# 16 The Linear Brick (Solid) Element

## 16.1 Basic Equations

The linear brick (solid) element is a three-dimensional finite element with both local and global coordinates. It is characterized by linear shape functions in each of the $x$, $y$, and z directions. It is also called a trilinear hexahedron. This is the third isoparametric element we deal with in this book. The linear brick element has modulus of elasticity $E$ and Poisson's ratio $\nu$. Each linear brick element has eight nodes with three degrees of freedom at each node as shown in Fig. 16.1. The global coordinates of the eight nodes are denoted by $(x_1, y_1, z_1)$, $(x_2, y_2, z_2)$, $(x_3, y_3, z_3)$, $(x_4, y_4, z_4)$, $(x_5, y_5, z_5)$, $(x_6, y_6, z_6)$, $(x_7, y_7, z_7)$, and $(x_8, y_8, z_8)$. The order of the nodes for each element is important – they should be numbered such that the volume of the element is positive. You can actually check this by using the MATLAB function *LinearBrickElementVolume* which is written specifically for this purpose. The element is mapped to a hexahedron through the use of the natural coordinates $\xi$, $\eta$, and $\mu$ as shown in Fig. 16.2. In this case the element stiffness matrix is not written explicitly but calculated through symbolic integration with the aid of the MATLAB Symbolic Math Toolbox. The eight shape functions for this element are listed explicitly as follows in terms of the natural coordinates $\xi$, $\eta$, and $\mu$ (see [1]).

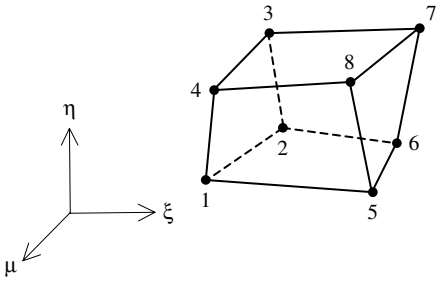


**Fig. 16.1.** The Linear Brick (Solid) Element

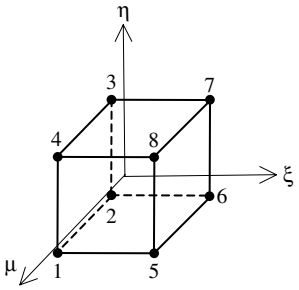


**Fig. 16.2.** The Linear Brick Element with Natural Coordinates

$$
\begin{aligned}
N_1 &= \frac{1}{8}(1-\xi)(1-\eta)(1+\mu) \\
N_2 &= \frac{1}{8}(1-\xi)(1-\eta)(1-\mu) \\
N_3 &= \frac{1}{8}(1-\xi)(1+\eta)(1-\mu) \\
N_4 &= \frac{1}{8}(1-\xi)(1+\eta)(1+\mu) \\
N_5 &= \frac{1}{8}(1+\xi)(1-\eta)(1+\mu) \\
N_6 &= \frac{1}{8}(1+\xi)(1-\eta)(1-\mu) \\
N_7 &= \frac{1}{8}(1+\xi)(1+\eta)(1-\mu) \\
N_8 &= \frac{1}{8}(1+\xi)(1+\eta)(1+\mu)
\end{aligned} \tag{16.1}
$$

The Jacobian matrix for this element is given by

$$
[J] = \begin{bmatrix}
\dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\
\dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\
\dfrac{\partial x}{\partial \mu} & \dfrac{\partial y}{\partial \mu} & \dfrac{\partial z}{\partial \mu}
\end{bmatrix} \tag{16.2}
$$

where $x$, $y$, and $z$ are given by

$$
\begin{aligned}
x &= N_1x_1 + N_2x_2 + N_3x_3 + N_4x_4 + N_5x_5 + N_6x_6 + N_7x_7 + N_8x_8 \\
y &= N_1y_1 + N_2y_2 + N_3y_3 + N_4y_4 + N_5y_5 + N_6y_6 + N_7y_7 + N_8y_8 \\
z &= N_1z_1 + N_2z_2 + N_3z_3 + N_4z_4 + N_5z_5 + N_6z_6 + N_7z_7 + N_8z_8
\end{aligned} \tag{16.3}
$$

The $[B]$ matrix is given as follows for this element:

$$[B] = [D^{/}][N] \tag{16.4}$$

where $[D^{/}]$ and $[N]$ are given by:

$$[D^{/}] = \begin{bmatrix} \dfrac{\partial(\,)}{\partial x} & 0 & 0 \\ 0 & \dfrac{\partial(\,)}{\partial y} & 0 \\ 0 & 0 & \dfrac{\partial(\,)}{\partial z} \\ \dfrac{\partial(\,)}{\partial y} & \dfrac{\partial(\,)}{\partial x} & 0 \\ 0 & \dfrac{\partial(\,)}{\partial z} & \dfrac{\partial(\,)}{\partial y} \\ \dfrac{\partial(\,)}{\partial z} & 0 & \dfrac{\partial(\,)}{\partial x} \end{bmatrix} \tag{16.5}$$

$$[N] = \begin{bmatrix} [N_0] & 0 & 0 \\ 0 & [N_0] & 0 \\ 0 & 0 & [N_0] \end{bmatrix} \tag{16.6}$$

and the submatrix $[N_0]$ is given by

$$[N_0] = \begin{bmatrix} N_1 & N_2 & N_3 & N_4 & N_5 & N_6 & N_7 & N_8 \end{bmatrix} \tag{16.7}$$

The partial derivatives in (16.5) are evaluated as follows

$$\frac{\partial f}{\partial x} = \frac{1}{|J|} \begin{vmatrix} \dfrac{\partial f}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\ \dfrac{\partial f}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial f}{\partial \mu} & \dfrac{\partial y}{\partial \mu} & \dfrac{\partial z}{\partial \mu} \end{vmatrix} \tag{16.8a}$$

$$\frac{\partial f}{\partial y} = \frac{1}{|J|} \begin{vmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial f}{\partial \xi} & \dfrac{\partial z}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial f}{\partial \eta} & \dfrac{\partial z}{\partial \eta} \\ \dfrac{\partial x}{\partial \mu} & \dfrac{\partial f}{\partial \mu} & \dfrac{\partial z}{\partial \mu} \end{vmatrix} \tag{16.8b}$$

$$\frac{\partial f}{\partial z} = \frac{1}{|J|}\begin{vmatrix} \dfrac{\partial x}{\partial \xi} & \dfrac{\partial y}{\partial \xi} & \dfrac{\partial f}{\partial \xi} \\ \dfrac{\partial x}{\partial \eta} & \dfrac{\partial y}{\partial \eta} & \dfrac{\partial f}{\partial \eta} \\ \dfrac{\partial x}{\partial \mu} & \dfrac{\partial y}{\partial \mu} & \dfrac{\partial f}{\partial \mu} \end{vmatrix} \tag{16.8c}$$

where $f$ is a dummy variable to stand for the empty parentheses () in (16.5), representing either $N_1$, $N_2$, or $N_3$.

For three-dimensional analysis, the matrix $[D]$ is given by

$$[D] = \frac{E}{(1+\nu)(1-2\nu)}\begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{1-2\nu}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \dfrac{1-2\nu}{2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \dfrac{1-2\nu}{2} \end{bmatrix} \tag{16.9}$$

The element stiffness matrix for the linear brick element is written in terms of a triple integral as follows:

$$[k] = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} [B]^T[D][B]\,|J|\; d\xi\, d\eta d\mu \tag{16.10}$$

The partial differentiation of (16.5) and (16.8), and the double integration of (16.10) are carried out symbolically with the aid of the MATLAB Symbolic Math Toolbox. See the details of the MATLAB code for the function *LinearBrickElementStiffness* which calculates the element stiffness matrix for this element. The reader should note the calculation of this matrix will be somewhat slow due to the symbolic computations involved.

It is clear that the linear brick element has twenty-four degrees of freedom – three at each node. Consequently for a structure with $n$ nodes, the global stiffness matrix $K$ will be of size $3n \times 3n$ (since we have three degrees of freedom at each node). The global stiffness matrix $K$ is assembled by making calls to the MATLAB function *LinearBrickAssemble* which is written specifically for this purpose. This process will be illustrated in detail in the example.

Once the global stiffness matrix $K$ is obtained we have the following structure equation:

$$[K]\{U\} = \{F\} \tag{16.11}$$

where $U$ is the global nodal displacement vector and $F$ is the global nodal force vector. At this step the boundary conditions are applied manually to the vectors $U$ and $F$. Then the matrix (16.11) is solved by partitioning and Gaussian elimination. Finally once the unknown displacements and reactions are found, the stress vector is obtained for each element as follows:

$$\{\sigma\} = [D][B]\{u\} \tag{16.12}$$

where $\sigma$ is the stress vector in the element (of size $6 \times 1$) and $u$ is the $24 \times 1$ element displacement vector. The vector $\sigma$ is written for each element as $\{\sigma\} = [\sigma_x \sigma_y \sigma_z \tau_{xy} \tau_{yz} \tau_{zx}]^{\mathrm{T}}$. Finally, the element volume is given by the following formula:

$$V = \int_{-1}^{1}\int_{-1}^{1}\int_{-1}^{1} |J| \, d\xi \, d\eta d\mu \tag{16.13}$$

## 16.2 MATLAB Functions Used

The five MATLAB functions used for the linear brick element are:

*LinearBrickElementVolume*($x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$, $x_3$, $y_3$, $z_3$, $x_4$, $y_4$, $z_4$, $x_5$, $y_5$, $z_5$, $x_6$, $y_6$, $z_6$, $x_7$, $y_7$, $z_7$, $x_8$, $y_8$, $z_8$) – This function returns the element volume given the coordinates of the first node ($x_1$, $y_1$, $z_1$), the coordinates of the second node ($x_2$, $y_2$, $z_2$), the coordinates of the third node ($x_3$, $y_3$, $z_3$), the coordinates of the fourth node ($x_4$, $y_4$, $z_4$), the coordinates of the fifth node ($x_5$, $y_5$, $z_5$), the coordinates of the sixth node ($x_6$, $y_6$, $z_6$), the coordinates of the seventh node ($x_7$, $y_7$, $z_7$), and the coordinates of the eighth node ($x_8$, $y_8$, $z_8$).

*LinearBrickElementStiffness*($E$, *NU*, $x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$, $x_3$, $y_3$, $z_3$, $x_4$, $y_4$, $z_4$, $x_5$, $y_5$, $z_5$, $x_6$, $y_6$, $z_6$, $x_7$, $y_7$, $z_7$, $x_8$, $y_8$, $z_8$) – This function calculates the element stiffness matrix for each linear brick element with modulus of elasticity $E$, Poisson's ratio *NU*, and coordinates of the first node ($x_1$, $y_1$, $z_1$), the coordinates of the second node ($x_2$, $y_2$, $z_2$), the coordinates of the third node ($x_3$, $y_3$, $z_3$), the coordinates of the fourth node ($x_4$, $y_4$, $z_4$), the coordinates of the fifth node ($x_5$, $y_5$, $z_5$), the coordinates of the sixth node ($x_6$, $y_6$, $z_6$), the coordinates of the seventh node ($x_7$, $y_7$, $z_7$), and the coordinates of the eighth node ($x_8$, $y_8$, $z_8$). It returns the $24 \times 24$ element stiffness matrix $k$.

*LinearBrickAssemble*($K, k, i, j, m, n, p, q, r, s$) – This function assembles the element stiffness matrix $k$ of the linear brick element joining nodes $i$, $j$, $m$, $n$, $p$, $q$, $r$, and $s$ into the global stiffness matrix $K$. It returns the $3n \times 3n$ global stiffness matrix $K$ every time an element is assembled.

*LinearBrickElementStresses*($E$, *NU*, $x_1$, $y_1$, $z_1$, $x_2$, $y_2$, $z_2$, $x_3$, $y_3$, $z_3$, $x_4$, $y_4$, $z_4$, $x_5$, $y_5$, $z_5$, $x_6$, $y_6$, $z_6$, $x_7$, $y_7$, $z_7$, $x_8$, $y_8$, $z_8$, $u$) – This function calculates the element stresses using the modulus of elasticity $E$, Poisson's ratio *NU*, the coordinates of the first node ($x_1$, $y_1$, $z_1$), the coordinates of the second node ($x_2$, $y_2$, $z_2$), the coordinates of the third node ($x_3$, $y_3$, $z_3$), the coordinates of the fourth node ($x_4$, $y_4$, $z_4$), the coordinates of the fifth node ($x_5$, $y_5$, $z_5$), the coordinates of the sixth node ($x_6$, $y_6$, $z_6$), the coordinates of the seventh node ($x_7$, $y_7$, $z_7$), and the coordinates of the eighth node ($x_8$, $y_8$, $z_8$). It returns the stress vector for the element.

*LinearBrickElementPStresses*(*sigma*) – This function calculates the element principal stresses using the element stress vector *sigma*. It returns a $3 \times 1$ vector in the form $[sigma1 \;\; sigma2 \;\; theta]^{\mathrm{T}}$ where *sigma1* and *sigma2* are the principal stresses for the element and *theta* is the principal angle.

