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 design it is rare to find a project that does not require some type of finite element analysis (FEA). When not actually required, FEA can usually be utilized to improve a design. The practical advantages of FEA in stress analysis and structural dynamics have made it the accepted design tool for the last two decades. It is also heavily employed in thermal analysis, especially in connection with 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 postprocessing 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. These classes 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 the designer and illustrate some of its theoretical formulations and practical applications. The modern designer 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.

1.2 Capabilities of FEA

There are many commercial and public-domain finite element systems that are available to the designer. 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% 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 a 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 (up to 81 for a solid). 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. 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. 1.2.1.

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*, 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 algebraic equations. Thus, for each element it is necessary to input, in some consistent order, the

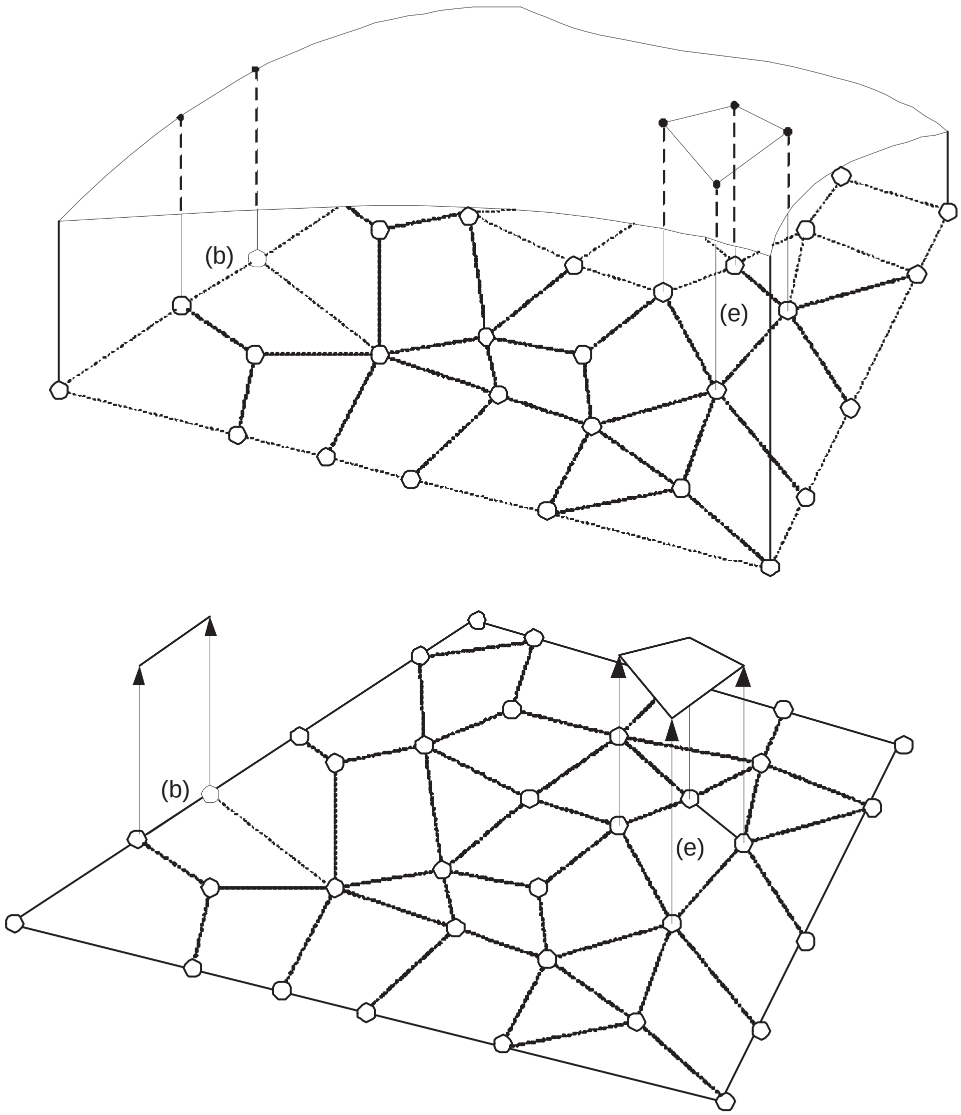


Figure 1.2.1 Piecewise approximation of a scalar function

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 associate a material code, or properties, with each element.

Finite element analysis can require very large amounts of input data. Thus, most FEA systems offer the user significant data generation or supplementation 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. Thus, stress file editors are usually provided to allow the designer to selectively extract such data. Most of the output options from an FEA system are available in graphical form. The most commonly needed information in the design process is the state of temperatures or stresses and displacements. 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 designs 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.

