XLVI CILAMCE
VITÓRIA - ES - BRAZIL - 2025
XLVI IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING

Python Implementation of the Positional Formulation of the Finite Element Method for Static Analysis Using Two-Dimensional Elements

Diego Reis Figueira [1], Juliano dos Santo Becho[1], Marcelo Greco[1]

[1]*Department of Structural Engineering, Universidade Federal de Minas Gerais*
Av. Presidente Antônio Carlos, 6627- Pampulha, Belo Horizonte, 31270-901, Minas Gerais, Brazil
diegorf@ufmg.br, julianobecho@dees.ufmg.br, mgreco@dees.ufmg.br

Abstract. This work presents a computational implementation, using the Python programming language, of the positional formulation of the Finite Element Method for the static analysis of plane structures modeled by cubic triangular finite elements. In this formulation, the nodal coordinates are considered the problem variables, relative to a fixed coordinate system in space, describing the kinematics of the finite elements. Due to the nonlinearity between the strain energy and its derivatives concerning the nodal positions, the equilibrium analysis of the structure in the deformed configuration is performed using the Newton-Raphson method, characterizing a geometrically nonlinear formulation. The use of Python is due to its widespread adoption in the scientific community, as it is a language with simple and readable syntax, object-oriented features, and the availability of libraries such as NumPy and Matplotlib, which assist in implementing the formulation, solving the system of equations, and graphically presenting the results. The Python implementation is used to analyze a cantilever beam with a concentrated force at the free end and subjected to a large displacement regime. The results are compared with results from the literature to evaluate the consistency of the formulation and the implementation developed in Python.

Keywords: Python, Newton-Raphson Method, Positional Formulation of the Finite Element Method, Two-Dimensional Finite Elements, Nonlinear Analysis

1 Introduction

The nonlinear analysis of structures is an important tool regarding cost reduction in a project, enabling the reduction of structural weight while maintaining safety. In linear analysis, the linear elastic behavior of the material and the regime of small strains and small displacements are considered. In nonlinear analysis, specifically concerning geometric nonlinearity, it is possible to analyze structures in a large displacement regime. In this context, the present work uses the positional formulation of the Finite Element Method, originally developed by Coda [1] and Coda & Greco [2]. In this formulation, the nodal coordinates are considered as the variables of the system of equations. Due to the nonlinearity between the strain energy and its derivatives with respect to the nodal coordinates, the equilibrium analysis of the structure in the deformed configuration is performed using the Newton-Raphson method, characterizing an intrinsically geometrically nonlinear formulation. Thus, this formulation is suitable for analyzing structures with large displacements. In this work, a static analysis of a cantilever beam is performed, with a concentrated force at the free end and subjected to a large displacement regime. The computational implementation uses the Python programming language, due to its clear syntax and the availability of scientific libraries such as NumPy, Pandas, and Matplotlib, which assist in the implementation of the positional formulation.

2 Positional Formulation of the Finite Element Method and Python Algorithm

2.1 Positional Formulation of the Finite Element Method

According to Coda [1], the positional formulation of the Finite Element Method differs from the conventional approach by adopting nodal positions as the fundamental unknowns of the system, instead of displacements. Thus, the strain measure is obtained directly from the current and initial geometric configurations of the structure. Therefore, the formulation is a total lagrangian approach that measures configuration changes from a fixed reference frame. Due to the nonlinearity between the strain energy and its derivatives with respect to the nodal positions, the equilibrium analysis of the structure in the deformed configuration is performed using the Newton-Raphson method, characterizing an intrinsically geometrically nonlinear formulation.

2.2 Total Potential Energy

For static problems, the Total Potential Energy functional, (Π), can be expressed as shown in eq. (1). In this equation, U_e is the internal strain energy, representing the energy stored when the finite elements are deformed, and V is the potential of the external forces, representing the work done by these forces as the structure deforms.

$$\Pi = U_e + \mathrm{V} \tag{1}$$

Figure 1 shows that the functional can be described in terms of the strain energy, U_e, and the applied external forces, F.

