

# Finite Element Analysis of a plate with a hole using Constant Strain triangle, four and eight noded isoparametric quadrilateral elements

SAGAR BHATT

Person Number: 50170651

Department of Mechanical and Aerospace Engineering,  
University at Buffalo

## CONTENTS

<b>I Introduction</b>	<b>2</b>
I Problem Set Up . . . . .	4
<b>II Method of Solution</b>	<b>4</b>
I Governing Equations . . . . .	4
I.1 For Isoparametric Elements: . . . . .	8
I.2 Stress Concentration Factor: . . . . .	9
I.3 Energy Norm: . . . . .	9
I.4 Boundary Conditions: . . . . .	9
<b>III Results</b>	<b>9</b>
I Constant Strain Triangle Element . . . . .	9
II Four Noded Quadrilateral Isoparametric Element . . . . .	11
III Eight Noded Quadrilateral Isoparametric Element . . . . .	12
IV Results from ABAQUS . . . . .	13
<b>IV Summary:</b>	<b>13</b>
<b>V Appendix</b>	<b>14</b>
I Code to Plot Deformation: . . . . .	14
II Code for CST Element: . . . . .	15
III Code for 4-noded quadrilateral isoparametric Element: . . . . .	22
IV Code for 4-noded quadrilateral isoparametric Element: . . . . .	34

## LIST OF FIGURES

1 A constant strain triangle element . . . . .	3
2 A 4-noded quadrilateral element . . . . .	3

3	An 8-noded quadrilateral element . . . . .	3
4	The problem setup . . . . .	4
5	The quadrant being used in this study considering symmetry . . . . .	4
6	Deformation of the plate due to the load for CST elements . . . . .	10
7	Error in energy norm for CST elements . . . . .	10
8	Deformation of the plate due to the load for Q4 elements . . . . .	11
9	Error in energy norm for Q4 elements . . . . .	11
10	Deformation of the plate due to the loading for Q8 elements . . . . .	12
11	Error in energy norm for Q8 elements . . . . .	12
12	Displacement plotted by ABAQUS . . . . .	13

## Abstract

*This project aims at studying the deformation of a thin plate with a central circular hole when the plate is loaded in tension with a constant load. Finite element codes were developed in MATLAB using constant strain triangle elements, four- and eight-noded isoparametric elements. A convergence study was performed based on the energy norm for all three cases and the stress concentration factor around the circular whole was also investigated. The computed results were then compared with the results obtained using commercial code ABAQUS.*

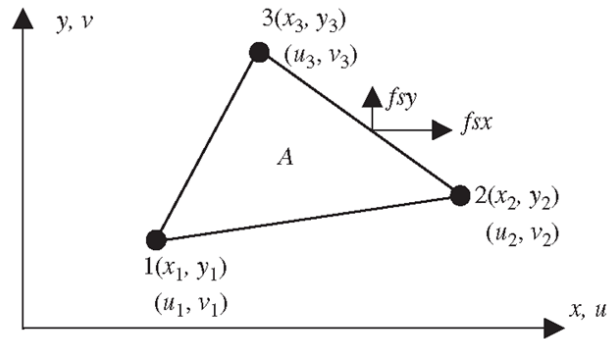
## I. INTRODUCTION

A machine component is bound to have irregularities like a hole in its geometry for various design reasons. These irregularities can contribute immensely to the strength of the part. Configuring the structures with discontinuities is one of the most important topics in the construction of ships, aero-planes, cars etc. Examples of problems in which discontinuities play prominent role in the physical behavior of a system are numerous. From mathematical point of view, analytical solutions are possible only for a limited class of such problems. Many times an accurate solution was not possible due to the complexity of the discontinuity configuration. However, with the advent of Finite element method (FEM), these analyses can now be performed with a degree of accuracy.[3]

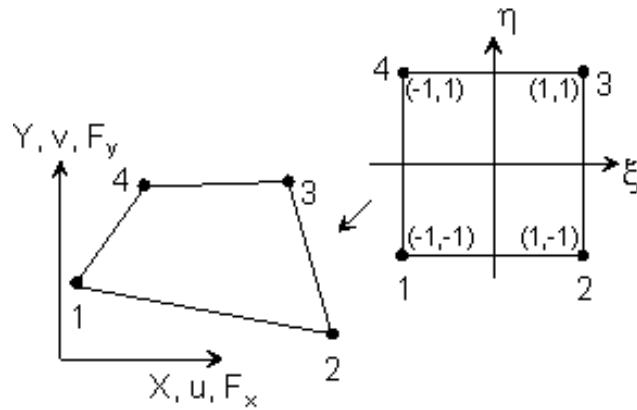
A Constant Strain Triangle element, also referred to as a CST element or a T3 element, has constant shape functions which when applied to plane stress or plane strain conditions, yield approximate solutions for stress and strain fields that are constant throughout the domain of the element.

Introduction of isoparametric element formulation in 1968 by Bruce Irons was one of the most important contributions to the field of Finite Elements because it gave us the tools to overcome the complexity of dealing with the consistency requirements for higher order elements with curved boundaries. The same shape functions are used to interpolate the nodal

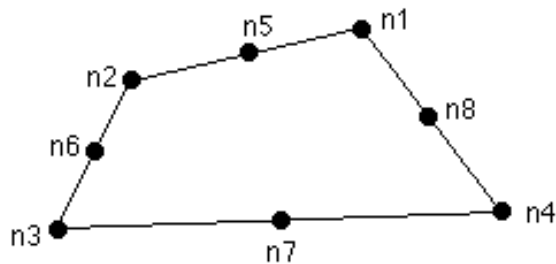
coordinates and displacements. The whole element is transformed into an ideal element (e.g. a square element) by mapping it into a different coordinate system. The shape functions are then defined for this idealized element. Here two quadrilateral isoparametric elements are being considered, 4-noded (also called Q4 element) and 8-noded (also called Q8 element).



**Figure 1:** A constant strain triangle element



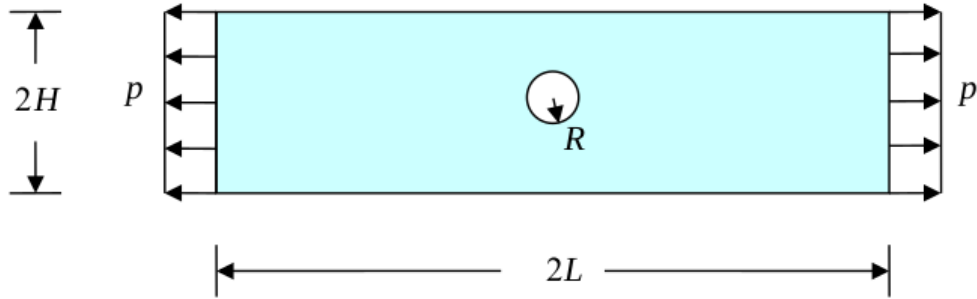
**Figure 2:** A 4-noded quadrilateral element



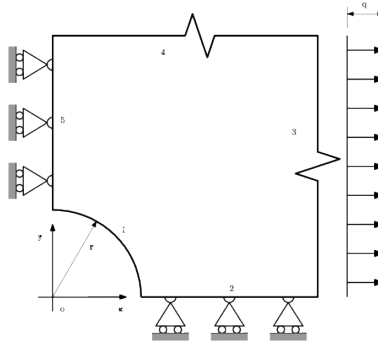
**Figure 3:** An 8-noded quadrilateral element

## I. Problem Set Up

The problem we are considering consists of a finite plate of length  $2L$  and width  $2H$  with a central hole of radius  $R$ , depicted in fig. 4. The relationship between these dimensions is given by:  $L/H = \alpha$ ,  $R/H = \beta$  and these vales were determined from the my UB person number (50170651). Let  $abcd ijkl$  be the person number, the the pooisson's ratio,  $\nu = j/20$ . ans  $\alpha = (k + 1)/2$ ,  $\beta = 1/(l + 3)$ . Young's modulus was arbitrarily assumed to be 2.5. Hence, in this case,  $\alpha = L/H = 3$ ,  $\beta = R/H = 0.25$  and  $\nu = 0.3$ . Since the problem was symmetrical, we broke the problem to a quarter plate and solved the problem for just one quadrant of the plate using these parameters.



**Figure 4:** *The problem setup*



**Figure 5:** *The quadrant being used in this study considering symmetry*

## II. METHOD OF SOLUTION

### I. Governing Equations

A general approach for a displacement based finite element formulation is given by the following nine steps[1]. We will demonstrate this formulation for 2D CST. The iso parametric formulation follows the same basic guideline with additional steps involving coordinate transformation and Gauss Quadrature.

1. Choose the coordinate system and define the node numbering system, nodal displacements and body forces for the element.

$$\vec{u}^e = \{u_1 \ v_1 \ u_2 \ v_2 \ u_3 \ v_3\}^T$$

$$\vec{f}^e = \{f_{x1} \ f_{y1} \ f_{x2} \ f_{y2} \ f_{x3} \ f_{y3}\}^T$$

2. Choose a displacement function that can represent the fundamental deformation of the elements. According to Principle of Virtual Work (PVW):

$$\int_{\Omega} \sigma_{ij} d\Omega = \int_{\Gamma_t} t_i \delta u_i d\Gamma + \int_{\Omega} f_i \delta u_i d\Omega$$

Here, if the highest order derivative is  $n^{th}$  order,  $C^{n-1}$  continuity is required. In case of elastic bodies, the highest order derivative is 1<sup>st</sup> order, hence we require  $C_0$  continuity. So, Let:

$$u(x) = \alpha_1 + \alpha_2 x + \alpha_3 y$$

$$v(x) = \alpha_4 + \alpha_5 x + \alpha_6 y$$

Then,

$$\vec{u}(\vec{x}) = \Phi(\vec{x})\vec{\alpha}$$

i.e.

$$\begin{bmatrix} u(x, y) \\ v(x, y) \end{bmatrix} = \begin{bmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix}$$

or,

$$\vec{u}^e = \mathbb{A}\vec{\alpha}$$

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} u(x_1, y_1) \\ v(x_1, y_1) \\ u(x_1, y_1) \\ v(x_1, y_1) \\ u(x_2, y_2) \\ v(x_3, y_3) \end{bmatrix} = \begin{bmatrix} 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_1 & y_1 \\ 1 & x_1 & y_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_2 & y_2 \\ 1 & x_3 & y_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x_3 & y_3 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix}$$

or,

$$\begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \end{bmatrix} = \begin{bmatrix} \bar{\mathbb{A}} & 0 \\ 0 & \bar{\mathbb{A}} \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix}$$

where,

$$\bar{\mathbb{A}} = \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{bmatrix}$$

