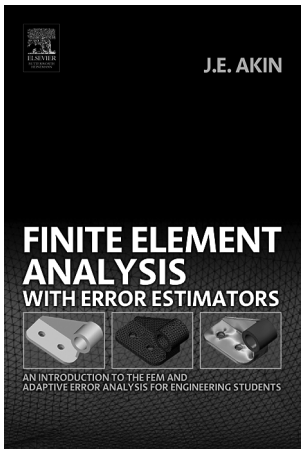

J.E. AKIN
FINITE ELEMENT ANALYSIS
WITH ERROR ESTIMATORS
AN INTRODUCTION TO THE FEM AND ADAPTIVE ERROR ANALYSIS FOR ENGINEERING STUDENTS

# Finite Element Analysis with Error Estimators

## An Introduction to the FEM and Adaptive Error Analysis for Engineering Students

# Finite Element Analysis with Error Estimators

## An Introduction to the FEM and Adaptive Error Analysis for Engineering Students

J. E. Akin


AMSTERDAM • BOSTON • HEIDELBERG • LONDON • NEW YORK • OXFORD
PARIS • SAN DIEGO • SAN FRANCISCO • SINGAPORE • SYDNEY • TOKYO

Elsevier Butterworth-Heinemann
Linacre House, Jordan Hill, Oxford OX2 8DP
30 Corporate Drive, Burlington, MA 01803

First published 2005



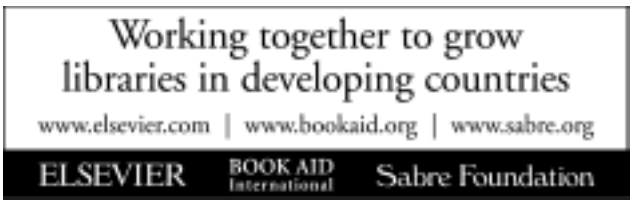
**British Library Cataloguing in Publication Data**
A catalogue record for this book is available from the British Library

**Library of Congress Cataloguing in Publication Data**
A catalogue record for this book is available from the Library of Congress

ISBN 0 7506 6722 2



Printed and bound in Great Britain



# Contents

* Denotes sections or chapters that can be omitted for a first reading or shorter course.

# Preface

There are many good texts on the application of finite element analysis techniques. Most do not address the concept and implementation of error estimation. Now that computers are so powerful there is no reason not to carry out a re-analysis until the error levels reach the point that the user is satisfied. Having an error estimation is critical to automating the adaptation of the finite element analysis process. Today several commercial programs include automatic adaptation, based on an error analysis. The user of such programs should have a clear concept of the theory and limitations of such tools. Thus, this text includes the basic finite element theory and its mathematical foundations, the error estimation processes, and the associated computational procedures, as well as several example applications.

This book is primarily intended for advanced undergraduate engineering students and beginning graduate students. The text contains more material than could be covered in a single quarter or semester course. Therefore, a number of chapters or sections that could be omitted in a first course have been marked with an asterisk ($*$). Most of the subject matter deals with linear heat transfer and elementary stress analysis.

The future of finite element analysis will probably heavily involve adaptive analysis methods. One should have a course in Functional Analysis to best understand those techniques. Most undergraduate curriculums do not contain such courses. Therefore, a chapter on mathematical preliminaries is included.

All the Fortran 95 source programs for the general finite element library (called MODEL), and the corresponding application and supporting data file can be downloaded from the World Wide Web (for non-commercial use only). They can be found at the Elsevier site http://www.books.elsevier.com/companions/. The same is true of a large library of small Matlab plotting scripts that display the input and output results shown in the text.

I would like to thank many current and former students at Rice University for their constructive criticisms and comments during the evolution of this book. Special thanks go to Prof. R. L. Taylor, of the University of California at Berkeley for his many detailed and constructive suggestions. Mr. Don Schroder helped with the preparation of a large part of the manuscript. Finally, this book would not have been completed without the support and patience of my wife Kimberly.

Ed Akin
Houston, Texas
2005

## Features of the text and accompanying resources

**End of chapter exercises**
Each chapter ends with a range of exercises that are suitable for homework and assignment work, as well as for private study.

Worked solutions to the exercises are freely available to teachers who adopt or recommend the text to their students. For details on accessing this material please visit http://books.elsevier.com/manuals and follow the registration instructions on screen.

**Fortran 95 source programs**
Source programs for the general finite element library, and the corresponding application and supporting data file can be freely downloaded from the accompanying website. Go to http://books.elsevier.com/companions and follow the instructions on screen. This material is presented for non-commercial use only.

**Matlab plotting scripts library**
A library of Matlab plotting scripts that display the input and output results shown in the text are also available for free download from the accompanying website. Go to http://books.elsevier.com/companions and follow the instructions on screen. This material is presented for non-commercial use only.

# Notation

The symbols most commonly used throughout the book are defined below. When appearing in the text matrices, tensors, and vectors are identified by boldface type.

**Mathematical symbols**

| | |
|---|---|
| $(\hat{.})$ | Based on element gradient |
| $(.^{*})$ | Based on nodally continuous gradient |
| $\{.\}$ | Column vector, n by 1 |
| $\lvert . \rvert$ | Determinant of a matrix |
| $\boldsymbol{\Delta}^T$ | Divergence operator |
| $\varnothing$ | Empty set |
| $\boldsymbol{\nabla}$ | Gradient operator |
| $\in$ | In |
| $\bigcap$ | Intersection |
| $[.]^{-1}$ | Inverse of a square matrix |
| $\square$ | Non-dimensional parametric space |
| $\lVert . \rVert$ | Norm of a matrix or vector |
| $]. , . [$ | Open one-dimensional domain |
| $\lfloor \ \rfloor^T \lfloor \ \rfloor$ | Outer product square matrix, m by m |
| $_{,(.)}$ | Partial differentiation with respect to ( . ) |
| $\boldsymbol{\partial}_G, \boldsymbol{\partial}_\Omega$ | Partial derivatives in global Cartesian space |
| $\boldsymbol{\partial}_L, \boldsymbol{\partial}_\square$ | Partial derivatives in local parametric space |
| $\propto$ | Proportional to |
| $[\,.\,]$ | Rectangular, m by n, or square matrix |
| $\lfloor\,.\,\rfloor$ | Row vector, 1 by m |
| $\subset$ | Subset |
| $[\,.\,]^T$ | Transpose of a matrix |
| $\bigcup$ | Union |

**Latin Symbols**

| | |
|---|---|
| $A$ | Area |
| $\mathbf{a}$ | Acceleration vector |
| $a, b, c$ | Natural coordinates on −1 to +1 |
| $(.)^b$ | Relating to a boundary domain |
| $\mathbf{B}$ | Differential operator acting on interpolation matrix **H** or **N** |
| $\mathbf{b}$ | Differential operator acting on global interpolation matrix **h** |
| $C^n$ | Field continuity of degree $n$ |
| $\mathbf{C}$ | System source vector |
| $\mathbf{C}^b$ | Source vector from a boundary segment |
| $\mathbf{C}^e$ | Source vector from an element |
| $\mathbf{D}$ | System degrees of freedom vector |
| $D$ | Differential operator. |
| $\mathbf{D}^b$ | Boundary segment degrees of freedom vector |
| $\mathbf{D}^e$ | Element degrees of freedom vector |
| $\mathbf{d}$ | Cartesian gradient of **H** |
| $\mathbf{d}_x$ | First row of **d**, etc. for y, z |
| $dof$ | Degree(s) of freedom |
| $E$ | Modulus of elasticity of a material |
| $\mathbf{E}$ | Constitutive law (stress-strain) matrix |
| $e$ | Error |
| $(.)^e$ | Relating to an element domain |
| $\mathbf{F}$ | Resultant force vector |
| $G$ | Shear modulus of a material |
| $\mathbf{G}$ | Geometry interpolation row matrix (usually $\mathbf{G} = \mathbf{H}$) |
| $\mathbf{H}^b$ | Boundary interpolation row matrix for a scalar |
| $\mathbf{H}^e$ | Element interpolation row matrix for a scalar |
| $h$ | Characteristic length. Convection coefficient |
| $\mathbf{h}$ | Global interpolation matrix |
| $I^e$, $\mathbf{I}^e$ | Integral of a scalar or matrix, respectively, on an element |
| $\mathbf{I}$ | Identity matrix |
| $\mathbf{J}$ | Jacobian matrix of a geometric transformation |
| $\mathbf{K}$ | Stiffness matrix |
| $k$ | Thermal conductivity of a material, or spring stiffness |
| $\mathbf{L}$ | Differential operator |
| $L$ | Length |
| $L_k$ | Barycentric coordinates, $\sum L_k = 1$ |
| $\mathbf{M}$ | Mass matrix of the system |
| $\mathbf{m}^e$ | Mass matrix, or thermal capacity matrix of an element |
| $m$ | Mass |
| $\mathbf{N}$ | Interpolation matrix for generalized degrees of freedom (often $\mathbf{N} = \mathbf{H}$) |
| $\mathbf{n}$ | Unit normal vector |
| $n_a$ | Number of adjacent elements, *NEIGH_L* |
| $n_b$ | Number of boundary segments, *N_MIXED + N_SEG* |
| $n_c$ | Number of constraint equations, *N_CEQ* |

