed on Γ_u in the form

$$\boxed{\mathbf{u} = \hat{\mathbf{u}}.} \tag{14.7}$$

Here $\hat{\mathbf{u}}$ are prescribed displacements. Often $\hat{\mathbf{u}} = \mathbf{0}$. This happens in fixed portions of the boundary, as the ones illustrated in Figure 14.5.

Force boundary conditions (also called stress BCs and traction BCs in the literature) are specified on Γ_t. They take the form

$$\boxed{\boldsymbol{\sigma}_n = \hat{\mathbf{t}}.} \tag{14.8}$$

Here $\hat{\mathbf{t}}$ are prescribed surface tractions specified as a force per unit area (that is, not integrated through the thickness), and $\boldsymbol{\sigma}_n$ is the stress vector shown in Figure 14.5.

An alternative form of (14.8) uses the internal plate forces:

$$\boxed{\mathbf{p}_n = \hat{\mathbf{q}}.} \tag{14.9}$$

Here $\mathbf{p}_n = \boldsymbol{\sigma}_n h$ and $\hat{\mathbf{q}} = \hat{\mathbf{t}}\, h$. This form is used more often than (14.8) in structural design, particularly when the plate wall construction is nonhomogeneous.

The components of $\boldsymbol{\sigma}_n$ in Cartesian coordinates follow from Cauchy's stress transformation formula

$$\boldsymbol{\sigma}_n = \begin{bmatrix} \sigma_{xx} n_x + \sigma_{xy} n_y \\ \sigma_{xy} n_x + \sigma_{yy} n_y \end{bmatrix} = \begin{bmatrix} n_x & 0 & n_y \\ 0 & n_y & n_x \end{bmatrix} \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{14.10}$$

in which n_x and n_y denote the Cartesian components of the unit normal vector $\mathbf{n}^{(e)}$ (also called the direction cosines of the normal). Thus (14.8) splits into two scalar conditions: $\hat{t}_x = \sigma_{nx}$ and $\hat{t}_y = \sigma_{ny}$.

It is sometimes convenient to write the condition (14.8) in terms of normal n and tangential t directions:

$$\sigma_{nn} = \hat{t}_n, \qquad \sigma_{nt} = \hat{t}_t \tag{14.11}$$

in which $\sigma_{nn} = \sigma_{nx} n_x + \sigma_{ny} n_y$ and $\sigma_{nt} = -\sigma_{nx} n_y + \sigma_{ny} n_x$.

REMARK 14.3

The separation of Γ into Γ_u and Γ_t is useful for conciseness in the mathematical formulation, such as the energy integrals presented below. It does not exhaust, however, all BC possibilities. Frequently at points of Γ one specifies a displacement in one direction and a force (or stress) in the other. An example of these are roller and sliding conditions as well as lines of symmetry and antisymmetry. To cover these situations one needs either a generalization of the split, in which Γ_u and Γ_t are permitted to overlap, or to define another portion Γ_m for "mixed" conditions. Such generalizations will not be presented here, as they become unimportant once the FE discretization is done.

§14.3.3. Weak Forms versus Strong Form

We introduce now some further terminology from variational calculus. The Tonti diagram of Figure 14.4 is said to display the Strong Form of the governing equations because all relations are verified point by point. These relations, called *strong links*, are shown in the diagram with black lines.

A Weak Form is obtained by *relaxing* one or more strong links. Those are replaced by *weak links*, which enforce relations in an average or integral sense rather than point by point. The weak links are then provided by the variational formulation chosen for the problem. Because in general many variational forms of the same problem are possible, there are many possible Weak Forms. On the other hand the Strong Form is unique.

The Weak Form associated with the Total Potential Energy (TPE) variational form is illustrated in Figure 14.6. The internal equilibrium equations and stress BC become weak links, which are indicated by gray lines. These equations are given by the variational statement $\delta\Pi = 0$, where the TPE functional Π is given in the next subsection. The FEM displacement formulation discussed below is based on this particular Weak Form.