Then,

$$\bar{\mathbb{A}}^{-1} = \begin{bmatrix} \bar{\mathbb{A}}^{-1} & 0 \\ 0 & \bar{\mathbb{A}}^{-1} \end{bmatrix}$$

$$\implies \vec{u}(\vec{x}) = \Phi(\vec{x})\bar{\mathbb{A}}^{-1}\vec{u}^e$$

3.

$$\vec{\epsilon}(\vec{x}) = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \end{bmatrix}$$

$$\epsilon_{xx} = \frac{\partial u}{\partial x}$$

$$\epsilon_{yy} = \frac{\partial v}{\partial y}$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

$$\vec{\epsilon}(\vec{x}) = \partial\Phi\mathbb{A}^{-1}\vec{u}^e$$

$$\implies \vec{\epsilon}(\vec{x}) = \mathbb{B}\vec{u}^e$$

4.

$$\vec{\sigma}(\vec{x}) = \mathbb{C}(\vec{x})\vec{\epsilon}(\vec{x})$$

Where,

$$\mathbb{C}(\vec{x}) = \frac{E(1-\nu)}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1 & \frac{\nu}{1-\nu} & 0 \\ \frac{\nu}{1-\nu} & 1 & 0 \\ 0 & 0 & \frac{2\nu}{2(1-\nu)} \end{bmatrix}$$

5.

$$\delta\vec{u}(\vec{x}) = (N)(\vec{x})\delta\vec{u}^e$$

$$\partial\vec{\epsilon}(\vec{x}) = \mathbb{B}\partial\vec{u}^e$$

6. Invoke PVW and develop elemental stiffness matrix:

$$\mathbb{K}^e = \int_{\Omega} \mathbb{B}^T \mathbb{C} \mathbb{B} d\Omega$$

Now,

$$\mathbb{K}^e \vec{u}^e = \vec{f}^e$$

7. Assemble and enforce boundary conditions

$$\mathbb{K}\vec{u} = \vec{f}$$

8. Solve for  $\vec{u}$

9. Post-Processing

## For Isoparametric Elements:

### Shape Functions:

- 4-noded element:

$$- N_1(\zeta, \eta) = \frac{1}{4}(1 - \zeta)(1 - \eta)$$

$$- N_2(\zeta, \eta) = \frac{1}{4}(1 + \zeta)(1 - \eta)$$

$$- N_3(\zeta, \eta) = \frac{1}{4}(1 + \zeta)(1 + \eta)$$

$$- N_4(\zeta, \eta) = \frac{1}{4}(1 - \zeta)(1 + \eta)$$

- 8-noded element:

$$- N_1(\zeta, \eta) = -\frac{1}{4}(1 - \zeta)(1 - \eta)(\zeta + \eta + 1)$$

$$- N_2(\zeta, \eta) = \frac{1}{4}(1 + \zeta)(1 - \eta)(\zeta - \eta - 1)$$

$$- N_3(\zeta, \eta) = \frac{1}{4}(1 + \zeta)(1 + \eta)(\zeta + \eta - 1)$$

$$- N_4(\zeta, \eta) = -\frac{1}{4}(1 - \zeta)(1 + \eta)(\zeta - \eta + 1)$$

$$- N_5(\zeta, \eta) = \frac{1}{2}(1 - \zeta^2)(1 - \eta)$$

$$- N_6(\zeta, \eta) = \frac{1}{2}(1 - \eta^2)(1 + \zeta)$$

$$- N_7(\zeta, \eta) = \frac{1}{2}(1 - \zeta^2)(1 + \eta)$$

$$- N_8(\zeta, \eta) = \frac{1}{2}(1 - \eta^2)(1 - \zeta)$$

Also,

$$x(\zeta, \eta) = \sum_i N_i(\zeta, \eta)x_i$$

$$y(\zeta, \eta) = \sum_i N_i(\zeta, \eta)y_i$$

$$u(\zeta, \eta) = \sum_i N_i(\zeta, \eta)u_i$$

$$v(\zeta, \eta) = \sum_i N_i(\zeta, \eta)v_i$$

$$\mathbf{Jacobian: } \mathbb{J} = \begin{bmatrix} \frac{\partial x}{\partial \zeta} & \frac{\partial y}{\partial \zeta} \\ \frac{\partial x}{\partial \eta} & \frac{\partial y}{\partial \eta} \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$$

**Strain:**

$$\vec{\epsilon}(\vec{x}) = \begin{bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{bmatrix}$$



$$\epsilon_{xx} = \frac{\partial u}{\partial x} = \sum_i (\hat{J}_{11} \frac{\partial N_i}{\partial \zeta} + \hat{J}_{12} \frac{\partial N_i}{\partial \eta}) u_i$$

$$\epsilon_{yy} = \frac{\partial v}{\partial y} = \sum_i (\hat{J}_{21} \frac{\partial N_i}{\partial \zeta} + \hat{J}_{22} \frac{\partial N_i}{\partial \eta}) v_i$$

$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$

### Stress Concentration Factor:

The stress concentration factor was computed base on the following formula:

$$SCF = \frac{\sigma_{max}}{\sigma_{nom}}, \text{ where,}$$

$$\sigma_{nom} = \frac{\text{load}}{\text{minimum cross section}}$$

### Energy Norm:

The energy norm was computed using:  $\frac{|U_{FE} - U_{EX}|}{|U_{EX}|}$

### Boundary Conditions:

Since we are considering only one quadrant of the plate, fig.5, the following boundary conditions were imposed on this geometry:

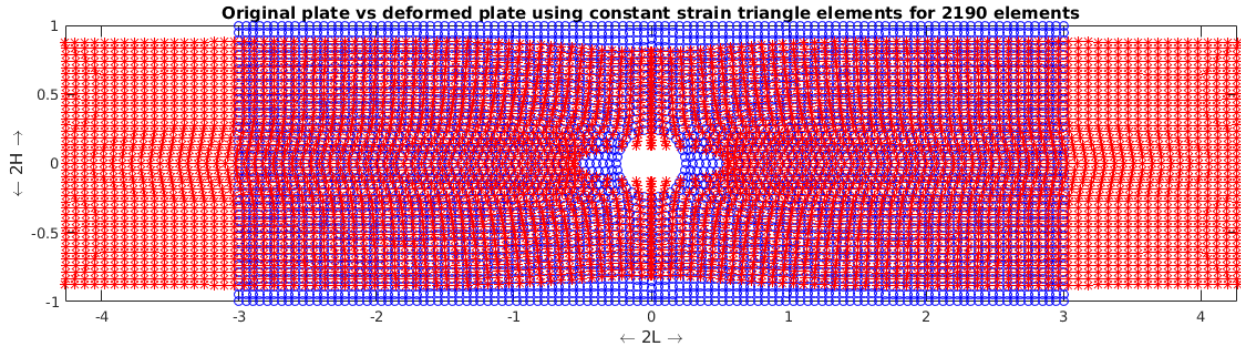
- Left edge : No displacement in x-direction i.e. 1<sup>st</sup> degree of freedom set to 0.
- Bottom edge : No displacement in y-direction i.e. 2<sup>nd</sup> degree of freedom set to 0.

## III. RESULTS

### I. Constant Strain Triangle Element

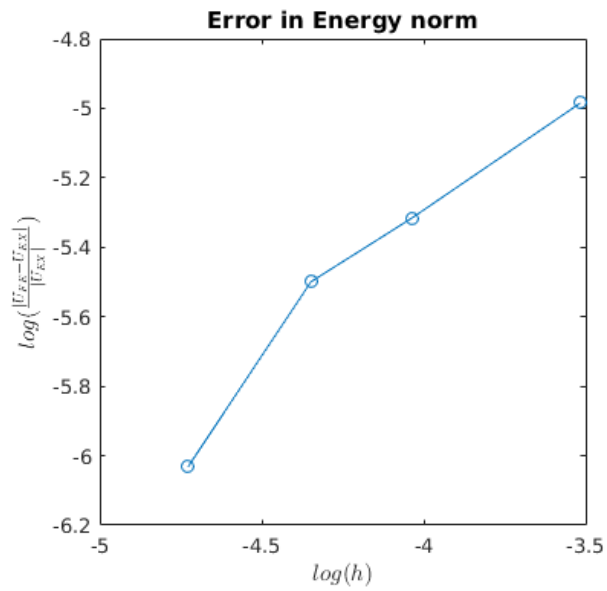
Number of Elements	Stress Concentration Factor	Maximum Displacement
2190	1.3363	1.2619
594	1.3580	1.2589
394	1.1950	1.2567
282	1.3594	1.2557
156	1.0475	1.2531

DEFORMATION:



**Figure 6:** Deformation of the plate due to the load for CST elements

CONVERGENCE BASED ON ENERGY NORM:

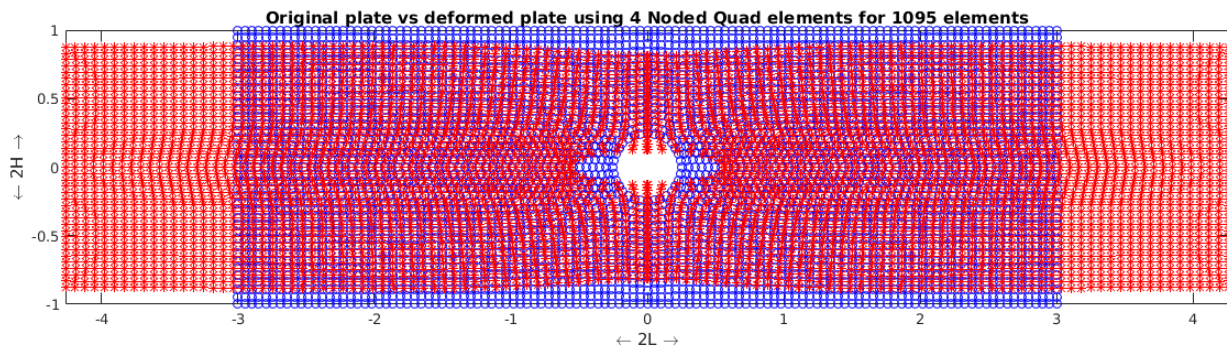


**Figure 7:** Error in energy norm for CST elements

## II. Four Noded Quadrilateral Isoparametric Element

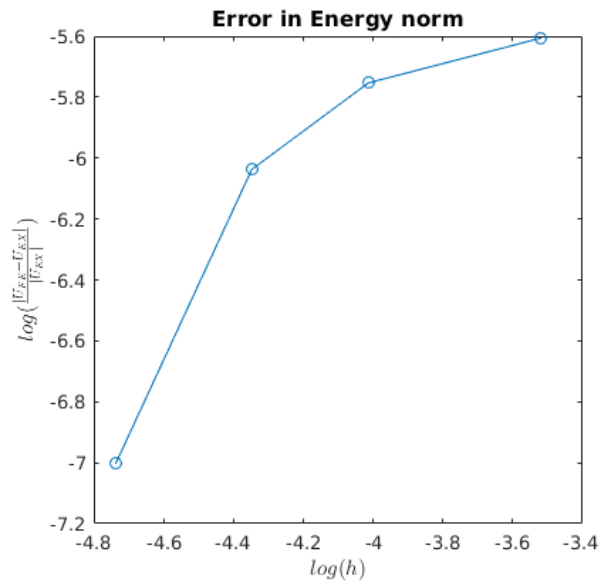
Number of Elements	Stress Concentration Factor	Maximum Displacement
1095	1.2679	1.2627
300	1.7638	1.2616
197	1.6378	1.2598
137	1.5582	1.2588
78	1.4677	1.2582

