CEE 576: Nonlinear Finite Elements - Fall 2016
Homework 2

Bhavesh Shrimali (NetID: bshrima2)

September 23, 2016

Sol$^{\text{n}}$ 1:

The general nonlinear finite element formulation for 3-D Elasticity problem can be described as follows:

$$a\left(\mathbf{w},\mathbf{u};\mathbf{u},\mathbf{x}\right) = \left(\mathbf{w},\mathbf{f}\right)_{\Omega} + \left(\mathbf{w},\mathbf{h}\right)_{\Gamma_h} \tag{1}$$

where

$$a\left(\mathbf{w},\mathbf{u};\mathbf{u},\mathbf{x}\right) = \int_{\Omega} w_{i,j}\sigma_{ij}\left(\epsilon_{ij},\mathbf{x}\right)\ d\Omega\ ; \qquad \left(\mathbf{w},\mathbf{f}\right)_{\Omega} = \int_{\Omega} w_i f_i\ d\Omega\ ; \qquad \left(\mathbf{w},\mathbf{h}\right)_{\Gamma_h} = \int_{\Gamma_h} w_i h_i\ d\Gamma$$

The corresponding 1-D version of Equation (1) be stated as follows:

$$a(w,u;u,x) = (w,f)_{\Omega} + (w,h)_{\Gamma}$$

The Galerkin form is similar to the above equation, except that the functions w^h, u^h and f^h are piece-wise continuous over each element. Moreover f^h can be thought of as the mapping of the forcing function of the finite element space of functions.

$$a(w^h,u^h;u^h,x) = \int_{\Omega} w^h_{,x}\sigma^h\left(u^h_{,x}\right); \qquad (w^h,f^h)_{\Omega} = \int_{\Omega} w^h f^h\ d\Omega\ ; \quad (w^h,h) = \int_{\Gamma_h} w^h h\ d\Gamma \tag{2}$$

(a) Strong Form (S):

For the nonlinear 1D elasticity problem, given external forcing function $f(x)$ and material parameters $\{\Omega = (0,1)\}$, find $u(x) : \Omega \to \mathbb{R}$ such that $u(x) \in \mathcal{U}$ and

$$\begin{aligned} \sigma_{,x}\left(u_{,x}\,;x\right) + f(x) = 0 \\ u(1) = g; \qquad \sigma(0) = h \end{aligned} \tag{3}$$

where

$$\mathcal{U} = \left\{u(x) \mid u(x) \in C^2\left(0,1\right), u(1) = g\right\}$$

The dependence on x (inhomogeneity) appears in the first term in the expression, which ultimately appears in the Consistent-tangent tensor. The forcing function $f(x)$ may (or may not) potentially depend on the spatial variable x.

(b) Weak Form (W):

We multiply Equation$(3)_1$ by a test function $w(x)$ and integrate by parts to get

$$\int_{\Omega} w_{,x}(x)\sigma\left(u_{,x};x\right)\ d\Omega = w(x)\sigma\left(u_{,x};x\right)|_{\Gamma_h} + \int_{\Omega} w(x)f(x)\ d\Omega$$

Thus the Weak form (W) can be stated as follows:
Given $f(x)$ and other material parameters $\{\Omega = (0,1)\}$, find $u(x) : \Omega \to \mathbb{R}$ such that $u(x) \in \mathcal{U}_0(\Omega)$

$$\int_{\Omega} w_{,x}(x)\sigma\left(u_{,x};x\right)\ d\Omega = w(x)\sigma\left(u_{,x};x\right)|_{\Gamma_h} + \int_{\Omega} w(x)f(x)\ d\Omega \qquad \forall w(x) \in \mathcal{V}(\Omega) \tag{4}$$

$$\begin{aligned}\mathcal{U}_0(\Omega) &= \left\{u\ |u(x) \in \mathbf{H}^1(\Omega),\ u(x)_{|\Gamma_g} = g\right\}\\ \mathcal{V}(\Omega) &= \left\{w\ |w(x) \in \mathbf{H}^1(\Omega),\ w(x)_{|\Gamma_g} = 0\right\}\end{aligned}$$

(c) Galerkin-Form (G):

We now introduce the concept of discretization (mesh). The Galerkin-form (G) is, find $u^h(x) : \Omega_e \to \mathbb{R}$ such that $u^h(x) \in \mathcal{U}_0^h(\Omega_e)$

$$\sum_{e=1}^{n_{el}} \int_{\Omega_e} w^h{}_{,x}(x)\ \sigma\left(u^h{}_{,x};x\right)\ d\Omega = w^h(x)\sigma\left(u^h{}_{,x};x\right)|_{\Gamma_h} + \sum_{e=1}^{n_{el}} \int_{\Omega_e} w^h(x)f^h(x)\ d\Omega \qquad \forall w^h(x) \in \mathcal{V}^h(\Omega_e) \tag{5}$$

$$\begin{aligned}\mathcal{U}_0^h(\Omega_e) &= \left\{u^h(x)\ |u^h(x) \in \mathbf{H}^1(\Omega_e),\ u^h(x)_{|\Gamma_g} = g\right\}\\ \mathcal{V}^h(\Omega_e) &= \left\{w^h(x)\ |w^h(x) \in \mathbf{H}^1(\Omega_e),\ w^h(x)_{|\Gamma_g} = 0\right\}\end{aligned}$$

