Application and Implementation of Finite Element Methods

J. E. AKIN

Application and Implementation of Finite Element Methods

J. E. AKIN
Department of Engineering Science and Mechanics,
The University of Tennessee,
Knoxville, Tennessee, USA

1982

ACADEMIC PRESS
A Subsidiary of Harcourt Brace Jovanovich, Publishers
London New York
Paris San Diego San Francisco São Paulo
Sydney Tokyo Toronto

ACADEMIC PRESS INC. (LONDON) LTD
24–28 Oval Road
London NW1

US edition published by
ACADEMIC PRESS INC.
111 Fifth Avenue
New York, New York 10003

British Library Cataloguing in Publication Data

Akin, J. E.
Application and implementation of finite element methods.
1. Finite element method—Data processing
I. Title
515.3'53 TA347.F5

ISBN 0-12-047650-9
LCCCN 81-69597

Printed in Great Britain by Page Bros (Norwich) Ltd

Preface

The finite element method has now become a well-established branch of computational mathematics and the theoretical foundations have been presented in several texts. However, the actual application of finite element procedures requires extensive programming effort. The present text has been developed to illustrate typical computational alogrithms and their applications. Whilst the theoretical discussions have been limited to the minimum required to introduce the topic, most of the computational procedures are discussed in detail. Many universities follow an introductory finite element course, with a course on the related computational procedures. This text should be well suited for such a course.

Numerous programs are presented and discussed. A particular controlling program and data structure have been included for completeness so that specific applications can be examined in detail. Emphasis has been placed on the use of isoparametric elements and numerical integration. The discussion of the programs for isoparametric elements, Chapter 5, and their example applications, Chapter 11, should clarify this important topic. Since practical problems often involve a large amount of data the subject of mesh generation is also examined. A very limited discussion of time integration procedures has been included. Also included is an Appendix which describes some subroutines of secondary interest, which are mentioned in Chapters 2 and 6, together with various sample applications and the input formats for MODEL.

Figures referred to in each Section are included at the end of the Section, followed by the Tables.

The text reflects the many studies and conversations on finite elements in which I was able to participate during a period in which I was on leave from the University of Tennessee. These studies were conducted at the University of Texas at Austin, Brunel University, and the California Institute of Technology. The support of a UK SRC Senior Visiting Fellow Grant and a US NSF Professional Development Grant is gratefully acknowledged. I would also like to acknowledge the support and encouragement of J. T. Oden, J. R. Whiteman, T. J. R. Hughes, E. B. Becker, and W. C. T. Stoddart.

Copies of the MODEL software are available either from the author, or from the Institute of Computational Mathematics at Brunel University.

