Advanced Topics in Finite Element Analysis

1 INTRODUCTION

We shall discuss some of the advanced topics in finite element analysis in this chapter. We start with a very important transformation law known as the law of contragredient transformation. We take a few topics wherein this transformation is used to solve some peculiar and interesting problems.

2 GENERAL RULE OF TRANSFORMATION

Consider two statically equivalent force systems $\{f\}$ and $\{F\}$ which are related to each other by the linear transformation

$$\{F\} = [T]^T \{f\}. \tag{2.1}$$

The transformation matrix $[T]$, in general, is a rectangular matrix. For example, consider the truss structure shown in Fig. 2.1. Here, the force system $\{F\}$ corresponds to the external loads acting on the joints of the truss and is given by

$$\{F\} = \begin{Bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{Bmatrix}.$$

On the other hand, the force system $\{f\}$ represents the member forces in the seven members of the truss given by

$$\{f\} = \begin{Bmatrix} f_1 \\ f_2 \\ \vdots \\ f_7 \end{Bmatrix}.$$

As the two systems of forces $\{f\}$ and $\{F\}$ are of two different sizes in this example, the transformation matrix $[T]$ of Eq. 2.1 is rectangular and is of size 7×4 (it is easy to see that T^T is of size 4×7). Moreover, $[T]$ represents the equilibrium between the two force systems (the member forces $\{f\}$ are produced due to the external forces $\{F\}$).

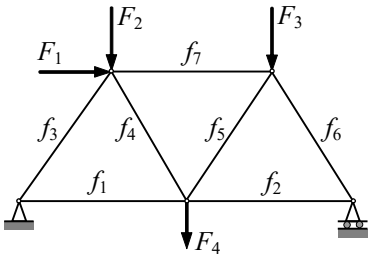

Figure 2.1 Two statically equivalent force systems $\{F\}$ and $\{f\}$ acting on a truss.

Now, let us assume that the displacements consistent with $\{f\}$ and $\{F\}$ are denoted by $\{u\}$ and $\{U\}$ respectively. Let $\{F\}$ and $\{U\}$ be of size $m\times1$ each, and $\{f\}$ and $\{u\}$ be of size $n\times1$ each. Then, $[T]$ will be of size $n\times m$ (or $[T]^T$ is of size $m\times n$). Equating the work done by the two force systems over the corresponding displacements, we get

$$\{F\}^T \{U\} = \{f\}^T \{u\}.$$

Substituting for $\{F\}$ from Eq. 2.1 in the above, we obtain

$$\{f\}^T [T] \{U\} = \{f\}^T \{u\}.$$

As $\{f\}$ is arbitrary, it follows from the above that the two sets of displacements $\{u\}$ and $\{U\}$ must be related by

$$\{u\} = [T] \{U\}. \tag{2.2}$$

Thus, if two systems of statically equivalent forces $\{f\}$ and $\{F\}$ are related by the transformation given by Eq. 2.1, then the corresponding consistent sets of displacements $\{u\}$ and $\{U\}$ are related by the transformation given by Eq. 2.2. This is known as the *law of contragredient transformation*.

Now, let the force system $\{f\}$ be related to the displacements $\{u\}$ by the force-displacement relations

$$[k]\{u\} = \{f\}. \tag{2.3}$$

Then, making use of Eq. 2.2 in 2.3, we obtain

$$[k][T]\{U\} = \{f\}.$$

Premultiplying both sides of the above equation by $[T]^T$, and making use of Eq. 2.1, we get

$$[K]\{U\} = \{F\}, \tag{2.4}$$

where $[K]$ is related to $[k]$ by

$$[K] = [T]^T [k][T]. \tag{2.5}$$

The relations given by Eqs. 2.1, 2.2 and 2.5 are of great use in many of the finite element applications as we shall see in the next few sections.

3 STATIC CONDENSATION

Static condensation is the process by which unwanted degrees of freedom are eliminated from the finite element equations by substitution. This is useful in removing the internal nodes of higher order Lagrangian elements and in substructuring as discussed in the next section. The finite element equations are written in partitioned form as

$$\begin{bmatrix} K_{rr} & K_{rc} \\ K_{cr} & K_{cc} \end{bmatrix} \begin{Bmatrix} U_r \\ U_c \end{Bmatrix} = \begin{Bmatrix} R_r \\ R_c \end{Bmatrix}, \tag{3.1}$$

where subscripts *r* and *c* denote the "retained" and the "condensed" degrees of freedom. Expanding the second row of this equation, we obtain

$$K_{cr}U_r + K_{cc}U_c = R_c, \tag{3.2}$$

where the matrix symbolism is ignored for ease. From this equation, we get

$$U_c = K_{cc}^{-1}\left(R_c - K_{cr}U_r\right). \tag{3.3}$$

Now, let us expand the first row of Eq. 3.1 and replace U_c by Eq. 3.3 to obtain

$$K_{rr}U_r + K_{rc}\,K_{cc}^{-1}\left(R_c - K_{cr}U_r\right) = R_r,$$

which when rearranged yields

$$\left(K_{rr} - K_{rc}\,K_{cc}^{-1}K_{cr}\right)U_r = R_r - K_{rc}\,K_{cc}^{-1}R_c. \tag{3.5}$$