The following is a listing of the MATLAB source code for each function:

---

```
function w =
 LinearBrickElementVolume(x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,
x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8)
% LinearBrickElementVolume    This function returns the volume
%                             of the linear brick element
%                             whose first node has coordinates
%                             (x1,y1,z1), second node has
%                             coordinates (x2,y2,z2), third node
%                             has coordinates (x3,y3,z3),
%                             fourth node has coordiantes
%                             (x4,y4,z4), fifth node has coordiantes
%                             (x5,y5,z5), sixth node has coordiantes
%                             (x6,y6,z6), seventh node has coordiantes
%                             (x7,y7,z7), and eighth node has
%                             coordiantes (x8,y8,z8).
syms s t u;
N1 = (1-s)*(1-t)*(1+u)/8;
N2 = (1-s)*(1-t)*(1-u)/8;
N3 = (1-s)*(1+t)*(1-u)/8;
N4 = (1-s)*(1+t)*(1+u)/8;
N5 = (1+s)*(1-t)*(1+u)/8;
N6 = (1+s)*(1-t)*(1-u)/8;
N7 = (1+s)*(1+t)*(1-u)/8;
N8 = (1+s)*(1+t)*(1+u)/8;
 x = N1*x1 + N2*x2 + N3*x3 + N4*x4 + N5*x5 + N6*x6 + N7*x7 + N8*x8;
 y = N1*y1 + N2*y2 + N3*y3 + N4*y4 + N5*y5 + N6*y6 + N7*y7 + N8*y8;
 z = N1*z1 + N2*z2 + N3*z3 + N4*z4 + N5*z5 + N6*z6 + N7*z7 + N8*z8;
xs = diff(x,s);
xt = diff(x,t);
xu = diff(x,u);
ys = diff(y,s);
yt = diff(y,t);
yu = diff(y,u);
```

```
zs = diff(z,s);
zt = diff(z,t);
zu = diff(z,u);
J = xs*(yt*zu - zt*yu) - ys*(xt*zu - zt*xu) + zs*(xt*yu - yt*xu);
Jnew = simplify(J);
r = int(int(int(Jnew, u, -1, 1), t, -1, 1), s, -1, 1);
w = double(r);
```

---

```
function w =
LinearBrickElementStiffness(E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,
x5,y5,z5,x6,y6,z6,x7,y7,z7,x8,y8,z8)
% LinearBrickElementStiffness This function returns the element
%                             stiffness matrix for a linear brick
%                             element with modulus of elasticity
%                             E, Poisson's ratio NU, coordinates of
%                             node 1 (x1,y1,z1), coordinates
%                             of node 2 (x2,y2,z2), coordinates of
%                             node 3 (x3,y3,z3), coordinates of
%                             node 4 (x4,y4,z4), coordinates of
%                             node 5 (x5,y5,z5), coordinates of
%                             node 6 (x6,y6,z6), coordinates of
%                             node 7 (x7,y7,z7), and coordinates
%                             of node 8 (x8,y8,z8).
%                             The size of the element
%                             stiffness matrix is 24 x 24.
syms s t u;
N1 = (1-s)*(1-t)*(1+u)/8;
N2 = (1-s)*(1-t)*(1-u)/8;
N3 = (1-s)*(1+t)*(1-u)/8;
N4 = (1-s)*(1+t)*(1+u)/8;
N5 = (1+s)*(1-t)*(1+u)/8;
N6 = (1+s)*(1-t)*(1-u)/8;
N7 = (1+s)*(1+t)*(1-u)/8;
N8 = (1+s)*(1+t)*(1+u)/8;
x = N1*x1 + N2*x2 + N3*x3 + N4*x4 + N5*x5 + N6*x6 + N7*x7 + N8*x8;
y = N1*y1 + N2*y2 + N3*y3 + N4*y4 + N5*y5 + N6*y6 + N7*y7 + N8*y8;
z = N1*z1 + N2*z2 + N3*z3 + N4*z4 + N5*z5 + N6*z6 + N7*z7 + N8*z8;
xs = diff(x,s);
xt = diff(x,t);
xu = diff(x,u);
ys = diff(y,s);
yt = diff(y,t);
yu = diff(y,u);
zs = diff(z,s);
zt = diff(z,t);
zu = diff(z,u);
J = xs*(yt*zu - zt*yu) - ys*(xt*zu - zt*xu) + zs*(xt*yu - yt*xu);
N1s = diff(N1,s);
N2s = diff(N2,s);
N3s = diff(N3,s);
```

```
N4s = diff(N4,s);
N5s = diff(N5,s);
N6s = diff(N6,s);
N7s = diff(N7,s);
N8s = diff(N8,s);
N1t = diff(N1,t);
N2t = diff(N2,t);
N3t = diff(N3,t);
N4t = diff(N4,t);
N5t = diff(N5,t);
N6t = diff(N6,t);
N7t = diff(N7,t);
N8t = diff(N8,t);
N1u = diff(N1,u);
N2u = diff(N2,u);
N3u = diff(N3,u);
N4u = diff(N4,u);
N5u = diff(N5,u);
N6u = diff(N6,u);
N7u = diff(N7,u);
N8u = diff(N8,u);
% The expressions below are not divided by J - they are adjusted for
later
% in the calculation of BD matrix below.
N1x = N1s*(yt*zu - zt*yu) - ys*(N1t*zu - zt*N1u) + zs*(N1t*yu - yt*N1u);
N2x = N2s*(yt*zu - zt*yu) - ys*(N2t*zu - zt*N2u) + zs*(N2t*yu - yt*N2u);
N3x = N3s*(yt*zu - zt*yu) - ys*(N3t*zu - zt*N3u) + zs*(N3t*yu - yt*N3u);
N4x = N4s*(yt*zu - zt*yu) - ys*(N4t*zu - zt*N4u) + zs*(N4t*yu - yt*N4u);
N5x = N5s*(yt*zu - zt*yu) - ys*(N5t*zu - zt*N5u) + zs*(N5t*yu - yt*N5u);
N6x = N6s*(yt*zu - zt*yu) - ys*(N6t*zu - zt*N6u) + zs*(N6t*yu - yt*N6u);
N7x = N7s*(yt*zu - zt*yu) - ys*(N7t*zu - zt*N7u) + zs*(N7t*yu - yt*N7u);
 N8x = N8s*(yt*zu - zt*yu) - ys*(N8t*zu - zt*N8u) + zs*(N8t*yu - yt*N8u);
N1y = xs*(N1t*zu - zt*N1u) - N1s*(xt*zu - zt*xu) + zs*(xt*N1u - N1t*xu);
N2y = xs*(N2t*zu - zt*N2u) - N2s*(xt*zu - zt*xu) + zs*(xt*N2u - N2t*xu);
N3y = xs*(N3t*zu - zt*N3u) - N3s*(xt*zu - zt*xu) + zs*(xt*N3u - N3t*xu);
N4y = xs*(N4t*zu - zt*N4u) - N4s*(xt*zu - zt*xu) + zs*(xt*N4u - N4t*xu);
N5y = xs*(N5t*zu - zt*N5u) - N5s*(xt*zu - zt*xu) + zs*(xt*N5u - N5t*xu);
N6y = xs*(N6t*zu - zt*N6u) - N6s*(xt*zu - zt*xu) + zs*(xt*N6u - N6t*xu);
N7y = xs*(N7t*zu - zt*N7u) - N7s*(xt*zu - zt*xu) + zs*(xt*N7u - N7t*xu);
N8y = xs*(N8t*zu - zt*N8u) - N8s*(xt*zu - zt*xu) + zs*(xt*N8u - N8t*xu);
N1z = xs*(yt*N1u - N1t*yu) - ys*(xt*N1u - N1t*xu) + N1s*(xt*yu - yt*xu);
N2z = xs*(yt*N2u - N2t*yu) - ys*(xt*N2u - N2t*xu) + N2s*(xt*yu - yt*xu);
N3z = xs*(yt*N3u - N3t*yu) - ys*(xt*N3u - N3t*xu) + N3s*(xt*yu - yt*xu);
N4z = xs*(yt*N4u - N4t*yu) - ys*(xt*N4u - N4t*xu) + N4s*(xt*yu - yt*xu);
N5z = xs*(yt*N5u - N5t*yu) - ys*(xt*N5u - N5t*xu) + N5s*(xt*yu - yt*xu);
N6z = xs*(yt*N6u - N6t*yu) - ys*(xt*N6u - N6t*xu) + N6s*(xt*yu - yt*xu);
N7z = xs*(yt*N7u - N7t*yu) - ys*(xt*N7u - N7t*xu) + N7s*(xt*yu - yt*xu);
N8z = xs*(yt*N8u - N8t*yu) - ys*(xt*N8u - N8t*xu) + N8s*(xt*yu - yt*xu);
% Next, the B matrix is calculated explicitly as follows:
B = [N1x N2x N3x N4x N5x N6x N7x N8x 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
  0 0 0 0 0 0 0 0 N1y N2y N3y N4y N5y N6y N7y N8y 0 0 0 0 0 0 0 0 ;
  0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 N1z N2z N3z N4z N5z N6z N7z N8z ;
  N1y N2y N3y N4y N5y N6y N7y N8y N1x N2x N3x N4x N5x N6x N7x N8x 0 0 0
0 0 0 0 0 ;
```

```
  0 0 0 0 0 0 0 0 N1z N2z N3z N4z N5z N6z N7z N8z N1y N2y N3y N4y N5y
N6y N7y N8y ;
  N1z N2z N3z N4z N5z N6z N7z N8z 0 0 0 0 0 0 0 0 N1x N2x N3x N4x N5x
N6x N7x N8x];
Bnew = simplify(B);
Jnew = simplify(J);
D = (E/((1+NU)*(1-2*NU)))*[1-NU NU NU 0 0 0 ; NU 1-NU NU 0 0 0 ; NU NU
1- NU 0 0 0 ;
  0 0 0 (1-2*NU)/2 0 0 ; 0 0 0 0 (1- 2*NU)/2 0 ; 0 0 0 0 0 (1- 2*NU)/2];
BD = transpose(Bnew)*D*Bnew/Jnew;
r = int(int(int(BD, u, -1, 1), t, -1, 1), s, -1, 1);
w = double(r);
```