University of Tennessee J. E. Akin
January 1982

Program Notation

```
AD = VECTOR CONTAINING FLOATING POINT VARIABLES
AJ = JACOBIAN MATRIX
AJINV = INVERSE JACOBIAN MATRIX

C = ELEMENT COLUMN MATRIX
CB = BOUNDARY SEGMENT COLUMN MATRIX
CC = COLUMN MATRIX OF SYSTEM EQUATIONS
CEQ = CONSTRAINT EQS COEFFS ARRAY
COORD = SPATIAL COORDINATES OF A SELECTED SET OF NODES
CP = PENALTY CONSTRAINT COLUMN MATRIX
CUTOFF IS SPECIFIED NUMBER FOR CUTTING OFF ITERATIONS

D = NODAL PARAMETERS ASSOCIATED WITH A GIVEN ELEMENT
DD = SYSTEM LIST OF NODAL PARAMETERS
DDOLD = SYSTEM LIST OF NODAL DOF FROM LAST ITERATION
DELTA = LOCAL DERIVATIVES OF INTERPOLATION FUNCTIONS H

ELPROP = ELEMENT ARRAY OF FLOATING POINT PROPERTIES

FLTEL = SYSTEM STORAGE OF FLOATING PT ELEMENT PROP
FLTMIS = SYSTEM STORAGE OF FLOATING PT MISC. PROP
FLTNP = SYSTEM STORAGE OF FLOATING PT NODAL PROP
FLUX = SPATIAL COMPONENTS OF SPECIFIED BOUNDARY FLUX

GLOBAL = GLOBAL DERIV.S OF INTERPOLATION FUNCTIONS H

H = INTERPOLATION FUNCTIONS FOR AN ELEMENT

ID = VECTOR CONTAINING FIXED POINT ARRAYS
IBC = NODAL POINT BOUNDARY RESTRAINT INDICATOR ARRAY
INDEX = SYSTEM DEGREE OF FREEDOM NUMBERS ARRAY
IF INRHS.NE.0 INITIAL VALUES OF CC ARE INPUT
IF IPTEST.GT.0 SOME PROPERTIES ARE DEFINED
IP1 = NUMBER OF ROWS IN ARRAY FLTNP
IP2 = NUMBER OF ROWS IN ARRAY NPFIX
IP3 = NUMBER OF ROWS IN ARRAY FLTEL
IP4 = NUMBER OF ROWS IN ARRAY LPFIX
IP5 = NUMBER OF ROWS IN ARRAY PRTLPT
ISAY = NO. OF USER REMARKS TO BE I/O

JP1 = NUMBER OF COLUMNS IN ARRAYS NPFIX AND NPROP
JP2 = NUMBER OF COLUMNS IN ARRAYS LPFIX AND LPROP
JP3 = NUMBER OF COLUMNS IN ARRAY MISFIX

KFIXED = ALLOCATED SIZE OF ARRAY ID
KFLOAT = ALLOCATED SIZE OF ARRAY AD
KODES = LIST OF DOF RESTRAINT INDICATORS AT A NODE
KP1 = NUMBER OF COL IN ARRAYS FLTNP & PRTLPT & PTPROP
KP2 = NUMBER OF COLUMNS IN ARRAYS FLTEL AND ELPROP
KP3 = NUMBER OF COLUMNS IN ARRAY FLTMIS
K1-K5 = NO. OF COLUMNS OF FLOATING PT CONSTRAINT DATA

LBN = NUMBER OF NODES ON AN ELEMENT BOUNDARY SEGMENT
IF LEMWRT = 0 LIST NODAL PARAMETERS BY ELEMENTS
IF LHOMO=1 ELEMENT PROPERTIES ARE HOMOGENEOUS
LNODE = THE N ELEMENT INCIDENCES OF THE ELEMENT
LPFIX = SYSTEM STORAGE ARRAY FOR FIXED PT ELEMENT PROP
LPROP = ARRAY OF FIXED POINT ELEMENT PROPERTIES
IF LPTEST.GT.0 ELEMENT PROPERTIES ARE DEFINED
```

```
M = NO. OF SYSTEM NODES
MAXBAN = MAX. HALF BANDWIDTH OF SYSTEM EQUATIONS
IF MAXTIM.GT.0 CALCULATE CPU TIMES OF MAJOR SEGMENTS
MAXACT = NO ACTIVE CONSTRAINT TYPES (<=MAXTYP)
MAXTYP = MAX NODAL CONSTRAINT TYPE (=3 NOW)
MISCFL = NO. MISC. FLOATING POINT SYSTEM PROPERTIES
MISCFX = NO. MISC. FIXED  POINT SYSTEM PROPERTIES
MISFIX = SYSTEM ARRAY OF MISC. FIXED POINT PROPERTIES
MTOTAL = REQUIRED SIZE OF ARRAY AD
M1 TO MNEXT = POINTERS FOR FLOATING POINT ARRAYS

N = NUMBER OF NODES PER ELEMENT
NCURVE = NO. CONTOUR CURVES CALCULATED PER PARAMETER
NDFREE = TOTAL NUMBER OF SYSTEM DEGREES OF FREEDOM
NDXC = CONSTRAINT EQS DOF NUMBERS ARRAY
NE = NUMBER OF ELEMENTS IN SYSTEM
NELFRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
NG = NUMBER OF NODAL PARAMETERS (DOF) PER NODE
IF NHOMO=1 NODAL SYSTEM PROPERTIES ARE HOMOGENEOUS
NITER = NO. OF ITERATIONS TO BE RUN
NLPFIX = NO.   FIXED  POINT ELEMENT PROPERTIES
NLPFLO = NO. FLOATING POINT ELEMENT PROPERTIES
NNPFIX = NO. FIXED POINT NODAL PROPERTIES
NNPFLO = NO. FLOATING POINT NODAL PROPERTIES
NOCOEF = NO COEFF IN SYSTEM SQ MATRIX
NODES = ELEMENT INCIDENCES OF ALL ELEMENTS
NOTHER = TOTAL NO. OF BOUNDARY RESTRAINTS .GT. TYPE1
NPROP = NODAL ARRAY OF FIXED POINT PROPERTIES
IF NPTWRT = 0 LIST NODAL PARAMETERS BY NODES
NRANGE = ARRAY CONTAINING NODE NO.S OF EXTREME VALUES
NREQ = NO. OF CONSTRAINT EQS. OF EACH TYPE
NRES = NO. OF CONSTRAINT FLAGS OF EACH TYPE
NSEG = NO OF ELEM BOUNDARY SEGMENTS WITH GIVEN FLUX
NSPACE = DIMENSION OF SPACE
NTAPE1 = UNIT FOR POST SOLUTION MATRICES STORAGE
NTAPE2,3,4 = OPTIONAL UNITS FOR USER (USED WHEN > 0)
NTOTAL = REQUIRED SIZE OF ARRAY ID
IF NULCOL.NE.0  ELEMENT COLUMN MATRIX IS ALWAYS ZERO
NUMCE = NUMBER OF CONSTRAINT EQS
N1 TO NNEXT = POINTERS FOR FIXED POINT ARRAYS

PRTLPT = FLOATING PT PROP ARRAY OF ELEMENT'S NODES
PTPROP = NODAL ARRAY OF FLOATING PT PROPERTIES

RANGE: 1-MAXIMUM VALUE, 2-MINIMUM VALUE OF DOF

S = ELEMENT SQUARE MATRIX
SB = BOUNDARY SEGMENT SQUARE MATRIX
SS = 'SQUARE'  TRIX OF SYSTEM EQUATIONS

TIME = ARRAY STORING CPU TIMES FOR VARIOUS SEGMENTS
TITLE = PROBLEM TITLE

X = SPATIAL COORDINATES OF ALL NODES IN THE SYSTEM
XPT = SPATIAL COORDINATES OF A CONTOUR POINT

.....................................................
```