| | |
|---|---|
| $n_d$ | Number of system degrees of freedom ($n_m \times n_g$), *N_D_FRE* |
| $n_e$ | Number of elements in the system, *N_ELEMS* |
| $n_f$ | Maximum number of flux components, *N_G_FLUX* |
| $n_g$ | Number of generalized dof per node, *N_G_DOF* |
| $n_h$ | Number of scalar interpolations in **H**, *LT_FREE* |
| $n_i$ | Number of element equation index terms ($n_n \times n_g$), *LT_FREE* |
| $n_l$ | Number of elements in a patch, *L_IN_PATCH* |
| $n_m$ | Maximum node number in the system, *MAX_NP* |
| $n_n$ | Maximum number of nodes per element, *NOD_PER_EL* |
| $n_o$ | Number of mixed or Robin BC segments, *N_MIXED* |
| $n_p$ | Dimension of the parametric space, *N_PARM* |
| $n_q$ | Number of quadrature points, *N_QP* |
| $n_r$ | Number of rows in the **B** matrix, *N_R_B* |
| $n_s$ | Dimension of the physical space, *N_SPACE* |
| $n_t$ | Number of different element types, *N_L_TYPE* |
| $n_v$ | Number of vector interpolations in **V**, *LT_FREE* |
| $n_x$ | Number of element geometry definition nodes, *N_GEOM* |
| $\mathbf{P}$ | Polynomial row matrix. Reaction vector |
| $p$ | Pressure |
| $\mathbf{Q}$ | Source per unit volume |
| $\mathbf{Q}^e$ | Source per unit volume at element node points |
| $q$ | Source per unit length |
| $q_n$ | Heat flux normal to boundary ($\mathbf{q}_n = q_n\,\mathbf{n}$) |
| $\mathbf{q}$ | Heat flux vector at a point |
| $\mathbf{R}$ | Matrix of position vectors, $\mathbf{R} = [\mathbf{x}\,\mathbf{y}\,\mathbf{z}]$ |
| $R$ | Residual error in $\Omega^e$ |
| $r, s, t$ | Unit coordinates on 0 to 1 |
| $\mathbf{S}$ | Square matrix of the system |
| $\mathbf{S}^b$ | Square matrix from a boundary segment |
| $\mathbf{S}^e$ | Square matrix from an element |
| $t$ | Thickness, time |
| $\mathbf{T}$ | Transformation matrix, or boundary traction matrix |
| $U$ | Strain energy |
| $\mathbf{u}$ | Displacement vector. Velocity vector |
| $u, v, w$ | Components of displacement vector |
| $V$ | Volume |
| $\mathbf{v}$ | Velocity vector |
| $W$ | Mechanical work |
| $x, y, z$ | Cartesian coordinates |
| $\mathbf{X}$ | Body force vector |
| $\mathbf{x}$ | Vector of x-coordinates |
| $\mathbf{x}^e$ | Vector of x-coordinates of the element nodes |
| $\mathbf{y}$ | Vector of y-coordinates |
| $\mathbf{z}$ | Vector of z-coordinates |

**Greek symbols**

| | |
|---|---|
| $\alpha$ | Coefficient of thermal expansion |
| $\boldsymbol{\beta}$ | Boolean gather matrix |
| $\boldsymbol{\beta}^T$ | Boolean scatter matrix |
| $\sum_e \boldsymbol{\beta}^{eT}\, \mathbf{C}^e$ | Column vector element assembly process |
| $\sum_e \boldsymbol{\beta}^{eT}\, \mathbf{S}^e\, \boldsymbol{\beta}^e$ | Square matrix element assembly process |
| $\boldsymbol{\Gamma}$ | Boundary of a domain, $\Omega$ |
| $\boldsymbol{\Gamma}^b$ | Segment of the boundary $\boldsymbol{\Gamma}$ |
| $\boldsymbol{\Gamma}^e$ | Boundary of an element domain, $\Omega^e$ |
| $\gamma$ | Weight per unit volume |
| $\boldsymbol{\Delta}$ | Local derivatives of the interpolation matrix **H** or **N** |
| $\boldsymbol{\delta}$ | Element or boundary segment dof. |
| $\boldsymbol{\varepsilon}$ | Strain or gradient |
| $\zeta$ | Refinement parameter |
| $\eta$ | Allowed percentage error |
| $\theta$ | Temperature, or angle |
| $\Theta$ | Effectivity index |
| $\lambda$ | Direction cosine wrt x. Lame' constant. |
| $\mu$ | Direction cosine wrt y. Lame' constant. |
| $\nu$ | Poisson's ratio of a material. Direction cosine wrt z. |
| $\boldsymbol{\Pi}$ | Total potential energy, $\boldsymbol{\Pi} = U - W$ |
| $\pi$ | Mathematical constant 3.14159... |
| $\rho$ | Mass density of a material |
| $\boldsymbol{\rho}$ | Position vector to a point, $\boldsymbol{\rho} = [x, y, z]$ |
| $\boldsymbol{\sigma}$ | Flux or stress |
| $\boldsymbol{\sigma}^*$ | Smoothed flux or stress approximation |
| $\hat{\boldsymbol{\sigma}}$ | Finite element flux or stress approximation |
| $\tau$ | Stabilization parameter |
| $\boldsymbol{\tau}$ | Shear stress |
| $\Phi$ | System degrees of freedom vector |
| $\Phi_k$ | $k$-th unknown |
| $\phi$ | Scalar unknown. Velocity potential |
| $\psi$ | Stream function |
| $\omega$ | Angular velocity |
| $\boldsymbol{\Omega}$ | Domain |
| $\boldsymbol{\Omega}^e$ | Element domain |

