Geometrically nonlinear truss analysis using Python's optimization libraries

Isabela Sena de Oliveira Cabral[1], Eduardo Magalhaes Vieira[1], Jorge Carvalho Costa[1]

[1]*Dept. of Civil Engineering, Universidade Federal de Sergipe*
Av. Gov. Marcelo Deda Chagas, S/N, Bairro Rosa Elze 49107-230, Sergipe, Brazil
isabelacbrl@hotmail.com, eduardomvieira2347@gmail.com, jorgecostase@gmail.com

Abstract. This paper presents an efficient computational framework for analyzing trusses subject to geometric nonlinearities using common Python libraries for numerical optimization, scipy, and graphical analysis, matplotlib. We will describe the deformed configuration of the lattice and its internal energy based on the nodal displacements. For a better result, we then evaluated some available numerical optimization libraries to select the most appropriate method for our problem, thus avoiding the explicit derivation of equilibrium equations and tangent matrices. The network equilibrium is defined as the configuration with minimum potential energy and this is taken as the objective function of the optimization algorithm. Finally, we conclude with numerical examples comparing our approach with classical solutions based on a system of nonlinear equations. The proposed framework offers an interesting alternative for solving the nonlinear equilibrium problem, with particular potential for educational purposes and code prototyping. Although our approach may not optimize computational runtime, it significantly simplifies coding time, thereby improving accessibility and usability in engineering applications.

Keywords: python programming language; scipy; geometrically nonlinear structural analysis; energy methods; numerical optimization.

1 Introduction

Over the years, computational methods have emerged to solve long-standing engineering problems. One of these challenges is the analysis of structures with large strains and displacements. When structures are analyzed linearly, the model may not accurately represent reality. Therefore, nonlinear computational mechanics is essential to obtain accurate results. In this article, the analysis of trusses subject to geometrical nonlinearity was explored using Python, stating the minimal energy principle as an optimization problem.

Most methods for geometrically nonlinear analysis rely on energy methods such as the Principle of Virtual Works or The Principle of Least Work to algebraically obtain the equilibrium equations and some sort of tangent matrix. The objective of the article is to avoid explicit derivatives and matrices and at the same time obtain results close to reality. For this, Python optimization libraries were used. Specifically, SciPy with the Nelder-Mead and BFGS methods, which iteratively finds the minimum values of a given numerical function.

The approach involves constructing a truss that initially receives assumed displacements. Through iterations, it refines these displacements based on the influence of applied loads. This analysis aligns with the concept of minimizing potential energy to find equilibrium.

2 Literature review

2.1 External work and strain energy due to normal stress

Before discussing strain energy, it is essential to define the concept of work. According to Hibbeler [1], considering an axial force gradually applied to the end of a bar and assuming the material behaves in a linear and elastic manner, the area under the force-displacement graph represents the work done by this force. Therefore,

$$U = \frac{1}{2} P\Delta$$

Thus, work is stored in the bar in the form of strain energy. Considering initially an infinitesimal element subjected to an axial load of $dF_z = \sigma_z d_x d_y$, we have the following expression:

$$dU_{int} = \frac{1}{2}\, dF_z\, d\Delta_z = \frac{1}{2}\, \sigma_z\, dx\, dy\, d\Delta_z\, \varepsilon_z\, dz = \frac{1}{2} \sigma_z\, \varepsilon_z\, dV$$

Applying Hooke's law, given that the analysis is geometrically nonlinear

$$U_{int} = \int_V \frac{\sigma_z^2}{2E}\, dV = \int_V \frac{N^2}{2EA^2}\, dV$$

For a prismatic bar of constant cross-sectional area A and length L,

$$U_{int} = \int_0^L \frac{N^2}{2EA^2} A\, dz = \frac{N^2 L}{2EA}$$

Thus, the sum of the internal energy of each bar, together with the energy due to the work done by external loads, represented by $U_{ext,j} = F_j \cdot u_j$, gives us the total energy of the system.

$$U_T = \sum_{i=1}^{n} U_{int,i} - \sum_{j=1}^{m} U_{ext,j} = \sum_{i=1}^{n} \frac{N_i^2 L_i}{2E_i A_i} - \sum_{j=1}^{m} \frac{P_j \Delta_j}{2}$$

In this case, the total energy will be our objective function to be optimized (minimized).

2.2 Principle of Least Work

As described in the previous section, the total potential energy in a system is composed by the internal (strain) energy reduced by the external energy spent by the loads applied to the structure. The Principle of Least Work or Principle of Minimum Potential Energy states that a structure is at a static equilibrium state when the total potential energy is at its minimum [2].

2.3 Optimization problem

Using the concept of minimum total energy of a truss, we can define the proposed question as an optimization problem.

The total energy of a truss depends on the axial strain energy and the energy due to external loads applied at the nodes. A system is in equilibrium when its total energy is minimized. Therefore, by minimizing the total energy, we obtain an equilibrated structure. From the displacements, we can determine the internal forces of each bar and, consequently, the support reactions without using the classical equilibrium equations of Vector Mechanics.