The above equation can be rewritten as

$$\hat{K}_{rr}U_r = \hat{R}_r, \tag{3.6}$$

where

$$\hat{K}_{rr} = K_{rr} - K_{rc}\,K_{cc}^{-1}K_{cr} \tag{3.7}$$

is the *condensed stiffness matrix* which is of size $r \times r$, and

$$\hat{R}_r = R_r - K_{rc}\,K_{cc}^{-1}R_c \tag{3.8}$$

is the *condensed load vector* of size $r \times 1$. The condensed stiffness matrix and the load vector, which are of lower sizes, are used in the place of the original matrices. Once the degrees of freedom U_r are obtained, the condensed degrees of freedom U_c are calculated using Eq. 3.3. This part of the procedure is known as the "recovery".

It is also possible to effect condensation using the following transformation. We can write

$$\begin{Bmatrix} U_r \\ U_c \end{Bmatrix} = [T_o]\{U_r\}, \tag{3.9}$$

where

$$[T_o] = \begin{bmatrix} I \\ -K_{cc}^{-1}K_{cr} \end{bmatrix}. \tag{3.10}$$

Then, the condensed stiffness matrix and the load vector are obtained by the contragredient law as

$$\hat{K}_{rr} = T_o{}^T K T_o \tag{3.11}$$

and

$$\hat{R}_r = T_o{}^T R\,. \tag{3.12}$$

It ought to be noted that there is no approximation involved in static condensation. The procedure is exact*. In actual computer implementation, static condensation and recovery are carried out using a modified form of Gauss elimination in which the elimination is carried out only on the degrees of freedom U_c. Moreover, the partitions shown in Eq. 3.1 is not a computational requirement. The degrees of freedom to be condensed out can appear in any order and they are eliminated one after the other†.

4 ANALYSIS OF LARGE STRUCTURES—SUBSTRUCTURING

When the structure or the elastic body to be analysed is very large or the finite element mesh needed to capture the solution with sufficient accuracy has a very large number of degrees of freedom and/or the available computer resources is limited, we make use of substructuring. The steps to be carried out are listed below:

1 The structure (the body) is divided into several substructures (subdomains) as shown in Fig. 4.1 such that each can be accommodated in the limited capacity of the available digital computer‡.

2 Set up stiffness matrix and load vector for each substructure.

3 Condense the stiffness matrix and load vector so as to retain only the boundary and interface nodes (for example, for the substructure-I shown in the figure retain the degrees of freedom corresponding to the nodes marked all round the substructure). Each substructure, thus, acts as a huge (or macro) element.

4 Assemble the matrices pertaining to the boundary-only degrees of freedom of the substructures to obtain the global stiffness matrix and load vector for the entire structure.

5 Solve and get the boundary nodal displacements.

6 Go back to each substructure. Knowing the values of unknowns at the boundary, obtain the values at all the interior nodes. Also, compute the stresses for each of the element.

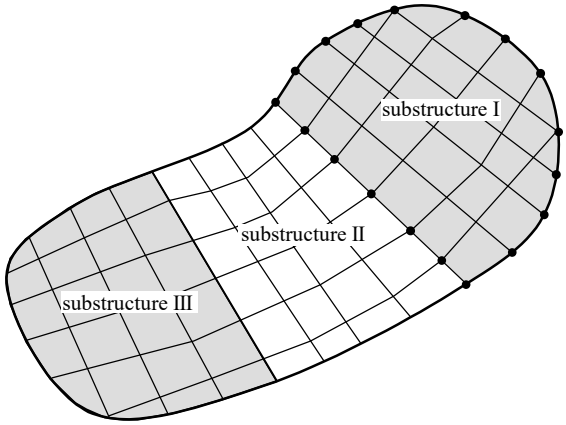

Figure 4.1 Substructuring to analyse large structures.

Thus, it is possible, using the available limited computer resources, to analyse relatively large problems and obtain accurate solutions. As the reader might have rightly guessed, there is no approximation involved in substructuring method. It only uses matrix manipulations.

5 SKEW SUPPORTS

It becomes necessary, often, to deal with roller supports with the plane of the roller making an angle with one of the coordinate planes. Fig. 5.1 shows one example wherein node 3 has such a skew support. We make use of the law of contragredient transformation to tackle these problems.

