International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2022): 7.942

A Graph-Based Approach for the Analysis of Framed Structures in Civil Engineering

Edgar Quispe Ccapacca[1*], Percy Huata Panca[2], Hugo David Calderón Vilca[3]

[1]Postgraduate unit of the Faculty of Statistical and Informatics Engineering, Escuela de Posgrado de la Universidad Nacional del Altiplano, Universidad Nacional del Altiplano de Puno, Puno - Perú.

[2]Faculty of Statistical and Informatics Engineering, Universidad Nacional del Altiplano de Puno, Puno - Perú.

[3]Department of Software Engineering, Universidad Nacional Mayor de San Marcos, Lima -Perú.

[*]Corresponding author. E-mail: *edgar.edqc[at]gmail.com*

Abstract: ***Framed structures such as trusses, beams, and frames are used in the construction of buildings, bridges, transmission towers, etc. The analysis of such structures is necessary as a basis for the design, since it allows knowing the behavior of the structure subjected to different load conditions. This article presents the computational aspects for the development of an interactive graphical computer application for the analysis of framed structures. The application was programmed in Java 2D using the stiffness method. For the programming of the application, a graph approach is proposed, since the models of framed structures are topologically graphs. The developed application covers the three stages of structure analysis: pre-processing, calculation and post-processing. The results of the analysis of structures are compared with those obtained by commercial software, obtaining coincidence in the results, for which it is concluded that the presented approach is useful to develop graphic software for the analysis of framed structures.***

Keywords: Framed Structures, Graph Abstract Data Type, Stiffness Method

1. Introduction

A structure is part of a construction whose purpose is to support the loads to which it is subjected. Framed structures such as: trusses, beams and frames are formed by prismatic members joined at their ends. Structural analysis studies the effects of loads on a structure [1–4].

A real structure presents complex geometry, therefore, its analysis requires simplifying it through an analytical model that outlines the structure as a line diagram [2, 3]. Topologically, this line diagram is a graph, where the data is associated to edges (members) and vertices (nodes).

The analysis of a structure consists of three stages: pre-processing, calculation and post-processing [5, 6].

In preprocessing, the geometry of the model (coordinates and connectivity between nodes), member properties and boundary conditions are defined.

The calculation consists of solving the model using the stiffness matrix method, which is described by Kassimali [2]. The stiffness method, in summary, consists of:

- Calculate the stiffness matrix ***K*** of the structure
- Calculate the force vector ***F*** of the structure.
- Apply boundary conditions on K and F, thus K and F are reduced.
- Solve the system ***KU***=***F*** to calculate the displacements ***U*** of the nodes of the model.
- Calculate the reaction forces at the supports.
- Calculate the internal forces at the ends of the members.

Post-processing consists of presenting the deformed shape of the model and the internal force diagrams: axial, shear and bending moments [7–9].

The analysis of a structure is laborious, therefore, many studies describe its computer implementation.

For preprocessing, Nogueira and Bezerra [10], Bakošová et al. [11], Neves et al. [12], Zotkin et al. [13], Barrantes and Hernández [14] and Barrera [15] describe graphical interfaces where the user must tabulate the data and the coordinates and connectivity between nodes to draw the structure model. Barhate and Ladhane [5] and Francois et al. [6] describe the data, edited in an m-file. Neiva et al. [8], Barreto Bezerra et al. [16], Htwe and Khaing [17] and Patil and Annigeri [18] describe the data, edited in text file. Pallares M. et al. [19], Chen [20], and Villagómez et al. [9] describe console applications where data is entered one by one. Pamnani et al. [21] describe the data input through Matlab command window. Godoi et al. [22] describe the data in an Excel spreadsheet. All these procedures are laborious when the model includes a lot of data, for which an appropriate data structure is required to facilitate the geometric representation of models and data storage.

For the calculation, Bakošová et al. [11], Barrantes and Hernández [14], Chen [20], Neves et al. [12], Neiva et al. [8], Barhate and Ladhane [5], Htwe and Khaing [17], and Pamnani et al. [21] describe MATLAB scripts to analyze structures subjected to certain types of loads. De Oliveira et al. [23], Nogueira and Bezerra [10] and Godoi et al. [22], describe the use of spreadsheets to analyze structures subjected to certain types of loads. This shows the lack of algorithms to implement a program that deals with various topics such as: thermal effects, fabrication errors, elastic supports, trapezoidal distributed loads, point loads, nodal loads, and support movements.

For post-processing, Pamnani et al. [21] present diagrams of shear forces and bending moments for the specific case of the analysis of four beams. Neves et al. [12] present the bending moment diagram corresponding to the analysis of a beam. Barrantes and Hernández [14] present the deformed shape of the analysis of three structures. Chen [20] presents the bending moment diagram of a frame analysis. Francois et al. [6] present the diagrams corresponding to the analysis of a frame. Villagomez et al. [9] present the diagrams of shear forces and bending moment corresponding to the analysis of a beam. Nogueira and Bezerra [10], Bakošová et al. [11], Barhate and Ladhane [5], Neiva et al. [8], Barreto Bezerra et al. [16], Htwe and Khaing [17], Godoi et al. [22], Barrera [15] and Patil and Annigeri [18] present the numerical results of the analysis of structures, printed in text files or on screen. Therefore, algorithms are required to draw the internal force diagrams and the deformed shape of any structure.

This research seeks to contribute with computational aspects to develop graphic and interactive software for the analysis of trusses, beams and frames. For preprocessing, a Graph Abstract Data Type (Graph ADT) is defined, which facilitates the use of the mouse to build the geometric model on a canvas, assign data to selected members and nodes and dynamically modify the model. For the calculation and post-processing, algorithms have been developed to calculate nodal displacements, internal forces, draw the deformed shape of the models and the internal force diagrams: axial, shear and bending moments. The algorithms cover topics such as: thermal effects, elastic supports, fabrication errors, imposed displacements, nodal loads, point loads, and trapezoidal distributed loads. This research is useful for engineering professionals, who only require basic programming knowledge to code the algorithms and obtain software with similar calculation capacity as commercial software and according to their requirements.

2. Methods

For the development of a program for analysis of framed structures, a graph-based approach is presented, which includes: pre-processing, calculation and post-processing, as illustrated in Fig. 1.

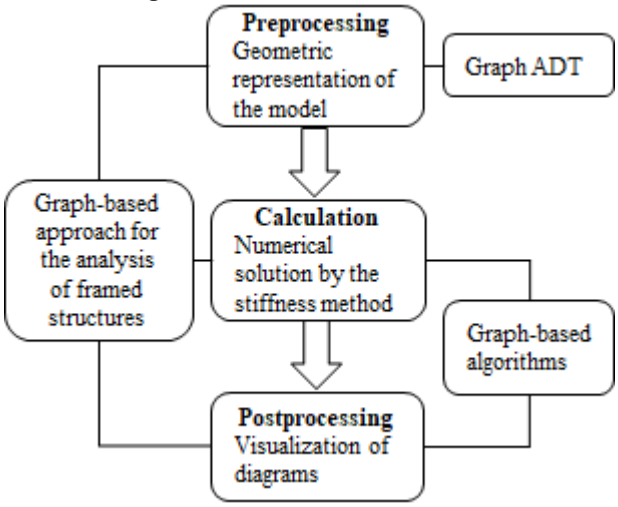

Figure 1: Graph-based approach for the analysis of framed structures

The computer implementation begins with the definition of the Graph ADT, which stores all the data, so that the model is redrawn on the screen each time its data or geometry are modified. The Graph ADT includes operations to add and remove vertices and edges, for model building. Subsequently, algorithms are designed for both calculation and post-processing. The calculation is performed by the stiffness method, where the solution of the ***KU=F*** system is obtained by the Gauss-Seidel method, whose algorithm is described by Chapra and Canale [24]. In the post-processing stage, the deformed shape of the members and the internal force diagrams are drawn from expressions formulated in terms of Macaulay functions. Finally, a graphical user interface is developed, which facilitates data input. The graphical interface includes a canvas where the model can be drawn. The numerical results are compared with those of the SAP2000 software.

3. Development of the graph-based approach

3.1 Definition of a Graph ADT to represent analytical models of framed structures

Figure 2 shows a structure model, which is a graph, whose vertices correspond to the nodes (joints) and the edges to the members of the model.

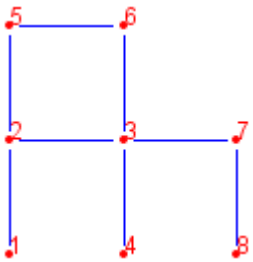

Figure 2: Graph representing a model

For each member, two edges directed in opposite directions are associated, as illustrated in Fig. 3.

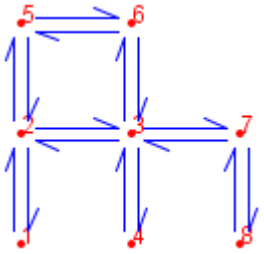

Figure 3: Directed edges of a model

The reason for using directed edges is that they establish two local *x-y* coordinate systems for each member, as shown in Fig. 4. The z^+ axis is assumed to be perpendicular to the screen and pointing at the viewer.

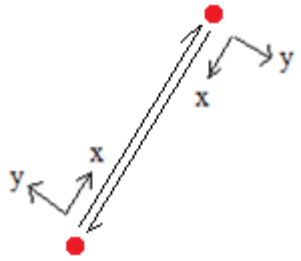

Figure 4: Local coordinate systems of a member given by its directed edges

Selecting a directed edge to assign loads on a member avoids ambiguity in the location and sign of the loads. For example, in Fig. 5 for edge AB, the force *P* is negative and is located at a distance *a* from A, while for edge BA the force P is positive and acts at a distance *b* from B.

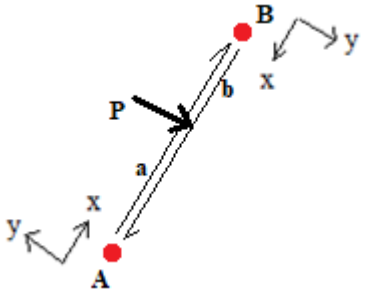

Figure 5: Load on directed edge

In a similar way, temperatures can be assigned on the surfaces of a member. In Fig. 6 for edge AB the temperature T1 acts on the upper surface of the member and T2 on the lower surface, while for BA, T2 acts on the upper surface and T1 on the lower surface.

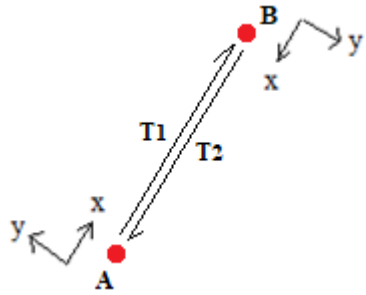

Figure 6: Temperatures on directed edge

To access the nodes of a member, each directed edge (e) stores a reference to its origin vertex (e.origin) and its opposite edge (e.opposite), thus the two nodes of a member are: e.origin and e.opposite.origin, as illustrated in Fig. 7.

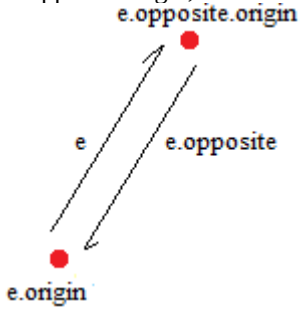

Figure 7: Incidence between edges and nodes

Each node *v* stores a list containing the edges that leave *v*, this makes it possible to access the incident members in *v* and the adjacent nodes of *v*.

Attributes related to topics such as: thermal effects, fabrication errors, elastic supports, trapezoidal distributed loads, and support movements have been included.

Each vertex *v* has the following attributes:

Id :Vertex identifier.
Inc :List of incident edges leaving *v*.
sel :Value that is *true* if *v* is selected, default is *false*.
x,y :Vertex coordinate, default value for *z* is 0.
i,j,k :Identifiers for the degrees of freedom in the *X* and *Y* directions and about the *Z* axis.
kx :Spring constant in *X* direction, default value is 0
ky :Spring constant in *Y* direction, default value is 0
kθ :Spring constant about the Z axis, default is 0.
tx :Restraint in *X* direction, 1: restrained, 0: free, default 0
ty :Restraint in *Y* direction, 1: restrained, 0: free, default 0
rz :Rotational restraint, 1: restrained, 0: free, default is 0.
Fx :Force in *X* direction, default value is 0.
Fy :Force in *Y* direction, default value is 0.
Mz :Moment about the global *Z* axis, default is 0.
u :Displacement in *X* direction, default is 0.
v :Displacement in *Y* direction, default is 0.
θ :Displacement about the *Z* axis, default is 0.

Each directed edge*e* has the following attributes:

opposite :Edge directed in the opposite direction to *e*.
origin :Origin vertex of *e*.
L :Length of the member.
cx,cy :Direction cosines of the directed edge *e*
sel :Value that is *true* if *e* is selected, default is*false*.
E :Modulus of elasticity.
A :Cross-sectional area.
I :Moment of inertia.
Tt :Temperature of the top surface of *e* default is 0.
Tb :Temperature of the bottom surface of *e* default is 0
h :Depth h of the member cross-section.
α :Coefficient of thermal expansion.
ΔL :Fabrication error, (+) if member is longer, (-) if member is shorter, default is 0.
Ni :Internal axial force at the beginning of *e*, default 0.
Vi :Internal shear force at the beginning of *e*, default 0
Mi :Internal bending moment at the beginning of *e*, default is 0.
ui :Displacement in the local *x* direction at the beginning of *e*.
vi :Displacement in the local *y* direction at the beginning of *e*.
θi :Displacement about the *Z* axis at the beginning of *e*.
P :Vector of transverse point loads.
N :Vector of axial point loads.
M :Vector of bending moments.
w :Vector of transverse distributed loads.
wN :Vector of axial distributed loads.

Graph G contains the following attributes:

V :List of vertices in *G*
E :List of directed edges in *G*

Graph operations are defined for insertion and elimination of

vertices and edges, which modify the incidence between edges and vertices when the model changes its geometry.

a) To insert a vertex, its coordinates (x,y) are required, as indicated by Algorithm 1.

Algorithm 1 Add a vertex

```
ADD-VERTEX (x, y)
1  v = new vertex (x, y)
2  add v to list V
3  return v
```

b) An edge $e1$ between $v1$ and $v2$ is inserted if e1 does not exist. Both $e1$ and its opposite $e2$ are added in E and in the lists $v1$.Inc and $v2$.Inc respectively. Algorithm 2 returns e1 (the edge coming out of v1), if $e1$ already exists it returns null. The length and direction cosines of the directed edges are calculated and stored when inserting the edge.

Algorithm 2 Add an edge

```
ADD-EDGE (v1, v2)
    //check if there is already an edge e
1   for each edge e∈v1.Inc
2       if e.opposite.origin == v2
3           return NIL
    //create e1 and e2 with origin at v1 and v2
4   e1 = new edge(v1)
5   e2 = new edge(v2)
    // link both directed edges e1 and e2
6   e2.opposite = e1
7   e1.opposite = e2
8   add e1 to list E
9   add e2 to list E
    // bind e1 to v1 and e2 to v2.
10  add e1 to list v1.Inc
11  add e2 to list v2.Inc
    // assign length L to e1 and e2
12  e1.L = e2.L = L = √((v2.x - v1.x)^2 + (v2.y - v1.y)^2)
    // direction cosines of e1 and e2
13  e1.cx = (v2.x - v1.x) / L
14  e1.cy = (v2.y - v1.y) / L
15  e2.cx = -e1.cx
16  e2.cy = -e1.cy
17  return e1
```

c) Removing a vertex v includes removing the edges coming out of v and their respective opposite edges, as described by Algorithm 3.

Algorithm 3 Remove a vertex

```
REMOVE-VERTEX (v)
1  if v == NIL
2      return
   // traverse the incident edges coming out of v
3  for each edge e∈v.Inc
4      b = e.opposite.origin
5      remove e.opposite from list b.Inc
6      remove e from list E
7      remove e.opposite from list E
   remove v from list V
```

d) Removing a member from the model includes removing its two directed edges. If the member is isolated, its end nodes must be removed. If the member has a free end, the free end node must be removed. If the member has no free ends, the end nodes are not removed. Algorithm 4 details the procedure.

Algorithm 4 Remove edges associated with a member

```
REMOVE-EDGE (e)
1   if e == NIL
2       return
3   a = e.origin
4   b = e.opposite.origin
5   if b.Inc.size== 1
6       if a.Inc.size== 1// edge with free ends
7           REMOVE-VERTEX(a)
8           REMOVE-VERTEX(b)
9       else// edge with a free end
10          remove e from list a.Inc
11          REMOVE-VERTEX(b)
12  else if a.Inc.size== 1 // edge with a free end
13      remove e.opposite from list b.Inc
14      REMOVE-VERTEX(a)
15  else    // edge with ends attached to other edges
16      remove e from list a.Inc
17      remove e.opposite from list b.Inc
18  remove e from list E
19  remove e.opposite from list E
```

1.1 Design of graph-based algorithms for the analysis of framed structures

The following nomenclature is used in the algorithms:

K :Structure stiffness matrix.
F :Nodal load vector of the model.
U :Nodal displacements.
k :Member stiffness matrix in the global system.
f :Nodal load vector of a member in the global system.
fe :Member local fixed-end force vector at the origin-node, due to loads, whose axial, shear, and bendingmoment components are respec-
n tively{*fab*,*fsb*,*fmb*}.
u(x) :Number of degrees of freedom of the structure.
v(x) :Displacement of the member's centroidal axis in the local *x* direction, at a distance *x* from the origin.
N(x) :Deflection of the member's centroidal axis in the local y direction, at a distance x from the origin.
V(x) :Axial force at the member section at a distance *x* from the origin of the local xy coordinate system.
M(x) :Shear force at the member section at a distance *x* from the origin of the local xy coordinate system.
:Bending moment at the member section at a distance *x* from the origin of the local xy coordinate system.

The algorithms deal with the calculation and post-processing, according to the sequence shown in Fig. 8.

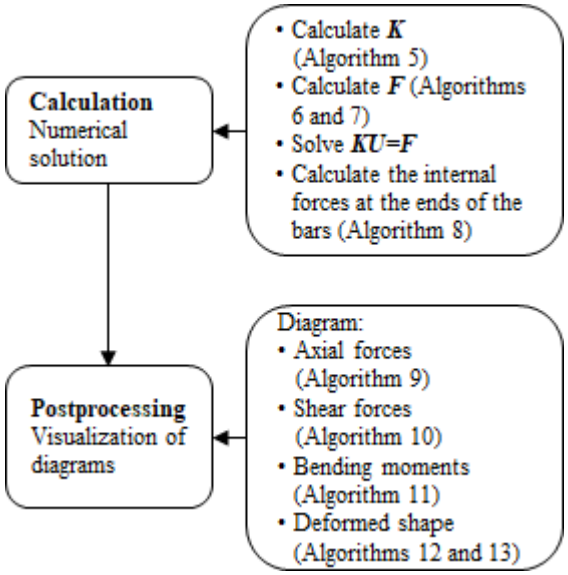

Figure 6: Diagram of the structural analysis procedure

3.1.1 Calculation of the global stiffness matrix of the model

The matrix associated with a directed edge of the member is defined by the submatrix 3x6 given by the upper half of the stiffness matrix 6x6 of the member, see [2] p. 277, as Eq. 1 illustrates. The terms *k1*, *k2*, *k3*, *k4*, *k5*, *k6* and *k7* of ***k*** are detailed in Algorithm 5.

$$\boldsymbol{k} = \overbrace{\begin{bmatrix} k_1 & k_2 & k_3 & -k_1 & -k_2 & k_3 \\ k_2 & k_4 & k_5 & -k_2 & -k_4 & k_5 \\ k_3 & k_5 & k_6 & -k_3 & -k_5 & k_7 \end{bmatrix}}^{s.i \quad s.j \quad s.k \quad e.i \quad e.j \quad e.k} \begin{matrix} s.i \\ s.j \\ s.k \end{matrix} \tag{1}$$

The complete matrix for a frame member can be obtained by applying Eq. 1 to both directed edges of the member.

The stiffness matrix ***K*** of the structure is given by the contribution of all the matrices ***k***, for which ***k*** has associated labels for rows and columns corresponding to the degrees of freedom *i, j, k* of the nodes of the member shown in Fig. 9. Such labels define the positions in ***K*** where the terms of ***k*** should be located.

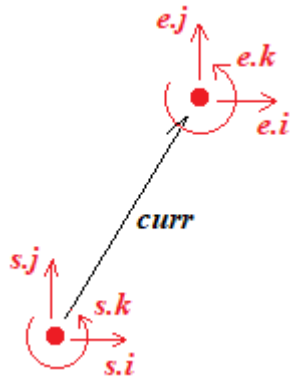

Figure 7: Degrees of freedom of a member according to the direction of a directed edge

Algorithm 5 calculates the stiffness matrix ***K*** for a frame ***G***. The list of directed edges of ***G*** is traversed. For each edge visited, the terms of ***k*** are added in ***K*** at the positions given by the degrees of freedom of the directed edge. The stiffness of elastic supports is also included.

Algorithm5 Calculates the stiffness matrix of a frame

```
COMPUTE-STIFFNESS-MATRIX-FRAME (G)
1   n = 3·G.V.size
2   let K be a new nxn matrix
    // traversing directed edges of G
3   for each edge curr∈G.E
4       s = curr.origin;
5       e = curr.opposite.origin
6       L = curr.L; cx = curr.cx; cy = curr.cy
7       E = curr.E ; A = curr.A; I = curr.I;
8       ki = E · I / L^3
9       ka = E · A / L
10      k1 = ka · cx · cx + 12 · ki · cy · cy
11      k2 = (ka - 12 · ki) · cx · cy
12      k3 = -6 · ki · L · cy
13      k4 = ka · cy · cy + 12 · ki · cx · cx
14      k5 = 6 · ki · L · cx
15      k6 = 4 · ki · L^2
16      k7 = 2 · ki · L^2
        // add first three columns of k
17      K[s.i][s.i] += k1;
18      K[s.i][s.j] += k2;
19      K[s.i][s.k] += k3;
20      K[s.j][s.i] += k2;
21      K[s.j][s.j] += k4;
22      K[s.j][s.k] += k5;
23      K[s.k][s.i] += k3;
24      K[s.k][s.j] += k5;
25      K[s.k][s.k] += k6;
        // add last three columns of k
26      K[s.i][e.i] += -k1;
27      K[s.i][e.j] += -k2;
28      K[s.i][e.k] += k3;
29      K[s.j][e.i] += -k2;
30      K[s.j][e.j] += -k4;
31      K[s.j][e.k] += k5;
32      K[s.k][e.i] += -k3;
33      K[s.k][e.j] += -k5;
34      K[s.k][e.k] += k7;
    // traverse vertices of G to add spring stiffness
35  for each vertex s∈G.V
36      K[s.i][s.i] += s.kx
37      K[s.j][s.j] += s.ky
38      K[s.k][s.k] += s.kθ
39  returnK
```

Similarly, stiffness matrices for directed edges of truss and beam members are defined.

For trusses:

$$\boldsymbol{k} = \frac{EA}{L} \overbrace{\begin{bmatrix} cx^2 & cxcy & -cx^2 & -cxcy \\ cxcy & cy^2 & -cxcy & cy^2 \end{bmatrix}}^{s.i \quad s.j \quad e.i \quad e.j} \begin{matrix} s.i \\ s.j \end{matrix} \tag{2}$$

For beams:

$$\boldsymbol{k} = \frac{EI}{L^3} \overbrace{\begin{bmatrix} 12cx^2 & 6Lcx & -12cx^2 & 6Lcx \\ 6Lcx & 4L^2 & -6Lcx & 2L^2 \end{bmatrix}}^{s.j \quad s.k \quad e.j \quad e.k} \begin{matrix} s.j \\ s.k \end{matrix} \tag{3}$$

In both cases, algorithms can be formulated following the

same approach of Algorithm 5.

3.1.2 Calculation of the fixed-end force vector for a model member

Loads on a member can act parallel to the member (axial load) or perpendicular (transverse load), as shown in Fig. 10.

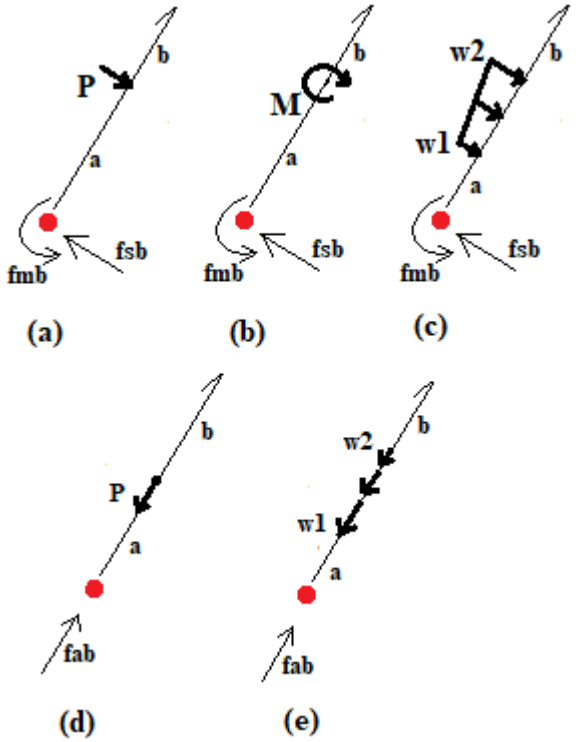

Figure 8 (a): Transverse point load; **b** Bending moment; **c** Transverse distributed trapezoidal load; **d** Axial point load; **e** Axial distributed trapezoidal load

As a result of the action of such loads on a frame member, forces are generated at the ends of the member. To calculate such end forces, there are expressions for each type of load acting on the member [2] p.656. Figure 10 shows the extreme forces *fab*, *fsb*, *fmb*, corresponding only to the origin node of the directed edge.

The data of each type of load: magnitude and location, are stored in the arrays ***P***, ***M***, ***N***, ***w*** and ***wN***, which are edge attributes. In each array, the data of each load is grouped sequentially considering the signs of the loads and their locations according to the direction of the directed edge.

a) For each transverse point load, two positions of ***P***:$\{P_1,a_1, \ldots, P_j,a_j\}$ are occupied, corresponding to its magnitude and location.
b) For each bending moment, two positions of ***M***:$\{M_1,a_1, \ldots, M_j,a_j\}$ are occupied, corresponding to its magnitude and location.
c) For each transverse trapezoidal load, four positions of ***w***:$\{w_1,a_1,w_2,b_2,\ldots,w_j,a_j,w_k,b_k\}$ are occupied, corresponding to their magnitudes *w*1 and *w*2 and the distances *a*, *b* from the ends respectively.
d) For each axial point load, two positions of ***N***:$\{N_1,a_1, \ldots, N_j,a_j\}$ are occupied, corresponding to its magnitude and location.
e) For each axial trapezoidal load, four positions of ***wN***:$\{w_1,a_1,w_2,b_2,\ldots, w_j,a_j,w_k,b_k\}$ are occupied, corresponding to their magnitudes *w*1 and *w*2 and the distances *a*, *b* from the ends respectively.

Algorithm 6 calculates the total magnitudes of the end forces *fab*, *fsb*, *fmb* at the origin of the edge *curr*, which are obtained by adding the end forces corresponding to all types of loads acting on a member.

Algorithm 6 Calculates the end forces at the origin of a directed edge

COMPUTE-END-FORCES-FRAME-BAR (*curr*)
1 L=*curr.L*
2 *fab = 0, fsb = 0, fmb = 0*
// transverse point load
3 **for** *j* = 1 **to** curr.*P.length* /2
4 P = *curr.P*[2 · *j*-1]
5 a = *curr.P*[2 · *j*]
6 b = L - a;
7 *fsb* += $P \cdot b^2 \cdot (3 \cdot a + b) / L^3$
8 *fmb* += $P \cdot a \cdot b^2 / L^2$
// bending moment
9 **for***j* = 1 **to***curr.M.length*/ 2
10 M = *curr.M*[2 · *j*-1]
11 a = *curr.M*[2 · *j*]
12 b = L - a;
13 *fsb* += $-6 \cdot M \cdot a \cdot b / L^3$
14 *fmb* += $M \cdot b \cdot (b - 2 \cdot a) / L^2$
// transverse trapezoidal load
15 **for***j* = 1 **to***curr.w.length*/ 4
16 *w1* = *curr.w*[4 · *j*-1]
17 a = *curr.w*[4 · *j*]
18 *w2* = *curr.w*[4 · *j* + 1]
19 b = *curr.w*[4 · *j* + 2]
20 d = L - a;
21 *fsb* += $w1 \cdot d^3 \cdot (7 \cdot L + 8 \cdot a - b \cdot (3 \cdot L + 2 \cdot a) / d \cdot (1 + b/d + b^2/d^2) + 2 \cdot b^4 / d^3) / (20 \cdot L^3) + w2 \cdot d^3 \cdot ((3 \cdot L + 2 \cdot a) \cdot (1 + b/d + b^2/d^2) - b^3/d^2 \cdot (2 + (15 \cdot L - 8 \cdot b)/d)) / (20 \cdot L^3)$
22 *fmb* += $w1 \cdot d^3 \cdot (3 \cdot (L + 4 \cdot a) - b \cdot (2 \cdot L + 3 \cdot a) / d \cdot (1 + b/d + b^2/d^2) + 3 \cdot b^4 / d^3) / (60 \cdot L^2) + w2 \cdot d^3 \cdot ((2 \cdot L + 3 \cdot a) \cdot (1 + b/d + b^2/d^2) - 3 \cdot b^3/d^2 \cdot (1 + (5 \cdot L - 4 \cdot b)/d)) / (60 \cdot L^2)$
// axial point load
23 **for***j* = 1 **to***curr.N.length*/ 2
24 P = *curr.N*[2 · *j*-1]
25 a = *curr.N*[2 · *j*]
26 b = L - a
27 *fab* += $P \cdot b / L$
// axial trapezoidal load
28 **for***j* = 1 **to***curr.wN.length*/ 4
29 *w1* = *curr.wN*[4 · *j*-1]
30 a = *curr.wN*[4 · *j*]
31 *w2* = *curr.wN*[4 · *j* + 1]
32 b = *curr.wN*[4 · *j* + 2]
33 d = L - a - b
34 c = L - a
35 *fab* += $w1/(2 \cdot L) \cdot c^2 + (w2 - w1) / (6 \cdot d \cdot L) \cdot (c^3 - b^3) - w2 \cdot b^2 / (2 \cdot L)$
// temperature variation
36 *fab* += *-curr.E·curr.A·curr.α ·(curr.Ts + curr.Ti) / 2*
37 *fmb* += *curr.E ·curr.I ·curr.α ·(curr.Ts - curr.Ti) /curr.h*
// error in member length
38 *fab* += *-curr.E · curr.A · curr.ΔL / curr.L*
39 **return***fe = {fab, fsb, fmb}*

For beams, Algorithm 6 is valid, but the action of axial forces should not be considered, that is, *fab*=0.

For trusses the procedure is similar to Algorithm 6but consider *fmb*=0 and *fsb* is calculated using the following expressions for load types *a*, *b* and *c* from Fig. 10.

a) For transverse point load

$$f_{sb} = \frac{P \cdot b}{L} \quad (4)$$

b) For bending moment

$$f_{sb} = -\frac{M}{L} \quad (5)$$

c) For transverse trapezoidal load

$$f_{sb} = \frac{(L-a-b)}{6 \cdot L} \cdot [w_1(2L-2a+b) + w_2(L-a+2b)] \quad (6)$$

3.1.3 Calculation of the vector of nodal loads of the model

Algorithm 7 returns the vector of nodal loads F of the model, for which the fixed-end forces at the origin node of each directed edge of a frame model G are calculated. Such forces are rotated to the global system and stored in F in the positions corresponding to the degrees of freedom of the origin node of the respective directed edge.

Algorithm 7 Calculate the vector of nodal forces for a frame model

```
Compute-Global-Vector-Frame (G)
1   n = 3·G.V.length
2   let F be a new vector of length n
3   for each edge curr∈G.E
4       s = curr.origin;
5       e = curr.opposite.origin
6       L = curr.L
7        cx = curr.cx
8        cy = curr.cy
9       fe=COMPUTE-END-FORCES-FRAME-BAR (curr)
        // rotar a global
10      F[s.i] += fe[0]· cx – fe[1] · cy
11      F[s.j] += fe[0]· cy + fe[1] · cx
12      F[s.k] += fe[2]
    // traverse vertices of G to add nodal loads
13  for each vertex s∈G.V
14      F[s.i] += s.Fx
15      F[s.j] += s.Fy
16      F[s.k] += s.Mz
17  returnF
```

For trusses and beams, the procedure is similar to Algorithm 7, but for trusses, bending moments are not considered and for beams, axial loads are not considered.

3.1.4 Calculation of the internal forces in the members of the model

At each end of the members the internal forces *Ni*, *Vi* and *Mi* are calculated. Figure 11 shows the internal forces at the origin end of a directed edge. The internal forces at the other end can be calculated from the opposite edge. The sign of the internal forces is interpreted according to the local coordinate system of the directed edge.

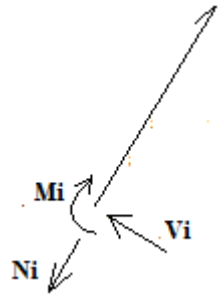

Figure 9: Internal forces at the origin end of a directed edge

The internal forces at the origin of a directed edge are calculated by applying the following matrix expression:

$$\begin{bmatrix} N_i \\ V_i \\ M_i \end{bmatrix} = \begin{bmatrix} \frac{EA}{L} & 0 & 0 & -\frac{EA}{L} & 0 & 0 \\ 0 & \frac{12EI}{L^3} & \frac{6EI}{L^2} & 0 & -\frac{12EI}{L^3} & \frac{6EI}{L^2} \\ 0 & \frac{6EI}{L^2} & \frac{4EI}{L} & 0 & -\frac{6EI}{L^2} & \frac{2EI}{L} \end{bmatrix} \cdot \begin{bmatrix} cx & cy & 0 & 0 & 0 & 0 \\ -cy & cx & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & cx & cy & 0 \\ 0 & 0 & 0 & -cy & cx & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ v_1 \\ \theta_1 \\ u_2 \\ v_2 \\ \theta_2 \end{bmatrix} - \begin{bmatrix} f_{ab} \\ f_{sb} \\ f_{mb} \end{bmatrix} \quad (7)$$

Algorithm 8 calculates the internal forces at the origin of each member, using simplified Equation 7. Such forces are stored in the attributes *Ni*, *Vi* and *Mi* of each directed edge.

Algorithm 8 Calculate the internal forces in the members of a frame

```
Compute-Internal-Forces-Vector-Frame (G)
1   for each edge curr∈G.E
2       s = curr.origin
3       e = curr.opposite.origin
4       L = curr.L; cx = curr.cx; cy = curr.cy
        // local matrix of a frame member
5       E = curr.E; A = curr.A; I = curr.I;
6       ki = E · I / L^3;
7       k1 = E · A / L
8       k2 = 12 · ki
9       k3 = 6 · ki · L
10      k4 = 4 · ki · L^2
11      k5 = 2 · ki · L^2
        // get displacements of the end nodes s and e
12      u1 = s.u;        v1 = s.v;        θ1 = s.θ;
13      u2 = e.u;        v2 = e.v;        θ2 = e.θ;
14      fe=COMPUTE-END-FORCES-FRAME-BAR (curr)
        // internal forces
15      curr.Ni=k1·(u1·cx+v1·cy)-k1·(u2·cx+v2·cy)-fe[0]
16      curr.Vi= k2·(-u1·cy+v1·cx)+k3·(θ1+θ2)
        -k2·(-u2·cy +v2·cx)-fe[1]
17      curr.Mi=k3·(-u1·cy+v1·cx)+k4·θ1
        -k3·(-u2·cy +v2·cx)+k5·θ2-fe[2]
```

For trusses and beams, the procedure is similar, but in each case use the corresponding matrices and consider *Mi=0* for trusses and *Ni=0* for beams.

3.1.5 Diagram of internal forces in the members of the

model

Once *Ni*, *Vi* and *Mi* have been calculated, expressions *N*(*x*), *V*(*x*) and *M*(*x*) are formulated for each member according to the local coordinate system defined for the member. For each type of load in Figure 12, there are expressions *N*(x), *V*(x) and *M*(x) formulated in terms of the *Macaulay* function whose definition is:

$$\langle x-a\rangle^n = \begin{cases} 0 & \forall \quad x < a \\ (x-a)^n & \forall \quad x \geq a \end{cases} \tag{8}$$

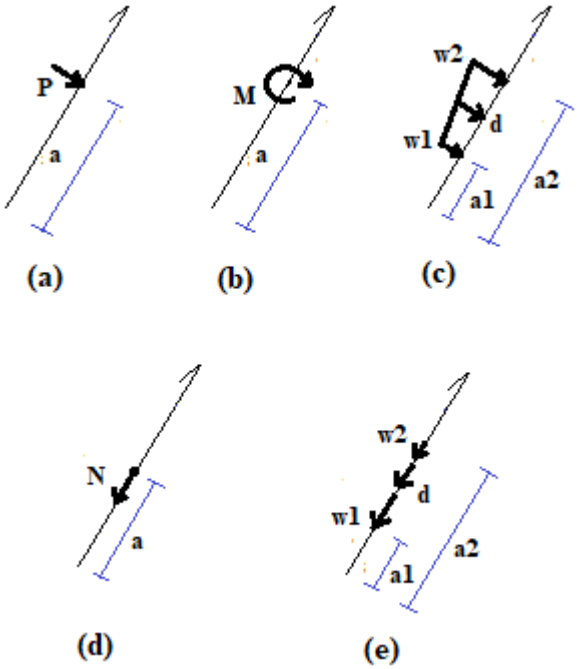

Figure 10: Types of load on a member

a) For transverse point load

$$M(x) = P\langle x-a\rangle^1 \tag{9}$$

b) For bending moment

$$M(x) = -M\langle x-a\rangle^0 \tag{10}$$

c) For transverse trapezoidal load

$$\begin{aligned} M(x) = w_2\frac{\langle x-a_1\rangle^2}{2} &- (w_2-w_1)\frac{\langle x-a_1\rangle^2}{2} \\ &+ (w_2-w_1)\frac{\langle x-a_1\rangle^3}{6\cdot d} \\ &- (w_2-w_1)\frac{\langle x-a_2\rangle^3}{6\cdot d} \\ &- w_2\frac{\langle x-a_2\rangle^2}{2} \end{aligned} \tag{11}$$

Where:

$a_2 = L - b$; $d = a_2 - a_1$

For load types *a*, *b* and *c*, the expressions for *V*(*x*) are obtained from $V(x) = dM(x)/dx$.

d) For axial point load

$$N(x) = -N\langle x-a\rangle^0 \tag{12}$$

e) For axial trapezoidal load

$$\begin{aligned} N(x) = -w_2\langle x-a_1\rangle^1 &+ (w_2-w_1)\langle x-a_1\rangle^1 \\ &- (w_2-w_1)\frac{\langle x-a_1\rangle^2}{2\cdot d} \\ &+ (w_2-w_1)\frac{\langle x-a_2\rangle^2}{2\cdot d} \\ &+ w_2\langle x-a_2\rangle^1 \end{aligned} \tag{13}$$

The values of *P*, *N*, *w*1, and *w*2 are substituted with signs according to the direction of the directed edge of the member. *M* is positive if it acts counterclockwise.

A member can be subjected to a combination of the loads in Fig. 12, for which the *Macaulay* expressions must be accumulated.

To evaluate *N*(*x*) the Macaulay expressions *N*(*x*) of the loads (*d*) and (*e*) in Fig. 12 are added as indicated in Eq. 14.

$$N(x) = -N_i + \sum_j N(x)_j \tag{14}$$

Algorithm 9 evaluates *N*(*x*) at a point *x* of a member, according to the direction of the directed edge *e*.

Algorithm 9 Calculate the axial force on a member

```
EVAL-AXIAL-FORCE-X (x, e)
1   Nx = -e.Ni
2   L = e.L
    // axial point load
3   forj = 1 toe.N.length/ 2
4       N = e.N[2 · j-1]
5       a = e.N[2 · j]
6       Nx = Nx - N · ⟨x - a⟩^0
    // axial trapezoidal load
7   forj = 1 toe.wN.length/ 4
8       w1 = e.wN[4 · j-1]
9       a1 = e.wN[4 · j]
10      w2 = e.wN[4 · j + 1]
11      b = e.wN[4 · j + 2]
12      a2 = L - b
13      d = a2 - a1
14      Nx = Nx - w2·⟨x - a1⟩^1 + (w2-w1)·⟨x - a1⟩^1
        - (w2 - w1)·⟨x - a1⟩^2 / (2 · d)
        + (w2 - w1) · ⟨x - a2⟩^2 / (2 · d) + w2 · ⟨x - a2⟩^1
15  returnNx
```

To evaluate *V*(*x*), the Macaulay expressions *V*(*x*) of loads (*a*) and (*c*) in Fig. 12 are added as indicated in Eq. 15.

$$V(x) = V_i + \sum_j V(x)_j \tag{15}$$

Algorithm 10 evaluates *V*(*x*) at a point *x* of a member, according to the direction of the directed edge *e*.

Algorithm 10 Calculate the shear force on a member

```
EVAL-SHEAR-FORCE-X (x, e)
1   Vx = e.Vi
2   L = e.L
    // transverse point load
3   forj = 1 toe.P.length/ 2
4       P = e.P[2 · j-1]
5       a = e.P[2 · j]
6       Vx = Vx + P ·⟨x - a⟩^0
    // transverse trapezoidal load
7   forj = 1 toe.w.length/ 4
8       w1 = e.w[4 · j-1]
9       a1 = e.w[4 · j]
10      w2 = e.w[4 · j + 1]
11      b = e.w[4 · j + 2]
12      a2 = L - b
13      d = a2 - a1
14      Vx = Vx + w2·⟨x - a1⟩^1 - (w2 - w1)·⟨x - a1⟩^1
```

+ $(w2 - w1)\cdot\langle x - a1\rangle^2 / (2\cdot d)$
- $(w2 - w1) \cdot \langle x - a2\rangle^2 / (2 \cdot d) - w2 \cdot \langle x - a2\rangle^1$
15 **return** Vx

To evaluate $M(x)$ the Macaulay expressions $M(x)$ of the loads (*a*), (*b*) and (*c*) of Fig. 12 are added as indicated in Eq. 16.

$$M(x) = V_i \cdot x - M_i + \sum_j M(x)_j \qquad (16)$$

Algorithm 11 evaluates $M(x)$ at a point x of a member, according to the direction of the directed edge e.

Algorithm 11 Calculates the bending moment on a member

EVAL-BENDING-MOMENT-X (x, e)
1 $Mx = e.Vi \cdot x - e.Mi$
2 $L = e.L$
// transverse point load
3 **for** j = 1 **to** $e.P.length/ 2$
4 $P = e.P[2 \cdot j\text{-}1]$
5 $a = e.P[2 \cdot j]$
6 $Mx = Mx + P \cdot \langle x - a\rangle^1$
// bending moment
7 **for** j = 1 **to** $e.M.length/ 2$
8 $M = e.M[2 \cdot j\text{-}1]$
9 $a = e.M[2 \cdot j]$
10 $Mx = Mx - M \cdot \langle x - a\rangle^0$
// transverse trapezoidal load
11 **for** j = 1 **to** $e.w.length/ 4$
12 $w1 = e.w[4 \cdot j\text{-}1]$
13 $a1 = e.w[4 \cdot j]$
14 $w2 = e.w[4 \cdot j + 1]$
15 $b = e.w[4 \cdot j + 2]$
16 $a2 = L - b$
17 $d = a2 - a1$
18 $Mx = Mx + w2\cdot\langle x - a1\rangle^2 /2 - (w2 - w1)\cdot\langle x - a1\rangle^2 / 2 + (w2 - w1)\cdot\langle x - a1\rangle^3 / (6\cdot d) - (w2 - w1) \cdot \langle x - a2\rangle^3 / (6 \cdot d) - w2 \cdot \langle x - a2\rangle^2 / 2$
19 **return** Mx

The signs for $N(x)$, $V(x)$ and $M(x)$, are interpreted according to the direction of the directed edge.

The points (x, $N(x)$) or (x, $V(x)$) are drawn according to the local coordinate systems of Fig. 13.

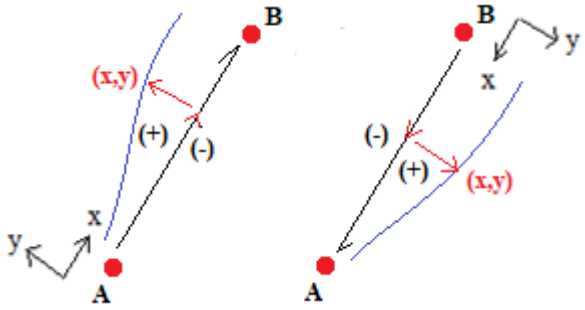

Figure 11: Sign convention for diagrams of axial and shear forces according to the direction of the directed edge.

If $M(x)$ is positive the points (x, $M(x)$) are drawn on the negative y-axis as shown in Fig. 14, so if the member is horizontal, the positive moments are drawn downwards.

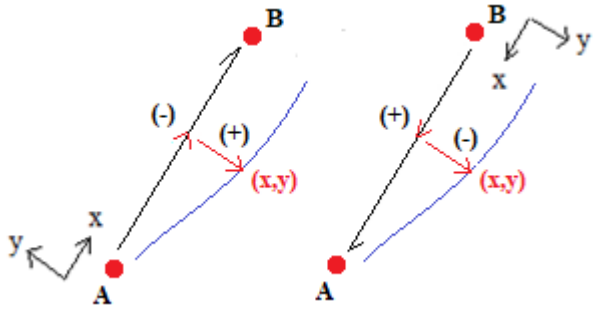

Figure 12: Sign convention for bending moment diagrams

3.1.6 Deformed shape of model members

To draw the deformed shape of a member, its axial deformation $u(x)$ and deflection $v(x)$ are drawn as ordered pairs (x+$u(x)$, $v(x)$) as illustrated in Fig. 15.

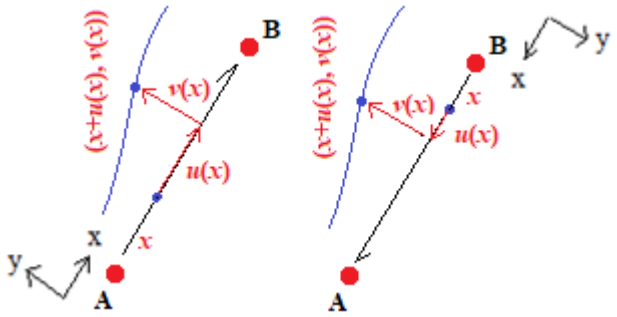

Figure 13: Deformed shape of a model member

Previously, the local displacements $\bar{u}_i, \bar{v}_i$ at the origin of each member are required, for which the displacements u_i, v_i of the origin nodes of the members are transformed by applying Eq. 17.

$$\begin{bmatrix}\bar{u}_i \\ \bar{v}_i\end{bmatrix} = \begin{bmatrix} cx & cy \\ -cy & cx\end{bmatrix}\begin{bmatrix}u_i \\ v_i\end{bmatrix} \qquad (17)$$

The axial deformation of a member is obtained from Eq. 18.

$$\frac{du}{dx} = \frac{N(x)}{EA}, \qquad u(0) = \bar{u}_i \qquad (18)$$

Whose solution is:

$$u(x) = \bar{u}_i + \frac{1}{EA}\int N(x)dx \qquad (19)$$

To the previous solution is added the deformation due to temperature changes and the deformation due to error in the length of the member.

$$u(x) = \bar{u}_i + \frac{1}{EA}\int_{\cdot x} N(x)dx + \frac{\alpha \cdot (T_t + T_b)}{2} \cdot x + \frac{\Delta L}{L} \qquad (20)$$

Algorithm 12 evaluates the axial deformation at a point x on a member, according to the directed edge direction e.

Algorithm 12 Calculates the axial deformation in a frame member

Eval-Ux (x, e)
1 $Ux = e.ui - e.Ni \cdot x /(e.E \cdot e.A) + e.\alpha \cdot(e.Tt + e.Tb)\cdot x/2 + e.\Delta L \cdot x / e.L$
// axial point load
2 **for** j = 1 **to** $e.N.length/ 2$
3 $P = e.N[2 \cdot j\text{-}1]$
4 $a = e.N[2 \cdot j]$

```
5       Ux = Ux + (-P · ⟨x - a⟩^1) / (e.E · e.A)
    // axial trapezoidal load
6   forj = 1 toe.wN.length/ 4
7       w1 = e.wN[4 · j-1]
8       a1 = e.wN[4 · j]
9       w2 = e.wN[4 · j + 1]
10      b = e.wN[4 · j + 2]
11      a2 = e.L - b
12      d = a2 - a1
13      Ux = Ux +(-w2·⟨x - a1⟩^2 /2 + (w2-w1)·⟨x - a1⟩^2/2-
    (w2-w1)·⟨x - a1⟩^3 / (6·d) + (w2 - w1)·⟨x - a2⟩^3 / (6·d)
        + w2 · ⟨x - a2⟩^2 / 2) / (e.E · e.A)
14  returnUx
```

For beams $u(x)=0$.

For trusses, Algorithm 12 is valid but the temperature must be considered uniform, that is, $T_t=T_b$.

The deflection of a member is obtained from Eq. 21.

$$\frac{d^2v}{dx^2} = \frac{M(x)}{EI}, \qquad \left.\frac{dv}{dx}\right|_{x=0} = \theta_i \quad v(0) = \bar{v}_i \tag{21}$$

Whose solution is

$$v(x) = \bar{v}_i + \theta_i x + \frac{1}{EI}\int\int M(x)dxdx \tag{22}$$

To the previous solution is added the deformation due to temperature changes.

$$v(x) = \bar{v}_i + \theta_i x + \frac{1}{EI}\int\int M(x)dxdx - \frac{\alpha\cdot(T_t - T_b)}{2\cdot h}\cdot x^2 \tag{23}$$

Algorithm 13 evaluates the deflection at a point x on a member, according to the direction of the directed edge e.