Contents

1 Finite element concepts

1.1 Introduction

The finite element method has become an important and practical numerical analysis tool. It has found application in almost all areas of engineering and applied mathematics. The literature on finite element methods is extensive and rapidly increasing. Extensive bibliographies are available [59, 76] but even these are incomplete and are rapidly becoming outdated. Numerous texts are available which present the theory of various finite element procedures. Most of these relegate programming considerations to a secondary, or lower, level. One exception is the text by Hinton and Owen [42]. The present work takes a similar position in that it aims to provide a complete overview of typical programming considerations while covering only the minimum theoretical aspects. Of course, the theory behind the illustrated implementation procedures and selected applications is discussed.

This chapter begins the introduction of various *building block* programs for typical use in finite element analysis. These modular programs may be utilized in numerous fields of study. Specific examples of the application of the finite element method will be covered in later chapters.

1.2 Foundation of finite element procedures

From the mathematical point of view the finite element method is based on integral formulations. By way of comparison the older finite difference methods are usually based on differential formulations. Finite element models of various problems have been formulated from simple physical

intuition and from mathematical principles. Historically, the use of physical intuition led to several early practical models. However, today there is increased emphasis on the now well established mathematical foundations of the procedure [10, 28, 61].

The mathematical rigor of the finite element method was lacking at first, but it is now a very active area of research. Modern finite element integral formulations are obtained by two different procedures: variational formulations and weighted residual formulations. The following sections will briefly review the common procedures for establishing finite element models. It is indeed fortunate that all of these techniques use the same *bookkeeping* operations to generate the final assembly of algebraic equations that must be solved for the unknown nodal parameters.

The earliest mathematical formulations for finite element models were based on variational techniques. Variational techniques still are very important in developing elements and in solving practical problems. This is especially true in the areas of structural mechanics and stress analysis. Modern analysis in these areas has come to rely on finite element techniques almost exclusively. Variational models usually involve finding the nodal parameters that yield a stationary (maximum or minimum) value of a specific integral relation known as a functional. In most cases it is possible to assign a physical meaning to the integral being extremized. For example, in solid mechanics the integral may represent potential energy, whereas in a fluid mechanics problem it may correspond to the rate of entropy production. Many physical problems have variational formulations that result in quadratic forms. These in turn yield algebraic equations for the system which are symmetric and positive definite. Another important practical advantage of variational formulations is that they often have error bound theorems associated with them. Numerous examples of variational formulations for finite element models can be found by examining the many texts available on the theory of variational calculus. Several applications of this type will be illustrated in the later chapters.