---

* Condensation in dynamics is termed "reduction" and introduces approximation.

† Condensation algorithms are available, for example, in Desai, C.K., and Abel, J.F, *Introduction to the Finite Element Method*, Van Nostrand Reinhold, 1972, and Cook, R.D., Malkus, D.S., Plesha, M.E., and Witt, R.J., *Concept and Application of Finite Element Analysis*, John Wiley, New York, 2003.

‡ As we keep progressing with the invention of increasingly powerful computer systems, we keep encountering problems that are computationally more demanding. This means that even with the best computer system available in the world there can always be problems which are beyond its capacity to contain and solve with the required accuracy. Thus, "substructuring" is a useful tool even in the years to come.

The two sets of degrees of freedom shown in the figure at the node with the skew support are related to each other by the simple transformation given by

$$\begin{Bmatrix} \delta u_3 \\ \delta v_3 \end{Bmatrix} = \begin{bmatrix} \cos(xn) & \cos(xt) \\ \cos(yn) & \cos(yt) \end{bmatrix} \begin{Bmatrix} \delta n_3 \\ \delta t_3 \end{Bmatrix} \tag{5.1a}$$

or

$$\{\delta_3\} = [C_3]\{\Delta_3\}. \tag{5.1b}$$

The global displacement vector of the entire structure can be written as

$$\{\delta\} = [T]\{\Delta\}, \tag{5.2a}$$

which in expanded form looks as

$$\begin{Bmatrix} \delta_1 \\ \delta_2 \\ \delta_3 \\ \delta_4 \\ \vdots \\ \delta_N \end{Bmatrix} = \begin{Bmatrix} \Delta_1 \\ \Delta_2 \\ C_3\Delta_3 \\ \Delta_4 \\ \vdots \\ \Delta_N \end{Bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 & \cdots & 0 \\ 0 & I & 0 & 0 & & 0 \\ 0 & 0 & C_3 & 0 & & 0 \\ 0 & 0 & 0 & I & & 0 \\ \vdots & & & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & I \end{bmatrix} \begin{Bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \\ \Delta_4 \\ \vdots \\ \Delta_N \end{Bmatrix}. \tag{5.2b}$$

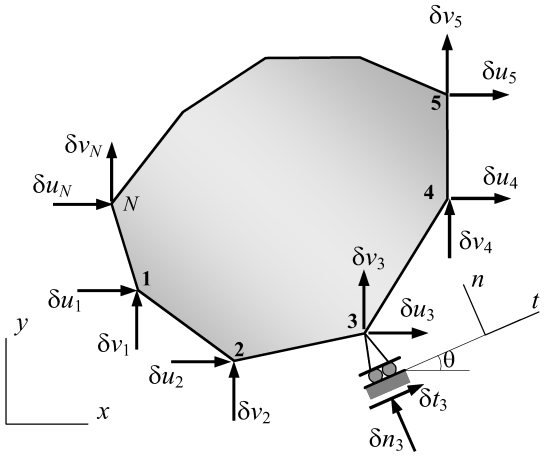

Figure 5.1 Structure with a skew support.

If $\{R\}$ is the global load vector consistent with $\{\Delta\}$, and $\{r\}$ with $\{\delta\}$, we have

$$\{R\} = [T]^T\{r\} \tag{5.3}$$

and

$$[K] = [T]^T[k][T], \tag{5.4}$$

where $[K]$ and $[k]$ are the stiffness matrices corresponding to the skew-supported structure and the one with no skew support. The matrix equation for the skew supported structure can now be written as

$$[K]\{\Delta\} = \{R\}, \tag{5.5}$$

and the support condition that

$$\delta n_3 = 0 \tag{5.6}$$

can be directly applied therein to obtain the solution of the problem.

6 SETTING IDENTICAL DISPLACEMENT BOUNDARY CONDITIONS AT TWO OR MORE DISTINCT NODES

At times, an analyst may have to impose identical displacement boundary conditions at two or more nodes. The law of contragredient transformation can be applied in this case too. Let us consider the global displacement vector as given by

$$\{\delta\} = \begin{Bmatrix} \delta_a \\ \delta_b \\ \delta_c \\ \delta_d \\ \vdots \end{Bmatrix}. \tag{6.1}$$

Let us consider the case in which we need to set, say, $\delta_b = \delta_d$. Then, we define a new displacement vector

$$\{\delta^*\} = \begin{Bmatrix} \delta_a \\ \delta_b \\ \delta_c \\ \delta_b - \delta_d \\ \vdots \end{Bmatrix}. \tag{6.2}$$