Today efficient utilization of materials in the design processes 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

Table 1.1 Typical variables in finite element analysis

Application	Primary	Associated	Secondary
Stress analysis	Displacement, Rotation	Force, Moment	Stress, Failure criterion Error estimates
Heat transfer	Temperature	Flux	Interior flux Error estimates
Potential flow	Potential function	Normal velocity	Interior velocity Error estimates
Navier-Stokes	Velocity	Pressure	Error estimates

large commercial FEA systems include buckling, creep, large deflections, and plasticity.

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 have the ability to handle completely arbitrary geometries, which is essential to practical engineering design. 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. Thus, these things mean that for practical use in design, 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.

Table 1.2 Typical given variables and corresponding reactions

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

All components employed in a design are three-dimensional but several common special cases have been defined that allow two-dimensional studies to provide useful design 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 modellers can generate three-dimensional meshes relative 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 "ballpark" checks on the three-dimensional answers. Of course, use of basic handbook calculations in estimating the answer before approaching a FEA system is also highly recommended.

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* 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

$$L(\phi) \ = \ Q\,,$$

where L denotes a differential operator acting on the primary unknown, ϕ, and Q is a source term. Generally we assume an approximate solution, say ϕ^*, and substitute this solution into the differential equation. Since the assumption is approximate, this operation defines a residual error term, R, in the differential equation

$$L(\phi^*) \ - \ Q = R \neq 0\,.$$

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

$$I = \int_V RW\, dV = 0\,.$$

Substituting an assumed spatial behavior for the approximate solution, ϕ^*, 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. The choice of weighting function defines the type of weighted residual technique being utilized. The Galerkin criterion selects

$$W = \phi^* ,$$

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 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. Since the domain will be split into pieces 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 1.2.2 shows some of these processes for a simple mesh with one scalar unknown per node in a one-dimensional physical space. To establish the local element space domain we must usually gather the coordinates of each of its nodes. Usually we also have to gather some data on the coefficients in the differential equation (material properties usually). If they 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 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 1.2.2 shows how the local coefficents 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

