2D Four-Node Isoparametric Element for Plane Stress Problems

Zhang Baiming, 3230100298

2025 年 6 月 29 日

Finite Element Method Project No.2

Question

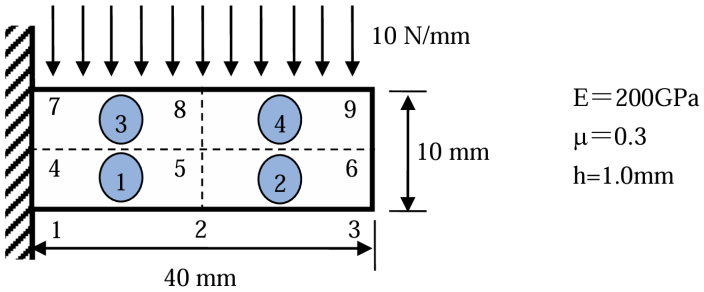
Using what we have learned about the 2D 4-node element isoparametric formulation to derive the FEM equilibrium equation to solve the plane stress problem shown below. You need to do the following:

(1) use a program that you are comfortable with (matlab, mathematica, fortran or C) to assemble the 4 element stiffness matrices into global matrix, establish the external force array corresponding to the global matrix, apply displacement constraints to node 1,4,7, then solve the displacements for other nodes.

(2) after you solve the displacements, calculate and report the stresses in element 3 at the four Gaussian integration points.

(3) Use abaqus to create the same model and solve it, compare the displacement of your solution to abaqus solution for nodal displacements at node 6.

(4) Use abaqus to solve the problem with refined meshes and report the convergence study.



Answer

1 Purpose and Scope

In this project, we delve into the analysis of the plane stress problem in thin plates. Our approach involves employing the two-dimensional four-node element isoparametric formulation. This method, coupled with the Gaussian integral technique, allows us to derive the stiffness matrix for each unit. After establishing the boundary conditions and applying loads, we utilize Matlab to compute the displacement at each node.

Furthermore, we conduct simulations using Abaqus, a robust finite element analysis software. The outcomes from these simulations align with those obtained from Matlab, thereby validating the accuracy of employing the isoparametric formulation to address the problem at hand. Also, we employ Abaqus to refine the model mesh, aiming to ascertain the convergence of displacement values. This step is crucial for ensuring the reliability and precision of our results.

2 Results and Discussion

2.1 Displacements of nods calculated by both Matlab and Abaqus

The Displacements calculation results of Matlab & Abaqus are shown in Table 1, which is exactly the same.

Table 1: Displacements calculation results

	horizontal (mm)		vertical (mm)	
point	Matlab	Abaqus	Matlab	Abaqus
2	-1.15×10^{-2}	-1.15×10^{-2}	-3.10×10^{-2}	-3.10×10^{-2}
3	-1.38×10^{-2}	-1.38×10^{-2}	-8.45×10^{-2}	-8.45×10^{-2}
5	8.66×10^{-5}	8.66×10^{-5}	-3.07×10^{-2}	-3.07×10^{-2}
6	2.60×10^{-4}	2.60×10^{-4}	-8.46×10^{-2}	-8.46×10^{-2}
8	1.17×10^{-2}	1.17×10^{-2}	-3.13×10^{-2}	-3.13×10^{-2}
9	1.43×10^{-2}	1.43×10^{-2}	-8.47×10^{-2}	-8.47×10^{-2}

