PYTHON ALGORITHM FOR CALCULATION OF INTERNAL FORCES AND DISPLACEMENTS IN BEAMS USING THE FINITE ELEMENT METHOD

Diego R. Figueira, Maria S. M. Sampaio

Department of Civil Engineering, School of Technology, University of Amazonas State
Darcy Vargas Avenue, 69050-020, Amazonas/Manaus, Brazil
drfg.eng@uea.edu.br, msampaio@uea.edu.br

Abstract. A beam finite element under linearly distributed loading is presented. The finite element has two nodes and two degrees of freedom per node, totaling four degrees of freedom per element. The nodal parameters are translations and rotations. A cubic polynomial and a linear polynomial are used, respectively, to approximate the solutions of the problem and describe the loading. The total potential energy functional and the stationarity principle of energy are used to obtain the finite element equations system. Python 3.9.0 programming language and PyCharm IDE 2020.2 are used to implement the developed algorithm. After determining the problem unknowns by solving the system of equations, the internal forces and support reactions are determined in the post-processing phase. To validate the implemented code, a simply supported beam subjected to a uniformly distributed load was chosen. The results shown by the algorithm were compared with analytical and FTOOL solutions, and show that the code was successfully implemented.

Keywords: Python algorithm, Finite Element Method, beams, structural analysis

1 Introduction

Engineering programs related to structural analysis help the user to find solutions for a given problem. Many engineering problems can be expressed in terms of partial differential derivatives and have analytical solution. However, in real problems, this feature is not observed and it is necessary to use a numerical tool to obtain approximate solutions to the problem. A very widespread tool for calculating approximate solutions is the Finite Element Method (FEM). Its main characteristic is to divide a body into finite elements, connected by nodes, and obtain an approximate solution to the analyzed problem. By dividing the body into elements, so that they can be studied one by one at the intersection of nodes, the analysis becomes more precise and specific for critical points. According to Martha [1], the (FEM) is used to analyze different types of engineering problems such as: displacements and stresses in mechanical parts, dams, mines, towers, buildings and roofs.

There are different methods to derive the finite element system of equations, among them the variational methods and the weighted residuals method stand out. In this paper an energy method was used to derive the system of equations of beam finite element. This method consists of writing the functional total potential energy of the element and applying the Principle of Energy Stationarity. The principle of energy stationarity states that of all displacements fields which satisfy the prescribed constraint conditions, the correct state is that which makes the total energy of the structure a minimum. The choice of an energy method is due to the simplicity of the physical concepts of external work and strain energy involved in this formulation when compared to the mathematical refining of other methods.

So, the main purpose of this paper is to develop a computational code in Python, so that it can calculate the internal forces, support reactions and displacements in beams using the FEM.

2 Beam finite element formulation and Python algoritm

2.1 Beam finite element formulation

According to ABNT NBR 6118 [2], a beam is a linear element in which bending is dominant, that is, physical effort in which the deformation occurs perpendicularly to the axis of the body. Geometrically, linear element is one in which the linear length is at least three times greater than the largest dimension of the cross section. For particular cases of geometry, loading and boundary conditions, these structures have analytical solutions. However, in real structures, such feature is not observed being necessary to use numerical tools to obtain an answer to the problem.

According to Moraes [3], the conceptual study of the finite element method theory is increasingly relevant for the analysis of structures, as these concepts are present in practically all available software and, undoubtedly, can and should be used as indispensable tools in the daily life of the structural engineer. According to the author, the FEM is based on a discretization of domains, and may have arbitrary irregular geometries, thus generating basic polynomial elements, which allow, through the resolution of the approximations in their nodes, to arrive at an approximate behavior of the structure as a whole. When using the finite element method, whenever possible, depending on the properties of the element, an approximate satisfactory solution with the smallest possible number of elements should be reached. This must be done in order to save computational resources and still achieve realistic results.

So, let a beam finite element of length (L), Young modulus (E) and inertial moment (I) be subjected to a linearly distributed load q(x) as shown in Fig. 1a. Since, by hypothesis, the beam is subjected to bending forces, the nodal parameters associated with nodes i and j of this finite element are rotations and translations (Fig. 1b).