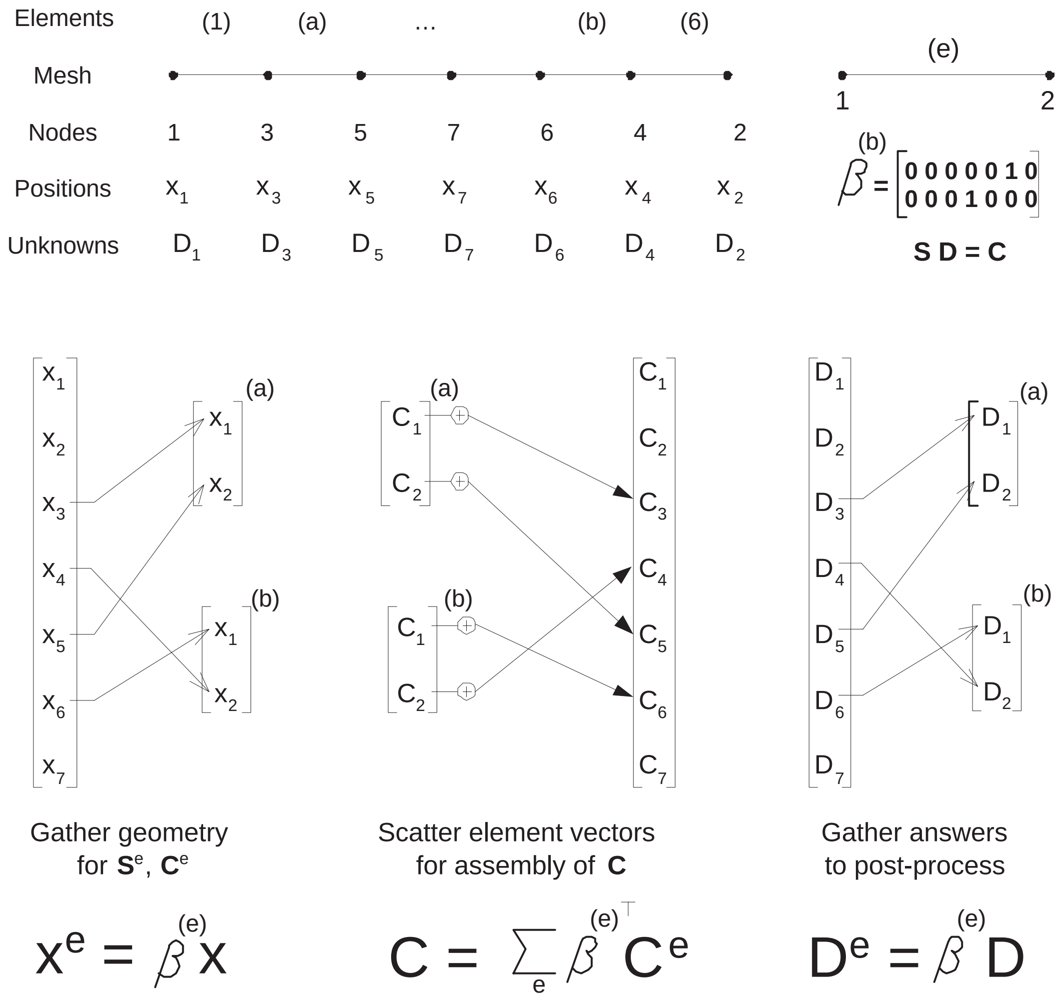

Elements
(1)
(a)
…
(b)
(6)
Mesh
Nodes
1 3 5 7 6 4 2
Positions
x_1 x_3 x_5 x_7 x_6 x_4 x_2
Unknowns
D_1 D_3 D_5 D_7 D_6 D_4 D_2
(e)
1
2
$\beta^{(b)} = \begin{bmatrix} 0&0&0&0&0&1&0 \\ 0&0&0&1&0&0&0 \end{bmatrix}$
S D = C
x_1 x_2 x_3 x_4 x_5 x_6 x_7
(a)
(b)
C_1 C_2 C_3 C_4 C_5 C_6 C_7
D_1 D_2 D_3 D_4 D_5 D_6 D_7
Gather geometry for $\mathbf{S}^e$, $\mathbf{C}^e$
Scatter element vectors for assembly of **C**
Gather answers to post-process
$x^e = \overset{(e)}{\beta} x$
$C = \sum_e \overset{(e)}{\beta}{}^T C^e$
$D^e = \overset{(e)}{\beta} D$

Figure 1.2.2 Gather and scatter concepts for finite elements

identical conversion is used to convert the local and system column numbers needed in assemblying each $\mathbf{S}^e$ into **S**. In the majority of problems $\mathbf{S}^e$, and thus, **S**, will be symmetric. Also, the system square matrix, **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).

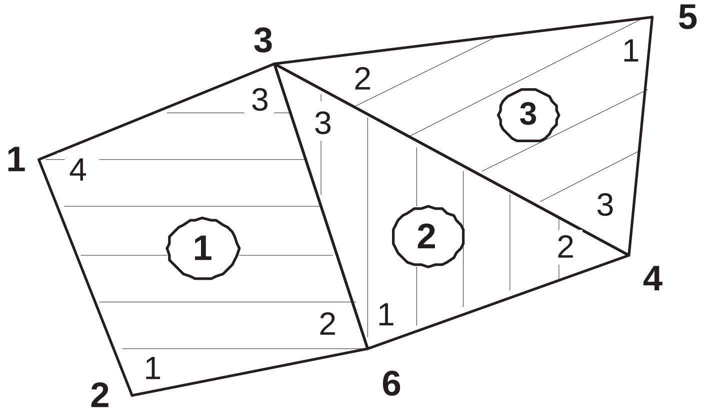
The sparseness (the relative percentage of non-zero entries)
of the square matrix, **S**, is an important consideration. It depends on the numbering of the nodes (or the elements). If the FEA system being employed does not have an automatic renumbering system to increase sparseness, then the user must 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 the designer check the results.

In some 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 postprocessing to be sure that the solution, **D**, has been obtained to the level of accuracy specified by the analyst. Figure 1.2.2 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.

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, 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 varational 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 especially true for elliptical equations. Recall that Euler's Theorem of Variational Calculus states that the solution, *u*, that satisfies the essential boundary conditions and renders stationary the functional