It is well known that the solution that yields a stationary value of the integral functional and satisfies the boundary conditions is equivalent to the solution of an associated differential equation, known as the Euler equation. If the functional is known, then it is relatively easy to find the corresponding Euler equation. Most engineering and physical problems are initially defined in terms of a differential equation. The finite element method requires an integral formulation. Thus, one must search for the functional whose Euler equation corresponds to the given differential equation (and boundary conditions). Unfortunately, this is generally a difficult, or impossible task. Therefore, there is increasing emphasis on the various weighted residual techniques that can generate an integral for-

mulation directly from the original differential equations. Both the differential equation and integral form are defined in physical coordinates, say (x, y, z).

As a simple one-dimensional example of an integral statement, consider the functional

$$I = \tfrac{1}{2}\int_0^L [K(\mathrm{d}t/\mathrm{d}x)^2 + Ht^2]\,\mathrm{d}x,$$

which will be considered in later applications. Minimizing this functional is equivalent to satisfying the differential equation

$$K\mathrm{d}^2t/\mathrm{d}x^2 - Ht = 0.$$

In addition the functional satisfies the natural conditions of $\mathrm{d}t/\mathrm{d}x = 0$ at an end where the essential boundary condition, $t = t_0$, is not applied.

The generation of finite element models by the utilization of weighted residual techniques is a relatively recent development. However, these methods are increasingly important in the solution of differential equations and other non-structural applications. The weighted residual method starts with the governing differential equation

$$L(\phi) = Q$$

and avoids the often tedious search for a mathematically equivalent variational statement. Generally one assumes an approximate solution, say ϕ^*, and substitutes this solution into the differential equation. Since the assumption is approximate, this operation defines a residual error term in the differential equation

$$L(\phi^*) - Q = R.$$

Although one cannot force the residual term to vanish, it is possible to force a weighted integral, over the solution domain, of the residual to vanish. That is, the integral over the solution domain, Ω, of the product of the residual term and some weighting function is set equal to zero, so that

$$I = \int_\Omega RW\mathrm{d}\Omega = 0.$$

Substituting interpolation functions for the approximate solution, ϕ^*, and the weighting function, W, results in a set of algebraic equations that can be solved for the unknown coefficients in the approximate solution. The choice of weighting function defines the type of weighted residual technique being utilized. To obtain the Galerkin criterion one selects

$$W = \phi^*,$$

while for a least squares criterion

$$W = \partial R/\partial \phi^*$$

gives the desired result. Similarly, selecting the Dirac delta function gives a point collocation procedure; i.e.

$$W = \delta.$$

Obviously, other choices of W are available and lead to alternate weighted residual procedures such as the subdomain procedure. The first two procedures seem to be most popular for finite element methods. Use of integration by parts with the Galerkin procedure usually reduces the continuity requirements of the approximating functions. If a variational procedure exists, the Galerkin criterion will lead to the same algebraic approximation. Thus it often offers optimal error estimates for the finite element solution.

For both variational and weighted residual formulations the following restrictions are now generally accepted as means for establishing convergence of the finite element model as the mesh refinement increases [87]:

1. (A necessary criterion) The element interpolation functions must be capable of modelling any constant values of the dependent variable or its derivatives, to the order present in the defining integral statement, in the limit as the element size decreases.
2. (A sufficient criterion) The element shape functions should be chosen so that at element interfaces the dependent variable and its derivatives, of one order less than those occurring in the defining integral statement, are continuous.

1.3 General finite element analysis procedure

1.3.1 Introduction

In the finite element method, the boundary and interior of the continuum (or more generally the solution domain) is subdivided (see Fig. 1.1—illustrations are arranged together at the end of the section in which they are mentioned) by imaginary lines (or surfaces) into a finite number of discrete sized subregions or *finite elements*. A discrete number of *nodal points* are established with the imaginary mesh that divides the region. These nodal points can lie anywhere along, or inside, the subdividing mesh lines, but they are usually located at intersecting mesh lines (or surfaces). Usually, the elements have straight boundaries and thus some geometric

approximations will be introduced in the geometric idealization if the actual region of interest has curvilinear boundaries.