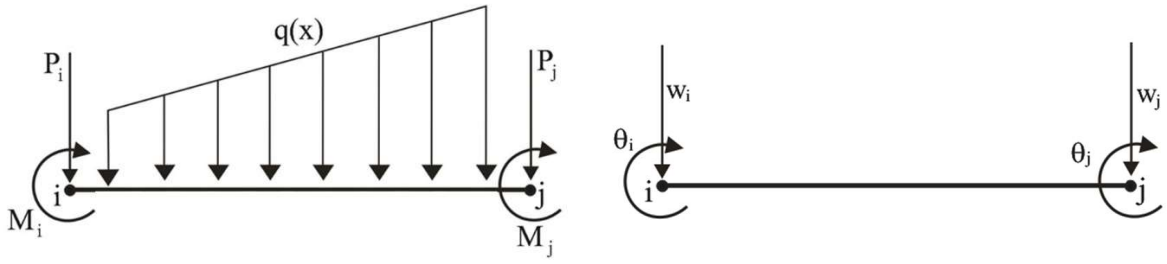

Figure 1. Beam finite element under a linearly distributed load and nodal parameters.

The total energy functional Π_i for this beam element is given by Eq. 1:

$$\Pi_i = \frac{1}{2}\int_0^L EI\, u''^2 dx - \int_0^L qu dx - Pu(x)\Big|_{x=j}^{x=i} - Mu'(x)\Big|_{x=j}^{x=i} \tag{1}$$

where, the first term after the equality is the strain energy of the beam element and the following terms are, respectively, the work done by the linearly distributed loading, the work done by the loads applied to the nodes, and the work done by the bending moments applied to the nodes.

For this element, a third degree polynomial is adopted as the approximation function of the displacements and a linear polynomial is adopted as the approximation function of the linearly distributed load q(x). These approximations are given respectively in Eqs. (2) e (3):

$$u(x) \cong v(x) = ax^3 + bx^2 + cx + d \tag{2}$$

$$q(x) = a_0 x + a_1 \tag{3}$$

In Eqs. (2) and (3) a, b, c, d, a_0 e a_1 are constant obtained from the boundary conditions of the finite element (Fig. 1b) and the linearly distributed load (Fig. 1a), respectively.

Applying the boundary conditions in Eqs. (2) e (3), performing all necessary operations and rewriting the approximations obtained in a dimensionless coordinate system with $\xi = x/l$, we obtain, respectively:

$$v(\xi) = \left(1 - 3\xi^2 + 2\xi^3\right) w_i + \left(\xi - 2\xi^2 + \xi^3\right) l\theta_i + \left(3\xi^2 - 2\xi^3\right) w_j + \left(\xi^3 - \xi^2\right) l\theta_j \tag{4}$$

$$q(\xi) = \left(1 - \xi\right) q_i + \xi q_j \tag{5}$$

Equation 4 can be written as:

$$v = \sum_{j=1}^{m} \alpha_j \phi_j \tag{6}$$

where α_j are the nodal parameters and ϕ_j are the shape functions.

Deriving Eq. (4) twice, substituting this result and Eq. (5) in Eq. (1), making the square, integrating, minimizing the functional, that is, deriving in relation to the unknowns w_i, θ_i, w_j, θ_j, one arrives at a system of equations written in matrix form as:

$$\begin{bmatrix} \frac{12EI}{L^3} & \frac{6EI}{L^2} & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{4EI}{L} & -\frac{6EI}{L^2} & \frac{2EI}{L} \\ -\frac{12EI}{L^3} & -\frac{6EI}{L^2} & \frac{12EI}{L^3} & -\frac{6EI}{L^2} \\ \frac{6EI}{L^2} & \frac{2EI}{L} & -\frac{6EI}{L^2} & \frac{4EI}{L} \end{bmatrix} \begin{Bmatrix} w_i \\ \theta_i \\ w_j \\ \theta_j \end{Bmatrix} = \begin{Bmatrix} P_i \\ M_i \\ P_j \\ M_j \end{Bmatrix} + \begin{Bmatrix} \frac{7}{20} Lq_i + \frac{3}{20} Lq_j \\ \frac{1}{20} L^2 q_i + \frac{1}{30} L^2 q_j \\ \frac{3}{20} Lq_i + \frac{7}{20} Lq_j \\ -\frac{1}{30} L^2 q_i - \frac{1}{20} L^2 q_j \end{Bmatrix} \tag{7}$$

wich can be conveniently rewrite as:

$$[K]\{u\} = \{F\} + \{Q\} \tag{8}$$

where $[K]$ is the element stiffness matrix, $\{u\}$ is the element nodal displacements vector, $\{F\}$ is applied load vector and $\{Q\}$ is the linearly distributed load vector.

Finally, for each finite element i, a functional Π_i is assembled that, added to the other finite elements, form the functional Π for the entire domain (Eq. 9). For more details the authors suggest to consult Assan [4].

$$\Pi = \sum_{i=1}^{n} \Pi_i \tag{9}$$

2.2 Computational implementation

Kay [5] emphasizes that "Python is an object-oriented and open source programming language often used for rapid application development. Having simple syntax, with an emphasis on readability, reduces the cost of maintaining the program, while its vast library of functions encourages reuse and extensibility".

Because of these characteristics, the beam finite elemento formulation was implemented in Python 3.9.0 and PyCharm 2020.2 The proposed algoritm can be subdivided into three phases, that is, pre-processing, processing and post-processing. Figire 2 shows a flowchart of the steps of the developed program. In the pre-processing phase, the discretization and the physical and geometric properties of the problem are provided and then, used by the processing phase to determine the unknowns of the problem. Finally, in the post-processing phase, the unknowns, which are nodal displacements, are used to determine the internal forces and support reactions.

It is important to mention that the contributions to the global matrix are effected through the incidence rules that relate the nodes of a given element with its final position in the global system of the structure. These incidence rules can be seen in Fig. 3 e the related subroutine in Python is shown in Fig. 4. For more details the authors suggest to consult Soriano [6].

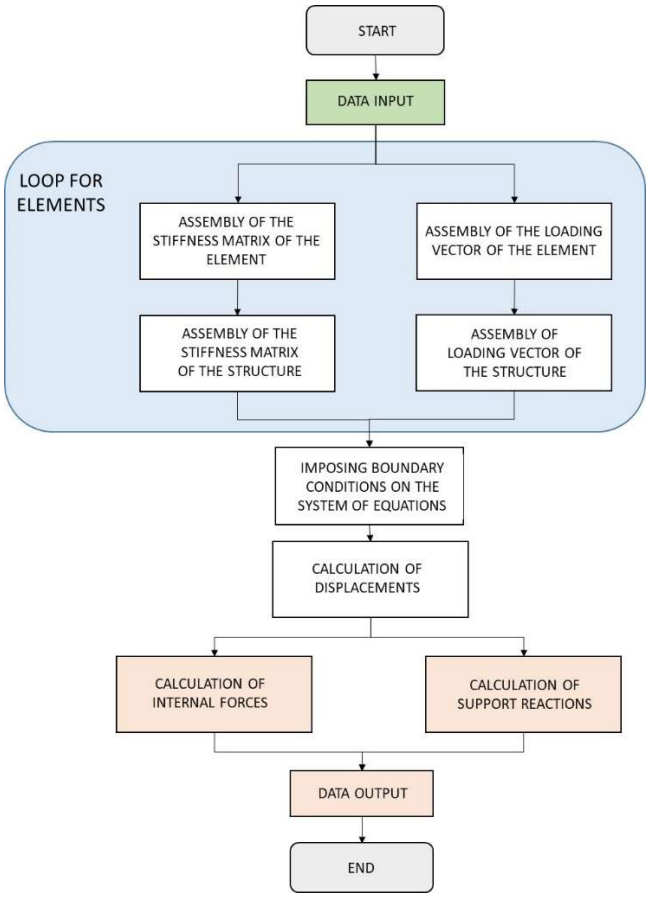

Figure 2. Algoritm flowchart

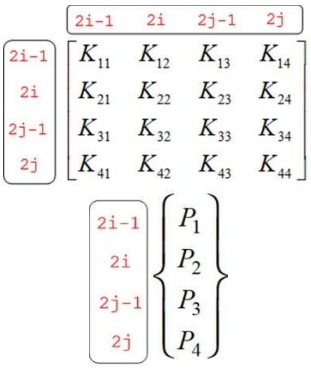

Figure 3. Incidence rules

```
KGLOBAL = []
for i in range((nno * 2)):
    KGLOBAL.append([])
    for j in range((nno * 2)):
        zero = 0
        KGLOBAL[i].append(zero)
for i in range(nelementos):
    a = 2 * NI[i] - 2
    b = 2 * NI[i] - 1
    c = 2 * NF[i] - 2
    d = 2 * NF[i] - 1

    KGLOBAL[a][a] = KGLOBAL[a][a] + KELEM[i][0][0]
    KGLOBAL[a][b] = KGLOBAL[a][b] + KELEM[i][0][1]
    KGLOBAL[a][c] = KGLOBAL[a][c] + KELEM[i][0][2]
    KGLOBAL[a][d] = KGLOBAL[a][d] + KELEM[i][0][3]

    KGLOBAL[b][a] = KGLOBAL[b][a] + KELEM[i][1][0]
    KGLOBAL[b][b] = KGLOBAL[b][b] + KELEM[i][1][1]
    KGLOBAL[b][c] = KGLOBAL[b][c] + KELEM[i][1][2]
    KGLOBAL[b][d] = KGLOBAL[b][d] + KELEM[i][1][3]

    KGLOBAL[c][a] = KGLOBAL[c][a] + KELEM[i][2][0]
    KGLOBAL[c][b] = KGLOBAL[c][b] + KELEM[i][2][1]
    KGLOBAL[c][c] = KGLOBAL[c][c] + KELEM[i][2][2]
    KGLOBAL[c][d] = KGLOBAL[c][d] + KELEM[i][2][3]

    KGLOBAL[d][a] = KGLOBAL[d][a] + KELEM[i][3][0]
    KGLOBAL[d][b] = KGLOBAL[d][b] + KELEM[i][3][1]
    KGLOBAL[d][c] = KGLOBAL[d][c] + KELEM[i][3][2]
    KGLOBAL[d][d] = KGLOBAL[d][d] + KELEM[i][3][3]
```

Figure 4. Global Matrix in Python

3 Numerical Example

A bi-supported beam under uniformly distributed load shown in Fig. 5 is analyzed by the implemented formulation. The adopted geometrical properties are L = 4 m, b = 0,20 m and h = 0,40 m. The transverse applied load and the Young Modulus are given, respectively, by q = 10 kN/m and E = 28 GPa.

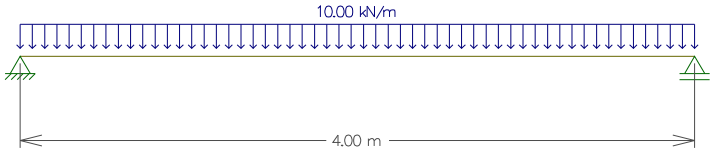

Figure 5. Bi-supported beam under uniformly distributed load

The solutions obtained with the implemented formulation are compared with the FTOOL [7] solution and analytical solution from Strength of Materials as can be consulted in Hibbeler [8], that is:

$$u = \frac{-qx}{24\,EI}\left(x^3 - 2Lx^2 + L^3\right); \quad u_{máx} = \frac{-5qL^4}{384\,EI}; \quad \theta_{máx} = \pm\frac{qL^3}{24\,EI} \tag{10}$$

Analyzing the relative differences obtained for maximum displacement and rotation, internal forces and support reactions for a mesh with two elements regarding the reference solutions (Table 1) it is possible to conclude that the implemented formulation is returning consistent solutions.