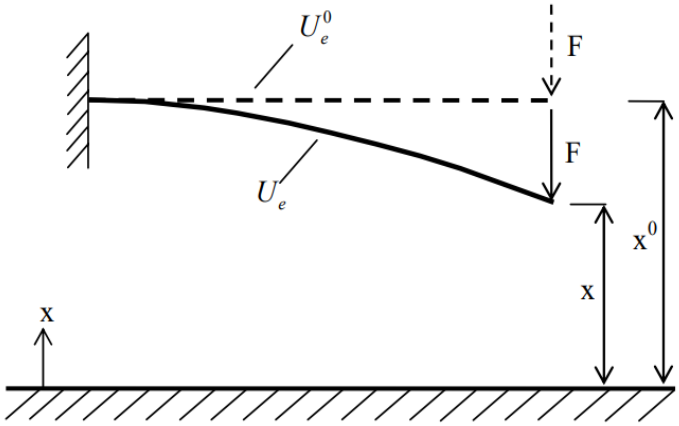


Figure 1. Potential Energy in a structure

The strain energy is provided by the integral (over the initial volume) of the specific strain energy (u_e) with respect to the positions, according to eq. (2), while the potential energy of the external forces is written according to eq. (3), where F_i are the applied forces and X_i are the coordinates where the forces are acting.

$$U_e = \int_{v_0} u_e dv_0 \tag{2}$$

$$V = F_i X_i \tag{3}$$

Thus, by substituting eqs. (2) and (3) into eq. (1), the following equation is obtained:

$$\Pi = \int_{v_0} u_e dv_0 - F_i X_i \tag{4}$$

2.3 Principle of Minimum Total Potential Energy

According to Reddy [3], the principle of minimum total potential energy is more general than the principle of virtual work, as it can be used not only to derive the equilibrium equations but also to investigate the stability of the equilibrium. Therefore, to find the minimum point of a function, one seeks the point where its derivative is zero. Consequently, eq. (5) shows the minimization of the system's equations with respect to the nodal coordinates (positions):

$$\frac{\partial \Pi}{\partial X_s} = \int_{v_0} \frac{\partial u_e(\xi, X_i)}{\partial X_s} dv_0 - F_s = 0 \tag{5}$$

Simplifying eq. (5) to represent the equilibrium equation, where $\frac{\partial \mathrm{U}_e}{\partial \mathrm{X}_s}$ is the internal force vector and $Fext_s$ is the external force vector.

$$\mathrm{g} = \frac{\partial \mathrm{U}_e}{\partial \mathrm{X}_s} - Fext_s = 0 \tag{6}$$

2.4 Newton-Raphson Method

According to Ruggiero [4], the Newton-Raphson method is an iterative method to find approximations for the roots (or zeros) of a real-valued function. The idea of the method is, starting from an initial estimate, to use the tangent line to the graph of the function at the point of that estimate to obtain a better approximation for the root. The process is repeated, generating a sequence of approximations that, under certain conditions, converges rapidly to the root of the function.

Given this, to solve eq. (6), which is of a nonlinear character, it is necessary to expand it in a Taylor series, truncating from the linear terms, from which one obtains:

$$\mathrm{g}(\mathrm{X}^0) + \nabla \mathrm{g}(\mathrm{X}^0)\Delta \mathrm{X} \approx \mathrm{g}(\mathrm{X}) = 0 \tag{7}$$

For optimal use of the Newton-Raphson method, the equation can be rearranged as shown in eq. (8):

$$-\mathrm{g}(\mathrm{X}^0) = \nabla \mathrm{g}(\mathrm{X}^0)\Delta \mathrm{X} \tag{8}$$

Revisiting eq. (6), it can be rewritten by evaluating it at the current position X^0 to obtain the residual vector, which serves as an indicator of equilibrium:

$$\mathrm{g}(\mathrm{X}^0) = \frac{\partial \mathrm{U}_e}{\partial \mathrm{X}_s} - Fext_s = f_{int_s} - f_{ext} \tag{9}$$

Finally, differentiating eq. (6) with respect to the nodal coordinates yields the Hessian matrix, presented in eq. (10):

$$\frac{\partial g}{\partial X_l} = \frac{\partial^2 U_e}{\partial X_s \partial X_l} \tag{10}$$

In the context of nonlinear analysis, the Hessian matrix takes on the role of the stiffness matrix, varying throughout the deformation process.

When the system of eq. (8) is solved, the value of ΔX is found, called the correction vector for the nodal positions. This vector is used to update the nodal positions, according to eq. (11) described below:

$$\mathrm{X} = \mathrm{X} + \Delta \mathrm{X} \tag{11}$$

The process is repeated iteratively, and the positions are updated until the convergence criterion is satisfied:

$$\text{Erro} = \frac{\|\Delta X\|}{\|\mathrm{X}_0\|} \leq tol \tag{12}$$

where tol represents the adopted tolerance.

Marques [5] presents a diagram of the Newton-Raphson method, as shown in the following Fig. 2.