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

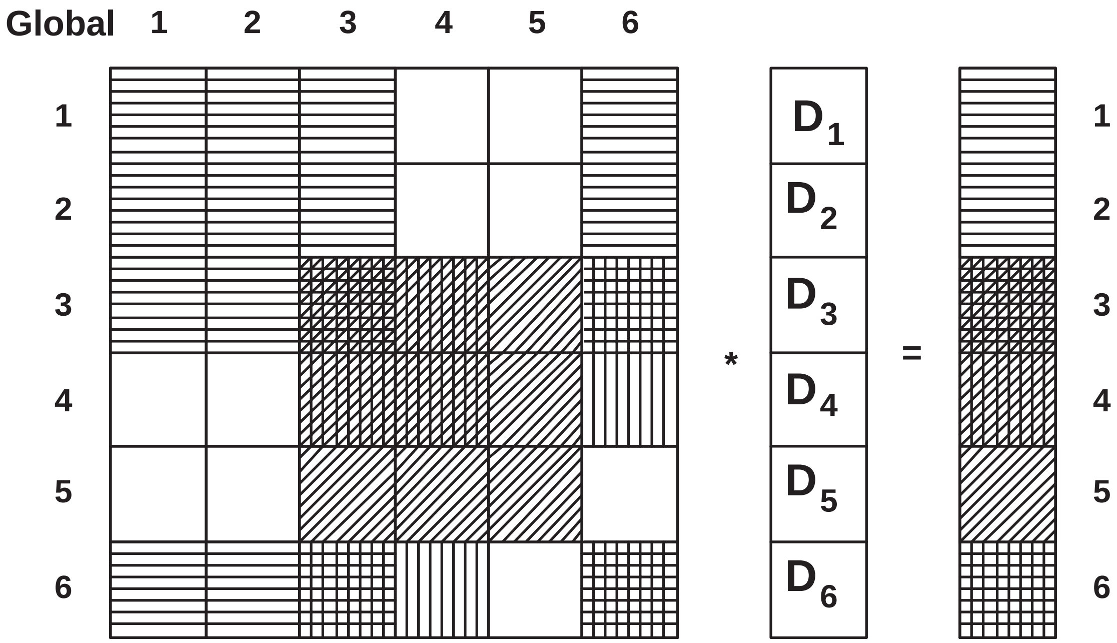


Figure 1.3.1 Graphical illustration of matrix assembly

```
SUBROUTINE STORE_COLUMN (N_D_FRE, N_EL_FRE, INDEX, C, CC)
! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
! STORE ELEMENT COLUMN MATRIX IN SYSTEM COLUMN MATRIX
! * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Use Precision_Module ! Defines DP for double precision
 IMPLICIT NONE
 INTEGER,  INTENT(IN)    :: N_D_FRE, N_EL_FRE
 INTEGER,  INTENT(IN)    :: INDEX (N_EL_FRE)
 REAL(DP), INTENT(IN)    :: C     (N_EL_FRE)
 REAL(DP), INTENT(INOUT) :: CC    (N_D_FRE)
 INTEGER :: I, J

! N_D_FRE  = NO DEGREES OF FREEDOM IN THE SYSTEM
! N_EL_FRE = NUMBER OF DEGREES OF FREEDOM PER ELEMENT
! INDEX    = SYSTEM DOF NOS OF THE ELEMENT DOF
! C        = ELEMENT COLUMN MATRIX
! CC       = SYSTEM COLUMN MATRIX

  DO I = 1, N_EL_FRE                     ! ELEMENT ROW
    J = INDEX (I)                        ! SYSTEM ROW NUMBER
    IF ( J > 0 ) CC (J) = CC (J) + C (I) ! SKIP INACTIVE ROW
  END DO ! OVER ROWS
END SUBROUTINE STORE_COLUMN

SUBROUTINE STORE_FULL_SQUARE (N_D_FRE, N_EL_FRE, S, SS, INDEX)
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
!     STORE ELEMENT SQ MATRIX IN FULL SYSTEM SQ MATRIX
! * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
Use Precision_Module ! Defines DP for double precision
 IMPLICIT NONE
 INTEGER,  INTENT(IN)    :: N_D_FRE, N_EL_FRE
 INTEGER,  INTENT(IN)    :: INDEX (N_EL_FRE)
 REAL(DP), INTENT(IN)    :: S     (N_EL_FRE, N_EL_FRE)
 REAL(DP), INTENT(INOUT) :: SS    (N_D_FRE,  N_D_FRE)
 INTEGER   :: I, II, J, JJ

! N_D_FRE  = TOTAL NO OF SYSTEM DEGREES OF FREEDOM
! N_EL_FRE = NO DEGREES OF FREEDOM PER ELEMENT
! INDEX    = SYSTEM DOF NOS OF ELEMENT PARAMETERS
! S        = FULL ELEMENT SQUARE MATRIX
! SS       = FULL SYSTEM SQUARE MATRIX

  DO I = 1, N_EL_FRE                     ! ELEMENT ROW
    II = INDEX (I)                       ! SYSTEM ROW NUMBER
    IF ( II > 0 ) THEN                   ! SKIP INACTIVE ROW
      DO J = 1, N_EL_FRE                 ! ELEMENT COLUMN
        JJ = INDEX (J)                   ! SYSTEM COLUMN
        IF ( JJ > 0 ) SS (II, JJ) = SS (II, JJ) + S (I, J)
      END DO ! OVER COLUMNS
    END IF
  END DO ! OVER ROWS
END SUBROUTINE STORE_FULL_SQUARE
```