Choice of Shape Functions:

For the first part of the problem we consider linear shape functions to compute the external force vector, internal force vector, and consistent tangent i.e. for $-1 \leq \xi \leq 1$

$$N(\xi) = \frac{1}{2}\begin{Bmatrix}1-\xi\\ 1+\xi\end{Bmatrix}$$

For the Part.B and Part.C of the problem, separate reference to quadratic shape functions is made. We use isoparametric elements throughout, and hence

$$x = \sum_{i=1}^{n_{en}} N_i^e(\xi)\ x_i$$

(d) Nonlinear Matrix Problem:

1. External Force Vector:

The external load vector, for an element is calculated as follows:

$$\begin{aligned} \mathbf{f}_a^e &= \int_{\Omega_e} N_a(\xi)\ \hat{f}(x)\ d\Omega \\ &= \int_{\Omega} N_a(\xi) N_b(\xi) f_b\ d\Omega \end{aligned} \tag{6}$$

where f_b represent the nodal values of $f^h(x)$. It can be interpreted as the projection of the actual forcing function on the finite element space of functions used to discretize the field. We use one point Gauss-Quadrature to evaluate the integral

$$\begin{aligned} \mathbf{f}_a^e &= \int_{\Omega_e} \mathbf{N}^T \mathbf{N}\ \mathbf{f}_{\text{nodal}}\ d\Omega \\ &= \int_{\Omega_e} \frac{1}{4} \begin{bmatrix} (1-\xi)^2 & (1-\xi^2) \\ (1-\xi^2) & (1+\xi)^2 \end{bmatrix} \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} d\Omega \\ &= \int_{-1}^{1} \frac{1}{4} \begin{bmatrix} (1-\xi)^2 & (1-\xi^2) \\ (1-\xi^2) & (1+\xi)^2 \end{bmatrix} \begin{Bmatrix} f_1 \\ f_2 \end{Bmatrix} \frac{1}{2}\ d\xi \\ \\ &= \frac{1}{4} \begin{Bmatrix} f_1 + f_2 \\ f_1 + f_2 \end{Bmatrix} + \begin{Bmatrix} h \\ 0 \end{Bmatrix} \end{aligned}$$

2. Internal Force Vector:

The internal force vector for an element is calculated as follows:

$$\begin{aligned} \mathbf{n}_a^e &= \int_{\Omega_e} N_{a,x}\, \sigma\left(\epsilon^h; x\right)\ d\Omega \\ \text{where,}\quad \epsilon^h &= u^h{}_{,x}(x) = \sum_{e=1}^{n_{en}} N_{e,x}(\xi)\ d_e \\ \mathbf{n}_a^e &= \int_{\xi=-1}^{1} \frac{2}{h^e} \frac{1}{2} \begin{Bmatrix} -1 \\ 1 \end{Bmatrix}\ \sigma\left(\frac{1}{h^e}\left(d_2^e - d_1^e\right)\ ; x(\xi)\right) \frac{h^e}{2}\ d\xi \end{aligned}$$

Using one point Quadrature ($W = 2$ and $\xi = 0$):

$$\begin{aligned} \mathbf{n}_a^e &= 2 \cdot \frac{1}{2} \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \sigma\left(\frac{1}{h^e}\left(d_2^e - d_1^e\right)\ ; \frac{x_1^e + x_2^e}{2}\right) \\ &= \begin{Bmatrix} -1 \\ 1 \end{Bmatrix} \sigma\left(d_2^e - d_1^e\ ; \frac{x_1^e + x_2^e}{2}\right) \end{aligned}$$

The Matrix Problem is just $\mathbb{A}_{e=1}^{n_{el}}\ \{\mathbf{n}_a^e\} = \mathbb{A}_{e=1}^{n_{el}} \mathbf{f}_a^e$, where the operator, $\mathbb{A}_{e=1}^{n_{el}}$, is the assembly operator over all elements in the mesh. Thus we get

$$\mathbf{f}_a^e = \int_{\Omega} \mathbf{N}_a \mathbf{N}_b f_b^{\text{nodal}}\ d\Omega + \mathbf{N}_{a_{x=0}} \cdot h$$

(e) Consistent Tangent Tensor:

The consistent tangent tensor can be obtained by taking the variational derivative (in this case the Taylor-Expansion) of the internal force vector:

$$\begin{aligned}\mathbb{D}\mathbf{n}_a^e &= \frac{\partial \mathbf{n}_a^e}{\partial d_b} \\ &= \frac{\partial}{\partial d_b}\int_{\Omega_e} N_{a},_{x}\, \sigma\left(\epsilon^h; x\right)\ d\Omega\end{aligned}$$

Applying the chain-rule of differentiation, with ($h^e = 1$):

$$\begin{aligned}&\int_{\Omega_e} N_{a},_{x}\left(\xi\right) \frac{\partial}{\partial \epsilon^h}\sigma\left(\epsilon^h; x\right)\frac{\partial \epsilon^h}{\partial d_b}\ d\Omega \\ &= \int_{\Omega_e} N_{a},_{x}\left(\xi\right)\,\mathbb{C}\left(\epsilon^h; x\right) N_{e},_{x}\left(\xi\right)\,\delta_{eb}\ d\Omega \\ &= \int_{\Omega_e} N_{a},_{x}\left(\xi\right)\,\mathbb{C}\left(\epsilon^h; x\right) N_{b},_{x}\left(\xi\right)\ d\Omega \\ &= \int_{-1}^{1} \frac{2}{h^e} N_{a},_{\xi}\left(\xi\right)\,\mathbb{C}\left(\epsilon^h(\xi); x(\xi)\right) N_{b},_{\xi}\left(\xi\right)\ d\xi \\ &= 2\cdot 2\frac{1}{4}\begin{Bmatrix}-1\\1\end{Bmatrix}\mathbb{C}\left(\frac{d_2^e - d_1^e}{h^e}; \frac{x_2^e + x_1^e}{2}\right)\begin{Bmatrix}-1 & 1\end{Bmatrix} \\ &= \begin{bmatrix}1 & -1\\ -1 & 1\end{bmatrix}\mathbb{C}\left(d_2^e - d_1^e; \frac{x_2^e + x_1^e}{2}\right)\end{aligned} \tag{7}$$

(f) Internal Force:

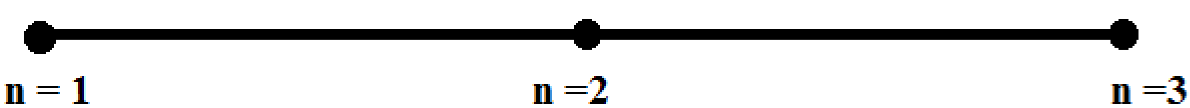
For this part, and the next, calculations are presented with respect to quadratic shape functions and 2-point numerical quadrature.

Choice of Shape Functions:

1-D Quadratic Shape Functions, listed for the given element as shown in the figure:

$$N\left(\xi\right) = \frac{1}{2}\begin{Bmatrix}\xi(\xi - 1)\\ 2(1+\xi)(1-\xi)\\ \xi(\xi+1)\end{Bmatrix}$$

Assumption: The internal node is equidistant from the other two nodes.



Calculations:

$$\mathbf{n}^e_a = \int_{\Omega_e} N_{a},_{x}\, \sigma\left(\epsilon^h; x\right)\ d\Omega \tag{8}$$

Again using isoparametric elements:

$$x = \sum_{i=1}^{n_{en}} N^e_i(\xi)\ x_i$$

Thus

$$J = \frac{\mathrm{d}x}{d\xi} = \sum_{i=1}^{n_{en}} \frac{\mathrm{d}N^e_i}{d\xi}(\xi)\ x^e_i = \frac{(2\xi-1)}{2}x^e_1 - 2\xi x^e_2 + \frac{(2\xi+1)}{2}x^e_3 = \frac{1}{2}$$

$$\mathbf{n}^e_a = \int_{-1}^{1} \frac{1}{2}\begin{bmatrix}2\xi-1\\ -4\xi\\ 2\xi+1\end{bmatrix} \sigma\left(\epsilon^h(\xi); x(\xi)\right)\ d\xi$$

Now using 2-point Gauss-Quadrature to integrate the above polynomial, we get:

$$\begin{aligned}
&\frac{1}{2}\begin{bmatrix}2\xi-1\\ -4\xi\\ 2\xi+1\end{bmatrix} \sigma\left(\epsilon^h(\xi); x(\xi)\right)\Big|_{\xi=-\frac{1}{\sqrt{3}}} + \frac{1}{2}\begin{bmatrix}2\xi-1\\ -4\xi\\ 2\xi+1\end{bmatrix} \sigma\left(\epsilon^h(\xi); x(\xi)\right)\Big|_{\xi=\frac{1}{\sqrt{3}}} \\
&= \begin{bmatrix}-1.07735\\ 1.155\\ -0.07735\end{bmatrix} \sigma\left(\epsilon^h(\frac{-1}{\sqrt{3}}); x(\frac{-1}{\sqrt{3}})\right) + \begin{bmatrix}0.07735\\ -1.155\\ 1.07735\end{bmatrix} \sigma\left(\epsilon^h(\frac{1}{\sqrt{3}}); x(\frac{1}{\sqrt{3}})\right)
\end{aligned} \tag{9}$$

