The Finite Element Method using MATLAB

Young W. Kwon
Hyochoong Bang

CRC Press
Boca Raton London New York Washington, D.C.

CHAPTER SIX

ISOPARAMETRIC ELEMENTS

6.1 One-Dimensional Elements

Isoparametric elements [10] use mathematical mapping from one coordinate system into the other coordinate system. The former coordinate system is called the *natural* coordinate system while the latter is called the *physical* coordinate system. The problem domain is provided in the *physical* coordinate system denoted xyz-axes in the following discussion. On the other hand, element shape functions are defined in terms of the *natural* coordinate system denoted $\xi\eta\zeta$-axes. As a result, mapping is needed between the two coordinate systems.

We consider a linear one-dimensional isoparametric element to discuss the basic characteristics of isoparametric elements. Multi-dimensional isoparametric elements will be discussed in the subsequent sections. Shape functions for the isoparametric element are given in terms of the *natural* coordinate system as seen in Fig. 6.1.1. The two nodes are located at $\xi_1 = -1.0$ and $\xi_2 = 1.0$. These nodal positions are arbitrary but the proposed selection is very useful for numerical integration because the element in the *natural* coordinate system is normalized between -1 and 1. The shape functions can be written as

$$H_1(\xi) = \frac{1}{2}(1-\xi) \tag{6.1.1}$$

and

$$H_2(\xi) = \frac{1}{2}(1+\xi) \tag{6.1.2}$$

The physical linear element may be located at any position in the *physical* coordinate system as shown in Fig. 6.1.2. The element has two nodal coordinate values x_1 and x_2 with corresponding nodal variables u_1 and u_2.

Any point between $\xi_1 = -1.0$ and $\xi_2 = 1.0$ in the *natural* coordinate system can be mapped onto a point between x_1 and x_2 in the *physical* coordinate system using the shape functions defined in Eqs (6.1.1) and (6.1.2).

$$x = H_1(\xi)x_1 + H_2(\xi)x_2 \tag{6.1.3}$$

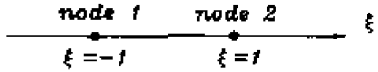

Figure 6.1.1 Linear Element In the Natural Coordinate

Figure 6.1.2 Linear Element in the Physical Coordinate

The same shape functions are also used to interpolate the variable u within the element.

$$u = H_1(\xi)u_1 + H_2(\xi)u_2 \tag{6.1.4}$$

If the same shape functions are used for the geometric mapping as well as nodal variable interpolation such as Eqs (6.1.3) and (6.1.4), the element is called the *isoparametric* element.

In order to compute $\frac{du}{dx}$, which is necessary in most cases to compute element matrices, we use the chain rule such that

$$\begin{aligned}\frac{du}{dx} &= \frac{dH_1(\xi)}{dx}u_1 + \frac{dH_2(\xi)}{dx}u_2 \\ &= \frac{dH_1(\xi)}{d\xi}\frac{d\xi}{dx}u_1 + \frac{dH_2(\xi)}{d\xi}\frac{d\xi}{dx}u_2\end{aligned} \tag{6.1.5}$$

where the expression requires $\frac{d\xi}{dx}$ which is the inverse of $\frac{dx}{d\xi}$. The latter can be computed from Eq. (6.1.3).

$$\frac{dx}{d\xi} = \frac{dH_1(\xi)}{d\xi}x_1 + \frac{dH_2(\xi)}{d\xi}x_2 = \frac{1}{2}(x_2 - x_1) \tag{6.1.6}$$

Substituting Eq. (6.1.6) into Eq. (6.1.5) yields

$$\frac{du}{dx} = -\frac{1}{x_2 - x_1}u_1 + \frac{1}{x_2 - x_1}u_2 \tag{6.1.7}$$

As a result, derivatives of shape functions with respect to the *physical* coordinate system are

$$\frac{dH_1(\xi)}{dx} = -\frac{1}{x_2 - x_1} = -\frac{1}{h_i} \tag{6.1.8}$$

$$\frac{dH_2(\xi)}{dx} = \frac{1}{x_2 - x_1} = \frac{1}{h_i} \tag{6.1.9}$$

in which $h_i = (x_2 - x_1)$ is the element size in the *physical* coordinate system. These derivative values are identical to those obtained directly from the linear shape functions expressed in terms of the *physical* coordinate system like Eqs (2.4.7) and (2.4.8).

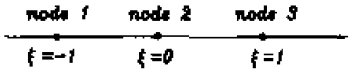

Figure 6.1.3 Quadratic Isoparametric Element

Let us compute the following integral using the linear isoparametric element.

$$\int_{x_1}^{x_2}\left(\frac{dw}{dx}\frac{du}{dx} + wu\right)dx \tag{6.1.10}$$

The integration is in terms of the *physical* coordinate system while the integrand is expressed in terms of the *natural* coordinate system because isoparametric shape functions are used for the trial and test functions u and w. Hence, we want to write the integral in terms of the *natural* coordinate system. To this end, we obtain

$$\int_{-1}^{1}\left(\frac{dw}{dx}\frac{du}{dx} + wu\right)J\,d\xi \tag{6.1.11}$$

where $J = \frac{dx}{d\xi}$ is called the *Jacobian*.

Substitution of the isoparametric shape functions for both u and w results in

$$\begin{aligned}&\int_{-1}^{1}\left(\frac{1}{h_i^2}\begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix} + \frac{1}{4}\begin{bmatrix}(1-\xi)^2 & (1-\xi^2)\\ (1-\xi^2) & (1+\xi)^2\end{bmatrix}\right)\frac{h_i}{2}d\xi\begin{Bmatrix}u_1\\ u_2\end{Bmatrix} \\ &= \begin{bmatrix}\frac{1}{h_i} + \frac{h_i}{3} & -\frac{1}{h_i} + \frac{h_i}{6}\\ -\frac{1}{h_i} + \frac{h_i}{6} & \frac{1}{h_i} + \frac{h_i}{3}\end{bmatrix}\begin{Bmatrix}u_1\\ u_2\end{Bmatrix}\end{aligned} \tag{6.1.12}$$

This expression is the same as that obtained from the conventional linear element.

At this point, the isoparametric element does not seem to have an advantage over the conventional element because the isoparametric element requires more procedures such as mapping and chain rule. The major advantage of isoparametric elements comes when analytical integration to compute element matrices and column vectors is either very complicated or almost impossible. This is the case either element shapes in the physical domain are not regular such as in the multi-dimensional problem or the differential equation is quite complex. Therefore, the numerical integration technique is needed. Because each isoparametric element is defined in terms of the normalized domain such as $\xi_1 = -1$ and $\xi_2 = 1$, it is much easier to apply any numerical integration technique. The application of numerical integration technique is discussed later in this chapter.

♣ **Example 6.1.1** Let us consider a quadratic one-dimensional isoparametric element as seen in Fig 6.1.3. Shape functions for this element are

$$H_1(\xi) = \frac{(\xi^2 - \xi)}{2} \tag{6.1.13}$$

$$H_2(\xi) = 1 - \xi^2 \tag{6.1.14}$$

and

$$H_3(\xi) = \frac{(\xi^2 + \xi)}{2} \tag{6.1.15}$$

The variable u can be interpolated using these shape functions.

$$u = H_1(\xi)u_1 + H_2(\xi)u_2 + H_3(\xi)u_3 \tag{6.1.16}$$

Geometric mapping from the *natural* coordinate to the *physical* coordinate is

$$x = H_1(\xi)x_1 + H_2(\xi)x_2 + H_3(\xi)x_3 \tag{6.1.17}$$

The *Jacobian* becomes

$$J = \frac{dx}{d\xi} = \sum_{i=1}^{3} \frac{dH_i(\xi)}{d\xi} x_i = (\xi - 0.5)x_1 - 2\xi x_2 + (\xi + 0.5)x_3 \tag{6.1.18}$$

If the mid-node x_2 is located between the two end-nodes x_1 and x_3 (i.e. $x_2 = \frac{(x_1+x_3)}{2}$), the *Jacobian* becomes $\frac{h_i}{2}$ in which $h_i = x_3 - x_1$ is the element length.

Derivatives of the shape functions, Eqs (6.1.13) through (6.1.15), are

$$\frac{dH_1(\xi)}{dx} = \frac{1}{J}\frac{dH_1}{d\xi} = \frac{1}{h_i}(2\xi - 1) \tag{6.1.19}$$

$$\frac{dH_2(\xi)}{dx} = \frac{1}{J}\frac{dH_2}{d\xi} = -\frac{4\xi}{h_i} \tag{6.1.20}$$

$$\frac{dH_3(\xi)}{dx} = \frac{1}{J}\frac{dH_3}{d\xi} = \frac{1}{h_i}(2\xi + 1) \tag{6.1.21}$$

‡

6.2 Quadrilateral Elements

The shape functions for the bilinear isoparametric element are given below:

$$H_1(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 - \eta) \tag{6.2.1}$$

$$H_2(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 - \eta) \tag{6.2.2}$$

$$H_3(\xi, \eta) = \frac{1}{4}(1 + \xi)(1 + \eta) \tag{6.2.3}$$

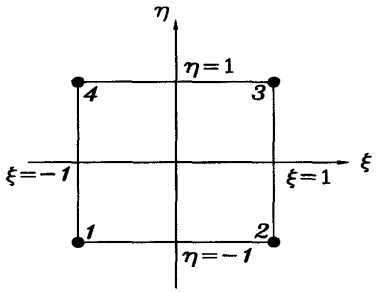

Figure 6.2.1 Bilinear Element in the Natural Coordinate

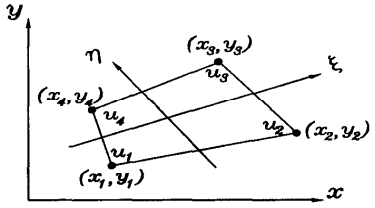

Figure 6.2.2 Bilinear Element in the Physical Coordinate

$$H_4(\xi, \eta) = \frac{1}{4}(1 - \xi)(1 + \eta) \tag{6.2.4}$$

for the nodes shown in Fig. 6.2.1. These shape functions are defined in terms of the normalized *natural* domain (i.e. $-1 \leq \xi \leq 1$ and $-1 \leq \eta \leq 1$).

