REPORT ON

DEVELOPMENT OF HIGHER ORDER FINITE ELEMENT.

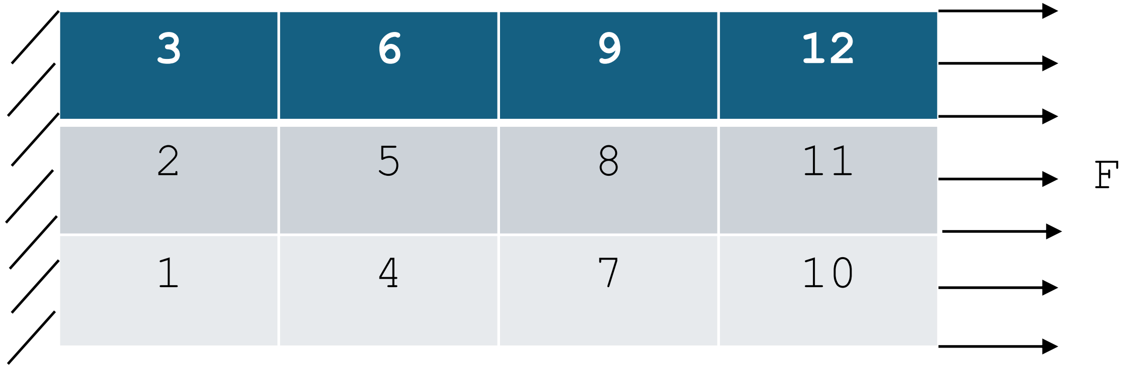
PRESENTED BY

NAME-BARUN KUMAR
ROLL-23MT0095
M.TECH(DESIGN)
DEPT.-MECHANICAL ENGINEERING
IIT(ISM), DHANBAD

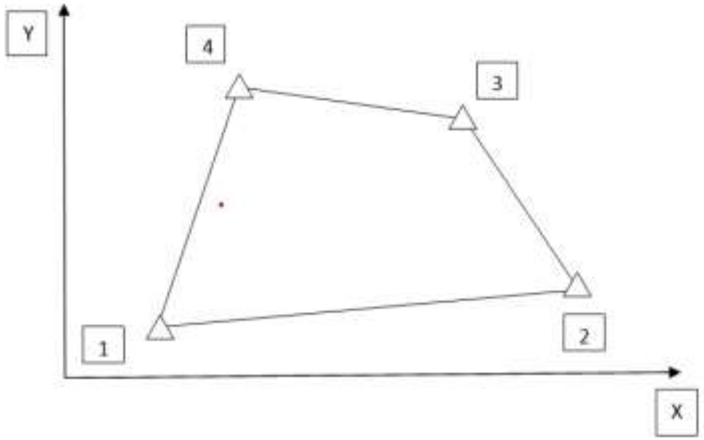
INDEX

INTRODUCTION:

- The process of dividing the body into an equivalent number of finite elements associated with nodes is called as discretization of an element in finite element analysis.
- Discretization helps out to know the stress and displacement values at a particular point also.

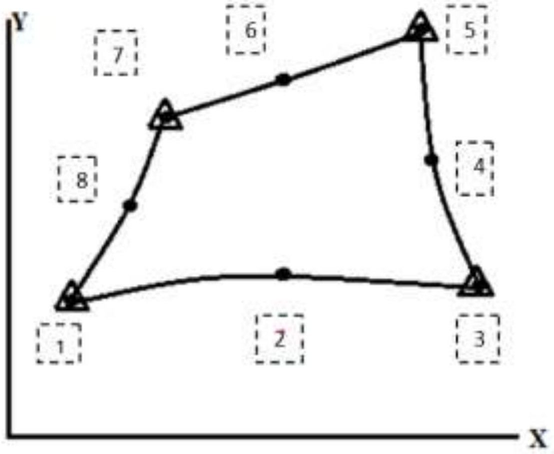


Discretization



Isoparametric element

Geometry = Displacem
Node N

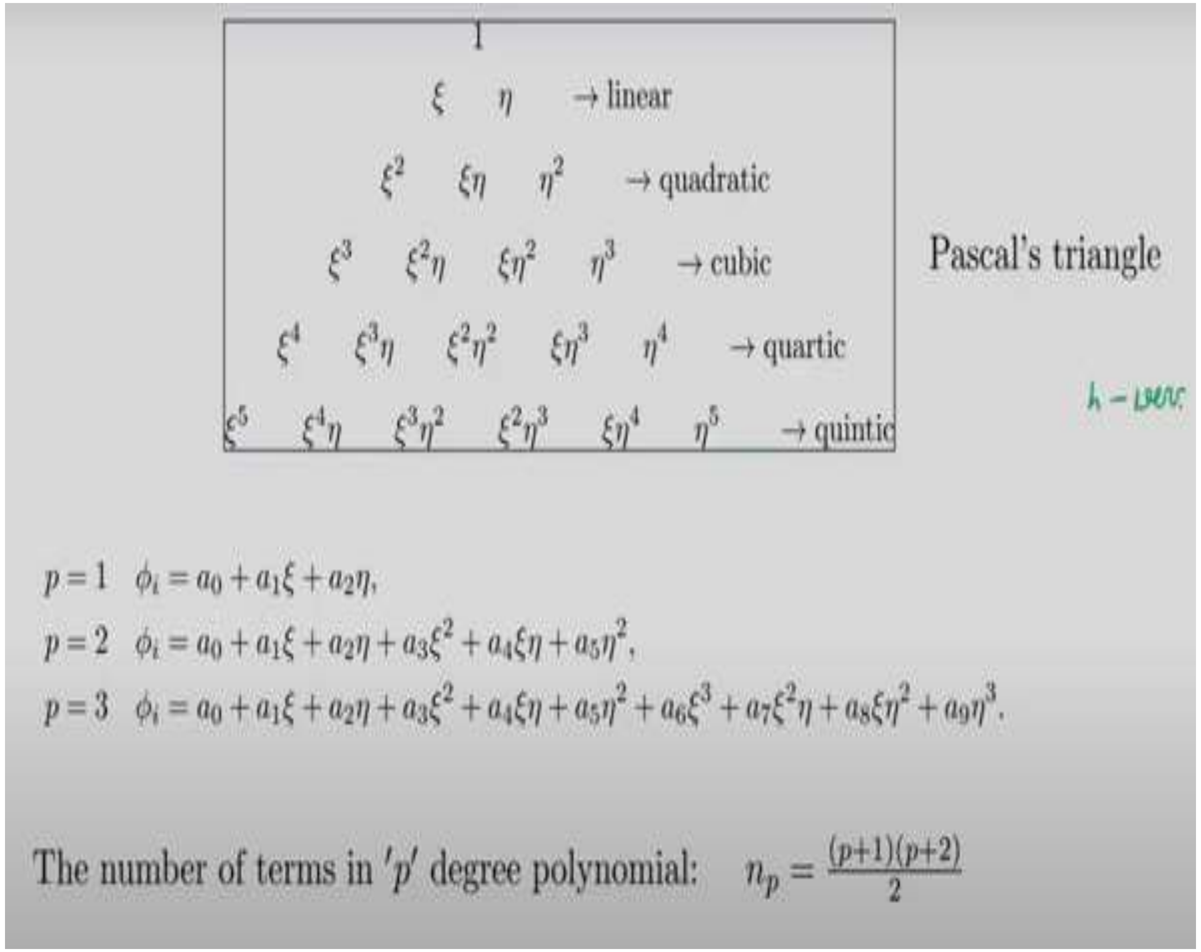


Subparametric element

Geometry < Displacem
Node N

INTRODUCTION:

- The finite element method (FEM) is a powerful numerical tool used across various fields, from structural engineering to fluid dynamics, for solving complex partial differential equations. While lower-order finite elements have been widely utilized for their simplicity and computational efficiency, higher-order finite elements (HOFE) offer significant advantages in terms of solution accuracy and convergence rates.



STATE OF THE ART:

TITLE	AUTHOR	Program	Year	Element used	FINDINGS
Shear Lag Analysis due to Flexure of Prismatic Beams with Arbitrary Cross-Sections by FEM	Dang-Bao Tran	MATLAB Code	2021	9-Noded	In this study, shear lag ef resulting from flexure of beams arbitrary c sections homogenous ela materials is exam using the fi element method (F
Finite Elements for Engineering Analysis: A	R w lewis	FORTRAN	2011	CST, LST, 4-	Comparison of all elemen with one problem.

TITLE	AUTHOR	Program	Year	Element used	FINDING
A FIRST COURSE IN THE FINITE ELEMENT METHOD	Daryl L. Logan	fortran	1986	CST, LST, 4-Noded	Systemic development theories,2 d elements
Finite Element Analysis: Theory and Programming	C. S. Krishnamoorthy	Fortran	1987	4-Noded	Emphasis on programming, ation to engg problems
Stiffness Matrices of Isoparametric Four-node Finite Elements by Exact Analytical Integration	Gautam Dasgupta	Mathematica programming language	2006	4-Noded	Analyti l integra on met is usec
Development of an Interactive Finite Element Solution Module for 2d Stress Problem Analysis Using Isoparametric	K.M. entwistle	MATLAB Code	2018	8-Noded	Variationa and energ methods

GAPS AND DOMAIN OF WORK

Automated Mesh Generation for Higher-Order Elements

•**Gap**: While mesh generation is well-developed for linear elements, creating high-quality meshes for higher-order elements, especially in 3D with curved boundaries, is still problematic.