---

```
function y = LinearBrickAssemble(K,k,i,j,m,n,p,q,r,s)
% LinearBrickAssemble    This function assembles the element stiffness
%                        matrix k of the linear brick (solid)
%                        element with nodes i, j, m, n, p, q, r,
%                        and s into the global stiffness matrix K.
%                        This function returns the global stiffness
%                        matrix K after the element stiffness matrix
%                        k is assembled.
K(3*i-2,3*i-2) = K(3*i-2,3*i-2) + k(1,1);
K(3*i-2,3*i-1) = K(3*i-2,3*i-1) + k(1,2);
K(3*i-2,3*i) = K(3*i-2,3*i) + k(1,3);
K(3*i-2,3*j-2) = K(3*i-2,3*j-2) + k(1,4);
K(3*i-2,3*j-1) = K(3*i-2,3*j-1) + k(1,5);
K(3*i-2,3*j) = K(3*i-2,3*j) + k(1,6);
K(3*i-2,3*m-2) = K(3*i-2,3*m-2) + k(1,7);
K(3*i-2,3*m-1) = K(3*i-2,3*m-1) + k(1,8);
K(3*i-2,3*m) = K(3*i-2,3*m) + k(1,9);
K(3*i-2,3*n-2) = K(3*i-2,3*n-2) + k(1,10);
K(3*i-2,3*n-1) = K(3*i-2,3*n-1) + k(1,11);
K(3*i-2,3*n) = K(3*i-2,3*n) + k(1,12);
K(3*i-2,3*p-2) = K(3*i-2,3*p-2) + k(1,13);
K(3*i-2,3*p-1) = K(3*i-2,3*p-1) + k(1,14);
K(3*i-2,3*p) = K(3*i-2,3*p) + k(1,15);
K(3*i-2,3*q-2) = K(3*i-2,3*q-2) + k(1,16);
K(3*i-2,3*q-1) = K(3*i-2,3*q-1) + k(1,17);
K(3*i-2,3*q) = K(3*i-2,3*q) + k(1,18);
K(3*i-2,3*r-2) = K(3*i-2,3*r-2) + k(1,19);
K(3*i-2,3*r-1) = K(3*i-2,3*r-1) + k(1,20);
K(3*i-2,3*r) = K(3*i-2,3*r) + k(1,21);
K(3*i-2,3*s-2) = K(3*i-2,3*s-2) + k(1,22);
K(3*i-2,3*s-1) = K(3*i-2,3*s-1) + k(1,23);
K(3*i-2,3*s) = K(3*i-2,3*s) + k(1,24);
K(3*i-1,3*i-2) = K(3*i-1,3*i-2) + k(2,1);
K(3*i-1,3*i-1) = K(3*i-1,3*i-1) + k(2,2);
K(3*i-1,3*i) = K(3*i-1,3*i) + k(2,3);
K(3*i-1,3*j-2) = K(3*i-1,3*j-2) + k(2,4);
K(3*i-1,3*j-1) = K(3*i-1,3*j-1) + k(2,5);
K(3*i-1,3*j) = K(3*i-1,3*j) + k(2,6);
K(3*i-1,3*m-2) = K(3*i-1,3*m-2) + k(2,7);
K(3*i-1,3*m-1) = K(3*i-1,3*m-1) + k(2,8);
K(3*i-1,3*m) = K(3*i-1,3*m) + k(2,9);
```

```
K(3*i-1,3*n-2) = K(3*i-1,3*n-2) + k(2,10);
K(3*i-1,3*n-1) = K(3*i-1,3*n-1) + k(2,11);
K(3*i-1,3*n) = K(3*i-1,3*n) + k(2,12);
K(3*i-1,3*p-2) = K(3*i-1,3*p-2) + k(2,13);
K(3*i-1,3*p-1) = K(3*i-1,3*p-1) + k(2,14);
K(3*i-1,3*p) = K(3*i-1,3*p) + k(2,15);
K(3*i-1,3*q-2) = K(3*i-1,3*q-2) + k(2,16);
K(3*i-1,3*q-1) = K(3*i-1,3*q-1) + k(2,17);
K(3*i-1,3*q) = K(3*i-1,3*q) + k(2,18);
K(3*i-1,3*r-2) = K(3*i-1,3*r-2) + k(2,19);
K(3*i-1,3*r-1) = K(3*i-1,3*r-1) + k(2,20);
K(3*i-1,3*r) = K(3*i-1,3*r) + k(2,21);
K(3*i-1,3*s-2) = K(3*i-1,3*s-2) + k(2,22);
K(3*i-1,3*s-1) = K(3*i-1,3*s-1) + k(2,23);
K(3*i-1,3*s) = K(3*i-1,3*s) + k(2,24);
K(3*i,3*i-2) = K(3*i,3*i-2) + k(3,1);
K(3*i,3*i-1) = K(3*i,3*i-1) + k(3,2);
K(3*i,3*i) = K(3*i,3*i) + k(3,3);
K(3*i,3*j-2) = K(3*i,3*j-2) + k(3,4);
K(3*i,3*j-1) = K(3*i,3*j-1) + k(3,5);
K(3*i,3*j) = K(3*i,3*j) + k(3,6);
K(3*i,3*m-2) = K(3*i,3*m-2) + k(3,7);
K(3*i,3*m-1) = K(3*i,3*m-1) + k(3,8);
K(3*i,3*m) = K(3*i,3*m) + k(3,9);
K(3*i,3*n-2) = K(3*i,3*n-2) + k(3,10);
K(3*i,3*n-1) = K(3*i,3*n-1) + k(3,11);
K(3*i,3*n) = K(3*i,3*n) + k(3,12);
K(3*i,3*p-2) = K(3*i,3*p-2) + k(3,13);
K(3*i,3*p-1) = K(3*i,3*p-1) + k(3,14);
K(3*i,3*p) = K(3*i,3*p) + k(3,15);
K(3*i,3*q-2) = K(3*i,3*q-2) + k(3,16);
K(3*i,3*q-1) = K(3*i,3*q-1) + k(3,17);
K(3*i,3*q) = K(3*i,3*q) + k(3,18);
K(3*i,3*r-2) = K(3*i,3*r-2) + k(3,19);
K(3*i,3*r-1) = K(3*i,3*r-1) + k(3,20);
K(3*i,3*r) = K(3*i,3*r) + k(3,21);
K(3*i,3*s-2) = K(3*i,3*s-2) + k(3,22);
K(3*i,3*s-1) = K(3*i,3*s-1) + k(3,23);
K(3*i,3*s) = K(3*i,3*s) + k(3,24);
K(3*j-2,3*i-2) = K(3*j-2,3*i-2) + k(4,1);
K(3*j-2,3*i-1) = K(3*j-2,3*i-1) + k(4,2);
K(3*j-2,3*i) = K(3*j-2,3*i) + k(4,3);
K(3*j-2,3*j-2) = K(3*j-2,3*j-2) + k(4,4);
K(3*j-2,3*j-1) = K(3*j-2,3*j-1) + k(4,5);
K(3*j-2,3*j) = K(3*j-2,3*j) + k(4,6);
K(3*j-2,3*m-2) = K(3*j-2,3*m-2) + k(4,7);
K(3*j-2,3*m-1) = K(3*j-2,3*m-1) + k(4,8);
K(3*j-2,3*m) = K(3*j-2,3*m) + k(4,9);
K(3*j-2,3*n-2) = K(3*j-2,3*n-2) + k(4,10);
K(3*j-2,3*n-1) = K(3*j-2,3*n-1) + k(4,11);
K(3*j-2,3*n) = K(3*j-2,3*n) + k(4,12);
K(3*j-2,3*p-2) = K(3*j-2,3*p-2) + k(4,13);
K(3*j-2,3*p-1) = K(3*j-2,3*p-1) + k(4,14);
```

```
K(3*j-2,3*p) = K(3*j-2,3*p) + k(4,15);
K(3*j-2,3*q-2) = K(3*j-2,3*q-2) + k(4,16);
K(3*j-2,3*q-1) = K(3*j-2,3*q-1) + k(4,17);
K(3*j-2,3*q) = K(3*j-2,3*q) + k(4,18);
K(3*j-2,3*r-2) = K(3*j-2,3*r-2) + k(4,19);
K(3*j-2,3*r-1) = K(3*j-2,3*r-1) + k(4,20);
K(3*j-2,3*r) = K(3*j-2,3*r) + k(4,21);
K(3*j-2,3*s-2) = K(3*j-2,3*s-2) + k(4,22);
K(3*j-2,3*s-1) = K(3*j-2,3*s-1) + k(4,23);
K(3*j-2,3*s) = K(3*j-2,3*s) + k(4,24);
K(3*j-1,3*i-2) = K(3*j-1,3*i-2) + k(5,1);
K(3*j-1,3*i-1) = K(3*j-1,3*i-1) + k(5,2);
K(3*j-1,3*i) = K(3*j-1,3*i) + k(5,3);
K(3*j-1,3*j-2) = K(3*j-1,3*j-2) + k(5,4);
K(3*j-1,3*j-1) = K(3*j-1,3*j-1) + k(5,5);
K(3*j-1,3*j) = K(3*j-1,3*j) + k(5,6);
K(3*j-1,3*m-2) = K(3*j-1,3*m-2) + k(5,7);
K(3*j-1,3*m-1) = K(3*j-1,3*m-1) + k(5,8);
K(3*j-1,3*m) = K(3*j-1,3*m) + k(5,9);
K(3*j-1,3*n-2) = K(3*j-1,3*n-2) + k(5,10);
K(3*j-1,3*n-1) = K(3*j-1,3*n-1) + k(5,11);
K(3*j-1,3*n) = K(3*j-1,3*n) + k(5,12);
K(3*j-1,3*p-2) = K(3*j-1,3*p-2) + k(5,13);
K(3*j-1,3*p-1) = K(3*j-1,3*p-1) + k(5,14);
K(3*j-1,3*p) = K(3*j-1,3*p) + k(5,15);
K(3*j-1,3*q-2) = K(3*j-1,3*q-2) + k(5,16);
K(3*j-1,3*q-1) = K(3*j-1,3*q-1) + k(5,17);
K(3*j-1,3*q) = K(3*j-1,3*q) + k(5,18);
K(3*j-1,3*r-2) = K(3*j-1,3*r-2) + k(5,19);
K(3*j-1,3*r-1) = K(3*j-1,3*r-1) + k(5,20);
K(3*j-1,3*r) = K(3*j-1,3*r) + k(5,21);
K(3*j-1,3*s-2) = K(3*j-1,3*s-2) + k(5,22);
K(3*j-1,3*s-1) = K(3*j-1,3*s-1) + k(5,23);
K(3*j-1,3*s) = K(3*j-1,3*s) + k(5,24);
K(3*j,3*i-2) = K(3*j,3*i-2) + k(6,1);
K(3*j,3*i-1) = K(3*j,3*i-1) + k(6,2);
K(3*j,3*i) = K(3*j,3*i) + k(6,3);
K(3*j,3*j-2) = K(3*j,3*j-2) + k(6,4);
K(3*j,3*j-1) = K(3*j,3*j-1) + k(6,5);
K(3*j,3*j) = K(3*j,3*j) + k(6,6);
K(3*j,3*m-2) = K(3*j,3*m-2) + k(6,7);
K(3*j,3*m-1) = K(3*j,3*m-1) + k(6,8);
K(3*j,3*m) = K(3*j,3*m) + k(6,9);
K(3*j,3*n-2) = K(3*j,3*n-2) + k(6,10);
K(3*j,3*n-1) = K(3*j,3*n-1) + k(6,11);
K(3*j,3*n) = K(3*j,3*n) + k(6,12);
K(3*j,3*p-2) = K(3*j,3*p-2) + k(6,13);
K(3*j,3*p-1) = K(3*j,3*p-1) + k(6,14);
K(3*j,3*p) = K(3*j,3*p) + k(6,15);
K(3*j,3*q-2) = K(3*j,3*q-2) + k(6,16);
K(3*j,3*q-1) = K(3*j,3*q-1) + k(6,17);
K(3*j,3*q) = K(3*j,3*q) + k(6,18);
K(3*j,3*r-2) = K(3*j,3*r-2) + k(6,19);
```

```
K(3*j,3*r-1) = K(3*j,3*r-1) + k(6,20);
K(3*j,3*r) = K(3*j,3*r) + k(6,21);
K(3*j,3*s-2) = K(3*j,3*s-2) + k(6,22);
K(3*j,3*s-1) = K(3*j,3*s-1) + k(6,23);
K(3*j,3*s) = K(3*j,3*s) + k(6,24);
K(3*m-2,3*i-2) = K(3*m-2,3*i-2) + k(7,1);
K(3*m-2,3*i-1) = K(3*m-2,3*i-1) + k(7,2);
K(3*m-2,3*i) = K(3*m-2,3*i) + k(7,3);
K(3*m-2,3*j-2) = K(3*m-2,3*j-2) + k(7,4);
K(3*m-2,3*j-1) = K(3*m-2,3*j-1) + k(7,5);
K(3*m-2,3*j) = K(3*m-2,3*j) + k(7,6);
K(3*m-2,3*m-2) = K(3*m-2,3*m-2) + k(7,7);
K(3*m-2,3*m-1) = K(3*m-2,3*m-1) + k(7,8);
K(3*m-2,3*m) = K(3*m-2,3*m) + k(7,9);
K(3*m-2,3*n-2) = K(3*m-2,3*n-2) + k(7,10);
K(3*m-2,3*n-1) = K(3*m-2,3*n-1) + k(7,11);
K(3*m-2,3*n) = K(3*m-2,3*n) + k(7,12);
K(3*m-2,3*p-2) = K(3*m-2,3*p-2) + k(7,13);
K(3*m-2,3*p-1) = K(3*m-2,3*p-1) + k(7,14);
K(3*m-2,3*p) = K(3*m-2,3*p) + k(7,15);
K(3*m-2,3*q-2) = K(3*m-2,3*q-2) + k(7,16);
K(3*m-2,3*q-1) = K(3*m-2,3*q-1) + k(7,17);
K(3*m-2,3*q) = K(3*m-2,3*q) + k(7,18);
K(3*m-2,3*r-2) = K(3*m-2,3*r-2) + k(7,19);
K(3*m-2,3*r-1) = K(3*m-2,3*r-1) + k(7,20);
K(3*m-2,3*r) = K(3*m-2,3*r) + k(7,21);
K(3*m-2,3*s-2) = K(3*m-2,3*s-2) + k(7,22);
K(3*m-2,3*s-1) = K(3*m-2,3*s-1) + k(7,23);
K(3*m-2,3*s) = K(3*m-2,3*s) + k(7,24);
K(3*m-1,3*i-2) = K(3*m-1,3*i-2) + k(8,1);
K(3*m-1,3*i-1) = K(3*m-1,3*i-1) + k(8,2);
K(3*m-1,3*i) = K(3*m-1,3*i) + k(8,3);
K(3*m-1,3*j-2) = K(3*m-1,3*j-2) + k(8,4);
K(3*m-1,3*j-1) = K(3*m-1,3*j-1) + k(8,5);
K(3*m-1,3*j) = K(3*m-1,3*j) + k(8,6);
K(3*m-1,3*m-2) = K(3*m-1,3*m-2) + k(8,7);
K(3*m-1,3*m-1) = K(3*m-1,3*m-1) + k(8,8);
K(3*m-1,3*m) = K(3*m-1,3*m) + k(8,9);
K(3*m-1,3*n-2) = K(3*m-1,3*n-2) + k(8,10);
K(3*m-1,3*n-1) = K(3*m-1,3*n-1) + k(8,11);
K(3*m-1,3*n) = K(3*m-1,3*n) + k(8,12);
K(3*m-1,3*p-2) = K(3*m-1,3*p-2) + k(8,13);
K(3*m-1,3*p-1) = K(3*m-1,3*p-1) + k(8,14);
K(3*m-1,3*p) = K(3*m-1,3*p) + k(8,15);
K(3*m-1,3*q-2) = K(3*m-1,3*q-2) + k(8,16);
K(3*m-1,3*q-1) = K(3*m-1,3*q-1) + k(8,17);