Figure 1.3.2   Assembly of element arrays into system arrays

```
FUNCTION GET_INDEX_AT_PT (I_PT) RESULT (INDEX)
! * * * * * * * * * * * * * * * * * * * * * * * * * *
!  DETERMINE DEGREES OF FREEDOM NUMBERS AT A NODE
! * * * * * * * * * * * * * * * * * * * * * * * * * *
Use System_Constants ! for N_G_DOF
 IMPLICIT NONE
 INTEGER, INTENT(IN)  :: I_PT
 INTEGER              :: INDEX (N_G_DOF)
 INTEGER :: J ! implied loop

! N_G_DOF = NUMBER OF PARAMETERS (DOF) PER NODE
! I_PT    = SYSTEM NODE NUMBER
! INDEX   = SYSTEM DOF NOS OF NODAL DOF
!           INDEX (J) = N_G_DOF*(I_PT - 1) + J

  INDEX = (/ (N_G_DOF*(I_PT - 1) + J, J = 1, N_G_DOF) /)
END FUNCTION GET_INDEX_AT_PT

FUNCTION GET_ELEM_INDEX (LT_N, ELEM_NODES) RESULT(INDEX)
! * * * * * * * * * * * * * * * * * * * * * * * * * * *
!   DETERMINE DEGREES OF FREEDOM NUMBERS OF ELEMENT
! * * * * * * * * * * * * * * * * * * * * * * * * * * *
Use System_Constants    ! for N_G_DOF
 IMPLICIT NONE
 INTEGER, INTENT(IN) :: LT_N, ELEM_NODES (LT_N)
 INTEGER             :: INDEX (LT_N * N_G_DOF)   ! OUT
 INTEGER :: EQ_ELEM, EQ_SYS, IG, K, SYS_K        ! LOOPS

! ELEM_NODES = NODAL INCIDENCES OF THE ELEMENT
! EQ_ELEM    = LOCAL  EQUATION NUMBER
! EQ_SYS     = SYSTEM EQUATION NUMBER
! INDEX      = SYSTEM DOF NUMBERS OF ELEMENT DOF NUMBERS
! INDEX (N_G_DOF*(K-1)+IG) = N_G_DOF*(ELEM_NODES(K)-1) + IG
! LT_N       = NUMBER OF NODES PER ELEMENT
! N_G_DOF    = NUMBER OF GENERAL PARAMETERS (DOF) PER NODE

  DO K = 1, LT_N               ! LOOP OVER NODES OF ELEMENT
    SYS_K = ELEM_NODES (K)     ! SYSTEM NODE NUMBER
    DO IG = 1, N_G_DOF         ! LOOP OVER GENERALIZED DOF
      EQ_ELEM = IG + N_G_DOF * (K     - 1)      ! LOCAL  EQ
      EQ_SYS  = IG + N_G_DOF * (SYS_K - 1)      ! SYSTEM EQ
      IF ( SYS_K > 0 ) THEN           ! VALID NODE
        INDEX (EQ_ELEM) = EQ_SYS
      ELSE                            ! ALLOW MISSING NODE
        INDEX (EQ_ELEM) = 0
      END IF ! MISSING NODE
    END DO ! OVER DOF
  END DO ! OVER LOCAL NODES
END FUNCTION GET_ELEM_INDEX
```

Figure 1.3.3   Computing equation numbers for homogeneous nodal dof

$$I \;=\; \int_{\Omega} f\left(x, y, z, u, \frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial u}{\partial z} \right) d\Omega \;+\; \int_{\Gamma} \left(qu + au^2/2 \right) d\,\Gamma$$