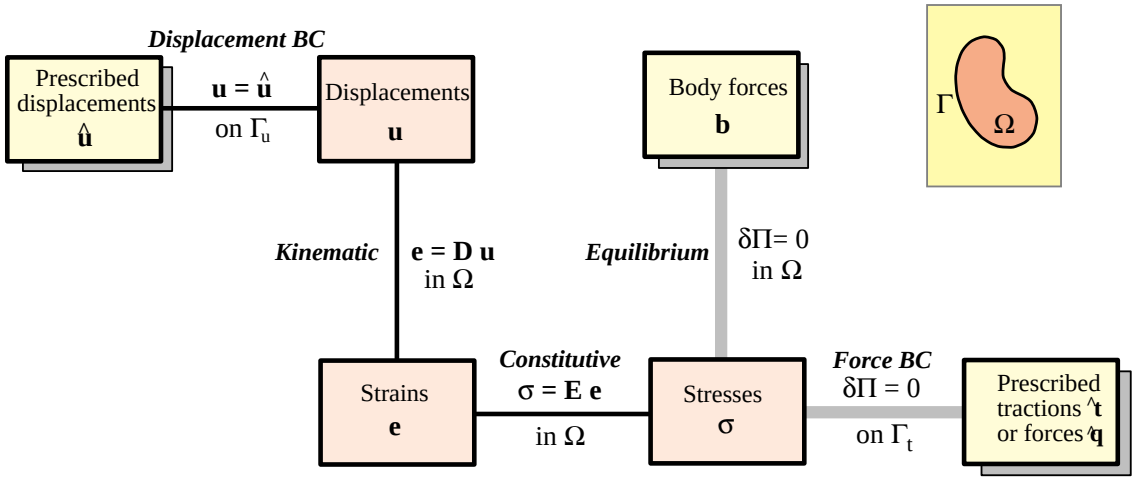


Figure 14.6. The TPE-based Weak Form of the plane stress equations of linear elastostatics. Weak links are marked with grey lines.

§14.3.4. Total Potential Energy

As usual the Total Potential Energy functional for the plane stress problem is given by

$$\boxed{\Pi = U - W.} \tag{14.12}$$

The internal energy is the elastic strain energy:

$$\boxed{U = \tfrac{1}{2}\int_\Omega h\,\boldsymbol{\sigma}^T\mathbf{e}\,d\Omega = \tfrac{1}{2}\int_\Omega h\,\mathbf{e}^T\mathbf{E}\mathbf{e}\,d\Omega.} \tag{14.13}$$

The derivation details are relegated to Exercise E14.5. The external energy is the sum of contributions from known interior and boundary forces:

$$\boxed{W = \int_\Omega h\,\mathbf{u}^T\mathbf{b}\,d\Omega + \int_{\Gamma_t} h\,\mathbf{u}^T\hat{\mathbf{t}}\,d\Gamma.} \tag{14.14}$$

Note that the boundary integral over Γ is taken only over Γ_t. That is, the portion of the boundary over which tractions or forces are specified.

§14.4. FINITE ELEMENT EQUATIONS

The necessary equations to apply the finite element method are collected here and expressed in matrix form. The domain of Figure 14.7(a) is discretized by a finite element mesh as illustrated in Figure 14.7(b). From this mesh we extract a generic element labeled (e) with $n \geq 3$ node points. In the subsequent derivation the number n is kept *arbitrary*. Therefore, the formulation is applicable to arbitrary two-dimensional elements, for example those sketched in Figure 14.8.

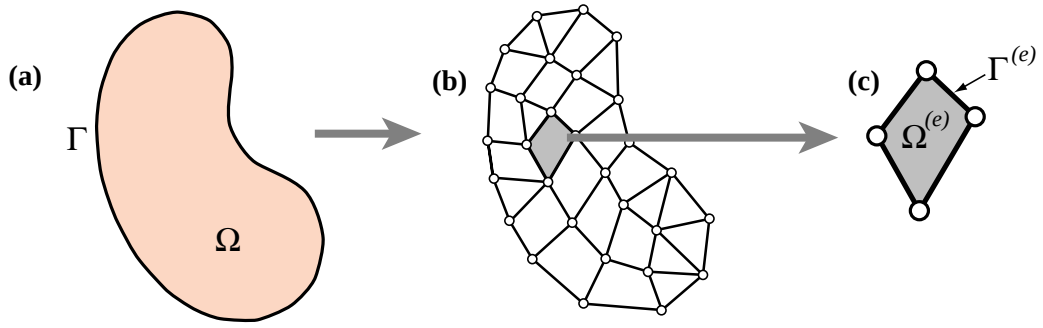


Figure 14.7. Finite element discretization and extraction of generic element.