The transformation rule connecting δ and δ^* is given by

$$\begin{Bmatrix} \delta_a \\ \delta_b \\ \delta_c \\ \delta_d \\ \vdots \end{Bmatrix} = \begin{bmatrix} I & 0 & 0 & 0 & \cdots \\ 0 & I & 0 & 0 & \\ 0 & 0 & I & 0 & \\ 0 & I & 0 & -I & \\ \vdots & & & & \ddots \end{bmatrix} \begin{Bmatrix} \delta_a \\ \delta_b \\ \delta_c \\ \delta_b - \delta_d \\ \vdots \end{Bmatrix}, \tag{6.3}$$

which can be written as

$$\{\delta\} = [T]\{\delta^*\}. \tag{6.4}$$

Hence, the transformation connecting the load vectors and the stiffness matrices are given by

$$\{R^*\} = [T]^T \{R\} \tag{6.5}$$

and

$$[K^*] = [T]^T [K][T]. \tag{6.6}$$

The finite element equations are set up as

$$[K^*]\{\delta^*\} = \{R^*\}. \tag{6.7}$$

Now, the solution of Eq. 6.7 with the degree of freedom corresponding to $\delta_b - \delta_d$ set equal to zero provides the solution of the problem with $\delta_b = \delta_d$.

It is interesting to note that the above procedure can be easily accommodated in a computer code by simply giving the same number for those degrees of freedom which are to have identical values. This is done by appropriately modifying the destination array.

7 ANALYSIS OF SYMMETRIC STRUCTURES

There are several interesting cases of symmetric structures. Their analyses can be carried out with great simplifications if we observe the symmetry and set up an appropriate transformation to make use of the symmetry to the best advantage. We shall consider one type of a symmetric structure known as the "cyclic symmetry" wherein a portion of the structure repeats periodically in the θ-direction (in a polar coordinate system setting).

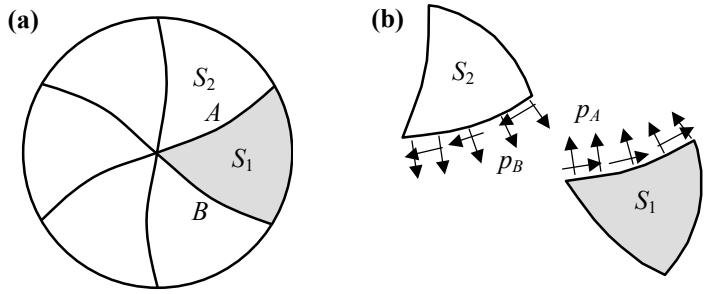

Figure 7.1 (a) A structure with cyclic symmetry. Note the "*A*" boundary of one repetitive segment is the same as the "*B*" boundary of the next. **(b)** Two adjacent segments S_1, S_2, and the interacting forces between them.

Consider the structure shown in Fig. 7.1 which has the above-mentioned cyclic symmetry. It is possible, in this case, to just analyse the shaded portion only to obtain the complete information about the entire structure. Moreover, even for this shaded portion, there is some further simplification available, as we shall see.

The load-displacement relation for any of the symmetric segment (such as the one shown shaded in figure) can be written as

$$[K]\{\delta\} = \{f\}, \tag{7.1a}$$

which in expanded form appears as

$$\begin{bmatrix} K_{ii} & K_{iA} & K_{iB} \\ K_{Ai} & K_{AA} & K_{AB} \\ K_{Bi} & K_{BA} & K_{BB} \end{bmatrix} \begin{Bmatrix} \delta_i \\ \delta_A \\ \delta_B \end{Bmatrix} = \begin{Bmatrix} f_i \\ f_A \\ f_B \end{Bmatrix}. \tag{7.1b}$$

The subscript i in the above denotes the internal degrees of freedom of the segment, and A and B denote the degrees of freedom along the boundaries A and B respectively.

The load vector $\{f\}$ can be resolved into two—the externally applied forces $\{q\}$ and the internally developed interacting forces $\{p\}$. Thus, we have

$$\begin{Bmatrix} f_i \\ f_A \\ f_B \end{Bmatrix} = \begin{Bmatrix} q_i \\ q_A \\ q_B \end{Bmatrix} + \begin{Bmatrix} 0 \\ p_A \\ p_B \end{Bmatrix}. \tag{7.2}$$

For compatibility between adjacent segments, we have

$$\{\delta_A\} = \{\delta_B\} \tag{7.3}$$

and for the equilibrium of the forces at the interface, we have

$$\{p_A\} + \{p_B\} = \{0\}. \tag{7.4}$$

Let us replace the original displacement vector $\{\delta\}$ of Eq. 7.1 by $\{\delta^*\}$, which is given by

$$\{\delta^*\} = \begin{Bmatrix} \delta_i \\ \delta_A \\ \delta_A - \delta_B \end{Bmatrix}. \tag{7.5}$$

