Modeling of Internal Forces and Deflections in Roof Trusses Using the Finite Element Method

Group 2

Team Members:

23110036	Apoorv Rane
23110043	Arnav Gogate
23110163	Kavya Shah
23110118	Goraksh Bendale

Modeling of Internal Forces and Deflections in Roof Trusses Using FEM

Apoorv Rane[1], **Kavya Shah**[2], **Goraksh Bendale**[3] **and Arnav Gogate**[4]

[a]*Prof. Harmeet Singh*

Abstract—Structural analysis of roof trusses is a critical aspect of civil and mechanical engineering, essential for ensuring the safety, efficiency, and longevity of buildings and infrastructure. Traditional methods of truss analysis often rely on simplifying assumptions that may not capture the full complexity of structural behavior, especially under varied loading conditions. The Finite Element Method (FEM) has emerged as a powerful tool to address these limitations, offering a more comprehensive and accurate approach to structural analysis. This project focuses on the application of FEM to analyze 2D roof trusses, specifically targeting the modeling of internal forces and deflections using beam elements. By leveraging computational methods, we aim to gain deeper insights into the structural behavior of trusses, which are fundamental components in many architectural and engineering designs.

Keywords—*FEM, 2D Truss*

Contents

1. Objective

The objective of this project is to perform a comprehensive analysis of a 2D truss structure using the Finite Element Method (FEM) by developing a python code. The purpose is to understand the mechanical behavior of roof trusses under static loading conditions and validate the results through *ANSYS*. This analysis will involve calculating nodal displacements, stresses in truss members, and reaction forces at supports. The detailed goals are as follows:

1. **Application of FEM Principles**
   - Utilize FEM to solve a structural mechanics problem involving trusses, which are widely used in engineering applications such as bridges, towers, and buildings.
   - Apply the fundamental principles of FEM, including discretization, stiffness matrix formulation, and global matrix assembly, to compute the structural response.
2. **Structural Analysis of Trusses**
   - **Calculate nodal displacements:** Determine how much each node in the truss deforms under applied loads.
   - **Compute axial stresses in each truss member:** Identify whether each element is under tension or compression and ensure it remains within safe limits.
   - **Determine reaction forces:** Evaluate the forces at constrained nodes to confirm equilibrium conditions.
3. **Material Behavior**
   - Incorporate material properties such as Young's modulus (E) and cross-sectional area (A) into the analysis to simulate realistic behavior of the truss elements.
4. **Develop Computational Skills**
   - Implement a Python program for FEM analysis of trusses
5. **Validation and Comparison**
   - Validate the Python-based FEM results by comparing them with simulations performed in *ANSYS*.
   - Analyze discrepancies between numerical and simulation results to identify potential sources of error, such as mesh refinement or numerical approximations.

2. Problem Statement

The project involves the structural analysis of a 2D roof truss system using the Finite Element Method (FEM). Trusses are widely used in engineering applications such as bridges, towers, and building frameworks due to their ability to efficiently transfer loads through axial forces. The goal is to determine the mechanical behavior of the truss under static loading conditions, including nodal displacements, stresses in truss members.

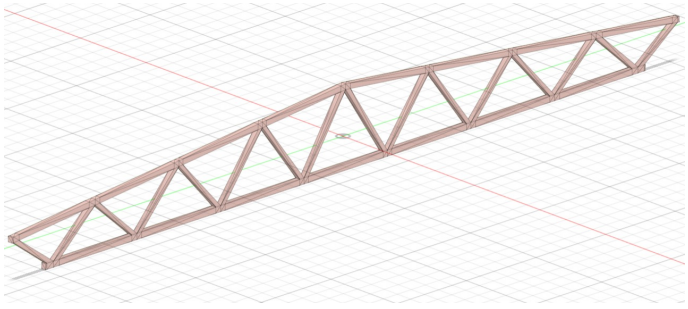
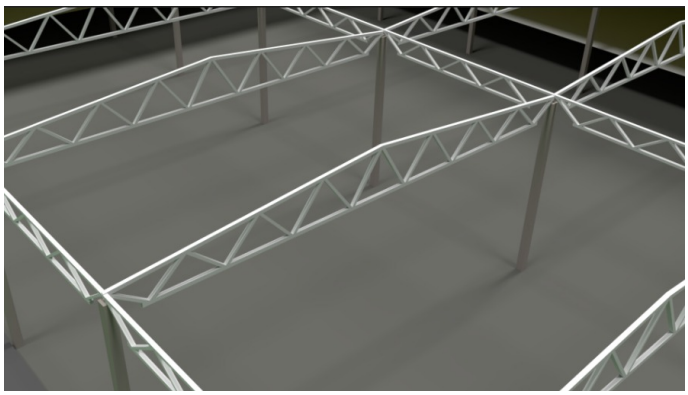


Figure 1. Roof Truss



2.1. Key Details of the Problem

Truss Structure: The truss consists of multiple nodes connected by linear elements (truss members). Each element is modeled as a two-noded linear truss finite element.

Material Properties: Each truss member is characterized by its Young's modulus ($E = 200 \times 10^9$ Pa $= 200$) and cross-sectional area ($A = 0.05\,\text{m}^2$). The material is assumed to remain within its elastic limit under applied loads.

Boundary Conditions: Certain nodes are constrained (fixed or pinned) to simulate supports. External forces are applied at specific nodes to simulate loading conditions.

Degrees of Freedom (DOFs): Each node in the truss has two degrees of freedom: displacement in the X and Y directions. The global stiffness matrix accounts for these DOFs and is reduced by applying boundary conditions. The number of DOFs is twice the number of nodes

Analysis Goals:
- Compute nodal displacements using FEM principles.
- Calculate axial stresses in each truss member based on deformation.

3. Theory

1. Hooke's Law and Strain Energy

The axial force in a bar element is given by Hooke's Law:

$$F = k\Delta x \tag{1}$$

where $k = \dfrac{AE}{L}$, with A = cross-sectional area, E = Young's modulus, L = element length.

The strain energy stored in the element is:

$$u = \int_0^Q kx\,dx = \frac{1}{2}kQ^2 \tag{2}$$

2. Deformation in Local Coordinates

Change in length in local coordinates:

$$Q = q_2' - q_1' \tag{3}$$

Strain energy becomes:

$$u = \frac{1}{2}k(q_2' - q_1')^2 \tag{4}$$

Expanding:

$$u = \frac{1}{2}k\left(q_1'^2 - 2q_1'q_2' + q_2'^2\right) \tag{5}$$

In matrix form, let

$$\boldsymbol{q}' = \begin{bmatrix} q_1' \\ q_2' \end{bmatrix}, \quad \boldsymbol{k}' = \frac{AE}{L}\begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$$

Then:

$$u = \frac{1}{2}\boldsymbol{q}'^T\boldsymbol{k}'\boldsymbol{q}' \tag{6}$$

3. Transformation to Global Coordinates

Global displacements:

$$\boldsymbol{q} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix}$$

Transformation matrix:

$$\boldsymbol{M} = \begin{bmatrix} c & s & 0 & 0 \\ 0 & 0 & c & s \end{bmatrix}, \quad \text{with } c = \cos\theta, \quad s = \sin\theta$$

Then:

$$\boldsymbol{q}' = \boldsymbol{M}\boldsymbol{q}$$

Substitute into energy equation:

$$u = \frac{1}{2}\boldsymbol{q}^T\boldsymbol{M}^T\boldsymbol{k}'\boldsymbol{M}\boldsymbol{q} \tag{7}$$

Define:

$$\boldsymbol{k} = \boldsymbol{M}^T\boldsymbol{k}'\boldsymbol{M} \tag{8}$$

So final global stiffness matrix for a truss element is:

$$\boldsymbol{k} = \frac{AE}{L}\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix}$$

4. Displacement Equation

For the full truss system:

$$\boldsymbol{K}\boldsymbol{Q} = \boldsymbol{F} \tag{9}$$

where:

- $\boldsymbol{K}$ = global stiffness matrix,
- $\boldsymbol{Q}$ = global displacement vector,
- $\boldsymbol{F}$ = global force vector.

5. Stress in the Element

The stress is:

$$\sigma = E \cdot \varepsilon = E \cdot \frac{q_2' - q_1'}{L} \tag{10}$$

In matrix form:

$$\sigma = \frac{E}{L}\begin{bmatrix} -1 & 1 \end{bmatrix}\boldsymbol{q}' = \frac{E}{L}\begin{bmatrix} -1 & 1 \end{bmatrix}\boldsymbol{M}\boldsymbol{q} \tag{11}$$

4. Methodology

The methodology for developing a code and analyzing a truss structure using the Finite Element Method (FEM) involves several systematic steps, which can be divided into pre-processing, processing, and post-processing phases. Below is a detailed explanation of each phase and the underlying principles of FEM applied in this project.

4.1. Pre-Processing

The pre-processing phase involves defining the problem, creating the model, and preparing it for numerical analysis. This includes discretization, defining material properties, and applying boundary conditions.

4.1.1. Step 1: Discretization (Mesh Generation)

The truss structure is divided into smaller elements (finite elements), with nodes at the joints of the truss members. Each element is treated as a 1D linear truss element that can only carry axial forces (tension or compression). The nodes are assigned degrees of freedom (DOFs), typically two per node for 2D trusses:- displacement in the x-direction

(u_x) and y-direction (u_y). The connectivity between nodes and elements is defined in terms of a node-element table, which specifies which nodes are connected by each element.

4.1.2. Step 2: Selection of Element Type

For this project, linear 1D truss elements are used, as they are computationally efficient and suitable for pin-jointed structures.

4.1.3. Step 3: Define Material Properties

Material properties such as Young's modulus (E) and cross-sectional area (A) are defined for all truss members. These properties influence the stiffness of each element.

```
def get_truss_elements():
    n = int(input("Enter the number of truss elements: "))
    elements = []


    for i in range(n):
        x1, y1, x2, y2 = map(float, input(f"Enter x1, y1, x2, y2 for element {i+1}: ").split())
        stress_nodes.append([x1 , y1 , x2 , y2])
        A = 0.0001
        E = 200e9
        L = np.sqrt((x2 - x1)**2 + (y2 - y1)**2)
        elements.append((x1, y1, x2, y2, A, E, L))

    return elements

connectivity = []
    for el in stress_nodes:  # Assuming elements is defined globally or passed
        node1 = coord_to_node[(el[0], el[1])]
        node2 = coord_to_node[(el[2], el[3])]
        connectivity.append((node1, node2))

    print("\nCoordinate to Node Mapping:")
    for coord, node in coord_to_node.items():
        print(f"  Node {node}: {coord}")

    print("\nElement Connectivity (Node Pairs):")
    for i, (n1, n2) in enumerate(connectivity, 1):
        print(f"  Element {i}: Node {n1} Node {n2}")

    return Q, coord_to_node, connectivity
```

Code 1. Code Snippet for Pre-Processing

4.2. Processing

The processing phase involves forming the governing equations of FEM and solving them to obtain displacements, stresses, and reaction forces.

4.2.1. Step 5: Formulate Element Stiffness Matrix

For each element, the stiffness matrix (k_e) is derived using the formula:

$$k_e = \frac{EA}{L}\begin{bmatrix} c^2 & cs & -c^2 & -cs \\ cs & s^2 & -cs & -s^2 \\ -c^2 & -cs & c^2 & cs \\ -cs & -s^2 & cs & s^2 \end{bmatrix}$$

where:

- E: Young's modulus,
- A: Cross-sectional area,
- L: Length of the element,
- $c = \cos(\theta)$ and $s = \sin(\theta)$: Direction cosines.

4.2.2. Step 6: Assemble Global Stiffness Matrix

The global stiffness matrix (K) is assembled by combining all element stiffness matrices. This matrix represents the entire truss structure. The size of K depends on the total number of DOFs in the system.

```
def assemble_global_stiffness(truss_elements, stiffness_matrices):
    all_nodes = []
    for (x1, y1, x2, y2, _, _, _) in truss_elements:
        if (x1, y1) not in all_nodes:
            all_nodes.append((x1, y1))
        if (x2, y2) not in all_nodes:
            all_nodes.append((x2, y2))

    all_nodes = sorted(all_nodes)
    node_index_map = {node: idx for idx, node in enumerate(all_nodes)}

    dof_per_node = 2
    total_dofs = len(all_nodes) * dof_per_node
    K_global = np.zeros((total_dofs, total_dofs))

    #Assembling global matrix
    for e_idx, element in enumerate(truss_elements):
        x1, y1, x2, y2, _, _, _ = element
        k_local = stiffness_matrices[e_idx]
        n1 = node_index_map[(x1, y1)]
        n2 = node_index_map[(x2, y2)]

        # Dof mapping wrt nodes
        dof_map = [
            n1 * dof_per_node, n1 * dof_per_node + 1,
            n2 * dof_per_node, n2 * dof_per_node + 1
        ]

        for i in range(4):
            for j in range(4):
                K_global[dof_map[i], dof_map[j]] += k_local[i, j]

    return K_global, all_nodes

K_global, all_nodes = assemble_global_stiffness(truss_elements, stiffness_matrices)
```

Code 2. Code Snippet for Pre-Processing

4.2.3. Step 7: Apply Boundary Conditions to Reduce DOFs

Modify the global stiffness matrix to account for fixed supports by removing rows and columns corresponding to constrained DOFs. Adjust the force vector to reflect applied loads and support reactions.

```
def apply_boundary_conditions_by_node(K, F, all_nodes):
    dof_per_node = 2
    total_dofs = len(all_nodes) * dof_per_node

    fixed_dofs = []
    print("\n Specify boundary conditions for each node (x/y DOFs)")

    coord_to_node = {coord: idx + 1 for idx, coord in enumerate(all_nodes)}

    for i, (x, y) in enumerate(all_nodes):
        print(f"\nNode {i+1} at ({x:.2f}, {y:.2f})")

        x_fixed = input("  Is X DOF fixed? (y/n): ").strip().lower() == 'y'
        y_fixed = input("  Is Y DOF fixed? (y/n): ").strip().lower() == 'y'
```

```
        if x_fixed:
            fixed_dofs.append(i * 2)      # x DOF
        if y_fixed:
            fixed_dofs.append(i * 2 + 1) # y DOF

    free_dofs = np.setdiff1d(np.arange(
    total_dofs), fixed_dofs)

    K_reduced = K[np.ix_(free_dofs, free_dofs)]
    F_reduced = F[free_dofs]

    Q = np.zeros(total_dofs)
    Q[free_dofs] = np.linalg.solve(K_reduced,
    F_reduced)

    print("\n Displacement Vector (Q):")
    print(Q)
    for i in range(len(all_nodes)):
        print(f"Node {i+1}: Ux = {Q[i*2]:.6f},
    Uy = {Q[i*2+1]:.6f}")
```

Code 3. Code Snippet for Pre-Processing

4.2.4. Step 9: Input the External Nodal Forces

The nodal forces are fed into the algorithm to calculate the nodal displacements.

4.2.5. Step 8: Solve System of Equations

Solve the equation:

$$KQ = F$$

where:

- K: Global stiffness matrix,
- Q: Displacement vector (unknowns),
- F: Force vector (known external loads).

Nodal displacements (Q) are computed using LU decomposition.

4.2.6. Step 9: Compute Element Stresses

Using computed nodal displacements, axial strain (ϵ) and stress (σ) in each element are calculated:

$$\sigma = \frac{E}{L}\begin{bmatrix}-1 & 1\end{bmatrix}\boldsymbol{q}' = \frac{E}{L}\begin{bmatrix}-1 & 1\end{bmatrix}\boldsymbol{Mq} \tag{12}$$

4.3. Post-Processing

The post-processing phase involves interpreting results for meaningful insights.

4.3.1. Step 11: Visualize Results

Nodal displacements are plotted to show how the truss deforms under loading. Axial stresses in each element are displayed, indicating whether members are in tension or compression.

4.3.2. Step 12: Validate Results Using ANSYS

The same truss structure is modeled in ANSYS Workbench for validation. Geometry, material properties, boundary conditions, and loads are replicated. Results from the Python-based FEM implementation (displacements, stresses, reaction forces) are compared with ANSYS outputs to ensure accuracy.

5. Results and Discussion

5.1. FEM Python Results

The complete Python code and analysis can be found in the Google Colab notebook linked below:

Open Google Colab Notebook

Following are the coordinates of all nodes of our truss.

Table 1. Node Coordinates

Node	X (m)	Y (m)
1	-22.436	1.142
2	-21.053	0.100
3	-19.516	1.437
4	-18.066	0.092
5	-16.541	1.737
6	-15.019	0.092
7	-13.537	2.040
8	-12.067	0.091
9	-10.536	2.317
10	-9.075	0.093
11	-7.564	2.043
12	-6.086	0.095
13	-4.586	1.742
14	-3.056	0.098
15	-1.529	1.434
16	0.059	0.100
17	1.363	1.142

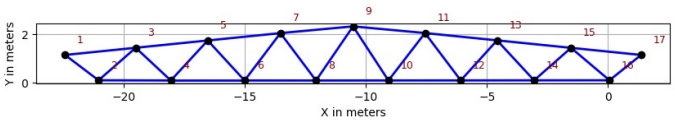


Figure 2. Original Truss

The following material properties were used in the analysis:

- Cross-sectional area, $A = 0.05\,\text{m}^2$
- Young's modulus, $E = 200 \times 10^9\,\text{Pa} = 200\,\text{GPa}$

Table 2. Forces Applied

Node	Force (in N)
3	1000
5	2000
7	2000
9	2000
11	2000
13	1000
15	2000

We have used the truss span to be 23.8 metres in length. Following table has the data for displacement of every node individually. Note that node 1 and node 17 are fixed in both degrees of freedom.

Table 3. Nodal Displacements

Node	X Displacement (m)	Y Displacement (m)
1	0.000000	0.000000
2	-0.000022	-0.000032
3	0.000005	-0.000066
4	-0.000019	-0.000092
5	0.000005	-0.000115
6	-0.000013	-0.000132
7	0.000001	-0.000143
8	-0.000007	-0.000149
9	-0.000004	-0.000150
10	-0.000001	-0.000149
11	-0.000009	-0.000143
12	0.000005	-0.000131
13	-0.000012	-0.000115
14	0.000011	-0.000093
15	-0.000012	-0.000065
16	0.000015	-0.000030
17	-0.000008	0.000000

Using these displacements, we have calculated the stresses and, eventually, the internal forces using the formulae given above.

Table 4. Axial Stresses and Forces in Each Member

Element	Connection	Stress (Pa)	Force (N)
1	1 ↔ 2	172240.70	8612.04
2	2 ↔ 3	-156878.21	-7843.91
3	3 ↔ 4	94232.89	4711.64
4	4 ↔ 5	-88320.51	-4416.03
5	5 ↔ 6	23308.06	1165.40
6	6 ↔ 7	-21320.65	-1066.03
7	7 ↔ 8	-24346.03	-1217.30
8	8 ↔ 9	23107.48	1155.37
9	9 ↔ 10	17057.89	852.90
10	10 ↔ 11	-18053.61	-902.68
11	11 ↔ 12	-28685.83	-1434.29
12	12 ↔ 13	30784.35	1539.22
13	13 ↔ 14	-67793.40	-3389.67
14	14 ↔ 15	75662.34	3783.12
15	15 ↔ 16	-168758.69	-8437.94
16	16 ↔ 17	174156.58	8707.83
17	17 ↔ 15	-136746.24	-6837.31
18	15 ↔ 13	-323845.93	-16192.30
19	13 ↔ 11	-391111.35	-19555.57
20	11 ↔ 9	-397086.47	-19854.32
21	9 ↔ 7	-400835.32	-20041.77
22	7 ↔ 5	-402923.84	-20146.19
23	5 ↔ 3	-326663.86	-16333.19
24	3 ↔ 1	-138265.69	-6913.29
25	2 ↔ 4	255929.28	12796.46
26	4 ↔ 6	385060.45	19253.02
27	6 ↔ 8	413798.89	20689.94
28	8 ↔ 10	386044.04	19302.20
29	10 ↔ 12	406467.67	20323.38
30	12 ↔ 14	368400.48	18420.02
31	14 ↔ 16	265270.69	13263.54

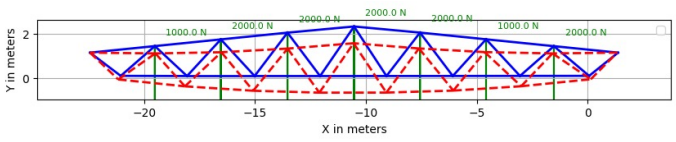


Figure 3. Deflected Truss Configuration with Loading

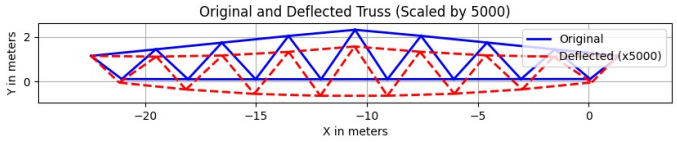


Figure 4. Deflected Truss

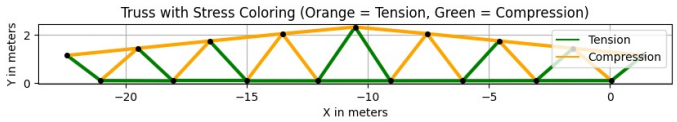


Figure 5. Deflected Truss

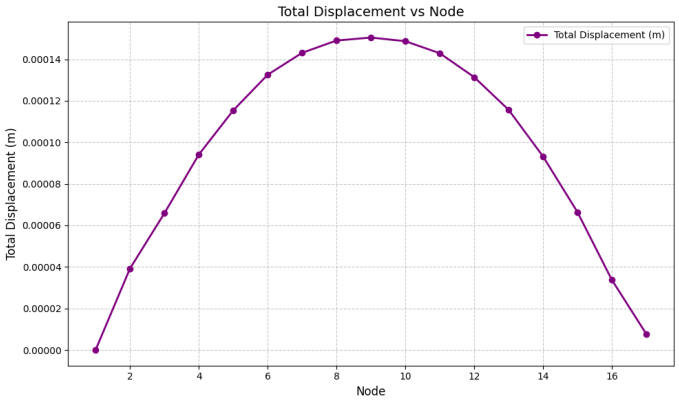


Figure 6. Displacement v/s Node

5.2. Ansys Validation

To validate the Python FEM results, the same truss geometry, material properties, boundary conditions, and loading were modeled in ANSYS