where,

$$\epsilon^h\left(\frac{-1}{\sqrt{3}}\right) = -2.1547d^e_1 + 2.311d^e_2 - 0.1547d^e_3\ ; \qquad x\left(\frac{-1}{\sqrt{3}}\right) = 0.455x^e_1 + 0.667x^e_2 - 0.122x^e_3 = 0.2115$$

$$\epsilon^h\left(\frac{1}{\sqrt{3}}\right) = 0.1547d^e_1 - 2.311d^e_2 + 2.1547d^e_3\ ; \qquad x\left(\frac{1}{\sqrt{3}}\right) = -0.122x^e_1 + 0.667x^e_2 + 0.455x^e_3 = 0.7885$$

(g) Consistent Tangent Tensor:

Using the calculation carried out in (7) we have

$$\begin{aligned}
\mathbb{D}\mathbf{n}^e_a &= \int_{-1}^{1} \frac{2}{h^e} N_{a},_{\xi}(\xi)\ \mathbb{C}\left(\epsilon^h(\xi); x(\xi)\right) N_{b},_{\xi}(\xi)\ d\xi \\
&= \int_{-1}^{1} \frac{2}{h^e}\frac{1}{4}\begin{bmatrix}2\xi-1\\ -4\xi\\ 2\xi+1\end{bmatrix} \mathbb{C}\left(\epsilon^h(\xi); x(\xi)\right) \begin{bmatrix}2\xi-1 & -4\xi & 2\xi+1\end{bmatrix}\ d\xi \\
&= \int_{-1}^{1} \frac{1}{2}\begin{bmatrix}(2\xi-1)^2 & -4\xi(2\xi-1) & 4\xi^2-1\\ -4\xi(2\xi-1) & 16\xi^2 & -4\xi(2\xi+1)\\ 4\xi^2-1 & -4\xi(2\xi+1) & (2\xi+1)^2\end{bmatrix} \mathbb{C}\left(\epsilon^h(\xi); x(\xi)\right)\ d\xi
\end{aligned} \tag{10}$$

Using 2-point Gauss-Quadrature to calculate the above integral we get,

$$\frac{1}{2}\left[\begin{bmatrix} 4.642 & -4.976 & 0.333 \\ -4.976 & 5.33 & -0.357 \\ 0.333 & -0.357 & 0.0239 \end{bmatrix} \mathbb{C}\left(\epsilon^h(\frac{-1}{\sqrt{3}}); x(\frac{-1}{\sqrt{3}})\right) + \begin{bmatrix} 0.0239 & -0.357 & 0.333 \\ -0.357 & 5.33 & -4.976 \\ 0.333 & -4.976 & 4.642 \end{bmatrix} \mathbb{C}\left(\epsilon^h(\frac{1}{\sqrt{3}}); x(\frac{1}{\sqrt{3}})\right)\right] \tag{11}$$

Thus the Consistent Tangent Tensor, evaluated using the 2-point Gauss-Quadrature would be as follows:

$$\begin{bmatrix} 2.321 & -2.488 & 0.167 \\ -2.488 & 2.667 & -0.179 \\ 0.167 & -0.179 & 0.012 \end{bmatrix} \mathbb{C}\left(\epsilon^h(\frac{-1}{\sqrt{3}}); x(\frac{-1}{\sqrt{3}})\right) + \begin{bmatrix} 0.012 & -0.179 & 0.167 \\ -0.179 & 2.667 & -2.488 \\ 0.167 & -2.488 & 2.321 \end{bmatrix} \mathbb{C}\left(\epsilon^h(\frac{1}{\sqrt{3}}); x(\frac{1}{\sqrt{3}})\right) \tag{12}$$

where,

$$\epsilon^h\left(\frac{-1}{\sqrt{3}}\right) = -2.1547d_1^e + 2.311d_2^e - 0.1547d_3^e\ ; \qquad x\left(\frac{-1}{\sqrt{3}}\right) = 0.455x_1^e + 0.667x_2^e - 0.122x_3^e = 0.2115$$

$$\epsilon^h\left(\frac{1}{\sqrt{3}}\right) = 0.1547d_1^e - 2.311d_2^e + 2.1547d_3^e\ ; \qquad x\left(\frac{1}{\sqrt{3}}\right) = -0.122x_1^e + 0.667x_2^e + 0.455x_3^e = 0.7885$$

Part C.:

No, 1-point Gauss-Quadrature cannot integrate this problem exactly.