To solve this problem, we use nonlinear optimization methods, as the objective function is a nonlinear function of the design variables. Additionally, we use deterministic methods, based on the objective function derivatives. Nevertheless, these derivatives are not explicitly obtained, their numerical evaluation is intrinsic to the numerical optimization library. It is important to note that the mathematical model has no constraints, as it only considers the free displacements of the nodes.

Thus, this is a problem that is:

- Single-objective (minimization of total energy);
- Multidimensional (the decision variables are the nodal displacements);
- Unconstrained;

- Deterministic;
- Continuous.

SciPy (Virtanen et al [4]) is a collaborative open-source Python library that provides many algorithms useful for scientific computation. Among its utilities, SciPy provides highly-enhanced implementations of numerical optimization algorithms including Nelder-Mead simplex and Broyden–Fletcher–Goldfarb–Shanno (BFGS).

3 Methodology

3.1 Numerical formulation

To test the hypothesis in the Principle of Least Work, we make use of plane trusses, as they are easily implemented, and the concepts are readily extended for more complex structures. The movement of a plane truss can be described by the displacement $\mathit{\Delta}$ of its nodes, gathered in vector $\boldsymbol{\mathit{\Delta}}$. Let each truss element (bar) have end nodes i, j, with reference, stress-free, positions $x_i^r, y_i^r, x_j^r, y_j^r$. After the loads are imposed and equilibrium is reached, the nodes assumed deformed positions x_i, y_i, x_j, y_j. The reference and deformed lengths of the bar are, then,

$$L^r = \sqrt{\left(x_j^r - x_i^r\right)^2 + \left(y_j^r - y_i^r\right)^2}$$

$$L(\mathit{\Delta}) = \sqrt{\left(x_j - x_i\right)^2 + \left(y_j - y_i\right)^2}$$

It is important to notice that L depends on $\mathit{\Delta}$ in a highly nonlinear manner. Assuming that each rod is of constant area and material, its deformation, strain, normal stress and internal force are easily derived to be

$$\delta = L - L^r; \varepsilon = \frac{\delta}{L^r}; \sigma = E\varepsilon; N = A\sigma.$$

Then, the internal energy of a deformed truss bar is

$$U_{int,i} = \frac{E}{2L^r}\delta^2.$$

As previously stated, the total strain energy of the structure will be the sum of $U_{int,i}$ over the truss bars reduced by the spent energy of the external loads.

3.2 Computational implementation

The computational implementation began by choosing the type of optimization. The idea was to work with a library that did not use explicit derivatives, as mentioned in the Scipy introduction, and specifically the Nelder-Mead method.

After this, a reference truss was assembled in Ftool (Martha [5]), information was provided about the nodes, the connectors (bars), the physical properties of the truss: Modulus of elasticity, area of each connector. The length of each bar was then noted.

The Python implementation defined a function with nodal displacements as inputs and the total system energy as the scalar output. Truss properties and nodal loads are defined inside the computational function. In each evaluation, the nodal displacements are used to compute each bar’s deformation, strain, internal force and strain energy and the work done by external forces. The system energy is defined as the difference between the internal and external work.

This energy function is then feed to Scipy’s optimization algorithms, with nodal displacements as design variables and system energy as the objective function. The optimization algorithm makes use of numerical derivatives in an iterative procedure to update the design variables (nodal displacements). If convergence is achieved, the position of minimal energy is the solution of the algorithm and thus the equilibrium position. If reaction values, internal forces or other quantities are needed, they can be derived in a post-processing phase.

4 Numerical example

4.1 Structure description

As an example of the solution presented, a truss was created consisting of 3 nodes, and 2 bars connecting the nodes, 1 load was also applied at different intensities to the highest node, node 2. Fig. 1 depicts the truss with its loaded node as a circle and the restricted nodes as squares.

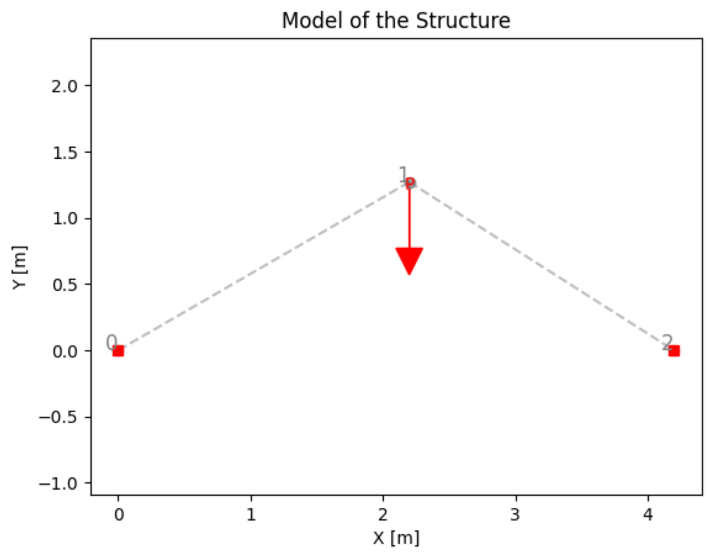

Figure 1: Two-bar truss in the algorithm.