While the element shape is a square in the *natural* coordinate system, it can be mapped into a general quadrilateral shape with distortion as seen in Fig. 6.2.2. When this mapping is undertaken, the relative positions of nodal points should be consistent between the two elements in the *natural* and *physical* domains. In other words, the second node is next to the first node in the counter-clockwise direction and similarly for the rest of the nodes. Then, a point (ξ, η) within the *natural* element is mapped into a point (x, y) within the *physical* element using the shape functions given in Eqs (6.2.1) through (6.2.4) as shown below:

$$x = \sum_{i=1}^{4} H_i(\xi, \eta)x_i \tag{6.2.5}$$

$$y = \sum_{i=1}^{4} H_i(\xi, \eta)y_i \tag{6.2.6}$$

in which x_i and y_i are the coordinate values of the i^{th} node. Similarly, any physical variable can be interpolated using the same shape functions.

$$u = \sum_{i=1}^{4} H_i(\xi,\eta) u_i \tag{6.2.7}$$

in which u_i is the nodal variable at node i.

Let us apply this bilinear isoparametric element to the *Laplace* equation discussed in Chapter 5. Then, we need to compute $\frac{\partial H_i(\xi,\eta)}{\partial x}$ and $\frac{\partial H_i(\xi,\eta)}{\partial y}$, respectively. In order to compute these derivatives, we use the chain rule, again.

$$\frac{\partial}{\partial \xi} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \xi} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \xi} \tag{6.2.8}$$

$$\frac{\partial}{\partial \eta} = \frac{\partial}{\partial x}\frac{\partial x}{\partial \eta} + \frac{\partial}{\partial y}\frac{\partial y}{\partial \eta} \tag{6.2.9}$$

Rewriting these in the matrix form provides

$$\begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{Bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{Bmatrix} \tag{6.2.10}$$

Here, the derivative shown in the left-hand-side column vector is called local derivative while that in the right-hand-side column vector is called global derivative. Furthermore, the square matrix in this equation is called the *Jacobian* matrix for the two-dimensional domain and denoted as

$$[J] = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial y}{\partial \xi} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} \tag{6.2.11}$$

The *Jacobian* matrix can be easily extended for the three-dimensional domain.

Inverse of the *Jacobian* matrix is denoted by

$$[R] = [J]^{-1} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \tag{6.2.12}$$

Then, Eq. (6.2.10) can be rewritten as

$$\begin{Bmatrix} \frac{\partial}{\partial x} \\ \frac{\partial}{\partial y} \end{Bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \begin{Bmatrix} \frac{\partial}{\partial \xi} \\ \frac{\partial}{\partial \eta} \end{Bmatrix} \tag{6.2.13}$$

As a result, the derivatives of shape functions with respect to x and y can be obtained from the above equation.

$$\begin{Bmatrix} \frac{\partial H_i}{\partial x} \\ \frac{\partial H_i}{\partial y} \end{Bmatrix} = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \begin{Bmatrix} \frac{\partial H_i}{\partial \xi} \\ \frac{\partial H_i}{\partial \eta} \end{Bmatrix} \tag{6.2.14}$$

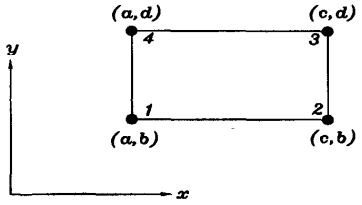

Figure 6.2.3 Rectangular Element

The components in the *Jacobian* matrix are computed as shown below:

$$J_{11} = \frac{\partial x}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial H_i(\xi,\eta)}{\partial \xi} x_i \tag{6.2.15}$$

$$J_{12} = \frac{\partial y}{\partial \xi} = \sum_{i=1}^{4} \frac{\partial H_i(\xi,\eta)}{\partial \xi} y_i \tag{6.2.16}$$

$$J_{21} = \frac{\partial x}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial H_i(\xi,\eta)}{\partial \eta} x_i \tag{6.2.17}$$

$$J_{22} = \frac{\partial y}{\partial \eta} = \sum_{i=1}^{4} \frac{\partial H_i(\xi,\eta)}{\partial \eta} y_i \tag{6.2.18}$$

Substitution of bilinear shape functions, Eqs (6.2.1) through (6.2.4), into the above expressions yields

$$J_{11} = -\frac{1}{4}(1-\eta)x_1 + \frac{1}{4}(1-\eta)x_2 + \frac{1}{4}(1+\eta)x_3 - \frac{1}{4}(1+\eta)x_4 \tag{6.2.19}$$

$$J_{12} = -\frac{1}{4}(1-\eta)y_1 + \frac{1}{4}(1-\eta)y_2 + \frac{1}{4}(1+\eta)y_3 - \frac{1}{4}(1+\eta)y_4 \tag{6.2.20}$$

$$J_{21} = -\frac{1}{4}(1-\xi)x_1 - \frac{1}{4}(1+\xi)x_2 + \frac{1}{4}(1+\xi)x_3 + \frac{1}{4}(1-\xi)x_4 \tag{6.2.21}$$

$$J_{22} = -\frac{1}{4}(1-\xi)y_1 - \frac{1}{4}(1+\xi)y_2 + \frac{1}{4}(1+\xi)y_3 + \frac{1}{4}(1-\xi)y_4 \tag{6.2.22}$$

These components are in general a function of ξ and η. However, they may be constant for a special case as shown in the following example. Once the *Jacobian* matrix is computed from Eqs (6.2.19) through (6.2.22), global derivatives of shape functions are computed as

$$\frac{\partial H_i(\xi,\eta)}{\partial x} = R_{11}\frac{\partial H_i(\xi,\eta)}{\partial \xi} + R_{12}\frac{\partial H_i(\xi,\eta)}{\partial \eta} \tag{6.2.23}$$

$$\frac{\partial H_i(\xi,\eta)}{\partial y} = R_{21}\frac{\partial H_i(\xi,\eta)}{\partial \xi} + R_{22}\frac{\partial H_i(\xi,\eta)}{\partial \eta} \tag{6.2.24}$$

♣ **Example 6.2.1** Let us compute the *Jacobian* matrix for the physical element shown in Fig. 6.2.3. Substituting the nodal coordinate values into Eqs (6.2.19) through (6.2.22) yields the following matrix.

$$[J] = \begin{bmatrix} \frac{c-a}{2} & 0 \\ 0 & \frac{d-b}{2} \end{bmatrix} \tag{6.2.25}$$

As seen in this example, the *Jacobian* matrix becomes a diagonal matrix (i.e. all off-diagonal components vanish) when the element in the *physical* domain is a rectangular shape. In addition, the diagonal components are constant not a function of ξ and η.

The inverse of the *Jacobian* matrix becomes

$$[R] = \begin{bmatrix} \frac{2}{c-a} & 0 \\ 0 & \frac{2}{d-b} \end{bmatrix} \tag{6.2.26}$$

The global derivatives of shape functions become

$$\frac{\partial H_1}{\partial x} = -\frac{1-\eta}{2(c-a)} \tag{6.2.27}$$

$$\frac{\partial H_2}{\partial x} = \frac{1-\eta}{2(c-a)} \tag{6.2.28}$$

$$\frac{\partial H_3}{\partial x} = \frac{1+\eta}{2(c-a)} \tag{6.2.29}$$

$$\frac{\partial H_4}{\partial x} = -\frac{1+\eta}{2(c-a)} \tag{6.2.30}$$

$$\frac{\partial H_1}{\partial y} = -\frac{1-\xi}{2(d-b)} \tag{6.2.31}$$

$$\frac{\partial H_2}{\partial y} = -\frac{1+\xi}{2(d-b)} \tag{6.2.32}$$

$$\frac{\partial H_3}{\partial y} = \frac{1+\xi}{2(d-b)} \tag{6.2.33}$$

$$\frac{\partial H_4}{\partial y} = \frac{1-\xi}{2(d-b)} \tag{6.2.34}$$

‡

♣ **Example 6.2.2** Let us compute the following integral using the same element as given in Example 6.2.1.

$$\int_{\Omega^e} \left[\left(\frac{\partial H_1}{\partial x}\right)^2 + \left(\frac{\partial H_1}{\partial y}\right)^2 \right] dxdy \tag{6.2.35}$$