Departing from previous practice in 1D elements, the element node points will be labelled 1 through n. These are called *local node numbers*. The element domain and boundary are denoted by $\Omega^{(e)}$ and $\Gamma^{(e)}$, respectively. The element has $2n$ degrees of freedom. These are collected in the element node displacement vector

$$\mathbf{u}^{(e)} = [\, u_{x1} \quad u_{y1} \quad u_{x2} \quad \ldots \quad u_{xn} \quad u_{yn} \,]^T \,. \tag{14.15}$$

§14.4.1. Displacement Interpolation

The displacement field $\mathbf{u}^{(e)}(x, y)$ over the element is interpolated from the node displacements. We shall assume that the same interpolation functions are used for both displacement components.[3] Thus

$$u_x(x, y) = \sum_{i=1}^{n} N_i^{(e)}(x, y)\, u_{xi}, \qquad u_y(x, y) = \sum_{i=1}^{n} N_i^{(e)}(x, y)\, u_{yi}, \tag{14.16}$$

where $N_i^{(e)}(x, y)$ are the element shape functions. In matrix form:

$$\boxed{\mathbf{u}(x, y) = \begin{bmatrix} u_x(x, y) \\ u_y(x, y) \end{bmatrix} = \begin{bmatrix} N_1^{(e)} & 0 & N_2^{(e)} & 0 & \ldots & N_n^{(e)} & 0 \\ 0 & N_1^{(e)} & 0 & N_2^{(e)} & \ldots & 0 & N_n^{(e)} \end{bmatrix} \mathbf{u}^{(e)} = \mathbf{N}^{(e)}\mathbf{u}^{(e)}.} \tag{14.17}$$

The minimum conditions on $N_i^{(e)}(x, y)$ is that it must take the value one at the i^{th} node and zero at all others, so that the interpolation (14.17) is correct at the nodes. Additional requirements on the shape functions are stated in later Chapters.

Differentiating the finite element displacement field yields the strain-displacement relations:

$$\mathbf{e}(x, y) = \begin{bmatrix} \dfrac{\partial N_1^{(e)}}{\partial x} & 0 & \dfrac{\partial N_2^{(e)}}{\partial x} & 0 & \ldots & \dfrac{\partial N_n^{(e)}}{\partial x} & 0 \\ 0 & \dfrac{\partial N_1^{(e)}}{\partial y} & 0 & \dfrac{\partial N_2^{(e)}}{\partial y} & \ldots & 0 & \dfrac{\partial N_n^{(e)}}{\partial y} \\ \dfrac{\partial N_1^{(e)}}{\partial y} & \dfrac{\partial N_1^{(e)}}{\partial x} & \dfrac{\partial N_2^{(e)}}{\partial y} & \dfrac{\partial N_2^{(e)}}{\partial x} & \ldots & \dfrac{\partial N_n^{(e)}}{\partial y} & \dfrac{\partial N_n^{(e)}}{\partial x} \end{bmatrix} \mathbf{u}^{(e)} = \mathbf{B}\,\mathbf{u}^{(e)}. \tag{14.18}$$

[3] This is the so called *element isotropy* condition, which is studied and justified in advanced FEM courses.

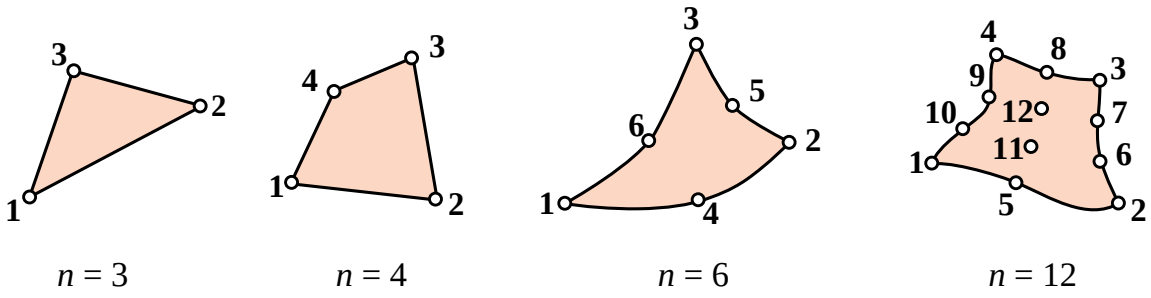


Figure 14.8. Example two-dimensional finite elements, characterized by their number of nodes n.

The strain-displacement matrix $\mathbf{B} = \mathbf{D}\mathbf{N}^{(e)}$ is $3 \times 2n$. The superscript (e) is omitted from it to reduce clutter. The stresses are given in terms of strains and displacements by $\boldsymbol{\sigma} = \mathbf{E}\,\mathbf{e} = \mathbf{E}\mathbf{B}\mathbf{u}^{(e)}$, which is assumed to hold at all points of the element.