This benchmark problem was addressed by many authors, Segnini [3] among them, whose values are taken for comparison. She used 2 bars with modulus of elasticity, $E = 206\,840\ MPa$, and cross-sectional area, $As = 6.4516\ cm^2$. Each bar has a length of $2.54\ m$ and makes an angle of β = 30° with the horizontal axis. The load is applied on the top node in increments of $889{,}644\ kN$ up to $8\,006{,}796\ kN$.

The optimization problem is run at each iteration, as the objective function changes for each load level. The initial guess is always taken as zero, although a better practice of using the previous load-step result as an initial guess should be investigated. The vertical displacement for the top node is take as the result for comparison between the optimization algorithm and the benchmark solution.

4.2 Validation of the Structural Model

The vertical displacement of the free node is compared, for each load level, to the results obtained by Segnini [3] based on a geometrically exact formulation solved using Newton-Raphson's method. For reference, a linear solution for the last load value is also included in Fig. 2.

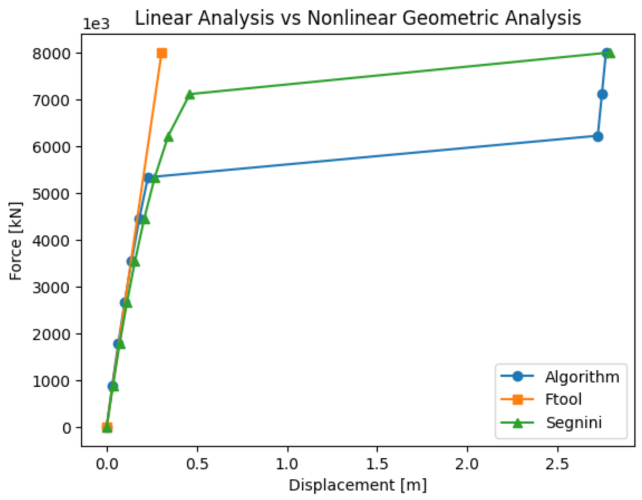

Figure 2: Comparison of results.

Table 1 includes the results for each load step for the linear analysis, the optimization approach and the exact equilibrium solution by Segnini [3].

Table 1. Comparison of results (displacement in meters)

Force [N]	FTool	Algorithm	Segnini [3]
0	0,000	0,0000	0,0000
889 644	0,0339	0,0310	0,0350
1 779 288	0,0678	0,0650	0,0720
2 668 932	0,102	0,100	0,113
3 558 576	0,136	0,139	0,158
4 448 220	0,169	0,181	0,208
5 337 864	0,203	0,229	0,267
6 227 508	0,237	2,724	0,342
7 117 152	0,271	2,747	0,461
8 006 796	0,305	2,770	2,790

5 Conclusions

The results reported on Table 1 and Fig. 2 show an agreement between the two nonlinear formulations shown as well as an expected difference to the linear model used by FTool.

The difference between the optimization and the equilibrium approaches are probably due to the numerical precision of the optimization algorithm. This initial stage explored the optimization library in its simplest form, with no regard to fine tuning error estimates or step sizes. The results prove that such parameters must be analyzed for better suited results.

The major differences noted on the higher load steps allude to the snap-through nature of the structure in the example. It is noted that after a critical load the bars will suffer major strains, and the top node will descent bellow the supports. By then, the truss bars will be in tension rather than in compression. As noted in Section 4.1, the optimization algorithm used a zero estimate for every load-step. For the loads above $6\,000\,kN$, the first optimization steps probably exceeded the threshold and converged for the snapped configuration.

Better results should arise from the use of the previous converged displacement as the initial guess for the next load-step, but the detailed load-displacement curve (with the descending load) should only be achieved by continuation methods.

Despite the need for better exploration of the optimization parameters, results show that formulating the nonlinear equilibrium as an optimization problem and solving this problem using a pure optimization library is possible and can be used for further development.

Acknowledgements. The authors acknowledge the help of Departamento de Engenharia Civil from Universidade Federal de Sergipe for supporting their research.

Authorship statement. The authors hereby confirm that they are the sole liable persons responsible for the authorship of this work, and that all material that has been herein included as part of the present paper is either the property (and authorship) of the authors, or has the permission of the owners to be included here.

References

[1] Hibbeler, Russell Charles. *Mechanics of materials*. MacMillan Publishing Company, 1994.
[2] Ananthasuresh, G. K. *Principles of minimum potential energy and virtual work and their implication in truss optimization*.2022. Slide no Power Point. Available at: Microsoft PowerPoint - Lecture8aVirtualWorkPrinciple (iisc.ac.in). Accessed on July 26, 2024.
[3] Segnini, S. C. A. *Estudo Comparativo de Formulações para a Análise Não-Linear Geométrica de Treliças*. 2000. 129 f. Dissertação (Mestrado em Engenharia Civil) – Faculdade de Engenharia Civil, Universidade Estadual de Campinas, Campinas, 2000.
[4] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3), 261-272.
[5] L. F. Martha. *Ftool – Interactive-Graphics Program for Structural Analysis*. June 2024