The nodal points are assigned identifying integer numbers (*node numbers*) beginning with unity and ranging to some maximum value, say M. Similarly, each element is assigned an identifying integer number. These *element numbers* also begin with unity and extend to a maximum value, say NE. As will be discussed later, the assignment of the nodal numbers and element numbers can have a significant effect on the solution time and storage requirements. The analyst assigns a number of (generalized) *degrees of freedom*, (dof), say NG, to each and every node. These are the (unknown) nodal parameters that have been chosen by the analyst to govern the formulation of the problem of interest. Common *nodal parameters* are pressure, velocity components, displacement components, displacement gradients, etc. The nodal parameters do not have to have a physical meaning, although they usually do. It will be assumed herein that each node in the system has the same number (NG) of nodal parameters. This is the usual case, but it is not necessary. A typical node, Fig. 1.1, will usually be associated with more than one element. The domains of influence of a typical node and typical element are also shown in Fig. 1.1. A typical element will have a number, say N, of nodal points associated with it located on or within its boundaries. It is assumed herein that every element has the same number (N) of nodes per element. This is the usual situation, but again it is not necessary in general.

This idealization procedure defines the total number of degrees of freedom associated with a typical node and a typical element. Obviously, the number of degrees of freedom in the system, say NDFREE, is the product of the number of nodes and the number of parameters per node, i.e. NDFREE = M* NG. Similarly, the number of degrees of freedom per element, say NELFRE, is defined by NELFRE = N* NG.

Recall that the total number of degrees of freedom of the system corresponds to the total number of nodal parameters. In general the system degree of freedom number, say NDF, associated with parameter number J at system node number I is defined (by induction) as:

$$\mathrm{NDF} = \mathrm{NG}^*(\mathrm{I} - 1) + \mathrm{J}, \tag{1.1}$$

where $1 \leq \mathrm{I} \leq \mathrm{M}$ and $1 \leq \mathrm{J} \leq \mathrm{NG}$ so that $1 \leq \mathrm{NDF} \leq \mathrm{NDFREE}$. This elementary equation forms the basis of the program "bookkeeping" method and thus is very important and should be clearly understood. Equation (1.1) is illustrated for a series of one-dimensional elements in Fig. 1.2, where a system with four line elements, five nodes and six degrees of freedom (dof) per node is illustrated (i.e. M = 5, NE = 4, NG = 6, N = 2). There are a total of thirty dof in the system. We wish to determine the

dof number of the fifth parameter (J = 5) at system node number four (I = 4). Equation (1.1) shows that the required result is NDF = 6(4 − 1) + 5 = 23 for the system dof number. For element three this corresponds to local dof number 11 while for element four it is local dof number 5. Therefore we note that contributions to system equation number 23 comes from parts of two different element equation sets.

In addition to the above constants it is necessary to define the dimension of the space, say NSPACE, that is associated with the problem. As will be pointed out as the discussion proceeds, these quantities can be used to calculate the size of the storage requirements for the matrices to be generated in the analysis. The actual programs that read the problem data will be discussed in a later section.

Data must be supplied to define the spatial coordinates of each nodal point. This array of data, say X, will have the dimensions of M*NSPACE. It is common to associate an integer with each nodal point. The purpose of the code is to indicate which, if any, of the nodal parameters at the node have boundary constraints specified. This vector of data, say IBC, contains M integer coefficients. To accomplish this nodal boundary condition coding process recall that there are NG parameters per node. Thus, one can define an integer code, IBC, (right justified) to consist of NG digits. Let the *i*th digit be a single digit indicator corresponding to the *i*th parameter at that node. If the indicator equals *j* where $0 \leq j \leq 9$ then this is defined to mean that the *i*th parameter has a boundary constraint of type *j*. If the single digit indicator is zero, this means that there is no boundary constraint on that parameter. As will be discussed later, the present program allow several common types of nodal parameter boundary constraints. Figure 1.3 illustrates a set of boundary condition codes for a typical set of nodes with six parameters per node (NG = 6). This code is also considered in the example in Section 7.3 and all the applications.

An important concept is that of *element connectivity*, ie the list of global node numbers that are attached to an element. The element connectivity data defines the topology of the mesh. Thus for each element it is necessary to input (in some consistent order) the N node numbers that are associated with that particular element. This array of data, say NODES, has the dimensions of NE*N. The list of node numbers connected to a particular element is usually referred to as the *element incident list* for that element. The identification of these data is important to the use of Eqn. (1.1).