§14.4.2. Element Energy

To obtain finite element stiffness equations, the variation of the TPE functional is decomposed into contributions from individual elements:

$$\boxed{\delta\Pi^{(e)} = \delta U^{(e)} - \delta W^{(e)} = 0.} \tag{14.19}$$

where

$$\boxed{U^{(e)} = \tfrac{1}{2}\int_{\Omega^{(e)}} h\,\boldsymbol{\sigma}^T\mathbf{e}\,d\Omega^{(e)} = \tfrac{1}{2}\int_{\Omega^{(e)}} h\,\mathbf{e}^T\mathbf{E}\mathbf{e}\,d\Omega^{(e)}} \tag{14.20}$$

and

$$\boxed{W^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{u}^T\mathbf{b}\,d\Omega^{(e)} + \int_{\Gamma^{(e)}} h\,\mathbf{u}^T\,\hat{\mathbf{t}}\,d\Gamma^{(e)}} \tag{14.21}$$

Note that in (14.21) $\Gamma_t^{(e)}$ has been taken equal to the complete boundary $\Gamma^{(e)}$ of the element. This is a consequence of the fact that displacement boundary conditions are applied *after* assembly, to a free-free structure. Consequently it does not harm to assume that all boundary conditions are of stress type insofar as forming the element equations.

§14.4.3. Element Stiffness Equations

Inserting the relations $\mathbf{u} = \mathbf{N}\mathbf{u}^{(e)}$, $\mathbf{e} = \mathbf{B}\mathbf{u}^{(e)}$ and $\boldsymbol{\sigma} = \mathbf{E}\mathbf{e}$ into $\Pi^{(e)}$ yields the quadratic form in the nodal displacements

$$\Pi^{(e)} = \tfrac{1}{2}\mathbf{u}^{(e)\,T}\mathbf{K}^{(e)}\mathbf{u}^{(e)} - \mathbf{u}^{(e)\,T}\mathbf{f}^{(e)}, \tag{14.22}$$

where the element stiffness matrix is

$$\boxed{\mathbf{K}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{B}^T\mathbf{E}\mathbf{B}\,d\Omega^{(e)},} \tag{14.23}$$

and the consistent element nodal force vector is

$$\boxed{\mathbf{f}^{(e)} = \int_{\Omega^{(e)}} h\,\mathbf{N}^T\mathbf{b}\,d\Omega^{(e)} + \int_{\Gamma^{(e)}} h\,\mathbf{N}^T\hat{\mathbf{t}}\,d\Gamma^{(e)}.} \tag{14.24}$$

In the second integral the matrix **N** is evaluated on the element boundary only.

The calculation of the entries of $\mathbf{K}^{(e)}$ and $\mathbf{f}^{(e)}$ for several elements of historical or practical interest is described in subsequent Chapters.

Homework Exercises for Chapter 14
The Plane Stress Problem

EXERCISE 14.1

[A:25] Suppose that the structural material is isotropic, with elastic modulus E and Poisson's ratio ν. The in-plane stress-strain relations for plane stress ($\sigma_{zz} = \sigma_{xz} = \sigma_{yz} = 0$) and plane strain ($e_{zz} = e_{xz} = e_{yz} = 0$) as given in any textbook on elasticity, are

$$\begin{aligned}
\text{plane stress:}\quad & \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}, \\
\text{plane strain:}\quad & \begin{bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \sigma_{xy} \end{bmatrix} = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{1-2\nu}{2(1-\nu)} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ 2e_{xy} \end{bmatrix}.
\end{aligned} \tag{E14.1}$$

Show that the constitutive matrix of plane strain can be formally obtained by replacing E by a fictitious modulus E^* and ν by a fictitious Poisson's ratio ν^* in the plane stress constitutive matrix and suppressing the stars. Find the expression of E^* and ν^* in terms of E and ν. This device permits "reusing" a plane stress FEM program to do plane strain, as long as the material is isotropic.