2.2 Stress of the four Gaussian integration points of unit 3

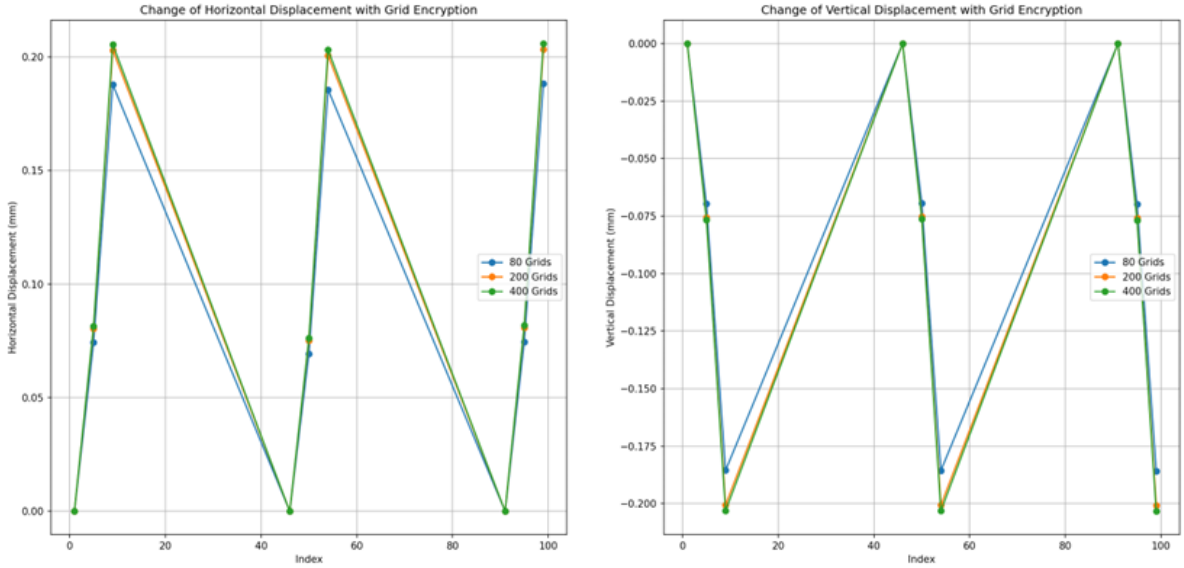
Stress of the four Gaussian integration points of unit 3 is shown in the Table 2 below.

Table 2: Stress of the Gaussian points

Gaussian Points	$\sigma_{11}(N/mm^2)$	$\sigma_{22}(N/mm^2)$	$\sigma_{12}(N/mm^2)$
1	0.6549	-20.6698	-21.2794
2	16.0297	-16.0574	-20.0859
3	4.7469	-7.0298	0.2453
4	20.1217	-2.4174	1.4388

2.3 Refining meshes and the convergence study

The simulation results of displacement converge with the gradual encryption of the grid, especially when the number of grids is greater than 200, as is shown in Figure downward.



Displacements' comparison of different grids

3 Conclusions and Recommendations

In this study, we have employed Matlab computations in conjunction with Abaqus simulations to ascertain the precision of the two-dimensional four-node isoparametric formulation and the Gaussian integration method for calculating displacements. Furthermore, we have examined the trend that indicates the gradual convergence of nodal displacement values as the grid count increases.

Due to constraints in time and other factors, we have not endeavored to utilize formulas of higher precision for simulation calculations. Utilizing the two-dimensional four-node isoparametric formulation, we can conduct a preliminary analysis of certain finite element problems. In practical applications, enhancing computational accuracy can be achieved through the method of grid refinement, thereby making significant contributions to engineering practice.

4 Appendices

4.1 Formula derivation

For 4 node isoparametric 2D elements, the isoparametric formula is given by:

$$N_1 = \frac{1}{4}(1-s)(1-t)$$
$$N_2 = \frac{1}{4}(1+s)(1-t)$$
$$N_3 = \frac{1}{4}(1+s)(1+t)$$
$$N_4 = \frac{1}{4}(1-s)(1+t)$$

The Jacobian matrix is defined as:

$$J = \begin{bmatrix} \frac{\partial x}{\partial s} & \frac{\partial y}{\partial s} \\ \frac{\partial x}{\partial t} & \frac{\partial y}{\partial t} \end{bmatrix}$$

where

$$\frac{\partial x}{\partial s} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial s} x_i$$
$$\frac{\partial y}{\partial s} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial s} y_i$$
$$\frac{\partial x}{\partial t} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial t} x_i$$
$$\frac{\partial y}{\partial t} = \sum_{i=1}^{4} \frac{\partial N_i}{\partial t} y_i$$