- A general m-point Quadrature can exactly integrate a polynomial of order $2m-1$, thus 1-point Quadrature can only integrate linear polynomials.
- For the consistent tangent tensor there are quadratic polynomials to be integrated (at-least considering $\mathbb{C}$ remains constant) or higher.
- Thus, we would need higher order quadrature, potentially 2 (in case of constant $\mathbb{C}$) or even more depending on the dependance of $\mathbb{C}$ on x, to exactly integrate this problem.

Some remarks about internal force vector:

- For the internal force vector, we have linear polynomials to be integrated at least,(if σ is constant), but in case of non-homogeneous materials $\sigma(\epsilon; x)$ and hence one point quadrature would not be able to exactly integrate the problem.

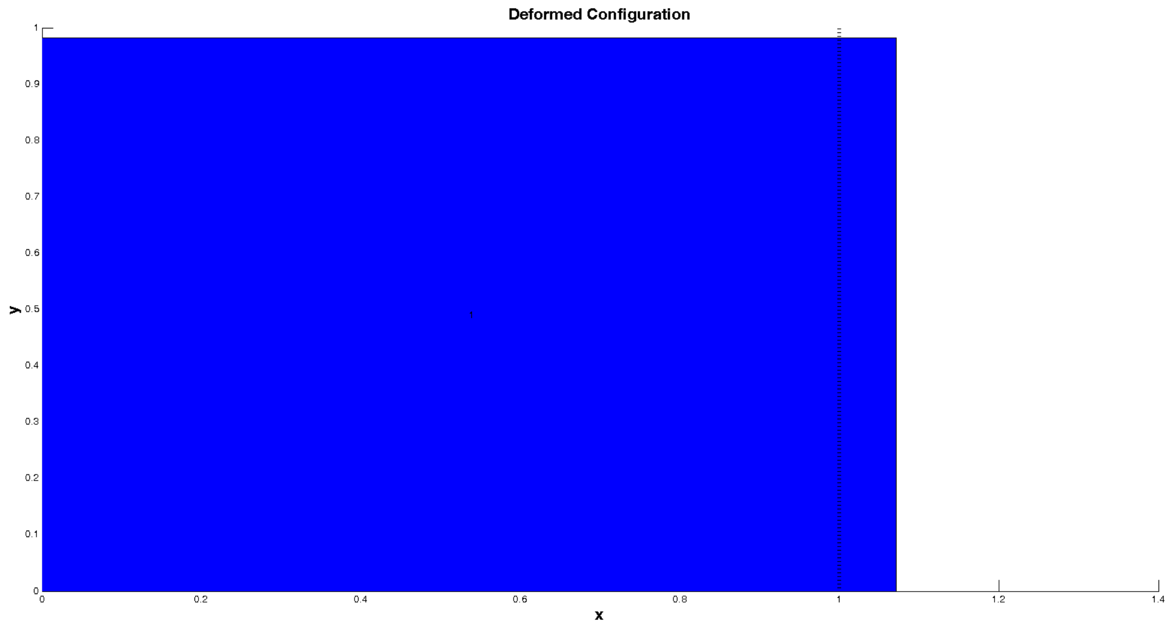
Some remarks about external force vector:

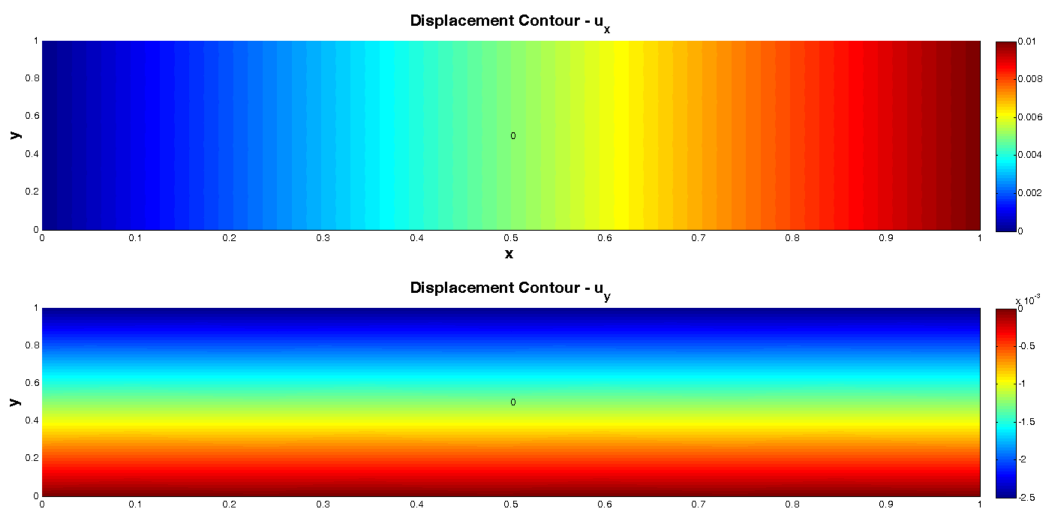
- In case of the external force vector we will have the product of shape functions N_iN_j and the nodal values of the forcing function.
- This would lead to a biquadratic polynomial and hence, even 2-point Gauss-Quadrature would not be sufficient to exactly integrate the external load vector.