Scalable and Efficient Solvers

•**Gap**: The computational cost of higher-order finite element methods is substantial, particularly for large-scale problems.

Adaptive Refinement Techniques

•**Gap**: Adaptive refinement for higher-order finite elements is less mature compared to its application in lower-order elements.

5. Robustness in Multi-Physics Simulations

•**Gap**: The application of HOFE in multi-physics and highly nonlinear problems.

6. Numerical Integration and Quadrature Rules

•**Gap**: Accurate numerical integration for high-order elements, especially for elements with curved boundaries or in 3D, is a challenging task.

Handling Complex Boundary Conditions

•**Gap**: Imposing complex boundary conditions such as mixed or non-standard conditions in HOFE is less straightforward than with lower-order elements.

• **Parallelization and High-Performance Computing (HPC)**

•**Gap**: HOFE methods inherently involve more computational data, which can challenge existing parallel computing frameworks.

Application-Specific Methodologies

•**Gap**: There is limited research focused on optimizing HOFE for specific applications, such as turbulent flow simulation.

METHODOLOGY

DISCRITIZATION ← Selection of element geometry

⇓

SELECTION OF APPROPRIATE SHAPE FUNCTIONS ← Determination of pattern/shape for the distribution of unknown variables within the continuum

⇓

DEVELOPMENT OF FINITE ELEMENT EQUATIONS ← Application of suitable principles/laws to form governing equations for the continuum and rewrite in the form of equations by incorporating suitable shape function

⇓

ASSEMBLE ELEMENT EQUATIONS TO OBTAIN GLOBAL/ASSEMBLAGE EQUATIONS IN THE FORM OF EQUATION

⇓

SOLVE FOR THE UNKNOWN

⇓

INTERPRETATION OF RESULTS

MATHEMATICAL FORMULATIONS

2. Strain-displacement relationships:

$$\varepsilon_x = \frac{\partial u}{\partial x}$$
$$\varepsilon_y = \frac{\partial v}{\partial y}$$
$$\varepsilon_z = \frac{\partial w}{\partial z}$$
$$\gamma_{xy} = \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}$$
$$\gamma_{yz} = \frac{\partial v}{\partial z} + \frac{\partial w}{\partial y}$$
$$\gamma_{zx} = \frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}$$

Displacement approximation in terms of shape functions

$$\boxed{\underline{u} = \underline{N}\ \underline{d}}$$

Strain approximation in terms of strain-displacement matrix

$$\boxed{\underline{\varepsilon} = \underline{B}\ \underline{d}}$$

Stress approximation

$$\boxed{\underline{\sigma} = \underline{D}\underline{B}\ \underline{d}}$$

Element stiffness matrix

$$\boxed{\underline{k} = \int_{V^e} \underline{B}^T \underline{D}\ \underline{B}\ dV}$$

Element nodal load vector

$$\boxed{\underline{f} = \underbrace{\int_{V^e} \underline{N}^T \underline{X}\ dV}_{\underline{f}_b} + \underbrace{\int_{S_T^e} \underline{N}^T \underline{T}_S\ dS}_{\underline{f}_S}}$$

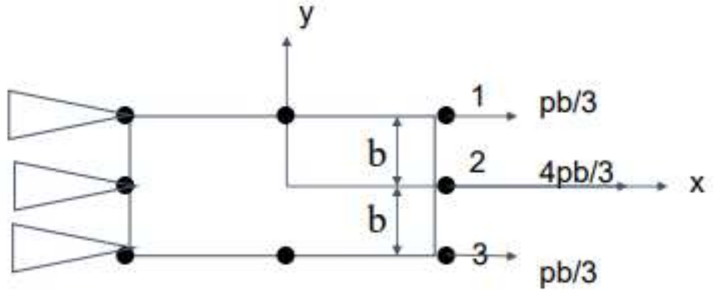
MATHEMATICAL FORMULATIONS

The **consistent nodal loads** are

$$F_{1x} = \int_{-b}^{b} p\, N_1 dy = \int_{-b}^{b} p\, \frac{y(b+y)}{2b^2} dy = \frac{pb}{3}$$

$$F_{2x} = \int_{-b}^{b} p\, N_2 dy = \int_{-b}^{b} p\, \frac{b^2-y^2}{b^2} dy = \frac{4pb}{3}$$

$$F_{3x} = \int_{-b}^{b} p\, N_3 dy = -\int_{-b}^{b} p\, \frac{y(b-y)}{2b^2} dy = \frac{pb}{3}$$

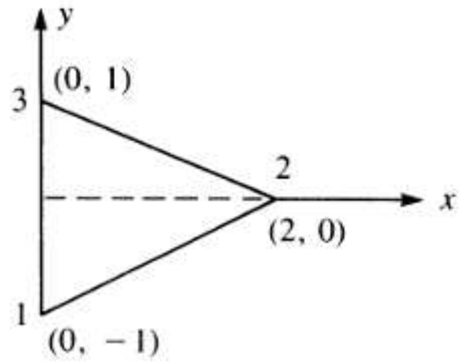


$$[k] = \frac{tE}{4A(1+\nu)(1-2\nu)} \begin{bmatrix} \beta_i & 0 & \gamma_i \\ 0 & \gamma_i & \beta_i \\ \beta_j & 0 & \gamma_j \\ 0 & \gamma_j & \beta_j \\ \beta_m & 0 & \gamma_m \\ 0 & \gamma_m & \beta_m \end{bmatrix}$$

$$\times \begin{bmatrix} 1-\nu & \nu & 0 \\ \nu & 1-\nu & 0 \\ 0 & 0 & \frac{1-2\nu}{2} \end{bmatrix} \begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix}$$

MATHEMATICAL FORMULATIONS

Evaluating the stiffness matrix for the elements shown in Figure . The coordinates are in units of inches. Assume plane stress conditions. Let E 30 106 psi, n ¼ 0:25, and thickness t ¼ 1 in.



$$[k] = t\,A\,[B]^T[D][B]$$

$$x_i = 0,\ y_i = -1,\ x_j = 2,\ y_j = 0,\ x_m = 0,\ y_m = 1$$

$$A = \frac{1}{2}\,b\,h = \frac{1}{2}(2)(2) = 2\ \text{in.}^2$$

$$\beta_i = y_j - y_m = 0 - 1 = -1$$

$$\beta_j = y_m - y_i = 1 - (-1) = 2$$

$$\beta_m = y_i - y_j = -1 - 0 = -1$$

$$\gamma_i = x_m - x_j = 0 - 2 = -2$$

$$\gamma_j = x_i - x_m = 0 - 0 = 0$$

$$\gamma_m = x_j - x_i = 2 - 0 = 2$$

$$[B] = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix}$$

$$[B] = \frac{1}{2A}\begin{bmatrix} \beta_i & 0 & \beta_j & 0 & \beta_m & 0 \\ 0 & \gamma_i & 0 & \gamma_j & 0 & \gamma_m \\ \gamma_i & \beta_i & \gamma_j & \beta_j & \gamma_m & \beta_m \end{bmatrix}$$

Since it is plane stress $[D] = \dfrac{E}{(1-v^2)}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}$

So $[B]^T[D] = \dfrac{30\times 10^6}{4(0.9375)}\begin{bmatrix} -1 & 0 & -2 \\ 0 & -2 & -1 \\ 2 & 0 & 0 \\ 0 & 0 & 2 \\ -1 & 0 & 2 \\ 0 & 2 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0.25 & 0 \\ 0.25 & 1 & 0 \\ 0 & 0 & 0.375 \end{bmatrix}$

$$\text{So } [B]^T[D] = \frac{30\times10^6}{4(0.9375)}\begin{bmatrix} -1 & 0 & -2 \\ 0 & -2 & -1 \\ 2 & 0 & 0 \\ 0 & 0 & 2 \\ -1 & 0 & 2 \\ 0 & 2 & -1 \end{bmatrix}\begin{bmatrix} 1 & 0.25 & 0 \\ 0.25 & 1 & 0 \\ 0 & 0 & 0.375 \end{bmatrix}$$

$$= \begin{bmatrix} -1 & -0.25 & -0.75 \\ -0.5 & -2 & -0.375 \\ 2 & 0.5 & 0 \\ 0 & 0 & 0.75 \\ -1 & -0.25 & 0.75 \\ 0.5 & 2 & -0.375 \end{bmatrix}\frac{30\times10^6}{4(0.9375)}$$

$$[k] = tA[B]^T[D]\,[B]$$

$$\Rightarrow \quad [k] = (1\text{ in.})(2)\frac{30\times10^6}{4(0.9375)}\begin{bmatrix} -1 & -0.25 & -0.75 \\ -0.5 & -2 & -0.375 \\ 2 & 0.5 & 0 \\ 0 & 0 & 0.75 \\ -1 & -0.25 & 0.75 \\ 0.5 & 2 & -0.375 \end{bmatrix}$$

$$\frac{1}{4}\begin{bmatrix} -1 & 0 & 2 & 0 & -1 & 0 \\ 0 & -2 & 0 & 0 & 0 & 2 \\ -2 & -1 & 0 & 2 & 2 & -1 \end{bmatrix}$$