DEFORMATION:



**Figure 8:** Deformation of the plate due to the load for  $Q_4$  elements

CONVERGENCE BASED ON ENERGY NORM:

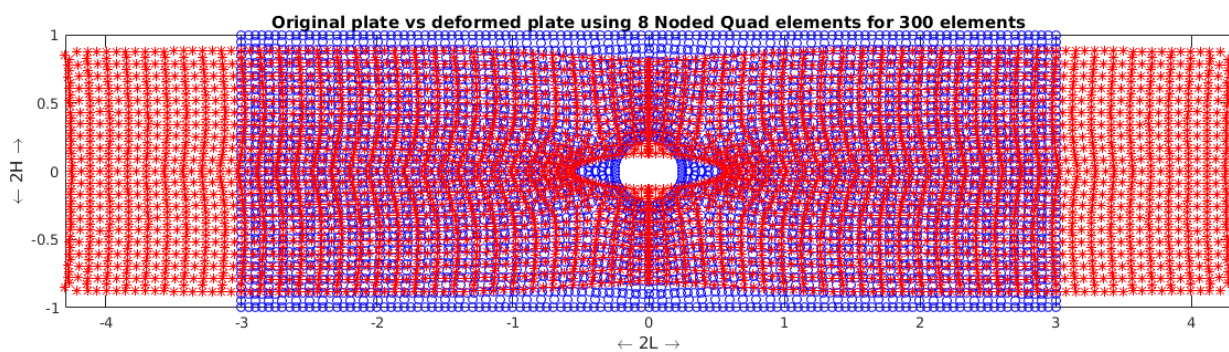


**Figure 9:** Error in energy norm for  $Q_4$  elements

### III. Eight Noded Quadrilateral Isoparametric Element

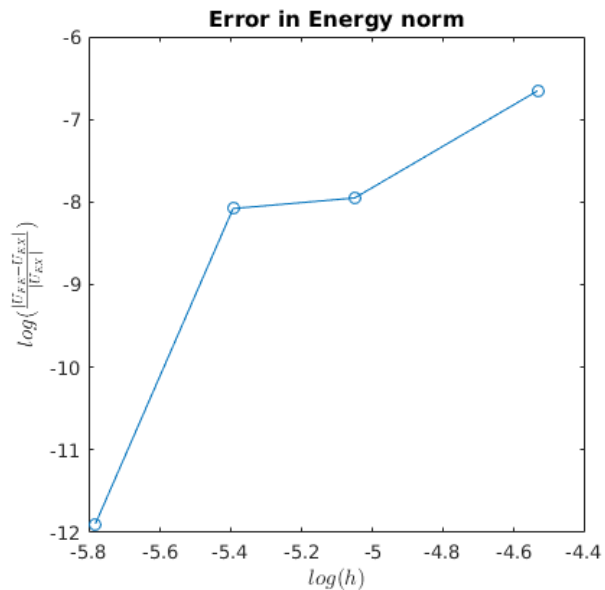
Number of Elements	Stress Concentration Factor	Maximum Displacement
300	1.7540	1.3005
297	1.7378	1.3005
197	1.6175	1.3070
137	1.5389	1.3093
78	1.4529	1.3224

DEFORMATION:



**Figure 10:** Deformation of the plate due to the loading for  $Q8$  elements

CONVERGENCE BASED ON ENERGY NORM:



**Figure 11:** Error in energy norm for  $Q8$  elements

## IV. Results from ABAQUS

The problem was solved using ABAQUS as well. The maximum displacement obtained through the FE code was compared with the maximum displacement obtained through ABAQUS:

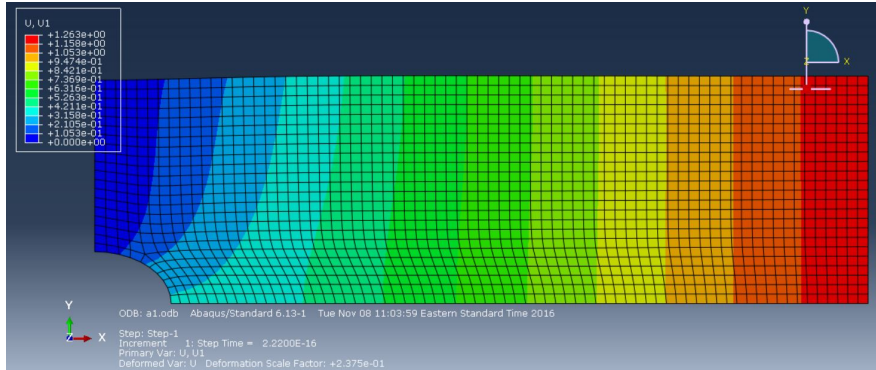
$U_{max}$ , Abaqus: 1.26394

$U_{max}$ , CST: 1.2619

$U_{max}$ , Q4: 1.2627

$U_{max}$ , Q8: 1.3005

As we can see from the above comparison, the closest result to ABAQUS's solution is that of Q4 elements. This is because the mesh chosen for the study on ABAQUS was equipped with Q4 elements. Hence, a comparative study between these two results is more suitable. However, the results obtained through any other elements should not be very different. The proximity of the results through all the methods corroborates the accuracy of the computation. Figure 12 demonstrates the deformation of a quadrant of the plate under the load.



**Figure 12:** *Displacement plotted by ABAQUS*

## IV. SUMMARY:

Finite element codes were developed in MATLAB using constant strain triangle elements, four- and eight-noded isoparametric elements. A convergence study was performed based on the energy norm  $a$  for all three cases and the stress concentration factor around the circular hole was also investigated. The computed results were then compared with the results obtained using commercial code ABAQUS. The computed maximum displacement was  $\approx 1.263$  (for Q4 elements) where as the result from the result from ABAQUS showed this to be  $= 1.264$ . The results are in very good agreement with each other. The stress concentration factor was also computed. The deformation was plotted using the MATLAB code as well as ABAQUS. The convergence study based on the error in energy norm was also

computed and is presented here in graphical form. We can see that the this error increases as 'h' increases or as the number of elements decreases. The results of deformation show that the maximum displacement corresponds to the node at the center of each loaded side. Due to symmetric loading, the central nodes do not move and the circle gets deformed to an ellipse. Another observation to be made here is as the number of nodes per element increases, we can see the computed deformation increases and the minor axis of now deformed hole is much smaller resulting in a much sharper ellipse.

## REFERENCES

- [1] Gary F. Dargush. Lecture notes in finite element analysis(mae529). *Department of Mechanical and Aerospace Engineering, University at Buffalo, The State University of New York*, Fall 2016.
- [2] E Hart and V Hudramovich. Projection-iterative schemes for the realization of the finite-element method in problems of deformation of plates with holes and inclusions. *Journal of Mathematical Sciences*, 203(1), 2014.
- [3] Sharaban Thohura and Md Shahidul Islam. Study of the effect of finite element mesh quality on stress concentration factor of plates with holes. *Proceedings of the 15th Annual Paper Meet*, 7:08, 2014.

## V. APPENDIX

### I. Code to Plot Deformation:

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %                                                                                               %
3 %                               PLOTTING THE DEFORMATION                                       %
4 %                                                                                               %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7
8 % PLoTting the initial position nodes
9
10 plot(Nodes(:,2),Nodes(:,3),'ob'); axis equal; axis tight; hold on;
11 plot(-Nodes(:,2),-Nodes(:,3),'ob'); axis equal; axis tight; hold on;
12 plot(-Nodes(:,2),Nodes(:,3),'ob'); axis equal; axis tight; hold on;
13 plot(Nodes(:,2),-Nodes(:,3),'ob'); axis equal; axis tight; hold on;
14
15 % Finding the final position of the nodes
16 j=1;
17 for i=1:2:size(U)
18     n_disp(j,1)=U(i);

```

```

19     n_disp(j,2)=U(i+1);
20     j=j+1;
21 end
22 n_final(:,1)=Nodes(:,2)+n_disp(:,1);
23 n_final(:,2)=Nodes(:,3)+n_disp(:,2);
24
25
26 % Plotting the final positions of the nodes
27
28 plot(n_final(:,1),n_final(:,2),'*r'); axis equal; axis tight; hold on;
29 plot(-n_final(:,1),n_final(:,2),'*r'); axis equal; axis tight; hold on;
30 plot(n_final(:,1),-n_final(:,2),'*r'); axis equal; axis tight; hold on;
31 plot(-n_final(:,1),-n_final(:,2),'*r'); axis equal; axis tight; hold on;set(
    gca,'color','black')

```

## II. Code for CST Element:

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %                                                                 %
3 % Elastic Constant Strain Triangular Elements                    %
4 %                                                                 %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 % Clear workspace
8 clc
9 clear all
10
11 nod1= csvread('Nodes_1.csv');
12 nod2= csvread('Nodes_2.csv');
13 nod3= csvread('Nodes_3.csv');
14 nod4= csvread('Nodes_4.csv');
15 nod5= csvread('Nodes_5.csv');
16
17 Elm1=csvread('Elements_1.csv');
18 Elm2=csvread('Elements_2.csv');
19 Elm3=csvread('Elements_3.csv');
20 Elm4=csvread('Elements_4.csv');
21 Elm5=csvread('Elements_5.csv');
22
23 for no=1:5
24
25     % Read nodes and coords
26     if no==1
27         Nodes = nod1;
28     end
29     if no==2
30         Nodes = nod2;
31     end
32     if no==3

