

PyTruss3D: Development of a Python-Based Finite Element Model of Space Trusses with Visualization of Deformed Shapes

Reymar S. Ledesma^{1,*}, Jomar M. Llanto², Dante L. Silva³, Christ John L. Marcos³

¹College of Engineering, University of Southeastern Philippines, Philippines

²School of Engineering and Architecture, National University, Philippines

³School of Civil, Environmental, and Geological Engineering, Mapúa University, Philippines

Received November 2, 2025; Revised March 2, 2026; Accepted April 14, 2026

Cite This Paper in the Following Citation Styles

(a): [1] Reymar S. Ledesma, Jomar M. Llanto, Dante L. Silva, Christ John L. Marcos, "PyTruss3D: Development of a Python-Based Finite Element Model of Space Trusses with Visualization of Deformed Shapes," *Civil Engineering and Architecture*, Vol. 14, No. 3, pp. 1690 - 1714, 2026. DOI: 10.13189/cea.2026.140321.

(b): Reymar S. Ledesma, Jomar M. Llanto, Dante L. Silva, Christ John L. Marcos (2026). *PyTruss3D: Development of a Python-Based Finite Element Model of Space Trusses with Visualization of Deformed Shapes*. *Civil Engineering and Architecture*, 14(3), 1690 - 1714. DOI: 10.13189/cea.2026.140321.

Copyright©2026 by authors, all rights reserved. Authors agree that this article remains permanently open access under the terms of the Creative Commons Attribution License 4.0 International License

Abstract The extensive review of matrix transformation assembly presents the development of PyTruss3D using the direct stiffness method (DSM), which was validated using the commercial software STAAD.Pro. Grounded in the DSM theory, the global stiffness matrices across five different representative models (ranging from simple to complex structures, such as lattice transmission towers to arched roofs) were formulated from the local elements' stiffness matrices. Prevalently, the program calculates the nodal/joint displacements and axial forces and visualizes deformations while storing the generated matrices. Analyzing the robustness and reliability of the program, the results produced were statistically compared with those from STAAD.Pro using the performance measures Root Mean Square Error (RMSE), correlation coefficient (r), and coefficient of determination (R^2). The statistical comparison yielded excellent agreement with an RMSE below 0.01 and R^2 greater than 0.99 for displacement, and r values ranging from 0.9998 to 1.0 for axial forces. Moreover, PyTruss3D displacement contours replicate the same deformation profiles produced by STAAD.Pro. Slightly robust estimations were observed in the results of the developed program, providing conservative values yet consistent performance. These findings further demonstrate that PyTruss3D can accurately and reliably perform finite element analysis of space trusses, serving as an alternative open-source

program for educational purposes (SDG 4) and structural design validation (SDG 9), as well as for future research.

Keywords Matrix Structural Analysis, Space Truss, Finite Element Analysis, PyTruss3D

1. Introduction

In Engineering disciplines, Finite Element Analysis (FEA) is the backbone of complex structural engineering solutions [1]. While the matrix analysis provides accurate solutions for computing a wide range of structural problems [2], and considering possible iterations dependent on changes in structural plan geometry, structural loads, and material properties, among others, recomputing the involved matrices is time-consuming and computationally demanding [1]. This gap drove software innovations such as STAAD.Pro. It has the capacity to bridge the identified gap, giving efficient and accurate solutions and time-saving capabilities [3]. A recent study focused on two-dimensional trusses, deriving analytical solutions using MATLAB [1], suggesting an exploration of space trusses. While other scholastic findings involve shape and sizing optimization based on a generated algorithm [4] and joint parametrization [5], the developed

program, PyTruss3D ([click link https://drive.google.com/drive/folders/1qQyEQzq8fyEFJAcCbDQyqnRp693sgPNQ?usp=drive_link](https://drive.google.com/drive/folders/1qQyEQzq8fyEFJAcCbDQyqnRp693sgPNQ?usp=drive_link) for the complete Python codes), focuses primarily on integrating DSM ideal for small and large-scale models [6].

Despite the widespread availability, STAAD.Pro conceals the underlying numerical finite element processes that scholars cannot openly modify and explore. In structural engineering, transparency in numerical computations is crucial for a deeper understanding of structural performance [7]. The present study addresses this gap by developing PyTruss3D, a Python-based finite element modeling software that utilizes the Direct Stiffness Method (DSM) [8]. In terms of implementation philosophy, PyTruss3D is capable of computing any type of space truss, including arched domes, transmission towers, and roof trusses. Similarly, the program enables complete transparency in matrix assembly, solution, and deformation visualization. The program aims to serve as an open, educational, and customizable platform for studying finite element behavior and validating results against commercial solvers. PyTruss3D now bridges the theoretical instructions and practical numerical computations in the study of pedagogically oriented Finite Element Methods (FEM).

The main objective of this study is to develop PyTruss3D, a Python-based FEA program utilizing the direct stiffness method as formulated by Kassimali, A. [2], for any space trusses, validated by the comparison of results obtained from the structural software. Specifically, this study aims to:

1. Formulate a global stiffness matrix from a local stiffness matrix using the Python programming language, for select 3D trusses, representing any space truss.
2. Determine the space truss nodal displacement and axial forces of each representative model.
3. Visualize the deformation of each truss based on the applied loads, and
4. Compare the results with the commercially available software, the STAAD.Pro.

This study is limited to modeling, computation, visualization, and validation of any 3D truss structures. The method employs linear static analysis, following the direct stiffness method; therefore, geometric and material nonlinearities are not considered. The program, PyTruss3D, utilizes Python visual and computation libraries (NumPy, Pandas, Matplotlib, etc.). Moreover, the program handles pin-connected members only (limited to carrying axial forces) and negligible bending or shear effects. For qualitative interpretation of the results, a graphical visualization of deformed and undeformed configurations is included. Dynamic, thermal, or nonlinear load effects are outside the scope of this study. Conversely, validation focuses primarily on numerical agreement with STAAD.Pro results for benchmarking truss examples.

2. Conceptual and Analytical Framework

Grounded in the DSM of FEA, the framework illustrates a logical sequence of development and validation of the Python-based programming language for analyzing space trusses (Fig. 1). Solving complex structural problems following numerical procedures in FEM discretizes a structure into smaller elements interconnected at nodes. Moreover, the internal forces and external loads must be in equilibrium for any linear static analysis of the structure, which subsequently calculates the deformation behavior of each element, thereby predicting the overall structure's displacement.

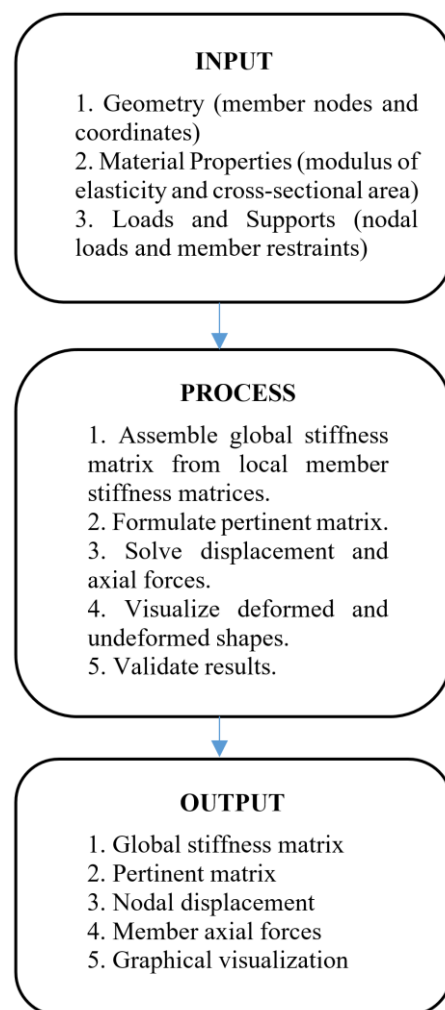


Figure 1. Conceptual framework of the study

This study employs the direct stiffness method, in which each element represents a space truss member separated by nodes. These elements and nodes are then mapped in a three-dimensional coordinate system (an X-Y-Z space).

Furthermore, material properties, particularly the modulus of elasticity (E) and cross-sectional area (A), must also be defined as they affect the structure's behavior. These inputs are needed to address the study's objectives.

Linear static analysis assumes that the structure behaves linearly and stationarily, attributed to the homogeneity of the matrices involved in the study, disregarding the nonlinear behavior of the structure, either global or local. Grounded by the theory [6], [9], [10], [11] presented in Eq. 1, the global stiffness matrix $[K]$ greatly affects the displacement vector $\{u\}$ of the structure when external loads or force vectors $\{F\}$ are applied.

$$[K]\{u\} = \{F\} \tag{1}$$

Attributed to the local member's modulus of elasticity (E), cross-sectional area (A), and length (L), the global stiffness matrix is determined. The member's local stiffness matrix $[k]$ as empirically expressed in Eq. 2 is obtained.

$$[k] = \frac{EA}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \tag{2}$$

Since PyTruss3D reads inputs of node numbers and coordinates, member numbers, E , A , and $\{F\}$, then coordinate transformations from local to global are feasible, as are the stiffness transformations. Consider first the figures where the local displacement and forces in arbitrary member, m , is reflected from the local coordinate system

(LCS) (Fig. 2) and subsequently to the global coordinate system (GCS) (Fig. 3). Local forces, Q_1 and Q_2 in LCS, represent the member forces, respectively, at the beginning and end nodes, with their corresponding u_1 and u_2 displacements. Equivalently, these are noted by $F_1, F_2,$ and F_3 for Q_1 ; $F_4, F_5,$ and F_6 for Q_2 ; $v_1, v_2,$ and v_3 for u_1 ; and $v_4, v_5,$ and v_6 for u_2 . These global equivalents correspond to the X, Y, and Z components of local forces and displacements.

On the other hand, the member transformation matrix is governed exclusively by the angular relationships between the local and global coordinate axes, regardless of whether the origins of the LCS and GCS coincide. In reference to Fig. 3, let b and e have global coordinates (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) , respectively. As argued, the local member's direction cosines are expressed in Eqs. 3, 4, and 5.

$$\cos \theta_{global\ x} = \frac{X_2 - X_1}{L} \tag{3}$$

$$\cos \theta_{global\ y} = \frac{Y_2 - Y_1}{L} \tag{4}$$

$$\cos \theta_{global\ z} = \frac{Z_2 - Z_1}{L} \tag{5}$$

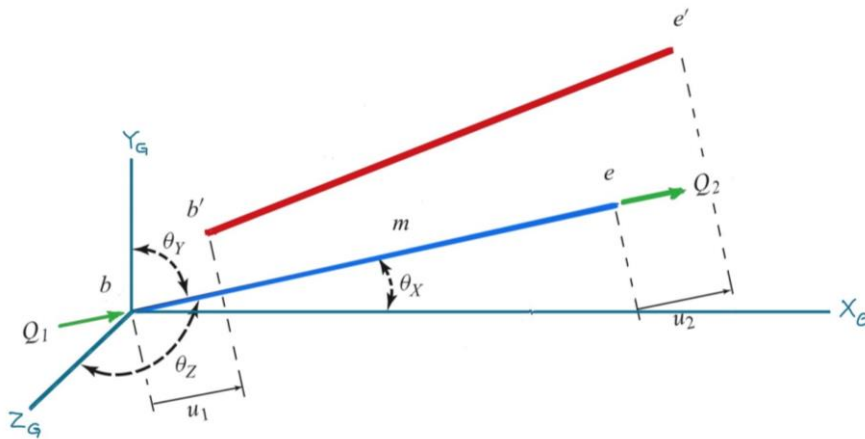


Figure 2. Arbitrary member behavior in LCS

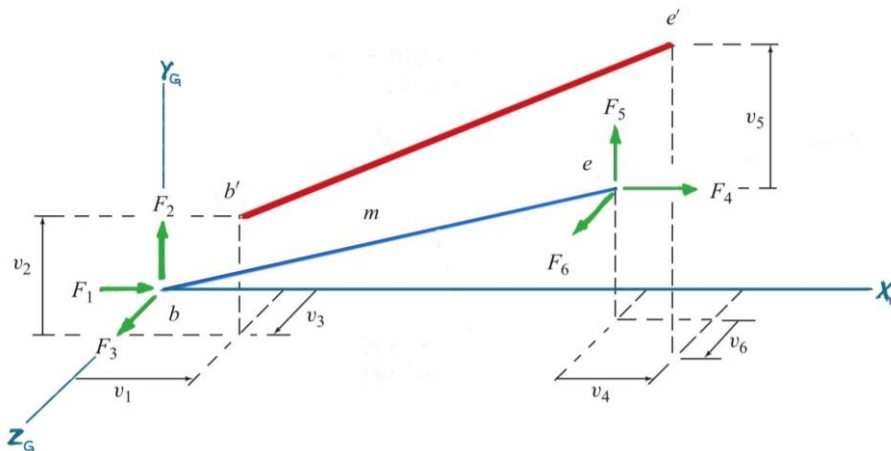


Figure 3. Arbitrary member behavior in GCS

Apparently, space truss displacement and member forces are derived from global to local systems. In a similar arbitrary representation of the space truss member in Fig. 2 and Fig. 3, member forces Q_1 and Q_2 are determined as presented in Eq. 6 relative to the product of direction cosines and external forces applied at the beginning and end nodes.

$$\begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} = \begin{bmatrix} \cos \theta_{gX} & \cos \theta_{gY} & \cos \theta_{gZ} & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos \theta_{gX} & \cos \theta_{gY} & \cos \theta_{gZ} \end{bmatrix} \begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} \quad (6)$$

Similarly, space truss member displacements are computed in the same manner as the local forces. The same transformation matrix of direction cosines is used to transform behavior from global to local sense. Eq. 7 presents this relationship.

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos \theta_{gX} & \cos \theta_{gY} & \cos \theta_{gZ} & 0 & 0 & 0 \\ 0 & 0 & 0 & \cos \theta_{gX} & \cos \theta_{gY} & \cos \theta_{gZ} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} \quad (7)$$

In the same manner, since end displacements and forces are in vectors (defined in the same directions), letting \mathbf{T} be the transformation matrix direction cosines, Eq. 8 is derived.

$$\begin{bmatrix} F_1 \\ F_2 \\ F_3 \\ F_4 \\ F_5 \\ F_6 \end{bmatrix} = [\mathbf{T}^T] \begin{bmatrix} Q_1 \\ Q_2 \end{bmatrix} \quad (8)$$

Explicitly, the global stiffness matrix is given by Eq. 9. The derived Eq. 2 and the defined transformation matrix, \mathbf{T} , result in an expanded member global stiffness matrix, \mathbf{K}_i , shown in Eq. 10. This expansion produces a 6x6 matrix for each space truss member/element. Conversely, each element differs by node number and coordinates; the first three columns of matrix \mathbf{K}_i correspond to the beginning node x-y-z components, and the last three columns of matrix \mathbf{K}_i correspond to the end node x-y-z components. The same index is assigned to rows 1 through 6. These assignments are used in crafting the structure's global stiffness matrix, \mathbf{K} , and subsequently, the formulation of the pertinent matrix, \mathbf{S} .

$$\mathbf{K}_i = [\mathbf{T}^T] [\mathbf{k}] [\mathbf{T}] \quad (9)$$

$$\mathbf{K}_i = \frac{EA}{L} \begin{bmatrix} \cos^2 \theta_{gX} & \cos \theta_{gX} \cos \theta_{gY} & \cos \theta_{gX} \cos \theta_{gZ} & -\cos^2 \theta_{gX} & -\cos \theta_{gX} \cos \theta_{gY} & -\cos \theta_{gX} \cos \theta_{gZ} \\ \cos \theta_{gX} \cos \theta_{gY} & \cos^2 \theta_{gY} & \cos \theta_{gY} \cos \theta_{gZ} & -\cos \theta_{gX} \cos \theta_{gY} & -\cos^2 \theta_{gY} & -\cos \theta_{gY} \cos \theta_{gZ} \\ \cos \theta_{gX} \cos \theta_{gZ} & \cos \theta_{gY} \cos \theta_{gZ} & \cos^2 \theta_{gZ} & -\cos \theta_{gX} \cos \theta_{gZ} & -\cos \theta_{gY} \cos \theta_{gZ} & -\cos^2 \theta_{gZ} \\ -\cos^2 \theta_{gX} & -\cos \theta_{gX} \cos \theta_{gY} & -\cos \theta_{gX} \cos \theta_{gZ} & \cos^2 \theta_{gX} & \cos \theta_{gX} \cos \theta_{gY} & \cos \theta_{gX} \cos \theta_{gZ} \\ -\cos \theta_{gX} \cos \theta_{gY} & -\cos^2 \theta_{gY} & -\cos \theta_{gY} \cos \theta_{gZ} & \cos \theta_{gX} \cos \theta_{gY} & \cos^2 \theta_{gY} & \cos \theta_{gY} \cos \theta_{gZ} \\ -\cos \theta_{gX} \cos \theta_{gZ} & -\cos \theta_{gY} \cos \theta_{gZ} & -\cos^2 \theta_{gZ} & \cos \theta_{gX} \cos \theta_{gZ} & \cos \theta_{gY} \cos \theta_{gZ} & \cos^2 \theta_{gZ} \end{bmatrix} \quad (10)$$

3. Methods

3.1. Simulation Models and Data Preprocessing

This study investigated five (5) space truss models to determine the extent of the program, PyTruss3D, in terms of computational capacity and results accuracy (Figs. 4, 5, 6, 7, and 8). Reflected in each model are the number of nodes, restraints, and members, varying from smaller to larger values as indicated. The increasing values of truss

specifications were done to test the capability of PyTruss3D to compute larger matrices. Furthermore, as part of the data input preparation, each model's member numbers, beginning and end nodes, nodal coordinates, nodal global forces, modulus of elasticity, and cross-sectional area were aligned in Excel (.csv) format as presented in Table 1. This will be loaded to Spyder using pandas. Each row of the input data corresponds to the details of a single element in the 3D truss model, along with its corresponding data type.

Space Truss Structure with Applied Loads Model 1

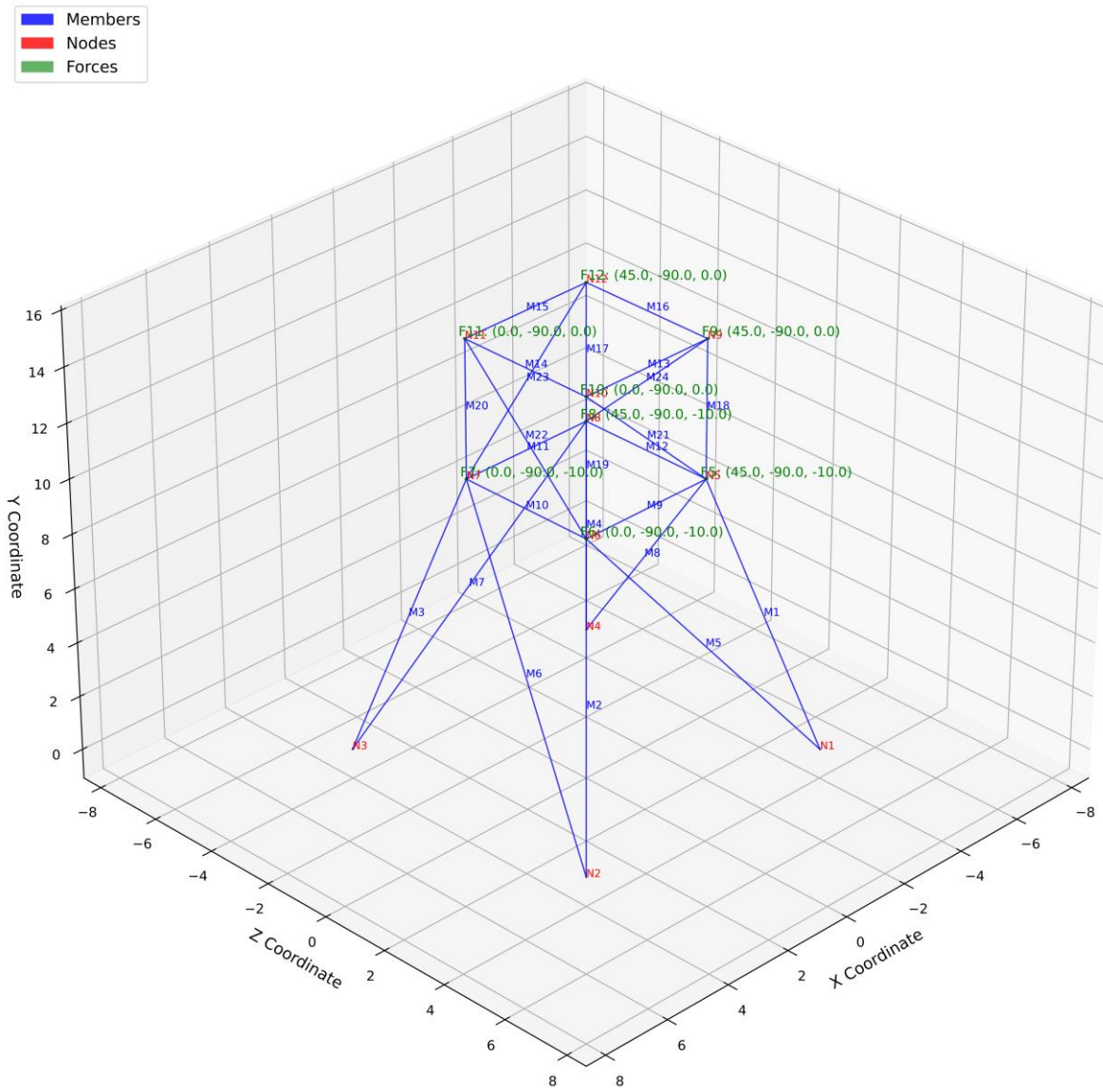


Figure 4. Model 1: Simple space truss (12 nodes, 12 restraints, 24 members)

Space Truss Structure with Applied Loads Model 2

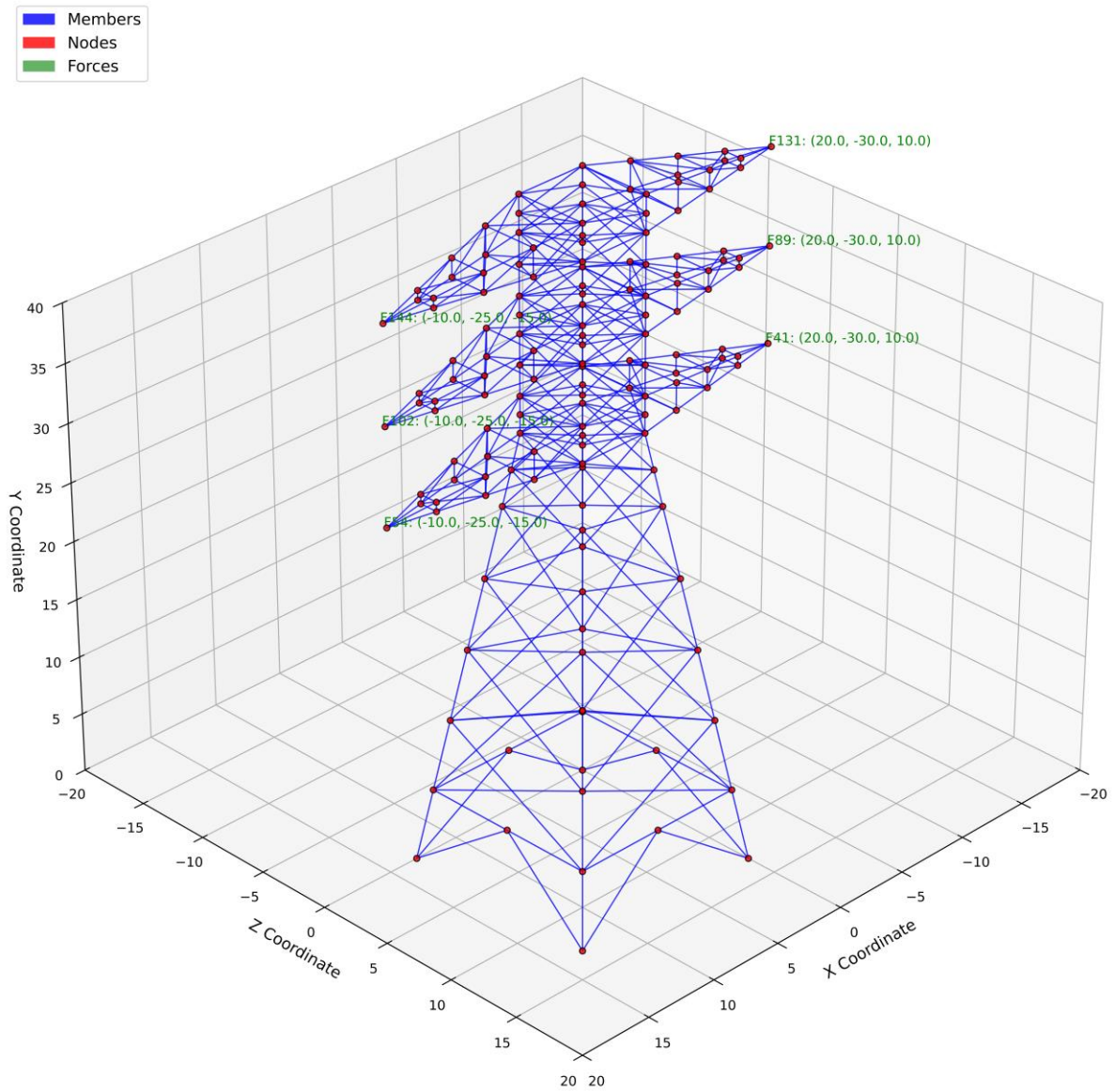


Figure 5. Model 2: Lattice transmission tower model (154 nodes, 12 restraints, 476 members)

Space Truss Structure with Applied Loads Model 3

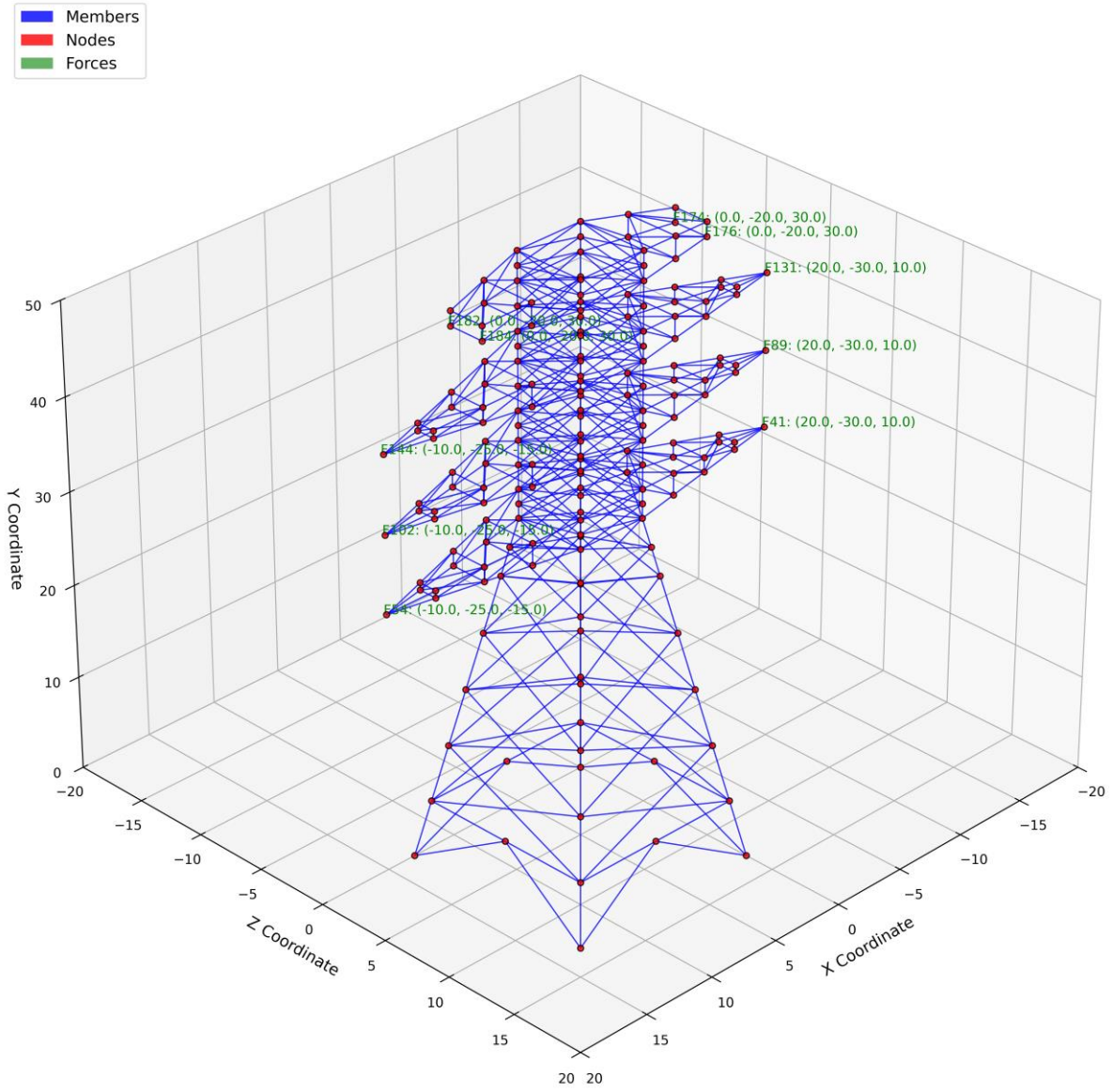


Figure 6. Model 3: Lattice transmission tower model (186 nodes, 12 restraints, 580 members)

Space Truss Structure with Applied Loads Model 4

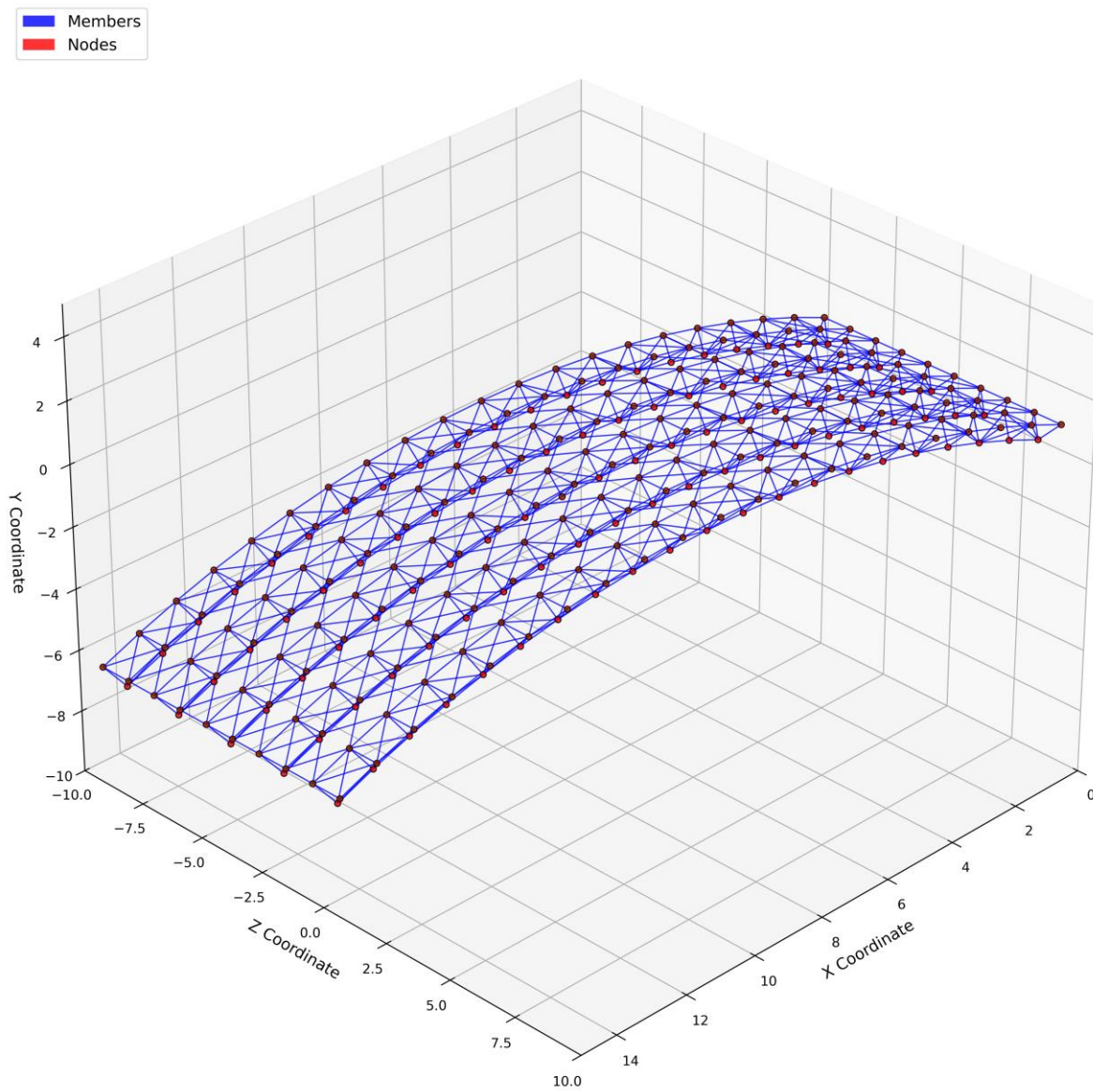


Figure 7. Model 4: Arched roof truss model (315 nodes, 30 restraints, 1083 members)

Space Truss Structure with Applied Loads Model 5

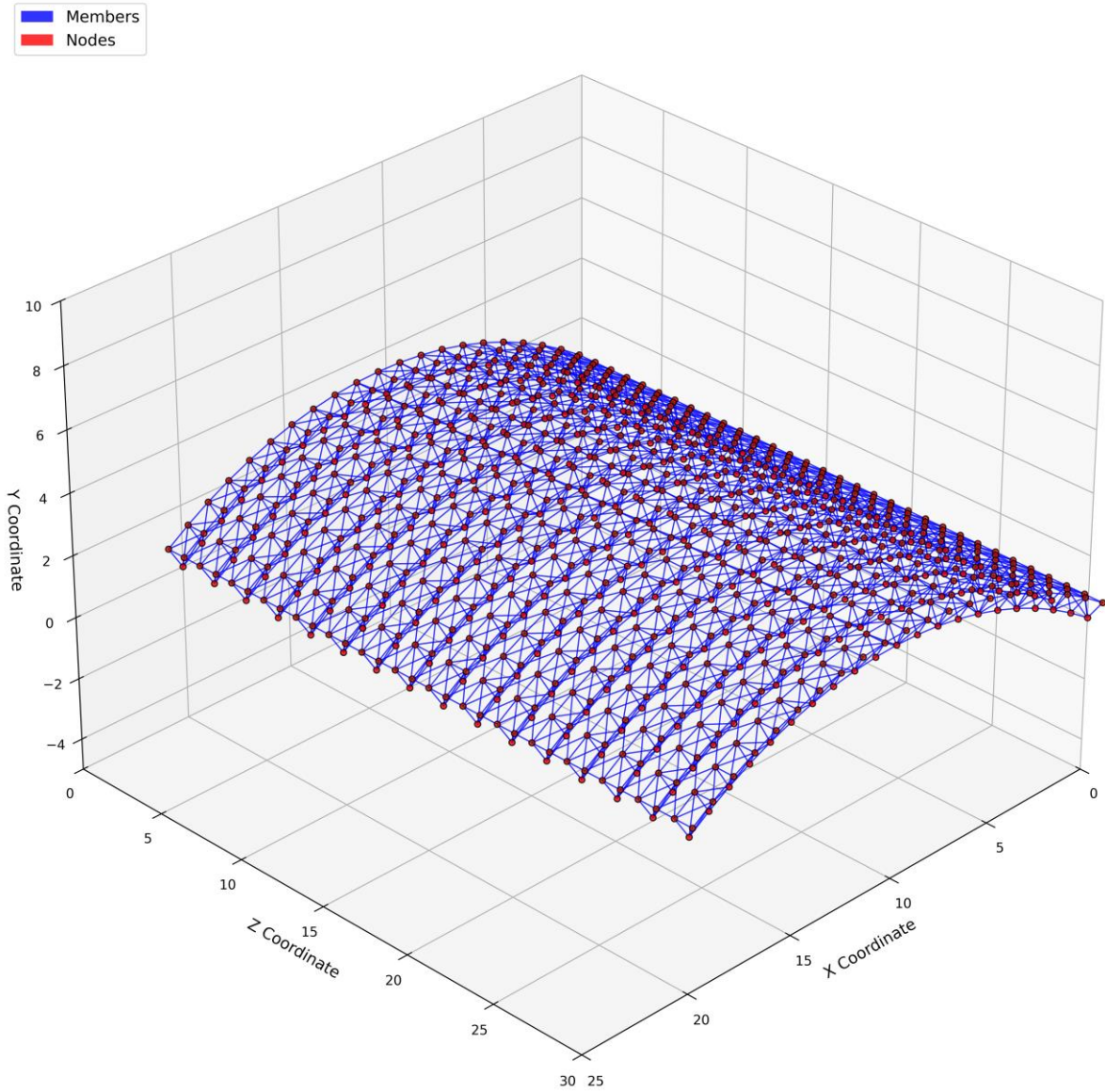


Figure 8. Model 5: Arched roof truss model (1008 nodes, 96 restraints, 3558 members)

Table 1. Data Preparation

Input Data	Member Constant Data			Node ID		Start Node Coordinates			End Node Coordinates			Start Node Forces			End Node Forces		
	Member ID	E	A	Start_node	End_node	x1	y1	z1	x2	y2	z2	Fx1	Fy1	Fz1	Fx2	Fy2	Fz2
Save as .csv	[int]	[flt]	[flt]	[int]	[int]	[int]	[int]	[int]	[int]	[int]	[int]	[flt]	[flt]	[flt]	[flt]	[flt]	[flt]

[int] – integer type

[flt] – float type

Global nodal forces were considered in the analysis. For Model 1, forces $F_x = 45 \text{ kN}$, $F_y = -90 \text{ kN}$, and $F_z = -10 \text{ kN}$ were applied on the nodes as illustrated in Fig. 4. Model 2 and Model 3 forces varies from -10 kN to 20 kN for F_x , -10 kN to -30 kN for F_y , and -15 kN to 10 kN for F_z were also applied as shown in Fig. 5 and Fig. 6, respectively.

Moreover, forces in Model 4 and Model 5 were applied in all the nodes at the top of the arched trusses. These forces amounted to 10 kN, -15 kN, and -30 kN, respectively, for F_x , F_y , and F_z in Model 4. Correspondingly -10 kN, -15 kN, and 20 kN were the nodal forces loaded for Model 5. It is to be noted that these forces are the theoretical assumptions set by the researchers. Since PyTruss3D limits the application of loads to the nodes of the structures, lateral forces, prior to the application to the model, should be converted to their equivalent nodal forces utilizing the positive and negative signs to account for the direction.

3.2. Global Matrix Assembly

Predominantly, the global matrix assembly is prevalent prior to the formulation of the pertinent matrix. Using the Python programming language, the local member global stiffness matrix was computed using Eq. 10. To ensure that each node index is unique, the node number assigned is multiplied by three, yielding the corresponding x, y, and z components of each node. Using Python, these indices are deducted by one as Python indices start with 0. Given this fact, the overall global stiffness matrix of the structure has a size three times the maximum node identified in the model. This phenomenon suggested that nodes were numbered consecutively from 1. Assigning the elements' global matrix to the structure's global matrix (GK), the following codes were considered.

```
for i in range(6): # Rows of element's global matrix
    global_i = dof_indices[i]
    for j in range(6): # Columns element's global matrix
        global_j = dof_indices[j]
        GK[global_i, global_j] += k_local[i, j]
```

3.3. Pertinent Matrix Assembly

Considering the structures as space trusses, the support restraints were accounted for as pin-connected, having three degrees of freedom (DOF) in translation (F_x , F_y , and F_z), which restrain forces along the three axes. Prevalently, at the support, the displacements are considered to be 0. From the structure's global stiffness matrix, a pertinent matrix, S , was formed for each model. This pertinent matrix served as a submatrix from GK to be used in the computations of member displacements, utilizing the concept of Eq. 1. Moreover, for PyTruss3D to read the

restraints, joints with supports are assigned with the lowest node numbers (i.e., starting nodes 1, 2, 3, ...). Hence, the pertinent matrix has the size and values, $S = \text{GK}[-\text{NDOF}, -\text{NDOF}]$, as the lower square matrix of GK, where NDOF is the number of degrees of freedom of the structure accounting for the number of restraints and number of joints/nodes.

3.4. Displacement and Axial Forces

From Eq. 1, the structure displacement can be determined by multiplying the inverse of the structure's stiffness matrix by the applied external forces. Utilizing this concept, the unrestrained joint/nodal displacement was calculated as the product of the inverse of the pertinent matrix by the pertinent applied loads. Using node indices, each element's global displacements (v_i) were determined. Locally, using Eq. 7, the member-specific displacements were formed, that is, $[u_i] = [T_i][v_i]$.

In analyzing the axial force of each member, Q_i , concept of Eq. 1 was applied, that is, solving the product of the stiffness matrix computed by Eq. 2 and the previously formed u_i . Here, the first element of Q_i served as the basis of identifying whether the member is in tension or compression. Furthermore, to determine the global projection of the member end forces, Eq. 8 can be applied.

3.5. Visualization of Deformed and Undeformed Models

PyTruss3D can also graphically present the space truss model, allowing for the visualization of both undeformed and deformed shapes when external loads are applied. Utilizing Python libraries such as Matplotlib and NumPy, PyTruss3D reflected the maximum compressive and tensile forces, maximum node displacement, and the globally displaced models relative to their undeformed shapes.

3.6. Validation of Results

Validating the results of each model, member by member, is essential to measure the accuracy of the developed program. Some of the measures used to compare the results are the Root Mean Square Error (RMSE), the Correlation Coefficient (r), and the Coefficient of Determination (R^2). Additionally, to graphically compare the results and identify potential biases, a linear regression analysis is conducted. RMSE is used in engineering simulations [12] when comparing models, and the researcher used this value to assume that larger errors are undesirable. The correlation coefficient also measures the variation of the predicted (PyTruss3D) and the actual (STAAD.Pro) results based on trends. This variability is evaluated using the R-squared value.

4. Results and Discussion

It is prevalent that the extent of use of the program, PyTruss3D, should be measured. Comparing the program’s results to those of commercial software (STAAD.Pro) means validating the program’s accuracy in computing larger-scale matrices.

4.1. Displacement and Axial Forces

Essentially, all member responses are compared in terms of displacement and axial forces, using the statistical error and fit measures (RMSE, r , and R^2) as the basis for comparison. Table 2 presents the statistical results of the five (5) trial models: Model ID M1, M2, M3, M4, and M5 correspond to Model 1, Model 2, Model 3, Model 4, and Model 5, respectively. These comparisons serve as validation of the developed PyTruss3D program in terms of joint displacements. Based on the table, M1, having the simplest geometry and truss configurations, yielded no discrepancy in displacement results between the program and the commercial software, with r -squared values of 1.0. Similarly, all models yielded RMSE values of less than 0.01, ranging from 0 to 0.002, while the r and r -squared values ranged from 1.0 to 0.9922. The comparison results indicate an excellent agreement between PyTruss3D and STAAD.Pro displacement results. Thus, the results reveal that the developed program can be used to validate any

space truss modelled in STAAD.Pro, and vice versa, for displacement checks and validations.

Table 2. Comparison of the Results of Nodal Displacements

Joint Displacement	PyTruss3D vs. STAAD.Pro		
	RMSE	Coeff. of Correlation (r)	Coeff. of Determination (R^2)
M1	0.00	1.00	1.00
M2	0.0006	0.9990	0.9979
M3	0.0007	0.9990	0.9980
M4	0.0018	0.9961	0.9922
M5	0.0020	0.9972	0.9944

Moreover, Figs. 9-13 illustrate the node-to-node comparison of joint displacements for M1-M5, respectively. Provided in each figure is the regression line between the data and the ideal equation of $y = x$. All the models were observed to have a y -intercept of zero in the best-fit equation, indicating that there is no constant bias between the results. Since the slope varies by only 2.7% more, this trend suggests that for up to 1008 nodes, 96 restraints, and 3558 members, the PyTruss3D program slightly overestimates the generated displacement compared to the benchmark software.

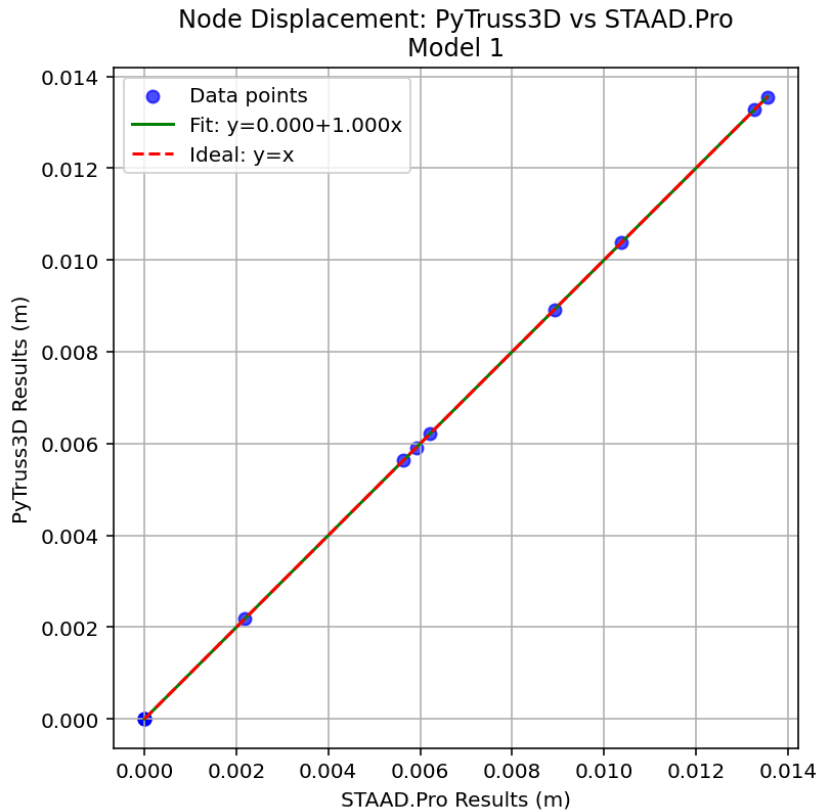


Figure 9. Model 1 fit of PyTruss3D vs. STAAD.Pro displacement results

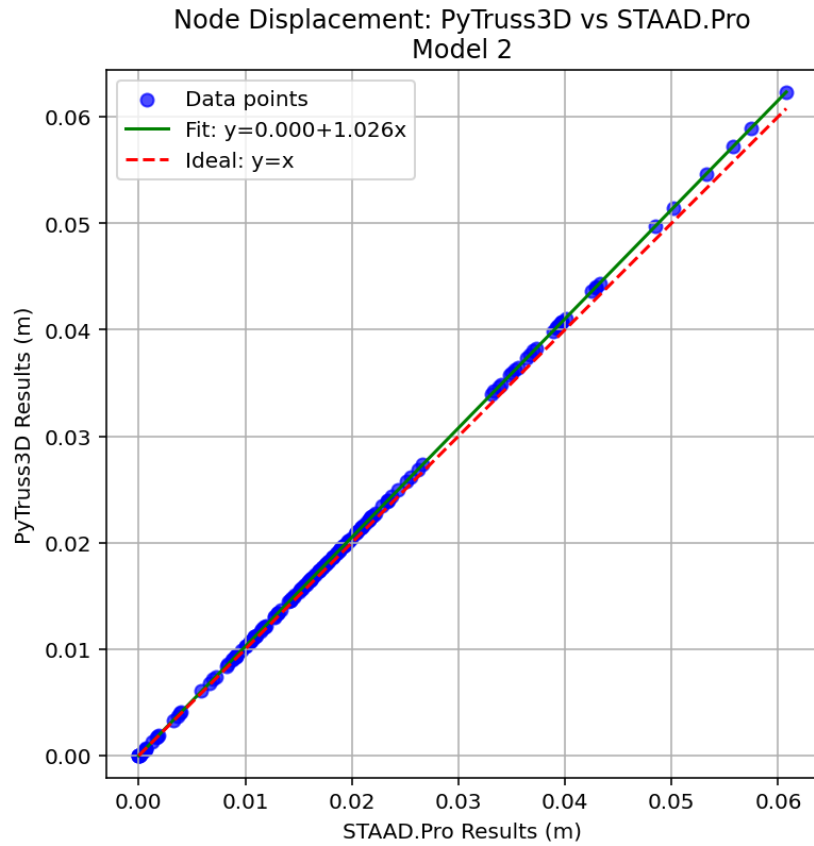


Figure 10. Model 2 fit of PyTruss3D vs. STAAD.Pro displacement results

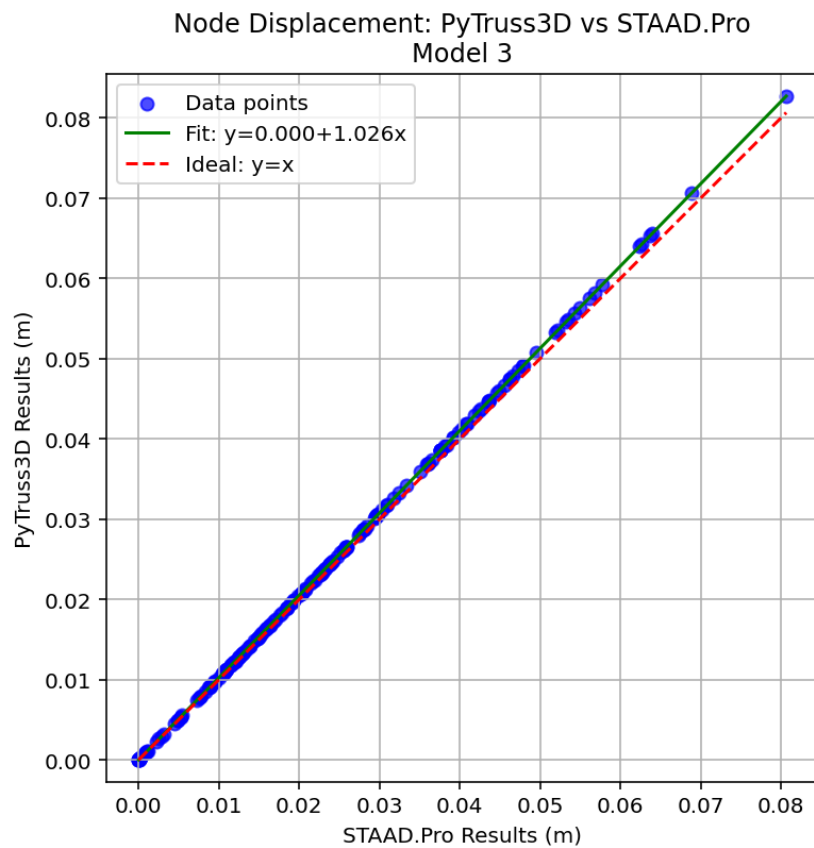


Figure 11. Model 3 fit of PyTruss3D vs. STAAD.Pro displacement results

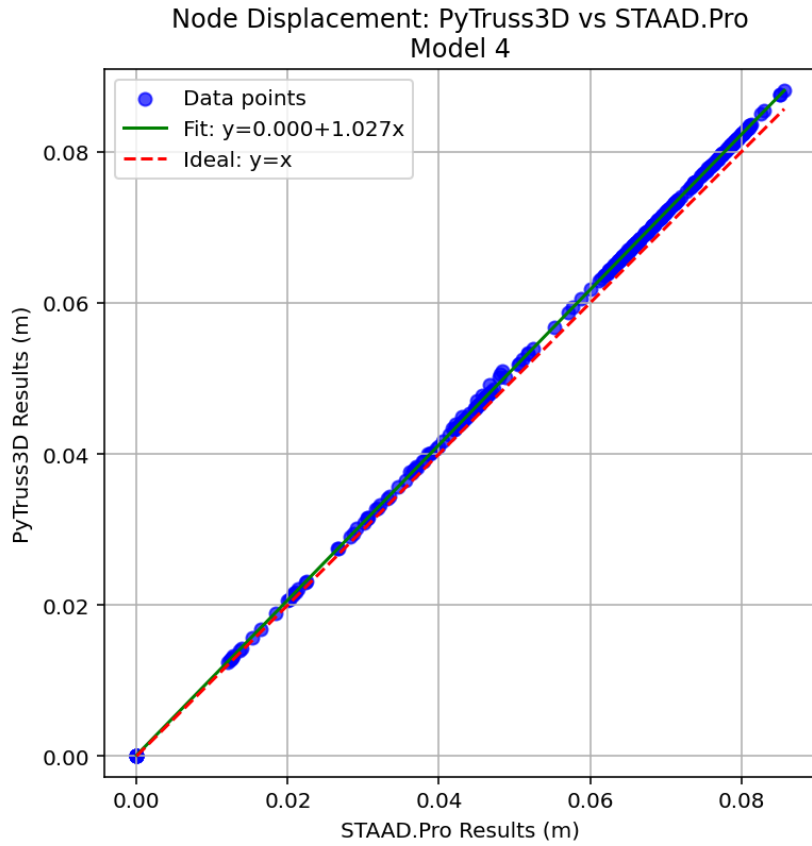


Figure 12. Model 4 fit of PyTruss3D vs. STAAD.Pro displacement results

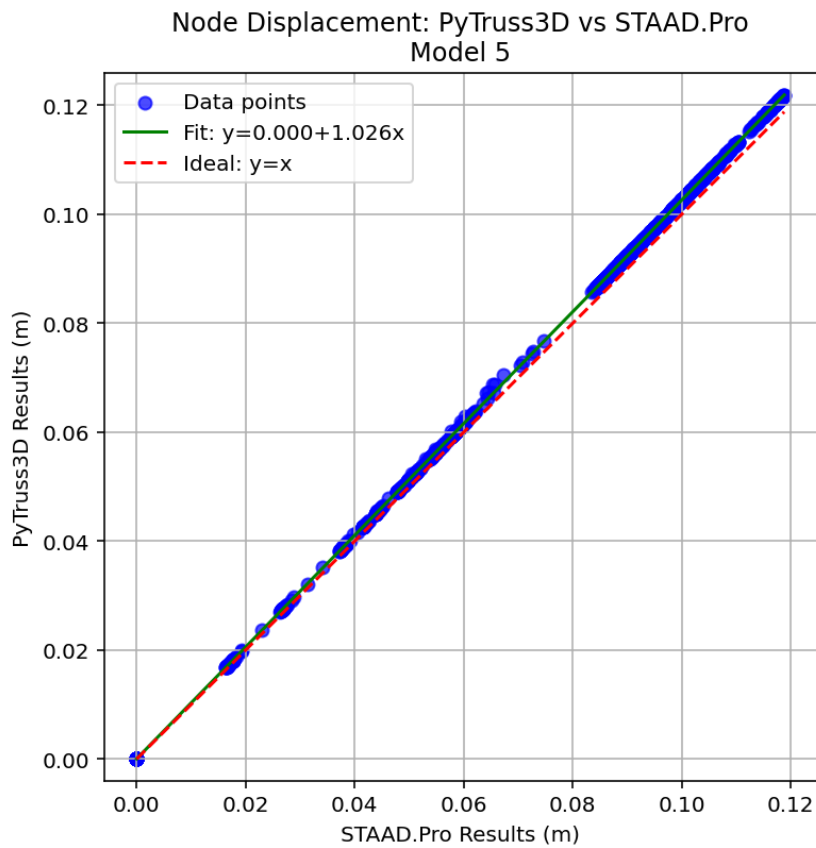


Figure 13. Model 5 fit of PyTruss3D vs. STAAD.Pro displacement results

Table 3. Comparison of the Results of Axial Forces

Axial Forces	PyTruss3D vs. STAAD.Pro		
	RMSE	Coeff. of Correlation (r)	Coeff. of Determination (R ²)
M1	0.0002	1.00	1.00
M2	0.0516	1.00	1.00
M3	0.0732	1.00	1.00
M4	3.9668	0.9999	0.9998
M5	2.7927	0.9999	0.9999

In contrast, Table 3 presents the statistical comparison of the axial force results. It can be observed that all models provided an excellent correlation ($r = 0.9998-1.0$) between the program and the benchmark software. In terms of RMSE, M1-M3 (up to 186 nodes, 12 restraints, 580 members) granted a less than 1.0, indicating practically identical results of axial forces generated. However, Model 4 and Model 5 generated RMSE values of 3.9668 and 2.7927, respectively. Although within an acceptable range

in engineering simulations, these values indicate variations in axial force of approximately 4 kN on average, which is relatively small compared to the forces of M4 and M5, ranging from 1100 kN to 1600 kN. Combined with the coefficient of determination, these values indicate an excellent correlation and minimal numerical errors.

Consistent with the findings of the generated displacements, the statistical comparison of the structure's element axial forces provided the same trend and coefficient of determination as depicted in Figs. 14-18, for M1-M5, respectively. Space trusses with up to 186 nodes, 12 restraints, 580 members (M1-M3) provided negligible biases and slope differences in terms of axial forces generated, as shown in Figs. 14-16. On the other hand, it was found that M4 and M5 (Fig. 17 and Fig. 18) exhibit biases of 0.166 kN and 0.022 kN, respectively, but produce negligible slope differences, analogous with the RMSE and R² results. With a slight average overestimation, PyTruss3D axial forces are consistent with the generated STAAD.Pro results.

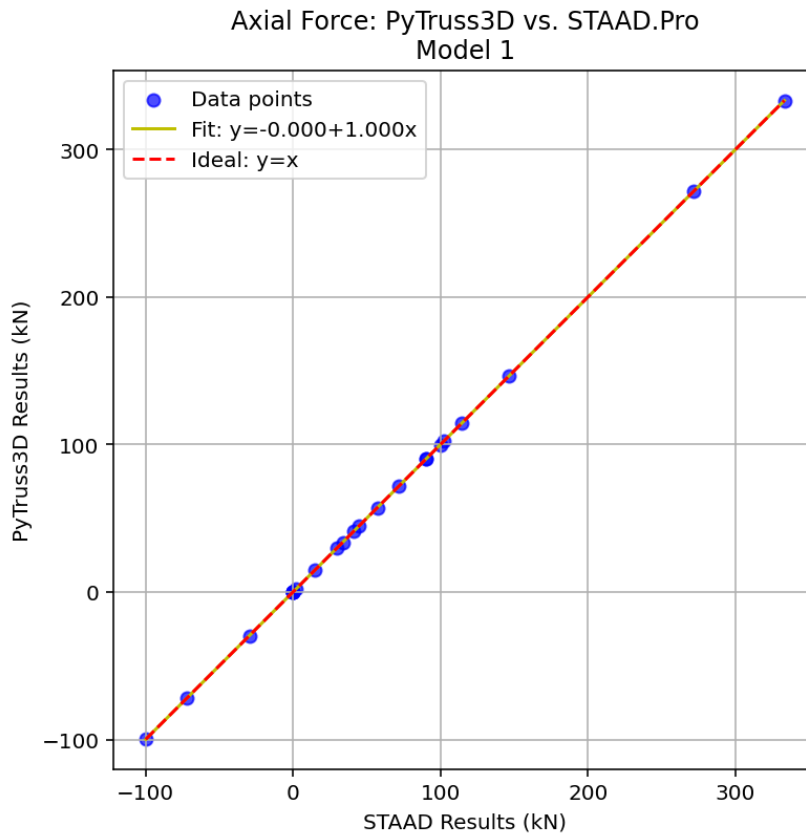


Figure 14. Model 1 fit of PyTruss3D vs. STAAD.Pro axial force results

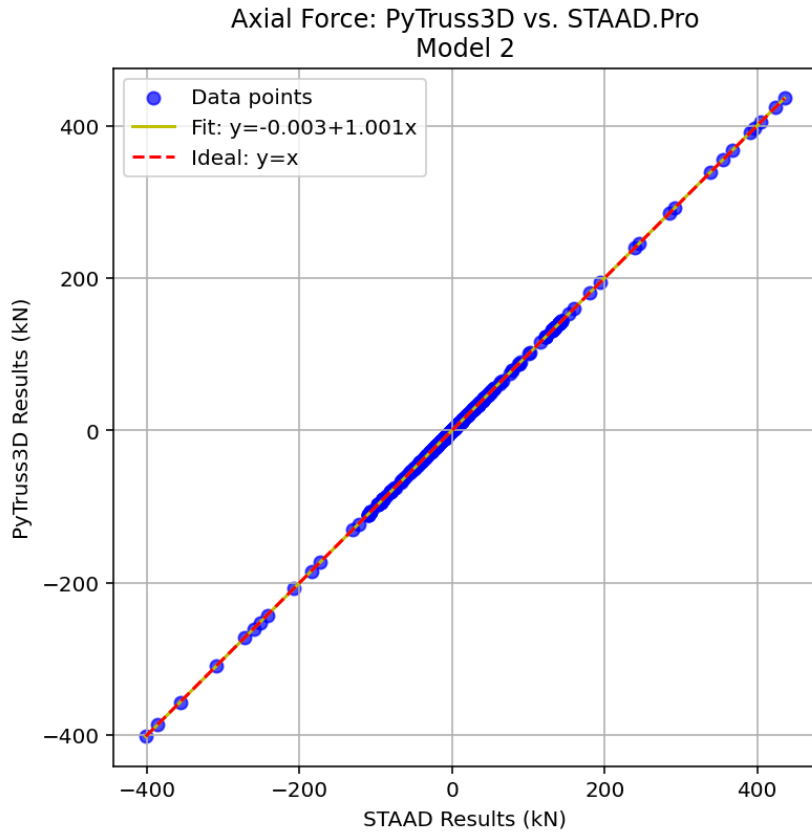


Figure 15. Model 2 fit of PyTruss3D vs. STAAD.Pro axial force results

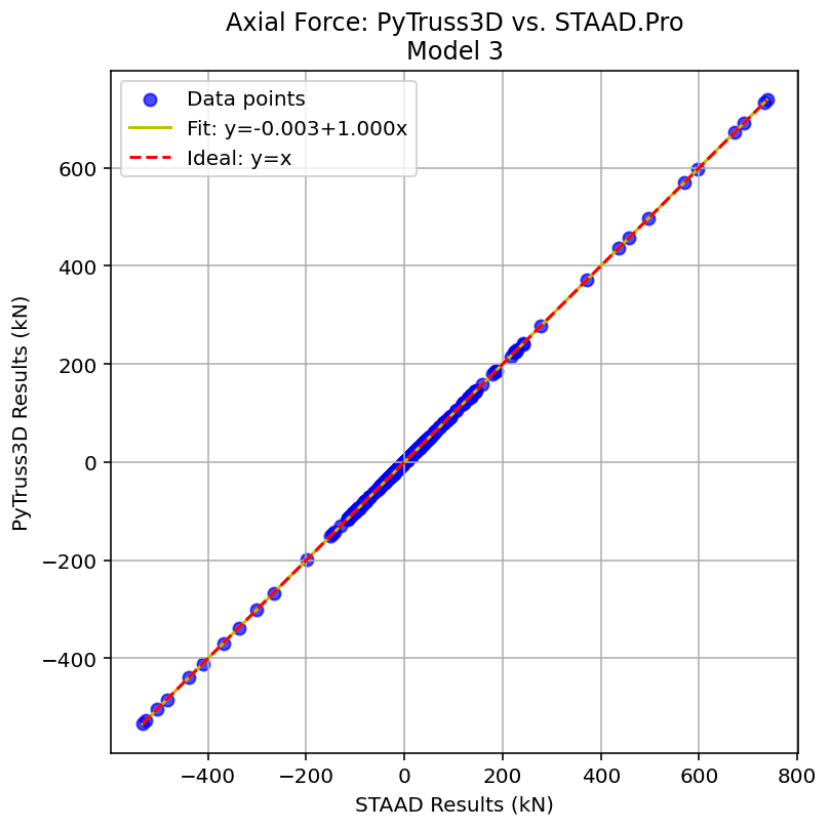


Figure 16. Model 3 fit of PyTruss3D vs. STAAD.Pro axial force results

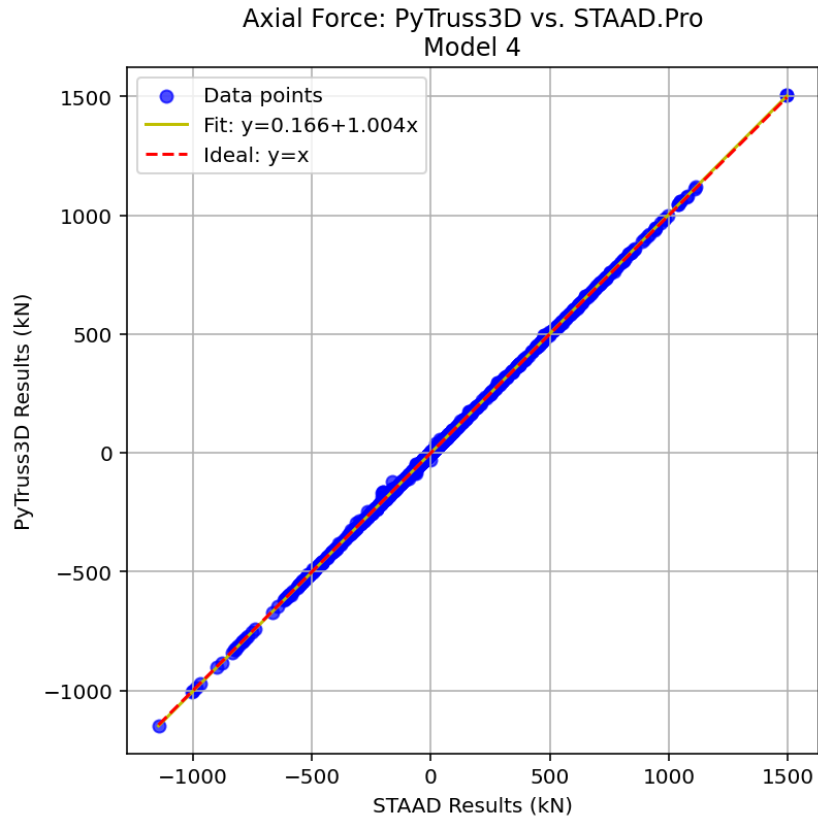


Figure 17. Model 4 fit of PyTruss3D vs. STAAD.Pro axial force results

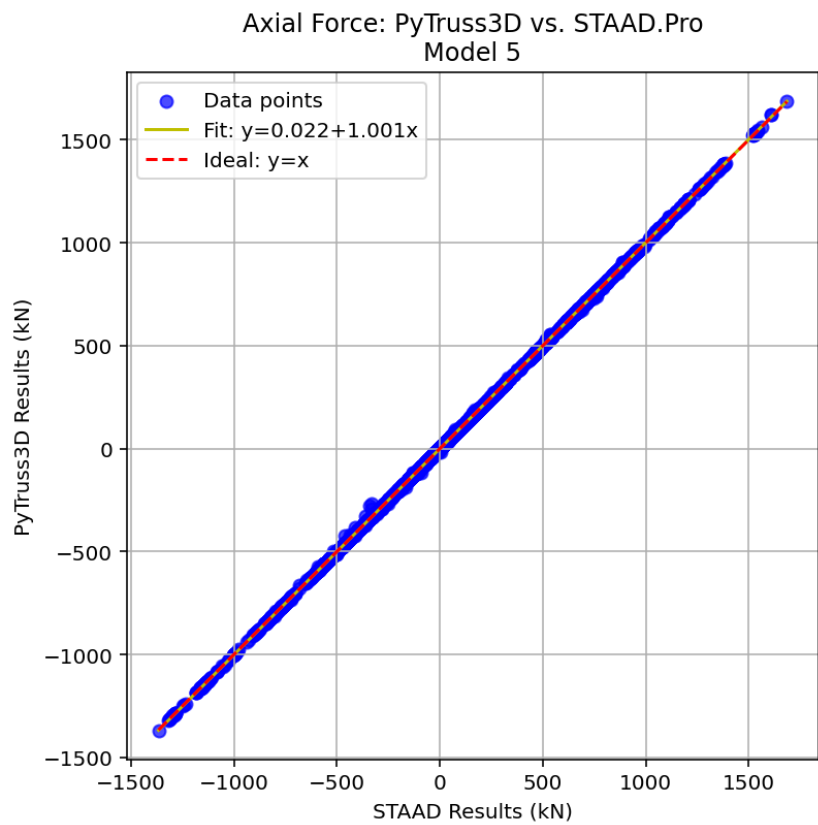


Figure 18. Model 5 fit of PyTruss3D vs. STAAD.Pro axial force results

In structural design, it is prevalent to identify the maximum response of the structure due to external loads [13]. Fig. 19, Fig. 20, and Fig. 21 illuminate the results of the maximum nodal displacements, maximum axial tensile forces, and maximum axial compressive forces, respectively. Each figure compares the response of the five models. These maximum values occur on the same nodes and members in both separate simulations. Fig. 19 depicts that there is a negligible difference in terms of displacement between the developed program and the benchmark software. Noticeably, a discrepancy of 0%, 1.64%, 2.47%, 2.33%, and 2.52% were observed for M1 to M5, respectively. Additionally, it is notable that the results in PyTruss3D overestimate those in STAAD.Pro results, suggesting more conservative values, giving a more robust

perception of the global behavior of the structure.

Primarily, the material sizing and strength design capacity of space truss members are related to axial forces based on demand loads [13]. Fig. 20 and Fig. 21 present the maximum values of tensile and compressive forces, respectively. In Fig. 20, the maximum axial tensile forces differed, varying from 0% on Model 1 to 0.38% on Model 4, and 0.22% on Model 5. It can be observed that there is a negligible difference in values between the results. Accounting for the minimal divergence in values, PyTruss3D never underestimates the results produced by STAAD.Pro, conserving the structural finite element analysis of space trusses, subsequently preserving the structural integrity.

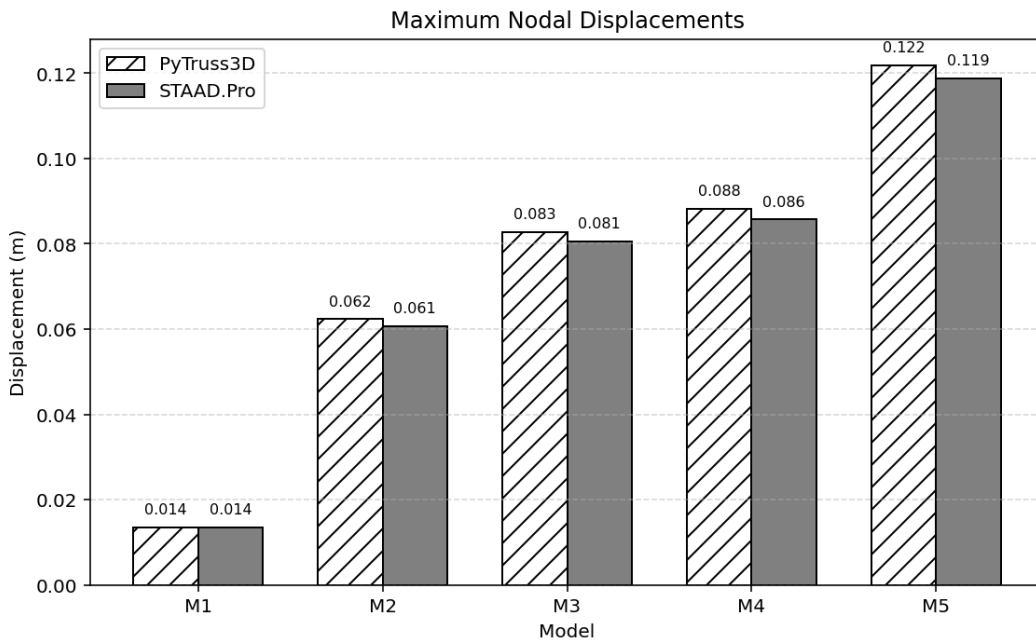


Figure 19. Maximum joint displacement of each trial model in meters

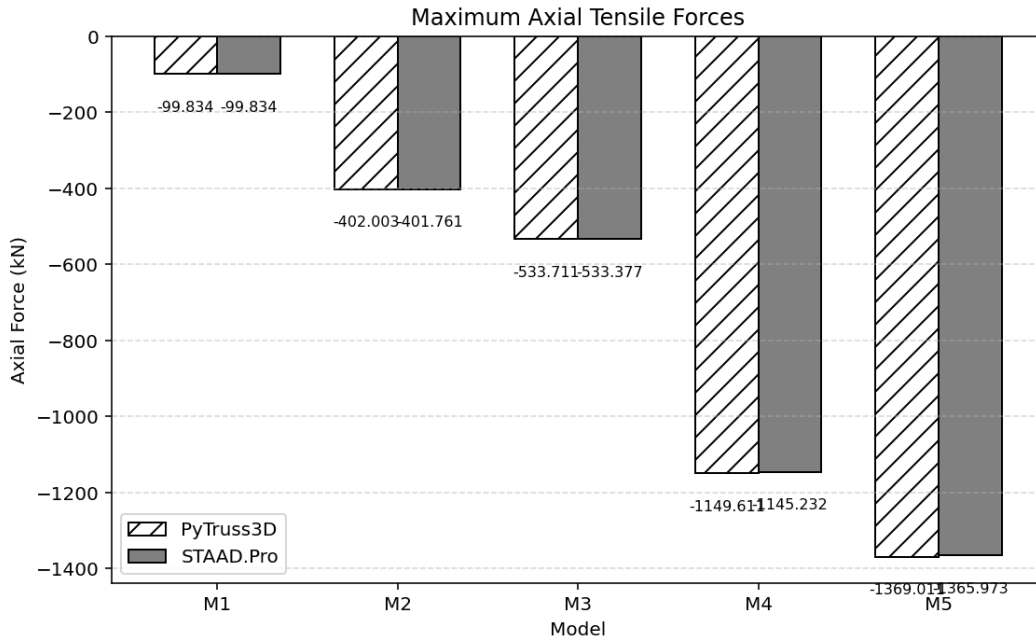


Figure 20. Maximum tensile force of each model in kilonewton

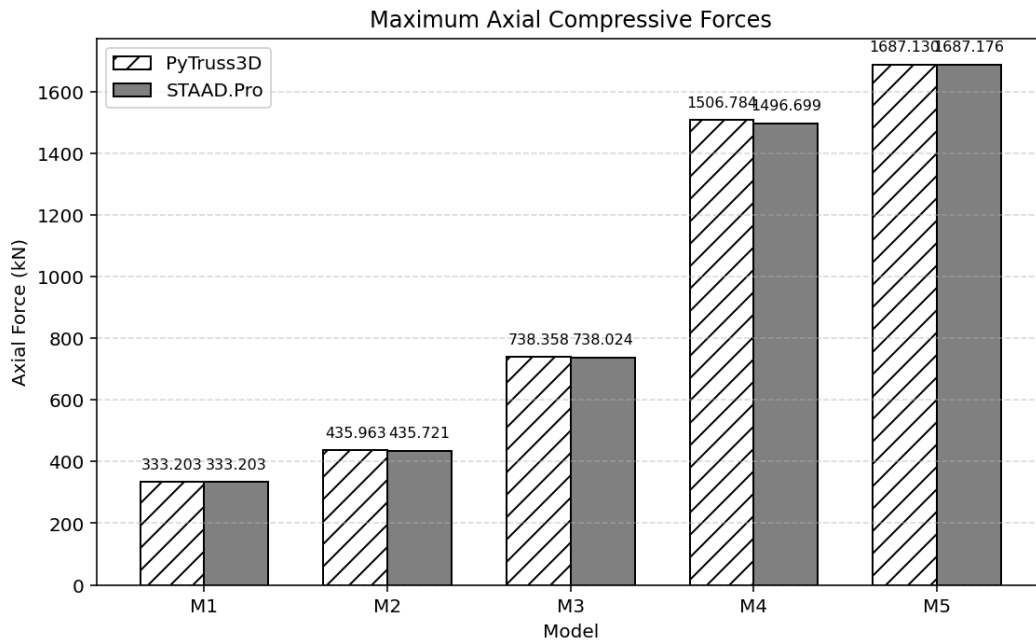


Figure 21. Maximum compressive force of each model in kilonewton

4.2. Visualization

Under the same applied loading conditions, each model exhibited displacement profiles (Figs. 22-26). These figures correspond to the displaced structures of Model 1 to Model 5, respectively. The left side of the figure represents the structure’s response modelled using PyTruss3D, while the right side figures illustrate the STAAD.Pro displaced structures. In all pairs of comparison, blue elements signify undeformed truss members, and red elements denote displaced members. It is notable that, based on axis

orientations, the STAAD.Pro illustrations are the mirror images of those of PyTruss3D, due to the interchange of the x- and z-axes. Considering all these conditions, the visualized displacement contours, generated by PyTruss3D and STAAD.Pro, exhibit the same directional behaviors and deformation shapes. Consistent with STAAD.Pro, PyTruss3D display identically for nodal movement. This is true for all models, regardless of the number of nodes, members, and restraints. Hence, findings imply that the stiffness matrix formulation and boundary conditions in PyTruss3D were implemented correctly.

Space Truss Structure with Displacement Visualization (Displacement scale: 50x) Model 1

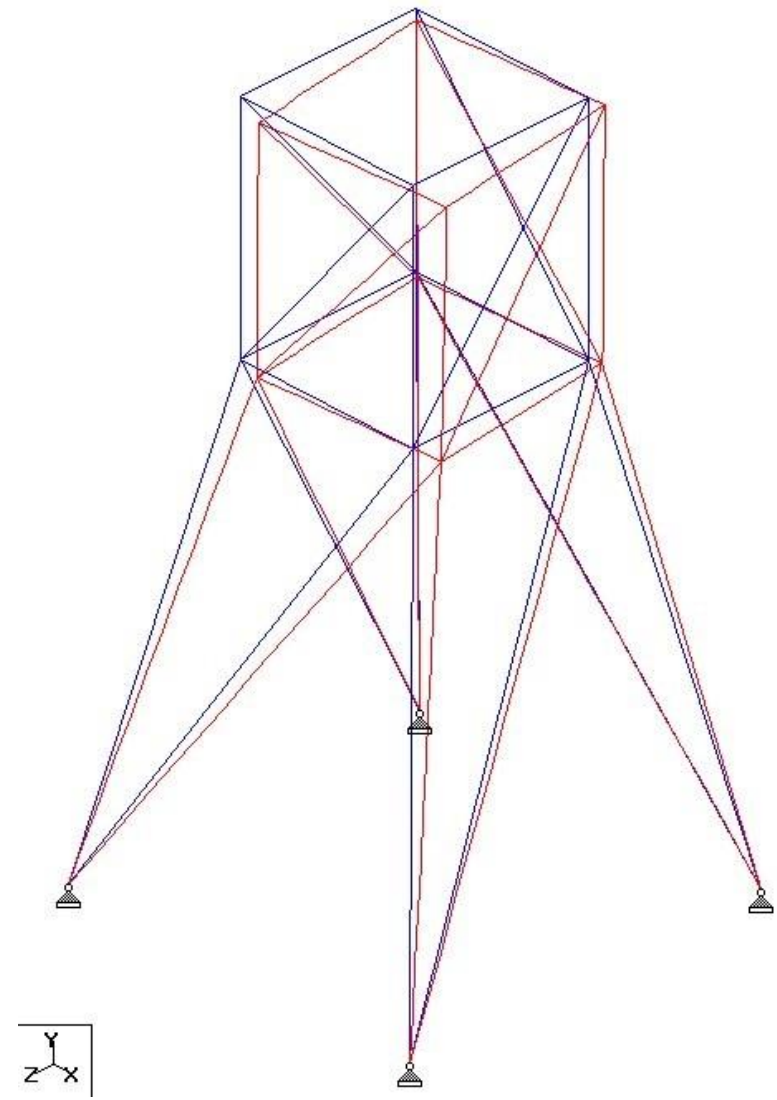
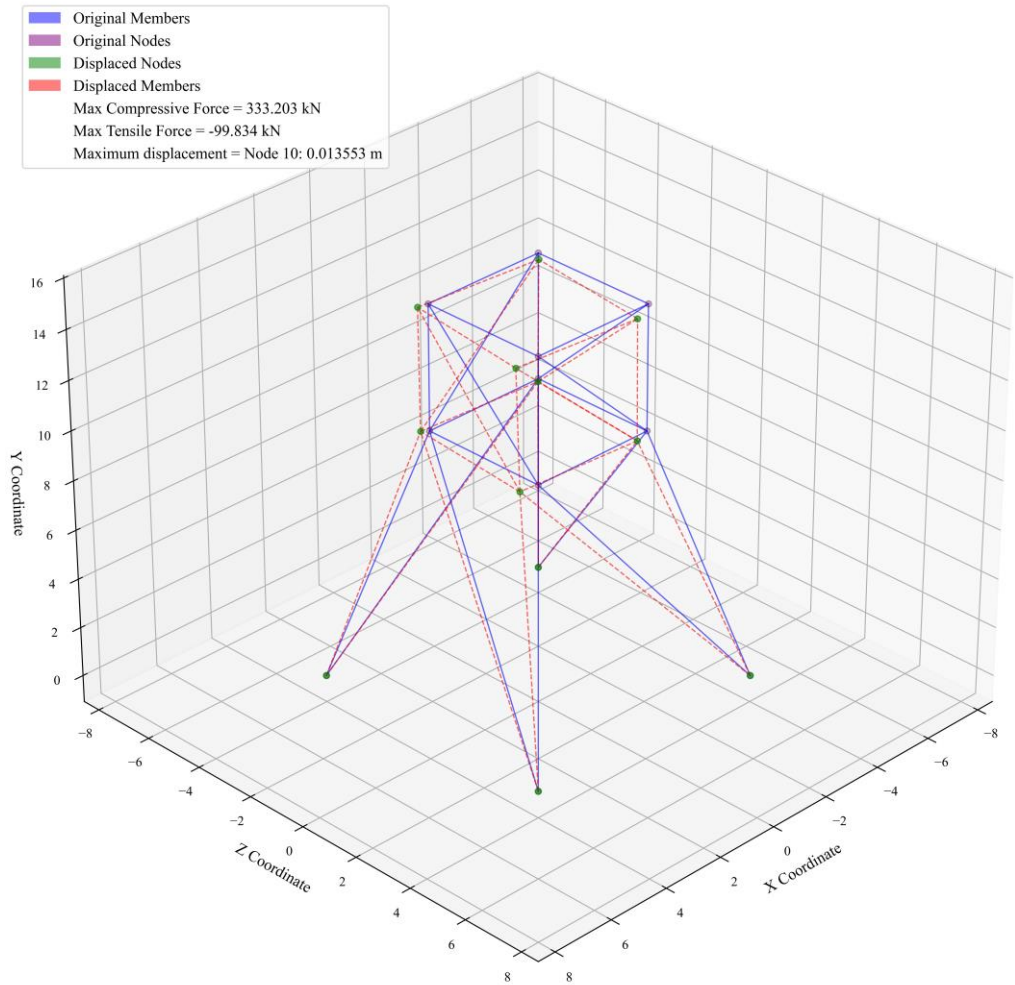


Figure 22. Displaced Model 1 structure relative to the original model: PyTruss3D (left) vs. STAAD.Pro (right)

Space Truss Structure with Displacement Visualization (Displacement scale: 50x) Model 2

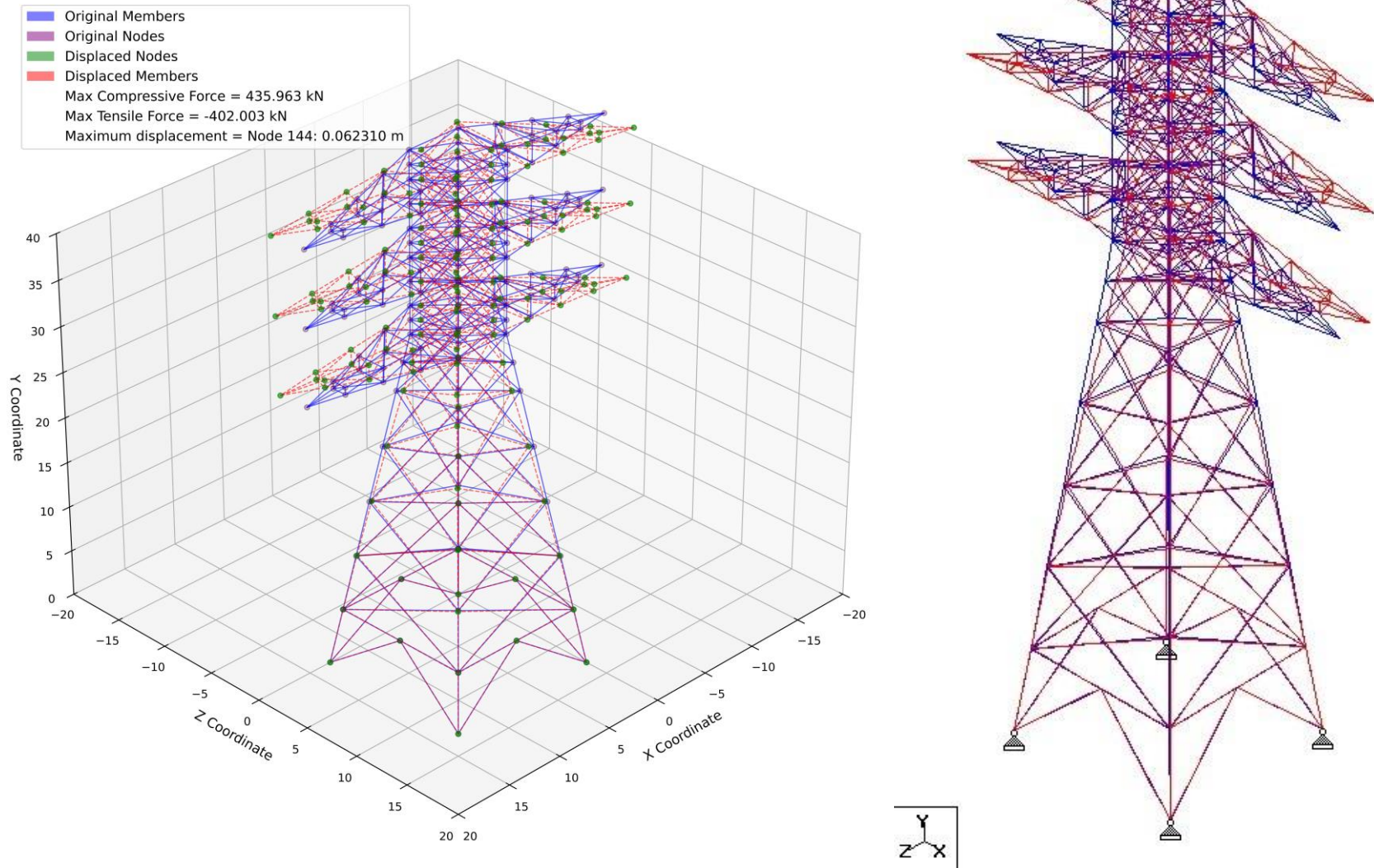


Figure 23. Displaced Model 2 structure relative to the original model: PyTruss3D (left) vs. STAAD.Pro (right)

Space Truss Structure with Displacement Visualization (Displacement scale: 50x) Model 3

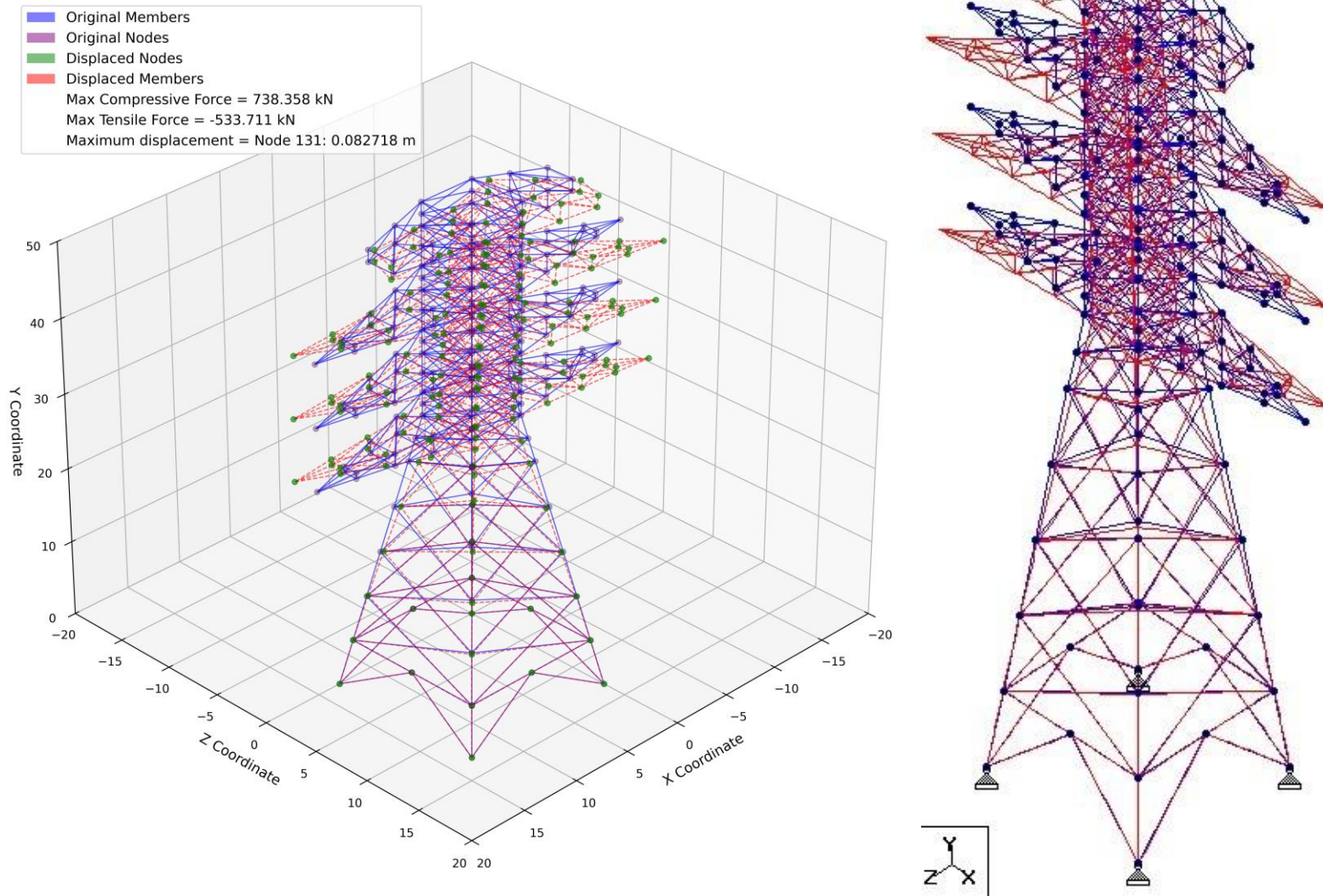


Figure 24. Displaced Model 3 structure relative to the original model: PyTruss3D (left) vs. STAAD.Pro (right)

Space Truss Structure with Displacement Visualization (Displacement scale: 50x) Model 4

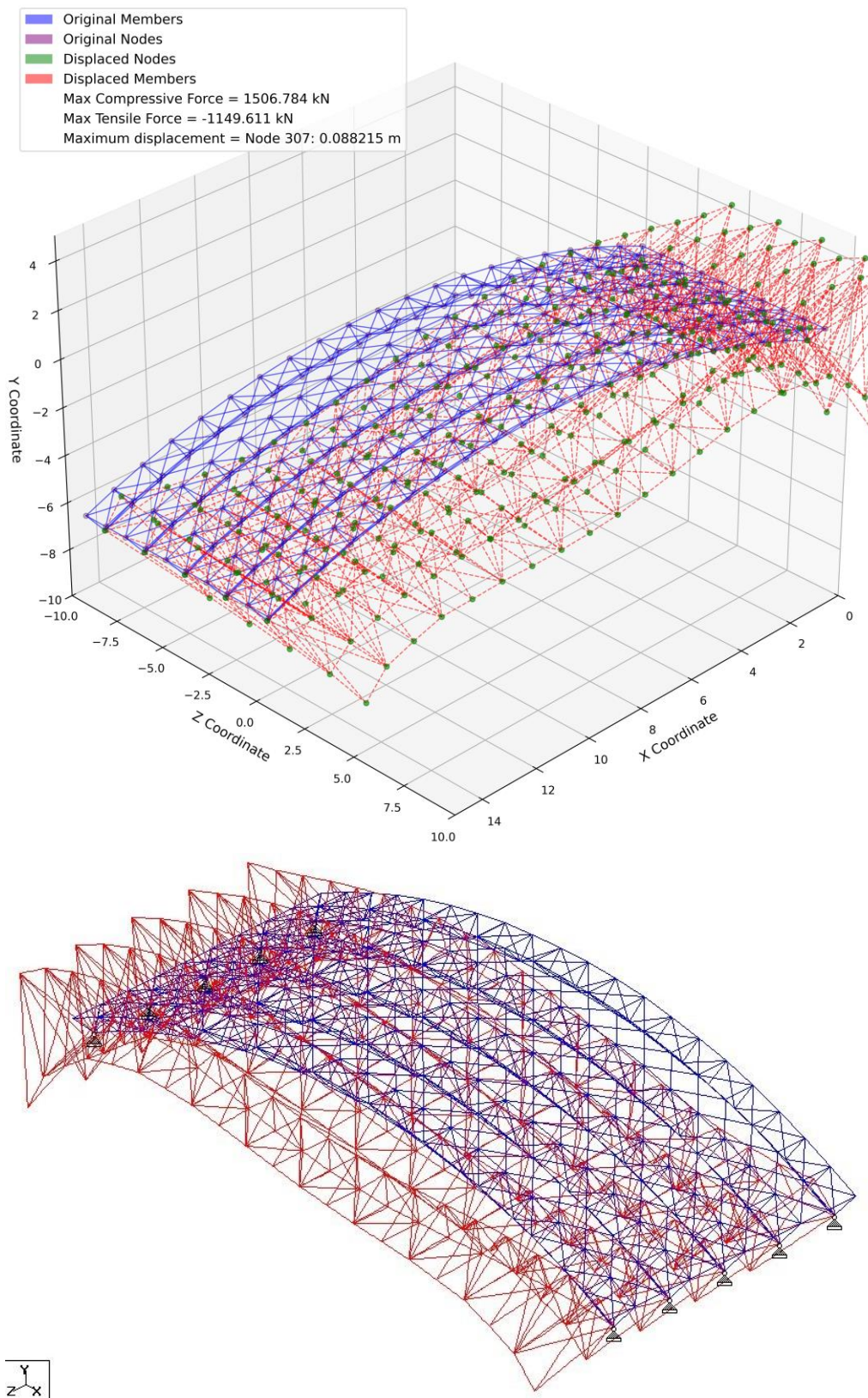


Figure 25. Displaced Model 4 structure relative to the original model: PyTruss3D (left) vs. STAAD.Pro (right)

Space Truss Structure with Displacement Visualization (Displacement scale: 50x) Model 5

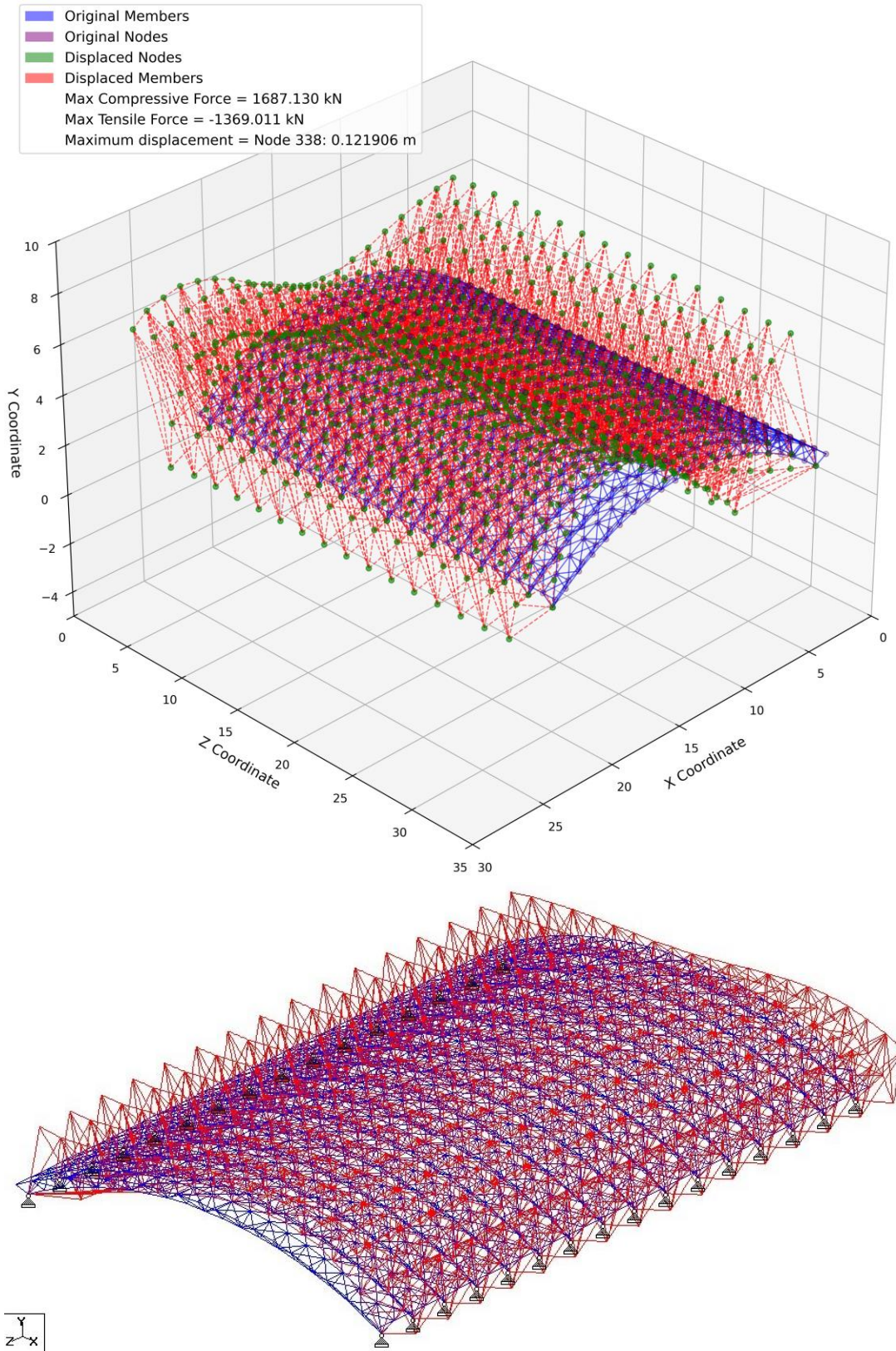


Figure 26. Displaced Model 5 structure relative to the original model: PyTruss3D (left) vs. STAAD.Pro (right)

The consistency between the displacement shapes reinforces the statistical results. These qualitative findings support the quantitative correlations, as evidenced by the excellent agreement between the RMSE, r , and R^2 . Furthermore, the results showed that PyTruss3D can accurately predict the magnitude and pattern of structural displacements.

5. Conclusions and Recommendations

Transforming the stiffness matrix from LCS to CGS was verified through PyTruss3D's ability to handle 3D space trusses from simple to complicated configurations (tested up to 1008 nodes, 96 restraints, and 3558 members). With the favorable results, PyTruss3D is expected to handle sophisticated 3D space truss structures with more than the tested configurations. The computed results from PyTruss3D are consistent with the benchmarked software, STAAD.Pro, confirming that the transformation matrices and matrix assembly were accurately implemented. The developed program's ability to export a matrix along with the computations is more advantageous for future integration with eigenvalue analysis, buckling factor identification, and modal shape visualization.

Attributed to the correct matrix stiffness assembly, PyTruss3D accurately determined the joint displacements and axial forces for the five models considered. Statistical analysis revealed that the developed program is in excellent agreement with the commercial software. Minimal numerical deviations were observed when comparing the RMSE and r -squared values for displacement, and these results remained strong for the axial forces. These findings implied that the response prediction of the structure using PyTruss3D is reliable.

Identical directional behavior and displacement contours were found in the visualized deformation shapes, noting minor differences in the axis orientation convention. In conclusion, under loading conditions, PyTruss3D accurately depicted the overall deformation behavior of the 3D space truss.

In summary, both numerical and visual comparative analyses demonstrated that the developed program yielded results nearly identical to those of STAAD.Pro with minor overestimation in axial forces and nodal displacement. Hence, slightly conservative values were generated by PyTruss3D, which are favorable for various engineering design applications. The robust and reliable results confirmed that PyTruss3D could effectively serve as a validation model for structural design (SDG 9 and 11), and an alternative open-source educational program (SDG 4) for analyzing 3D space trusses across complexities in structural geometry.

Further, based on the work carried out, the program is only limited to capturing the linear static behavior of the 3D trusses. A research on the nonlinear eigenvalue (buckling) analysis on various complex truss geometry should be explored.

Nomenclature of Symbols and Abbreviations

Symbols:

$[K]$ = stiffness matrix

$\{u\}$ = displacement vector

$\{F\}$ = force vector

E = modulus of elasticity

A = cross-sectional area

Q_i = local forces

F_i = global forces

u_i = local displacement

v_i = global displacement

$\cos \theta_{axis}$ = direction cosine on the specified axis

T = transformation matrix

S = pertinent matrix

Abbreviations:

FEA – Finite Element Analysis

STAAD.Pro – Structural Analysis and Design Program

DSM – Direct Stiffness Method

FEM – Finite Element Method

LCS – Local Coordinate System

GCS – Global Coordinate System

DOF – Degrees of Freedom

Acknowledgements

The authors express their utmost gratitude to the invaluable suggestions of the anonymous arbitrators.

REFERENCES

- [1] V. Plevris and A. Ahmad, "Deriving analytical solutions using symbolic matrix structural analysis: part 2 – Plane trusses," *Heliyon*, vol. 11, no. 4, p. e42372, Feb. 2025. doi: 10.1016/j.heliyon.2025.e42372.
- [2] A. Kassimali, *Matrix Analysis of Structures*, 1st ed. Cengage Learning, Southern Illinois University, 2012.
- [3] Bentley Systems, Incorporated, STAAD.Pro Documentation, 2025. <https://www.bentley.com/> (accessed Nov. 2, 2025).
- [4] B. Etaati, M. Neshat, A. Dehkordi, N. Pargoo, M. El-Abd, A. Sadollah, and A. Gandomi, "Shape and sizing optimization of space truss structures using a new cooperative coevolutionary-based algorithm," *Results in Engineering*, vol. 21, p. 101859, 2024. doi: 10.1016/j.rineng.2024.101859.
- [5] R. Zhang, L. Wang, J. Liu, and J. Zhu, "Lightweight design of space trusses considering joint parameterization," *Acta Aeronautica et Astronautica Sinica*, vol. 45, no. 5, p. 529715, 2024. doi: 10.7527/S1000-6893.2023.29715.
- [6] M. Papadrakakis and E. Sapountzakis, "Chapter One – Introduction to Matrix Methods of Structural Analysis," in

- Matrix Methods for Advanced Structural Analysis*, pp. 1–15, 2018. doi: 10.1016/B978-0-12-811708-8.00001-5.
- [7] A. Haider, “Enhancing transparency and reproducibility in finite element analysis through comprehensive reporting parameters: A review,” *El-Cezeri Journal of Science and Engineering*, vol. 11, no. 3, pp. 212–222, 2024. doi: 10.31202/ecjse.1436203.
- [8] J. Feng, C. Li, Y. Xu, Q. Zhang, F. Wang, and J. Cai, “Analysis of key elements of truss structures based on the tangent stiffness method,” *Symmetry*, vol. 12, no. 6, p. 1008, 2020. doi: 10.3390/sym12061008.
- [9] O. Eugenio, *Structural Analysis with Finite Element Method, Vol. 1: Basis and Solids*. Springer Dordrecht. doi: 10.1007/978-1-4020-8733-2.
- [10] N. Saeed, A. Manguri, M. Szczepanski, and R. Jankowski, “Non-linear analysis of structures utilizing load-discretization of stiffness matrix method with coordinate update,” *Applied Sciences*, vol. 12, p. 2394, 2022. doi: 10.3390/app12052394.
- [11] X. Wang, Y. Wang, J. Bai, Z. Zhao, and C. Luo, “Finite element analysis for linear viscoelastic materials considering time-dependent Poisson’s ratio: variable stiffness method,” *Applied Sciences*, vol. 14, p. 3189, 2024. doi: 10.3390/app14083189.
- [12] R. Ledesma, D. Silva, C. J. Marcos, and K. L. de Jesus, “Structural member strength prediction using backpropagation neural network: a tool for retrofitting intervention integration non-linear static analysis,” in *Proceedings of the 4th International Civil Engineering and Architecture Conference*, M. Casini, Ed., Lecture Notes in Civil Engineering, vol. 534. Springer Nature Singapore Pte Ltd., 2025. doi: 10.1007/978-981-97-5477-9_6.
- [13] W. Segui, *Steel Design*, 6th ed. Cengage Learning, 2018.