K(3*m-1,3*q) = K(3*m-1,3*q) + k(8,18);
K(3*m-1,3*r-2) = K(3*m-1,3*r-2) + k(8,19);
K(3*m-1,3*r-1) = K(3*m-1,3*r-1) + k(8,20);
K(3*m-1,3*r) = K(3*m-1,3*r) + k(8,21);
K(3*m-1,3*s-2) = K(3*m-1,3*s-2) + k(8,22);
K(3*m-1,3*s-1) = K(3*m-1,3*s-1) + k(8,23);
K(3*m-1,3*s) = K(3*m-1,3*s) + k(8,24);
```

```
K(3*m,3*i-2) = K(3*m,3*i-2) + k(9,1);
K(3*m,3*i-1) = K(3*m,3*i-1) + k(9,2);
K(3*m,3*i) = K(3*m,3*i) + k(9,3);
K(3*m,3*j-2) = K(3*m,3*j-2) + k(9,4);
K(3*m,3*j-1) = K(3*m,3*j-1) + k(9,5);
K(3*m,3*j) = K(3*m,3*j) + k(9,6);
K(3*m,3*m-2) = K(3*m,3*m-2) + k(9,7);
K(3*m,3*m-1) = K(3*m,3*m-1) + k(9,8);
K(3*m,3*m) = K(3*m,3*m) + k(9,9);
K(3*m,3*n-2) = K(3*m,3*n-2) + k(9,10);
K(3*m,3*n-1) = K(3*m,3*n-1) + k(9,11);
K(3*m,3*n) = K(3*m,3*n) + k(9,12);
K(3*m,3*p-2) = K(3*m,3*p-2) + k(9,13);
K(3*m,3*p-1) = K(3*m,3*p-1) + k(9,14);
K(3*m,3*p) = K(3*m,3*p) + k(9,15);
K(3*m,3*q-2) = K(3*m,3*q-2) + k(9,16);
K(3*m,3*q-1) = K(3*m,3*q-1) + k(9,17);
K(3*m,3*q) = K(3*m,3*q) + k(9,18);
K(3*m,3*r-2) = K(3*m,3*r-2) + k(9,19);
K(3*m,3*r-1) = K(3*m,3*r-1) + k(9,20);
K(3*m,3*r) = K(3*m,3*r) + k(9,21);
K(3*m,3*s-2) = K(3*m,3*s-2) + k(9,22);
K(3*m,3*s-1) = K(3*m,3*s-1) + k(9,23);
K(3*m,3*s) = K(3*m,3*s) + k(9,24);
K(3*n-2,3*i-2) = K(3*n-2,3*i-2) + k(10,1);
K(3*n-2,3*i-1) = K(3*n-2,3*i-1) + k(10,2);
K(3*n-2,3*i) = K(3*n-2,3*i) + k(10,3);
K(3*n-2,3*j-2) = K(3*n-2,3*j-2) + k(10,4);
K(3*n-2,3*j-1) = K(3*n-2,3*j-1) + k(10,5);
K(3*n-2,3*j) = K(3*n-2,3*j) + k(10,6);
K(3*n-2,3*m-2) = K(3*n-2,3*m-2) + k(10,7);
K(3*n-2,3*m-1) = K(3*n-2,3*m-1) + k(10,8);
K(3*n-2,3*m) = K(3*n-2,3*m) + k(10,9);
K(3*n-2,3*n-2) = K(3*n-2,3*n-2) + k(10,10);
K(3*n-2,3*n-1) = K(3*n-2,3*n-1) + k(10,11);
K(3*n-2,3*n) = K(3*n-2,3*n) + k(10,12);
K(3*n-2,3*p-2) = K(3*n-2,3*p-2) + k(10,13);
K(3*n-2,3*p-1) = K(3*n-2,3*p-1) + k(10,14);
K(3*n-2,3*p) = K(3*n-2,3*p) + k(10,15);
K(3*n-2,3*q-2) = K(3*n-2,3*q-2) + k(10,16);
K(3*n-2,3*q-1) = K(3*n-2,3*q-1) + k(10,17);
K(3*n-2,3*q) = K(3*n-2,3*q) + k(10,18);
K(3*n-2,3*r-2) = K(3*n-2,3*r-2) + k(10,19);
K(3*n-2,3*r-1) = K(3*n-2,3*r-1) + k(10,20);
K(3*n-2,3*r) = K(3*n-2,3*r) + k(10,21);
K(3*n-2,3*s-2) = K(3*n-2,3*s-2) + k(10,22);
K(3*n-2,3*s-1) = K(3*n-2,3*s-1) + k(10,23);
K(3*n-2,3*s) = K(3*n-2,3*s) + k(10,24);
K(3*n-1,3*i-2) = K(3*n-1,3*i-2) + k(11,1);
K(3*n-1,3*i-1) = K(3*n-1,3*i-1) + k(11,2);
K(3*n-1,3*i) = K(3*n-1,3*i) + k(11,3);
K(3*n-1,3*j-2) = K(3*n-1,3*j-2) + k(11,4);
K(3*n-1,3*j-1) = K(3*n-1,3*j-1) + k(11,5);
K(3*n-1,3*j) = K(3*n-1,3*j) + k(11,6);
K(3*n-1,3*m-2) = K(3*n-1,3*m-2) + k(11,7);
```

```
K(3*n-1,3*m-1) = K(3*n-1,3*m-1) + k(11,8);
K(3*n-1,3*m) = K(3*n-1,3*m) + k(11,9);
K(3*n-1,3*n-2) = K(3*n-1,3*n-2) + k(11,10);
K(3*n-1,3*n-1) = K(3*n-1,3*n-1) + k(11,11);
K(3*n-1,3*n) = K(3*n-1,3*n) + k(11,12);
K(3*n-1,3*p-2) = K(3*n-1,3*p-2) + k(11,13);
K(3*n-1,3*p-1) = K(3*n-1,3*p-1) + k(11,14);
K(3*n-1,3*p) = K(3*n-1,3*p) + k(11,15);
K(3*n-1,3*q-2) = K(3*n-1,3*q-2) + k(11,16);
K(3*n-1,3*q-1) = K(3*n-1,3*q-1) + k(11,17);
K(3*n-1,3*q) = K(3*n-1,3*q) + k(11,18);
K(3*n-1,3*r-2) = K(3*n-1,3*r-2) + k(11,19);
K(3*n-1,3*r-1) = K(3*n-1,3*r-1) + k(11,20);
K(3*n-1,3*r) = K(3*n-1,3*r) + k(11,21);
K(3*n-1,3*s-2) = K(3*n-1,3*s-2) + k(11,22);
K(3*n-1,3*s-1) = K(3*n-1,3*s-1) + k(11,23);
K(3*n-1,3*s) = K(3*n-1,3*s) + k(11,24);
K(3*n,3*i-2) = K(3*n,3*i-2) + k(12,1);
K(3*n,3*i-1) = K(3*n,3*i-1) + k(12,2);
K(3*n,3*i) = K(3*n,3*i) + k(12,3);
K(3*n,3*j-2) = K(3*n,3*j-2) + k(12,4);
K(3*n,3*j-1) = K(3*n,3*j-1) + k(12,5);
K(3*n,3*j) = K(3*n,3*j) + k(12,6);
K(3*n,3*m-2) = K(3*n,3*m-2) + k(12,7);
K(3*n,3*m-1) = K(3*n,3*m-1) + k(12,8);
K(3*n,3*m) = K(3*n,3*m) + k(12,9);
K(3*n,3*n-2) = K(3*n,3*n-2) + k(12,10);
K(3*n,3*n-1) = K(3*n,3*n-1) + k(12,11);
K(3*n,3*n) = K(3*n,3*n) + k(12,12);
K(3*n,3*p-2) = K(3*n,3*p-2) + k(12,13);
K(3*n,3*p-1) = K(3*n,3*p-1) + k(12,14);
K(3*n,3*p) = K(3*n,3*p) + k(12,15);
K(3*n,3*q-2) = K(3*n,3*q-2) + k(12,16);
K(3*n,3*q-1) = K(3*n,3*q-1) + k(12,17);
K(3*n,3*q) = K(3*n,3*q) + k(12,18);
K(3*n,3*r-2) = K(3*n,3*r-2) + k(12,19);
K(3*n,3*r-1) = K(3*n,3*r-1) + k(12,20);
K(3*n,3*r) = K(3*n,3*r) + k(12,21);
K(3*n,3*s-2) = K(3*n,3*s-2) + k(12,22);
K(3*n,3*s-1) = K(3*n,3*s-1) + k(12,23);
K(3*n,3*s) = K(3*n,3*s) + k(12,24);
K(3*p-2,3*i-2) = K(3*p-2,3*i-2) + k(13,1);
K(3*p-2,3*i-1) = K(3*p-2,3*i-1) + k(13,2);
K(3*p-2,3*i) = K(3*p-2,3*i) + k(13,3);
K(3*p-2,3*j-2) = K(3*p-2,3*j-2) + k(13,4);
K(3*p-2,3*j-1) = K(3*p-2,3*j-1) + k(13,5);
K(3*p-2,3*j) = K(3*p-2,3*j) + k(13,6);
K(3*p-2,3*m-2) = K(3*p-2,3*m-2) + k(13,7);
K(3*p-2,3*m-1) = K(3*p-2,3*m-1) + k(13,8);
K(3*p-2,3*m) = K(3*p-2,3*m) + k(13,9);
K(3*p-2,3*n-2) = K(3*p-2,3*n-2) + k(13,10);
K(3*p-2,3*n-1) = K(3*p-2,3*n-1) + k(13,11);
K(3*p-2,3*n) = K(3*p-2,3*n) + k(13,12);
K(3*p-2,3*p-2) = K(3*p-2,3*p-2) + k(13,13);
```

```
K(3*p-2,3*p-1) = K(3*p-2,3*p-1) + k(13,14);
K(3*p-2,3*p) = K(3*p-2,3*p) + k(13,15);
K(3*p-2,3*q-2) = K(3*p-2,3*q-2) + k(13,16);
K(3*p-2,3*q-1) = K(3*p-2,3*q-1) + k(13,17);
K(3*p-2,3*q) = K(3*p-2,3*q) + k(13,18);
K(3*p-2,3*r-2) = K(3*p-2,3*r-2) + k(13,19);
K(3*p-2,3*r-1) = K(3*p-2,3*r-1) + k(13,20);
K(3*p-2,3*r) = K(3*p-2,3*r) + k(13,21);
K(3*p-2,3*s-2) = K(3*p-2,3*s-2) + k(13,22);
K(3*p-2,3*s-1) = K(3*p-2,3*s-1) + k(13,23);
K(3*p-2,3*s) = K(3*p-2,3*s) + k(13,24);
K(3*p-1,3*i-2) = K(3*p-1,3*i-2) + k(14,1);
K(3*p-1,3*i-1) = K(3*p-1,3*i-1) + k(14,2);
K(3*p-1,3*i) = K(3*p-1,3*i) + k(14,3);
K(3*p-1,3*j-2) = K(3*p-1,3*j-2) + k(14,4);
K(3*p-1,3*j-1) = K(3*p-1,3*j-1) + k(14,5);
K(3*p-1,3*j) = K(3*p-1,3*j) + k(14,6);
K(3*p-1,3*m-2) = K(3*p-1,3*m-2) + k(14,7);
K(3*p-1,3*m-1) = K(3*p-1,3*m-1) + k(14,8);
K(3*p-1,3*m) = K(3*p-1,3*m) + k(14,9);
K(3*p-1,3*n-2) = K(3*p-1,3*n-2) + k(14,10);
K(3*p-1,3*n-1) = K(3*p-1,3*n-1) + k(14,11);
K(3*p-1,3*n) = K(3*p-1,3*n) + k(14,12);
K(3*p-1,3*p-2) = K(3*p-1,3*p-2) + k(14,13);
K(3*p-1,3*p-1) = K(3*p-1,3*p-1) + k(14,14);
K(3*p-1,3*p) = K(3*p-1,3*p) + k(14,15);
K(3*p-1,3*q-2) = K(3*p-1,3*q-2) + k(14,16);
K(3*p-1,3*q-1) = K(3*p-1,3*q-1) + k(14,17);
K(3*p-1,3*q) = K(3*p-1,3*q) + k(14,18);
K(3*p-1,3*r-2) = K(3*p-1,3*r-2) + k(14,19);
K(3*p-1,3*r-1) = K(3*p-1,3*r-1) + k(14,20);
K(3*p-1,3*r) = K(3*p-1,3*r) + k(14,21);
K(3*p-1,3*s-2) = K(3*p-1,3*s-2) + k(14,22);
K(3*p-1,3*s-1) = K(3*p-1,3*s-1) + k(14,23);
K(3*p-1,3*s) = K(3*p-1,3*s) + k(14,24);
K(3*p,3*i-2) = K(3*p,3*i-2) + k(15,1);
K(3*p,3*i-1) = K(3*p,3*i-1) + k(15,2);
K(3*p,3*i) = K(3*p,3*i) + k(15,3);
K(3*p,3*j-2) = K(3*p,3*j-2) + k(15,4);
K(3*p,3*j-1) = K(3*p,3*j-1) + k(15,5);
K(3*p,3*j) = K(3*p,3*j) + k(15,6);
K(3*p,3*m-2) = K(3*p,3*m-2) + k(15,7);
K(3*p,3*m-1) = K(3*p,3*m-1) + k(15,8);
K(3*p,3*m) = K(3*p,3*m) + k(15,9);
K(3*p,3*n-2) = K(3*p,3*n-2) + k(15,10);
K(3*p,3*n-1) = K(3*p,3*n-1) + k(15,11);
K(3*p,3*n) = K(3*p,3*n) + k(15,12);
K(3*p,3*p-2) = K(3*p,3*p-2) + k(15,13);
K(3*p,3*p-1) = K(3*p,3*p-1) + k(15,14);
K(3*p,3*p) = K(3*p,3*p) + k(15,15);
K(3*p,3*q-2) = K(3*p,3*q-2) + k(15,16);
K(3*p,3*q-1) = K(3*p,3*q-1) + k(15,17);
K(3*p,3*q) = K(3*p,3*q) + k(15,18);
K(3*p,3*r-2) = K(3*p,3*r-2) + k(15,19);
```

```
K(3*p,3*r-1) = K(3*p,3*r-1) + k(15,20);
K(3*p,3*r) = K(3*p,3*r) + k(15,21);
K(3*p,3*s-2) = K(3*p,3*s-2) + k(15,22);
K(3*p,3*s-1) = K(3*p,3*s-1) + k(15,23);
K(3*p,3*s) = K(3*p,3*s) + k(15,24);
K(3*q-2,3*i-2) = K(3*q-2,3*i-2) + k(16,1);
K(3*q-2,3*i-1) = K(3*q-2,3*i-1) + k(16,2);
K(3*q-2,3*i) = K(3*q-2,3*i) + k(16,3);
K(3*q-2,3*j-2) = K(3*q-2,3*j-2) + k(16,4);
K(3*q-2,3*j-1) = K(3*q-2,3*j-1) + k(16,5);
K(3*q-2,3*j) = K(3*q-2,3*j) + k(16,6);
K(3*q-2,3*m-2) = K(3*q-2,3*m-2) + k(16,7);
K(3*q-2,3*m-1) = K(3*q-2,3*m-1) + k(16,8);
K(3*q-2,3*m) = K(3*q-2,3*m) + k(16,9);
K(3*q-2,3*n-2) = K(3*q-2,3*n-2) + k(16,10);
K(3*q-2,3*n-1) = K(3*q-2,3*n-1) + k(16,11);
K(3*q-2,3*n) = K(3*q-2,3*n) + k(16,12);
K(3*q-2,3*p-2) = K(3*q-2,3*p-2) + k(16,13);
K(3*q-2,3*p-1) = K(3*q-2,3*p-1) + k(16,14);
K(3*q-2,3*p) = K(3*q-2,3*p) + k(16,15);
K(3*q-2,3*q-2) = K(3*q-2,3*q-2) + k(16,16);
K(3*q-2,3*q-1) = K(3*q-2,3*q-1) + k(16,17);
K(3*q-2,3*q) = K(3*q-2,3*q) + k(16,18);
K(3*q-2,3*r-2) = K(3*q-2,3*r-2) + k(16,19);
K(3*q-2,3*r-1) = K(3*q-2,3*r-1) + k(16,20);
K(3*q-2,3*r) = K(3*q-2,3*r) + k(16,21);