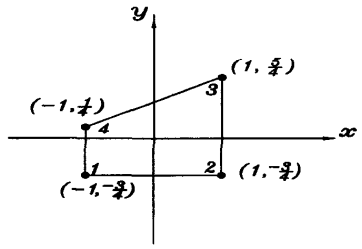

Figure 6.2.4 Element of Trapezoidal Shape

Substitution of Eqs (6.2.27) and (6.2.31) into Eq. (6.2.35) results in

$$\int_b^d \int_a^c \left[\frac{1}{4(c-a)^2}(1-\eta)^2 + \frac{1}{4(d-b)^2}(1-\xi)^2 \right] dxdy \tag{6.2.36}$$

The lower and upper limits of the integrals can be changed using

$$dxdy = |J| d\xi d\eta \tag{6.2.37}$$

where $|J|$ is the determinant of the *Jacobian* matrix and is equal to $\frac{(c-a)(d-b)}{4}$ for the present element. That is, $|J|$ is a constant value for a rectangular shape of *physical* element. Then, we obtain

$$\int_{-1}^1 \int_{-1}^1 \left[\frac{1}{4(c-a)^2}(1-\eta)^2 + \frac{1}{4(d-b)^2}(1-\xi)^2 \right] \frac{(c-a)(d-b)}{4} d\xi d\eta \tag{6.2.38}$$

Integration of Eq. (6.2.37) finally yields $\frac{(c-a)^2+(d-b)^2}{3(c-a)(d-b)}$. ‡

♣ **Example 6.2.3** Fine the *Jacobian* matrix for the quadrilateral element shown in Fig. 6.2.4. Equations (6.2.19) through (6.2.22) along with the nodal coordinate values as specified in the figure yield

$$[J] = \begin{bmatrix} 1 & \frac{1}{4}(1+\eta) \\ 0 & \frac{1}{4}(3+\xi) \end{bmatrix} \tag{6.2.39}$$

The determinant of *Jacobian* is $|J| = \frac{3+\xi}{4}$ which is always positive for $-1 \le \xi \le 1$. Inverting the matrix gives

$$[R] = \begin{bmatrix} 1 & -\frac{1+\eta}{3+\xi} \\ 0 & \frac{4}{3+\xi} \end{bmatrix} \tag{6.2.40}$$

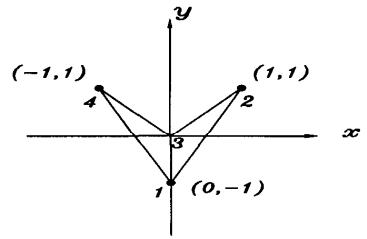

Figure 6.2.5 Element of Quadrilateral Shape

The determinant of the *Jacobian* matrix is $|J| = \frac{1}{4}(3+\xi)$.

In order to compute the integration as given in Eq. (6.2.35) for the present element, we first compute

$$\frac{\partial H_1}{\partial x} = -\frac{1-\eta}{4} + \frac{(1-\xi)(1+\eta)}{4(3+\xi)} \tag{6.2.41}$$

$$\frac{\partial H_1}{\partial y} = \frac{\xi-1}{3+\xi} \tag{6.2.42}$$

The expression for the integral becomes

$$\int_{-1}^{1}\int_{-1}^{1}\left[\left\{-\frac{1-\eta}{4} + \frac{(1-\xi)(1+\eta)}{4(3+\xi)}\right\}^2 + \left\{\frac{\xi-1}{3+\xi}\right\}^2\right]\frac{(3+\xi)}{4}d\xi d\eta \tag{6.2.43}$$

This integral can be conducted analytically. However, if the shape of the *physical* element has more severe distortion, the integral becomes more complicated and may be beyond the analytical computation. Even if analytical integration may be possible, performing the analytical integration for every element of different shape is not pratically possible. Therefore, the numerical integration technique is used along with the isoparametric element. ‡

♣ **Example 6.2.4** The *physical* element has a severe distortion as seen in Fig. 6.2.5. The corresponding *Jacobian* matrix is

$$[J] = \begin{bmatrix} \frac{1}{2} & \frac{1-3\eta}{4} \\ -\frac{1}{2} & \frac{1-3\xi}{4} \end{bmatrix} \tag{6.2.44}$$

and its determinant is $\frac{1}{8}(2-3\xi-3\eta)$. This determinant can be zero or negative for $-1 \le \xi \le 1$ and $-1 \le \eta \le 1$. Hence, this shape of element should be avoided in discretizing the *physical* domain. ‡

Some other popular quadrilateral isoparametric elements are eight-node and nine-node elements as shown in Fig. 6.2.6. Their shape functions are given below.

Eight-node element:

$$H_1 = \frac{1}{4}(1-\xi)(1-\eta)(-1-\xi-\eta) \tag{6.2.45}$$

$$H_2 = \frac{1}{4}(1+\xi)(1-\eta)(-1+\xi-\eta) \tag{6.2.46}$$

$$H_3 = \frac{1}{4}(1+\xi)(1+\eta)(-1+\xi+\eta) \tag{6.2.47}$$

$$H_4 = \frac{1}{4}(1-\xi)(1+\eta)(-1-\xi+\eta) \tag{6.2.48}$$

$$H_5 = \frac{1}{2}(1-\xi^2)(1-\eta) \tag{6.2.49}$$

$$H_6 = \frac{1}{2}(1+\xi)(1-\eta^2) \tag{6.2.50}$$

$$H_7 = \frac{1}{2}(1-\xi^2)(1+\eta) \tag{6.2.51}$$

$$H_8 = \frac{1}{2}(1-\xi)(1-\eta^2) \tag{6.2.52}$$

Nine-node element:

$$H_1 = \frac{1}{4}(\xi^2-\xi)(\eta^2-\eta) \tag{6.2.53}$$

$$H_2 = \frac{1}{4}(\xi^2+\xi)(\eta^2-\eta) \tag{6.2.54}$$

$$H_3 = \frac{1}{4}(\xi^2+\xi)(\eta^2+\eta) \tag{6.2.55}$$

$$H_4 = \frac{1}{4}(\xi^2-\xi)(\eta^2+\eta) \tag{6.2.56}$$

$$H_5 = \frac{1}{2}(1-\xi^2)(\eta^2-\eta) \tag{6.2.57}$$

$$H_6 = \frac{1}{2}(\xi^2+\xi)(1-\eta^2) \tag{6.2.58}$$

$$H_7 = \frac{1}{2}(1-\xi^2)(\eta^2+\eta) \tag{6.2.59}$$

$$H_8 = \frac{1}{2}(\xi^2-\xi)(1-\eta^2) \tag{6.2.60}$$

$$H_9 = (1-\xi^2)(1-\eta^2) \tag{6.2.61}$$

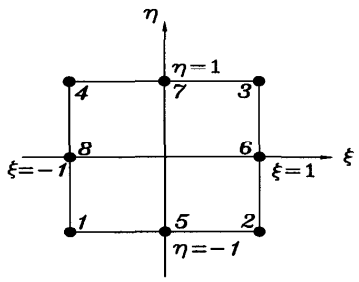

Figure 6.2.6 Eight-Node Isoparametric Element

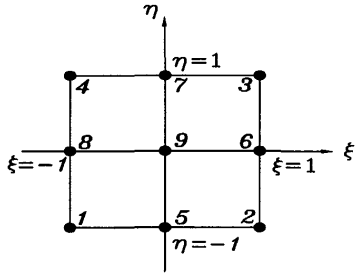

Figure 6.2.7 Nine-Node Isoparametric Element

6.3 Triangular Elements

Like quadrilateral isoparametric elements, triagular isoparametric elements can be defined. Shape functions of the linear triangular element are in terms of the *natural* coordinate system

$$H_1 = 1 - \xi - \eta \tag{6.3.1}$$

$$H_2 = \xi \tag{6.3.2}$$

$$H_3 = \eta \tag{6.3.3}$$

for the nodes shown in Fig. 6.3.1. The quadratic triangular element has the following shape functions with reference to Fig. 6.3.2.

$$H_1 = (1 - \xi - \eta)(1 - 2\xi - 2\eta) \tag{6.3.4}$$

$$H_2 = \xi(2\xi - 1) \tag{6.3.5}$$

$$H_3 = \eta(2\eta - 1) \tag{6.3.6}$$

$$H_4 = 4\xi(1 - \xi - \eta) \tag{6.3.7}$$

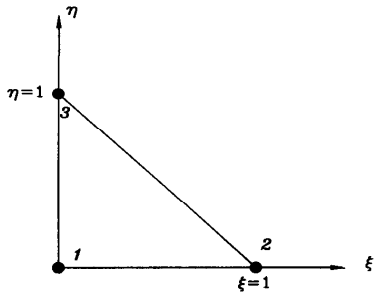

Figure 6.3.1 Three-Node Triangular Element in the Natural Coordinate

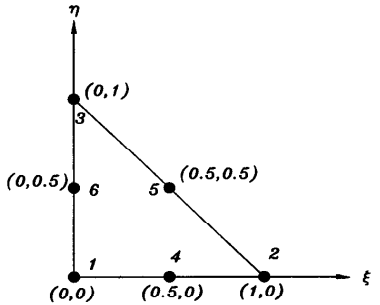

Figure 6.3.2 Six-Node Triangular Element in the Natural Coordinate

$$H_5 = 4\xi\eta \tag{6.3.8}$$

$$H_6 = 4\eta(1 - \xi - \eta) \tag{6.3.9}$$

♣ **Example 6.3.1** Consider an element as shown in Fig. 6.3.3. The *Jacobian* matrix for the element is

$$[J] = \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{bmatrix} \tag{6.3.10}$$

and its determinant is $|J| = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)$ which equals twice the triangular area in the *physical* domain. ‡