The transformation rule for the above is given by

$$\begin{Bmatrix} \delta_i \\ \delta_A \\ \delta_B \end{Bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & I & -I \end{bmatrix} \begin{Bmatrix} \delta_i \\ \delta_A \\ \delta_A - \delta_B \end{Bmatrix} \tag{7.6a}$$

or

$$\{\delta\} = [T]\{\delta^*\}. \tag{7.6b}$$

Now, from the law of contragredient transformation, it follows that the load vector transforms as

$$\{f^*\} = [T]^T \{f\}. \tag{7.7}$$

That is

$$\begin{Bmatrix} f_i{}^* \\ f_A{}^* \\ f_B{}^* \end{Bmatrix} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & I \\ 0 & 0 & -I \end{bmatrix} \begin{Bmatrix} q_i \\ q_A + p_A \\ q_B + p_B \end{Bmatrix} = \begin{Bmatrix} q_i \\ q_A + q_B \\ -q_B - p_B \end{Bmatrix}, \tag{7.8}$$

wherein the equilibrium relations given by Eq. 7.4 has been used. The new force-displacement relations can be written as

$$\{f^*\} = [K^*]\{\delta^*\}, \tag{7.9}$$

which in expanded form appears as

$$\begin{Bmatrix} q_i \\ q_A + q_B \\ -q_B - p_B \end{Bmatrix} = \begin{bmatrix} K_{ii}^* & K_{iA}^* & K_{iB}^* \\ K_{Ai}^* & K_{AA}^* & K_{AB}^* \\ K_{Bi}^* & K_{BA}^* & K_{BB}^* \end{bmatrix} \begin{Bmatrix} \delta_i \\ \delta_A \\ \delta_A - \delta_B \end{Bmatrix}. \tag{7.10}$$

Now, the compatibility relations as given by Eq. 7.3 can be applied in the above. After simplification (by deleting the rows and columns corresponding to $\{\delta_A\}-\{\delta_B\} = 0$), we obtain

$$\begin{Bmatrix} q_i \\ q_A + q_B \end{Bmatrix} = \begin{bmatrix} K_{ii}^* & K_{iA}^* \\ K_{Ai}^* & K_{AA}^* \end{bmatrix} \begin{Bmatrix} \delta_i \\ \delta_A \end{Bmatrix}. \tag{7.11}$$

Eq. 7.11 can be solved to obtain the degrees of freedom at the internal nodes (i.e., $\{\delta\}$) and along the boundary "A" (i.e., $\{\delta_A\}$). The stress field can thereafter be found for each of the elements.

It must be noted that the number of nodes, their locations and the directions of the degrees of freedom along the boundaries A and B must all match exactly for this procedure to be applicable.

8 SOME ASPECTS REGARDING FINITE ELEMENT MESH

Some of the ways of developing finite element mesh automatically and cares to be taken in laying an appropriate mesh are discussed in this section. The general rule to be remembered is that at points where we expect sharp variation of the field variable, we need to employ a fine mesh of elements. At points away from such sharp variations, we could use a coarse mesh. That the solution accuracy improves and the computational cost simultaneously shoots up with finer mesh are well known.

We shall first discuss how a simple mesh generation program can be developed.

12.8.1 Automatic Mesh Generation Programs

Most of the commercial finite element packages available in the market come with powerful mesh generators. At times, however, it may be far easier to develop simple mesh generation programs of our own which are problem specific. Here, we present some simple principles to be used in such a case.

One of the methods of generating mesh over an irregular area is to first lay a mesh over a square area (a cube in the case of a three-dimensional domain) and then invoke a mapping. It is possible to employ the same set of shape functions as in the case of isoparametric elements for this mapping, as we shall see.

LAYING A MESH OVER A SQUARE

We shall first see how a uniform mesh can be laid over a square area. Let us assume that we plan to use four-noded isoparametric quadrilateral elements. The input parameters here are simply the number of divisions along the horizontal and vertical directions (`nDiv_X` and `nDiv_Y`). Consider the square domain of size 2×2 shown in Fig. 8.1. The ξ- and η-coordinates vary between −1 and +1 so that the interpolation polynomials in terms of natural coordinates that we had seen in the last chapter can be used directly to carry out mapping to another domain of any desired shape. In the figure, we have used three divisions along the horizontal direction and four along the vertical. The elements and nodes are numbered as shown. It is easy to note the regular pattern of node and element numbers in such a simple mesh laid over a square domain. This pattern is used in the computer code given below to obtain the nodal coordinates and the element connectivity arrays automatically. It is needless to mention that it is possible to adopt any other orderly way of numbering the elements and nodes.