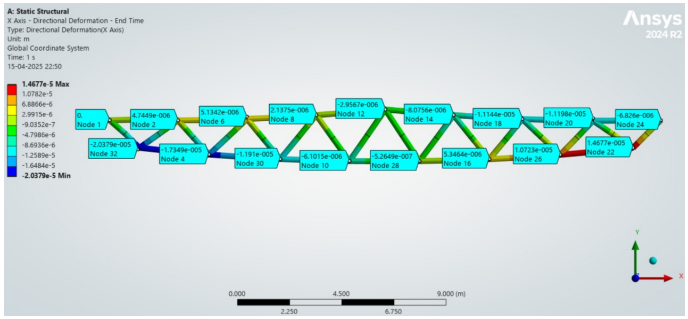


Figure 7. Deflections in X-direction

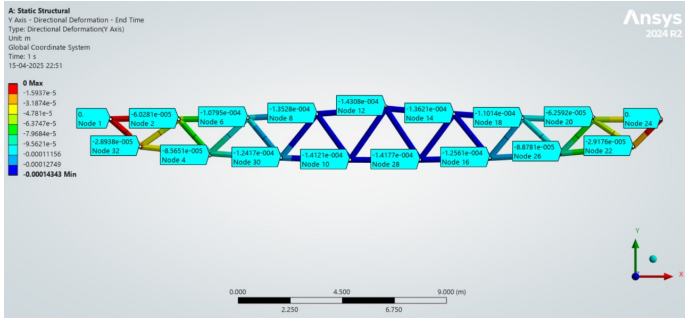


Figure 8. Deflections in Y-direction

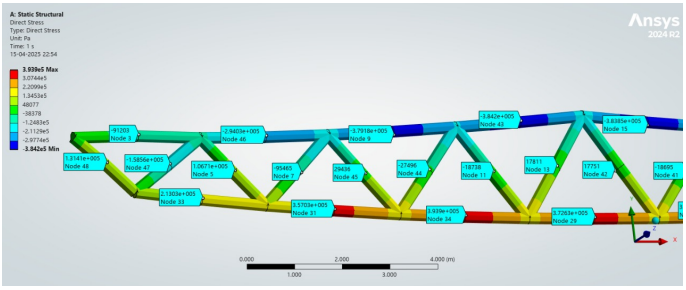


Figure 9. Stresses at each node

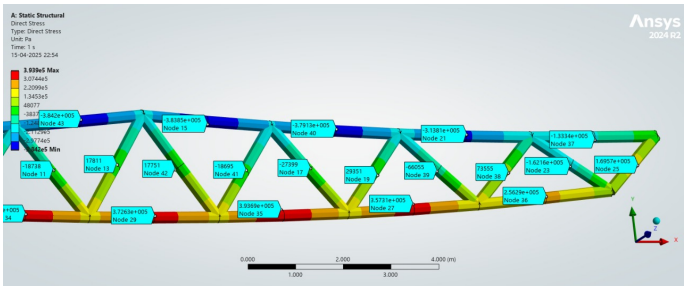


Figure 10. Stresses at each node

One of the most impressive observations that we made was using ANSYS. As we varied the cross-section shape of the elements while keeping the cross-sectional area constant, there was no significant difference in the displacements. This displays the beauty and reliability of this method.

5.3. Deviation from ANSYS results

- The stresses obtained from ANSYS deviated from our code by 10.14 percent on average.
- The x-deformations obtained from ANSYS deviated from our code by 15.61 percent on average.
- The y-deformations obtained from ANSYS deviated from our code by 5.23 percent.

The elements in ANSYS are sub-divided into 2 elements in order to calculate the direct stress in each element at the central node due to axial loading. The same meshing settings are also used to calculate deformations.

6. Conclusion

This project successfully analyzed the structural response of a truss system subjected to static loading using both a custom Python-based Finite Element Method (FEM) implementation and validation through ANSYS Workbench. The study provided a comprehensive understanding of the mechanical behavior of trusses by computing and analyzing nodal displacements, internal axial stresses, and reaction forces at the supports.

The deformation pattern and stress distribution were consistent with engineering principles: members farther from the supports showed the greatest displacements, while internal members displayed alternating tension and compression depending on their orientation and location within the structure. All stress values remained within the elastic limit of the material, affirming that the design is structurally safe under the applied loads.

Python FEM Results

The FEM code developed in Python yielded accurate results, including small nodal displacements and internal stresses distributed in accordance with structural expectations.

Insights and Engineering Implications

- The analysis highlights the importance of evaluating both displacement and internal force distributions when designing truss systems.
- The observed agreement between theoretical expectations, numerical results, and commercial software reinforces the reliability of the FEM approach.
- Minor discrepancies between the two methods stemmed from numerical rounding, solver tolerance settings, and different approaches to enforcing boundary conditions.

Future Scope

This foundational model can be further extended to accommodate:

- Dynamic or time-dependent loading conditions,
- Nonlinear material behavior (e.g., plasticity),
- Thermal expansion effects,
- Parametric studies or structural optimization.

Overall, the methodology employed in this project equips engineers with a deeper understanding of truss behavior and provides a validated framework for analyzing more complex structural systems in future engineering applications.

7. References

Finite Element Truss Notes (UNM) https://www.unm.edu/~bgreen/ME360/Finite%20Element%20Truss.pdf

Wikipedia contributors. (2025). *Finite element method in structural mechanics*. Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Finite_element_method_in_structural_mechanics

Bhattacharjee, S. (2024). *Finite Element Analysis of Solids and Structures* (1st ed.). Routledge.

Nguyen Van Thuan, Ta Duy Hien, Nguyen Duy Hung, Nguyen Trong Hiep, Giap Van Tan. (2023). Finite Element Analysis of a Continuous Sandwich Beam resting on Elastic Support and Subjected to Two Degree of Freedom Sprung Vehicles. *Engineering, Technology Applied Science Research*, 13(2), 10310–10315. https://www.etasr.com/index.php/ETASR/article/view/5464

Wikipedia contributors. (2025). *Finite element method*. Wikipedia, The Free Encyclopedia. Retrieved from https://en.wikipedia.org/wiki/Finite_element_method

Zienkiewicz, O. C., Taylor, R. L. (2005). *The Finite Element Method: Its Basis and Fundamentals* (6th ed.). Elsevier.

Cook, R. D., Malkus, D. S., Plesha, M. E., Witt, R. J. (2002). *Concepts and Applications of Finite Element Analysis* (4th ed.). Wiley.