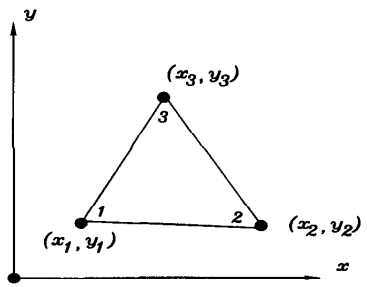

Figure 6.3.3 Three-Node Triangular Element in the Physical Coordinate

6.4 Gauss Quadrature

Integral is defined as

$$\int_a^b f(x)dx = \lim_{n\to\infty} \sum_{i=1}^{n} f(x_i)dx_i \tag{6.4.1}$$

This is shown in Fig. 6.4.1. In the numerical integration, we take a finite number of calculations. Therefore, Eq. (6.4.1) is approximated as

$$\int_a^b f(x)dx = \sum_{i=1}^{N} f(x_i)\Delta x_i \tag{6.4.2}$$

where N is a finite number. Rewriting this expression in a general way gives

$$\int_a^b f(x)dx = \sum_{i=1}^{M} f(x_i)W_i \tag{6.4.3}$$

in which M is the number of integration points, x_i is the integration point (or sampling point), and W_i is called the weighting coefficient. The weighting coefficient can be interpreted as the width of the rectangular strip whose height is $f(x_i)$ by comparing Eqs (6.4.2) and (6.4.3). Any numerical integration may be expressed in this form. In order to derive standard values for the integration points and weighting coefficients, the integration domain is normalized such that $-1 \le x \le 1$. Of course, there are other ways for normalization.

♣ **Example 6.4.1** Let us find the proper integration points and weighting coefficients for the two point trapezoidal rule. The trapezoidal rule gives

$$\int_{-1}^{1} g(\xi)d\xi = \big(g(-1) + g(1)\big) \tag{6.4.4}$$

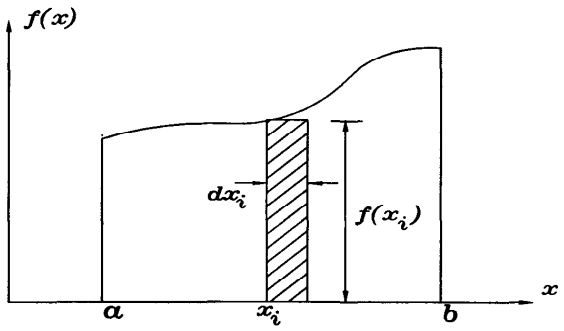

Figure 6.4.1 Integration

Comparing Eq. (6.4.4) to Eq. (6.4.3) indicates that the integration points for this case are $x_1 = -1$ and $x_2 = 1$ while the weighting coefficients are $W_1 = 1$ and $W_2 = 1$. ‡

♣ **Example 6.4.2** Repeat Example 6.4.1 using Simpson's $\frac{1}{3}$ rule with three point integration. This integration results in

$$\int_{-1}^{1} g(\xi)d\xi = \frac{1}{3}\big(g(-1) + 4g(0) + g(1)\big) \tag{6.4.5}$$

Therefore, we obtain $x_1 = -1$, $x_2 = 0$, $x_3 = 1$, $W_1 = \frac{1}{3}$, $W_2 = \frac{4}{3}$, and $W_3 = \frac{1}{3}$. ‡

Gauss-Legendre quadrature is very useful for integration of polynomial functions. It can integrate a polynomial function of order $2n - 1$ using the n-point quadrature exactly. Integration points and weighting coefficients for Gauss-Legendre quadrature are provided in Table 6.4.1. Similarly, Table 6.4.2 gives integration points and weighting coefficients for the triangular domain shown in Fig. 6.3.1 and Fig. 6.3.2. If the integrand is not a polynomial expression, Gauss-Legendre quadrature gives an approximate result. In this case, an optimal number of integration points should be selected in consideration of accuracy and computational cost. The next example shows how to determine the integration points and weighting coefficients for the Gauss-Legendre quadrature.

♣ **Example 6.4.3** This example shows a way how to compute the sampling points and weighting coefficients for Gauss-Legendre quadrature. Let us integrate

Table 6.4.1 Sampling points and weights in Gauss-Legendre numerical integration

n	*Int.Point*	Weight
1	0. 0.00000 00000 00000	2.00000 00000 00000
2	±0.57735 02691 89626	1.00000 00000 00000
3	±0.77459 66692 41483	0.55555 55555 55556
	0.00000 00000 00000	0.88888 88888 88889
4	±0.86113 63115 94053	0.34785 48451 37454
	±0.33998 10435 84856	0.65214 51548 62546
5	±0.90617 98459 38664	0.23692 68850 56189
	±0.53846 93101 05683	0.47862 86704 99366
	0.0000 00000 00000	0.56888 88888 88889
6	±0.93246 95142 03152	0.17132 44923 79170
	±0.66120 93864 66265	0.36076 15730 48139
	±0.23861 91860 83197	0.46791 39345 72691

a cubic polynomial as shown in Fig. 6.4.2. In Gauss-Legendre quadrature, we want to make the integration of the cubic polynomial the same as that of a linear function. In other words, the two different hatched areas in Fig. 6.4.2 are the same (i.e. Area(A)=Area(B) in Fig. 6.4.2). Then we can write

$$\int_{-1}^{1} f(x)dx = \int_{-1}^{1} g(x)dx = \sum_{s=1}^{2} W_s f(x_s) \tag{6.4.6}$$

where

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{6.4.7}$$

$$g(x) = c_0 + c_1 x \tag{6.4.8}$$

and W_s and x_s are the weighting coefficient and sampling point for the two point Gauss-Legendre quadrature because the two point rule integrates a cubic polynomial exactly.

Let us rewite the cubic polynomial in the following way.

$$f(x) = c_0 + c_1 x + (x - x_1)(x - x_2)(b_0 + b_1 x) \tag{6.4.9}$$

In this expression, x_1 and x_2 are fixed constants to be determined later. However, there are still four general constants c_0, c_1, b_0 and b_1 to be determined to make

Table 6.4.2 Numerical integrations over triangular domains

Int. order	ξ-coordinate	η-coordinate	Weight
	0.16666 66666 667	0.16666 66666 667	0.33333 33333 333
3-points	0.66666 66666 667	0.16666 66666 667	0.33333 33333 333
	0.16666 66666 667	0.66666 66666 667	0.33333 33333 333
	0.10128 65073 235	0.10128 65073 235	0.12593 91805 448
	0.79742 69853 531	0.10128 65073 235	0.12593 91805 448
	0.10128 65073 235	0.79742 69853 531	0.12593 91805 448
7-points	0.47014 20641 051	0.05971 58717 898	0.13239 41527 885
	0.47014 20641 051	0.47014 20641 051	0.13239 41528 885
	0.05971 58717 898	0.47014 20641 051	0.13239 41528 885
	0.33333 33333 333	0.33333 33333 333	0.22500 00000 000
	0.06513 01029 022	0.06513 01029 022	0.05334 72356 088
	0.86973 97941 956	0.06513 01029 022	0.05334 72356 088
	0.06513 01029 022	0.86973 97941 956	0.05334 72356 088
	0.31286 54960 049	0.04869 03154 253	0.07711 37608 903
	0.63844 41885 698	0.31286 54960 049	0.07711 37608 903
13-points	0.04869 03154 253	0.63844 41885 698	0.07711 37608 903
	0.63844 41885 698	0.04869 03154 253	0.07711 37608 903
	0.31286 54960 049	0.63844 41885 698	0.07711 37608 903
	0.04869 03154 253	0.04869 03154 253	0.07711 37608 903
	0.26034 59660 790	0.26034 59660 790	0.17561 52574 332
	0.47930 80678 419	0.26034 59660 790	0.17561 52574 332
	0.26034 59660 790	0.47930 80678 419	0.17561 52574 332
	0.33333 33333 333	0.33333 33333 333	-0.14957 00444 677

Eq. (6.4.9) the same as Eq. (6.4.7) for arbitrary constants a_i. Substitution of Eq. (6.4.9) into Eq. (6.4.6) states

$$\int_{-1}^{1} (x - x_1)(x - x_2)(b_0 + b_1 x)dx = 0 \tag{6.4.10}$$

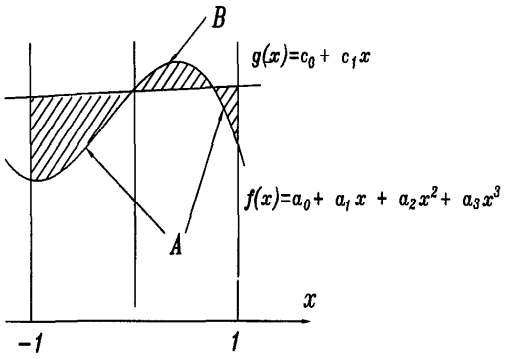

Figure 6.4.2 Two-Point Gauss-Legendre Quadrature

Equation (6.4.10) must be true independent of b_0 and b_1 because the integration rule holds for a general cubic polynomial. Therefore,

$$\int_{-1}^{1}(x-x_1)(x-x_2)dx = 0 \tag{6.4.11}$$

and

$$\int_{-1}^{1}x(x-x_1)(x-x_2)dx = 0 \tag{6.4.12}$$

These two equations determine $x_1 = -\frac{1}{\sqrt{3}}$ and $x_2 = \frac{1}{\sqrt{3}}$. These are two sampling points for the two-pont Gauss-Legendre quadrature. In order to find the corresponding weighting coefficients, we integrate

$$I = \int_{-1}^{1}(c_0 + c_1 x)dx = 2c_0 \tag{6.4.13}$$

From Eq. (6.4.6), this integration is equal to

$$\begin{aligned} I &= \sum_{s=1}^{2} W_s f(x_s) = W_1(c_0 + c_1 x_1) + W_2(c_0 + c_1 x_2) \\ &= c_0(W_1 + W_2) - \frac{1}{\sqrt{3}}c_1(W_1 - W_2) \end{aligned} \tag{6.4.14}$$