also satisfies the partial differential equation

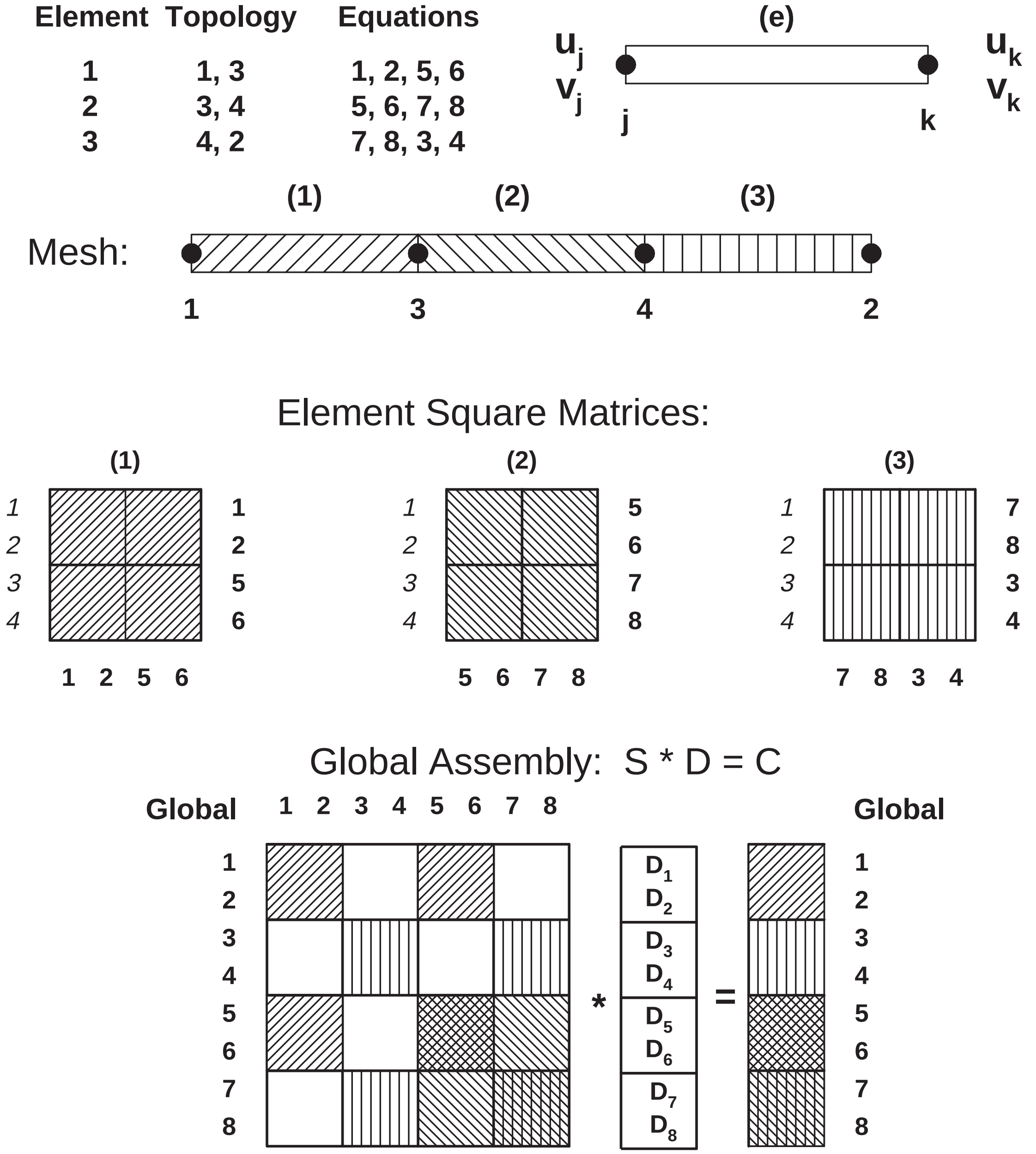
$$\frac{\partial f}{\partial u} - \frac{\partial}{\partial x} \frac{\partial f}{\partial(\partial u/\partial x)} - \frac{\partial}{\partial y} \frac{\partial f}{\partial(\partial u/\partial y)} - \frac{\partial}{\partial z} \frac{\partial f}{\partial(\partial u/\partial z)} \;=\; 0$$

in Ω, and satisfies the natural boundary condition that

$$n_x \frac{\partial f}{\partial(\,\partial u/\partial x\,)} + n_y \frac{\partial f}{\partial(\,\partial u/\partial y\,)} + n_z \frac{\partial f}{\partial(\,\partial u/\partial z\,)} + q + au \;=\; 0$$

on Γ 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, Γ. Note that this theorem also lets us verify what is a natural boundary condition, as well as letting us verify the corresponding differential equation (in case we don't remember its form). In Chapter 3 we will examine some common variational forms and their extensions to finite element analysis.

The assembly process, introduced in Fig. 1.2.2, is graphically illustrated in Fig. 1.3.1 for a three-element mesh consisting of a four-node quadrilateral and two three-node triangles, with one parameter per node. 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 occuring in the matrices assembled in the lower part of the figure. The assembly of the system **S** and **C** matrices is 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) is seen to be the sum of contributions from elements 2 and 3, which are hatched with horizontal (–) and oblique (/) 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. Figure 1.3.2 shows how the assembly can be implemented for column matrices (subroutine STORE_COLUMN) and full (non-sparse) square matrices (STORE_FULL_SQUARE) if one has an integer index that relates the local element degrees of freedom (dof) to the system dof.