$$[k] = 4.0\times10^6 \begin{array}{c} \begin{array}{ccc} \quad i=1 \quad & \quad j=2 \quad & \quad m=3 \quad \end{array} \\ \begin{bmatrix} 2.5 & 1.25 & -2 & -1.5 & -0.5 & 0.25 \\ 1.25 & 4.375 & -1 & -0.75 & -0.25 & -3.625 \\ -2 & -1 & 4 & 0 & -2 & 1 \\ -1.5 & -0.75 & 0 & 1.5 & 1.5 & -0.75 \\ -0.5 & -0.25 & -2 & 1.5 & 2.5 & -1.25 \\ 0.25 & -3.625 & 1 & -0.75 & -1.25 & 4.375 \end{bmatrix} \end{array}$$

main.cpp Share Run

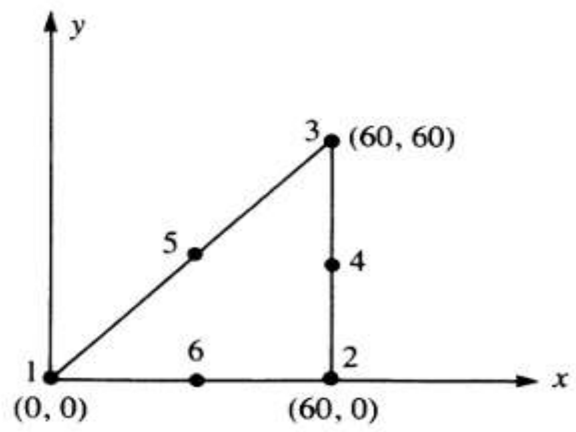
```cpp
    // Compute D * B
    std::vector<std::vector<double>> DB = multiplyMatrices(D, B);
    if(DB.empty()) {
        std::cerr << "Error in multiplying D and B matrices for Element "
            << elem.id << ". Skipping.\n";
        continue;
    }

    // Compute BT * (D * B)
    std::vector<std::vector<double>> BTDB = multiplyMatrices(BT, DB);
    if(BTDB.empty()) {
        std::cerr << "Error in multiplying B^T and (D * B) matrices for
            Element " << elem.id << ". Skipping.\n";
        continue;
    }

    // Compute stiffness matrix K = t * A * BTDB
    std::vector<std::vector<double>> K = BTDB;
    for(auto& row : K) {
        for(auto& val : row) {
            val *= t * area;
        }
    }

    // Display stiffness matrix
    std::cout << "\nStiffness Matrix for Element " << elem.id << ":\n";
    printMatrix(K);
```

Output Clear

```
Enter the number of nodes: 3

Enter node data (ID x y) for each node:
Format: ID x y
1 0 -1
2 2 0
3 0 1

Enter the number of elements: 1

Enter element data (ID node1 node2 node3) for each element:
Format: ID node1 node2 node3
1 1 2 3

Stiffness Matrix for Element 1:
  10000000.0000   5000000.0000  -8000000.0000  -6000000.0000  -2000000
      .0000   1000000.0000
   5000000.0000  17500000.0000  -4000000.0000  -3000000.0000  -1000000
      .0000  -14500000.0000
  -8000000.0000  -4000000.0000  16000000.0000        0.0000  -8000000
      .0000   4000000.0000
  -6000000.0000  -3000000.0000        0.0000   6000000.0000   6000000
      .0000  -3000000.0000
  -2000000.0000  -1000000.0000  -8000000.0000   6000000.0000  10000000
      .0000  -5000000.0000
   1000000.0000 -14500000.0000   4000000.0000  -3000000.0000  -5000000
      .0000  17500000.0000
```

DOMAIN DISCUSSIONS

Evaluating the shape functions for the linear-strain triangle shown. Then evaluate theB matrix. Units are millimeters



$$u_1 = u(0, 0) = a_1 \quad (1)$$

$$u_2 = u(60, 0) = a_1 + 60\, a_2 + 3600\, a_4 \quad (2)$$

$$u_3 = u(60, 60) = a_1 + 60\, a_2 + 60\, a_3 + 3600\, a_4 + 3600\, a_5 + 3600\, a_6 \quad (3)$$

$$u_4 = u(60, 30) = a_1 + 60\, a_2 + 30\, a_3 + 3600\, a_4 + 1800\, a_5 + 800\, a_6 \quad (4)$$

$$u_5 = u(30, 30) = a_1 + 30\, a_2 + 30\, a_3 + 900\, a_4 + 900\, a_5 + 900\, a_6 \quad (5)$$

$$u_6 = u(30, 0) = a_1 + 30\, a_2 + 900\, a_4 \quad (6)$$

By (1) $\Rightarrow a_1 = u_1$

By (2) – 2 (6) $\Rightarrow a_4 = \dfrac{u_2 - 2u_6 + u_1}{1800}$

By – (2) + 4 (6) $\Rightarrow a_2 = \dfrac{4u_6 - u_2 + 3u_1}{60}$

By 2(4) – (3) $\Rightarrow a_6 = \dfrac{u_2 + u_3 - 2u_4}{1800}$

(4) – (5) $\Rightarrow a_5 = \dfrac{-u_2 + u_4 - u_5 + u_6}{900}$

(4) $\Rightarrow a_3 = \dfrac{u_2 - u_3 + 4u_5 - 4u_6}{60}$

Can verify by substituting all a's into Equation (3)

$$\therefore \quad u = u_1 + \left(\frac{4u_6 - u_2 + 3u_1}{60}\right)x + \left(\frac{u_2 - u_3 + 4u_5 - 4u_6}{60}\right)y$$

$$+ \left(\frac{u_2 - 2u_6 + u_1}{1800}\right)x^2 + \left(\frac{-u_2 + u_4 - u_5 + u_6}{900}\right)xy$$

$$+ \left(\frac{u_2 + u_3 - 2u_4}{1800}\right)y^2$$

$\therefore$ Shape functions are

$$N_1 = 1 - \frac{3x}{60} + \frac{x^2}{1800} \quad \text{(From all } u_1 \text{ coefficient)}$$

$$N_2 = \frac{-x}{60} + \frac{y}{60} + \frac{x^2}{1800} - \frac{xy}{900} + \frac{y^2}{1800}$$

$$N_3 = \frac{-y}{60} + \frac{y^2}{1800}$$

$$N_4 = \frac{xy}{900} - \frac{2y^2}{1800}$$

$$N_5 = \frac{4y}{60} - \frac{xy}{900}$$

$$N_6 = \frac{4x}{60} - \frac{4y}{60} - \frac{2x^2}{1800} + \frac{xy}{900}$$

$$2A = 2\left(\frac{1}{2}\right)(60)(60) = 3600$$

$$\beta_1 = 2A\left(\frac{\partial N_1}{\partial x}\right) = 3600\left(-\frac{3}{60} + \frac{2x}{1800}\right) = -180 + 4x$$

$$\beta_2 = 3600\left[-\frac{1}{60} + \frac{2x}{1800} - \frac{y}{900}\right] = -60 + 4x - 4y$$

$$\beta_3 = 0, \ \beta_4 = 3600\left(\frac{y}{900}\right) = 4y$$

$$\beta_5 = 3600\left(\frac{-y}{900}\right) = -4y$$

$$\beta_6 = 3600\left[\frac{4}{60} - \frac{4x}{1800} + \frac{y}{900}\right] = 240 - 8x + 4y$$

$$\gamma_1 = 2A\frac{\partial N_1}{\partial y} = 0, \ \gamma_5 = 3600\left(\frac{4}{60} - \frac{x}{900}\right) = 240 - 4x$$

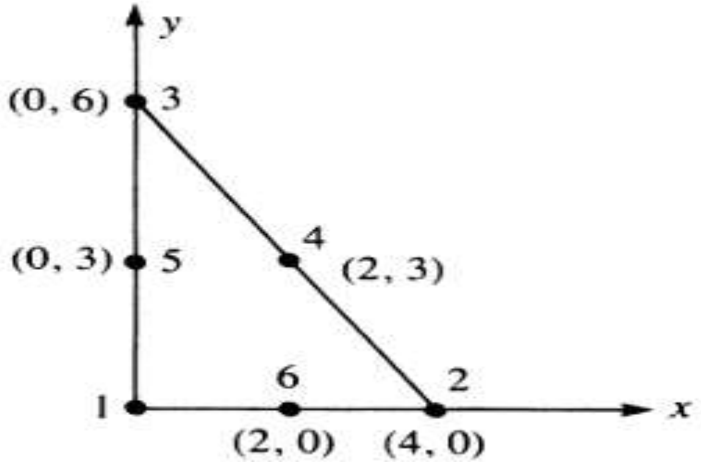
$$\gamma_2 = 3600\left(\frac{1}{60} - \frac{x}{900} + \frac{2y}{1800}\right) = 60 - 4x + 4y$$

$$\gamma_3 = 3600\left(-\frac{1}{60} + \frac{2y}{1800}\right) = -60 + 4y$$

$$\gamma_4 = 3600\left(\frac{x}{900} - \frac{4y}{1800}\right) = 4x - 8y$$

$$\gamma_6 = 3600\left(\frac{-4}{60} + \frac{x}{900}\right) = -240 + 4x$$

DOMAIN DISCUSSIONS



For the linear-strain elements shown in Figure P8–5, determine the strains $\varepsilon_x, \varepsilon_y$, and γ_{xy}. Evaluate the stresses σ_x, σ_y, and τ_{xy} at the centroids. The coordinates of the nodes are shown in units of inches. Let $E = 30 \times 10^6$ psi, $v = 0.25$, and $t = 0.25$ in. for both elements. Assume plane stress conditions apply. The nodal displacements are given as

$$u_1 = 0.0 \qquad v_1 = 0.0$$
$$u_2 = 0.001 \text{ in.} \qquad v_2 = 0.002 \text{ in.}$$
$$u_3 = 0.0005 \text{ in.} \qquad v_3 = 0.0002 \text{ in.}$$
$$u_4 = 0.0002 \text{ in.} \qquad v_4 = 0.0001 \text{ in.}$$
$$u_5 = 0.0 \qquad v_5 = 0.0001 \text{ in.}$$
$$u_6 = 0.0005 \text{ in.} \qquad v_6 = 0.001 \text{ in.}$$