Equating Eq. (6.4.13) and Eq. (6.4.14) yields two weighting coefficients $W_1 = 1$ and $W_2 = 1$. ‡

♣ **Example 6.4.4** Perform the following integral:

$$\int_{-1}^{1}(1 + 2\xi + 3\xi^2)d\xi \tag{6.4.15}$$

Because the order of polynomial is 2, $2n - 1 = 2$. From this, we get $n = 1.5$. The number of integration points should be an integer. So, we use the two point quadrature rule. From Table 6.4.1, the two integration points are $-\frac{1}{\sqrt{3}}$ and $\frac{1}{\sqrt{3}}$ with weighting coefficient 1 for each point, respectively. The numerical integration becomes

$$\begin{aligned} &\int_{-1}^{1}(1 + 2\xi + 3\xi^2)dx = \\ &(1)\{1 + 2(-\frac{1}{\sqrt{3}}) + 3(-\frac{1}{\sqrt{3}})^2\} + (1)\{1 + 2(\frac{1}{\sqrt{3}}) + 3(\frac{1}{\sqrt{3}})^2\} = 4 \end{aligned} \tag{6.4.16}$$

This is the exact solution. If we use the three point quadrature to integrate Eq. (6.4.15) (i.e. $\xi_1 = -\frac{\sqrt{15}}{5}$, $\xi_2 = 0$, $\xi_1 = \frac{\sqrt{15}}{5}$, $W_1 = \frac{5}{9}$, $W_2 = \frac{8}{9}$, and $W_3 = \frac{5}{9}$), we also obtain the same exact solution. Therefore, the quadrature rule using two or higher number of integration points will always yield the exact solution for this problem. ‡

The quadrature rule can be extended for multi-dimensional integration. For example, numerical integral in the normalized two-dimensional domain can be conducted in the following way.

$$\begin{aligned} &\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta)d\xi d\eta \\ &= \int_{-1}^{1}\sum_{i=1}^{M_1} W_i g(\xi_i,\eta)d\eta \\ &= \sum_{j=1}^{M_2}\bar{W}_j \sum_{i=1}^{M_1} W_i g(\xi_i,\eta_j) \\ &= \sum_{i=1}^{M_1}\sum_{j=1}^{M_2} W_i\bar{W}_j g(\xi_i,\eta_j) \end{aligned} \tag{6.4.17}$$

in which M_1 and M_2 are the number of integration points in the ξ and η axes, respectively. In addition, (ξ_i, η_j) are the integration points and W_i and $\bar{W}_j$ are weighting coefficients. Table 6.4.1 can be used for these values. Similarly, numerical integration in three-dimension becomes

$$\int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} g(\xi,\eta,\zeta)d\xi d\eta d\zeta = \sum_{i=1}^{M_1}\sum_{j=1}^{M_2}\sum_{k=1}^{M_3} W_i\bar{W}_j\tilde{W}_k g(\xi_i,\eta_j,\zeta_k) \tag{6.4.18}$$

♣ **Example 6.4.5** Integrate the following expression:

$$\int_{-1}^{1}\int_{-1}^{1} 9\xi^2\eta^2 d\xi d\eta \tag{6.4.19}$$

The integrand is the second order in terms of ξ and η, respectively. That is, $2M_1 - 1 = 2M_2 - 1 = 2$ for both axes. Therefore, we use two point quadrature in both ξ and η directions. The integration points are $\xi_1 = \eta_1 = -\frac{1}{\sqrt{3}}$ and $\xi_2 = \eta_2 = \frac{1}{\sqrt{3}}$. The weighting coefficients are $W_1 = \bar{W}_1 = 1$ and $W_2 = \bar{W}_2 = 1$. Applying these values to Eq. (6.4.17) results in 4. ‡

♣ **Example 6.4.6** We want to integrate

$$\int_{-1}^{1}\int_{-1}^{1} 15\xi^2\eta^4 d\xi d\eta \tag{6.4.20}$$

The integrand is the second order in terms of ξ and fourth order in terms of η. Therefore $M_1 = 2$ and $M_2 = 3$. Using the two point quadrature in the ξ direction and three point quadrature in the η direction from Table 6.4.1, we obtain the solution of 4. ‡

6.5 MATLAB Application to Gauss Quadrature

This section shows MATLAB examples for numerical integration of one-, two- or three-dimensional functions using Gauss-Legendre quadrature. The domain of integration is normalized between -1 and 1 for every axis.

♣ **Example 6.5.1** We want to integrate

$$f(x) = 1 + x^2 - 3x^3 + 4x^5 \tag{6.5.1}$$

over the domain $-1 < x < 1$ using Gauss-Legèndre quadrature. Because the highest order of polynomial is 5, we need the 3-point quadrature rule for exact integration from $2n - 1 = 5$. The numerical result is $\frac{8}{3}$. The MATLAB program is shown below.

```
%-------------------------------------------------------------
% Example 6.5.1
% Gauss-Legendre quadrature of a function in 1-dimension
%
% Problem dcscription
% Integrate f(x)=1+x^2-3x^3+4x^5 between x=-1 and x=1
%
% Variable descriptions
% point1 = integration (or sampling) points
% weight1 = weighting coefficients
% ngl = number of integration points
%-------------------------------------------------------------
%
ngl=3;                                          % (2*ngl-1)=5
%
[point1,weight1]=feglqd1(ngl);     % extract sampling points and weights
%
%-----------------------------
% summation for numerical integration
%-----------------------------
%
value=0.0;
%
for int=1:ngl
x=point1(int);
wt=weight1(int);
func=1+x^2-3*x^3+4*x^5;          % evaluate function at sampling point
value=value+func*wt;
end
%
value                                           % print the solution
%
%-------------------------------------------------------------
```



```
function [point1,weight1]=feglqd1(ngl)
%-------------------------------------------------------------
% Purpose:
% determine the integration points and weighting coefficients
% of Gauss-Legendre quadrature for one-dimensional integration
%
% Synopsis:
% [point1,weight1]=feglqd1(ngl)
%
% Variable Description:
% ngl - number of integration points
% point1 - vector containing integration points
% weight1 - vector containing weighting coefficients
%-------------------------------------------------------------
%
% initialization
%
point1=zeros(ngl,1);
weight1=zeros(ngl,1);
%
% find corresponding integration points and weights
%
if ngl==1                                       % 1-point quadrature rule
```

```
point1(1)=0.0;
weight1(1)=2.0;
%
elseif ngl==2                          % 2-point quadrature rule
point1(1)=-0.577350269189626;
point1(2)=-point1(1);
weight1(1)=1.0;
weight1(2)=weight1(1);
%
elseif ngl==3                          % 3-point quadrature rule
point1(1)=-0.774596669241483;
point1(2)=0.0;
point1(3)=-point1(1);
weight1(1)=0.555555555555556;
weight1(2)=0.888888888888889;
weight1(3)=weight1(1);
%
elseif ngl==4                          % 4-point quadrature rule
point1(1)=-0.861136311594053;
point1(2)=-0.339981043584856;
point1(3)=-point1(2);
point1(4)=-point1(1);
weight1(1)=0.347854845137454;
weight1(2)=0.652145154862546;
weight1(3)=weight1(2);
weight1(4)=weight1(1);
%
else                                   % 5-point quadrature rule
point1(1)=-0.906179845938664;
point1(2)=-0.538469310105683;
point1(3)=0.0;
point1(4)=-point1(2);
point1(5)=-point1(1);
weight1(1)=0.236926885056189;
weight1(2)=0.478628670499366;
weight1(3)=0.568888888888889;
weight1(4)=weight1(2);
weight1(5)=weight1(1);
%
end
%
%-------------------------------------------------------------
```