```

```

33     Nodes = nod3;
34 end
35 if no==4
36     Nodes = nod4;
37 end
38 if no==5
39     Nodes = nod5;
40 end
41 [N,1] = size(Nodes);
42
43 % Read element material id, thickness and nodal connectivity
44 if no==1
45     Elems = Elm1;
46 end
47 if no==2
48     Elems = Elm2;
49 end
50 if no==3
51     Elems = Elm3;
52 end
53 if no==4
54     Elems = Elm4;
55 end
56 if no==5
57     Elems = Elm5;
58 end
59
60 [E,1] = size(Elems);
61 j_dbc=1;
62 j_nbc=1;
63
64 % Read material info
65 Mats = load('Materials.txt');
66 [M,1] = size(Mats);
67
68 %Determine Dirichlet BC
69 for (i=1:N)
70     if (Nodes(i,2)==0)
71         DBC(j_dbc,1)=Nodes(i,1);
72         DBC(j_dbc,2)=1;
73         DBC(j_dbc,3)=0;
74         j_dbc=j_dbc+1;
75     end
76     if (Nodes(i,3)==0)
77         DBC(j_dbc,1)=Nodes(i,1);
78         DBC(j_dbc,2)=2;
79         DBC(j_dbc,3)=0;
80         j_dbc=j_dbc+1;
81     end

```



```

82     end
83     [P,1] = size(DBC);
84     % Determine Neumann BC
85     for (i=1:N)
86         if (Nodes(i,2)==3)
87             right(j_nbc,1)=Nodes(i,1);
88             right(j_nbc,2)=1;
89             right(j_nbc,3)=0;
90             j_nbc=j_nbc+1;
91         end
92     end
93     j_nbc=1;
94
95
96     comb=combnk(right(:,1),2);
97
98     for i=1:E
99         for j=1:size(comb(:,1))
100            if comb(j,:) == Elems(i,4:5) | comb(j,:) == Elems(i,5:6) | comb(j
, :) == Elems(i,[4 6]) | comb(j,:) == Elems(i,[6 4]) | comb(j,:) == Elems(i
,[5 4]) | comb(j,:) == Elems(i,[6 5])
101                NBC(j_nbc,1)=Elems(i,1);
102                NBC(j_nbc,2:3)=comb(j,:);
103                NBC(j_nbc,4)=1;
104                NBC(j_nbc,5)=1;
105                j_nbc=j_nbc+1;
106            end
107        end
108    end
109    [Q,1] = size(NBC);
110
111    % Determining the hole nodes
112    i_hol=1;
113    for i=1:N
114        if (Nodes(i,2)<=0.25 && Nodes(i,3)<=0.25)
115            hole(i_hol)=Nodes(i,1);
116            i_hol=i_hol+1;
117        end
118    end
119
120    % Determining the hole elements
121    i_hol=1;
122    for i=1:E
123        if (hole(i_hol)==Elems(i,4) || hole(i_hol)==Elems(i,5) || hole(i_hol)==
Elems(i,6))
124            hol_el(i_hol)=Elems(i,1);
125            i_hol=i_hol+1;
126        end
127    end

```

```

128     hol_el=unique(hol_el);
129
130     % Identify out-of-plane conditions
131     % ipstrn = 1    Plane strain
132     % ipstrn = 2    Plane stress
133     ipstrn = 2;
134
135     % Determine total number of degrees-of-freedom
136     udof = 2;      % Degrees-of-freedom per node
137     NDOF = N*udof;
138
139     % Initialize global matrix and vectors
140     K = zeros(NDOF,NDOF); % Stiffness matrix
141     U = zeros(NDOF,1);   % Displacement vector
142     F = zeros(NDOF,1);   % Force vector
143
144     % Set penalty for displacement constraints
145     Klarge = 10^10;
146
147     % Loop over CST element
148     for e = 1:E
149
150         % Establish element connectivity and coordinates
151         Nnums = Elems(e,4:6);
152         xy = Nodes(Nnums(:),2:3);
153
154         % Extract element thickness for plane stress
155         h = Elems(e,3);
156
157         % Extract element elastic Young's modulus and Poisson's ratio
158         Y = Mats(Elems(e,2),2);
159         nu = Mats(Elems(e,2),3);
160
161         % Construct element stiffness matrix
162         [Ke] = CST_El_Stiff(ipstrn,xy,h,Y,nu);
163
164         % Assemble element stiffness matrix into global stiffness matrix
165         ig = udof*(Nnums(:)-1);
166         for ni = 1:3
167             i0 = udof*(ni-1);
168             for nj = 1:3
169                 j0 = udof*(nj-1);
170                 for i = 1:udof
171                     for j = 1:udof
172                         K(ig(ni)+i,ig(nj)+j) = K(ig(ni)+i,ig(nj)+j) + Ke(i0+i,
173 j0+j);
174                                     end
175                                 end
176                             end
177                         end
178                     end
179                 end
180             end
181         end

```

```

176         end
177     end
178     % K
179
180     % Construct global force vector
181     for q = 1:Q
182
183         % Determine loaded edge
184         e = NBC(q,1);
185         in1 = NBC(q,2);
186         in2 = NBC(q,3);
187         idof = NBC(q,4);
188         tval = NBC(q,5);
189         h = Elems(e,3);
190
191         % Establish edge length
192         xlen2 = (Nodes(in2,2)-Nodes(in1,2))^2;
193         ylen2 = (Nodes(in2,3)-Nodes(in1,3))^2;
194         elen = sqrt(xlen2+ylen2);
195         fval = tval*elen*h/2;
196         iloc1 = udof*(in1-1)+idof;
197         iloc2 = udof*(in2-1)+idof;
198         F(iloc1) = F(iloc1) + fval;
199         F(iloc2) = F(iloc2) + fval;
200         F;
201
202     end
203
204     for p = 1:P
205         inode = DBC(p,1);
206         idof = DBC(p,2);
207         idiag = udof*(inode-1) + idof;
208         K(idiag,idiag) = Klarge;
209         F(idiag) = Klarge*DBC(p,3);
210     end
211     % K
212     % F
213
214     % Solve system to determine displacements
215     U = K\F;
216
217     % Recover internal element displacement, strains and stresses
218     Disp = zeros(E,6);
219     Eps = zeros(E,3);
220     Sig = zeros(E,3);
221
222     for e = 1:E
223
224         % Establish element connectivity and coordinates

```

```

225     Nnums = Elems(e,4:6);
226     xy = Nodes(Nnums(:),2:3);
227
228     % Extract element thickness for plane stress
229     h = Elems(e,3);
230
231     % Extract element elastic Young's modulus and Poisson's ratio
232     Y = Mats(Elems(e,2),2);
233     nu = Mats(Elems(e,2),3);
234
235     % Extract element nodal displacements
236     inode1 = Nnums(1);
237     inode2 = Nnums(2);
238     inode3 = Nnums(3);
239     Disp(e,1) = U(udof*(inode1-1)+1);
240     Disp(e,2) = U(udof*inode1);
241     Disp(e,3) = U(udof*(inode2-1)+1);
242     Disp(e,4) = U(udof*inode2);
243     Disp(e,5) = U(udof*(inode3-1)+1);
244     Disp(e,6) = U(udof*inode3);
245
246     u = Disp(e,:)';
247     [eps,sig] = CST_El_Str(ipstrn,xy,u,h,Y,nu);
248
249     % Store element strains
250     Eps(e,:) = eps;
251
252     % Store element stresses
253     Sig(e,:) = sig;
254
255     end
256
257     % Computing Strain concentration factor
258
259     sig_nom= 1/0.75;
260
261     for i=1:size(hol_el)
262         sig_max=max(Sig(hol_el(i),:,:));
263     end
264     SCF(no)=mean(sig_max)/sig_nom;
265
266
267     PE(no,1)=0.5*U'*K*U;
268
269     PE(no,2)=3/N;
270     %     if (no==1)
271     %         str=sprintf('Original plate vs deformed plate using constant strain
triangle elements for %d elements',E);
272     %         figure;

```

```

273 %           Plot_deformation;
274 %           title(str);
275 %           xlabel('\leftarrow 2L \rightarrow');
276 %           ylabel('\leftarrow 2H \rightarrow');
277         max(U)
278     end
279 %     E
280     clearvars -except nod1 nod2 nod3 nod4 nod5 Elm1 Elm2 Elm3 Elm4 Elm5 PE SCF
;
281 end
282
283 for i=1:5
284     if (PE(i,2)==min(PE(:,2)))
285         PE_ex=PE(i,1);
286     end
287 end
288 PE(:,1)=abs(PE(:,1)-PE_ex)/abs(PE_ex);
289
290 % figure;
291 % plot(log(PE(:,2)),log(PE(:,1)), '-o');
292 % title('Error in Energy norm');
293 % xlabel('$\log(h)$', 'Interpreter','latex');
294 % ylabel('$\log(\frac{|U_{FE}-U_{EX}|}{|U_{EX}|})$', 'Interpreter','latex');
295 % axis square;
296 %
297 % Disp
298 % Eps
299 % Sig

1 function Ke = CST_El_Stiff(ipstrn,xy,h,Y,nu)
2
3 ndof = 6;
4 Ke = zeros(ndof,ndof);
5
6 Abar = [ 1 xy(1,1) xy(1,2); 1 xy(2,1) xy(2,2); 1 xy(3,1) xy(3,2) ];
7 A = det(Abar)/2;
8
9 B = (1/A/2)*[ xy(2,2)-xy(3,2) 0 xy(3,2)-xy(1,2) 0 xy(1,2)-xy(2,2) 0;
10              0 xy(3,1)-xy(2,1) 0 xy(1,1)-xy(3,1) 0 xy(2,1)-xy(1,1);
11              xy(3,1)-xy(2,1) xy(2,2)-xy(3,2) xy(1,1)-xy(3,1) ...
12              xy(3,2)-xy(1,2) xy(2,1)-xy(1,1) xy(1,2)-xy(2,2) ];
13
14 if (ipstrn == 1)
15     c = Y*(1-nu)/(1-2*nu)/(1+nu);
16     C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];
17 else
18     c = Y/(1-nu)/(1+nu);
19     C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
20 end
21