$h =$ 6 in.
$b =$ 4 in.
$2A =$ 24 in.^{2}
Centroid at $\left(\frac{4}{3}, 2\right)$

$$\{\varepsilon\} = [B]\{d\}$$

$$[B] = \frac{1}{2A}\begin{bmatrix} \beta_1 & 0 & \beta_2 & 0 & \beta_3 & 0 & \beta_4 & 0 & \beta_5 & 0 & \beta_6 & 0 \\ 0 & \gamma_1 & 0 & \gamma_2 & 0 & \gamma_3 & 0 & \gamma_4 & 0 & \gamma_5 & 0 & \gamma_6 \\ \gamma_1 & \beta_1 & \gamma_2 & \beta_2 & \gamma_3 & \beta_3 & \gamma_4 & \beta_4 & \gamma_5 & \beta_5 & \gamma_6 & \beta_6 \end{bmatrix}$$

$$\beta_1 = -3h + \frac{4hx}{6} + 4y = 6x + 4y - 18$$

$$\beta_2 = -h + \frac{4hx}{6} = 6x - 6,\ \beta_3 = 0$$

$$\beta_4 = 4y,\ \beta_5 = -4y$$

$$\beta_6 = 4h - \frac{8hx}{6} - 4y = -12x - 4y + 24$$

$$\gamma_1 = -3b + 4x + \frac{4by}{h} = 4x + \frac{8}{3}y - 12$$

$$\gamma_2 = 0$$

$$\gamma_3 = -b + \frac{4by}{h} = \frac{8}{3}y - 4$$

$$\gamma_4 = 4x$$

DOMAIN DISCUSSIONS

$$\gamma_5 = 4b - 4x - \frac{8by}{h} = -4x - \frac{16}{3}y + 16$$

$$\gamma_6 = -4x$$

$$\therefore \quad 2A\,\varepsilon_x = \beta_2 u_2 + \beta_4 u_4 + \beta_6 u_6$$

$$= 0.001\,(6x - 6) + 0.0002\,(4y) + 0.0005\,(-12x - 4y + 24)$$

$$2A\,\varepsilon_x = -0.0012y + 0.006$$

$$\therefore \quad \varepsilon_x = -5 \times 10^{-5}\,y + 2.5 \times 10^{-4}$$

$$2A\,\varepsilon_y = \gamma_3 v_3 + \gamma_4 v_4 + \gamma_5 v_5 + \gamma_6 v_6$$

$$= 0.0002\left(\frac{8}{3}y - 4\right) + 0.0001\,(4x) + 0.0001\left(-4x - \frac{16}{3}y + 16\right) + 0.001\,(-4x)$$

$$2A\,\varepsilon_y = -0.004x + 0.0008$$

$$\therefore \quad \varepsilon_y = -1.67 \times 10^{-4}x + 3.33 \times 10^{-5}$$

$$2A\,\gamma_{xy} = 0.002\,(6x - 6) + 0.0005\left(\frac{8}{3}y - 4\right) + 0.0002\,(4x) + 0.0001\,(4y)$$

$$+ 0.0001\,(-4y) + 0.0005\,(-4x) + 0.001\,(-12x - 4y + 24)$$

$$2A\,\gamma_{xy} = -0.0012x - 0.00267y + 0.01$$

$$\therefore \quad \gamma_{xy} = -5 \times 10^{-5}\,x - 1.11 \times 10^{-4}\,y + 4.167 \times 10^{-4}$$

Evaluate stresses at centroid

$$\{\sigma\} = [D]\,\{\varepsilon\}$$

$$\{\varepsilon\}\Big|_{\left(\frac{4}{3},\,2\right)} = \begin{Bmatrix} 0.00015 \\ -1.89 \times 10^{-4} \\ 0.000128 \end{Bmatrix}$$

$$\{\sigma\} = \frac{E}{1 - v^2}\begin{bmatrix} 1 & v & 0 \\ v & 1 & 0 \\ 0 & 0 & \frac{1-v}{2} \end{bmatrix}\begin{Bmatrix} 1.5 \times 10^{-4} \\ -1.89 \times 10^{-4} \\ 1.28 \times 10^{-4} \end{Bmatrix}$$

$$\{\sigma\} = \begin{Bmatrix} 3288 \\ -4848 \\ 1536 \end{Bmatrix} \text{psi}$$

$$I=\int_0^1\int_0^{1-t} f(s,t)\,ds\,dt \approx \sum_{i=1}^{m} w_i f(s_i,t_i)$$

- where w_i = weight and (s_i, t_i) = coordinates at the integration point and m is the total number of points.

Evaluate $I=\int_{-1}^{1}\int_{-1}^{1}\left(8s^7+7t^6\right)dsdt$ using Gauss quadrature.

(i) Using 1 x 1 formula:

$f(0,0) = 0 \qquad I = 0$

(ii) Using 2 x 2 formula

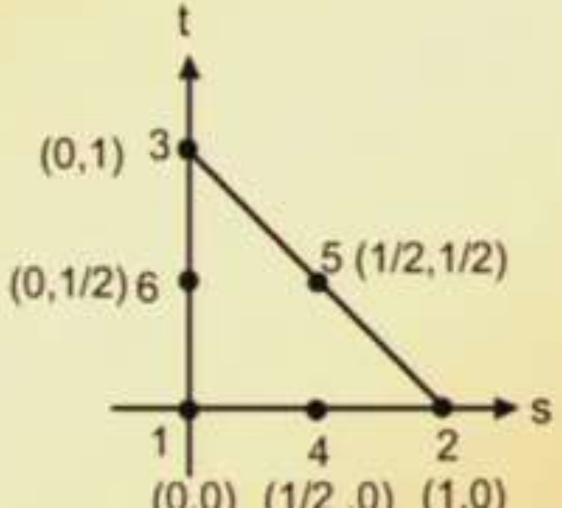
$f(s,t) = 8s^7 + 7t^6$

- n = number of Gauss points in the t direction
- w_i and w_j = Gauss weights in s and t directions
- $f(s_i,t_j)$ = value of the integrand at the point (s_i,t_j)
- The total number of gauss points = m x n

The calculations are summarized in the following table

Point	s_i	t_j	$f(s_i,t_j)$	w_i	w_j	$w_i\, w_j\, f(s_i,t_j)$
1	0.57735	-0.57735	0.43032	1	1	0.43032
2	0.57735	0.57735	0.43032	1	1	0.43032
3	-0.57735	0.57735	0.088192	1	1	0.088192
4	-0.57735	-0.57735	0.088192	1	1	0.088192
Sum						1.03703

Evaluate $\iint_A N_1^2 N_5 \, ds\,dt$ over of a quadratic triangular element shown in figure below.



One point integration:

$$I = 1/2\ f(1/3, 1/3) = 0.002744$$

Three point integration:

$$I = 1/6\ f(1/6, 1/6) + 1/6\ f(2/3, 1/6) + 1/6\ f(1/6, 2/3)$$

$$= 0.0009145 + 0.0009145 + 0.0009145 = 0.002744$$

Four point integration:

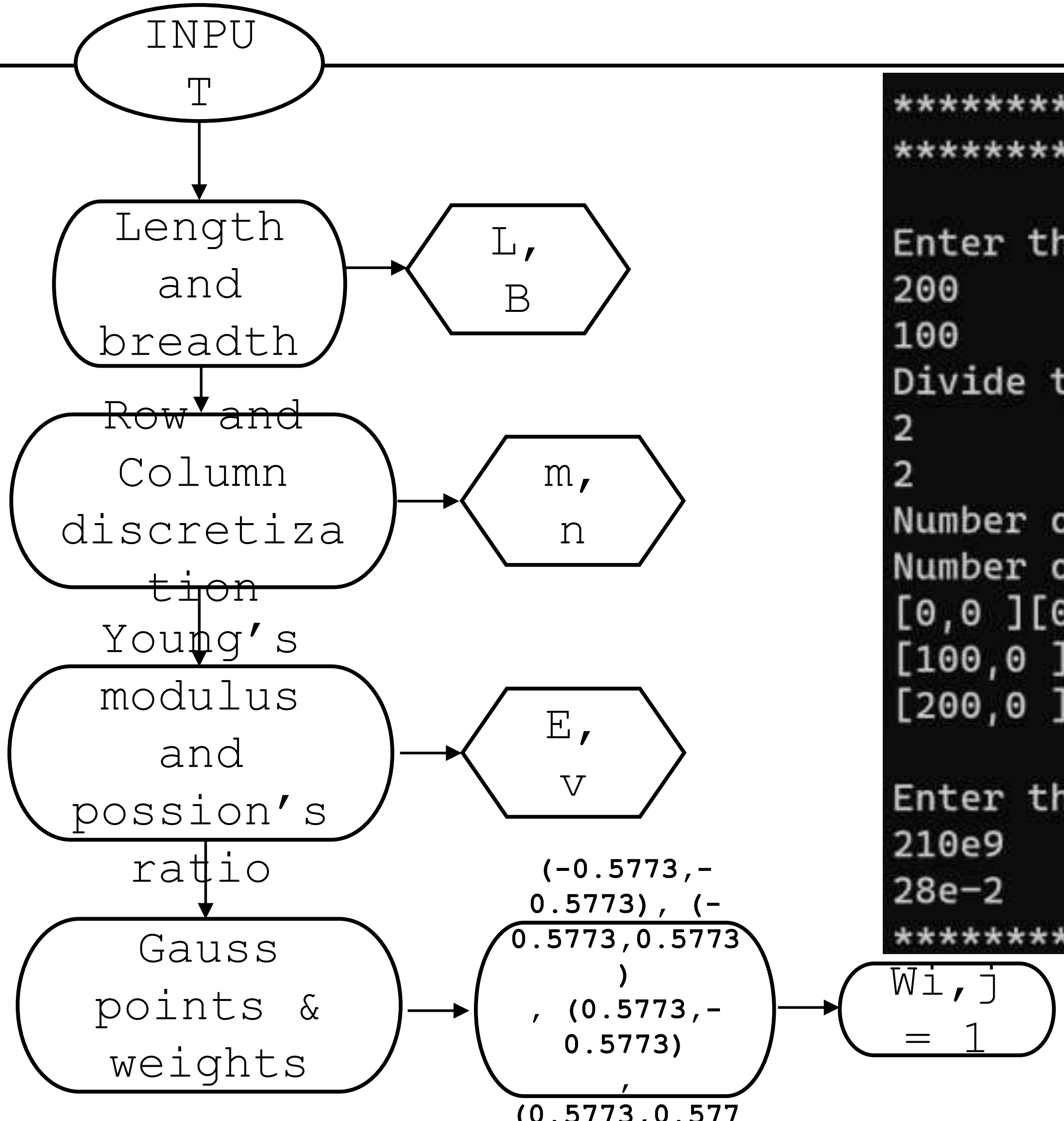
$$I = -9/32\ f(1/3, 1/3) + 25/96\ f(1/5, 1/5) + 25/96\ f(3/5, 1/5) + 25/96\ f(1/5, 3/5)$$

$$= -0.001543 + 0.0006 + 0.0018 + 0.0018 = 0.002657$$

It can easily be verified that the exact integral = 0.001587

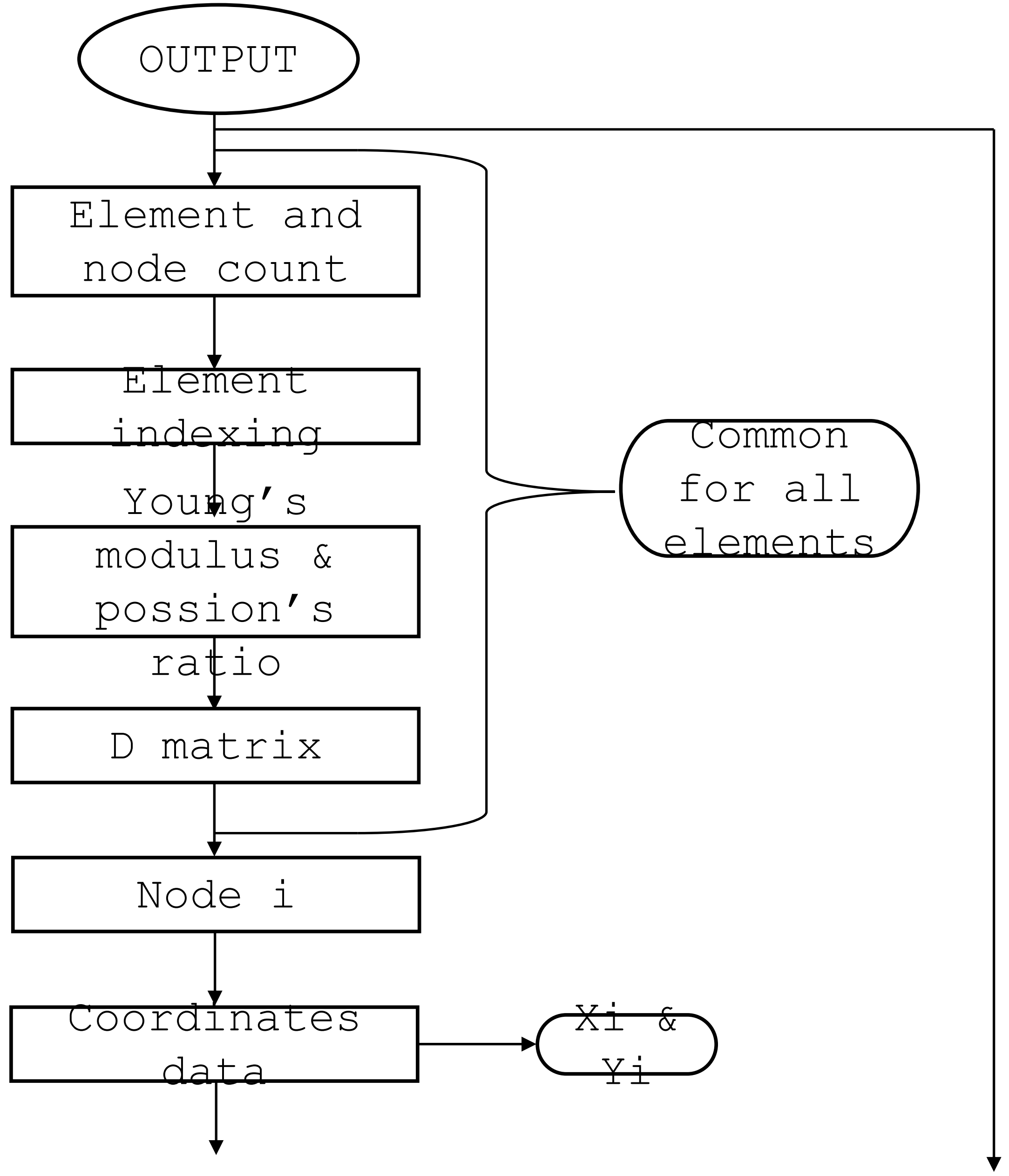
RESULT DISCUSSION

MAIN FLOW CHART ALGORITHM:



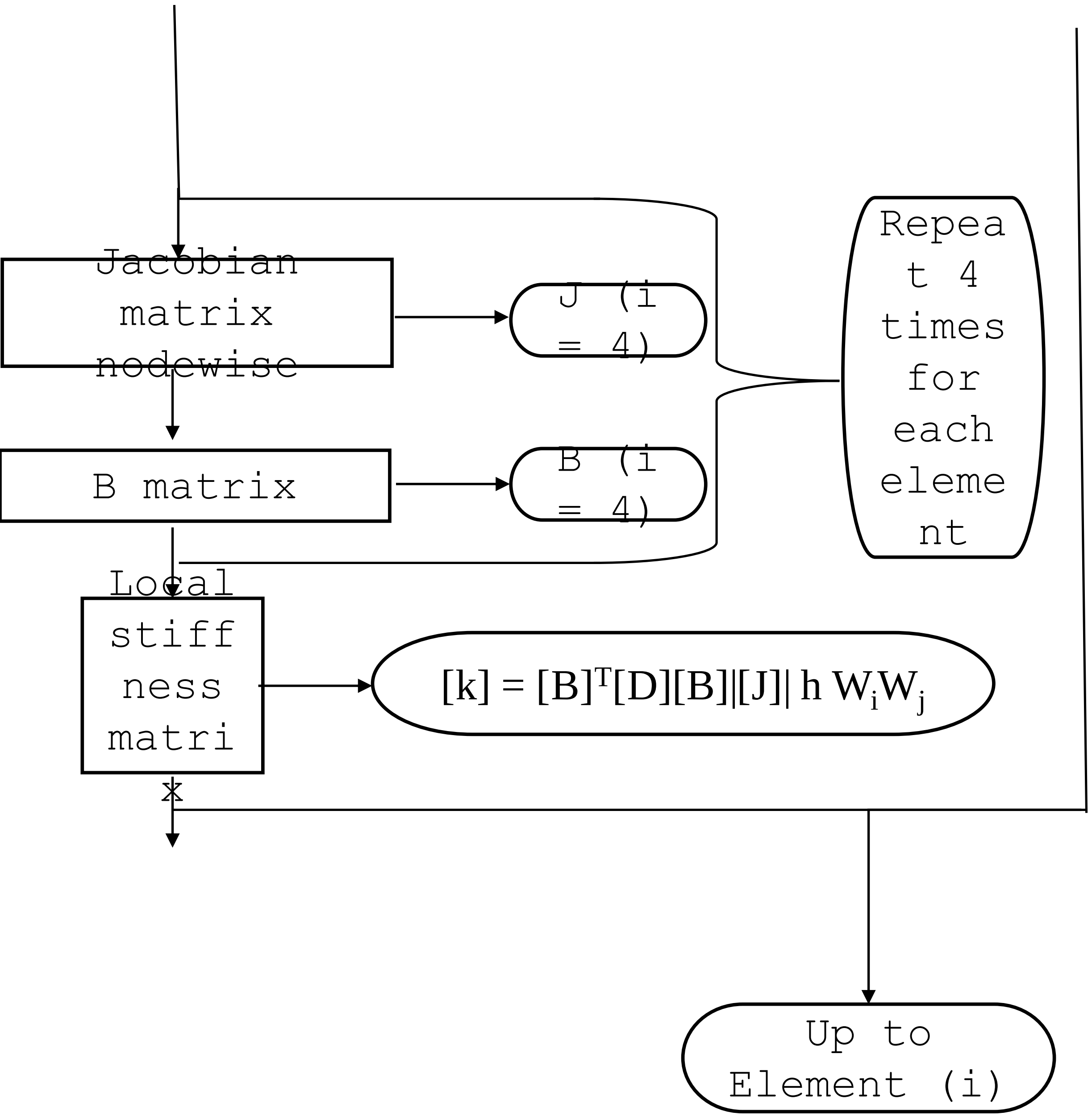

```
**************************************Hello Welcome to Meshing world!
*********

Enter the length and breadth of the plate.
200
100
Divide the length and breadth into column (p) and rows (m)  pieces.
2
2
Number of elements present:4
Number of nodes present:9
[0,0 ][0,50 ][0,100 ]
[100,0 ][100,50 ][100,100 ]
[200,0 ][200,50 ][200,100 ]

Enter the values of E = Young's modulus and v = Poisson's ratio
210e9
28e-2
**********************************************************************
```