‡

♣ **Example 6.5.2** Use Gauss-Legendre quadrature for integration of

$$f(x,y) = 1 + 4xy - 3x^2y^2 + x^4y^6 \tag{6.5.2}$$

over the domain $-1 < x < 1$ and $-1 < y < 1$. We use 3-point quadrature rule along the x-axis and 4-point quadrature rule along the y-axis. The result is 2.7810.

```
%-------------------------------------------------------------
% Example 6.5.2
% Gauss-Legendre quadrature of a function in 2-dimension
%
% Problem description
% Integrate f(x,y)=1+4xy-3x^2y^2+x^4y^6 over -1<x<1 and -1<y<1
%
% Variable descriptions
% point2 = integration (or sampling) points
% weight2 = weighting coefficients
% nglx = number of integration points along x-axis
% ngly = number of integration points along y-axis
%-------------------------------------------------------------
%
nglx=3;                                        % (2*nglx-1)=4
ngly=4;                                        % (2*ngly-1)=6
%
[point2,weight2]=feglqd2(nglx,ngly);     % sampling points and weights
%
%-------------------------
% summation for numerical integration
%-------------------------
%
value=0.0;
%
for intx=1:nglx
x=point2(intx,1);                        % sampling point in x-axis
wtx=weight2(intx,1);                     % weight in x-axis
for inty=1:ngly
y=point2(inty,2);                        % sampling point in y-axis
wty=weight2(inty,2) ;                    % weight in y-axis
func=1+4*x*y-3*x^2*y^2+x^4*y^6;          % evaluate function
value=value+func*wtx*wty;
end
end
%
value %
%-------------------------------------------------------------
```

```
function [point2,weight2]=feglqd2(nglx,ngly)
%-------------------------------------------------------------
% Purpose:
% determine the integration points and weighting coefficients
% of Gauss-Legendre quadrature for two-dimensional integration
%
% Synopsis:
% [point2,weight2]=feglqd2(nglx,ngly)
%
% Variable Description:
% nglx - number of integration points in the x-axis
% ngly - number of integration points in the y-axis
% point2 - vector containing integration points
% weight2 - vector containing weighting coefficients
%-------------------------------------------------------------
%
% determine the largest one between nglx and ngly
%
if nglx > ngly
ngl=nglx;
else
ngl=ngly;
end
%
% initialization
%
point2=zeros(ngl,2);
weight2=zeros(ngl,2);
%
% find corresponding integration points and weights
%
[pointx,weightx]=feglqd1(nglx); [pointy,weighty]=feglqd1(ngly); %
% quadrature for two-dimension
%
for intx=1:nglx point2(intx,1)=pointx(intx);
weight2(intx,1)=weightx(intx);
end
%
for inty=1:ngly point2(inty,2)=pointy(inty);
weight2(inty,2)=weighty(inty);
end
%
%-------------------------------------------------------------
```

‡

♣ **Example 6.5.3** The following three-dimensional function is integrated using Gauss-Legendre quadrature.

$$f(x,y,z) = 1 + 4x^2y^2 - 3x^2z^4 + y^4z^6 \tag{6.6.3}$$

over the normalized domain $-1 < x < 1$, $-1 < y < 1$ and $-1 < z < 1$. The integrated value is 10.1841.

```
%-------------------------------------------------------------
% Example 6.5.3
% Gauss-Legendre quadrature of a function in 3-dimension
%
% Problem description
% Integrate f(x,y,z)=1+4x^2y^2-3x^2z^4+y^4z^6 over -1<(x,y,z)<1
%
% Variable descriptions
% point3 = integration (or sampling) points
% weight3 = weighting coefficients
% nglx = number of integration points along x-axis
% ngly = number of integration points along y-axis
% nglz = number of integration points along z-axis
%-------------------------------------------------------------
%
nglx=2;                                          % (2*nglx-1)=2
ngly=3;                                          % (2*ngly-1)=4
nglz=4;                                          % (2*nglz-1)=6
%
[point3,weight3]=feglqd3(nglx,ngly,nglz);    % sampling point & weight
%
%-------------------------
% summation for numerical integration
%-------------------------
%
value=0.0;
%
for intx=1:nglx
x=point3(intx,1);                          % sampling point in x-axis
wtx=weight3(intx,1);                           % weight in x-axis
for inty=1:ngly
y=point3(inty,2);                          % sampling point in y-axis
wty=weight3(inty,2) ;                          % weight in y-axis
for intz=1:nglz
z=point3(intz,3);                          % sampling point in z-axis
wtz=weight3(intz,3) ;                          % weight in z-axis
func=1+4*x^2*y^2-3*x^2*z^4+y^4*z^6;            % evaluate function
value=value+func*wtx*wty*wtz;
end
end
```

```
end
%
value                                          % print the solution
%
%-------------------------------------------------------------------


function [point3,weight3]=feglqd3(nglx,ngly,nglz)
%-------------------------------------------------------------------
% Purpose:
% determine the integration points and weighting coefficicnts
% of Gauss-Legendre quadrature for three-dimensional integration
%
% Synopsis:
% [point3,weight3]=feglqd3(nglx,ngly,nglz)
%
% Variable Description:
% nglx - number of integration points in the x-axis
% ngly - number of integration points in the y-axis
% nglz - number of integration points in the z-axis
% point3 - vector containing integration points
% weight3 - vector containing weighting coefficients
%-------------------------------------------------------------------
%
% determine the largest one between nglx and ngly
%
if nglx > ngly
if nglx > nglz
ngl=nglx;
else
ngl=nglz;
end
else
if ngly > nglz
ngl=ngly;
else
ngl=nglz;
end
end
%
% initialization
%
point3=zeros(ngl,3);
weight3=zeros(ngl,3);
%
% find corresponding integration points and weights
%
```



```
[pointx,weightx]=feglqd1(nglx);          % quadrature rule for x-axis
[pointy,weighty]=feglqd1(ngly);          % quadrature rule for y-axis
[pointz,weightz]=feglqd1(nglz);          % quadrature rule for z-axis
%
% quadrature for two-dimension
%
for intx=1:nglx                          % quadrature in x-axis
point3(intx,1)=pointx(intx);
weight3(intx,1)=weightx(intx);
end
%
for inty=1:ngly                          % quadrature in y-axis
point3(inty,2)=pointy(inty);
weight3(inty,2)=weighty(inty);
end
%
for intz=1:nglz                          % quadrature in z-axis
point3(intz,3)=pointz(intz);
weight3(intz,3)=weightz(intz);
end
%
%-------------------------------------------------------------------
```

‡

6.6 MATLAB Application to Laplace Equation

Isoparametric elements are used to solve the *Laplace* equation which was discussed in Chapter 5.

♣ **Example 6.6.1** This example shows how to compute the element matrix for the Laplace equation. The element matrix is expressed as

$$K_{ij}^{e} = \int_{\Omega^e} \left\{ \frac{\partial H_i}{\partial x}\frac{\partial H_j}{\partial x} + \frac{\partial H_i}{\partial y}\frac{\partial H_j}{\partial y} \right\} d\Omega \tag{6.6.1}$$