Sol^n 2:

The given problem is solved using the provided Matlab-Code:

(a) Results: Plots





(b) Stress-Strain values

The values for stresses and strains obtained at the integration points are described below:

Table 1: Stress at Integration points

Integration Point \ Stress	σ_{xx}	σ_{yy}	σ_{xy}
1	1	4.547E-13	0
2	1	4.547E-13	0
3	1	4.547E-13	0
4	1	4.547E-13	0

Table 2: Strain at Integration Points

Integration Point \ Strain	ϵ_{xx}	ϵ_{yy}	ϵ_{xy}
1	0.01	-0.0025	0
2	0.01	-0.0025	0
3	0.01	-0.0025	0
4	0.01	-0.0025	0

Table 3: Nodal Displacements

Node	Ux	Uy
1	0	0
2	0.01	0
3	0.01	-0.0025
4	0	-0.0025

Matlab Code:

```
% Input File: One Quadrilateral Element Under Axial Load
%
% Copyright (C) Arif Masud and Tim Truster
%
% This input file should be run prior to executing the FEA_Program routine.
%
% Format of required input:
%
%   numnp:          = number of nodes in the mesh (length(NodeTable))
%
%   numel:          = number of elements in the mesh
%
%   nen:            = maximum number of nodes per element (4)
%
%   PSPS:           = flag for plane stress ('s') or plane strain ('n')
%
%   NodeTable:      = table of mesh nodal coordinates defining the
```

```
%                          geometry of the mesh; format of the table is as
%                          follows:
%                              Nodes   |               x-coord  y-coord
%                              n1      |   NodeTable = [x1      y1
%                              n2      |                x2      y2
%                              ...     |                ..      ..
%                              nnumnp  |                xnumnp ynumnp];
%
%   ix:                 = table of mesh connectivity information, specifying
%                          how nodes are attached to elements and how materials
%                          are assigned to elements; entries in the first nen
%                          columns correspond to the rows of NodeTable
%                          representing the nodes attached to element e;
%                          entries in the last nen+1 column are rows from MateT
%                          signifying the material properties assigned to
%                          element e; format of the table is as follows:
%                              Elements   |          n1     n2     n3     n4    mat
%                              e1         |   ix = [e1n1   e1n2   e1n3   e1n4  e1mat
%                              e2         |         e2n1   e2n2   e2n3   e2n4  e2mat
%                              ...        |          ..     ..     ..     ..    ..
%                              enumel     |         values for element numel    ];
%
%   MateT:              = table of mesh material properties for each distinct
%                          set of material properties; these sets are
%                          referenced by element e by setting the value of
%                          ix(e,nen+1) to the row number of the desired
%                          material set; format of the table is as follows:
%                              Materials   |            E    v    t
%                              mat1        |   MateT = [E1   v1   t1
%                              mat2        |            E2   v2   t2
%                              ...         |            ..   ..   ..];
%
%   BCLIndex:           = list of the number of boundary conditions and loads
%                          applied to the mesh; first entry is the number of
%                          prescribed displacements at nodes; second entry is
%                          the number of nodal forces
%
%   NodeBC:             = table of prescribed nodal displacement boundary
%                          conditions; it contains lists of nodes, the
%                          direction of the displacement prescribed (x=1, y=2),
%                          and the value of the displacement (set 0 for fixed
%                          boundary); the length of the table must match the
%                          entry in BCLIndex(1), otherwise an error will result
%                          if too few conditions are given or extra BCs will be
%                          ignored in the model input module;  format of the
%                          table is as follows:
%                              BCs   |              nodes  direction  value
%                              bc1   |   NodeBC = [bc1n    bc1dir    bc1u
%                              bc2   |             bc2n    bc2dir    bc2u
%                              ...   |              ..       ..        .. ];
```

```
%
%   NodeLoad:        = table of prescribed nodal forces; it contains lists
%                      of nodes, the direction of the force prescribed
%                      (x=1, y=2), and the value of the force; the length
%                      of the table must match the entry in BCLIndex(2),
%                      otherwise an error will result if too few conditions
%                      are given or extra loads will be ignored in the
%                      model input module; format of the table is as
%                      follows:
%                         Loads  |               nodes  direction  value