K(3*q-2,3*s-2) = K(3*q-2,3*s-2) + k(16,22);
K(3*q-2,3*s-1) = K(3*q-2,3*s-1) + k(16,23);
K(3*q-2,3*s) = K(3*q-2,3*s) + k(16,24);
K(3*q-1,3*i-2) = K(3*q-1,3*i-2) + k(17,1);
K(3*q-1,3*i-1) = K(3*q-1,3*i-1) + k(17,2);
K(3*q-1,3*i) = K(3*q-1,3*i) + k(17,3);
K(3*q-1,3*j-2) = K(3*q-1,3*j-2) + k(17,4);
K(3*q-1,3*j-1) = K(3*q-1,3*j-1) + k(17,5);
K(3*q-1,3*j) = K(3*q-1,3*j) + k(17,6);
K(3*q-1,3*m-2) = K(3*q-1,3*m-2) + k(17,7);
K(3*q-1,3*m-1) = K(3*q-1,3*m-1) + k(17,8);
K(3*q-1,3*m) = K(3*q-1,3*m) + k(17,9);
K(3*q-1,3*n-2) = K(3*q-1,3*n-2) + k(17,10);
K(3*q-1,3*n-1) = K(3*q-1,3*n-1) + k(17,11);
K(3*q-1,3*n) = K(3*q-1,3*n) + k(17,12);
K(3*q-1,3*p-2) = K(3*q-1,3*p-2) + k(17,13);
K(3*q-1,3*p-1) = K(3*q-1,3*p-1) + k(17,14);
K(3*q-1,3*p) = K(3*q-1,3*p) + k(17,15);
K(3*q-1,3*q-2) = K(3*q-1,3*q-2) + k(17,16);
K(3*q-1,3*q-1) = K(3*q-1,3*q-1) + k(17,17);
K(3*q-1,3*q) = K(3*q-1,3*q) + k(17,18);
K(3*q-1,3*r-2) = K(3*q-1,3*r-2) + k(17,19);
K(3*q-1,3*r-1) = K(3*q-1,3*r-1) + k(17,20);
K(3*q-1,3*r) = K(3*q-1,3*r) + k(17,21);
K(3*q-1,3*s-2) = K(3*q-1,3*s-2) + k(17,22);
K(3*q-1,3*s-1) = K(3*q-1,3*s-1) + k(17,23);
```

```
K(3*q-1,3*s) = K(3*q-1,3*s) + k(17,24);
K(3*q,3*i-2) = K(3*q,3*i-2) + k(18,1);
K(3*q,3*i-1) = K(3*q,3*i-1) + k(18,2);
K(3*q,3*i) = K(3*q,3*i) + k(18,3);
K(3*q,3*j-2) = K(3*q,3*j-2) + k(18,4);
K(3*q,3*j-1) = K(3*q,3*j-1) + k(18,5);
K(3*q,3*j) = K(3*q,3*j) + k(18,6);
K(3*q,3*m-2) = K(3*q,3*m-2) + k(18,7);
K(3*q,3*m-1) = K(3*q,3*m-1) + k(18,8);
K(3*q,3*m) = K(3*q,3*m) + k(18,9);
K(3*q,3*n-2) = K(3*q,3*n-2) + k(18,10);
K(3*q,3*n-1) = K(3*q,3*n-1) + k(18,11);
K(3*q,3*n) = K(3*q,3*n) + k(18,12);
K(3*q,3*p-2) = K(3*q,3*p-2) + k(18,13);
K(3*q,3*p-1) = K(3*q,3*p-1) + k(18,14);
K(3*q,3*p) = K(3*q,3*p) + k(18,15);
K(3*q,3*q-2) = K(3*q,3*q-2) + k(18,16);
K(3*q,3*q-1) = K(3*q,3*q-1) + k(18,17);
K(3*q,3*q) = K(3*q,3*q) + k(18,18);
K(3*q,3*r-2) = K(3*q,3*r-2) + k(18,19);
K(3*q,3*r-1) = K(3*q,3*r-1) + k(18,20);
K(3*q,3*r) = K(3*q,3*r) + k(18,21);
K(3*q,3*s-2) = K(3*q,3*s-2) + k(18,22);
K(3*q,3*s-1) = K(3*q,3*s-1) + k(18,23);
K(3*q,3*s) = K(3*q,3*s) + k(18,24);
K(3*r-2,3*i-2) = K(3*r-2,3*i-2) + k(19,1);
K(3*r-2,3*i-1) = K(3*r-2,3*i-1) + k(19,2);
K(3*r-2,3*i) = K(3*r-2,3*i) + k(19,3);
K(3*r-2,3*j-2) = K(3*r-2,3*j-2) + k(19,4);
K(3*r-2,3*j-1) = K(3*r-2,3*j-1) + k(19,5);
K(3*r-2,3*j) = K(3*r-2,3*j) + k(19,6);
K(3*r-2,3*m-2) = K(3*r-2,3*m-2) + k(19,7);
K(3*r-2,3*m-1) = K(3*r-2,3*m-1) + k(19,8);
K(3*r-2,3*m) = K(3*r-2,3*m) + k(19,9);
K(3*r-2,3*n-2) = K(3*r-2,3*n-2) + k(19,10);
K(3*r-2,3*n-1) = K(3*r-2,3*n-1) + k(19,11);
K(3*r-2,3*n) = K(3*r-2,3*n) + k(19,12);
K(3*r-2,3*p-2) = K(3*r-2,3*p-2) + k(19,13);
K(3*r-2,3*p-1) = K(3*r-2,3*p-1) + k(19,14);
K(3*r-2,3*p) = K(3*r-2,3*p) + k(19,15);
K(3*r-2,3*q-2) = K(3*r-2,3*q-2) + k(19,16);
K(3*r-2,3*q-1) = K(3*r-2,3*q-1) + k(19,17);
K(3*r-2,3*q) = K(3*r-2,3*q) + k(19,18);
K(3*r-2,3*r-2) = K(3*r-2,3*r-2) + k(19,19);
K(3*r-2,3*r-1) = K(3*r-2,3*r-1) + k(19,20);
K(3*r-2,3*r) = K(3*r-2,3*r) + k(19,21);
K(3*r-2,3*s-2) = K(3*r-2,3*s-2) + k(19,22);
K(3*r-2,3*s-1) = K(3*r-2,3*s-1) + k(19,23);
K(3*r-2,3*s) = K(3*r-2,3*s) + k(19,24);
K(3*r-1,3*i-2) = K(3*r-1,3*i-2) + k(20,1);
K(3*r-1,3*i-1) = K(3*r-1,3*i-1) + k(20,2);
K(3*r-1,3*i) = K(3*r-1,3*i) + k(20,3);
K(3*r-1,3*j-2) = K(3*r-1,3*j-2) + k(20,4);
K(3*r-1,3*j-1) = K(3*r-1,3*j-1) + k(20,5);
```

```
K(3*r-1,3*j) = K(3*r-1,3*j) + k(20,6);
K(3*r-1,3*m-2) = K(3*r-1,3*m-2) + k(20,7);
K(3*r-1,3*m-1) = K(3*r-1,3*m-1) + k(20,8);
K(3*r-1,3*m) = K(3*r-1,3*m) + k(20,9);
K(3*r-1,3*n-2) = K(3*r-1,3*n-2) + k(20,10);
K(3*r-1,3*n-1) = K(3*r-1,3*n-1) + k(20,11);
K(3*r-1,3*n) = K(3*r-1,3*n) + k(20,12);
K(3*r-1,3*p-2) = K(3*r-1,3*p-2) + k(20,13);
K(3*r-1,3*p-1) = K(3*r-1,3*p-1) + k(20,14);
K(3*r-1,3*p) = K(3*r-1,3*p) + k(20,15);
K(3*r-1,3*q-2) = K(3*r-1,3*q-2) + k(20,16);
K(3*r-1,3*q-1) = K(3*r-1,3*q-1) + k(20,17);
K(3*r-1,3*q) = K(3*r-1,3*q) + k(20,18);
K(3*r-1,3*r-2) = K(3*r-1,3*r-2) + k(20,19);
K(3*r-1,3*r-1) = K(3*r-1,3*r-1) + k(20,20);
K(3*r-1,3*r) = K(3*r-1,3*r) + k(20,21);
K(3*r-1,3*s-2) = K(3*r-1,3*s-2) + k(20,22);
K(3*r-1,3*s-1) = K(3*r-1,3*s-1) + k(20,23);
K(3*r-1,3*s) = K(3*r-1,3*s) + k(20,24);
K(3*r,3*i-2) = K(3*r,3*i-2) + k(21,1);
K(3*r,3*i-1) = K(3*r,3*i-1) + k(21,2);
K(3*r,3*i) = K(3*r,3*i) + k(21,3);
K(3*r,3*j-2) = K(3*r,3*j-2) + k(21,4);
K(3*r,3*j-1) = K(3*r,3*j-1) + k(21,5);
K(3*r,3*j) = K(3*r,3*j) + k(21,6);
K(3*r,3*m-2) = K(3*r,3*m-2) + k(21,7);
K(3*r,3*m-1) = K(3*r,3*m-1) + k(21,8);
K(3*r,3*m) = K(3*r,3*m) + k(21,9);
K(3*r,3*n-2) = K(3*r,3*n-2) + k(21,10);
K(3*r,3*n-1) = K(3*r,3*n-1) + k(21,11);
K(3*r,3*n) = K(3*r,3*n) + k(21,12);
K(3*r,3*p-2) = K(3*r,3*p-2) + k(21,13);
K(3*r,3*p-1) = K(3*r,3*p-1) + k(21,14);
K(3*r,3*p) = K(3*r,3*p) + k(21,15);
K(3*r,3*q-2) = K(3*r,3*q-2) + k(21,16);
K(3*r,3*q-1) = K(3*r,3*q-1) + k(21,17);
K(3*r,3*q) = K(3*r,3*q) + k(21,18);
K(3*r,3*r-2) = K(3*r,3*r-2) + k(21,19);
K(3*r,3*r-1) = K(3*r,3*r-1) + k(21,20);
K(3*r,3*r) = K(3*r,3*r) + k(21,21);
K(3*r,3*s-2) = K(3*r,3*s-2) + k(21,22);
K(3*r,3*s-1) = K(3*r,3*s-1) + k(21,23);
K(3*r,3*s) = K(3*r,3*s) + k(21,24);
K(3*s-2,3*i-2) = K(3*s-2,3*i-2) + k(22,1);
K(3*s-2,3*i-1) = K(3*s-2,3*i-1) + k(22,2);
K(3*s-2,3*i) = K(3*s-2,3*i) + k(22,3);
K(3*s-2,3*j-2) = K(3*s-2,3*j-2) + k(22,4);
K(3*s-2,3*j-1) = K(3*s-2,3*j-1) + k(22,5);
K(3*s-2,3*j) = K(3*s-2,3*j) + k(22,6);
K(3*s-2,3*m-2) = K(3*s-2,3*m-2) + k(22,7);
K(3*s-2,3*m-1) = K(3*s-2,3*m-1) + k(22,8);
K(3*s-2,3*m) = K(3*s-2,3*m) + k(22,9);
K(3*s-2,3*n-2) = K(3*s-2,3*n-2) + k(22,10);
K(3*s-2,3*n-1) = K(3*s-2,3*n-1) + k(22,11);
```

```
K(3*s-2,3*n) = K(3*s-2,3*n) + k(22,12);
K(3*s-2,3*p-2) = K(3*s-2,3*p-2) + k(22,13);
K(3*s-2,3*p-1) = K(3*s-2,3*p-1) + k(22,14);
K(3*s-2,3*p) = K(3*s-2,3*p) + k(22,15);
K(3*s-2,3*q-2) = K(3*s-2,3*q-2) + k(22,16);
K(3*s-2,3*q-1) = K(3*s-2,3*q-1) + k(22,17);
K(3*s-2,3*q) = K(3*s-2,3*q) + k(22,18);
K(3*s-2,3*r-2) = K(3*s-2,3*r-2) + k(22,19);
K(3*s-2,3*r-1) = K(3*s-2,3*r-1) + k(22,20);
K(3*s-2,3*r) = K(3*s-2,3*r) + k(22,21);
K(3*s-2,3*s-2) = K(3*s-2,3*s-2) + k(22,22);
K(3*s-2,3*s-1) = K(3*s-2,3*s-1) + k(22,23);
K(3*s-2,3*s) = K(3*s-2,3*s) + k(22,24);
K(3*s-1,3*i-2) = K(3*s-1,3*i-2) + k(23,1);
K(3*s-1,3*i-1) = K(3*s-1,3*i-1) + k(23,2);
K(3*s-1,3*i) = K(3*s-1,3*i) + k(23,3);
K(3*s-1,3*j-2) = K(3*s-1,3*j-2) + k(23,4);
K(3*s-1,3*j-1) = K(3*s-1,3*j-1) + k(23,5);
K(3*s-1,3*j) = K(3*s-1,3*j) + k(23,6);
K(3*s-1,3*m-2) = K(3*s-1,3*m-2) + k(23,7);
K(3*s-1,3*m-1) = K(3*s-1,3*m-1) + k(23,8);
K(3*s-1,3*m) = K(3*s-1,3*m) + k(23,9);
K(3*s-1,3*n-2) = K(3*s-1,3*n-2) + k(23,10);
K(3*s-1,3*n-1) = K(3*s-1,3*n-1) + k(23,11);
K(3*s-1,3*n) = K(3*s-1,3*n) + k(23,12);
K(3*s-1,3*p-2) = K(3*s-1,3*p-2) + k(23,13);
K(3*s-1,3*p-1) = K(3*s-1,3*p-1) + k(23,14);
K(3*s-1,3*p) = K(3*s-1,3*p) + k(23,15);
K(3*s-1,3*q-2) = K(3*s-1,3*q-2) + k(23,16);
K(3*s-1,3*q-1) = K(3*s-1,3*q-1) + k(23,17);
K(3*s-1,3*q) = K(3*s-1,3*q) + k(23,18);
K(3*s-1,3*r-2) = K(3*s-1,3*r-2) + k(23,19);
K(3*s-1,3*r-1) = K(3*s-1,3*r-1) + k(23,20);
K(3*s-1,3*r) = K(3*s-1,3*r) + k(23,21);
K(3*s-1,3*s-2) = K(3*s-1,3*s-2) + k(23,22);
K(3*s-1,3*s-1) = K(3*s-1,3*s-1) + k(23,23);
K(3*s-1,3*s) = K(3*s-1,3*s) + k(23,24);
K(3*s,3*i-2) = K(3*s,3*i-2) + k(24,1);
K(3*s,3*i-1) = K(3*s,3*i-1) + k(24,2);
K(3*s,3*i) = K(3*s,3*i) + k(24,3);
K(3*s,3*j-2) = K(3*s,3*j-2) + k(24,4);
K(3*s,3*j-1) = K(3*s,3*j-1) + k(24,5);
K(3*s,3*j) = K(3*s,3*j) + k(24,6);
K(3*s,3*m-2) = K(3*s,3*m-2) + k(24,7);
K(3*s,3*m-1) = K(3*s,3*m-1) + k(24,8);
K(3*s,3*m) = K(3*s,3*m) + k(24,9);
K(3*s,3*n-2) = K(3*s,3*n-2) + k(24,10);
K(3*s,3*n-1) = K(3*s,3*n-1) + k(24,11);
K(3*s,3*n) = K(3*s,3*n) + k(24,12);
K(3*s,3*p-2) = K(3*s,3*p-2) + k(24,13);
K(3*s,3*p-1) = K(3*s,3*p-1) + k(24,14);
K(3*s,3*p) = K(3*s,3*p) + k(24,15);
K(3*s,3*q-2) = K(3*s,3*q-2) + k(24,16);
```

```
K(3*s,3*q-1) = K(3*s,3*q-1) + k(24,17);
K(3*s,3*q) = K(3*s,3*q) + k(24,18);
K(3*s,3*r-2) = K(3*s,3*r-2) + k(24,19);
K(3*s,3*r-1) = K(3*s,3*r-1) + k(24,20);
K(3*s,3*r) = K(3*s,3*r) + k(24,21);
K(3*s,3*s-2) = K(3*s,3*s-2) + k(24,22);
K(3*s,3*s-1) = K(3*s,3*s-1) + k(24,23);
K(3*s,3*s) = K(3*s,3*s) + k(24,24);
y = K;
```