**Selected program notation** (Array sizes follow in parentheses.)

| | |
|---|---|
| AJ | Jacobian matrix: (N_SPACE, N_SPACE) |
| AVE | Average quantities at a system node: (N_R_B + 2, MAX_NP) |
| B | Gradient versus dof matrix: (N_R_B, LT_FREE) |
| C | Element column matrix: (LT_FREE) |
| CC | Column matrix of system equations: (N_D_FRE) |
| COORD | Coordinates of all nodes on an element: (LT_N, N_SPACE) |
| C_B | Boundary segment column matrix: (LT_FREE) |
| D | Nodal parameters associated with an element: (LT_FREE) |
| DD | System list of nodal parameters: (N_D_FRE) |
| DGH | Global derivatives of scalar functions $\mathbf{H}$ : (N_SPACE, LT_N) |
| DGV | Global derivatives of vector functions $\mathbf{V}$ : (N_SPACE, LT_FREE) |
| DLG | Local derivatives of geometry functions $\mathbf{G}$ : (LT_PARM, LT_GEOM) |
| DLH | Local derivatives of scalar functions $\mathbf{H}$ : (LT_PARM, LT_N) |
| E | Constitutive matrix: (N_R_B, N_R_B) |
| EL_M | Element mass matrix: (LT_FREE, LT_FREE) |
| FLUX_LT | Flux at element nodes from a SCP: (SCP_FIT, LT_N |
| G | Interpolation functions for geometry: (LT_GEOM) |
| GLOBAL | Global derivatives of scalar interpolation functions $\mathbf{H}$ |
| H | Interpolation functions for an element scalar: (LT_N) |
| H_INTG | Integral of scalar interpolation functions $\mathbf{H}$ : (LT_N) |
| H_QP | Interpolation for $\mathbf{H}$ at quadrature point: (LT_N, LT_QP) |
| INDEX | System degree of freedom numbers array: (LT_FREE) |
| L_B_N | Maximum number of nodes on an element boundary segment |
| LT | Element type number |
| LT_FREE | Number of degrees of freedom per element |
| LT_GEOM | Number of geometric nodes per element |
| LT_N | Maximum number of nodes for element type |
| LT_PARM | Dimension of parametric space for element type |
| LT_QP | Number of quadrature points for element type |
| LT_SHAP | Current element type shape flag number |
| L_B_N | Number of nodes on an element boundary segment |
| L_SHAPE | Shape: 0=Point 1=Line 2=Triangle 3=Quadrilateral 4=Hexahedron 5=Tetrahedron etc. |
| L_TYPE | Type number array of all elements: (L_S_TOT) |
| MAT_FLO | Number of real material properties |
| MAX_NP | Number of system nodes |
| MISC_FL | Number of miscellaneous floating point (real) system properties |
| MISC_FX | Number of miscellaneous fixed point (integer) system properties |
| M_B_N | Number of nodes on a mixed boundagy condition segment |
| NODES | Node incidences of all elements: (L_S_TOT, NOD_PER_EL) |
| NOD_PER_EL | Maximum number of nodes per element |
| N_BS_FIX | Number of boundary segment integer properties |
| N_BS_FLO | Number of boundary segment real properties |
| N_CEQ | Number of system constraint equations |
| N_D_FLUX | Maximum number of flux segment dof = L_B_N * N_G_DOF |

| | |
|---|---|
| N_D_FRE | Total number of system degrees of freedom |
| N_ELEMS | Number of elements in the system |
| N_EL_FRE | Maximum number of degrees of freedom per element |
| N_GEOM | Maximum number of element geometry nodes |
| N_G_DOF | Number of generalized parameters (dof) per node |
| N_G_FLUX | Number of flux components per segment node |
| N_LP_FIX | Number of integer element properties |
| N_LP_FLO | Number of floating point (real) element properties |
| N_MAT | Number of materail types |
| N_MX_FIX | Number of fixed point (integer) mixed segment properties |
| N_MX_FLO | Number of floating point (real) mixed segment properties |
| N_NP_FIX | Number of fixed point (integer) nodal properties |
| N_NP_FLO | Number of floating point (real) nodal properties |
| N_PARM | Dimension of parametric space |
| N_PATCH | Number of SCP patches = MAX_NP or N_ELEMS |
| N_QP | Maximum number of element quadrature points |
| N_R_B | Number of rows in $\mathbf{B}$ and $\mathbf{E}$ matrices |
| N_SEG | Number of element boundary segments with given flux |
| N_SPACE | Dimension of space |
| PATCH_FIT | Local patch flux values at its nodes: (SCP_N, SCP_FIT) |
| PT | Quadrature coordinates: (LT_PARM, LT_QP) |
| S | Element square matrix: (LT_FREE, LT_FREE) |
| SCP_COUNTS | Number of patches used for each nodal averages: (MAX_NP) |
| SCP_FIT | Number of terms being fit in a patch, N_R_B usually |
| SCP_GEOM | Number of patch geometry nodes |
| SCP_H | Interpolation functions for patch, usually is $\mathbf{H}$ (SCP_N) |
| SCP_LT | Patch type number |
| SCP_N | Number of nodes per patch |
| SCP_PARM | Number of parametric spaces for patch |
| SCP_QP | Number of quadrature points needed in a SCP patch |
| SIGMA_HAT | Flux components at a point in original element: (SCP_FIT) |
| SIGMA_SCP | Flux components at a point in smoothed SCP: (SCP_FIT) |
| SS | Square matrix of system equations: (N_D_FREE, N_D_FREE) |
| STRAIN | Strain or gradient vector: (N_R_B + 2) |
| STRAIN_0 | Initial strain or gradient vector, if any: (N_R_B) |
| STRESS | Stress vector at a point: (N_R_B + 2) |
| S_B | Boundary segment square matrix, if any: (LT_FREE, LT_FREE) |
| THIS_EL | Current element number |
| THIS_LT | Current element type number |
| THIS_STEP | Current time step number |
| TIME | Current time in dynamic or transient solution |
| V | Interpolation functions for vectors: (LT_FREE) |
| WT | Quadrature weights: (LT_QP) |
| X | Coordinates of all system nodes: (MAX_NP, N_SPACE) |
| XYZ | Spatial coordinates at a point: (N_SPACE) |

# Chapter 1

# Introduction

## 1.1 Finite element methods

The goal of this text is to introduce finite element methods from a rather broad perspective. We will consider the basic theory of finite element methods as utilized as an engineering tool. Likewise, example engineering applications will be presented to illustrate practical concepts of heat transfer, stress analysis, and other fields. Today the subject of error analysis for adaptivity of finite element methods has reached the point that it is both economical and reliable and should be considered in an engineering analysis. Finally, we will consider in some detail the typical computational procedures required to apply modern finite element analysis, and the associated error analysis. In this chapter we will begin with an overview of the finite element method. We close it with consideration of modern programming approaches and a discussion of how the software provided differs from the author's previous implementations of finite element computational procedures.

In modern engineering analysis it is rare to find a project that does not require some type of finite element analysis (FEA). The practical advantages of FEA in stress analysis and structural dynamics have made it the accepted tool for the last two decades. It is also heavily employed in thermal analysis, especially for thermal stress analysis.

Clearly, the greatest advantage of FEA is its ability to handle truly arbitrary geometry. Probably its next most important features are the ability to deal with general boundary conditions and to include nonhomogeneous and anisotropic materials. These features alone mean that we can treat systems of arbitrary shape that are made up of numerous different material regions. Each material could have constant properties or the properties could vary with spatial location. To these very desirable features we can add a large amount of freedom in prescribing the loading conditions and in the post-processing of items such as the stresses and strains. For elliptical boundary value problems the FEA procedures offer significant computational and storage efficiencies that further enhance its use. That class of problems include stress analysis, heat conduction, electrical fields, magnetic fields, ideal fluid flow, etc. FEA also gives us an important solution technique for other problem classes such as the nonlinear Navier-Stokes equations for fluid dynamics, and for plasticity in nonlinear solids.

Here we will show what FEA has to offer and illustrate some of its theoretical formulations and practical applications. A design engineer should study finite element methods in more detail than we can consider here. It is still an active area of research. The current trends are toward the use of error estimators and automatic adaptive FEA procedures that give the maximum accuracy for the minimum computational cost. This is also closely tied to shape modification and optimization procedures.