Element	Topology	Equations
1	1, 3	1, 2, 5, 6
2	3, 4	5, 6, 7, 8
3	4, 2	7, 8, 3, 4

Figure 1.3.4 Assemblying two unknowns per node

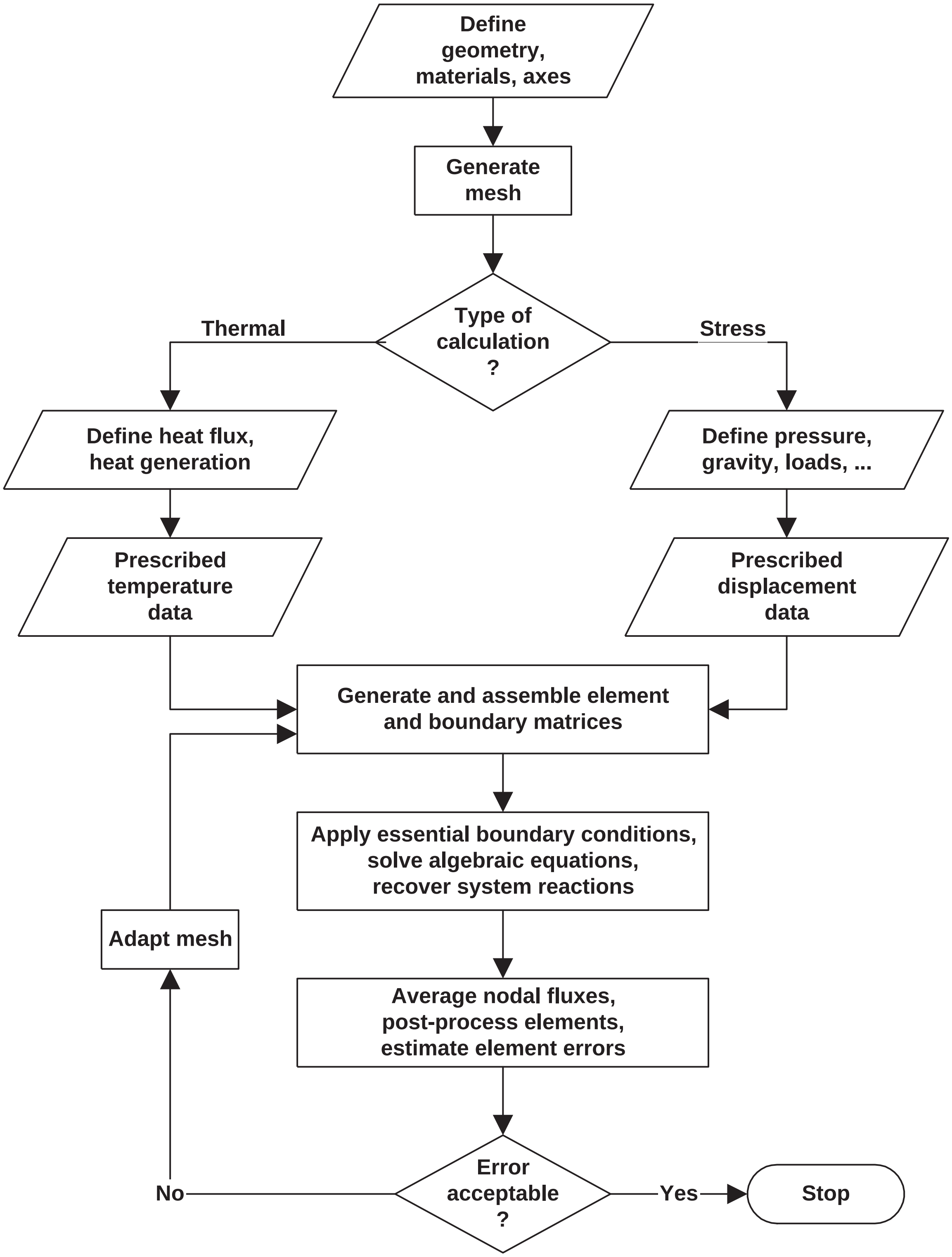

Define geometry, materials, axes
Generate mesh
Type of calculation ?
Thermal
Stress
Define heat flux, heat generation
Define pressure, gravity, loads, ...
Prescribed temperature data
Prescribed displacement data
Generate and assemble element and boundary matrices
Apply essential boundary conditions, solve algebraic equations, recover system reactions
Adapt mesh
Average nodal fluxes, post-process elements, estimate element errors
Error acceptable ?
No
Yes
Stop