EXERCISE 14.2

[A:25] In the finite element formulation of near incompressible isotropic materials (as well as plasticity and viscoelasticity) it is convenient to use the so-called *Lamé constants* λ and μ instead of E and ν in the constitutive equations. Both λ and μ have the physical dimension of stress and are related to E and ν by

$$\lambda = \frac{\nu E}{(1+\nu)(1-2\nu)}, \quad \mu = G = \frac{E}{2(1+\nu)}. \tag{E14.2}$$

Conversely

$$E = \frac{\mu(3\lambda+2\mu)}{\lambda+\mu}, \quad \nu = \frac{\lambda}{2(\lambda+\mu)}. \tag{E14.3}$$

Substitute (E14.3) into (E14.1) to express the two stress-strain matrices in terms of λ and μ. Then split the stress-strain matrix $\mathbf{E}$ of plane strain as

$$\mathbf{E} = \mathbf{E}_\mu + \mathbf{E}_\lambda \tag{E14.4}$$

in which $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ contain only μ and λ, respectively, with $\mathbf{E}_\mu$ diagonal and $E_{\lambda 33} = 0$. This is the Lamé or $\{\lambda, \mu\}$ splitting of the plane strain constitutive equations, which leads to the so-called **B**-bar formulation of near-incompressible finite elements.[4] Express $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ also in terms of E and ν.

For the plane stress case perform a similar splitting in which where $\mathbf{E}_\lambda$ contains only $\bar{\lambda} = 2\lambda\mu/(\lambda+2\mu)$ with $E_{\lambda 33} = 0$, and $\mathbf{E}_\mu$ is a diagonal matrix function of μ and $\bar{\lambda}$.[5] Express $\mathbf{E}_\mu$ and $\mathbf{E}_\lambda$ also in terms of E and ν.

---

[4] Equation (E14.4) is sometimes referred to as the deviatoric+volumetric splitting of the stress-strain law, on account of its physical meaning in plane strain. That meaning is lost, however, for plane stress.

[5] For the physical significance of $\bar{\lambda}$ see I. Sokolnikoff, *The Mathematical Theory of Elasticity*, McGraw-Hill, 2nd ed., 1956, p. 254ff.

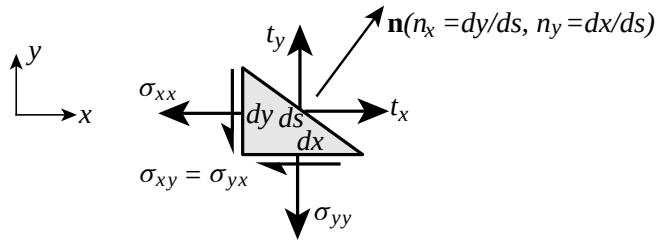


Figure E14.1. Geometry for deriving (14.10).

EXERCISE 14.3

[A:20] Derive the Cauchy stress-to-traction equations (14.10) using force equilibrium along x and y and the geometric relations shown in Figure E14.1. Hint: $t_x\, ds = \sigma_{xx}\, dy + \sigma_{xy}\, dx$, etc.

EXERCISE 14.4

[A:15] Include thermoelastic effects in the plane stress field equations, assuming a thermally isotropic material with coefficient of linear expansion α.

EXERCISE 14.5

[A:25=5+5+15] A plate is in linearly elastic plane stress. It is shown in courses in elasticity that the internal strain energy density stored per unit volume is

$$\mathcal{U} = \tfrac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + \sigma_{xy}e_{xy} + \sigma_{yx}e_{yx}) = \tfrac{1}{2}(\sigma_{xx}e_{xx} + \sigma_{yy}e_{yy} + 2\sigma_{xy}e_{xy}). \tag{E14.5}$$

(a) Show that (E14.5) can be written in terms of strains only as

$$\mathcal{U} = \tfrac{1}{2}\mathbf{e}^T\mathbf{E}\,\mathbf{e}, \tag{E14.6}$$

and hence justify (14.13).

(b) Show that (E14.5) can be written in terms of stresses only as

$$\mathcal{U} = \tfrac{1}{2}\boldsymbol{\sigma}^T\mathbf{C}\,\boldsymbol{\sigma}, \tag{E14.7}$$

where $\mathbf{C} = \mathbf{E}^{-1}$ is the elastic compliance (strain-stress) matrix.

(c) Suppose you want to write (E14.5) in terms of the extensional strains $\{e_{xx}, e_{yy}\}$ and of the shear stress $\sigma_{xy} = \sigma_{yx}$. This is known as a mixed representation. Show that

$$\mathcal{U} = \tfrac{1}{2}\begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}^T \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{12} & A_{22} & A_{23} \\ -A_{13} & -A_{23} & A_{33} \end{bmatrix} \begin{bmatrix} e_{xx} \\ e_{yy} \\ \sigma_{xy} \end{bmatrix}, \tag{E14.8}$$

and explain how the entries A_{ij} can be calculated[6] in terms of the elastic moduli E_{ij}.

[6] The process of computing **A** is an instance of "partial inversion" of matrix **E**. See Remark 11.3.