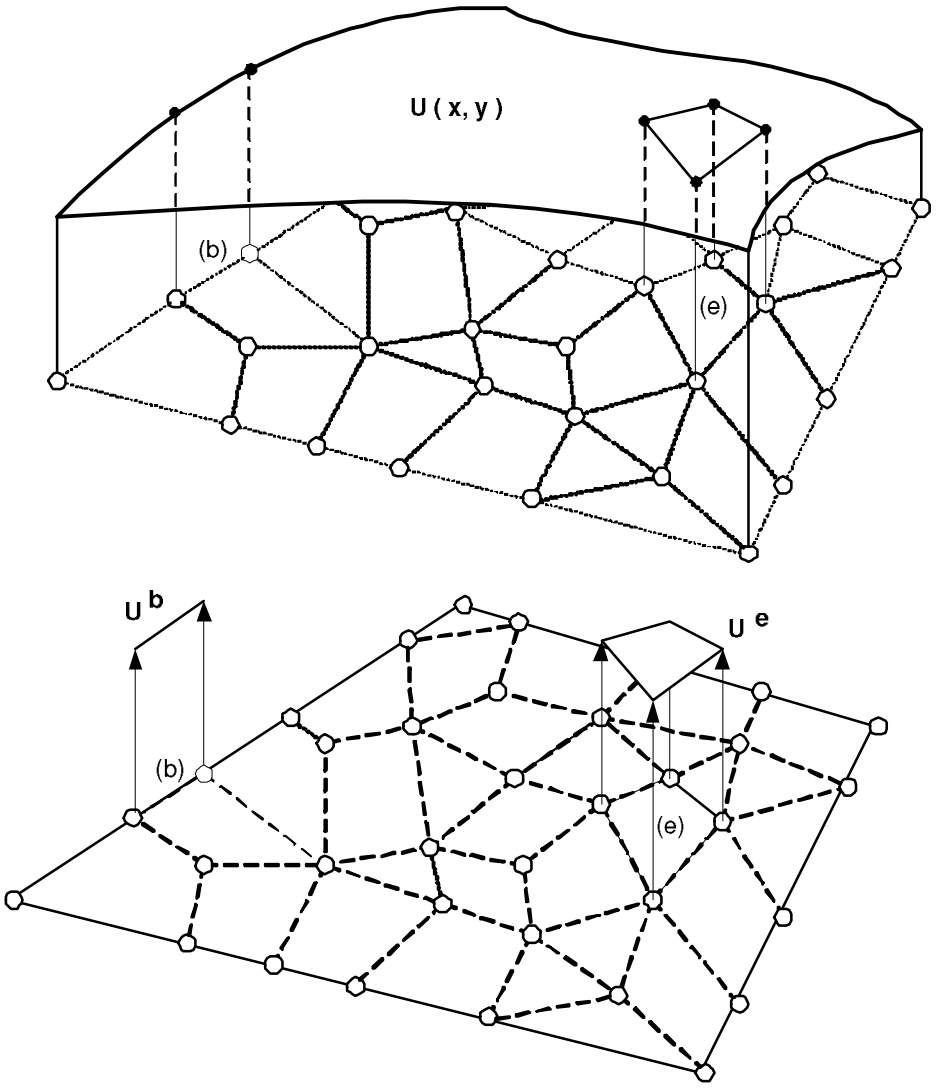



**Figure 1.1** *Piecewise approximation of a scalar function*

## 1.2 Capabilities of FEA

There are many commercial and public-domain finite element systems that are available today. To summarize the typical capabilities, several of the most widely used software systems have been compared to identify what they have in common. Often we find about 90 percent of the options are available in all the systems. Some offer very specialized capabilities such as aeroelastic flutter or hydroelastic lubrication. The mainstream capabilities to be listed here are found to be included in the majority of the commercial systems. The newer adaptive systems may have fewer options installed but they are rapidly adding features common to those given above. Most of these systems are available on engineering workstations and personal computers as well as mainframes and supercomputers. The extent of the usefulness of an FEA system is directly related to the extent of its element library. The typical elements found within a single system usually include membrane, solid, and axisymmetric elements that offer linear, quadratic, and cubic approximations with a fixed number of unknowns per node. The new hierarchical elements have relatively few basic shapes but they do offer a potentially large number of unknowns per node (more than 80). Thus, the actual effective element library size is extremely large.

In the finite element method, the boundary and interior of the region are subdivided by lines (or surfaces) into a finite number of discrete sized subregions or finite elements. A number of nodal points are established with the mesh. The size of an element is usually associated with a reference length denoted by $h$. It, for example, may be the diameter of the smallest sphere that can enclose the element. These nodal points can lie anywhere along, or inside, the subdividing mesh, but they are usually located at intersecting mesh lines (or surfaces). The elements may have straight boundaries and thus, some geometric approximations will be introduced in the geometric idealization if the actual region of interest has curvilinear boundaries. These concepts are graphically represented in Fig. 1.1.

The nodal points and elements are assigned identifying integer numbers beginning with unity and ranging to some maximum value. The assignment of the nodal numbers and element numbers will have a significant effect on the solution time and storage requirements. The analyst assigns a number of generalized degrees of freedom 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 displacement components, temperatures, and velocity components. The nodal parameters do not have to have a physical meaning, although they usually do. For example, the hierarchical elements typically use the derivatives up to order six as the midside nodal parameters. This idealization procedure defines the total number of degrees of freedom associated with a typical node, a typical element, and the total system. Data must be supplied to define the spatial coordinates of each nodal point. It is common to associate some code to each node to indicate which, if any, of the parameters at the node have boundary constraints specified. In the new adaptive systems the number of nodes, elements, and parameters per node usually all change with each new iteration.

Another important concept is that of *element connectivity*, (or topology) i.e., the list of global node numbers that are attached to an element. The element connectivity data defines the topology of the (initial) mesh, which is used, in turn, to assemble the system

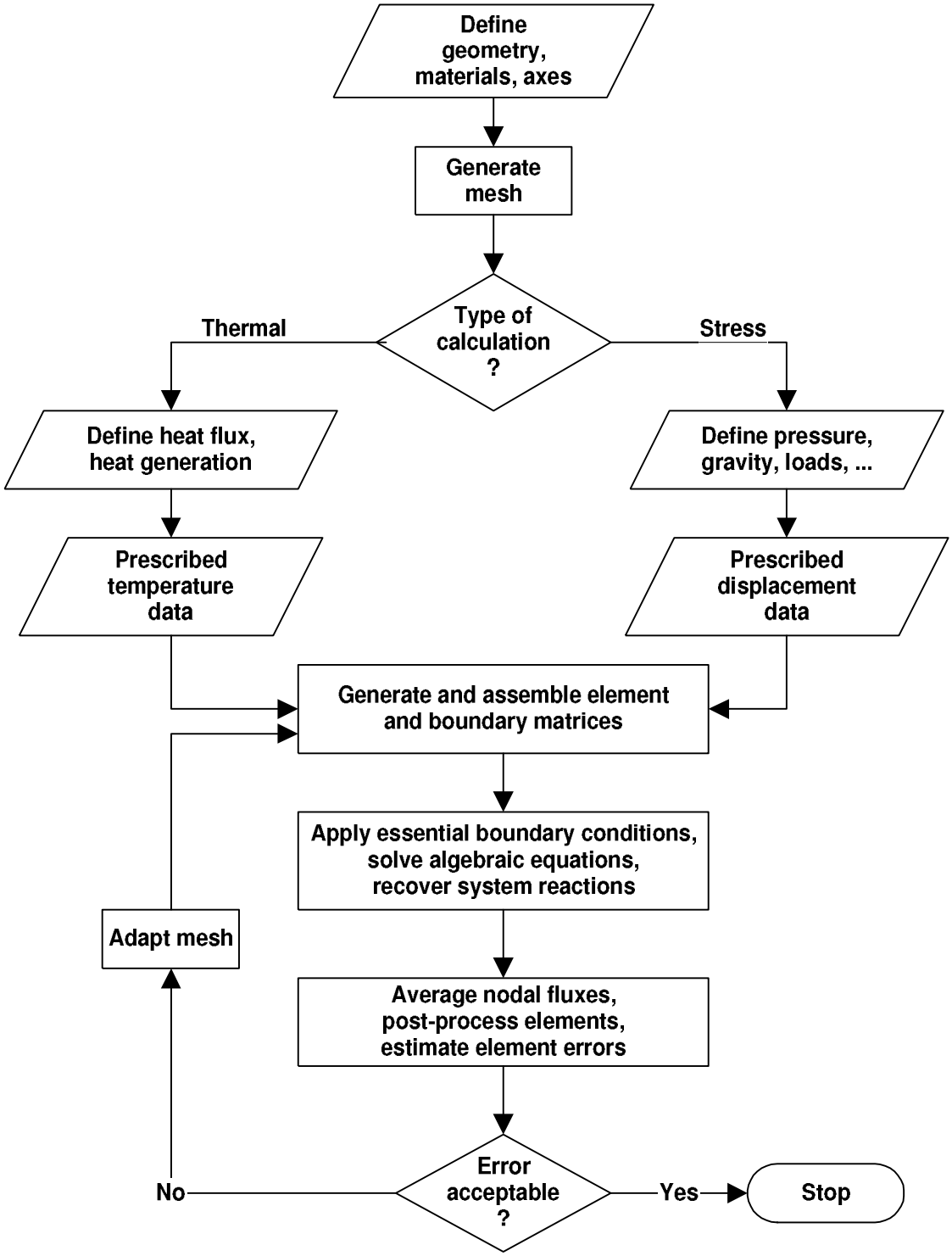



**Figure 1.2** *Typical stages in a finite element analysis*

algebraic equations. Thus, for each element it is necessary to input, in some consistent order, the node numbers that are associated with that particular element. The list of node numbers connected to a particular element is usually referred to as the element incident list for that element. We usually assign a material code, or properties, to each element.