Thus,

$$\begin{bmatrix} \frac{\partial N_i}{\partial x} \\ \frac{\partial N_i}{\partial y} \end{bmatrix} = J^{-1} \begin{bmatrix} \frac{\partial N_i}{\partial s} \\ \frac{\partial N_i}{\partial t} \end{bmatrix}$$

The horizontal and vertical displacements are:

$$u = \sum_{i=1}^{4} N_i(s,t) u_i$$
$$v = \sum_{i=1}^{4} N_i(s,t) v_i$$

The strain matrix ε is:

$$\varepsilon = \begin{bmatrix} \frac{\partial}{\partial x} & 0 \\ 0 & \frac{\partial}{\partial y} \\ \frac{\partial}{\partial y} & \frac{\partial}{\partial x} \end{bmatrix} \times \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{\partial N_1}{\partial x} & 0 & \frac{\partial N_2}{\partial x} & 0 & \frac{\partial N_3}{\partial x} & 0 & \frac{\partial N_4}{\partial x} & 0 \\ 0 & \frac{\partial N_1}{\partial y} & 0 & \frac{\partial N_2}{\partial y} & 0 & \frac{\partial N_3}{\partial y} & 0 & \frac{\partial N_4}{\partial y} \\ \frac{\partial N_1}{\partial y} & \frac{\partial N_1}{\partial x} & \frac{\partial N_2}{\partial y} & \frac{\partial N_2}{\partial x} & \frac{\partial N_3}{\partial y} & \frac{\partial N_3}{\partial x} & \frac{\partial N_4}{\partial y} & \frac{\partial N_4}{\partial x} \end{bmatrix} \times \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix}$$

Thus, $\varepsilon = Bd$, where B is the strain-displacement matrix.

The material stiffness matrix D is:

$$D = \frac{E}{1-\mu^2} \times \begin{bmatrix} 1 & \mu & 0 \\ \mu & 1 & 0 \\ 0 & 0 & \frac{1-\mu}{2} \end{bmatrix}$$

where E is Young's modulus and μ is Poisson's ratio.

The element stiffness matrix is:

$$k_e = \int B^T D B \, dV$$

The stress in each unit is:

$$\sigma = DBu$$

4.2 Matlab program and the result

```
clear; clc;
format short;

% Material Properties
E = 200e3;    % Young's Modulus (N/mm^2)
nu = 0.3;     % Poisson's Ratio
h = 1;        % Thickness (mm)

% Elasticity Matrix D
D = E/(1-nu^2) * [1, nu, 0;
                  nu, 1, 0;
                  0, 0, (1-nu)/2];

% Node Coordinates
x = [0, 20, 40, 0, 20, 40, 0, 20, 40];
y = [0, 0, 0, 5, 5, 5, 10, 10, 10];
nodes = {[1, 2, 5, 4],[2, 3, 6, 5],[5, 6, 9, 8],[4, 5, 8, 7]};

% Gaussian Integration Parameters
gauss_points = [-1/sqrt(3), 1/sqrt(3)];
weights = [1, 1];

% Initialize Element Stiffness Matrices
Ke1 = zeros(8, 8);
Ke2 = zeros(8, 8);
Ke3 = zeros(8, 8);
Ke4 = zeros(8, 8);

% Compute Stiffness Matrices
```

```
for m = 1:4
    % Gaussian Quadrature
    for i = 1:length(gauss_points)
        s = gauss_points(i);
        for j = 1:length(gauss_points)
            t = gauss_points(j);
            % Current Weight
            w = weights(i) * weights(j);

            % Derivatives of Shape Functions with respect to s, t
            dN_ds = 0.25 * [-(1 - t), (1 - t), (1 + t), -(1 + t)];
            dN_dt = 0.25 * [-(1 - s), -(1 + s), (1 + s), (1 - s)];

            % Compute Jacobian Matrix
            dx_ds = dN_ds * x(nodes{m})';
            dy_ds = dN_ds * y(nodes{m})';
            dx_dt = dN_dt * x(nodes{m})';
            dy_dt = dN_dt * y(nodes{m})';
            J = [dx_ds, dy_ds;
                 dx_dt, dy_dt];
            detJ = det(J);
            invJ = inv(J);

            % Derivatives of Shape Functions with respect to x, y
            dN_dx = dN_ds * invJ(1,1) + dN_dt * invJ(2,1);
            dN_dy = dN_ds * invJ(1,2) + dN_dt * invJ(2,2);

            % Assemble Strain-Displacement Matrix B
            B = zeros(3, 8);
            for k = 1:4
                B(1, 2*k-1) = dN_dx(k);
                B(2, 2*k)   = dN_dy(k);
                B(3, 2*k-1) = dN_dy(k);
                B(3, 2*k)   = dN_dx(k);
            end

            % Stiffness Matrix Integration
            switch m
                case 1
                    Ke1 = Ke1 + (B' * D * B) * h * abs(detJ) * w;
                case 2
                    Ke2 = Ke2 + (B' * D * B) * h * abs(detJ) * w;
```

```
                case 3
                    Ke3 = Ke3 + (B' * D * B) * h * abs(detJ) * w;
                case 4
                    Ke4 = Ke4 + (B' * D * B) * h * abs(detJ) * w;
            end
        end
    end
end

% Assemble Stiffness Matrix
% Initialize Stiffness Matrix
Ke = zeros(18);

for i = 1:4
    points = [];

    for node = nodes{i}
        points = [points, (node-1)*2 + 1, (node-1)*2 + 2];
    end

    switch i
        case 1
            K_element = Ke1;
        case 2
            K_element = Ke2;
        case 3
            K_element = Ke3;
        case 4
            K_element = Ke4;
    end

    Ke(points, points) = Ke(points, points) + K_element;
end

% Solve
% Displacement Matrix
u = zeros(18,1);

% Force Matrix
f = zeros(18,1);
f(14) = -100;
f(16) = -200;
```

```
f(18) = -100;

% Fixed Degrees of Freedom
fixed_points = [1, 2, 7, 8, 13, 14];
free_points = setdiff(1:18, fixed_points);

% Row and Column Reduction Method to Solve
Ke_new = Ke(free_points, free_points);
f_new = f(free_points);

u_new = Ke_new \ f_new;

% Displacement Results
u(free_points) = u_new;

% Display Results
disp("Displacement Matrix u=");
disp(u);

% Stress at Gaussian Points of Element 3
% Extract Displacement Vector for Element 3
u3 = [];
for node = nodes{3}
    u3 = [u3; u(2*node-1); u(2*node)];
end

% Initialize Stress Matrix
stress = zeros(3, 4);

% Compute Stress at Each Gaussian Point
m = 1;
for i = 1:2
    s = gauss_points(i);
    for j = 1:2
        t = gauss_points(j);

        % Derivatives of Shape Functions with respect to s, t
        dN_ds = 0.25 * [-(1 - t), (1 - t), (1 + t), -(1 + t)];
        dN_dt = 0.25 * [-(1 - s), -(1 + s), (1 + s), (1 - s)];

        % Jacobian Matrix
        dx_ds = dN_ds * x(nodes{3})';
```

```
156         dy_ds = dN_ds * y(nodes{3})';
157         dx_dt = dN_dt * x(nodes{3})';
158         dy_dt = dN_dt * y(nodes{3})';
159         J = [dx_ds, dy_ds; dx_dt, dy_dt];
160         invJ = inv(J);
161 
162         % Derivatives of Shape Functions with respect to x, y
163         dN_dx = dN_ds * invJ(1,1) + dN_dt * invJ(2,1);
164         dN_dy = dN_ds * invJ(1,2) + dN_dt * invJ(2,2);
165 
166         % Assemble B Matrix
167         B = zeros(3, 8);
168         for k = 1:4
169             B(1, 2*k-1) = dN_dx(k);
170             B(2, 2*k)   = dN_dy(k);
171             B(3, 2*k-1) = dN_dy(k);
172             B(3, 2*k)   = dN_dx(k);
173         end
174 
175         % Compute Stress
176         stress(:, m) = D * B * u3;
177         m = m + 1;
178     end
179 end
180 
181 % Output Results
182 disp('Stress at four Gaussian integration points of Element 3 σ(11,
        σ22, σ12, N/mm^2):');
183 for i = 1:4
184     fprintf('Integration Point %d: [%.4f, %.4f, %.4f]\n', i, stress
            (1, i), stress(2, i), stress(3, i));
185 end
```