---

```
function w = LinearBrickElementStresses
             (E,NU,x1,y1,z1,x2,y2,z2,x3,y3,z3,x4,y4,z4,x5,y5,z5,x6,y6,
             z6,x7,y7,z7,x8,y8,z8,u)
%LinearBrickElementStresses This function returns the element
%                            stress vector for a linear brick
%                            (solid) element with modulus
%                            of elasticity E, Poisson's ratio
%                            NU, coordinates of
%                            node 1 (x1,y1,z1), coordinates
%                            of node 2 (x2,y2,z2), coordinates of
%                            node 3 (x3,y3,z3), coordinates of
%                            node 4 (x4,y4,z4), coordinates of
%                            node 5 (x5,y5,z5), coordinates
%                            of node 6 (x6,y6,z6), coordinates of
%                            node 7 (x7,y7,z7), and coordinates of
%                            node 8 (x8,y8,z8).
syms s t u;
N1 = (1-s)*(1-t)*(1+u)/8;
N2 = (1-s)*(1-t)*(1-u)/8;
N3 = (1-s)*(1+t)*(1-u)/8;
N4 = (1-s)*(1+t)*(1+u)/8;
N5 = (1+s)*(1-t)*(1+u)/8;
N6 = (1+s)*(1-t)*(1-u)/8;
N7 = (1+s)*(1+t)*(1-u)/8;
N8 = (1+s)*(1+t)*(1+u)/8;
x = N1*x1 + N2*x2 + N3*x3 + N4*x4 + N5*x5 + N6*x6 + N7*x7 + N8*x8;
y = N1*y1 + N2*y2 + N3*y3 + N4*y4 + N5*y5 + N6*y6 + N7*y7 + N8*y8;
z = N1*z1 + N2*z2 + N3*z3 + N4*z4 + N5*z5 + N6*z6 + N7*z7 + N8*z8;
xs = diff(x,s);
xt = diff(x,t);
xu = diff(x,u);
ys = diff(y,s);
yt = diff(y,t);
yu = diff(y,u);
zs = diff(z,s);
zt = diff(z,t);
zu = diff(z,u);
J = xs*(yt*zu - zt*yu) - ys*(xt*zu - zt*xu) + zs*(xt*yu - yt*xu);
N1s = diff(N1,s);
N2s = diff(N2,s);
N3s = diff(N3,s);
N4s = diff(N4,s);
N5s = diff(N5,s);
N6s = diff(N6,s);
```

```
N7s = diff(N7,s);
N8s = diff(N8,s);
N1t = diff(N1,t);
N2t = diff(N2,t);
N3t = diff(N3,t);
N4t = diff(N4,t);
N5t = diff(N5,t);
N6t = diff(N6,t);
N7t = diff(N7,t);
N8t = diff(N8,t);
N1u = diff(N1,u);
N2u = diff(N2,u);
N3u = diff(N3,u);
N4u = diff(N4,u);
N5u = diff(N5,u);
N6u = diff(N6,u);
N7u = diff(N7,u);
N8u = diff(N8,u);
% The expressions below are not divided by J - they are adjusted for
later
N1x = N1s*(yt*zu - zt*yu) - ys*(N1t*zu - zt*N1u) + zs*(N1t*yu - yt*N1u);
N2x = N2s*(yt*zu - zt*yu) - ys*(N2t*zu - zt*N2u) + zs*(N2t*yu - yt*N2u);
N3x = N3s*(yt*zu - zt*yu) - ys*(N3t*zu - zt*N3u) + zs*(N3t*yu - yt*N3u);
N4x = N4s*(yt*zu - zt*yu) - ys*(N4t*zu - zt*N4u) + zs*(N4t*yu - yt*N4u);
N5x = N5s*(yt*zu - zt*yu) - ys*(N5t*zu - zt*N5u) + zs*(N5t*yu - yt*N5u);
N6x = N6s*(yt*zu - zt*yu) - ys*(N6t*zu - zt*N6u) + zs*(N6t*yu - yt*N6u);
N7x = N7s*(yt*zu - zt*yu) - ys*(N7t*zu - zt*N7u) + zs*(N7t*yu - yt*N7u);
N8x = N8s*(yt*zu - zt*yu) - ys*(N8t*zu - zt*N8u) + zs*(N8t*yu - yt*N8u);
N1y = xs*(N1t*zu - zt*N1u) - N1s*(xt*zu - zt*xu) + zs*(xt*N1u - N1t*xu);
N2y = xs*(N2t*zu - zt*N2u) - N2s*(xt*zu - zt*xu) + zs*(xt*N2u - N2t*xu);
N3y = xs*(N3t*zu - zt*N3u) - N3s*(xt*zu - zt*xu) + zs*(xt*N3u - N3t*xu);
N4y = xs*(N4t*zu - zt*N4u) - N4s*(xt*zu - zt*xu) + zs*(xt*N4u - N4t*xu);
N5y = xs*(N5t*zu - zt*N5u) - N5s*(xt*zu - zt*xu) + zs*(xt*N5u - N5t*xu);
N6y = xs*(N6t*zu - zt*N6u) - N6s*(xt*zu - zt*xu) + zs*(xt*N6u - N6t*xu);
N7y = xs*(N7t*zu - zt*N7u) - N7s*(xt*zu - zt*xu) + zs*(xt*N7u - N7t*xu);
N8y = xs*(N8t*zu - zt*N8u) - N8s*(xt*zu - zt*xu) + zs*(xt*N8u - N8t*xu);
N1z = xs*(yt*N1u - N1t*yu) - ys*(xt*N1u - N1t*xu) + N1s*(xt*yu - yt*xu);
N2z = xs*(yt*N2u - N2t*yu) - ys*(xt*N2u - N2t*xu) + N2s*(xt*yu - yt*xu);
N3z = xs*(yt*N3u - N3t*yu) - ys*(xt*N3u - N3t*xu) + N3s*(xt*yu - yt*xu);
N4z = xs*(yt*N4u - N4t*yu) - ys*(xt*N4u - N4t*xu) + N4s*(xt*yu - yt*xu);
N5z = xs*(yt*N5u - N5t*yu) - ys*(xt*N5u - N5t*xu) + N5s*(xt*yu - yt*xu);
N6z = xs*(yt*N6u - N6t*yu) - ys*(xt*N6u - N6t*xu) + N6s*(xt*yu - yt*xu);
N7z = xs*(yt*N7u - N7t*yu) - ys*(xt*N7u - N7t*xu) + N7s*(xt*yu - yt*xu);
N8z = xs*(yt*N8u - N8t*yu) - ys*(xt*N8u - N8t*xu) + N8s*(xt*yu - yt*xu);
% Next, the B matrix is calculated explicitly as follows:
B = [N1x N2x N3x N4x N5x N6x N7x N8x 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ;
    0 0 0 0 0 0 0 0 N1y N2y N3y N4y N5y N6y N7y N8y 0 0 0 0 0 0 0 0 ;
    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 N1z N2z N3z N4z N5z N6z N7z N8z ;
    N1y N2y N3y N4y N5y N6y N7y N8y N1x N2x N3x N4x N5x N6x N7x N8x 0 0
    0 0 0 0 0 0 0 ;
    0 0 0 0 0 0 0 0 0 N1z N2z N3z N4z N5z N6z N7z N8z N1y N2y N3y N4y N5y
    N6y N7y N8y ;
    N1z N2z N3z N4z N5z N6z N7z N8z 0 0 0 0 0 0 0 0 N1x N2x N3x N4x N5x
    N6x N7x N8x];
```

```
Bnew = simplify(B);
Jnew = simplify(J);
D = (E/((1+NU)*(1-2*NU)))*[1-NU NU NU 0 0 0 ; NU 1-NU NU 0 0 0 ;
    1- NU 0 0 0 ;
    0 0 0 (1-2*NU)/2 0 0 ; 0 0 0 0 (1-2*NU)/2 0 ; 0 0 0 0 0
    (1-2*NU)/2];
Bfinal = Bnew/Jnew;
w = D*Bfinal*u
%
% We also calculate the stresses at the centroid of the element
%
wcent = subs(w, {s,t,u}, {0,0,0});
w = double(wcent);
```