The element domain is shown in Fig. 6.2.4. The MATLAB program is listed below to evaluate the element matrix.

```
%-------------------------------------------------------------------
% Example 6.6.1
% Compute element matrix for two-dimensional Laplace equation
%
% Problem description
% Determine the element matrix for Laplace equation using
% isoparametric four-node quadrilateral element and Gauss-Legendre
```

```
% quadrature for a single element shown in Fig. 6.2.4.
%
% Variable descriptions
% k - element matrix
% point2 - integration (or sampling) points
% weight2 - weighting coefficients
% nglx - number of integration points along x-axis
% ngly - number of integration points along y-axis
% xcoord - x coordinate values of nodes
% ycoord - y coordinate values of nodes
% jacob2 - Jacobian matrix
% shape - four-node quadrilateral shape functions
% dhdr - derivatives of shape functions w.r.t. natural coord. r
% dhds - derivatives of shape functions w.r.t. natural coord. s
% dhdx - derivatives of shape functions w.r.t. physical coord. x
% dhdy - derivatives of shape functions w.r.t. physical coord. y
%-------------------------------------------------------------
%
nnel=4;                                % number of nodes per element
ndof=1;                                % degrees of freedom per node
edof=nnel*ndof;                        % degrees of freedom per element
%
nglx=2; ngly=2;                        % use 2x2 integration rule
%
xcoord=[-1 1 1 -1];                    % x coordinate values
ycoord=[-0.75 -0.75 1.25 0.25];        % y coordinate values
%
[point2,weight2]=feglqd2(nglx,ngly);   % sampling points & weights
%
%-----------------------
% numerical integration
%-----------------------
k=zeros(edof,edof);                    % initialization to zero
%
for intx=1:nglx
x=point2(intx,1);                      % sampling point in x-axis
wtx=weight2(intx,1);                   % weight in x-axis
for inty=1:ngly
y=point2(inty,2);                      % sampling point in y-axis
wty=weight2(inty,2) ;                  % weight in y-axis
%
[shape,dhdr,dhds]=feisoq4(x,y);        % compute shape functions and
                                       % derivatives at sampling point
%
jacob2=fejacob2(nnel,dhdr,dhds,xcoord,ycoord);   % compute Jacobian
detjacob=det(jacob2);                  % determinant of Jacobian
invjacob=inv(jacob2);                  % inverse of Jacobian matrix
%
```



```
[dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob);   % derivatives w.r.t.
                                                 % physical coordinate
%
%-----------------
% element matrix loop
%-----------------
for i=1:edof
for j=1:edof
k(i,j)=k(i,j)+(dhdx(i)*dhdx(j)+dhdy(i)*dhdy(j))*wtx*wty*detjacob;
end
end
%
end
end                                    % end of numerical integration loop
%
k                                      % print the element matrix
%
%-------------------------------------------------------------
```

```
function [dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob)
%-------------------------------------------------------------
% Purpose:
% determine derivatives of 2-D isoparametric shape functions with
% respect to physical coordinate system
%
% Synopsis:
% [dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob)
%
% Variable Description:
% dhdx - derivative of shape function w.r.t. physical coordinate x
% dhdy - derivative of shape function w.r.t. physical coordinate y
% nnel - number of nodes per element
% dhdr - derivative of shape functions w.r.t. natural coordinate r
% dhds - derivative of shape functions w.r.t. natural coordinate s
% invjacob - inverse of 2-D Jacobian matrix
%-------------------------------------------------------------
%
for i=1:nnel
dhdx(i)=invjacob(1,1)*dhdr(i)+invjacob(1,2)*dhds(i);
dhdy(i)=invjacob(2,1)*dhdr(i)+invjacob(2,2)*dhds(i);
end
%
%-------------------------------------------------------------
```

```
function [shapeq4,dhdrq4,dhdsq4]=feisoq4(rvalue,svalue)
%-----------------------------------------------------------------------
% Purpose:
% compute isoparametric four-node quadrilateral shape functions
% and their derivatves at the selected (integration) point
% in terms of the natural coordinate
%
% Synopsis:
% [shapeq4,dhdrq4,dhdsq4]=feisoq4(rvalue,svalue)
%
% Variable Description:
% shapeq4 - shape functions for four-node element
% dhdrq4 - derivatives of the shape functions w.r.t. r
% dhdsq4 - derivatives of the shape functions w.r.t. s
% rvalue - r coordinate value of the selected point
% svalue - s coordinate value of the selected point
%
% Notes:
% 1st node at (-1,-1), 2nd node at (1,-1)
% 3rd node at (1,1), 4th node at (-1,1)
%-----------------------------------------------------------------------
%
% shape functions
%
shapeq4(1)=0.25*(1-rvalue)*(1-svalue);
shapeq4(2)=0.25*(1+rvalue)*(1-svalue);
shapeq4(3)=0.25*(1+rvalue)*(1+svalue);
shapeq4(4)=0.25*(1-rvalue)*(1+svalue);
%
% derivatives
%
dhdrq4(1)=-0.25*(1-svalue);
dhdrq4(2)=0.25*(1-svalue);
dhdrq4(3)=0.25*(1+svalue);
dhdrq4(4)=-0.25*(1+svalue);
%
dhdsq4(1)=-0.25*(1-rvalue);
dhdsq4(2)=-0.25*(1+rvalue);
dhdsq4(3)=0.25*(1+rvalue);
dhdsq4(4)=0.25*(1-rvalue);
%
%-----------------------------------------------------------------------



function [jacob2]=fejacob2(nnel,dhdr,dhds,xcoord,ycoord)
%-----------------------------------------------------------------------
% Purpose:
```



```
% determine the Jacobian for two-dimensional mapping
%
% Synopsis:
% [jacob2]=fejacob2(nnel,dhdr,dhds,xcoord,ycoord)
%
% Variable Description:
% jacob2 - Jacobian for one-dimension
% nnel - number of nodes per element
% dhdr - derivative of shape functions w.r.t. natural coordinate r
% dhds - derivative of shape functions w.r.t. natural coordinate s
% xcoord - x axis coordinate values of nodes
% ycoord - y axis coordinate values of nodes
%-----------------------------------------------------------------------
%
jacob2=zeros(2,2);
%
for i=1:nnel
jacob2(1,1)=jacob2(1,1)+dhdr(i)*xcoord(i);
jacob2(1,2)=jacob2(1,2)+dhdr(i)*ycoord(i);
jacob2(2,1)=jacob2(2,1)+dhds(i)*xcoord(i);
jacob2(2,2)=jacob2(2,2)+dhds(i)*ycoord(i);
end
%
%-----------------------------------------------------------------------
```

The computed element matrix is

$$[K] = \begin{bmatrix} 0.7500 & 0.0000 & -0.2500 & -0.5000 \\ 0.0000 & 0.7500 & -0.2500 & -0.5000 \\ -0.2500 & -0.2500 & 0.5000 & 0.0000 \\ -0.5000 & -0.5000 & 0.0000 & 1.0000 \end{bmatrix} \tag{6.6.2}$$

‡

♣ **Example 6.6.2** Repeat Example 5.9.2 using isoparametric elements. Four-node quadrilateral elements are used. The finite element solution is the same as that obtained in Example 5.9.2. As a result, the solution is not repeated here. The MATLAB program is shown below.

```
%-----------------------------------------------------------------------
% Example 6.6.2
% to solve the two-dimensional Laplace's equation given as
% u,xx + u,yy =0, 0 < x < 5, 0 < y < 10
% u(x,0) = 0, u(x,10) = 100sin(pi*x/10),
% u(0,y) = 0, u,x(5,y) = 0
% using isoparametric four-node quadrilateral elements
```

```
% (see Fig. 5.9.2 for the finite element mesh)
%
% Variable descriptions
% k = element matrix
% f = element vector
% kk = system matrix
% ff = system vector
% gcoord = coordinate values of each node
% nodes = nodal connectivity of each element
% index = a vector containing system dofs associated with each element
% bcdof = a vector containing dofs associated with boundary conditions
% bcval = a vector containing boundary condition values associated with
%                 the dofs in bcdof
% point2 - integration (or sampling) points
% weight2 - weighting coefficients
% nglx - number of integration points along x-axis
% ngly - number of integration points along y-axis
% xcoord - x coordinate values of nodes
% ycoord - y coordinate values of nodes
% jacob2 - Jacobian matrix
% shape - four-node quadrilateral shape functions
% dhdr - derivatives of shape functions w.r.t. natural coord. r
% dhds - derivatives of shape functions w.r.t. natural coord. s
% dhdx - derivatives of shape functions w.r.t. physical coord. x
% dhdy - derivatives of shape functions w.r.t. physical coord. y
%-------------------------------------------------------------------
clear
%-------------------------------
% input data for control parameters
%-------------------------------
nel=16;                                  % number of elements
nnel=4;                                  % number of nodes per element
ndof=1;                                  % number of dofs per node
nnode=25;                                % total number of nodes in system
nglx=2; ngly=2;                          % use 2x2 integration rule
sdof=nnode*ndof;                         % total system dofs
edof=nnel*ndof;                          % dofs per element
%
%-------------------------------------------
% input data for nodal coordinate values
% gcoord(i,j) where i-> node no. and j-> x or y
%-------------------------------------------
gcoord(1,1)=0.0; gcoord(1,2)=0.0;
gcoord(2,1)=1.25; gcoord(2,2)=0.0;
gcoord(3,1)=2.5; gcoord(3,2)=0.0;
gcoord(4,1)=3.75; gcoord(4,2)=0.0;
gcoord(5,1)=5.0; gcoord(5,2)=0.0;
gcoord(6,1)=0.0; gcoord(6,2)=2.5;
```



```
gcoord(7,1)=1.25; gcoord(7,2)=2.5;
gcoord(8,1)=2.5; gcoord(8,2)=2.5;
gcoord(9,1)=3.75; gcoord(9,2)=2.5;
gcoord(10,1)=5.0; gcoord(10,2)=2.5;
gcoord(11,1)=0.0; gcoord(11,2)=5.0;
gcoord(12,1)=1.25; gcoord(12,2)=5.0;
gcoord(13,1)=2.5; gcoord(13,2)=5.0;
gcoord(14,1)=3.75; gcoord(14,2)=5.0;
gcoord(15,1)=5.0; gcoord(15,2)=5.0;
gcoord(16,1)=0.0; gcoord(16,2)=7.5;
gcoord(17,1)=1.25; gcoord(17,2)=7.5;
gcoord(18,1)=2.5; gcoord(18,2)=7.5;
gcoord(19,1)=3.75; gcoord(19,2)=7.5;
gcoord(20,1)=5.0; gcoord(20,2)=7.5;
gcoord(21,1)=0.0; gcoord(21,2)=10.;
gcoord(22,1)=1.25; gcoord(22,2)=10.;
gcoord(23,1)=2.5; gcoord(23,2)=10.;
gcoord(24,1)=3.75; gcoord(24,2)=10.;
gcoord(25,1)=5.0; gcoord(25,2)=10.;
%
%-------------------------------------------------------------------
% input data for nodal connectivity for each element
% nodes(i,j) where i-> element no. and j-> connected nodes
%-------------------------------------------------------------------
nodes(1,1)=1; nodes(1,2)=2; nodes(1,3)=7; nodes(1,4)=6;
nodes(2,1)=2; nodes(2,2)=3; nodes(2,3)=8; nodes(2,4)=7;
nodes(3,1)=3; nodes(3,2)=4; nodes(3,3)=9; nodes(3,4)=8;
nodes(4,1)=4; nodes(4,2)=5; nodes(4,3)=10; nodes(4,4)=9;
nodes(5,1)=6; nodes(5,2)=7; nodes(5,3)=12; nodes(5,4)=11;
nodes(6,1)=7; nodes(6,2)=8; nodes(6,3)=13; nodes(6,4)=12;
nodes(7,1)=8; nodes(7,2)=9; nodes(7,3)=14; nodes(7,4)=13;
nodes(8,1)=9; nodes(8,2)=10; nodes(8,3)=15; nodes(8,4)=14;
nodes(9,1)=11; nodes(9,2)=12; nodes(9,3)=17; nodes(9,4)=16;
nodes(10,1)=12; nodes(10,2)=13; nodes(10,3)=18; nodes(10,4)=17;
nodes(11,1)=13; nodes(11,2)=14; nodes(11,3)=19; nodes(11,4)=18;
nodes(12,1)=14; nodes(12,2)=15; nodes(12,3)=20; nodes(12,4)=19;
nodes(13,1)=16; nodes(13,2)=17; nodes(13,3)=22; nodes(13,4)=21;
nodes(14,1)=17; nodes(14,2)=18; nodes(14,3)=23; nodes(14,4)=22;
nodes(15,1)=18; nodes(15,2)=19; nodes(15,3)=24; nodes(15,4)=23;
nodes(16,1)=19; nodes(16,2)=20; nodes(16,3)=25; nodes(16,4)=24;
%
%-------------------------------
% input data for boundary conditions
%-------------------------------
%
bcdof(1)=1;                              % first node is constrained
bcval(1)=0;                              % whose described value is 0
bcdof(2)=2;                              % second node is constrained
```

```
bcval(2)=0;                          % whose described value is 0
bcdof(3)=3;                          % third node is constrained
bcval(3)=0;                          % whose described value is 0
bcdof(4)=4;                          % 4th node is constrained
bcval(4)=0;                          % whose described value is 0
bcdof(5)=5;                          % 5th node is constrained
bcval(5)=0;                          % whose described value is 0
bcdof(6)=6;                          % 6th node is constrained
bcval(6)=0;                          % whose described value is 0
bcdof(7)=11;                         % 11th node is constrained
bcval(7)=0;                          % whose described value is 0
bcdof(8)=16;                         % 16th node is constrained
bcval(8)=0;                          % whose described value is 0
bcdof(9)=21;                         % 21st node is constrained
bcval(9)=0;                          % whose described value is 0
bcdof(10)=22;                        % 22nd node is constrained
bcval(10)=38.2683;                   % whose described value is 38.2683
bcdof(11)=23;                        % 23rd node is constrained
bcval(11)=70.7107;                   % whose described value is 70.7107
bcdof(12)=24;                        % 24th node is constrained
bcval(12)=92.3880;                   % whose described value is 92.3880
bcdof(13)=25;                        % 25th node is constrained
bcval(13)=100;                       % whose described value is 100
%
%-------------------------------
% initialization of matrices and vectors
%-------------------------------
ff=zeros(sdof,1);                    % initialization of system force vector
kk=zeros(sdof,sdof);                 % initialization of system matrix
index=zeros(nnel*ndof,1);            % initialization of index vector
%
%-----------------------------------------
% loop for computation and assembly of element matrices
%-----------------------------------------
[point2,weight2]=feglqd2(nglx,ngly);  % sampling points & weights
%
for iel=1:nel                        % loop for the total number of elements
%
for i=1:nnel
nd(i)=nodes(iel,i);                  % extract connected node for (iel)-th element
xcoord(i)=gcoord(nd(i),1);           % extract x value of the node
ycoord(i)=gcoord(nd(i),2);           % extract y value of the node
end
%
k=zeros(edof,edof);                  % initialization of element matrix to zero
%
%-------------------
% numerical integration
```



```
%-------------------
for intx=1:nglx
x=point2(intx,1);                    % sampling point in x-axis
wtx=weight2(intx,1);                 % weight in x-axis
for inty=1:ngly
y=point2(inty,2);                    % sampling point in y-axis
wty=weight2(inty,2) ;                % weight in y-axis
%
[shape,dhdr,dhds]=feisoq4(x,y);      % compute shape functions and
                                     % derivatives at sampling point
%
jacob2=fejacob2(nnel,dhdr,dhds,xcoord,ycoord);  % compute Jacobian
%
detjacob=det(jacob2);                % determinant of Jacobian
invjacob=inv(jacob2);                % inverse of Jacobian matrix
%
[dhdx,dhdy]=federiv2(nnel,dhdr,dhds,invjacob);  % derivatives w.r.t.
                                                % physical coordinate
%
%-------------------
% compute element matrix
%-------------------
for i=1:edof
for j=1:edof
k(i,j)=k(i,j)+(dhdx(i)*dhdx(j)+dhdy(i)*dhdy(j))*wtx*wty*detjacob;
end
end
%
end
end                                  % end of numerical integration loop
%
index=feeldof(nd,nnel,ndof);         % extract system dofs for the element
%
%-------------------
% assemble element matrices
%-------------------
%
kk=feasmbl1(kk,k,index);
%
end %
%-------------------
% apply boundary conditions
%-------------------
[kk,ff]=feaplyc2(kk,ff,bcdof,bcval);
%
%-------------------
% solve the matrix equation
%-------------------
```

```
fsol=kk\ff;
%
%-----------------
% analytical solution
%-----------------
for i=1:nnode
x=gcoord(i,1); y=gcoord(i,2);
esol(i)=100*sinh(0.31415927*y)*sin(0.31415927*x)/sinh(3.1415927);
end
%
%-----------------------------
% print both exact and fem solutions
%-----------------------------
num=1:1:sdof;
store=[num' fsol esol']
%
%-------------------------------------------------------------
```