A. INICIALIZAÇÕES

A.1 Inicializa o vetor posição X (com a última posição de equilíbrio conhecida) e faz-se $\Delta X = 0$

B. PARA CADA INCREMENTO DE CARGA

B.1 Atualiza-se o vetor de cargas externas aplicadas, fazendo $F_{ext} = F_{ext} + \Delta F_{ext}$

B.2 Calcula-se o vetor de forças internas global $\left(F_{\text{int}}\right)$

B.3 Calcula-se o vetor de resíduos ou de forças desequilibradas $g = F_{\text{int}} - F_{ext}$

B.4 Monta-se a matriz hessiana global da estrutura ∇g

B.5 Resolve-se o sistema $\nabla g \Delta X = -g$

B.6 Atualiza-se posição $X = X + \Delta X$

B.7 Calcula-se o erro e verifica-se o critério de convergência $erro = \dfrac{\|\Delta X\|}{\|X_0\|} \leq TOL$

B.8 Se o critério de convergência for satisfeito vá para B.1 senão volte para B.2

C. FIM

Figure 2. Diagram of the Newton-Raphson Method

During the computational implementation, a triangular finite element with cubic approximation, named QST, was used, as shown in Assan [6] and Savassi [7]. This element is presented in Fig. 3.

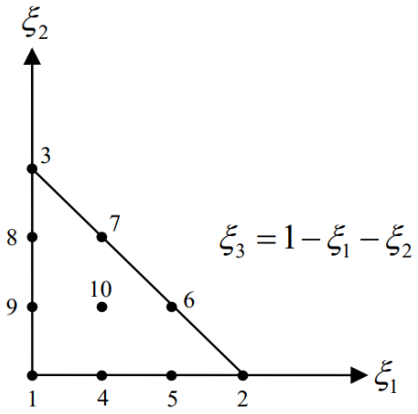


Figure 3. Triangular element with cubic approximation

2.5 Computational Implementation

Kay [8] notes that Python is an object-oriented and open-source programming language, frequently used for rapid application development. Because it has a simple syntax with an emphasis on readability, it reduces the cost of program maintenance, while its vast function library encourages reusability and extensibility.

Given these characteristics, the positional formulation of the finite element method for two-dimensional static analysis was implemented in Python 3.13 and PyCharm 2025.2. In simple terms, the proposed algorithm can be

subdivided into three phases: pre-processing, processing, and post-processing. Fig. 4 presents a flowchart of the developed program's steps. In the pre-processing phase, the problem's discretization and its physical and geometric properties are provided. These are then used by the processing phase to determine the problem's unknowns. Finally, in the post-processing phase, the results are presented through graphs, output files, or tables.

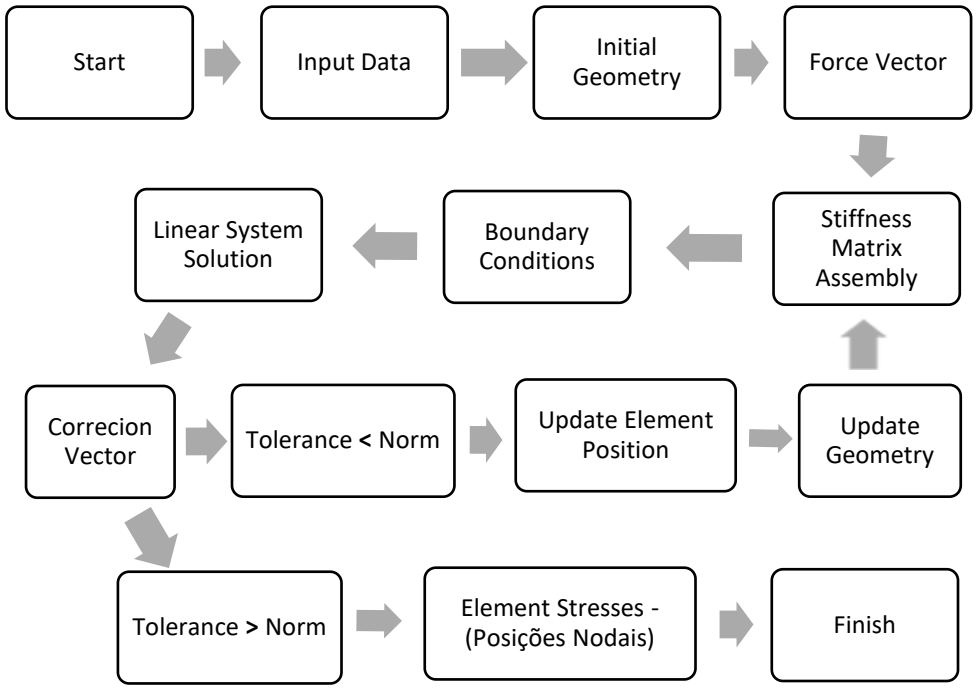


Figure 4. Algorithm Flowchart

3 Numerical Example

To validate the developed computational code, this example analyzes a cantilever beam with a transverse force applied at its free end, as shown in Fig. 5.