%                         P1     |   NodeLoad = [ P1n    P1dir     P1P
%                         P2     |                P2n    P2dir     P2P
%                         ...    |                ..     ..        .. ];
%
%
%           4 ---------------- 3       2
%           |                  |       | \
%           |                  |       |  \
%           |                  |       |   \
%           |                  |       |    \
%           |                  |       |     \
%           |                  |       |      \
%           1 ---------------- 2       3-------1
%

% Arbitrary data for assistance in defining the mesh

% Mesh Nodal Coordinates
NodeTable = [0  0
             1  0
             1  1
             0  1];
numnp = length(NodeTable);

% Mesh Element Connectivities
ix = [1  2  3  4  1];

nen = 4;
numel = 1;

% Mesh Boundary Conditions and Loads
BCLIndex = [6  0]';
NodeBC = [1  1  0
          1  2  0
          2  1  0.01
          2  2  0
          3  1  0.01
          4  1  0
          ];
NodeLoad = 0;
```

```
% Mesh Material Properties
young = 100;
pois = .25;
thick = 1;
PSPS = 's';
MateT = [young pois thick];
FEA_Program

function [strain,stress] = CompStrainStress_Elem_Cee570(xl,ul,mateprop,nel,
    ndf,PSPS)
%
% Subroutine to compute strain and stress for linear
% 2-dimensional elasticity element. Element currently supports bilinear
% quadrilateral elements with the following node and shape function
% labelling scheme:
%
%  (-1, 1)  4 --------------- 3 ( 1, 1)
%           |        s        |
%           |        ^        |
%           |        |        |
%           |        .-> r    |
%           |                 |
%           |                 |
%  (-1,-1)  1 --------------- 2 ( 1,-1)
%
% Element local coordinates (r,s) are defined by a coordinate axis with the
% origin at the center of the element; the corners of the element have
% local coordinate values as shown in the figure.
%
% Definitions for input:
%
%   xl:              = local array containing (x,y) coordinates of nodes
%                      forming the element; format is as follows:
%                          Nodes    |        n1  n2  n3  n4
%                          x-coord  |  xl = [x1  x2  x3  x4
%                          y-coord  |        y1  y2  y3  y4];
%
%   mateprop:        = vector of material properties:
%                          mateprop = [E v t];
%                                   = [(Young's Modulus) (Poisson's Ratio)
%                                     (thickness)];
%
%   nel:             = number of nodes on current element (4)
%
%   ndf:             = max number of DOF per node (2)
%
%   ndm:             = space dimension of mesh (2)
%
%   PSPS:            = flag for plane stress ('s') or plane strain ('n')
```

```
%
% Definitions for output:
%
%   strain:          = strain array containing strain components
%                      at integration points:
%                                    xx    yy    xy
%                      int1   strain[ .     .     .
%                      int2           .     .     .
%                      int3           .     .     .
%                      int4           .     .     .   ];
%
%   stress:          = stress array containing stress components
%                      at integration points:
%                                    xx    yy    xy
%                      int1   stress[ .     .     .
%                      int2           .     .     .
%                      int3           .     .     .
%                      int4           .     .     .   ];
%
% Definitions of local constants:
%
%   nst:             = size of element arrays (ndf*nel)
%
%

% Set Material Properties

ElemE = mateprop(1);
Elemv = mateprop(2);
thick = mateprop(3);

if PSPS == 's' %Plane Stress

  Dmat = ElemE/(1-Elemv^2)*[1        Elemv   0
                            Elemv    1       0
                            0        0       (1-Elemv)/2];

else %Plane Strain

%     Dmat =

end

% Load Guass Integration Points

if nel == 3
    lint = 4;
else
    lint = 4;
end
```

```
% Initialize Matrix and Vector

nst = nel*ndf;
ul_elem = reshape(ul,ndf*nel,1);
strain = zeros(lint,3);
stress = zeros(lint,3);
strain_temp = zeros(3,1);
stress_temp = zeros(3,1);

% Loop over integration points
for l = 1:lint

        if nel == 3
            [Wgt,r,s] = intpntt(l,lint,0);
        else
            [Wgt,r,s] = intpntq(l,lint,0);
        end

        % Evaluate local basis functions at integration point
        shp = shpl_2d(r,s,nel);

        % Evaluate first derivatives of basis functions at int. point
        [Qxy, Jdet] = shpg_2d(shp,xl,nel);

        % Form B matrix
       if nel == 3
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0
                0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3)
                Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3)];
        else
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0        Qxy
            (1,4) 0
                0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3) 0
                          Qxy(2,4)
                Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3) Qxy
                    (2,4) Qxy(1,4)];