```

```

22 Ke = h*A*B'*C*B;

1 function [eps, str] = CST_El_Str(ipstrn, xy, u, h, Y, nu)
2
3 ndof = 6;
4
5 Abar = [ 1 xy(1,1) xy(1,2); 1 xy(2,1) xy(2,2); 1 xy(3,1) xy(3,2) ];
6 A = det(Abar)/2;
7
8 B = (1/A/2)*[ xy(2,2)-xy(3,2) 0 xy(3,2)-xy(1,2) 0 xy(1,2)-xy(2,2) 0;
9              0 xy(3,1)-xy(2,1) 0 xy(1,1)-xy(3,1) 0 xy(2,1)-xy(1,1);
10             xy(3,1)-xy(2,1) xy(2,2)-xy(3,2) xy(1,1)-xy(3,1) ...
11             xy(3,2)-xy(1,2) xy(2,1)-xy(1,1) xy(1,2)-xy(2,2) ];
12
13 if (ipstrn == 1)
14     c = Y*(1-nu)/(1-2*nu)/(1+nu);
15     C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];
16 else
17     c = Y/(1-nu)/(1+nu);
18     C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
19 end
20
21 eps = B*u;
22 str = C*eps;

```

### III. Code for 4-noded quadrilateral isoparametric Element:

```

1 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2 %
3 % Elastic 4-node Quadrilateral Elements %
4 % %
5 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7 % Clear workspace
8 clc
9 clear
10
11 % Read nodes and coords
12 nod1= csvread('Nodes_1.csv');
13 nod2= csvread('Nodes_2.csv');
14 nod3= csvread('Nodes_3.csv');
15 nod4= csvread('Nodes_4.csv');
16 nod5= csvread('Nodes_5.csv');
17
18 Elm1=csvread('Elements_1.csv');
19 Elm2=csvread('Elements_2.csv');
20 Elm3=csvread('Elements_3.csv');
21 Elm4=csvread('Elements_4.csv');
22 Elm5=csvread('Elements_5.csv');
23

```

```

24 for no=1:5
25
26     % Read nodes and coords
27     if no==1
28         Nodes = nod1;
29     end
30     if no==2
31         Nodes = nod2;
32     end
33     if no==3
34         Nodes = nod3;
35     end
36     if no==4
37         Nodes = nod4;
38     end
39     if no==5
40         Nodes = nod5;
41     end
42     [N,1] = size(Nodes);
43
44     % Read element material id, thickness and nodal connectivity
45     if no==1
46         Elems = Elm1;
47     end
48     if no==2
49         Elems = Elm2;
50     end
51     if no==3
52         Elems = Elm3;
53     end
54     if no==4
55         Elems = Elm4;
56     end
57     if no==5
58         Elems = Elm5;
59     end
60
61     [E,1] = size(Elems);
62     j_dbc=1;
63     j_nbc=1;
64     % Number of nodes per element
65     NE = 1-3;
66
67     % Read material info
68     Mats = load('Materials.txt');
69     [M,1] = size(Mats);
70
71     % Identify out-of-plane conditions
72     % ipstrn = 1    Plane strain

```

```

73 % ipstrn = 2    Plane stress
74 ipstrn = 2;
75 nstrn = 3;
76
77 %Determine Dirichlet BC
78 for (i=1:N)
79     if (Nodes(i,2)==0)
80         DBC(j_dbc,1)=Nodes(i,1);
81         DBC(j_dbc,2)=1;
82         DBC(j_dbc,3)=0;
83         j_dbc=j_dbc+1;
84     end
85     if (Nodes(i,3)==0)
86         DBC(j_dbc,1)=Nodes(i,1);
87         DBC(j_dbc,2)=2;
88         DBC(j_dbc,3)=0;
89         j_dbc=j_dbc+1;
90     end
91 end
92 [P,1] = size(DBC);
93
94 % Determine Neumann BC
95 for (i=1:N)
96     if (Nodes(i,2)==3)
97         right(j_nbc,1)=Nodes(i,1);
98         right(j_nbc,2)=1;
99         right(j_nbc,3)=0;
100        j_nbc=j_nbc+1;
101    end
102 end
103 j_nbc=1;
104
105
106 for i=1:E
107     for j=1:size(right(:,1))
108         for k=4:7
109
110             if Elems(i,k)==right(j,1)
111
112                 el_list(j_nbc,1)=Elems(i,1);
113                 el_list(j_nbc,2)=right(j,1);
114                 j_nbc=j_nbc+1;
115                 break
116             end
117         end
118     end
119 end
120 end
121 NBC(:,1)=unique(el_list(:,1));

```



```

122     for i=1:2:size(el_list(:,1))
123         for j=1:size(NBC(:,1))
124
125             k=0;
126             if (NBC(j,1)==el_list(i,1))
127                 NBC(j,2:3)=[el_list(i,2) el_list(i+1,2)];
128             end
129         end
130     end
131     NBC(:,4)=1;
132     NBC(:,5)=1;
133     [Q,1] = size(NBC);
134
135     % Determining the hole nodes
136     i_hol=1;
137     for i=1:N
138         if (Nodes(i,2)<=0.25 && Nodes(i,3)<=0.25)
139             hole(i_hol)=Nodes(i,1);
140             i_hol=i_hol+1;
141         end
142     end
143
144     % Determining the hole elements
145     i_hol=1;
146     for i=1:E
147         if (hole(i_hol)==Elems(i,4) || hole(i_hol)==Elems(i,5) || hole(i_hol)==
148             Elems(i,6) || hole(i_hol)==Elems(i,7))
149             hol_el(i_hol)=Elems(i,1);
150             i_hol=i_hol+1;
151         end
152     end
153     hol_el=unique(hol_el);
154     % Determine total number of degrees-of-freedom
155     udof = 2; % Degrees-of-freedom per node
156     NDOF = N*udof;
157
158     % Initialize global matrix and vectors
159     K = zeros(NDOF,NDOF); % Stiffness matrix
160     U = zeros(NDOF,1); % Displacement vector
161     F = zeros(NDOF,1); % Force vector
162
163     % Set penalty for displacement constraints
164     Klarge = 10^8;
165
166     % Set Gauss point locations and weights
167     NG = 4;
168     [XG,WG] = Q4_El_Gauss_Points(NG);
169

```

```

170 % Loop over Q4 elements
171 for e = 1:E
172
173     % Establish element connectivity and coordinates
174     Nnums = Elems(e,4:3+NE);
175     xy = Nodes(Nnums(:),2:3);
176
177     % Extract element thickness for plane stress
178     h = Elems(e,3);
179
180     % Extract element elastic Young's modulus and Poisson's ratio
181     Y = Mats(Elems(e,2),2);
182     nu = Mats(Elems(e,2),3);
183
184     % Construct element stiffness matrix
185     [Ke] = Q4_El_Stiff(ipstrn,xy,h,Y,nu,udof,NE,NG,XG,WG);
186
187     % Assemble element stiffness matrix into global stiffness matrix
188     ig = udof*(Nnums(:)-1);
189     for ni = 1:NE
190         i0 = udof*(ni-1);
191         for nj = 1:NE
192             j0 = udof*(nj-1);
193             for i = 1:udof
194                 for j = 1:udof
195                     K(ig(ni)+i,ig(nj)+j) = K(ig(ni)+i,ig(nj)+j) + Ke(i0+i,
196 j0+j);
197
198                 end
199             end
200         end
201     end
202
203     % Construct global force vector for loaded edges with constant traction
204     NES = 2;
205     % Set Gauss point locations and weights for traction integration
206     NGS = 2;
207     [XGS,WGS] = Q4_El_Gauss_Points_Surf(NGS);
208
209     for q = 1:Q
210
211         in = zeros(NES);
212         tval = zeros(NES,1);
213         fval = zeros(NES,1);
214
215         % Determine loaded edge
216         e = NBC(q,1);
217         in1 = NBC(q,2);

```

```

218     in2 = NBC(q,3);
219     idof = NBC(q,4);
220     tval(:,1) = NBC(q,4:5);
221     h = Elems(e,3);
222
223     for i=1:NGS
224
225         % Evaluate force contributions at Gauss points
226         xi = XGS(i);
227         wgt = WGS(i);
228
229         [NshapeS] = Q4_El_Shape_Surf(NES, xi);
230         [DNshapeS] = Q4_El_DShape_Surf(NES, xi);
231
232         xyS(1,1) = Nodes(in1,2);
233         xyS(1,2) = Nodes(in1,3);
234         xyS(2,1) = Nodes(in2,2);
235         xyS(2,2) = Nodes(in2,3);
236         [detJS] = Q4_El_Jacobian_Surf(NES, xi, xyS, DNshapeS);
237
238         fval = fval + h*wgt*NshapeS'*NshapeS*tval*detJS;
239
240     end
241     %fval
242
243     iloc1 = udof*(in1-1)+idof;
244     iloc2 = udof*(in2-1)+idof;
245     F(iloc1) = F(iloc1) + fval(1);
246     F(iloc2) = F(iloc2) + fval(2);
247     %F
248
249 end
250
251 % Impose Dirichlet boundary conditions
252 for p = 1:P
253     inode = DBC(p,1);
254     idof = DBC(p,2);
255     idiag = udof*(inode-1) + idof;
256     K(idiag, idiag) = Klarge;
257     F(idiag) = Klarge*DBC(p,3);
258 end
259 %K
260 %F
261
262 % Solve system to determine displacements
263 U = K\F;
264
265 % Recover internal element displacement, strains and stresses
266 nedof = udof*NE;

```

```

267 Disp = zeros(E, nedof);
268 Eps = zeros(E, nstrn, NG);
269 Sig = zeros(E, nstrn, NG);
270
271 for e = 1:E
272
273     % Establish element connectivity and coordinates
274     Nnums = Elems(e, 4:3+NE);
275     xy = Nodes(Nnums(:), 2:3);
276
277     % Extract element thickness for plane stress
278     h = Elems(e, 3);
279
280     % Extract element elastic Young's modulus and Poisson's ratio
281     Y = Mats(Elems(e, 2), 2);
282     nu = Mats(Elems(e, 2), 3);
283
284     % Extract element nodal displacements
285     for i=1:NE
286         inode = Nnums(i);
287         iglb1 = udof*(inode-1)+1;
288         iglb2 = udof*inode;
289         iloc1 = udof*(i-1)+1;
290         iloc2 = udof*i;
291         Disp(e, iloc1) = U(iglb1);
292         Disp(e, iloc2) = U(iglb2);
293     end
294     %Disp
295
296     u = Disp(e, :)';
297     [eps, sig] = Q4_El_Str(ipstrn, xy, u, h, Y, nu, udof, NE, NG, XG);
298     %eps
299     %sig
300
301     % Store element strains
302     Eps(e, :, :) = eps(:, :, :);
303
304     % Store element stresses
305     Sig(e, :, :) = sig(:, :, :);
306
307 end
308
309 % Computing Strain concentration factor
310
311 sig_nom= 1/0.75;
312
313 for i=1:size(hol_el)
314     sig_max=max(Sig(hol_el(i), :, :));
315 end