Finite element analysis can require very large amounts of input data. Thus, most FEA systems offer the user significant data generation or supplemental capabilities. The common data generation and validation options include the generation and/or replication of coordinate systems, node locations, element connectivity, loading sets, restraint conditions, etc. The verification of such extensive amounts of input and generated data is greatly enhanced by the use of computer graphics.

In the adaptive methods we must also compute the error indicators, error estimators, and various energy norms. All these quantities are usually output at 1 to 27 points in each of thousands of elements. The most commonly needed information in an engineering analysis is the state of temperatures, or displacements and stresses. Thus, almost every system offers linear static stress analysis capabilities, and linear thermal analysis capabilities for conduction and convection that are often needed to provide temperature distributions for thermal stress analysis. Usually the same mesh geometry is used for the temperature analysis and the thermal stress analysis. Of course, some studies require information on the natural frequencies of vibration or the response to dynamic forces or the effect of frequency driven excitations. Thus, dynamic analysis options are usually available. The efficient utilization of materials often requires us to employ nonlinear material properties and/or nonlinear equations. Such resources require a more experienced and sophisticated user. The usual nonlinear stress analysis features in large commercial FEA systems include buckling, creep, large deflections, and plasticity. Those advanced methods will not be considered here.

There are certain features of finite element systems which are so important from a practical point of view that, essentially, we cannot get along without them. Basically we

**Table 1.1** *Typical unknown variables in finite element analysis*

| Application | Primary | Associated | Secondary |
|---|---|---|---|
| Stress analysis | Displacement,<br>Rotation | Force,<br>Moment | Stress,<br>Failure criterion<br>Error estimates |
| Heat transfer | Temperature | Flux | Interior flux<br>Error estimates |
| Potential flow | Potential function | Normal velocity | Interior velocity<br>Error estimates |
| Navier-Stokes | Velocity | Pressure | Error estimates |
| Eigen-problem | Eigenvalues | Eigenvectors | Error estimates |

**Table 1.2** *Typical given variables and corresponding reactions*

| Application | Given | Reaction |
|---|---|---|
| Stress analysis | Displacement | Force |
| | Rotation | Moment |
| | Force | Displacement |
| | Couple | Rotation |
| Heat transfer | Temperature | Heat flux |
| | Heat flux | Temperature |
| Potential flow | Potential | Normal velocity |
| | Normal velocity | Potential |
| Navier-Stokes | Velocity | Force |

have the ability to handle completely arbitrary geometries, which is essential to practical engineering analysis. Almost all the structural analysis, whether static, dynamic, linear or nonlinear, is done by finite element techniques on large problems. The other abilities provide a lot of flexibility in specifying loading and restraints (support capabilities). Typically, we will have several different materials at different arbitrary locations within an object and we automatically have the capability to handle these nonhomogeneous materials. Just as importantly, the boundary conditions that attach one material to another are usually automatic, and we don't have to do anything to describe them unless it is possible for gaps to open between materials. Most important, or practical, engineering components are made up of more than one material, and we need an easy way to handle that. What takes place less often is the fact that we have *anisotropic materials* (one whose properties vary with direction, instead of being the same in all directions). There is a great wealth of materials that have this behavior, although at the undergraduate level, anisotropic materials are rarely mentioned. Many materials, such as reinforced concrete, plywood, any filament-wound material, and composite materials, are essentially anisotropic. Likewise, for heat-transfer problems, we will have thermal conductivities that are directionally dependent and, therefore, we would have to enter two or three thermal conductivities that indicate how this material is directionally dependent. These advantages mean that for practical use finite element analysis is very important to us. The biggest disadvantage of the finite element method is that it has so much power that large amounts of data and computation will be required.

All real objects are three-dimensional but several common special cases have been defined that allow two-dimensional studies to provide useful insight. The most common examples in solid mechanics are the states of *plane stress* (covered in undergraduate mechanics of materials) and *plane strain*, the *axisymmetric solid* model, the *thin-plate* model, and the *thin-shell* model. The latter is defined in terms of two parametric surface coordinates even though the shell exists in three dimensions. The *thin beam* can be thought of as a degenerate case of the thin-plate model. Even though today's solid modelers can generate three-dimensional meshes relatively easily one should learn to

approach such problems carefully. A well planned series of two-dimensional approximations can provide important insight into planning a good three-dimensional model. They also provide good 'ball-park' checks on the three-dimensional answers. Of course, the use of basic handbook calculations in estimating the answer before approaching an FEA system is also highly recommended.

The typical unknown variables in a finite element analysis are listed in Table 1.1 and a list of related action-reaction variables are cited in Table 1.2. Figure 1.2 outlines as a flow chart the major steps needed for either a thermal analysis or stress analysis. Note that these segments are very similar. One of the benefits of developing a finite element approach is that most of the changes related to a new field of application occur at the element level and usually represent less than 5 percent of the total coding.

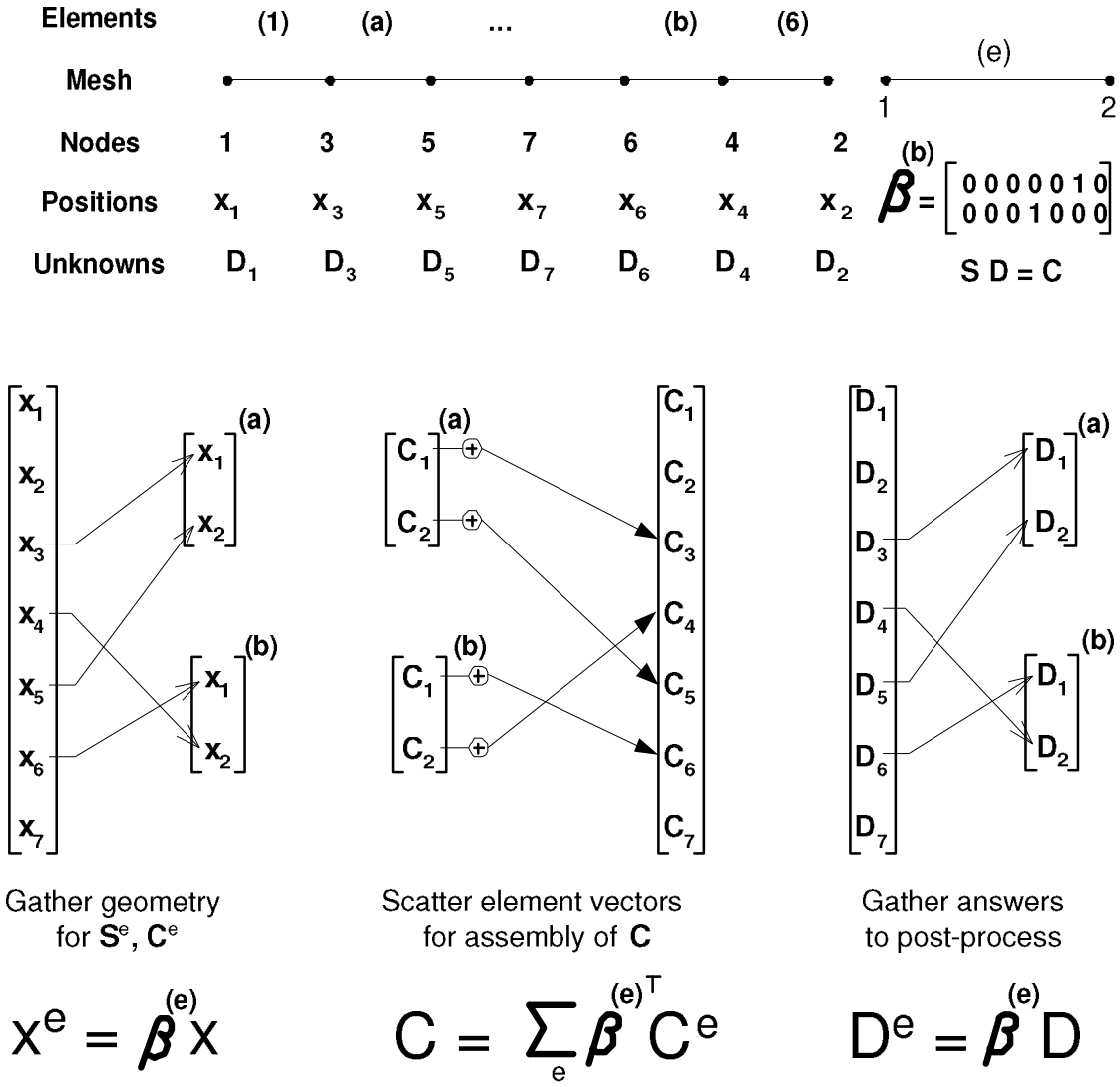



**Figure 1.3** *Gather and scatter concepts for finite elements*

## 1.3 Outline of finite element procedures

From the mathematical point of view the finite element method is an integral formulation. Modern finite element integral formulations are usually obtained by either of two different procedures: *weighted residual* or *variational formulations*. The following sections briefly outline the common procedures for establishing finite element models. It is fortunate that all these techniques use the same bookkeeping operations to generate the final assembly of algebraic equations that must be solved for the unknowns.