        end

        % Compute strain
        strain_temp = Bmat*ul_elem;

        strain(l,1) = strain_temp(1,1);
        strain(l,2) = strain_temp(2,1);
        strain(l,3) = strain_temp(3,1);

        % Compute stress
        stress_temp = Dmat*strain_temp;
```

```
        stress(l,1) = stress_temp(1,1);
        stress(l,2) = stress_temp(2,1);
        stress(l,3) = stress_temp(3,1);

end

function [ElemK, ElemF] = Elast2d_Elem(xl,mateprop,nel,ndf,ndm,PSPS)
%
% Copyright (C) Arif Masud and Tim Truster
%
% Subroutine to compute stiffness matrix and force vector for linear
% 2-dimensional elasticity element. Element currently supports bilinear
% quadrilateral elements with the following node and shape function
% labelling scheme:
%
%  (-1, 1)   4 ------------------ 3 ( 1, 1)
%            |        s           |
%            |        ^           |
%            |        |           |
%            |        .-> r       |
%            |                    |
%            |                    |
%  (-1,-1)   1 ------------------ 2 ( 1,-1)
%
% Element local coordinates (r,s) are defined by a coordinate axis with the
% origin at the center of the element; the corners of the element have
% local coordinate values as shown in the figure.
%
% Definitions for input:
%
%   xl:              = local array containing (x,y) coordinates of nodes
%                      forming the element; format is as follows:
%                          Nodes    |         n1  n2  n3  n4
%                          x-coord  |   xl = [x1  x2  x3  x4
%                          y-coord  |         y1  y2  y3  y4];
%
%   mateprop:        = vector of material properties:
%                          mateprop = [E v t];
%                                   = [(Young's Modulus) (Poisson's Ratio)
%                                      (thickness)];
%
%   nel:             = number of nodes on current element (4)
%
%   ndf:             = max number of DOF per node (2)
%
%   ndm:             = space dimension of mesh (2)
%
%   PSPS:            = flag for plane stress ('s') or plane strain ('n')
%
```

```
% Definitions for output:
%
%   ElemK:          = element stiffness matrix containing stiffness
%                     entries in the following arrangement, where
%                     wij corresponds to weighting function (i),
  coordinate
%                     direction (j), and ukl corresponds to displacement
%                     function (k), coordinate direction (l):
%                               u1x  u1y  u2x  u2y  u3x  u3y  u4x  u4y
%                     w1x  ElemK[ .    .    .    .    .    .    .    .
%                     w1y         .    .    .    .    .    .    .    .
%                     w2x         .    .    .    .    .    .    .    .
%                     w2y         .    .    .    .    .    .    .    .
%                     w3x         .    .    .    .    .    .    .    .
%                     w3y         .    .    .    .    .    .    .    .
%                     w4x         .    .    .    .    .    .    .    .
%                     w4y         .    .    .    .    .    .    .    . ];
%
%   ElemF:          = element force vector containing force entries in the
%                     following arrangement:
%                     w1x  ElemF[ .
%                     w1y         .
%                     w2x         .
%                     w2y         .
%                     w3x         .
%                     w3y         .
%                     w4x         .
%                     w4y         . ];
%
% Definitions of local constants:
%
%   nst:            = size of element arrays (ndf*nel)
%
%



% Set Material Properties

ElemE = mateprop(1);
Elemv = mateprop(2);
thick = mateprop(3);

if PSPS == 's' %Plane Stress

    Dmat = ElemE/(1-Elemv^2)*[1       Elemv   0
                              Elemv   1       0
                              0       0       (1-Elemv)/2];

else %Plane Strain
```

```
end

% Initialize Matrix and Vector

nst = nel*ndf;
ElemK = zeros(nst);
ElemF = zeros(nst,1);

% Load Guass Integration Points

if nel == 3
    lint = 4;
else
    lint = 4;
end

% Loop over integration points
for l = 1:lint

        if nel == 3
            [Wgt,r,s] =  intpntt(l,lint,0);
        else
            [Wgt,r,s] =  intpntq(l,lint,0);
        end

        % Evaluate local basis functions at integration point
        shp = shpl_2d(r,s,nel);

        % Evaluate first derivatives of basis functions at int. point
        [Qxy, Jdet] = shpg_2d(shp,xl,nel);

        % Form B matrix
        if nel == 3
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0
                0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3)
                Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3)];
        else
        Bmat = [Qxy(1,1) 0        Qxy(1,2) 0        Qxy(1,3) 0        Qxy
            (1,4) 0
                0        Qxy(2,1) 0        Qxy(2,2) 0        Qxy(2,3) 0
                  Qxy(2,4)
                Qxy(2,1) Qxy(1,1) Qxy(2,2) Qxy(1,2) Qxy(2,3) Qxy(1,3) Qxy
                    (2,4) Qxy(1,4)];

        end

        % Update integration weighting factor
        W = Wgt*Jdet*thick;
```

```
        ElemK = ElemK + W*Bmat'*Dmat*Bmat;
end %je
```