```

```

316     SCF(no)=mean(sig_max)/sig_nom;
317
318     PE(no,1)=0.5*U'*K*U;
319
320     PE(no,2)=3/N;
321     % if(no==1)
322     %     str=sprintf('Original plate vs deformed plate using 4 Noded Quad
elements for %d elements',E);
323     %     figure;
324     %     Plot_deformation;
325     %     title(str);
326     %     xlabel('\leftarrow 2L \rightarrow');
327     %     ylabel('\leftarrow 2H \rightarrow');
328     %     max(U)
329     % end
330     E
331     clearvars -except nod1 nod2 nod3 nod4 nod5 Elm1 Elm2 Elm3 Elm4 Elm5 PE SCF
;
332 end
333
334 for i=1:5
335     if (PE(i,2)==min(PE(:,2)))
336         PE_ex=PE(i,1);
337     end
338 end
339 PE(:,1)=abs(PE(:,1)-PE_ex)/abs(PE_ex);
340
341 % figure;
342 % plot(log(PE(:,2)),log(PE(:,1)), '-o');
343 % title('Error in Energy norm');
344 % xlabel('$\log(h)$', 'Interpreter','latex');
345 % ylabel('$\log(\frac{|U_{FE}-U_{EX}|}{|U_{EX}|})$', 'Interpreter','latex');
346 % axis square;
347 % Disp;
348 % Eps;
349 % Sig;

1 function [DNshape] = Q4_El_DShape(NE,xi,eta)
2
3 DNshape(1,1) = -(1-eta)/4;
4 DNshape(2,1) = +(1-eta)/4;
5 DNshape(3,1) = +(1+eta)/4;
6 DNshape(4,1) = -(1+eta)/4;
7
8 DNshape(1,2) = -(1-xi)/4;
9 DNshape(2,2) = -(1+xi)/4;
10 DNshape(3,2) = +(1+xi)/4;
11 DNshape(4,2) = +(1-xi)/4;

1 function [DNshapeS] = Q4_El_DShape_Surf(NES,xi)

```

```

2
3 DNshapeS(1) = -1/2;
4 DNshapeS(2) = +1/2;

1 function [XG,WG] = Q4_El_Gauss_Points(NG)
2
3 if (NG == 4)
4
5     alf = sqrt(1/3);
6
7     XG(1,1) = -alf;
8     XG(2,1) = +alf;
9     XG(3,1) = +alf;
10    XG(4,1) = -alf;
11
12    XG(1,2) = -alf;
13    XG(2,2) = -alf;
14    XG(3,2) = +alf;
15    XG(4,2) = +alf;
16
17    for i=1:NG
18        WG(i) = 1;
19    end
20
21 else
22
23     alf = sqrt(3/5);
24
25     XG(1,1) = -alf;
26     XG(2,1) = 0;
27     XG(3,1) = +alf;
28     XG(4,1) = -alf;
29     XG(5,1) = 0;
30     XG(6,1) = +alf;
31     XG(7,1) = -alf;
32     XG(8,1) = 0;
33     XG(9,1) = +alf;
34
35     XG(1,2) = -alf;
36     XG(2,2) = -alf;
37     XG(3,2) = -alf;
38     XG(4,2) = 0;
39     XG(5,2) = 0;
40     XG(6,2) = 0;
41     XG(7,2) = +alf;
42     XG(8,2) = +alf;
43     XG(9,2) = +alf;
44
45     WG(1) = 25/81;
46     WG(2) = 40/81;

```

```

47     WG(3) = 25/81;
48     WG(4) = 40/81;
49     WG(5) = 64/81;
50     WG(6) = 40/81;
51     WG(7) = 25/81;
52     WG(8) = 40/81;
53     WG(9) = 25/81;
54
55 end

1 function [XGS,WGS] = Q4_El_Gauss_Points_Surf(NGS)
2
3 if (NGS == 2)
4
5     alf = sqrt(1/3);
6
7     XGS(1,1) = -alf;
8     XGS(2,1) = +alf;
9
10    WGS(1) = 1;
11    WGS(2) = 1;
12
13 else
14
15    alf = sqrt(3/5);
16
17    XGS(1,1) = -alf;
18    XGS(2,1) = 0;
19    XGS(3,1) = +alf;
20
21    WGS(1) = 5/9;
22    WGS(2) = 8/9;
23    WGS(3) = 5/9;
24
25 end

1 function [Jac ,detJ ,Jhat] = Q4_El_Jacobian(NE,xi ,eta ,xy ,DNshape)
2
3 Jac = zeros(2,2);
4
5 for i=1:NE
6     Jac(1,1) = Jac(1,1) + DNshape(i,1)*xy(i,1);
7     Jac(1,2) = Jac(1,2) + DNshape(i,1)*xy(i,2);
8     Jac(2,1) = Jac(2,1) + DNshape(i,2)*xy(i,1);
9     Jac(2,2) = Jac(2,2) + DNshape(i,2)*xy(i,2);
10 end
11
12 detJ = det(Jac);
13 Jhat = inv(Jac);

```

```

1 function [detJS] = Q4_El_Jacobian_Surf(NES, xi , xyS , DNshapeS)
2
3 dxdxi = 0;
4 dydxi = 0;
5
6 for i=1:NES
7     dxdxi = dxdxi + DNshapeS(i)*xyS(i,1);
8     dydxi = dydxi + DNshapeS(i)*xyS(i,2);
9 end
10
11 detJS = sqrt( dxdxi*dxdxi + dydxi*dydxi );

1 function [Nshape] = Q4_El_Shape(NE, xi , eta)
2
3 Nshape(1) = (1-xi)*(1-eta)/4;
4 Nshape(2) = (1+xi)*(1-eta)/4;
5 Nshape(3) = (1+xi)*(1+eta)/4;
6 Nshape(4) = (1-xi)*(1+eta)/4;

1 function [NshapeS] = Q4_El_Shape_Surf(NES, xi)
2
3 NshapeS(1) = (1-xi)/2;
4 NshapeS(2) = (1+xi)/2;
5
6 %NshapeS = NshapeS';

1 function Ke = CST_El_Stiff(ipstrn ,xy ,h ,Y ,nu , udof ,NE,NG,XG,WG)
2
3 ndof = NE*udof;
4 nstrn = 3;
5 Ke = zeros(ndof , ndof);
6
7 for i=1:NG
8
9     xi = XG(i,1);
10    eta = XG(i,2);
11    wgt = WG(i);
12
13    %[Nshape] = Q4_El_Shape(NE, xi , eta);
14    [DNshape] = Q4_El_DShape(NE, xi , eta);
15    [Jac , detJ , Jhat] = Q4_El_Jacobian(NE, xi , eta , xy , DNshape);
16
17    B = zeros(nstrn , ndof);
18    for j=1:NE
19        jloc1 = 2*(j-1)+1;
20        jloc2 = jloc1 + 1;
21        B(1,jloc1) = B(1,jloc1) + Jhat(1,1)*DNshape(j,1) ...
22            + Jhat(1,2)*DNshape(j,2);
23        B(2,jloc2) = B(2,jloc2) + Jhat(2,1)*DNshape(j,1) ...
24            + Jhat(2,2)*DNshape(j,2);

```



```

25     B(3,jloc1) = B(3,jloc1) + Jhat(2,1)*DNshape(j,1) ...
26         + Jhat(2,2)*DNshape(j,2);
27     B(3,jloc2) = B(3,jloc2) + Jhat(1,1)*DNshape(j,1) ...
28         + Jhat(1,2)*DNshape(j,2);
29 end
30
31 if (ipstrn == 1)
32     c = Y*(1-nu)/(1-2*nu)/(1+nu);
33     C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];
34 else
35     c = Y/(1-nu)/(1+nu);
36     C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
37 end
38
39 Ke = Ke + h*wgt*B'*C*B*detJ;
40
41 end

1 function [eps, sig] = Q4_El_Str(ipstrn, xy, u, h, Y, nu, udof, NE, NG, XG);
2
3 ndof = NE*udof;
4 nstrn = 3;
5 eps = zeros(nstrn, NG);
6 sig = zeros(nstrn, NG);
7
8 for i=1:NG
9
10     xi = XG(i,1);
11     eta = XG(i,2);
12
13     [DNshape] = Q4_El_DShape(NE, xi, eta);
14     [Jac, detJ, Jhat] = Q4_El_Jacobian(NE, xi, eta, xy, DNshape);
15
16     B = zeros(nstrn, ndof);
17     for j=1:NE
18         jloc1 = 2*(j-1)+1;
19         jloc2 = jloc1 + 1;
20         B(1,jloc1) = B(1,jloc1) + Jhat(1,1)*DNshape(j,1) ...
21             + Jhat(1,2)*DNshape(j,2);
22         B(2,jloc2) = B(2,jloc2) + Jhat(2,1)*DNshape(j,1) ...
23             + Jhat(2,2)*DNshape(j,2);
24         B(3,jloc1) = B(3,jloc1) + Jhat(2,1)*DNshape(j,1) ...
25             + Jhat(2,2)*DNshape(j,2);
26         B(3,jloc2) = B(3,jloc2) + Jhat(1,1)*DNshape(j,1) ...
27             + Jhat(1,2)*DNshape(j,2);
28     end
29
30     if (ipstrn == 1)
31         c = Y*(1-nu)/(1-2*nu)/(1+nu);
32         C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];

```

```

33     else
34         c = Y/(1-nu)/(1+nu);
35         C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
36     end
37
38     eps(:, i) = B*u;
39     sig(:, i) = C*eps(:, i);
40
41 end

```

#### IV. Code for 4-noded quadrilateral isoparametric Element:

```

1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %                                                                                                                                                  %
3  % Elastic 8-node Quadrilateral Elements                                                                                                                                                   %
4  %                                                                                                                                                  %
5  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
6
7  % Clear workspace
8  clc
9  clear
10 close all
11 % Read nodes and coords
12 nod1= csvread('Nodes_1.csv');
13 nod2= csvread('Nodes_2.csv');
14 nod3= csvread('Nodes_3.csv');
15 nod4= csvread('Nodes_4.csv');
16 nod5= csvread('Nodes_5.csv');
17
18 Elm1=csvread('Elements_1.csv');
19 Elm2=csvread('Elements_2.csv');
20 Elm3=csvread('Elements_3.csv');
21 Elm4=csvread('Elements_4.csv');
22 Elm5=csvread('Elements_5.csv');
23
24 for no=1:5
25
26     % Read nodes and coords
27     if no==1
28         Nodes = nod1;
29     end
30     if no==2
31         Nodes = nod2;
32     end
33     if no==3
34         Nodes = nod3;
35     end
36     if no==4
37         Nodes = nod4;