The generation of finite element models by the utilization of weighted residual techniques is increasingly important in the solution of differential equations for non-structural applications. The weighted residual method starts with the governing differential equation to be satisfied in a domain $\Omega$:

$$L(\phi) = Q, \tag{1.1}$$

where $L$ denotes a differential operator acting on the primary unknown, $\phi$, and $Q$ is a source term. Generally we assume an approximate solution, say $\phi^*$, for the spatial distribution of the unknown, say

$$\phi(x) \approx \phi^* = \sum_i^n h_i(x)\, \Phi_i^*, \tag{1.2}$$

where the $h_i(x)$ are spatial distributions associated with the coefficient $\Phi_i^*$. That assumption leads to a corresponding assumption for the spatial gradient of the assumed behavior. Next we substitute these assumptions on spatial distributions into the differential equation. Since the assumption is approximate, this operation defines a residual error term, $R$, in the differential equation

$$L(\phi^*) - Q = R \neq 0. \tag{1.3}$$

Although we 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 of the product of the residual term and some weighting function is set equal to zero, so that

$$I_i = \int_\Omega R\, w_i\, d\Omega = 0 \tag{1.4}$$

leads to the same number of equations as there are unknown $\Phi_i^*$ values. Most of the time we will find it very useful to employ integration by parts on this governing integral.

Substituting an assumed spatial behavior for the approximate solution, $\phi^*$, and the weighting function, $w$, results in a set of algebraic equations that can be solved for the unknown nodal coefficients in the approximate solution. This is because the unknown coefficients can be pulled out of the spatial integrals involved in the assembly process.

The choice of weighting function defines the type of weighted residual technique being utilized. The Galerkin criterion selects

$$w_i = h_i(x), \tag{1.5}$$

to make the residual error orthogonal to the approximate solution. Use of integration by parts with the Galerkin procedure (i.e., the Divergence Theorem) reduces the continuity requirements of the approximating functions. If a Euler variational procedure exists, the Galerkin criterion will lead to the same element matrices.

A spatial interpolation, or blending 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 connected to that particular element. For both weighted residual and variational formulations, the following restrictions are accepted for establishing convergence of the finite element model as the mesh refinement increases:

1. The element interpolation functions must be capable of modeling 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. The element interpolation 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.

Through the assumption of the spatial interpolations, the variables of interest and their derivatives are uniquely specified throughout the solution domain by the nodal parameters associated with the nodal points of the system. The parameters at a particular node directly influence only the elements connected to that particular node. The domain will be split into a mesh. That will require that we establish some bookkeeping processes to keep up with data going to, or coming from a node or element. Those processes are commonly called gather and scatter, respectively. Figure 1.3 shows some of these processes for a simple mesh with one generalized scalar unknown per node, $n_g = 1$, in a one-dimensional physical space. There the system node numbers are shown numbered in an arbitrary fashion. To establish the local element space domain we must usually gather the coordinates of each of its nodes. For example, for element $b(= 5)$ it gathers the data for system node numbers 6 and 4, respectively, so that the element length, $L^{(5)} = x_6 - x_4$, can be computed. Usually we also have to gather some data on the coefficients in the differential equation (material properties). If the coefficients vary over space they may be supplied as data at the nodes that must also be gathered to form the element matrices.

After the element behavior has been described by spatial assumptions, then the derivatives of the space functions are used to approximate the spatial derivatives required in the integral form. The remaining fundamental problem is to establish the element matrices, $\mathbf{S}^e$ and $\mathbf{C}^e$. This involves substituting the approximation space functions and their derivatives into the governing integral form and moving the unknown coefficients, $\mathbf{D}^e$, outside the integrals. Historically, the resulting matrices have been called the element stiffness matrix and load vector, respectively.

Once the element equations have been established the contribution of each element is added, using its topology (or connectivity), to form the system equations. The system of algebraic equations resulting from FEA (of a linear system) will be of the form $\mathbf{S}\,\mathbf{D} = \mathbf{C}$. The vector $\mathbf{D}$ contains the unknown nodal parameters, and the matrices $\mathbf{S}$ and $\mathbf{C}$ are obtained by assembling the known element matrices, $\mathbf{S}^e$ and $\mathbf{C}^e$, respectively. Figure 1.3 shows how the local coefficients of the element source vector, $\mathbf{C}^e$, are scattered and added into the resultant system source, $\mathbf{C}$. That illustration shows a conversion of local row numbers to the corresponding system row numbers (by using the element connectivity data). An identical conversion is used to convert the local and system column numbers needed in assembling each $\mathbf{S}^e$ into $\mathbf{S}$. In the majority of problems $\mathbf{S}^e$, and thus, $\mathbf{S}$, will be symmetric. Also, the system square matrix, $\mathbf{S}$, is usually banded

about the diagonal or at least *sparse*. If **S** is unsymmetric its upper and lower triangles have the same sparsity.

After the system equations have been assembled, it is necessary to apply the *essential boundary constraints* before solving for the unknown nodal parameters. The most common types of essential boundary conditions (EBC) are (1) defining explicit values of the parameter at a node and (2) defining constraint equations that are linear combinations of the unknown nodal quantities. The latter constraints are often referred to in the literature as *multi-point constraints* (MPC). An essential boundary condition should not be confused with a forcing condition of the type that involves a flux or traction on the boundary of one or more elements. These element boundary source, or forcing, terms contribute additional terms to the governing integral form and thus to the element square and/or column matrices for the elements on which the sources were applied. Thus, although these (*Neumann-type*, and *Robin* or *mixed-type*) conditions do enter into the system equations, their presence may not be obvious at the system level. Wherever essential boundary conditions do not act on part of the boundary, then at such locations, source terms from a lower order differential equation automatically apply. If one does not supply data for the source terms, then they default to zero. Such portions of the boundary are said to be subject to natural boundary conditions (NBC). The natural boundary condition varies with the integral form, and typical examples will appear later.

The initial sparseness (the relative percentage of zero entries) of the square matrix, **S**, is an important consideration since we only want to store non-zero terms. If we employ a direct solver then many initially zero terms will become non-zero during the solution process and the assigned storage must allow for that. The 'fill-in' depends on the numbering of the nodes. If the FEA system being employed does not have an automatic renumbering system to increase sparseness, then the user must learn how to number nodes (or elements) efficiently. After the system algebraic equations have been solved for the unknown nodal parameters, it is usually necessary to output the parameters, **D**. For every essential boundary condition on **D**, there is a corresponding unknown *reaction* term in **C** that can be computed after **D** is known. These usually have physical meanings and should be output to help check the results.

In rare cases the problem would be considered completed at this point, but in most cases it is necessary to use the calculated values of the nodal parameters to calculate other quantities of interest. For example, in stress analysis we use the calculated nodal displacements to solve for the strains and stresses. All adaptive programs must do a very large amount of post-processing to be sure that the solution, **D**, has been obtained to the level of accuracy specified by the analyst. Figure 1.3 also shows that the gather operation is needed again for extracting the local results, $\mathbf{D}^e$, from the total results, **D**, so they can be employed in special element post-processing and/or error estimates.

Usually the post-processing calculations involve determining the spatial derivatives of the solution throughout the mesh. Those gradients are continuous within each element domain, but are discontinuous across the inter-element boundaries. The true solution usually has continuous derivatives so it is desirable to somehow average the individual element gradient estimates to create continuous gradient estimate values that can be reported at the nodes. Fortunately, this addition gradient averaging process also provides

new information that allows the estimate of the problem error norm to be calculated. That gradient averaging process will be presented in Chapters 2 and 6.