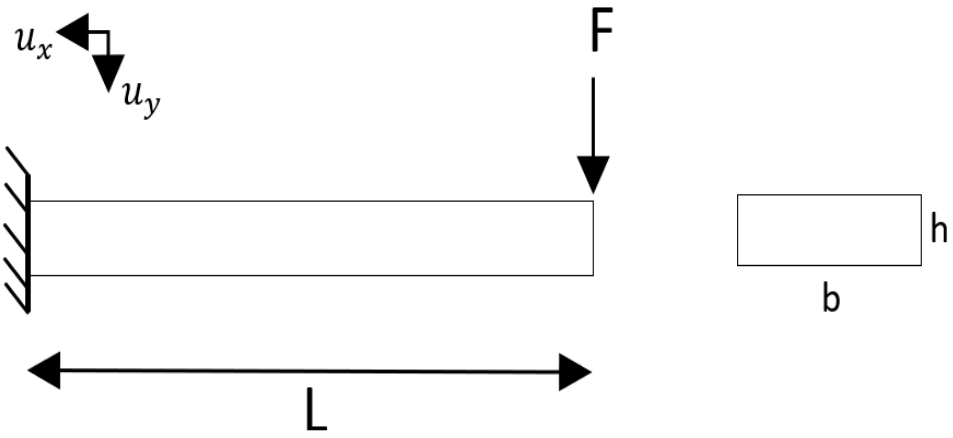


Figure 5. Cantilever beam

The physical and geometric properties of the beam are presented in Tab 1.

Table 1. Input Data

Constitutive relation	Unit	Value
Young's Modulus	Pa	210×10^9
Beam Length	m	10
Section Height	m	0,0178
Section Thickness	m	1
Poisson's Ratio	-	0

Fifty load steps of 200 N each were applied, amounting to a total applied force of 10000 N.

The results obtained from analyzing the beam with the developed computational code are presented in Figs. 6, 7, and 8. For comparison, these figures also include the analytical and numerical results presented by Mattiasson [9], Fujii [10], and Marques [5]. Fig 6 shows the horizontal displacements of the beam's free end throughout the incremental loading process. Meanwhile, Fig. 7 presents the vertical displacements of the free end during the same process. Finally, Fig. 8 displays the final deformed configuration of the beam after the application of 10000 N.