Although the results are satisfactory, it is important to remember that FEM provides discrete solutions as a function of the chosen discretization and that at the limit of the discretization the numerical solution tends to the analytical solution. To demonstrate these characteristics, the beam was discretized into uniform meshes with 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 elements. The behavior of the displacement solution according to the mesh refinement can be seen in Figure 6.

Table 1. Numerical and analytical values for selected points: analysis of results.

	Python Algorithm	Analytical Solution	Relative diference (%)	Software FTOOL	Relative diference (%)
Vertical reaction in x = 0 (kN)	20,00001632	20,0	0,0000816%	20,0	0,0000816%
Deflection in x = 2 m (mm)	1,116037	1,116071	0,00304%	1,116071	0,00304%
Rotation in x = 0 (rad)	$8{,}928299 \times 10^{-4}$	$8{,}928571 \times 10^{-4}$	0,0030%	$8{,}928571 \times 10^{-4}$	0,0030%
Shear force in x = 0 (kN)	20,00001632	20,0	0,0000816%	20,0	0,0000816%
Bending moment in x = 2 m (kN.m)	20,000006726	20,0	0,00003363%	20,0	0,00003363%

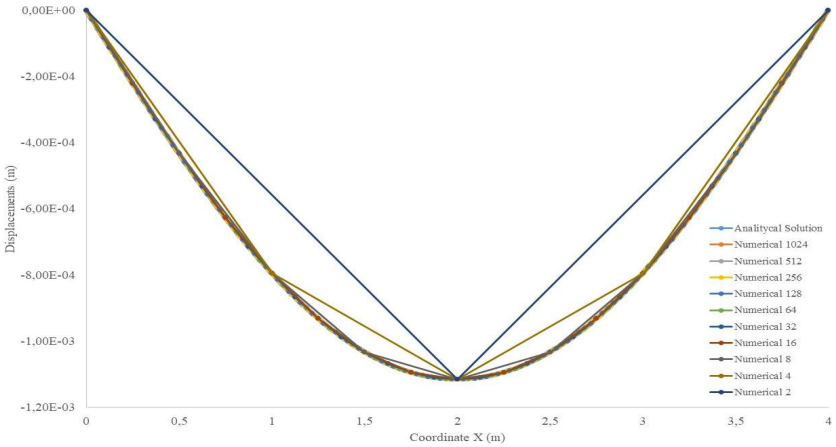

Figure 6. Beam displacements for different discretizations.

4 Conclusions

A beam finite element formulation in Python language was successfully implemented as it is possible to conclude from the analysis of the obtained results. From the values in Table 1, it was possible to verify that the algorithm made in Python proved to be efficient, since the values of vertical reaction in the nodes, deformation, rotation and internal forces were close to or equal to the analytical solution and the Ftool software, confirming that the algorithm can be used to calculate internal forces displacements of a beam.

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]MARTHA, L.F. *Análise de Estruturas: Conceitos e Métodos Básicos*. Rio de Janeiro: Elsevier, 2010.
[2]ABNT NBR 6118: Projeto de estruturas de concreto – Procedimento. Rio de Janeiro, 2014.
[3]MORAES, A.J. "O Método dos Elementos Finitos e a Engenharia Civil". *Revista Especialize On-line IPOG* – Goiânia, v. 01, n. 10, dez. 2015.
[4]ASSAN, A. E. *Método dos Elementos Finitos: primeiros passos*. Campinas, S.P: Editora da UNICAMP, 1999.
[5]KAY, R. "Python". *Computerworld*, May 2, 2005. Available at: https://www.computerworld.com/article/2556925/python.html. Accessed on: August 14, 2021.
[6]SORIANO, H.L. *Formulação Matricial e Implementação Computacional*. Rio de Janeiro: Editora Ciência Moderna, 2005.
[7]FTOOL. Disponível em: https://www.ftool.com.br/Ftool.
[8]HIBBELER, R.C. *Resistência dos Materiais*. 7a. Ed. São Paulo: Pearson Prentice Hall, 2010.