In the next chapter we will review the historical approach of the method of weighted residuals and its extension to finite element analysis. The earliest formulations for finite element models were based on variational techniques. This is especially true in the areas of structural mechanics and stress analysis. Modern analysis in these areas has come to rely on FEA almost exclusively. Variational models find the nodal parameters that yield a minimum (or stationary) value of an integral known as a functional. In most cases it is possible to assign a physical meaning to the integral. For example, in solid mechanics the integral represents the *total potential energy*, whereas in a fluid mechanics problem it may correspond to the rate of entropy production. Most physical problems with variational formulations result in quadratic forms that yield algebraic equations for the system which are symmetric and positive definite. The solution that yields a minimum value of the integral functional and satisfies the essential boundary conditions is equivalent to the solution of an associated differential equation. This is known as the Euler theorem.

Compared to the method of weighted residuals, where we start with the differential equation, it may seem strange to start a variational formulation with an integral form and then check to see if it corresponds to the differential equation we want. However, from Euler's work more than two centuries ago we know the variational forms of most even order differential equations that appear in science, engineering, and applied mathematics. This is especially true for elliptical equations. Euler's Theorem of Variational Calculus states that the solution, $u$, that satisfies the essential boundary conditions and renders stationary the functional

$$I = \int_{\Omega} f\left( x, y, z, \phi, \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z} \right) d\Omega + \int_{\Gamma} \left( q\phi + a\phi^2/2 \right) d\Gamma \tag{1.6}$$

also satisfies the partial differential equation

$$\frac{\partial f}{\partial \phi} - \frac{\partial}{\partial x} \frac{\partial f}{\partial (\partial \phi / \partial x)} - \frac{\partial}{\partial y} \frac{\partial f}{\partial (\partial \phi / \partial y)} - \frac{\partial}{\partial z} \frac{\partial f}{\partial (\partial \phi / \partial z)} = 0 \tag{1.7}$$

in $\Omega$, and satisfies the natural boundary condition that

$$n_x \frac{\partial f}{\partial (\partial \phi / \partial x)} + n_y \frac{\partial f}{\partial (\partial \phi / \partial y)} + n_z \frac{\partial f}{\partial (\partial \phi / \partial z)} + q + a\phi = 0 \tag{1.8}$$

on $\Gamma$ that is not subject to an essential boundary. Here $n_x$, $n_y$, $n_z$ are the components of the normal vector on the boundary, $\Gamma$. Note that this theorem also defines the natural boundary condition, as well as the corresponding differential equation. In Chapter 7 we will examine some common Euler variational forms for finite element analysis.

## 1.4 Assembly into the system equations

### 1.4.1 Introduction

An important but often misunderstood topic is the procedure for *assembling* the system equations from the element equations and any boundary contributions. Here assemblying is defined as the operation of adding the coefficients of the element equations into the proper locations in the system equations. There are various methods for accomplishing this but most are numerically inefficient. The numerically efficient *direct assembly* technique will be described here in some detail. We begin by reviewing the simple but important relationship between a set of local (nodal point, or element) degree of freedom numbers and the corresponding system degree of freedom numbers.

The assembly process, introduced in part in Fig. 1.3, is graphically illustrated in Fig. 1.4 for a mesh consisting of six nodes ($n_m = 6$), three elements ($n_e = 3$). It has a four-node quadrilateral and two three-node triangles, with one generalized parameter per node ($n_g = 1$). The top of the figure shows the nodal connectivity of the three elements and a cross-hatching to define the source of the various coefficients that are occurring in the matrices assembled in the lower part of the figure. The assembly of the system **S** and **C** matrices is graphically coded to denote the sources of the contributing terms but not their values. A hatched area indicates a term that was added in from an element that has the same hash code. For example, the load vector term C(6), coming from the only parameter at node 6, is seen to be the sum of contributions from elements 1 and 2, which are hatched with horizontal (-) and vertical (|) lines, respectively. The connectivity table implies the same thing since node 6 is only connected to those two elements. By way of comparison, the term C(1) has a contribution only from element 2. The connectivity table shows only that element is connected to that corner node.

Note that we have to set **S** = **0** to begin the summation. Referring to Fig. 1.4 we see that 10 of the coefficients in **S** remain initially zero. So that example is initially about 27 percent sparse. (This will changed if a direct solution process is used.) In practical problems the assembled matrix may initially be 90 percent sparse, or more. Special equation solving techniques take advantage of this feature to save on memory and operation counts.

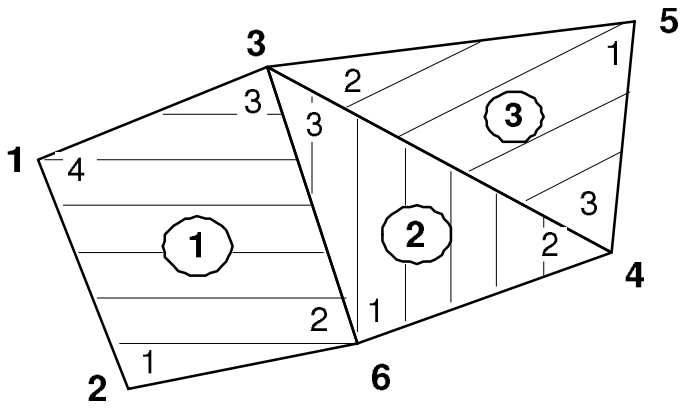
### 1.4.2 Computing the equation index

There are a number of ways to assign the equation numbers of the algebraic system that results from a finite element analysis procedure. Here we will select one that has a simple equation that is valid for most applications. Consider a typical nodal point in the system and assume that there are $n_g$ parameters associated with each node. Thus, at a typical node there will be $n_g$ local degree of freedom numbers ($1 \le J \le n_g$) and a corresponding set of system degree of freedom numbers. If $I$ denotes the system node number of the point, then the $n_g$ corresponding system degrees of freedom, $\Phi_k$ have their equation number, $k$ assigned as

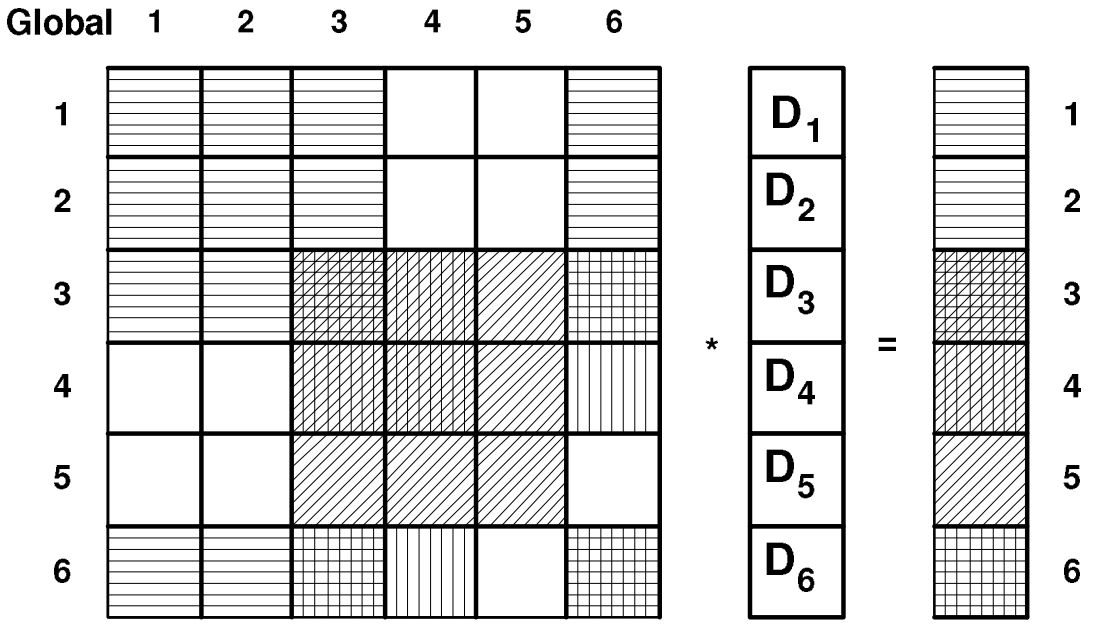
$$k\,(I, J) \;=\; n_g * (\,I - 1\,) + J \qquad 1 \le I \le n_m\,, \qquad 1 \le J \le n_g\,, \tag{1.9}$$

where $n_m$ is the maximum node number in the system. That is, they start at 1 and range to $n_g$ at the first system node then at the second node they range from ($n_g$ + 1) to ($2\,n_g$)

| Element | Topology |
|---|---|
| 1 | 2, 6, 3, 1 |
| 2 | 6, 4, 3, 0 |
| 3 | 5, 3, 4, 0 |
| local | 1, 2, 3, 4 |