RESULT DISCUSSION




```
For Element: 1
00
10
11
01
Young's Modulus =2.1e+011
Poisson's ratio =0.28
D matrix is:
2.27865e+011    6.38021e+010    0
6.38021e+010    2.27865e+011    0
0       0       8.20312e+010
Nodes of element: 1
X Co-ordinates for 4 Nodes of element is:1
 0
 100
 100
 0

Transpose of X is Xt
 0 100 100 0
Y Co-ordinate for 4 Nodes of element is:1
 0
 0
 50
 50
```

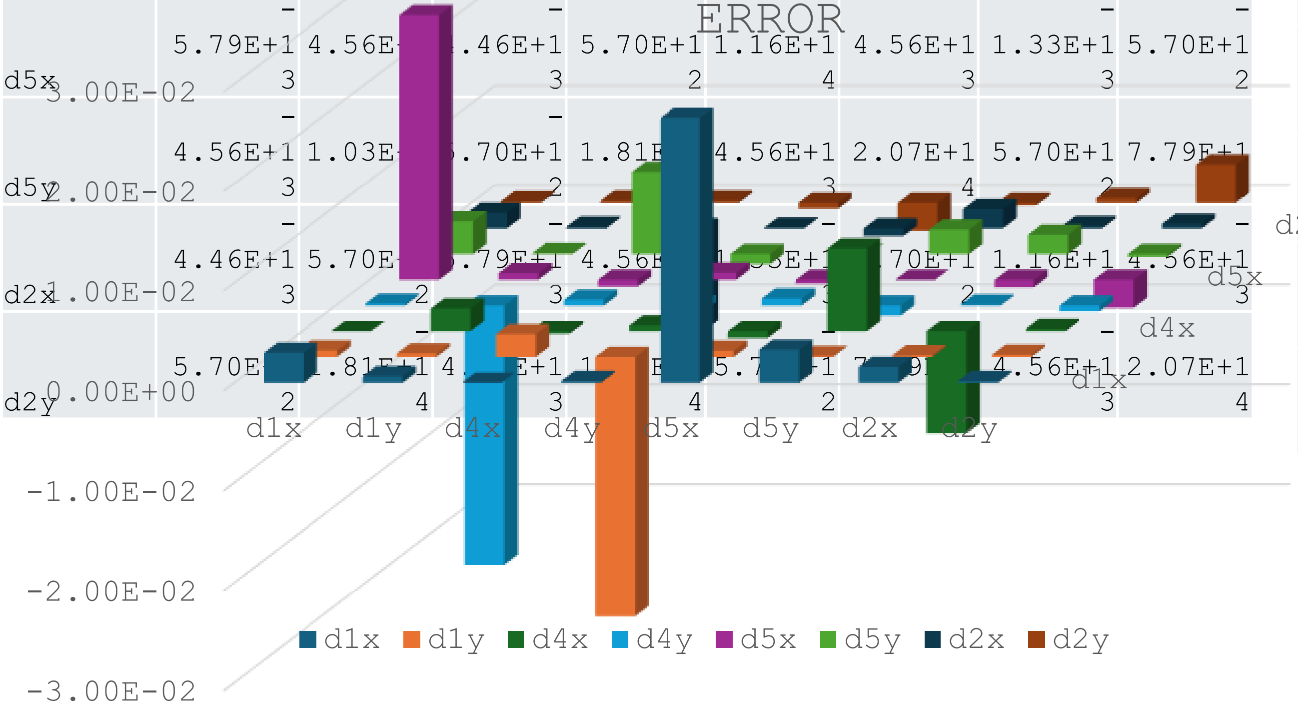
RESULT DISCUSSION




```
Jacobian Matrix of element:2
0	0.197163	0	-0.197163
-0.197163	0	0.0528375	0.144325
0	-0.0528375	0	0.0528375
0.197163	-0.144325	-0.0528375	0
1250
5.40325e+013	2.83449e+013	-1.35988e+013	-1.26207e+013	-1.44882e+013	-7.59615e+012	-2.59536e+013	-8.12807e+012
2.83449e+013	9.65499e+013	-8.12807e+012	1.57659e+013	-7.59615e+012	-2.58744e+013	-1.26207e+013	-8.64415e+013
-1.35988e+013	-8.12807e+012	2.44346e+013	-7.59615e+012	3.64434e+012	2.17824e+012	-1.44882e+013	1.3546e+013
-1.26207e+013	1.57659e+013	-7.59615e+012	1.43335e+013	3.38222e+012	-4.22511e+012	1.68346e+013	-2.58744e+013
-1.44882e+013	-7.59615e+012	3.64434e+012	3.38222e+012	3.88053e+012	2.03569e+012	6.95528e+012	2.17824e+012
-7.59615e+012	-2.58744e+013	2.17824e+012	-4.22511e+012	2.03569e+012	6.93486e+012	3.38222e+012	2.31654e+013
-2.59536e+013	-1.26207e+013	-1.44882e+013	1.68346e+013	6.95528e+012	3.38222e+012	3.34784e+013	-7.59615e+012
-8.12807e+012	-8.64415e+013	1.3546e+013	-2.58744e+013	2.17824e+012	2.31654e+013	-7.59615e+012	8.91504e+013
0	0.0528375	0.144325	-0.197163
-0.0528375	0	0.0528375	0
-0.144325	-0.0528375	0	0.197163
0.197163	0	-0.197163	0

Jacobian Matrix of element:2
1250
3.34784e+013	7.59615e+012	6.95528e+012	-3.38222e+012	-1.44882e+013	-1.68346e+013	-2.59536e+013	1.26207e+013
7.59615e+012	8.91504e+013	-2.17824e+012	2.31654e+013	-1.3546e+013	-2.58744e+013	8.12807e+012	-8.64415e+013
6.95528e+012	-2.17824e+012	3.88053e+012	-2.03569e+012	3.64434e+012	-3.38222e+012	-1.44882e+013	7.59615e+012
-3.38222e+012	2.31654e+013	-2.03569e+012	6.93486e+012	-2.17824e+012	-4.22511e+012	7.59615e+012	-2.58744e+013
-1.44882e+013	-1.3546e+013	3.64434e+012	-2.17824e+012	2.44346e+013	7.59615e+012	-1.35988e+013	8.12807e+012
-1.68346e+013	-2.58744e+013	-3.38222e+012	-4.22511e+012	7.59615e+012	1.43335e+013	1.26207e+013	1.57659e+013
-2.59536e+013	8.12807e+012	-1.44882e+013	7.59615e+012	-1.35988e+013	1.26207e+013	5.40325e+013	-2.83449e+013
1.26207e+013	-8.64415e+013	7.59615e+012	-2.58744e+013	8.12807e+012	1.57659e+013	-2.83449e+013	9.65499e+013

Jacobian Matrix of element:2
0	0.197163	-0.144325	-0.0528375
-0.197163	0	0.197163	0
0.144325	-0.197163	0	0.0528375
0.0528375	0	-0.0528375	0
1250
2.44346e+013	7.59615e+012	-1.35988e+013	8.12807e+012	-1.44882e+013	-1.3546e+013	3.64434e+012	-2.17824e+012
7.59615e+012	1.43335e+013	1.26207e+013	1.57659e+013	-1.68346e+013	-2.58744e+013	-3.38222e+012	-4.22511e+012
-1.35988e+013	1.26207e+013	5.40325e+013	-2.83449e+013	-2.59536e+013	8.12807e+012	-1.44882e+013	7.59615e+012
8.12807e+012	1.57659e+013	-2.83449e+013	9.65499e+013	1.26207e+013	-8.64415e+013	7.59615e+012	-2.58744e+013
-1.44882e+013	-1.68346e+013	-2.59536e+013	1.26207e+013	3.34784e+013	7.59615e+012	6.95528e+012	-3.38222e+012
-1.3546e+013	-2.58744e+013	8.12807e+012	-8.64415e+013	7.59615e+012	8.91504e+013	-2.17824e+012	2.31654e+013
3.64434e+012	-3.38222e+012	-1.44882e+013	7.59615e+012	6.95528e+012	-2.17824e+012	3.88053e+012	-2.03569e+012
-2.17824e+012	-4.22511e+012	7.59615e+012	-2.58744e+013	-3.38222e+012	2.31654e+013	-2.03569e+012	6.93486e+012

Jacobian Matrix of element:2
0	0.0528375	0	-0.0528375
-0.0528375	0	0.197163	-0.144325
0	-0.197163	0	0.197163
0.0528375	0.144325	-0.197163	0
1250
3.88053e+012	2.03569e+012	6.95528e+012	2.17824e+012	-1.44882e+013	-7.59615e+012	3.64434e+012	3.38222e+012
2.03569e+012	6.93486e+012	3.38222e+012	2.31654e+013	-7.59615e+012	-2.58744e+013	2.17824e+012	-4.22511e+012
6.95528e+012	3.38222e+012	3.34784e+013	-7.59615e+012	-2.59536e+013	-1.26207e+013	-1.44882e+013	1.68346e+013
2.17824e+012	2.31654e+013	-7.59615e+012	8.91504e+013	-8.12807e+012	-8.64415e+013	1.3546e+013	-2.58744e+013
-1.44882e+013	-7.59615e+012	-2.59536e+013	-8.12807e+012	5.40325e+013	2.83449e+013	-1.35988e+013	-1.26207e+013
-7.59615e+012	-2.58744e+013	-1.26207e+013	-8.64415e+013	2.83449e+013	9.65499e+013	-8.12807e+012	1.57659e+013
3.64434e+012	2.17824e+012	-1.44882e+013	1.3546e+013	-1.35988e+013	-8.12807e+012	2.44346e+013	-7.59615e+012
3.38222e+012	-4.22511e+012	1.68346e+013	-2.58744e+013	-1.26207e+013	1.57659e+013	-7.59615e+012	1.43335e+013
Local stiffness matrix: 2 is
1.15826e+014	4.55729e+013	-1.32871e+013	-5.69661e+012	-5.79206e+013	-4.55729e+013	-4.46184e+013	5.69661e+012
4.55729e+013	2.06968e+014	5.69661e+012	7.78627e+013	-4.55729e+013	-1.83498e+014	-5.69661e+012	-1.81333e+014
-1.32871e+013	5.69661e+012	1.15826e+014	-4.55729e+013	-4.46184e+013	-5.69661e+012	-5.79206e+013	4.55729e+013
-5.69661e+012	7.78627e+013	-4.55729e+013	2.06968e+014	5.69661e+012	-1.81333e+014	4.55729e+013	-1.83498e+014
-5.79206e+013	-4.55729e+013	-4.46184e+013	5.69661e+012	1.15826e+014	4.55729e+013	-1.32871e+013	-5.69661e+012
-4.55729e+013	-1.83498e+014	-5.69661e+012	-1.81333e+014	4.55729e+013	2.06968e+014	5.69661e+012	7.78627e+013
-4.46184e+013	-5.69661e+012	-5.79206e+013	4.55729e+013	-1.32871e+013	5.69661e+012	1.15826e+014	-4.55729e+013
5.69661e+012	-1.81333e+014	4.55729e+013	-1.83498e+014	-5.69661e+012	7.78627e+013	-4.55729e+013	2.06968e+014
****************************************************************************************************
```

Comparison of Computed and analytical result :

COMPUTED	d1x	d1y	d4x	d4y	d5x	d5y	d2x	d2y
d1x	1.16E+14	4.56E+13	-1.33E+13	-5.70E+12	-5.79E+13	-4.56E+13	-4.46E+13	5.70E+12
d1y	4.56E+13	2.07E+14	5.70E+12	7.79E+13	-4.56E+13	-1.03E+14	-5.70E+12	-1.81E+14
d4x	-1.33E+13	5.70E+12	1.16E+14	-4.56E+13	-4.46E+13	-5.70E+12	-5.79E+13	4.56E+13
d4y	-5.70E+12	7.79E+13	-4.56E+13	2.07E+14	5.70E+12	-1.81E+14	4.56E+13	-1.03E+14
d5x	-5.79E+13	-4.56E+13	-4.46E+13	5.70E+12	1.16E+14	4.56E+13	-1.33E+13	-5.70E+12
d5y	-4.56E+13	-1.03E+14	-5.70E+12	-1.81E+14	4.56E+13	2.07E+14	5.70E+12	7.79E+13
d2x	-4.46E+13	5.70E+12	-5.79E+13	4.56E+13	1.33E+13	5.70E+12	1.16E+14	-4.56E+13
d2y	5.70E+12	-1.81E+14	4.56E+13	1.03E+14	-5.70E+12	7.79E+13	-4.56E+13	2.07E+14



ANALYTICAL	d1x	d1y	d4x	d4y	d5x	d5y	d2x	d2y
d1x	1.16E+14	4.56E+13	-1.33E+13	-5.70E+12	-5.79E+13	-4.56E+13	-4.46E+13	5.70E+12
d1y	4.56E+13	2.07E+14	5.70E+12	7.79E+13	-4.56E+13	-1.03E+14	-5.70E+12	-1.81E+14
d4x	-1.33E+13	5.70E+12	1.16E+14	-4.56E+13	-4.46E+13	-5.70E+12	-5.79E+13	4.56E+13
d4y	-5.70E+12	7.79E+13	-4.56E+13	2.07E+14	5.70E+12	-1.81E+14	4.56E+13	-1.03E+14
d5x	-5.79E+13	-4.56E+13	-4.46E+13	5.70E+12	1.16E+14	4.56E+13	-1.33E+13	-5.70E+12
d5y	-4.56E+13	-1.03E+14	-5.70E+12	-1.81E+14	4.56E+13	2.07E+14	5.70E+12	7.79E+13
d2x	-4.46E+13	5.70E+12	-5.79E+13	4.56E+13	1.33E+13	5.70E+12	1.16E+14	-4.56E+13
d2y	5.70E+12	-1.81E+14	4.56E+13	1.03E+14	-5.70E+12	7.79E+13	-4.56E+13	2.07E+14

- This are local stiffness matrix of element 1.
- Error = computed result – analytical result
- Maximum error = 2.66%
- Minimum error = 7.53e-10%
- Error might due to approximation made during the calculations.

CONCLUSION

Advantages over Linear Elements:

- **Increased Accuracy:** Higher-order finite elements allow for better representation of the solution space, achieving the same level of accuracy as linear elements but with fewer overall elements.
- **Smooth Transitions:** Utilizing higher-degree polynomial shape functions helps ensure smoother transitions across adjacent elements.
- **Handling Complex Geometries:** These elements are better suited for intricate or curved boundaries and areas where changes in the material properties or forces are non-linear.

Structural Analysis: Higher-order finite elements are particularly valuable in structural analysis, where precision and the ability to model complex stress distributions are crucial.

Electromagnetic Simulations: In electromagnetic field simulations, higher-order finite elements provide significant advantages when dealing with wave propagation and field distribution problems.

CONCLUSION

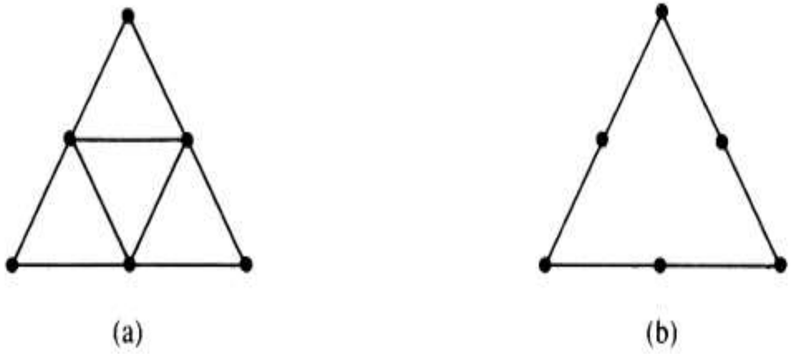


Figure 8–4 Basic triangular element: (a) four-CST and (b) one-LST

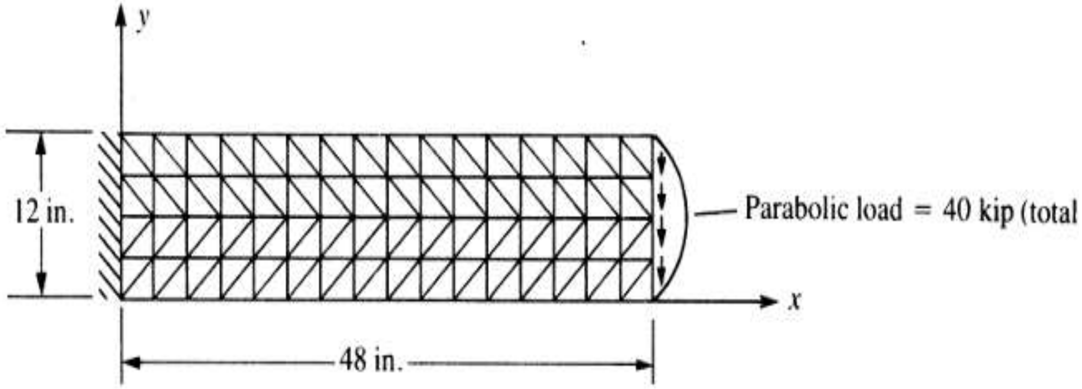
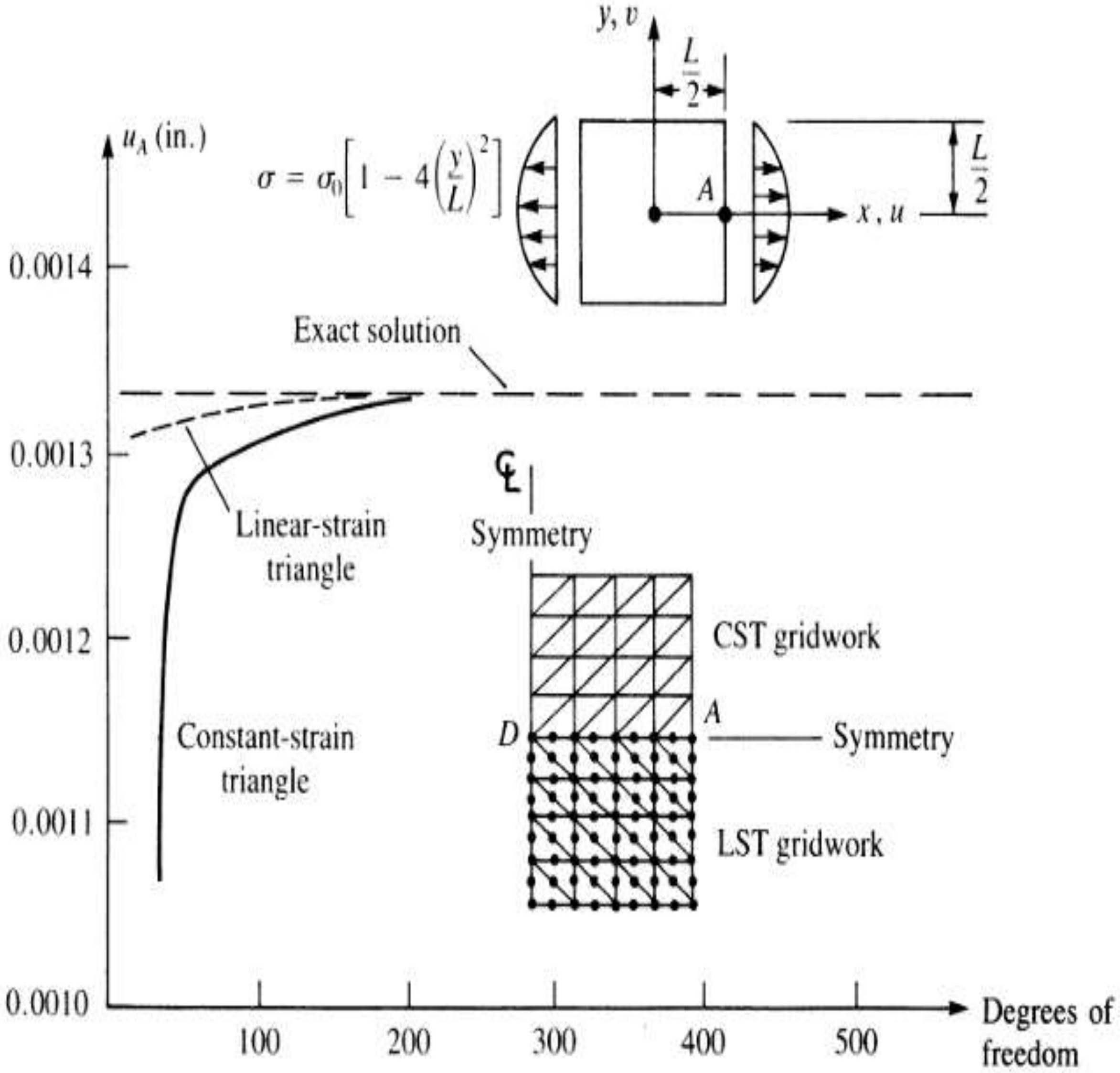


Figure 8–5 Cantilever beam used to compare the CST and LST elements with a 4 × 16 mesh



conclusion

- HOFE incorporates polynomial basis functions of higher degrees, allowing for more precise approximations with fewer elements compared to linear or low-order methods. This results in more accurate simulations, especially in problems involving complex geometries, wave propagation, or high-gradient fields. Despite their potential, HOFE poses unique challenges such as increased computational costs, handling complex geometries, and ensuring numerical stability.

WORK TO BE CARRIED OUT

1. **Optimization of Computational Algorithms**:
    1. Development and implementation of more efficient solvers specifically designed for higher-order finite element systems to reduce computational costs.
2. **Adaptive Mesh Refinement**:
    1. Incorporating automatic mesh refinement based on error estimations to ensure that higher-order elements are selectively applied where they are most needed.
3. **Numerical Integration Improvements**

- Investigation into more effective numerical integration schemes that can handle the increased complexity of higher-order elements without compromising accuracy.

4. **Application to Multiphysics Problems**:
    1. Extending the use of higher-order finite elements to multiphysics simulations, where interactions between different physical domains are present (e.g., fluid-structure interaction, thermal-electrical analysis).
5. **Error Estimation and Control**:
    1. Enhancement of error estimation techniques that are capable of quantifying the accuracy of higher-order finite element solution.
6. **Software Development and Integration**:
    1. Further development of user-friendly software packages that support higher-order finite elements and their integration into common finite element analysis frameworks.

References:

- [1] Moore, J.L., Morgan, N.R. and Horstemeyer, M.F., 2019. ELEMENTS: A high-order finite element library in C++. *SoftwareX*, *10*, p.100257.
- [2] Logan, D.L., 2016. *A first course in the finite element method*. Cengage Learning.
- [3] Krishnamoorthy, C.S., 1994. *Finite element analysis: theory and programming*. Tata McGraw-Hill Education.
- [4] Ergatoudis, I.R.O.N.S., Irons, B.M. and Zienkiewicz, O.C., 1968. Curved, isoparametric,"quadrilateral" elements for finite element analysis. *International journal of solids and structures*, *4*(1), pp.31-42.
- [5] Dasgupta, G., 2008. Stiffness matrices of isoparametric four-node finite elements by exact analytical integration. *Journal of Aerospace Engineering*, *21*(2), pp.45-50..
- [6] Tran, D.B. and Navrátil, J., 2022. Shear Lag Analysis due to Flexure of Prismatic Beams with Arbitrary Cross-Sections by FEM. *Periodica Polytechnica Civil Engineering*, *66*(1), pp.244-255.
- [7] Odunfa, K.M. and Akinmolayan, O.P., Development of an Interactive Finite Element Solution Module for 2d Stress Problem Analysis Using Isoparametric Element.
- [8] Perumal, L. and Mon, D.T.T., 2011. Finite elements for engineering analysis: a brief review. In *Proceedings of the International Conference on Modeling, Simulation and Control, Singapore*.

Project work with Tentative Experimental / Theoretical Framework:

Activities	Tentative workplan
• Study the finite elements and methods.	June – July
• Formulation of Higher order elements . • Implementation of C++ for efficient solution of complex geometry.	August – December
• Using FEA software to analyse the complex geometry and effective,fast and correct solution. • Validation of the both stress result coming from the FEA software and the developed C++ program.	January-February
• Initialization of thesis writing.	March - April

THANK YOU

```
D for plane stress
(3x3)std::vector<std::vector<double>>
computeDMatrix(double E, double nu) {
double coeff = E / (1.0 - nu * nu);
std::vector<std::vector<double>> D(3,
std::vector<double>(3, 0.0));
D[0][0] = 1.0;    D[0][1] = nu;    D[1][0]
= nu;    D[1][1] = 1.0;    D[2][2] = (1.0 -
nu) / 2.0;        for(int i=0;i<3;i++) {
for(int j=0;j<3;j++) {            D[i][j]
*= coeff;        }    }        return D;}//
Function to transpose a
matrixstd::vector<std::vector<double>>
transposeMatrix(const
std::vector<std::vector<double>>& mat) {
if(mat.empty()) return {};    int rows =
mat.size();    int cols = mat[0].size();
std::vector<std::vector<double>>
trans(cols, std::vector<double>(rows,
0.0));        for(int i=0;i<rows;i++) {
for(int j=0;j<cols;j++) {
trans[j][i] = mat[i][j];        }    }
return trans;}// Function to multiply two
matricesstd::vector<std::vector<double>>
multiplyMatrices(const
```