---

```
function y = LinearBrickElementPStresses(sigma)
%LinearBrickElementPStresses    This function returns the three
%                               principal stresses for the element
%                               given the element stress vector.
%                               The principal angles are not returned.
s1 = sigma(1) + sigma(2) + sigma(3);
s2 = sigma(1)*sigma(2) + sigma(1)*sigma(3) + sigma(2)*sigma(3)
-sigma(4)*sigma(4) -sigma(5)*sigma(5) -sigma(6)*sigma(6);
ms3 = [sigma(1) sigma(4) sigma(6) ; sigma(4) sigma(2) sigma(5) ;
sigma(6) sigma(5) sigma(3)];
s3 = det(ms3);
y = [s1 ; s2 ; s3];
```

---

### Example 16.1:

Consider the thin plate subjected to a uniformly distributed load as shown in Fig. 16.3. Use one linear brick element to solve this problem as shown in Fig. 16.4. Given $E = 210\,\text{GPa}$, $\nu = 0.3$, $t = 0.025\,\text{m}$, and $w = 3000\,\text{kN/m}^2$, determine:

1. the global stiffness matrix for the structure.
2. the displacements at nodes 5, 6, 7, and 8.

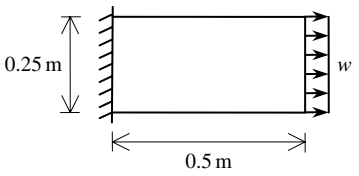


**Fig. 16.3.** Thin Plate for Example 16.1

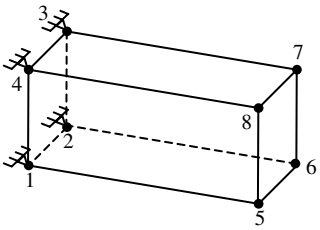


**Fig. 16.4.** Discretization of Thin Plate into One Linear Brick

### *Solution:*

Use the six steps outlined in Chap. 1 to solve this problem using the linear brick element.

### *Step 1 – Discretizing the Domain:*

We subdivide the plate into one linear brick element only for illustration purposes. More elements must be used in order to obtain reliable results. Thus the domain is subdivided into one element and eight nodes as shown in Fig. 16.4. The total force due to the distributed load is divided equally between nodes 5, 6, 7, and 8. The units used in the MATLAB calculations are kN and meter. Table 16.1 shows the element connectivity for this example.

**Table 16.1.** Element Connectivity for Example 16.1

| Element Number | Node $i$ | Node $j$ | Node $m$ | Node $n$ | Node $p$ | Node $q$ | Node $r$ | Node $s$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

### *Step 2 – Writing the Element Stiffness Matrices:*

The element stiffness matrix $k_1$ is obtained by making a call to the MATLAB function *LinearBrickElementStiffness*. This matrix has size $24 \times 24$.

```
» k1 = LinearBrickElementStiffness
       (210e6,0.3,0,0,0.025,0,0,0,0,0.25,0,0,0.25,0.0,25,
       0.5,0,0.025,0.5,0,0,0.5,0.25,0,0.5,0.25,0.025)

k1 =

   1.0e+008 *
```

Columns 1 through 10

```
  0.4571   -0.4445   -0.2256    0.2218    0.2227   -0.2252   -0.1143    0.1080    0.0042    0.0021
 -0.4445    0.4571    0.2218   -0.2256   -0.2252    0.2227    0.1080   -0.1143    0.0021    0.0042
 -0.2256    0.2218    0.4571   -0.4445   -0.1143    0.1080    0.2227   -0.2252    0.0004    0.0008
  0.2218   -0.2256   -0.4445    0.4571    0.1080   -0.1143   -0.2252    0.2227    0.0008    0.0004
  0.2227   -0.2252   -0.1143    0.1080    0.4571   -0.4445   -0.2256    0.2218   -0.0008   -0.0004
 -0.2252    0.2227    0.1080   -0.1143   -0.4445    0.4571    0.2218   -0.2256   -0.0004   -0.0008
 -0.1143    0.1080    0.2227   -0.2252   -0.2256    0.2218    0.4571   -0.4445   -0.0021   -0.0042
  0.1080   -0.1143   -0.2252    0.2227    0.2218   -0.2256   -0.4445    0.4571   -0.0042   -0.0021
  0.0042    0.0021    0.0004    0.0008   -0.0008   -0.0004   -0.0021   -0.0042    0.4655   -0.4403
  0.0021    0.0042    0.0008    0.0004   -0.0004   -0.0008   -0.0042   -0.0021   -0.4403    0.4655
 -0.0004   -0.0008   -0.0042   -0.0021    0.0021    0.0042    0.0008    0.0004   -0.2319    0.2092
 -0.0008   -0.0004   -0.0021   -0.0042    0.0042    0.0021    0.0004    0.0008    0.2092   -0.2319
  0.0008    0.0004    0.0021    0.0042   -0.0042   -0.0021   -0.0004   -0.0008    0.2311   -0.2210
  0.0004    0.0008    0.0042    0.0021   -0.0021   -0.0042   -0.0008   -0.0004   -0.2210    0.2311
 -0.0021   -0.0042   -0.0008   -0.0004    0.0004    0.0008    0.0042    0.0021   -0.1164    0.1038
 -0.0042   -0.0021   -0.0004   -0.0008    0.0008    0.0004    0.0021    0.0042    0.1038   -0.1164
 -0.0421   -0.0084   -0.0042   -0.0210    0.0084    0.0421    0.0210    0.0042   -0.0841   -0.0168
  0.0084    0.0421    0.0210    0.0042   -0.0421   -0.0084   -0.0042   -0.0210    0.0168    0.0841
  0.0042    0.0210    0.0421    0.0084   -0.0210   -0.0042   -0.0084   -0.0421    0.0841    0.0168
 -0.0210   -0.0042   -0.0084   -0.0421    0.0042    0.0210    0.0421    0.0084   -0.0168   -0.0841
 -0.0084   -0.0421   -0.0210   -0.0042    0.0421    0.0084    0.0042    0.0210   -0.0421   -0.0084
  0.0421    0.0084    0.0042    0.0210   -0.0084   -0.0421   -0.0210   -0.0042    0.0084    0.0421
  0.0210    0.0042    0.0084    0.0421   -0.0042   -0.0210   -0.0421   -0.0084    0.0421    0.0084
 -0.0042   -0.0210   -0.0421   -0.0084    0.0210    0.0042    0.0084    0.0421   -0.0084   -0.0421
```

Columns 11 through 20

```
 -0.0004   -0.0008    0.0008    0.0004   -0.0021   -0.0042   -0.0421    0.0084    0.0042   -0.0210
 -0.0008   -0.0004    0.0004    0.0008   -0.0042   -0.0021   -0.0084    0.0421    0.0210   -0.0042
 -0.0042   -0.0021    0.0021    0.0042   -0.0008   -0.0004   -0.0042    0.0210    0.0421   -0.0084
 -0.0021   -0.0042    0.0042    0.0021   -0.0004   -0.0008   -0.0210    0.0042    0.0084   -0.0421
  0.0021    0.0042   -0.0042   -0.0021    0.0004    0.0008    0.0084   -0.0421   -0.0210    0.0042
  0.0042    0.0021   -0.0021   -0.0042    0.0008    0.0004    0.0421   -0.0084   -0.0042    0.0210
  0.0008    0.0004   -0.0004   -0.0008    0.0042    0.0021    0.0210   -0.0042   -0.0084    0.0421
  0.0004    0.0008   -0.0008   -0.0004    0.0021    0.0042    0.0042   -0.0210   -0.0421    0.0084
 -0.2319    0.2092    0.2311   -0.2210   -0.1164    0.1038   -0.0841    0.0168    0.0841   -0.0168
  0.2092   -0.2319   -0.2210    0.2311    0.1038   -0.1164   -0.0168    0.0841    0.0168   -0.0841
  0.4655   -0.4403   -0.1164    0.1038    0.2311   -0.2210    0.0841   -0.0168   -0.0841    0.0168
 -0.4403    0.4655    0.1038   -0.1164   -0.2210    0.2311    0.0168   -0.0841   -0.0168    0.0841
 -0.1164    0.1038    0.4655   -0.4403   -0.2319    0.2092   -0.0421    0.0084    0.0421   -0.0084
  0.1038   -0.1164   -0.4403    0.4655    0.2092   -0.2319   -0.0084    0.0421    0.0084   -0.0421
  0.2311   -0.2210   -0.2319    0.2092    0.4655   -0.4403    0.0421   -0.0084   -0.0421    0.0084
 -0.2210    0.2311    0.2092   -0.2319   -0.4403    0.4655    0.0084   -0.0421   -0.0084    0.0421
  0.0841    0.0168   -0.0421   -0.0084    0.0421    0.0084    1.5761   -1.5677   -0.7872    0.7813
 -0.0168   -0.0841    0.0084    0.0421   -0.0084   -0.0421   -1.5677    1.5761    0.7813   -0.7872
 -0.0841   -0.0168    0.0421    0.0084   -0.0421   -0.0084   -0.7872    0.7813    1.5761   -1.5677
  0.0168    0.0841   -0.0084   -0.0421    0.0084    0.0421    0.7813   -0.7872   -1.5677    1.5761
  0.0421    0.0084   -0.0841   -0.0168    0.0841    0.0168    0.7864   -0.7847   -0.3940    0.3898
 -0.0084   -0.0421    0.0168    0.0841   -0.0168   -0.0841   -0.7847    0.7864    0.3898   -0.3940
 -0.0421   -0.0084    0.0841    0.0168   -0.0841   -0.0168   -0.3940    0.3898    0.7864   -0.7847
  0.0084    0.0421   -0.0168   -0.0841    0.0168    0.0841    0.3898   -0.3940   -0.7847    0.7864
```

Columns 21 through 24

```
-0.0084   0.0421   0.0210   -0.0042
-0.0421   0.0084   0.0042   -0.0210
-0.0210   0.0042   0.0084   -0.0421
```

```
-0.0042   0.0210   0.0421  -0.0084
 0.0421  -0.0084  -0.0042   0.0210
 0.0084  -0.0421  -0.0210   0.0042
 0.0042  -0.0210  -0.0421   0.0084
 0.0210  -0.0042  -0.0084   0.0421
-0.0421   0.0084   0.0421  -0.0084
-0.0084   0.0421   0.0084  -0.0421
 0.0421  -0.0084  -0.0421   0.0084
 0.0084  -0.0421  -0.0084   0.0421
-0.0841   0.0168   0.0841  -0.0168
-0.0168   0.0841   0.0168  -0.0841
 0.0841  -0.0168  -0.0841   0.0168
 0.0168  -0.0841  -0.0168   0.0841
 0.7864  -0.7847  -0.3940   0.3898
-0.7847   0.7864   0.3898  -0.3940
-0.3940   0.3898   0.7864  -0.7847
 0.3898  -0.3940  -0.7847   0.7864
 1.5761  -1.5677  -0.7872   0.7813
-1.5677   1.5761   0.7813  -0.7872
-0.7872   0.7813   1.5761  -1.5677
 0.7813  -0.7872  -1.5677   1.5761
```

### *Step 3 – Assembling the Global Stiffness Matrix:*

Since the structure has eight nodes, the size of the global stiffness matrix is $24 \times 24$. Therefore to obtain K we first set up a zero matrix of size $24 \times 24$ then make one call to the MATLAB function *LinearBrickAssemble* since we have only one element in the structure. In this case, it is clear that the element stiffness matrix is the same as the global stiffness matrix. The following are the MATLAB commands.

```
» K = zeros(24,24);

» K = LinearBrickAssemble(K,k1,1,2,3,4,5,6,7,8)

K =

   1.0e+008 *

   Columns 1 through 10

    0.4571   -0.4445   -0.2256    0.2218    0.2227   -0.2252   -0.1143    0.1080    0.0042    0.0021
   -0.4445    0.4571    0.2218   -0.2256   -0.2252    0.2227    0.1080   -0.1143    0.0021    0.0042
   -0.2256    0.2218    0.4571   -0.4445   -0.1143    0.1080    0.2227   -0.2252    0.0004    0.0008
    0.2218   -0.2256   -0.4445    0.4571    0.1080   -0.1143   -0.2252    0.2227    0.0008    0.0004
```

```
    0.2227   -0.2252   -0.1143    0.1080    0.4571   -0.4445   -0.2256    0.2218   -0.0008   -0.0004
   -0.2252    0.2227    0.1080   -0.1143   -0.4445    0.4571    0.2218   -0.2256   -0.0004   -0.0008
   -0.1143    0.1080    0.2227   -0.2252   -0.2256    0.2218    0.4571   -0.4445   -0.0021   -0.0042
    0.1080   -0.1143   -0.2252    0.2227    0.2218   -0.2256   -0.4445    0.4571   -0.0042   -0.0021
    0.0042    0.0021    0.0004    0.0008   -0.0008   -0.0004   -0.0021   -0.0042    0.4655   -0.4403
    0.0021    0.0042    0.0008    0.0004   -0.0004   -0.0008   -0.0042   -0.0021   -0.4403    0.4655
   -0.0004   -0.0008   -0.0042   -0.0021    0.0021    0.0042    0.0008    0.0004   -0.2319    0.2092
   -0.0008   -0.0004   -0.0021   -0.0042    0.0042    0.0021    0.0004    0.0008    0.2092   -0.2319
    0.0008    0.0004    0.0021    0.0042   -0.0042   -0.0021   -0.0004   -0.0008    0.2311   -0.2210
    0.0004    0.0008    0.0042    0.0021   -0.0021   -0.0042   -0.0008   -0.0004   -0.2210    0.2311
   -0.0021   -0.0042   -0.0008   -0.0004    0.0004    0.0008    0.0042    0.0021   -0.1164    0.1038
   -0.0042   -0.0021   -0.0004   -0.0008    0.0008    0.0004    0.0021    0.0042    0.1038   -0.1164
   -0.0421   -0.0084   -0.0042   -0.0210    0.0084    0.0421    0.0210    0.0042   -0.0841   -0.0168
    0.0084    0.0421    0.0210    0.0042   -0.0421   -0.0084   -0.0042   -0.0210    0.0168    0.0841
    0.0042    0.0210    0.0421    0.0084   -0.0210   -0.0042   -0.0084   -0.0421    0.0841    0.0168
   -0.0210   -0.0042   -0.0084   -0.0421    0.0042    0.0210    0.0421    0.0084   -0.0168   -0.0841
   -0.0084   -0.0421   -0.0210   -0.0042    0.0421    0.0084    0.0042    0.0210   -0.0421   -0.0084
    0.0421    0.0084    0.0042    0.0210   -0.0084   -0.0421   -0.0210   -0.0042    0.0084    0.0421
    0.0210    0.0042    0.0084    0.0421   -0.0042   -0.0210   -0.0421   -0.0084    0.0421    0.0084
   -0.0042   -0.0210   -0.0421   -0.0084    0.0210    0.0042    0.0084    0.0421   -0.0084   -0.0421
```