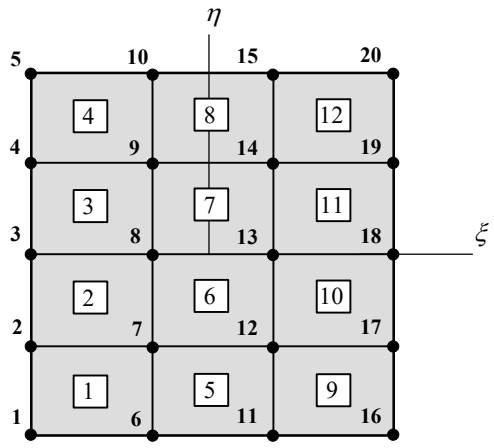

Figure 8.1 A simple mesh over a square domain of size 2×2. Here, `nDiv_X` = 3 and `nDiv_Y` = 4. Note the arrangement of the node and element numbers.

A simple program segment is presented below for this purpose. The reader will find this easy to understand.

```
// Program segment to generate nodal coordinates and
// element connectivity matrix

   nElems = nDiv_X * nDiv_Y;
   nNodes = (nDiv_X + 1)*(nDiv_Y + 1);

// Generating nodal coordinate data: The nodal coordinates
// of node 1 are (-1, -1) and the last node are (1,1)

   node = 0;
   double xx;
   for (j=1; j<=nDiv_X + 1; ++j)
   {
      xx = -1 + 2.0/nDiv_X*(j - 1);
      for (i=1; i<=nDiv_Y + 1; ++i)
      {
         node ++;
         x[node] = xx;
         y[node] = -1 + 2.0/nDiv_Y*(i - 1);
      }
   }
// Generate element connectivity
   // First element
   elemConn[1][1] = 1;
   elemConn[2][1] = nDiv_Y + 2;
   elemConn[3][1] = elemConn[2][1] + 1;
   elemConn[4][1] = 2;
   // First layer along left strip
   elem = 1;
   for (i=2; i<=nDiv_Y; ++i)
   {
      elem ++;
      for (j=1; j<=4; ++j)
         elemConn[j][elem] = elemConn[j][elem-1] + 1;
   }
   // Subsequent layers
   elem = nDiv_Y;
   for (i=2; i<=nDiv_X; ++i)
   {
      for (j=1; j<=nDiv_Y; ++j)
      {
         elem ++;
         for (k=1; k<=4; ++k)
            elemConn[k][elem] = elemConn[k][elem-nDiv_Y]
                                          + (nDiv_Y + 1);
      }
   }
```

Having obtained the nodal coordinates (in terms of the natural coordinates ξ and η) and the element connectivity for the square domain, the next task is to map the square into another domain of the desired shape. This is done by making use of the same set of interpolation polynomials we saw in the last chapter. Depending on the shape of the domain, we may use linear, quadratic or higher order interpolation polynomials. The nodal coordinates of any point in the physical domain (in terms of x- and y-coordinates) is calculated from that of the square domain (in terms of ξ and η) by the simple formulae

$$x_i = \sum_{i=1}^{N} N_i X_i \qquad 8.1a$$

and

$$y_i = \sum_{i=1}^{N} N_i Y_i \,. \qquad 8.1b$$

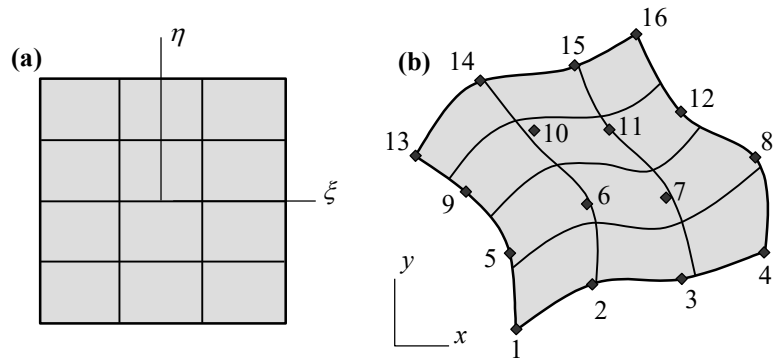

Figure 8.2 (a) The mesh generated over a square domain, and **(b)** the mesh mapped onto the physical domain. The nodes marked 1 to 16 are the key points which are used for the mapping.