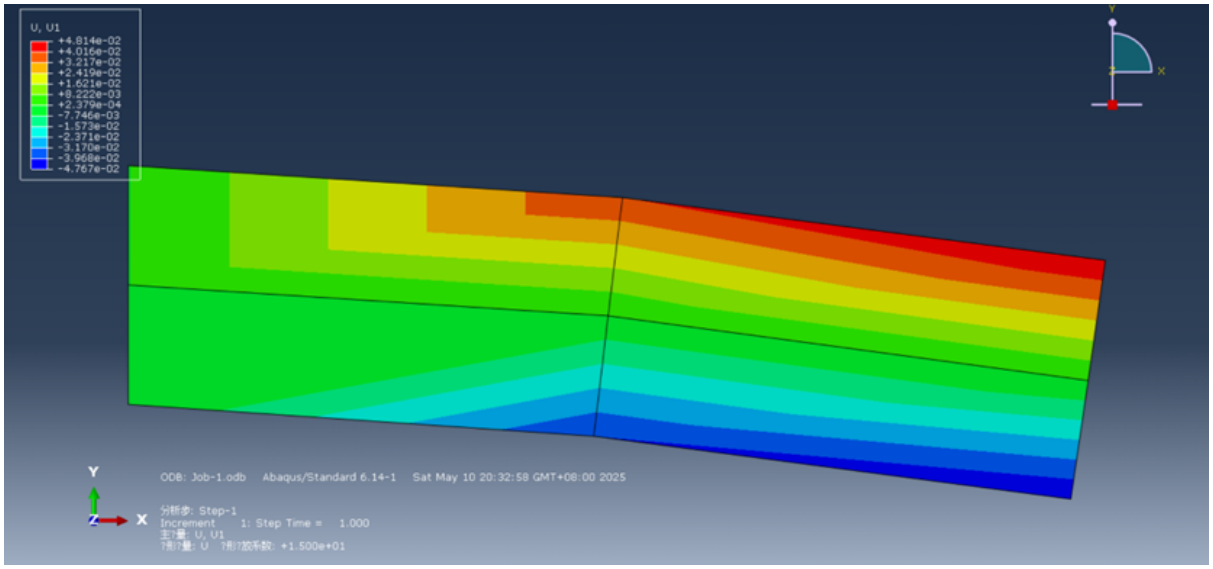
Matlab code

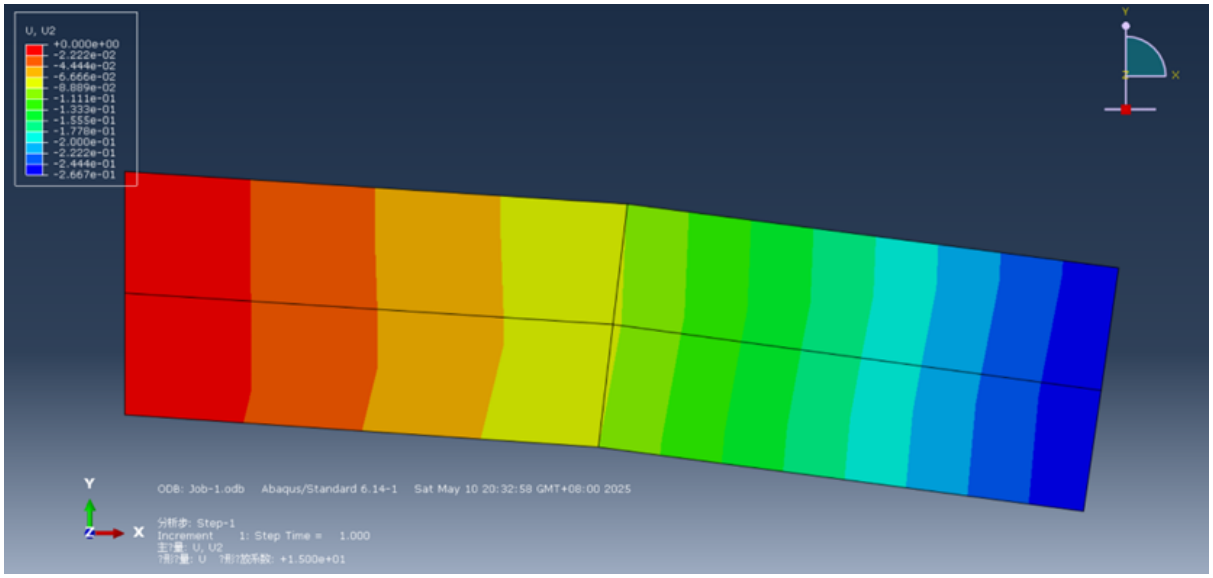
```
命令行窗口
Displacement Matrix u=
         0
         0
   -0.0115
   -0.0310
   -0.0138
   -0.0845
         0
         0
    0.0001
   -0.0307
    0.0003
   -0.0846
         0
         0
    0.0117
   -0.0313
    0.0143
   -0.0847

Stress at four Gaussian integration points of Element 3 (σ11, σ22, σ12, N/mm^2):
Integration Point 1: [0.6549, -20.6698, -21.2794]
Integration Point 2: [16.0297, -16.0574, -20.0859]
Integration Point 3: [4.7469, -7.0298, 0.2453]
Integration Point 4: [20.1217, -2.4174, 1.4388]
```

Result from Matlab

4.3 Visualization of Abaqus model





4.4 Displacements of nods calculated by Abaqus

Point2

```
基本坐标:                    2.00000e+01,  0.00000e+00,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    1.99885e+01, -3.09950e-02,  0.00000e+00,      -
变形后的坐标 (已缩放):      1.94592e+01, -1.46316e+00,  0.00000e+00,      -
位移 (未缩放):           -1.14555e-02, -3.09950e-02,  0.00000e+00,  3.30441e-02
```

Point3

```
基本坐标:                    4.00000e+01,  0.00000e+00,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    3.99862e+01, -8.45105e-02,  0.00000e+00,      -
变形后的坐标 (已缩放):      3.93505e+01, -3.98943e+00,  0.00000e+00,      -
位移 (未缩放):           -1.37581e-02, -8.45105e-02,  0.00000e+00,  8.56230e-02
```

Point5

```
基本坐标:                    2.00000e+01,  5.00000e+00,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    2.00001e+01,  4.96933e+00,  0.00000e+00,      -
变形后的坐标 (已缩放):      2.00041e+01,  3.55218e+00,  0.00000e+00,      -
位移 (未缩放):            8.66034e-05, -3.06700e-02,  0.00000e+00,  3.06701e-02
```

Point6

```
基本坐标:                    4.00000e+01,  5.00000e+00,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    4.00003e+01,  4.91536e+00,  0.00000e+00,      -
变形后的坐标 (已缩放):      4.00123e+01,  1.00462e+00,  0.00000e+00,      -
位移 (未缩放):            2.60077e-04, -8.46367e-02,  0.00000e+00,  8.46371e-02
```