Columns 11 through 20

```
   -0.0004   -0.0008    0.0008    0.0004   -0.0021   -0.0042   -0.0421    0.0084    0.0042   -0.0210
   -0.0008   -0.0004    0.0004    0.0008   -0.0042   -0.0021   -0.0084    0.0421    0.0210   -0.0042
   -0.0042   -0.0021    0.0021    0.0042   -0.0008   -0.0004   -0.0042    0.0210    0.0421   -0.0084
   -0.0021   -0.0042    0.0042    0.0021   -0.0004   -0.0008   -0.0210    0.0042    0.0084   -0.0421
    0.0021    0.0042   -0.0042   -0.0021    0.0004    0.0008    0.0084   -0.0421   -0.0210    0.0042
    0.0042    0.0021   -0.0021   -0.0042    0.0008    0.0004    0.0421   -0.0084   -0.0042    0.0210
    0.0008    0.0004   -0.0004   -0.0008    0.0042    0.0021    0.0210   -0.0042   -0.0084    0.0421
    0.0004    0.0008   -0.0008   -0.0004    0.0021    0.0042    0.0042   -0.0210   -0.0421    0.0084
   -0.2319    0.2092    0.2311   -0.2210   -0.1164    0.1038   -0.0841    0.0168    0.0841   -0.0168
    0.2092   -0.2319   -0.2210    0.2311    0.1038   -0.1164   -0.0168    0.0841    0.0168   -0.0841
    0.4655   -0.4403   -0.1164    0.1038    0.2311   -0.2210    0.0841   -0.0168   -0.0841    0.0168
   -0.4403    0.4655    0.1038   -0.1164   -0.2210    0.2311    0.0168   -0.0841   -0.0168    0.0841
   -0.1164    0.1038    0.4655   -0.4403   -0.2319    0.2092   -0.0421    0.0084    0.0421   -0.0084
    0.1038   -0.1164   -0.4403    0.4655    0.2092   -0.2319   -0.0084    0.0421    0.0084   -0.0421
    0.2311   -0.2210   -0.2319    0.2092    0.4655   -0.4403    0.0421   -0.0084   -0.0421    0.0084
   -0.2210    0.2311    0.2092   -0.2319   -0.4403    0.4655    0.0084   -0.0421   -0.0084    0.0421
    0.0841    0.0168   -0.0421   -0.0084    0.0421    0.0084    1.5761   -1.5677   -0.7872    0.7813
   -0.0168   -0.0841    0.0084    0.0421   -0.0084   -0.0421   -1.5677    1.5761    0.7813   -0.7872
   -0.0841   -0.0168    0.0421    0.0084   -0.0421   -0.0084   -0.7872    0.7813    1.5761   -1.5677
    0.0168    0.0841   -0.0084   -0.0421    0.0084    0.0421    0.7813   -0.7872   -1.5677    1.5761
    0.0421    0.0084   -0.0841   -0.0168    0.0841    0.0168    0.7864   -0.7847   -0.3940    0.3898
   -0.0084   -0.0421    0.0168    0.0841   -0.0168   -0.0841   -0.7847    0.7864    0.3898   -0.3940
   -0.0421   -0.0084    0.0841    0.0168   -0.0841   -0.0168   -0.3940    0.3898    0.7864   -0.7847
    0.0084    0.0421   -0.0168   -0.0841    0.0168    0.0841    0.3898   -0.3940   -0.7847    0.7864
```

Columns 21 through 24

```
   -0.0084    0.0421    0.0210   -0.0042
   -0.0421    0.0084    0.0042   -0.0210
   -0.0210    0.0042    0.0084   -0.0421
   -0.0042    0.0210    0.0421   -0.0084
    0.0421   -0.0084   -0.0042    0.0210
    0.0084   -0.0421   -0.0210    0.0042
    0.0042   -0.0210   -0.0421    0.0084
    0.0210   -0.0042   -0.0084    0.0421
```

```
-0.0421    0.0084    0.0421   -0.0084
-0.0084    0.0421    0.0084   -0.0421
 0.0421   -0.0084   -0.0421    0.0084
 0.0084   -0.0421   -0.0084    0.0421
-0.0841    0.0168    0.0841   -0.0168
-0.0168    0.0841    0.0168   -0.0841
 0.0841   -0.0168   -0.0841    0.0168
 0.0168   -0.0841   -0.0168    0.0841
 0.7864   -0.7847   -0.3940    0.3898
-0.7847    0.7864    0.3898   -0.3940
-0.3940    0.3898    0.7864   -0.7847
 0.3898   -0.3940   -0.7847    0.7864
 1.5761   -1.5677   -0.7872    0.7813
-1.5677    1.5761    0.7813   -0.7872
-0.7872    0.7813    1.5761   -1.5677
 0.7813   -0.7872   -1.5677    1.5761
```

### *Step 4 – Applying the Boundary Conditions:*

The matrix (16.11) for this structure can be written using the global stiffness matrix obtained in the previous step. The boundary conditions for this problem are given as:

$$
\begin{aligned}
&U_{1x} = U_{1y} = U_{1z} = U_{2x} = U_{2y} = U_{2z} = 0 \\
&U_{3x} = U_{3y} = U_{3z} = U_{4x} = U_{4y} = U_{4z} = 0 \\
&F_{5x} = 4.6875\ ,\ F_{5y} = 0\ ,\ F_{5z} = 0 \\
&F_{6x} = 4.6875\ ,\ F_{6y} = 0\ ,\ F_{6z} = 0 \\
&F_{7x} = 4.6875\ ,\ F_{7y} = 0\ ,\ F_{7z} = 0 \\
&F_{8x} = 4.6875\ ,\ F_{8y} = 0\ ,\ F_{8z} = 0
\end{aligned}
\tag{16.14}
$$

We next insert the above conditions into the matrix equation for this structure (not shown here) and proceed to the solution step below.

### *Step 5 – Solving the Equations:*

Solving the resulting system of equations will be performed by partitioning (manually) and Gaussian elimination (with MATLAB). First we partition the resulting equation by extracting the submatrix in rows 13 to 24, and columns 13 to 24. Therefore we obtain the following equation noting that the numbers are shown to only two decimal places although MATLAB carries out the calculations using at least four decimal places.

$$10^8 \begin{bmatrix} 0.47 & -0.44 & -0.23 & 0.21 & -0.04 & 0.01 & 0.04 & -0.01 & -0.08 & 0.02 & 0.08 & 0-0.02 \\ -0.44 & 0.47 & 0.21 & -0.23 & -0.01 & 0.04 & 0.01 & -0.04 & -0.02 & 0.08 & 0.02 & -0.08 \\ -0.23 & 0.21 & 0.47 & -0.44 & 0.04 & -0.01 & -0.04 & 0.01 & 0.08 & -0.02 & -0.08 & 0.02 \\ 0.21 & -0.23 & -0.44 & 0.47 & 0.01 & -0.04 & -0.01 & 0.04 & 0.02 & -0.08 & -0.02 & 0.08 \\ -0.04 & -0.01 & 0.04 & 0.01 & 1.58 & -1.57 & -0.79 & 0.78 & 0.79 & -0.78 & -0.39 & 0.39 \\ 0.01 & 0.04 & -0.01 & -0.04 & -1.57 & 1.58 & 0.78 & -0.79 & -0.78 & 0.79 & 0.39 & -0.39 \\ 0.04 & 0.01 & -0.04 & -0.01 & -0.79 & 0.78 & 1.58 & -1.57 & -0.39 & 0.39 & 0.79 & -0.78 \\ -0.01 & -0.04 & 0.01 & 0.04 & 0.78 & -0.79 & -1.57 & 1.58 & 0.39 & -0.39 & -0.78 & 0.79 \\ -0.08 & -0.02 & 0.08 & 0.02 & 0.79 & -0.78 & -0.39 & 0.39 & 1.58 & -1.57 & -0.79 & 0.78 \\ 0.02 & 0.08 & -0.02 & -0.08 & -0.78 & 0.79 & 0.39 & -0.39 & -1.57 & 1.58 & 0.78 & -0.79 \\ 0.08 & 0.02 & -0.08 & -0.02 & -0.39 & 0.39 & 0.79 & -0.78 & -0.79 & 0.78 & 1.58 & -1.57 \\ -0.02 & -0.08 & 0.02 & 0.08 & 0.39 & -0.39 & -0.78 & 0.79 & 0.78 & -0.79 & -1.57 & 1.58 \end{bmatrix}$$

$$\begin{Bmatrix} U_{5x} \\ U_{5y} \\ U_{5z} \\ U_{6x} \\ U_{6y} \\ U_{6z} \\ U_{7x} \\ U_{7y} \\ U_{7z} \\ U_{8x} \\ U_{8y} \\ U_{8z} \end{Bmatrix} = \begin{Bmatrix} 4.6875 \\ 0 \\ 0 \\ 4.6875 \\ 0 \\ 0 \\ 4.6875 \\ 0 \\ 0 \\ 4.6875 \\ 0 \\ 0 \end{Bmatrix} \tag{16.15}$$

The solution of the above system is obtained using MATLAB as follows. Note that the backslash operator "\" is used for Gaussian elimination.

```
» k = K(13:24,13:24)

k =

    1.0e+008 *

    Columns 1 through 10

    0.4655   -0.4403   -0.2319    0.2092   -0.0421    0.0084    0.0421   -0.0084   -0.0841    0.0168
   -0.4403    0.4655    0.2092   -0.2319   -0.0084    0.0421    0.0084   -0.0421   -0.0168    0.0841
   -0.2319    0.2092    0.4655   -0.4403    0.0421   -0.0084   -0.0421    0.0084    0.0841   -0.0168
    0.2092   -0.2319   -0.4403    0.4655    0.0084   -0.0421   -0.0084    0.0421    0.0168   -0.0841
   -0.0421   -0.0084    0.0421    0.0084    1.5761   -1.5677   -0.7872    0.7813    0.7864   -0.7847
    0.0084    0.0421   -0.0084   -0.0421   -1.5677    1.5761    0.7813   -0.7872   -0.7847    0.7864
    0.0421    0.0084   -0.0421   -0.0084   -0.7872    0.7813    1.5761   -1.5677   -0.3940    0.3898
   -0.0084   -0.0421    0.0084    0.0421    0.7813   -0.7872   -1.5677    1.5761    0.3898   -0.3940
   -0.0841   -0.0168    0.0841    0.0168    0.7864   -0.7847   -0.3940    0.3898    1.5761   -1.5677
    0.0168    0.0841   -0.0168   -0.0841   -0.7847    0.7864    0.3898   -0.3940   -1.5677    1.5761
    0.0841    0.0168   -0.0841   -0.0168   -0.3940    0.3898    0.7864   -0.7847   -0.7872    0.7813
   -0.0168   -0.0841    0.0168    0.0841    0.3898   -0.3940   -0.7847    0.7864    0.7813   -0.7872
```

```
Columns 11 through 12

    0.0841   -0.0168
    0.0168   -0.0841
   -0.0841    0.0168
   -0.0168    0.0841
   -0.3940    0.3898
    0.3898   -0.3940
    0.7864   -0.7847
   -0.7847    0.7864
   -0.7872    0.7813
    0.7813   -0.7872
    1.5761   -1.5677
   -1.5677    1.5761

» f = [4.6875 ; 0 ; 0 ; 4.6875 ; 0 ; 0 ; 4.6875 ; 0 ; 0 ;
      4.6875 ; 0 ; 0]

f =

    4.6875
         0
         0
    4.6875
         0
         0
    4.6875
         0
         0
    4.6875
         0
         0

» u = k\f

Warning: Matrix is close to singular or badly scaled.
         Results may be inaccurate.
         RCOND = 7.687107e-018.

u =

   1.0e+008 *

    0.0000
    0.0000
```

```
0.0000
0.0000
2.0972
2.0972
2.0972
2.0972
2.0972
2.0972
2.0972
2.0972
```

It is clear that the stiffness matrix is singular or badly scaled. This is due to using one element only in this example. This is also due to the element having a bad aspect ratio. Therefore, the results obtained above for the displacements are not reliable, even totally wrong. It is anticipated that using more elements will result in reliable answers to the displacements. This will be done by the reader in Problem 16.1.

### *Problems:*

### *Problem 16.1:*

Consider the thin plate problem solved in Example 16.1. Solve the problem again using two linear brick elements instead of one element as shown in Fig. 16.5. Compare your answers for the displacements at nodes 9, 10, 11, and 12 with the answers obtained in the example. Compare also your answers with those obtained in the related examples and problems in Chaps. 11 to 15.

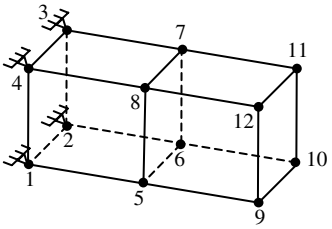


**Fig. 16.5.** Discretization of Thin Plate into Two Linear Bricks

*Hint*: Table 16.2 shows the element connectivity for this problem.

**Table 16.2.** Element connectivity for problem 16.2

| Element Number | Node $i$ | Node $j$ | Node $m$ | Node $n$ | Node $p$ | Node $q$ | Node $r$ | Node $s$ |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |