

A Comparative Study of Python Tool with Software for Beams Under Various Loading Conditions

1st Raubin Kumar Ravi

Department of Civil and Environmental
Engineering
University of Alberta, Edmonton, AB,
T6G 1H9, Canada
raubinkr.ravi@gmail.com

2nd Sakshi Suman

Department of Computer Science &
Engineering
Gaya College of Engineering
Gaya, India
sakshisuman2015@gmail.com

3rd Raushan Kumar Ravi

Department of Civil Engineering
Department of Science and Technology
and Technical Education
Bihar, India
raushan23.dstte@bihar.gov.in

Abstract - Structural analysis is an essential component of civil engineering, guaranteeing that structures resist external stresses while preserving safety and functionality. This work investigates the creation of a generalized Python-based methodology for structural analysis, providing a more efficient and automated solution. This study aims to compare the analysis results of beams with different loading conditions from Python and finite element software. The Python libraries such as Matplotlib, NumPy, and plainsections are used for structural analysis. The program is designed to generate shear forces and bending moment diagrams for each beam type with different loading conditions. The results obtained from the Python code are compared and validated against the results from SAP2000 and STAAD Pro software. The findings indicate a high degree of correlation between Python-based computations and software-based solutions, demonstrating the potential of Python in structural engineering. This study highlights the potential of Python as a powerful tool for structural analysis, particularly in automating repetitive tasks and reducing the time required for preprocessing and analysis. The Python tool is capable of handling different load types and can be easily extended to analyze more complex structural elements.

Keywords—*Structural Analysis, Python, Shear Force, Bending Moment, Automation.*

I. INTRODUCTION

Structural analysis forms the fundamental basis of civil and mechanical engineering design, enabling engineers to predict structural behavior under various loading conditions and ensure safety and serviceability requirements are met [1,2]. Traditional structural analysis methods rely heavily on manual calculations or commercial software packages, which can be time-consuming, expensive, and limit customization possibilities [3,4]. The emergence of programming languages such as Python has opened new avenues for developing customized, automated, and cost-effective structural analysis tools [5,6].

Python, with its extensive scientific computing libraries and intuitive syntax, has gained significant traction in the engineering community [7,8]. The language offers powerful libraries such as NumPy for numerical computations, SciPy for scientific computing, and Matplotlib for visualization, making it an ideal platform for developing engineering applications [9,10]. Recent developments in finite element analysis libraries, particularly OpenSeesPy and supporting packages like PlaneSections, have made sophisticated structural analysis accessible through Python programming [11,12].

This study details the development of a novel Python-based tool designed for the comprehensive structural analysis of beams subjected to diverse loading conditions. The increasing demand for efficient and versatile tools in structural engineering necessitates the exploration of advanced computational methods and programming languages. Python, with its extensive libraries for numerical computation and data visualization, presents a compelling choice for developing such a tool. Existing software packages often lack the flexibility and customization options required for specific research or engineering projects [13,14].

Structural analysis of beams is fundamental in civil engineering, involving the determination of internal actions (shear forces, bending moments) under applied loads. Traditionally, design checks have used closed-form formulas and commercial tools like SAP2000 or STAAD Pro. However, advances in computational methods and open-source tools have enabled custom analysis using scripting languages. Python, in particular, has gained popularity for engineering workflows due to its readability and rich ecosystem of libraries (NumPy, SciPy, etc.). Recent works demonstrate Python's use in structural modeling (e.g., the Anastruct module for 2D frames) and finite element analysis (e.g., scikit-fem). Open-source FEM libraries such as OpenSeesPy (for advanced nonlinear and seismic analysis) and PyNite (for 2D/3D frame analysis) are now mature. This paper builds on that trend by developing a generalized Python code to automate beam analysis for a variety of cases, and validating it against industry software.

II. LITERATURE REVIEW

The use of Python in structural analysis has grown significantly in recent years. Several existing tools and methodologies provide a foundation for this project. The PANDORA framework [13], a Python-based tool for aircraft structural sizing, showcases the potential of Python in structural analysis. However, PANDORA's focus on aircraft structures differs from the broader scope of beam analysis considered here. Object-oriented programming (OOP) in Python has been demonstrated as effective for managing complex data structures [15].

These tools allow us to define geometry, materials, and loads directly in code and obtain internal force diagrams. For example, PlaneSections is explicitly designed to “quickly run beam analyses” and produce force diagrams to document structural calculations. Other Python packages, such as scikit-fem, provide general finite-element assembly capabilities, but often focus on academic research. Recent tutorials and reviews highlight these resources: Maldonado et al. (2024) survey five Python libraries for structural

engineers (OpenSeesPy, Anastruct, PythTB) and emphasize their advantages in ease of use and custom automation. Several researchers have explored the application of Python in structural engineering applications [16]. Guran et al. [17] developed a Python-based platform for automated seismic design of steel moment-resisting frames, demonstrating the language's capability in complex structural applications. Similarly, Jarasjarungkiat [18] presented a structural influence line analysis program based on Python, highlighting the potential for developing specialized analysis tools. The development of OpenSeesPy has significantly enhanced Python's capabilities in nonlinear structural analysis [19]. Moore et al. [20] presented a nonlocal results processor for finite element analysis, while Slotboom [21] developed the planesections library specifically for beam analysis applications. These developments have established a robust foundation for Python-based structural analysis tools.

Recent studies have demonstrated successful implementations of Python in various structural engineering domains. Kothari et al. [22] explored the integration of Python in civil engineering for design automation and analysis enhancements, while several researchers have focused on specific applications such as seismic analysis and optimization [23,24].

In parallel, computational structural analysis research has explored automated model generation and cross-platform integration. Studies have shown how Python can script commercial tools, as well as implement classical stiffness-matrix solutions [5]. Many of these studies have shown that results from Python FEM code match closely with established software when properly validated [8]. This work is carried out by implementing the Python libraries for analysis algorithmically and verifying against both software-based SAP2000 and STAAD results. Additionally, as highlighted by Slotboom, some Python libraries are still evolving, so care must be taken regarding interpolation schemes and units[1,21]. This study aims to address this gap and verify both Python-based and software-based results by analysing beams under various loading scenarios.

III. OBJECTIVE

- Perform shear force and bending moment analysis of structural members using commercial software SAP2000 and STAAD Pro.
- Perform shear force and bending moment analysis Using a Python-based computational tool to automate the structural analysis of beams under various loading conditions.
- Validate the accuracy of the Python-generated results by comparing them with those obtained from SAP2000 and STAAD Pro.

IV. METHODOLOGY

The methodology for this study involves a systematic approach to structural analysis using both conventional software and Python programming. Initially, relevant data such as material properties, geometric details, loading conditions, and boundary constraints are used. To establish a benchmark, structural analysis is first performed using industry-standard software SAP2000 [25] and STAAD Pro [26], where beams are modelled with various loading conditions and analyzed to obtain shear force and bending

moment diagrams. Following this, a custom Python program is written with libraries to automate the structural analysis using libraries such as planesections, NumPy. A general Python code was written to analyze 2D beam elements using the planesections library. This library encapsulates a builder-module API for beams and loads, an analysis module (using OpenSees or PyNite as solvers), and a plotting module. The program defines beam elements, applies various load conditions, and computes shear force and bending moments through numerical methods. The results obtained from the Python model are then validated by comparing them with the outputs from SAP2000 and STAAD Pro.

A. Framework Development

The Python framework was developed using the planesections library, which provides a lightweight finite element beam analysis. The framework incorporates several key components and modules:

1) Core Libraries

- planesections: Primary library for beam analysis functionality
- NumPy: Numerical computations and array operations

2) Framework Architecture

- Beam Definition Module: Creates beam objects with specified length and properties
- Support Condition Module: Defines various support conditions, including pinned, roller, fixed, and free supports
- Loading Module: Handles different types of loads, including point loads, distributed loads, and moment loads
- Analysis Module: Performs finite element analysis using OpenSeesPy backend
- Visualization Module: Generates beam diagrams, shear force diagrams, and bending moment diagrams

3) Support Conditions Implementation

The framework implements support conditions using a three-component fixity vector $[x, y, \theta]$ where:

- x: Horizontal restraint (0 = free, 1 = restrained)
- y: Vertical restraint (0 = free, 1 = restrained)
- θ : Rotational restraint (0 = free, 1 = restrained)

Common support types are defined as:

- Fixed support: [1, 1, 1]
- Pinned support: [1, 1, 0]
- Roller support: [0, 1, 0]
- Free end support: [0, 0, 0]

The steps for each case, which are used in this study with software and Python, are given below:

- (1) Instantiate a beam of given length.
- (2) Add support conditions (free/roller/pinned/fixed).
- (3) Add loads (point loads or distributed loads).
- (4) Run the analysis.

- (5) Extract shear force and bending moment data.
- (6) Comparison of results.

V. FRAMEWORK ARCHITECTURE

The Python framework is built upon several key components. A PlaneSections is a specialized Python library for beam analysis built on OpenSeesPy, providing simplified interfaces for beam modeling, loading application, and results visualization developed by Christian Slotboom [21]. Below is the given generalized Python code to create the beam geometry diagrams, shear force diagrams, and bending moment diagrams.

A. Installation and Import of Python library

```
pip install planesections
import planesections as ps
```

B. Beam Object Creation:

```
L = (put the length) # beam length in meters
beam = ps.newEulerBeam2D(L)
```

C. Support Condition Definition

```
pinned = [1,1,0]
roller = [0,1,0]
fixed = [1,1,1]
free = [0,0,0]
beam.setFixity(position, condition_type)
```

D. Load Application

```
# Point load
beam.addVerticalLoad(position, magnitude)
# Distributed load
beam.addDistLoadVertical(start_pos, end_pos,
magnitude)
```

E. Analysis Execution

```
analysis = ps.PyNiteAnalyzer2D(beam)
analysis.runAnalysis()
```

F. Results Visualization

```
ps.plotBeamDiagram(beam)
ps.plotShear2D(beam, scale=1, yunit='N')
ps.plotMoment2D(beam, scale=1, yunit='Nm')
```

VI. DEVELOPMENT OF TEST EXAMPLE FOR VALIDATION

For checking and validating coverage with the help of the Python program, SAP2000, and STAAD Pro. A simple test example was taken. The subsequent section describes the test example code and the generated beam geometry, Shear force, and bending moment diagram.

A. Simply Supported Beam with point Load at mid-point

```
pip install planesections
import planesections as ps
# Define node locations, and support conditions
L = 20
beam = ps.newEulerBeam(L)
# Define beam and fixities
pinned = [1,1,0]
```

```
roller = [0,1,0]
beam.setFixity(0, pinned, label = 'A')
beam.setFixity(20, roller, label = 'B')
# Define loads
Pz = -800
beam.addVerticalLoad(10, Pz)
# Run the analysis
analysis = ps.PyNiteAnalyzer2D(beam)
analysis.runAnalysis()
# Plot the beam diagram
ps.plotBeamDiagram(beam)
# Plot the SFD and BMD
ps.plotShear(beam)
ps.plotMoment(beam)
```

The below is default label, unit, and size are shown after the completion of the analysis. The last accessed of the plansections library on August 25, 2025.

```
(<Figure size 1920x1440 with 1 Axes>,
<Axes:      xlabel='Distance      (m) ',
ylabel='Internal      Force      (Nm) '>,
[<matplotlib.lines.Line2D
0x7b1d53b80aa0>])
```

After running the above program (section VI(A)), Fig. 1, Fig. 2 Fig. 3 show the output. Further, the generated shear force and bending diagram is compared with the generated diagram with the SAP2000 and STAAD Pro results.



Fig. 1. Generated simply supported beam geometry

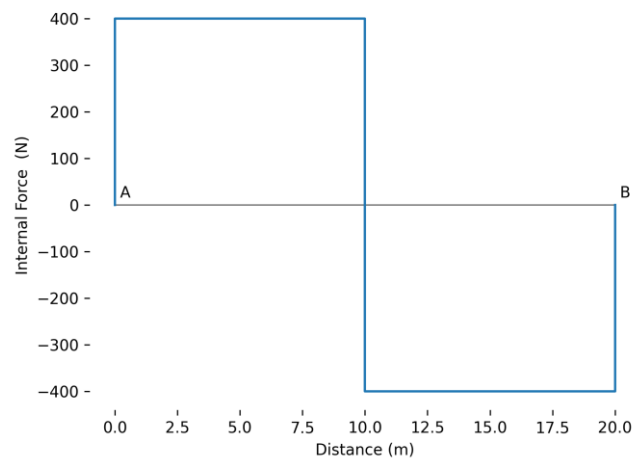


Fig. 2. Generated Shear force diagram

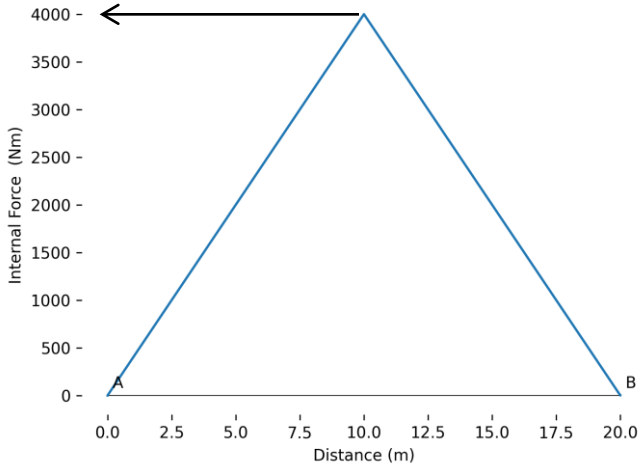


Fig. 3. Generated bending moment diagram

The same test example is modeled in the SAP2000 and STAAD Pro software for validation purposes. Fig. 4 shows the software-based shear force and bending moment diagram of the Simply Supported beam with a point load in the middle test example. The corresponding values are shown at a distance of 10 m from the end, which is the middle of the beam. In the shear force diagram, only the sign convention is different, but it is up to the users. The result will not be affected by the sign convention. It purely depends on the user.

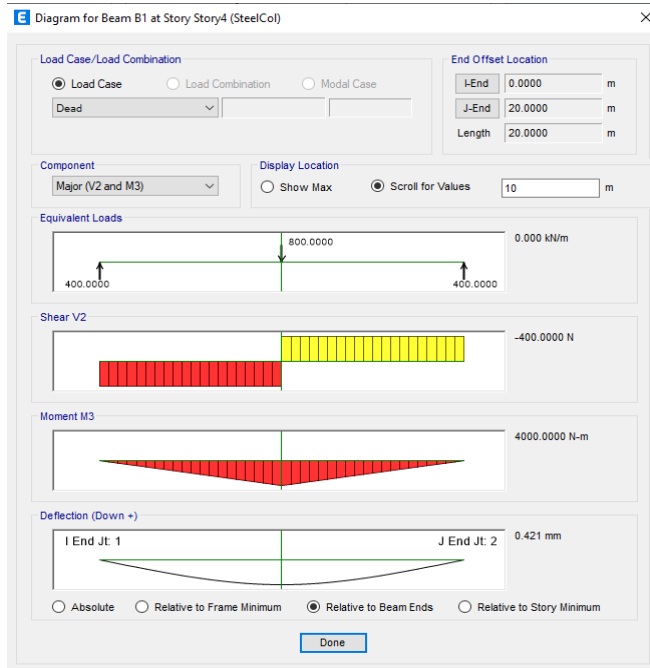


Fig. 4. Similar Output generated from both software

VII. RESULT AND DISCUSSION

The structural analysis conducted using the developed Python code was validated against SAP2000 and STAAD Pro, demonstrating high accuracy across various beam configurations and loading conditions. The results for shear force and bending moment closely matched with software result and the developed Python tool result. The Python

program efficiently automated the analysis, significantly reducing computation time compared to manual calculations and traditional software. Similarly, several test samples with different load and support combinations were developed for verification. All the test examples have been verified with SAP2000 and STAAD Pro software.

VIII. CONCLUSION

Based on the findings of this study, the ‘plainsections’ Python library is verified with the software-generated results. This study demonstrates that the Python-based tool provides results for beam analysis under various loading conditions that are consistent with those obtained from established structural analysis software. While commercial software such as SAP2000 and STAAD Pro offer advanced features, graphical interfaces, and comprehensive design capabilities, the Python tool proves to be a reliable and flexible alternative for fundamental analysis. Overall, the study highlights that Python can serve as a powerful and efficient tool for structural analysis when tailored to specific requirements.

IX. RECOMMENDATION

As this study is a preliminary comparative study of Python tools, this study can be further improved and made automated after developing the advanced Python libraries with the implementation of Artificial intelligence and modern machine learning tools.

ACKNOWLEDGMENT

The authors acknowledge the valuable contributions of the open-source community, particularly the developers of the plainsections libraries.