Global assembly: S * D = C




$$\beta^{(2)} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

**Figure 1.4** *Graphical illustration of matrix assembly*

```
FUNCTION GET_INDEX_AT_PT (I_PT) RESULT (INDEX)                    ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ! 2
!  DETERMINE DEGREES OF FREEDOM NUMBERS AT A NODE                 ! 3
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ! 4
Use System_Constants ! for N_G_DOF                                ! 5
 IMPLICIT NONE                                                    ! 6
 INTEGER, INTENT(IN)  :: I_PT                                     ! 7
 INTEGER              :: INDEX (N_G_DOF)                          ! 8
 INTEGER :: J ! implied loop                                      ! 9
                                                                  !10
! N_G_DOF = NUMBER OF PARAMETERS (DOF) PER NODE                   !11
! I_PT    = SYSTEM NODE NUMBER                                    !12
! INDEX   = SYSTEM DOF NOS OF NODAL DOF                           !13
!           INDEX (J) = N_G_DOF*(I_PT - 1) + J                    !14
                                                                  !15
  INDEX = (/ (N_G_DOF*(I_PT - 1) + J, J = 1, N_G_DOF) /)          !16
END FUNCTION GET_INDEX_AT_PT                                      !17

FUNCTION GET_ELEM_INDEX (LT_N, ELEM_NODES) RESULT(INDEX)          ! 1
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ! 2
!   DETERMINE DEGREES OF FREEDOM NUMBERS OF ELEMENT               ! 3
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *    ! 4
Use System_Constants    ! for N_G_DOF                             ! 5
 IMPLICIT NONE                                                    ! 6
 INTEGER, INTENT(IN) :: LT_N, ELEM_NODES (LT_N)                   ! 7
 INTEGER             :: INDEX (LT_N * N_G_DOF)   ! OUT            ! 8
 INTEGER :: EQ_ELEM, EQ_SYS, IG, K, SYS_K        ! LOOPS          ! 9
                                                                  !10
! ELEM_NODES = NODAL INCIDENCES OF THE ELEMENT                    !11
! EQ_ELEM    = LOCAL  EQUATION NUMBER                             !12
! EQ_SYS     = SYSTEM EQUATION NUMBER                             !13
! INDEX      = SYSTEM DOF NUMBERS OF ELEMENT DOF NUMBERS          !14
! INDEX (N_G_DOF*(K-1)+IG) = N_G_DOF*(ELEM_NODES(K)-1) + IG      !15
! LT_N       = NUMBER OF NODES PER ELEMENT                        !16
! N_G_DOF    = NUMBER OF GENERAL PARAMETERS (DOF) PER NODE        !17
                                                                  !18
  DO K = 1, LT_N               ! LOOP OVER NODES OF ELEMENT       !19
    SYS_K = ELEM_NODES (K)     ! SYSTEM NODE NUMBER               !20
    DO IG = 1, N_G_DOF         ! LOOP OVER GENERALIZED DOF        !21
      EQ_ELEM = IG + N_G_DOF * (K     - 1)      ! LOCAL  EQ       !22
      EQ_SYS  = IG + N_G_DOF * (SYS_K - 1)      ! SYSTEM EQ       !23
      IF ( SYS_K > 0 ) THEN           ! VALID NODE                !24
        INDEX (EQ_ELEM) = EQ_SYS                                  !25
      ELSE                            ! ALLOW MISSING NODE        !26
        INDEX (EQ_ELEM) = 0                                       !27
      END IF ! MISSING NODE                                       !28
    END DO ! OVER DOF                                             !29
  END DO ! OVER LOCAL NODES                                       !30
END FUNCTION GET_ELEM_INDEX                                       !31
```

**Figure 1.5** *Computing equation numbers for homogeneous nodal dof*

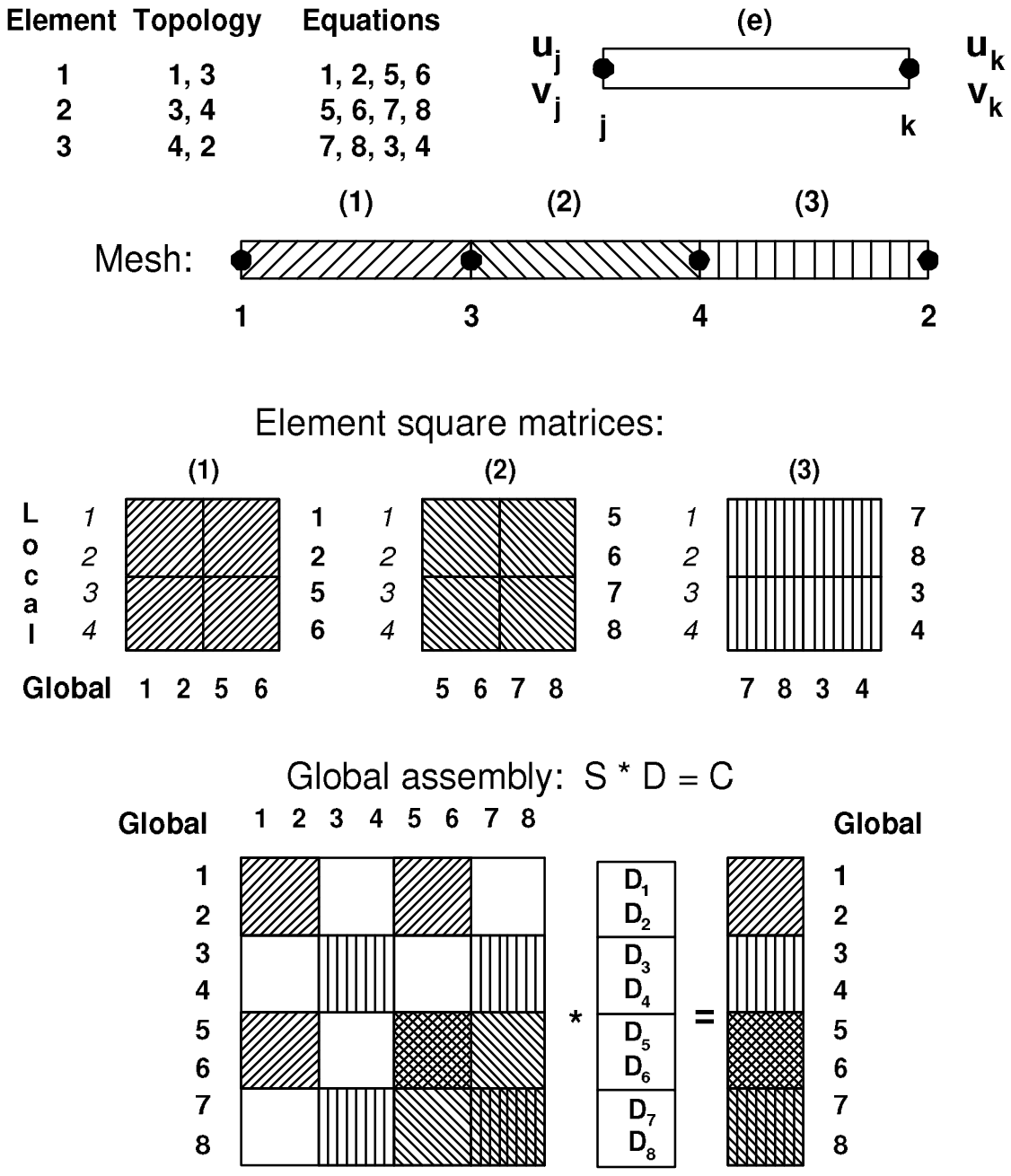



**Figure 1.6** *Assembling two unknowns per node*