```

```

38 end
39 if no==5
40     Nodes = nod5;
41 end
42 [N,1] = size(Nodes);
43
44 % Read element material id, thickness and nodal connectivity
45 if no==1
46     Elems = Elm1;
47 end
48 if no==2
49     Elems = Elm2;
50 end
51 if no==3
52     Elems = Elm3;
53 end
54 if no==4
55     Elems = Elm4;
56 end
57 if no==5
58     Elems = Elm5;
59 end
60 [E,1] = size(Elems);
61 j_dbc=1;
62 j_nbc=1;
63 % Number of nodes per element
64 NE = 1-3;
65
66 % Read material info
67 Mats = load('Materials.txt');
68 [M,1] = size(Mats);
69
70 % Identify out-of-plane conditions
71 % ipstrn = 1    Plane strain
72 % ipstrn = 2    Plane stress
73 ipstrn = 2;
74 nstrn = 3;
75
76 %Determine Derichlet BC
77 for (i=1:N)
78     if (Nodes(i,2)==0)
79         DBC(j_dbc,1)=Nodes(i,1);
80         DBC(j_dbc,2)=1;
81         DBC(j_dbc,3)=0;
82         j_dbc=j_dbc+1;
83     end
84     if (Nodes(i,3)==0)
85         DBC(j_dbc,1)=Nodes(i,1);
86         DBC(j_dbc,2)=2;

```

```

87         DBC(j_dbc ,3)=0;
88         j_dbc=j_dbc+1;
89     end
90 end
91 [P,1] = size(DBC);
92 % Determine Neumann BC
93 for (i=1:N)
94     if (Nodes(i ,2)==3)
95         right(j_nbc ,1)=Nodes(i ,1);
96         right(j_nbc ,2)=1;
97         right(j_nbc ,3)=0;
98         j_nbc=j_nbc+1;
99     end
100 end
101 j_nbc=1;
102 for i=1:E
103     for j=1:size(right (: ,1))
104         for k=4:11
105             if Elems(i ,k)==right(j ,1)
106                 el_list(j_nbc ,1)=Elems(i ,1);
107                 el_list(j_nbc ,2)=right(j ,1);
108                 j_nbc=j_nbc+1;
109                 break
110             end
111         end
112     end
113 end
114 NBC(:,1)=unique(el_list (: ,1));
115 j_nbc=1;
116 for i=1:3:size(el_list (: ,1))
117     for j=4:7
118         if (el_list(i ,2)==Elems(el_list(i ,1) ,j) || el_list(i+1,2)==Elems(
el_list(i ,1) ,j))
119             if (el_list(i ,2)==Elems(el_list(i ,1) ,j))
120                 NBC(j_nbc ,2)=el_list(i ,2);
121                 NBC(j_nbc ,4)=el_list(i+1,2);
122                 NBC(j_nbc ,3)=el_list(i+2,2);
123                 j_nbc=j_nbc+1;
124                 break;
125             else
126                 NBC(j_nbc ,2)=el_list(i+1,2);
127                 NBC(j_nbc ,4)=el_list(i ,2);
128                 NBC(j_nbc ,3)=el_list(i+2,2);
129                 j_nbc=j_nbc+1;
130                 break;
131             end
132         end
133     end
134 end

```

```

135
136 NBC(:,5)=1;
137 NBC(:,6)=1;
138 [Q,1] = size(NBC);
139
140
141 % Determining the hole nodes
142 i_hol=1;
143 for i=1:N
144     if (Nodes(i,2)<=0.25 && Nodes(i,3)<=0.25)
145         hole(i_hol)=Nodes(i,1);
146         i_hol=i_hol+1;
147     end
148 end
149
150 % Determining the hole elements
151 i_hol=1;
152 for i=1:E
153     if (hole(i_hol)==Elems(i,4) || hole(i_hol)==Elems(i,5) || hole(i_hol)==
154     Elems(i,6) || hole(i_hol)==Elems(i,7) || hole(i_hol)==Elems(i,8) || hole(i_hol)==
155     Elems(i,9) || hole(i_hol)==Elems(i,10) || hole(i_hol)==Elems(i,11))
156         hol_el(i_hol)=Elems(i,1);
157         i_hol=i_hol+1;
158     end
159 end
160 hol_el=unique(hol_el);
161
162 % Determine total number of degrees-of-freedom
163 udof = 2; % Degrees-of-freedom per node
164 NDOF = N*udof;
165
166 % Initialize global matrix and vectors
167 K = zeros(NDOF,NDOF); % Stiffness matrix
168 U = zeros(NDOF,1); % Displacement vector
169 F = zeros(NDOF,1); % Force vector
170
171 % Set penalty for displacement constraints
172 Klarge = 10^8;
173
174 % Set Gauss point locations and weights
175 NG = 4;
176 [XG,WG] = Q8_El_Gauss_Points(NG);
177
178 % Loop over Q8 elements
179 for e = 1:E
180
181     % Establish element connectivity and coordinates

```

```

182     Nnums = Elems(e,4:3+NE);
183     xy = Nodes(Nnums(:),2:3);
184
185     % Extract element thickness for plane stress
186     h = Elems(e,3);
187
188     % Extract element elastic Young's modulus and Poisson's ratio
189     Y = Mats(Elems(e,2),2);
190     nu = Mats(Elems(e,2),3);
191
192     % Construct element stiffness matrix
193     [Ke] = Q8_El_Stiff(ipstrn,xy,h,Y,nu,udof,NE,NG,XG,WG);
194
195     % Assemble element stiffness matrix into global stiffness matrix
196     ig = udof*(Nnums(:)-1);
197     for ni = 1:NE
198         i0 = udof*(ni-1);
199         for nj = 1:NE
200             j0 = udof*(nj-1);
201             for i = 1:udof
202                 for j = 1:udof
203                     K(ig(ni)+i,ig(nj)+j) = K(ig(ni)+i,ig(nj)+j) + Ke(i0+i,
204 j0+j);
205
206                 end
207             end
208         end
209     end
210
211     % Construct global force vector for loaded edges with constant traction
212     NES = 3;
213     % Set Gauss pint locations and weights for traction integration
214     NGS = 3;
215     [XGS,WGS] = Q8_El_Gauss_Points_Surf(NGS);
216
217     for q = 1:Q
218
219         in = zeros(NES);
220         tval = zeros(NES,1);
221         fval = zeros(NES,1);
222
223         % Determine loaded edge
224         e = NBC(q,1);
225         in1 = NBC(q,2);
226         in2 = NBC(q,3);
227         in3 = NBC(q,4);
228         idof = NBC(q,5);
229         tval(:,1) = NBC(q,6);

```

```

230     h = Elems(e,3);
231
232     for i=1:NGS
233
234         % Evaluate force contributions at Gauss points
235         xi = XGS(i);
236         wgt = WGS(i);
237
238         [NshapeS] = Q8_El_Shape_Surf(NES, xi);
239         [DNshapeS] = Q8_El_DShape_Surf(NES, xi);
240
241         xyS(1,1) = Nodes(in1,2);
242         xyS(1,2) = Nodes(in1,3);
243         xyS(2,1) = Nodes(in2,2);
244         xyS(2,2) = Nodes(in2,3);
245         xyS(3,1) = Nodes(in3,2);
246         xyS(3,2) = Nodes(in3,3);
247         [detJS] = Q8_El_Jacobian_Surf(NES, xi, xyS, DNshapeS);
248
249         fval = fval + h*wgt*NshapeS'*NshapeS*tval*detJS;
250
251     end
252     % fval
253
254     iloc1 = udof*(in1-1)+idof;
255     iloc2 = udof*(in2-1)+idof;
256     iloc3 = udof*(in3-1)+idof;
257     F(iloc1) = F(iloc1) + fval(1);
258     F(iloc2) = F(iloc2) + fval(2);
259     F(iloc3) = F(iloc3) + fval(3);
260     %F
261
262 end
263
264 % Impose Dirichlet boundary conditions
265 for p = 1:P
266     inode = DBC(p,1);
267     idof = DBC(p,2);
268     idiag = udof*(inode-1) + idof;
269     K(idiag, idiag) = Klarge;
270     F(idiag) = Klarge*DBC(p,3);
271 end
272 %K
273 %F
274
275 % Solve system to determine displacements
276 U = K\F;
277
278 % Recover internal element displacement, strains and stresses

```

```

279 nedof = udof*NE;
280 Disp = zeros(E, nedof);
281 Eps = zeros(E, nstrn, NG);
282 Sig = zeros(E, nstrn, NG);
283
284 for e = 1:E
285
286     % Establish element connectivity and coordinates
287     Nnums = Elems(e, 4:3+NE);
288     xy = Nodes(Nnums(:), 2:3);
289
290     % Extract element thickness for plane stress
291     h = Elems(e, 3);
292
293     % Extract element elastic Young's modulus and Poisson's ratio
294     Y = Mats(Elems(e, 2), 2);
295     nu = Mats(Elems(e, 2), 3);
296
297     % Extract element nodal displacements
298     for i=1:NE
299         inode = Nnums(i);
300         iglb1 = udof*(inode-1)+1;
301         iglb2 = udof*inode;
302         iloc1 = udof*(i-1)+1;
303         iloc2 = udof*i;
304         Disp(e, iloc1) = U(iglb1);
305         Disp(e, iloc2) = U(iglb2);
306     end
307     %Disp
308
309     u = Disp(e, :)';
310     [eps, sig] = Q8_El_Str(ipstrn, xy, u, h, Y, nu, udof, NE, NG, XG);
311     %eps
312     %sig
313
314     % Store element strains
315     Eps(e, :, :) = eps(:, :, :);
316
317     % Store element stresses
318     Sig(e, :, :) = sig(:, :, :);
319
320 end
321
322 % Computing Strain concentration factor
323
324 sig_nom= 1/0.75;
325
326 for i=1:size(hol_el)
327     sig_max=max(Sig(hol_el(i), :, :));