‡

Problems

6.1 Compute the following integral using the quadratic isoparametric element:

$$K_{11} = \int_2^6 \left[\left(\frac{dH_1}{dx} \right)^2 + (H_1)^2 \right] dx$$

The shape functions are given in Eqs (6.1.13) through (6.1.15) and the element has nodes x_1=2, x_2=4, and x_3=6 in the physical coordinate system.

6.2 Consider one-dimensional isoparametric shape functions as given in Eqs (6.1.13) through (6.1.15). The isoparametric element is mapped into a physical domain with nodal points located at $x_1 = 0$, $x_2 = a$, and $x_3 = 4$, where $a = 1.5$, $a = 1$, or $a = 0.5$. Compute the Jacobian J and its inverse for these cases.

6.3 Compute the Jacobian matrix for the following bilinear element shown in Fig. P6.3 and evaluate

$$K_{12} = \int_\Omega \left(\frac{\partial H_1}{\partial x} \frac{\partial H_2}{\partial x} + \frac{\partial H_1}{\partial y} \frac{\partial H_2}{\partial y} \right) d\Omega$$

Using the isoparametric element and 3 by 3 Gauss-Legendre quadrature. The shape functions are provided in Eqs (6.2.1) through (6.2.4).

6.4 For the linear triangular isoparametric element shown in Fig. P6.4, (a) compute the Jacobian matrix and (b) find $\frac{\partial H_1}{\partial x}$ in which H_1 is given in Eq. (6.3.1).

6.5 Evaluate the Jacobian matrix for the four-node element shown in Fig. P6.5 using the bilinear isoparametric element.

6.6 Gauss-Legendre quadrature rule is used to evaluate the integral

$$\int_{-1}^{1} \int_{-1}^{1} H_1(\xi, \eta) H_3(\xi, \eta) |J| d\xi d\eta$$

in which H_1 and H_2 are quadratic shape functions of ξ and η, respectively. What order of integration is necessary for exact integration of the integral if the element has no distortion (i.e. a rectangular shape of element in the physical domain)?

6.7 Two different isoparametric elements are used together as shown in Fig. P6.7. There is an interelement boundary between (x, y)=(1,1) and (x, y)=(2,2). Show that variable is continuous across the interface boundary. In other words, variable interpolation from the quadrilateral element is the same as that from the triangular element at the interface.

6.8 Consider two elements shown in Fig. P6.7 again. For the elements, we use the following interpolation for each element. For the triangular element, we use

$$u = a_0 + a_1 x + a_2 y$$

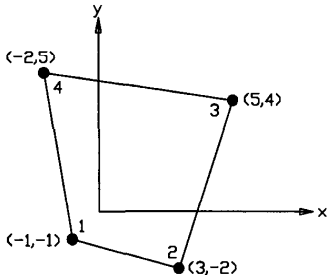

Figure P6.3 Problem 6.3

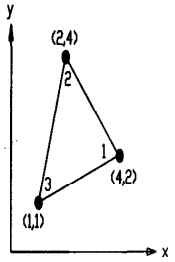

Figure P6.4 Problem 6.4

and for the quadrilateral element we use

$$u = b_0 + b_1 x + b_2 y + b_3 xy$$

Is u compatible at the element interface of the two elements?

6.9 Two kinds of quadrilateral isoparametric elements are utilized together to mesh a domain as seen in Fig. P6.9. One is a bilinear element and the other is a biquadratic element. Is it compatible between the element interface?

6.10 Solve Prob. 5.11 using isoparametric elements and computer programs provided in this chapter.

6.11 Solve Prob. 5.12 using isoparametric elements and MATLAB programs.

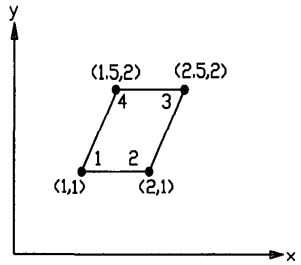

Figure P6.5 Problem 6.5

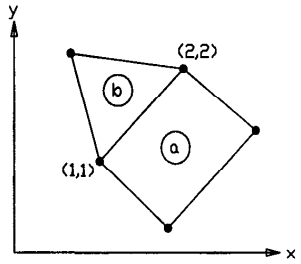

Figure P6.7 Problem 6.7

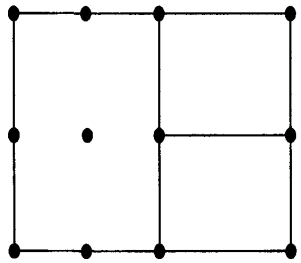

Figure P6.9 Problem 6.9