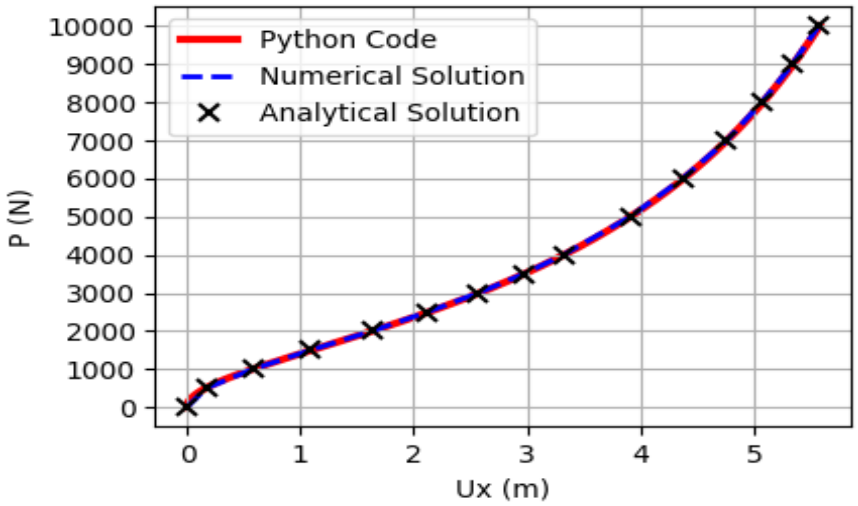


Figure 6. Horizontal displacement versus applied load

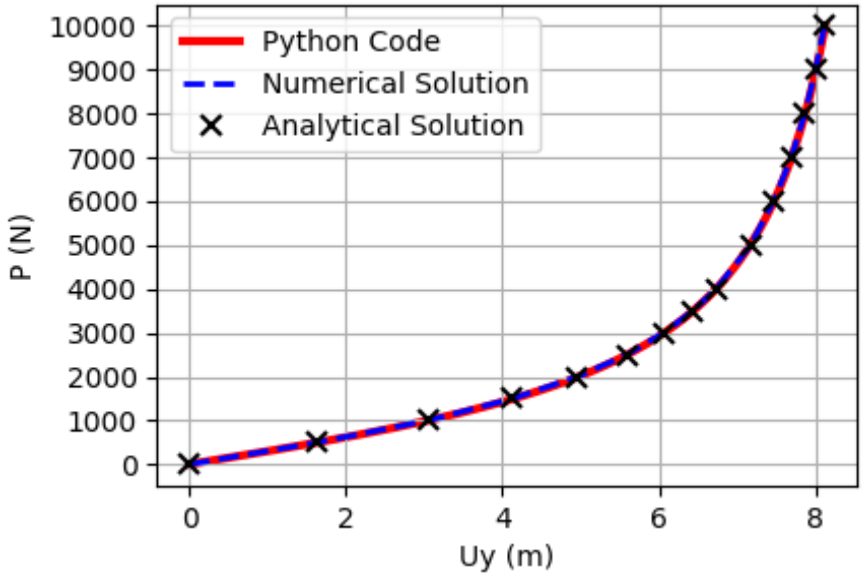


Figure 7. Vertical displacement versus applied load

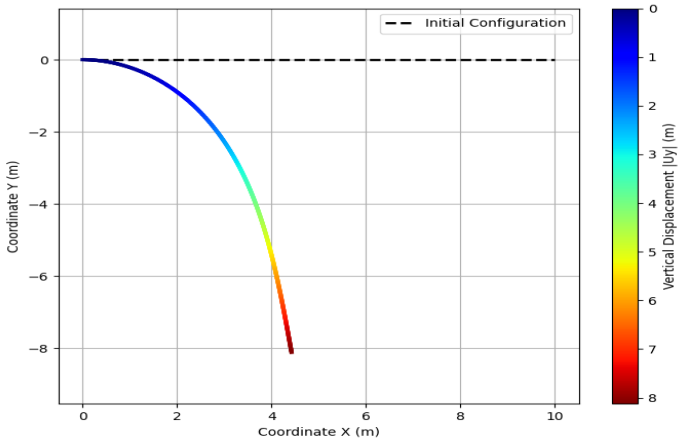


Figure 8. Final deformation of the Beam

4 Conclusions

From the analysis of the cantilever beam, based on the results presented in Figs. 6, 7, and 8, it can be concluded that the computational implementation of the positional formulation of the Finite Element Method in Python was successful. This conclusion is supported by the fact that the obtained results are in good agreement with the expected and observed results in the literature. Furthermore, the results demonstrate the capacity of the positional formulation of the Finite Element Method to describe geometrically nonlinear behavior in cases involving large displacements.

Acknowledgements. The authors would like to thank Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG), Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) and Universidade Federal do Minas Gerais (UFMG) for their research financial support.

Authorship statement. This section is mandatory and should be positioned immediately before the References section. The text should be exactly as follows: 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] Coda, H. B. (2003). "Análise não linear geométrica de sólidos e estruturas: uma formulação posicional baseada no MEF". Tese de Doutorado, Escola de Engenharia de São Carlos, Universidade de São Paulo (USP), São Carlos.
[2] Coda, H.B.; Greco, M. (2003). A simple FEM formulation for large deflection 2D frame analysis based on position description. *Computer methods in applied mechanics and engineering*.
[3] Reddy, J. N. (2006). An Introduction to the Finite Element Method (3rd ed.). McGraw-Hill.
[4] Ruggiero, M. A. G.; Lopes, V. L. R. Cálculo Numérico: Aspectos Teóricos e Computacionais. 2. ed. São Paulo: Pearson Education, 2000.
[5] Marques, G. C. S. C., Estudo e Desenvolvimento de Código Computacional Baseado no Método dos Elementos Finitos para Análise Dinâmica não Linear Geométrica de Sólidos Bidimensionais. Dissertação de Mestrado da Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos. (2006).
[6] Assan, A. E. (2003). Método dos Elementos Finitos: primeiros passos. 2. ed. Editora UNICAMP, Campinas, São Paulo.
[7] Savassi, W. (1996). Introdução ao método dos elementos finitos em análise linear de estruturas. Escola de Engenharia de São Carlos, USP-SP. 260 p.
[8] Kay, R. "Python". Computerworld, May 2, 2005. Available at: https://www.computerworld.com/article/2556925/python.html. Acessed on: Augut 05, 2025.
[9] Mattiasson, K. (1981). Numerical results from large deflection beam and frame problems analysed by means of elliptic integrals. *International journal for numerical methods in engineering*, v.17, p.145-153.
[10] Fujii, F. (1983). A simple mixed formulation for elastica problems. *Computer & Structures*, v.17, p.79-88.