Figure 1.3.5 Typical stages in a finite element analysis

1.4 Equation Assembly

Figure 1.3.3 shows how that index could be computed for a node or element for the common case where the number of generalized degrees of freedom per node, N_G_DOF, is everywhere constant. When there is only a single unknown per node (N_G_DOF = 1), as shown in Fig. 1.3.1, then the nodal degree of freedom loop (at lines 16 and 21 in Fig. 1.3.3) simply equates the equation number to the global node number. An example where there are two unknowns per node is illustrated in Fig. 1.3.4. That figure shows a line element mesh with two nodes per element and two dof per node (such as a standard beam element). A more complicated indexing is needed when the number of dof per node is variable (as with hierarchical elements), and when one must account for special storage methods used for large sparse matrices. Additional considerations of the assembly process will be given in the next chapter. Finally, Figure 1.3.5 illustrates the general processes we will consider in this implementation of FEA.

1.5 Exercises

1. Assume (unrealistically) that all the entries in an element square matrix and column vector are equal to the element number. Carry out the assembly of the system in Fig. 1.3.1 to obtain the final numerical values for each coefficient in the **S** and **C** matrices.

2. Assume (unrealistically) that all the entries in an element square matrix and column vector are equal to the element number. Carry out the assembly of the system in Fig. 1.3.4 to obtain the final numerical values for each coefficient in the **S** and **C** matrices.

3. In Fig. 1.3.4 assume that the global nodes are number consectutively from 1 to 4 (from left to right). Write the element index vector for each of the three elements.

1.6 References

[1] Adams, V. and Askenazi, A., *Building Better Products with Finite Element Analysis*, Santa Fe: Onword Press (1999).

[2] Akin, J.E., *Computer Assisted Mechanical Design*, Englewood Cliffs: Prentice-Hall (1990).

[3] Akin, J.E., *Finite Elements for Analysis and Design*, London: Academic Press (1994).

[4] Akin, J.E. and Singh, M., "Object-Oriented Fortran 90 P-Adaptive Finite Element Method," pp. 141−149 in *Developments in Engineering Computational Technology*, ed. B.H.V. Topping, Edinburgh: Civil_Comp Press (2000).

[5] Akin, J.E., *Object-Oriented Programming Via Fortran 90/95*, Cambridge: Cambridge University Press (2003).

[6] Bathe, K.J., *Finite Element Procedures*, Englewood Cliffs: Prentice-Hall (1996).

[7] Becker, E.B., Carey, G.F., and Oden, J.T., *Finite Elements − An Introduction*, Englewood Cliffs: Prentice-Hall (1981).

[8] Cook, R.D., Malkus, D.S., and Plesha, N.E., *Concepts and Applications of Finite Element Analysis*, New York: John Wiley (1989).

[9] Cook, R.D., Malkus, D.S., Plesha, N.E., and Witt, R.J., *Concepts and Applications of Finite Element Analysis*, New York: John Wiley (2002).

[10] Desai, C.S. and , T. Kundu, *Introduction to the Finite Element Method*, Boca Raton, FL: CRC Press (2001).

[11] Gupta, K.K and Meek, J.L., *Finite Element Multidisciplinary Analysis,* Reston: AIAA (2000).

[12] Huebner, K.H., Thornton, E.A., and Byrom, T.G., *Finite Element Method for Engineers*, New York: John Wiley (1994).

[13] Hughes, T.J.R., *The Finite Element Method*, Englewood Cliffs: Prentice-Hall (1987).

[14] Norrie, D.H. and DeVries, G., *Finite Element Bibliography*, New York: Plenum Press (1976).

[15] Pironneau, O., *Finite Element Methods for Fluids*, New York: John Wiley (1991).

[16] Rao, S.S., *The Finite Element Method in Engineering*, Boston : Butterworth Heinemann (1999).

[17] Segerlind, L.J., *Applied Finite Element Analysis*, New York: John Wiley (1987).

[18] Silvester, P.P. and Ferrari, R.L., *Finite Elements for Electrical Engineers*, Cambridge: Cambridge University Press (1996).

[19] Smith, I.M. and Griffiths, D.V., *Programming the Finite Element Method,* 3nd Edition, Chichester: John Wiley (1998).

[20] Szabo, B. and Babuska, I., *Finite Element Analysis*, New York: John Wiley (1991).

[21] Whiteman, J.R., *A Bibliography of Finite Elements*, London: Academic Press (1975).

[22] Zienkiewicz, O.C. and Taylor, R.L., *The Finite Element Method,* 4th Edition, New York: McGraw-Hill (1991).