1.3.2 Approximation of element behaviour and equations

It is assumed that the variable(s) of interest, and perhaps its derivatives, can be uniquely specified throughout the solution domain by the nodal

parameters associated with the nodal points of the system. These nodal parameters will be the unknown parameters of the problem. It is assumed that the parameters at a particular node influence only the values of the quantity of interest within the elements that are connected to that particular node. Next, an interpolation function is assumed for the purpose of relating the quantity of interest within the element in terms of the values of the nodal parameters at the nodes that are connected to that particular element. Figure 1.4 illustrates a common interpolation associated with that particular element. Figure 1.4 illustrates a common interpolation function and its constituent parts defined in terms of the nodal coordinates (x_i, y_i), the element area A^e, the spatial location (x, y) and the nodal parameters T_i, T_j and T_k.

After the element behaviour has been assumed, the remaining fundamental problem is to establish the element matrices $\mathbf{S}^e$ and $\mathbf{C}^e$. Generally, they involve substituting the interpolation functions into the governing integral form. Historically these matrices have been called the *element stiffness matrix* and *load vector,* respectively. Although these matrices can sometimes be developed from physical intuition, they are usually formulated by the minimization of a functional or by the method of weighted residuals. These procedures are described in several texts, and will be illustrated in detail in later chapters.

Almost all element matrix definitions involve some type of defining properties, or coefficients. A few finite element problems require the definition of properties at the nodal points. For example, in a stress analysis one may wish to define variable thickness elements by specifying the thickness of the material at each node point. The finite element method is very well suited to the solution of non-homogeneous problems; therefore, most finite element programs also require the analyst to assign certain properties to each element. It is usually desirable to have any data that are common to every element (or every node) stored as miscellaneous system data. The analyst must decide which data are required and how best to input and recover them.

1.3.3 Assembly and solution of equations

Once the element equations have been established the contribution of each element is added to form the *system equations.* The programming details of the assembly procedure will be discussed in Section 7.2. The system equations resulting from a finite element analysis will usually be of the form

$$\mathbf{S}\,\mathbf{D} = \mathbf{C}, \tag{1.2}$$

where the square matrix **S** is NDFREE*NDFREE in size and the vectors **D** and **C** contain NDFREE coefficients each. The vector **D** will contain the unknown nodal parameters and the matrices **S** and **C** are obtained by assembling (as described later) the element matrices, $\mathbf{S}^e$ and $\mathbf{C}^e$, respectively. Matrices $\mathbf{S}^e$ and $\mathbf{C}^e$ are NELFRE*NELFRE and NELFRE*1 in size. In the majority of problems $\mathbf{S}^e$, and thus **S**, will be symmetric. Also, the system, **S**, is usually banded about the diagonal. It will be assumed herein that the system equations are banded and symmetric. Thus, the half-bandwidth, including the diagonal, is an important quantity to be considered in any finite element analysis. From consideration of the technique used to approximate the element behaviour, it is known that the half-bandwidth, say IBW, of the system equations due to a typical element "e" is defined by

$$\text{IBW} = \text{NG}^* (\text{NDIFF} + 1), \tag{1.3}$$

where NDIFF is the absolute value of the maximum difference in node numbers of the nodes connected to the element. Equation (1.3) will be derived later. The maximum half-bandwidth, say MAXBAN, of the system is the largest value of IBW that exists in the system. That is,

$$\text{MAXBAN} = \text{IBW}_{\text{maximum}}. \tag{1.4}$$

The quantity MAXBAN is one of the important quantities which govern the storage requirements and solution time of the system equations. Thus, although the assignment of node numbers is arbitrary, the analyst, in practice, should try to minimize the maximum difference in node numbers (and the bandwidth) associated with a typical element. The assembly process is illustrated in Fig. 1.5 for a four element mesh consisting of three-node triangles with one parameter per node. The top of the figure shows an assembly of the system **S** and **C** matrices that is coded to denote the sources of the contributing terms but not their values. A hatched area in these indicates a term that was added in from an element that has the hash code. For example, the load vector term C(6) is seen to be the sum of contributions from elements 2 and 3 which are hatched with horizontal (−) and oblique (/) lines, respectively. By way of comparison the term C(1) has only a contribution from element 2.

In closing, it should be noted that several efficient finite element codes do not utilize a banded matrix solution technique. Instead, they may employ a frontal solution or a sparse matrix solution. These important topics will be covered in some detail.

After the system equations have been assembled, it is necessary to apply the boundary constraints before solving for the unknown nodal parameters. The most common types of nodal parameter boundary constraints are