```



```

328     end
329     SCF(no)=mean(sig_max)/sig_nom;
330
331     PE(no,1)=0.5*U'*K*U;
332
333     PE(no,2)=3/N;
334     % if (no==1)
335     %     str=sprintf('Original plate vs deformed plate using 8 Noded Quad
elements for %d elements',E);
336     %     figure;
337     %     Plot_deformation;
338     %     title(str);
339     %     xlabel('\leftarrow 2L \rightarrow');
340     %     ylabel('\leftarrow 2H \rightarrow');
341     %     max(U)
342     % end
343     E
344     clearvars -except nod1 nod2 nod3 nod4 nod5 Elm1 Elm2 Elm3 Elm4 Elm5 PE SCF
;
345 end
346
347 for i=1:5
348     if (PE(i,2)==min(PE(:,2)))
349         PE_ex=PE(i,1);
350     end
351 end
352 PE(:,1)=abs(PE(:,1)-PE_ex)/abs(PE_ex);
353
354 % figure;
355 plot(log(PE(:,2)),log(PE(:,1)), '-o');
356 title('Error in Energy norm');
357 xlabel('$\log(h)$', 'Interpreter', 'latex');
358 ylabel('$\log(\frac{|U_{FE}-U_{EX}|}{|U_{EX}|})$', 'Interpreter', 'latex'); axis
square;
359 % Disp
360 % Eps
361 % Sig

1 function [DNshape] = Q8_El_DSshape(NE,xi,eta)
2
3
4 DNshape(:,1)=[-(xi/4 - 1/4)*(eta - 1) - ((eta - 1)*(eta + xi + 1))/4;
5              ((eta - 1)*(eta - xi + 1))/4 - (xi/4 + 1/4)*(eta - 1);
6              (xi/4 + 1/4)*(eta + 1) + ((eta + 1)*(eta + xi - 1))/4;
7              (xi/4 - 1/4)*(eta + 1) + ((eta + 1)*(xi - eta + 1))/4;
8              xi*(eta - 1);
9              1/2 - eta^2/2;
10             -xi*(eta + 1);
11             eta^2/2 - 1/2]';
12

```

```

13
14 DNshape(:,2)=[- (xi/4 - 1/4)*(eta - 1) - (xi/4 - 1/4)*(eta + xi + 1);
15                (xi/4 + 1/4)*(eta - xi + 1) + (xi/4 + 1/4)*(eta - 1);
16                (xi/4 + 1/4)*(eta + 1) + (xi/4 + 1/4)*(eta + xi - 1);
17                (xi/4 - 1/4)*(xi - eta + 1) - (xi/4 - 1/4)*(eta + 1);
18                xi^2/2 - 1/2;
19                -eta*(xi + 1);
20                1/2 - xi^2/2;
21                eta*(xi - 1)]';

1 function [DNshapeS] = Q8_El_DShape_Surf(NES, xi)
2
3 DNshapeS(1) = xi - 1/2;
4 DNshapeS(2) = -2*xi;
5 DNshapeS(3) = -xi - 1/2;

1 function [XG,WG] = Q8_El_Gauss_Points(NG)
2
3 if (NG == 4)
4
5     alf = sqrt(1/3);
6
7     XG(1,1) = -alf;
8     XG(2,1) = +alf;
9     XG(3,1) = +alf;
10    XG(4,1) = -alf;
11
12    XG(1,2) = -alf;
13    XG(2,2) = -alf;
14    XG(3,2) = +alf;
15    XG(4,2) = +alf;
16
17    for i=1:NG
18        WG(i) = 1;
19    end
20
21 else
22
23     alf = sqrt(3/5);
24
25     XG(1,1) = -alf;
26     XG(2,1) = 0;
27     XG(3,1) = +alf;
28     XG(4,1) = -alf;
29     XG(5,1) = 0;
30     XG(6,1) = +alf;
31     XG(7,1) = -alf;
32     XG(8,1) = 0;
33     XG(9,1) = +alf;
34

```

```

35     XG(1,2) = -alf;
36     XG(2,2) = -alf;
37     XG(3,2) = -alf;
38     XG(4,2) = 0;
39     XG(5,2) = 0;
40     XG(6,2) = 0;
41     XG(7,2) = +alf;
42     XG(8,2) = +alf;
43     XG(9,2) = +alf;
44
45     WG(1) = 25/81;
46     WG(2) = 40/81;
47     WG(3) = 25/81;
48     WG(4) = 40/81;
49     WG(5) = 64/81;
50     WG(6) = 40/81;
51     WG(7) = 25/81;
52     WG(8) = 40/81;
53     WG(9) = 25/81;
54
55 end

1 function [XGS,WGS] = Q8_El_Gauss_Points_Surf(NGS)
2
3 if (NGS == 2)
4
5     alf = sqrt(1/3);
6
7     XGS(1,1) = -alf;
8     XGS(2,1) = +alf;
9
10    WGS(1) = 1;
11    WGS(2) = 1;
12
13 else
14
15     alf = sqrt(3/5);
16
17     XGS(1,1) = -alf;
18     XGS(2,1) = 0;
19     XGS(3,1) = +alf;
20
21     WGS(1) = 5/9;
22     WGS(2) = 8/9;
23     WGS(3) = 5/9;
24
25 end

1 function [Jac , detJ , Jhat ] = Q8_El_Jacobian(NE, xi , eta , xy , DNshape)
2

```

```

3 Jac = zeros(2,2);
4
5 for i=1:NE
6     Jac(1,1) = Jac(1,1) + DNshape(i,1)*xy(i,1);
7     Jac(1,2) = Jac(1,2) + DNshape(i,1)*xy(i,2);
8     Jac(2,1) = Jac(2,1) + DNshape(i,2)*xy(i,1);
9     Jac(2,2) = Jac(2,2) + DNshape(i,2)*xy(i,2);
10 end
11
12 detJ = det(Jac);
13 Jhat = inv(Jac);

1 function [detJS] = Q8_El_Jacobian_Surf(NES, xi, xyS, DNshapeS)
2
3 dxdxi = 0;
4 dydxi = 0;
5
6 for i=1:NES
7     dxdxi = dxdxi + DNshapeS(i)*xyS(i,1);
8     dydxi = dydxi + DNshapeS(i)*xyS(i,2);
9 end
10
11 detJS = sqrt(dxdxi*dxdxi + dydxi*dydxi);

1 function [Nshape] = Q8_El_Shape(NE, xi, eta)
2
3
4 Nshape = [-1/4*(1-xi)*(1-eta)*(xi+eta+1);
5           1/4*(1+xi)*(1-eta)*(xi-eta-1);
6           1/4*(1+xi)*(1+eta)*(xi+eta-1);
7           -1/4*(1-xi)*(1+eta)*(xi-eta+1);
8           1/2*(1-xi^2)*(1-eta);
9           1/2*(1-eta^2)*(1+xi);
10          1/2*(1-xi^2)*(1+eta);
11          1/2*(1-eta^2)*(1-xi) ]';

1 function [NshapeS] = Q8_El_Shape_Surf(NES, xi)
2
3 NshapeS(1) = ((xi-0)*(xi-1))/((-1-0)*(-1-1));
4 NshapeS(2) = ((xi+1)*(xi-1))/((0+1)*(0-1));
5 NshapeS(3) = ((xi+1)*(xi-0))/((1+1)*(0-1));
6
7 %NshapeS = NshapeS';

1 function [Ke] = CST_El_Stiff(ipstrn, xy, h, Y, nu, udof, NE, NG, XG, WG)
2
3 ndof = NE*udof;
4 nstrn = 3;
5 Ke = zeros(ndof, ndof);
6

```

```

7 for i=1:NG
8
9     xi = XG(i,1);
10    eta = XG(i,2);
11    wgt = WG(i);
12
13    %[Nshape] = Q8_El_Shape(NE,xi,eta);
14    [DNshape] = Q8_El_DShape(NE,xi,eta);
15    [Jac,detJ,Jhat] = Q8_El_Jacobian(NE,xi,eta,xy, DNshape);
16
17    B = zeros(nstrn,ndof);
18    for j=1:NE
19        jloc1 = 2*(j-1)+1;
20        jloc2 = jloc1 + 1;
21        B(1,jloc1) = B(1,jloc1) + Jhat(1,1)*DNshape(j,1) ...
22            + Jhat(1,2)*DNshape(j,2);
23        B(2,jloc2) = B(2,jloc2) + Jhat(2,1)*DNshape(j,1) ...
24            + Jhat(2,2)*DNshape(j,2);
25        B(3,jloc1) = B(3,jloc1) + Jhat(2,1)*DNshape(j,1) ...
26            + Jhat(2,2)*DNshape(j,2);
27        B(3,jloc2) = B(3,jloc2) + Jhat(1,1)*DNshape(j,1) ...
28            + Jhat(1,2)*DNshape(j,2);
29    end
30
31    if (ipstrn == 1)
32        c = Y*(1-nu)/(1-2*nu)/(1+nu);
33        C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];
34    else
35        c = Y/(1-nu)/(1+nu);
36        C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
37    end
38
39    Ke = Ke + h*wgt*B'*C*B*detJ;
40
41 end

1 function [eps,sig] = Q8_El_Str(ipstrn,xy,u,h,Y,nu,udof,NE,NG,XG);
2
3 ndof = NE*udof;
4 nstrn = 3;
5 eps = zeros(nstrn,NG);
6 sig = zeros(nstrn,NG);
7
8 for i=1:NG
9
10    xi = XG(i,1);
11    eta = XG(i,2);
12
13    [DNshape] = Q8_El_DShape(NE,xi,eta);
14    [Jac,detJ,Jhat] = Q8_El_Jacobian(NE,xi,eta,xy, DNshape);

```

```

15
16 B = zeros(nstrn, ndof);
17 for j=1:NE
18     jloc1 = 2*(j-1)+1;
19     jloc2 = jloc1 + 1;
20     B(1, jloc1) = B(1, jloc1) + Jhat(1,1)*DNshape(j,1) ...
21         + Jhat(1,2)*DNshape(j,2);
22     B(2, jloc2) = B(2, jloc2) + Jhat(2,1)*DNshape(j,1) ...
23         + Jhat(2,2)*DNshape(j,2);
24     B(3, jloc1) = B(3, jloc1) + Jhat(2,1)*DNshape(j,1) ...
25         + Jhat(2,2)*DNshape(j,2);
26     B(3, jloc2) = B(3, jloc2) + Jhat(1,1)*DNshape(j,1) ...
27         + Jhat(1,2)*DNshape(j,2);
28 end
29
30 if (ipstrn == 1)
31     c = Y*(1-nu)/(1-2*nu)/(1+nu);
32     C = c*[ 1 nu/(1-nu) 0; nu/(1-nu) 1 0; 0 0 (1-2*nu)/(1-nu)/2 ];
33 else
34     c = Y/(1-nu)/(1+nu);
35     C = c*[ 1 nu 0; nu 1 0; 0 0 (1-nu)/2 ];
36 end
37
38 eps(:, i) = B*u;
39 sig(:, i) = C*eps(:, i);
40
41 end

```