Point8

```
基本坐标:                    2.00000e+01,  1.00000e+01,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    2.00117e+01,  9.96869e+00,  0.00000e+00,      -
变形后的坐标 (已缩放):      2.05502e+01,  8.52219e+00,  0.00000e+00,      -
位移 (未缩放):            1.16546e-02, -3.13053e-02,  0.00000e+00,  3.34043e-02
```

Point9

```
基本坐标:                    4.00000e+01,  1.00000e+01,  0.00000e+00,      -
缩放:                              4.72063e+01,  4.72063e+01,  4.72063e+01,      -
变形后的坐标 (未缩放):    4.00143e+01,  9.91527e+00,  0.00000e+00,      -
变形后的坐标 (已缩放):      4.06728e+01,  6.00000e+00,  0.00000e+00,      -
位移 (未缩放):            1.42514e-02, -8.47345e-02,  0.00000e+00,  8.59246e-02
```

4.5 Displacements of Refined meshes

Calculated Results by 80 Grids

Index	$\sigma_{11}(N/mm^2)$	$\sigma_{22}(N/mm^2)$	$\sigma_{12}(N/mm^2)$
1	0	-341.954×10^{-36}	-191.952×10^{-36}
5	74.2143×10^{-3}	-25.5194×10^{-3}	-69.6887×10^{-3}
9	187.829×10^{-3}	-29.1347×10^{-3}	-185.555×10^{-3}
46	0	-1.11183×10^{-36}	-2.82606×10^{-36}
50	69.3207×10^{-3}	134.031×10^{-6}	-69.3205×10^{-3}
54	185.592×10^{-3}	283.921×10^{-6}	-185.592×10^{-3}
91	0	349.553×10^{-36}	-184.615×10^{-36}
95	74.5410×10^{-3}	25.7872×10^{-3}	-69.9334×10^{-3}
99	188.165×10^{-3}	29.7026×10^{-3}	-185.805×10^{-3}

Calculated Results by 200 Grids

Index	$\sigma_{11}(N/mm^2)$	$\sigma_{22}(N/mm^2)$	$\sigma_{12}(N/mm^2)$
1	0	-290.477×10^{-36}	-110.540×10^{-36}
11	80.4790×10^{-3}	-27.5926×10^{-3}	-75.6011×10^{-3}
21	203.018×10^{-3}	-31.3795×10^{-3}	-200.578×10^{-3}
106	0	-1.11875×10^{-36}	-12.4718×10^{-36}
116	75.2035×10^{-3}	138.144×10^{-6}	-75.2034×10^{-3}
126	200.612×10^{-3}	288.122×10^{-6}	-200.612×10^{-3}
211	0	297.191×10^{-36}	-110.212×10^{-36}
221	80.8087×10^{-3}	27.8688×10^{-3}	-75.8510×10^{-3}
231	203.355×10^{-3}	31.9557×10^{-3}	-200.828×10^{-3}

Calculated Results by 400 Grids

Index	$\sigma_{11}(N/mm^2)$	$\sigma_{22}(N/mm^2)$	$\sigma_{12}(N/mm^2)$
1	0	-280.703×10^{-36}	-82.5862×10^{-36}
21	81.5330×10^{-3}	-27.9334×10^{-3}	-76.5987×10^{-3}
41	205.535×10^{-3}	-31.7473×10^{-3}	-203.068×10^{-3}
206	0	-1.11209×10^{-36}	-14.1039×10^{-36}
226	76.1965×10^{-3}	139.023×10^{-6}	-76.1964×10^{-3}
246	203.100×10^{-3}	289.007×10^{-6}	-203.100×10^{-3}
411	0	287.080×10^{-36}	-84.9477×10^{-36}
431	81.8633×10^{-3}	28.2114×10^{-3}	-76.8487×10^{-3}
451	205.872×10^{-3}	32.3254×10^{-3}	-203.318×10^{-3}