REFERENCES

- [1] A. Kassimali, Structural Analysis, 6th ed. Boston, MA, USA: Cengage Learning, 2020.
- [2] R. C. Hibbeler, Structural Analysis, 10th ed. Boston, MA, USA: Pearson, 2021.
- [3] D. L. Logan, A First Course in the Finite Element Method, 6th ed. Boston, MA, USA: Cengage Learning, 2016.
- [4] K. J. Bathe, Finite Element Procedures, 2nd ed. Watertown, MA, USA: Klaus-Jürgen Bathe, 2014.
- [5] W. McKinney, Python for Data Analysis, 2nd ed. Sebastopol, CA, USA: O’Reilly Media, 2017.
- [6] H. P. Langtangen and S. Linge, Finite Difference Computing with PDEs. Cham, Switzerland: Springer, 2017.
- [7] T. E. Oliphant, A Guide to NumPy. USA: Trelgol Publishing, 2006.
- [8] E. Jones, T. Oliphant, and P. Peterson, “SciPy: Open source scientific tools for Python,” 2001. [Online]. Available: <https://www.scipy.org/>
- [9] J. D. Hunter, “Matplotlib: A 2D graphics environment,” Comput. Sci. Eng., vol. 9, no. 3, pp. 90–95, 2007.
- [10] G. Van Rossum and F. L. Drake, Python 3 Reference Manual. Scotts Valley, CA, USA: CreateSpace, 2009.
- [11] F. McKenna, M. H. Scott, and G. L. Fenves, “Nonlinear finite-element analysis software

- architecture using object composition,” *J. Comput. Civ. Eng.*, vol. 24, no. 1, pp. 95–107, 2010.
- [12] M. Zhu, F. McKenna, and M. H. Scott, “OpenSeesPy: Python library for the OpenSees finite element framework,” *SoftwareX*, vol. 7, pp. 6–11, 2018.
- [13] M. K. Petsch, D. M., and J. H., “PANDORA—A Python-based framework for modelling and structural sizing of transport aircraft,” 2018.
- [14] R. Rangel and L. F. Martha, “LESMAAn object-oriented MATLAB program for structural analysis of linear element models,” 2019.
- [15] D. P. N., Ferreira, and Proena, “An object-oriented class design for the generalized finite element method programming,” 2013.
- [16] “scikit-fem: A Python package for finite element assembly,” *J. Open Source Softw.* Available: <https://joss.theoj.org/papers/10.21105/joss.02369>
- [17] X. Guan, H. Burton, and T. Sabol, “Python-based computational platform to automate seismic design, nonlinear structural model construction and analysis of steel moment resisting frames,” *Eng. Struct.*, vol. 224, p. 111199, 2020.
- [18] A. Jarasjarungkiat, “Structural influence line analysis program based on Python,” *AIP Conf. Proc.*, vol. 2397, no. 1, p. 030004, 2021.
- [19] S. Mazzoni, F. McKenna, M. H. Scott, and G. L. Fenves, *OpenSees Command Language Manual*. Berkeley, CA, USA: Pacific Earthquake Engineering Research (PEER) Center, 2006.
- [20] J. A. Moore, C. Martinez, and A. Chandel, “NOCAL-FEA: A NonLOCAL results processor for Finite Element Analysis,” *SoftwareX*, vol. 24, p. 101532, 2023.
- [21] C. Slotboom, “PlaneSections”, 2025. [Online]. Available: <https://planesections.readthedocs.io> and <https://pypi.org/project/planesections/>. Last accessed: August 25, 2025.
- [22] M. Kothari, M. Kothari, and S. G. Ban, “Integration of Python in civil engineering for enhancements in design, analysis, and management,” *IOSR J. Mech. Civ. Eng.*, vol. 21, no. 4, pp. 24–27, 2024.
- [23] S. Das and N. M. A. Krishnan, “Integrating experiments, finite element analysis, and interpretable machine learning to evaluate the auxetic response of 3D printed re-entrant metamaterials,” *J. Mater. Res. Technol.*, vol. 25, pp. 1606–1621, 2023.
- [24] O. Bolaji, M. Helio, K. N. M. Anoop, and D. Sumanta, “Machine learning for structural engineering: A state-of-the-art review,” *Structures*, vol. 38, pp. 448–491, 2023.
- [25] SAP2000. Computers and Structures Inc., Berkeley, CA, USA.
- [26] STAAD Pro. Bentley Systems Inc., Exton, PA, USA.