In the above, X_i and Y_i are the coordinates of some of the key points in the physical domain. For the mesh shown in Fig. 8.2b, thus, we need to know only the *x*-and *y*-coordinates of the key points marked 1 to 16 in order to obtain the location of any of the finite element node in the physical domain. It is very easy to write a program to carry out the above. The following is a segment of a program which carries out the transformations as given by Eq. 8.1.

```
   int degree, nPoints;
// degree of interpolation for the mapping
   fin >> degree;
   nPoints = (degree+1)*(degree+1);

// Read Global Coordinates of the Key Points
   for (i=1; i<=nPoints; i++)
   {
      fin >> k;
      fin >> Xg[k] >> Yg[k];
   }
   nElems = nDiv_X*nDiv_Y;
// Transform the coordinates and print
   for (i=1; i<=nNodes; i++)
   {
      double pxi, pet;
      pxi = x[i];
      pet = y[i];
      shapeFunc(degree, pxi, pet);
      xNew[i] = yNew[i] = 0;
      for (j=1; j<=nPoints; j++)
      {
         xNew[i] += N[j]*Xg[j];
         yNew[i] += N[j]*Yg[j];
      }
   }
```

The function call to "shapeFunc" is used to obtain the Lagrangian shape functions of any degree. The listing of this function is given below.

```
void shapeFunc(int degree, double pxi, double pet)
{
   double xxi[10], Nxii[10], Neti[10];

   int i, j, k;
   xxi[1] = -1;
   for (i=2; i<=degree + 1; i++)
      xxi[i] = xxi[i-1] + 2.0/degree;

   for (i=1; i<=degree + 1; i++)
   {
      Nxii [i] = 1;
      Neti [i] = 1;
      for (j=1; j<=degree + 1; j++)
      {
         if (j != i)
```

```
         {
            Nxii [i] *= (pxi - xxi[j])/(xxi[i] - xxi[j]);
            Neti [i] *= (pet - xxi[j])/(xxi[i] - xxi[j]);
         }
      }
   }

   int node = 0;
   for (i=1; i<=degree + 1; i++)
   {
      for (j=1; j<=degree + 1; j++)
      {
         node ++;
         N[node] = Nxii[i] * Neti[j];
      }
   }
}
```

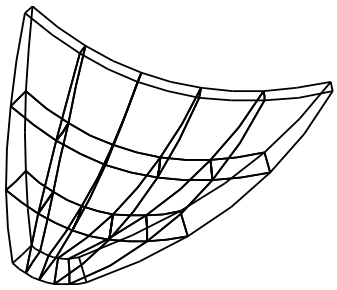

Figure 8.3 A three-dimensional finite element mesh generated using isoparametric mapping principles.

At times, we could use polynomials of different degrees in the two directions (say, `degree_X` and `degree_Y`). It is possible to extend the above idea into the case of a three-dimensional domain without much of a difficulty. We first generate the mesh within a cube of size 2×2×2 in natural coordinates ξ, η, ζ. Then, a mapping similar to the one given by Eqs. 8.1, now including the z-coordinate as well, leads to the required mesh over the three-dimensional domain of interest. One such mesh for a three-dimensional domain is shown in Fig. 8.3. The reader may extend the code given above for this purpose.

12.8.2 Element Connection and Grading

It is important to note that the interpolation polynomials that we use will make the element deform in a particular way. As a result, we cannot connect two neighbouring elements as shown in Fig. 8.4a or b. In Fig. 8.4a, although the two elements are quadratic, a point such as *P* located on the common boundary between the elements will deform differently for the two elements, giving rise to either a gap or an overlap. Such elements are known as incompatible elements, and are best avoided for most of the problems*.

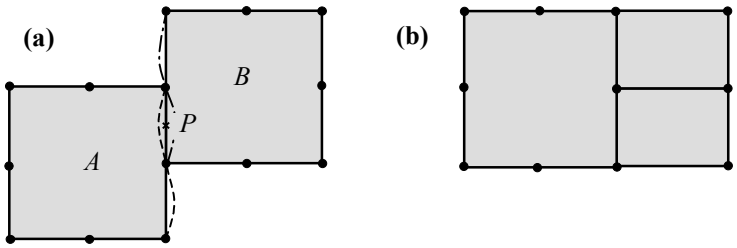

Figure 8.4 Incorrect connections: **(a)** Two quadratic elements wrongly connected, and **(b)** a quadratic element connected to two linear elements.

As pointed out earlier, it may become essential to grade the finite element mesh from fine to coarse depending on the expected variation of the field variable. Several common patterns are in use for this purpose. Fig. 8.5a and b show two possible ways of moving from a coarse mesh to a fine one.

* Of course, at times, incompatibility improves the accuracy as the excessive stiffening effect due to the assumed displacement in a displacement-based finite element method may cancel the extra compliance introduced due to the incompatibility.

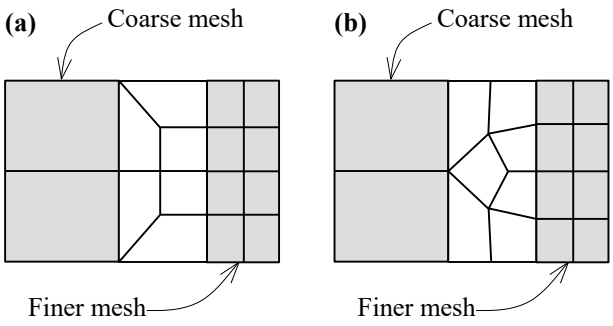

Figure 8.5 Grading the mesh from coarse to fine. Two different possible transitions shown in **(a)** and **(b)**.

Another problem facing an analyst may be how to discretise a circular domain using quadrilateral elements. One possible mesh for a quarter of a circle is shown in Fig. 8.6.

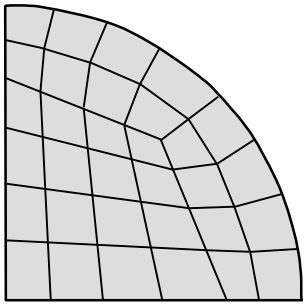

Figure 8.6 A possible finite element mesh for a quarter of a circle.

SUMMARY

Let us list down the important points that we have seen in this chapter before we close our discussion on the finite element method.

- We started the chapter with the derivation of the law of contragredient transformation.
- We presented static condensation next. We pointed out that static condensation involves no approximation whatsoever. We also showed that condensation could be carried out using a simple transformation. In computer implementation, condensation is best effected by means of a modified form of Gauss elimination, without any rearrangement of the degrees of freedom.
- We described substructuring as a useful means to analyse bodies with very large number of degrees of freedom. Essentially, condensation is carried out for each of the substructure, thus eliminating the internal degrees of freedom altogether. Each substructure acts like a "macro" element.
- We demonstrated how the case of skew supports could be handled as a transformation problem, and how it leads to a simple solution.
- We showed that it is possible to prescribe the displacements of two nodes to have identical values. Although mathematically this amounts to the use of the contragredient transformation law, we indicated that it is easily incorporated in computer coding by manipulating the destination array. Those degrees of freedom, which are to have identical displacements, are numbered the same.
- We presented a simple way of analysing structures having cyclic symmetry. Here also, we saw the contragredient law at play.
- Before closing the chapter, we discussed some important aspects regarding laying a finite element mesh. We indicated how not to connect two neighbouring finite elements. We presented one simple way to automatically generate finite element meshes. We also indicated some ways of grading a mesh from coarse to fine.

With this chapter, we come to the end of our discussion on the finite element method. The three chapters (Chapters 10 to 12) have not been exhaustive. The reader may refer to some of the popular books on the topic given below for further study.

SOME OF THE POPULAR BOOKS ON THE FINITE ELEMENT METHOD

1. Cook, R.D., Malkus, D.S., Plesha, M.E., and Witt, R.J., *Concepts and Applications of Finite Element Analysis*, John Wiley, New York, 2003. (An excellent book to both beginners and users of the finite element method, very well written).

2. Bathe, K.-J., *Finite Element Procedures*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1996. (A very good reference book especially for problems related to structural dynamics. An exhaustive book, which is a bit hard to read at places).
3. Bathe, K.-J. and Wilson, E.L., *Numerical Methods in Finite Element Analysis*, Prentice-Hall, Englewood Cliffs, New Jersey, 1976. (A much smaller book in comparison to the above; contains almost all materials related to dynamics as in the above).
4. Zienkiewicz, O.C., and Taylor, R.L., *The Finite Element Method*, Volumes I to III, McGraw-Hill, London, 2000. (An excellent reference book).
5. Zienkiewicz, O.C., and Morgan, K., *Finite Elements and Approximation*, John Wiley, New York, 1983. (A good book on weighted residual approach).
6. Szabo, B., and Babuska, I., *Finite Element Analysis*, John Wiley, New York, 1991.
7. Rao, S.S., *The Finite Element Method in Engineering*, Pergamon Press, New York, 1982.
8. Shames, I.H., and Dym, C.L., *Energy and Finite Element Methods in Structural Mechanics*, McGraw Hill, New York, 1985.
9. Huebner, K.H, Thornton, E.A., and Byrom, T.G., *The Finite Element Method*, John Wiley, New York, 1995.
10. Hughes, T.J.R., *The Finite Element Method*, Prentice Hall Inc., Englewood Cliffs, New Jersey, 1987. (Rigorous mathematical treatment).
11. Reddy, J.N., *An Introduction to the Finite Element Method*, McGraw Hill, New York, 1993.
12. Krishnamoorthy, C.S., *Finite Element Analysis*, McGraw Hill, Tata India, 1994.
13. Young, W.K., and Hyochoong, B., *Finite Element Method using MATLAB*, CRC Press, London, 2000. (An interesting book in which the authors show how MATLAB could be used to develop easy computer codes for the finite element analysis).
14. Kardestuncer, H., and Norrie, D.H., *Finite Element Handbook*, McGraw-Hill, New York, 1987.