Algorithm 13 Calculates the deflection in a frame member

```
EVAL-YX (x, e)
1   Yx = e.vi + e.θi·x + (e.Vi·x^3/6 - e.Mi·x^2/2)/(e.E·e.I) -
    e.α·(e.Tt - e.Tb)·x^2/ (2·e.h)
    // transverse point load
2   forj = 1 toe.P.length/ 2
3       P = e.P[2 · j-1]
4       a = e.P[2 · j]
5       Yx = Yx + (P · ⟨x - a⟩^3 / 6) / (e.E · e.I)
    // bending moment
6   forj = 1 toe.M.length/ 2
7       M = e.M[2 · j-1]
8       a = e.M[2 · j]
9       Yx = Yx + (-M · ⟨x - a⟩^2 / 2) / (e.E · e.I)
    // transverse trapezoidal load
10  forj = 1 toe.w.length/ 4
11      w1 = e.w[4 · j-1]
12      a1 = e.w[4 · j]
13      w2 = e.w[4 · j + 1]
14      b = e.w[4 · j + 2]
15      a2 = e.L - b
16      d = a2 - a1
17      Yx = Yx + (w2·⟨x - a1⟩^4/ 24 - (w2- w1)·⟨x - a1⟩^4 /24 +
    (w2-w1)·⟨x - a1⟩^5 / (120·d) - (w2-w1)·⟨x - a2⟩^5 / (120·d)
        - w2·⟨x - a2⟩^4 / 24) / (e.E · e.I)
18  returnYx
```

To draw the diagrams, all members are assumed to be horizontal with origin at (0,0) as shown in Fig. 16a. The points $(x+u(x),v(x))$, $(x, N(x))$, $(x, V(x))$, $(x, M(x))$, must be rotated around the origin (0,0) an angle θ according to the direction of the directed edge and translate such rotated points, a distance (x_1, y_1), corresponding to the origin coordinate of the member. In this way the diagram is as shown in Fig. 16b.

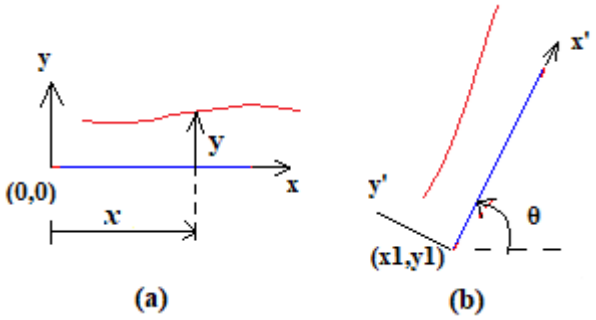

Figure 14: Rotation and translation of diagrams

The transformation matrix that rotates a point (x, y) around the origin and translates it a distance (x_1, y_1) is:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\operatorname{sen}\theta & x_1 \\ \operatorname{sen}\theta & \cos\theta & y_1 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{24}$$

4. Program developed

From the programming of the TAD Grafo and the algorithms, GBSA (Graph-Based Structural Analysis) is obtained, whose graphical interface allows interaction in the three stages of structural analysis, as illustrated in Fig. 17.

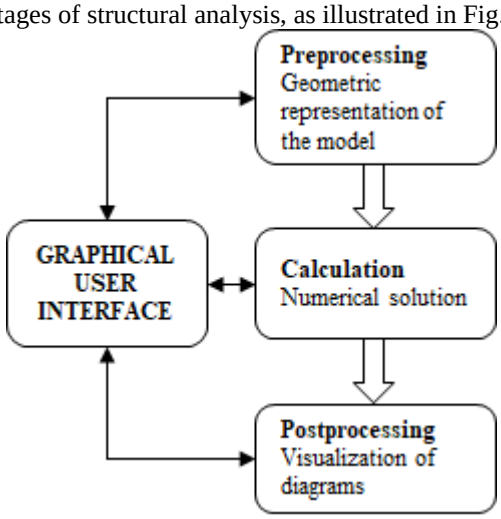

Figure 15: Interaction with the graphical user interface

The graphical user interface includes a canvas where the user can draw a model using the mouse and view the diagrams associated with the model. It also includes a context menu to perform each stage of the structural analysis.

Figure 18 shows the options for inserting and deleting members and nodes are available for preprocessing. The user can assign data to groups of selected nodes or members.

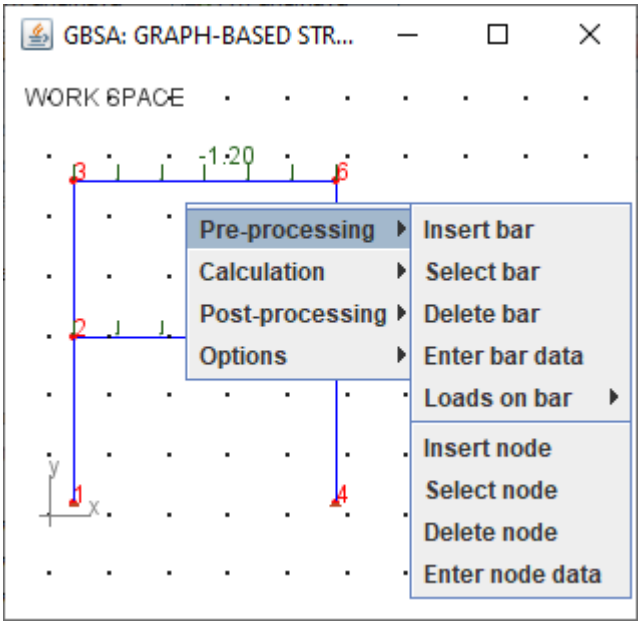

Figure 16: Options for preprocessing

For the calculation, the Run option is available, as shown in Fig. 19, which integrates the set of algorithms corresponding to this stage.

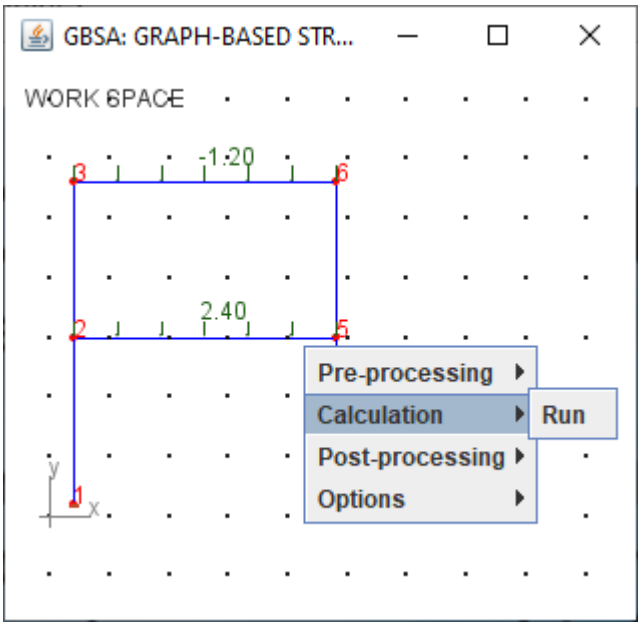

Figure 17: Option for calculation

For post-processing, Figure 20 shows the options to view the diagrams. It is possible to select the members on which to display the diagrams.

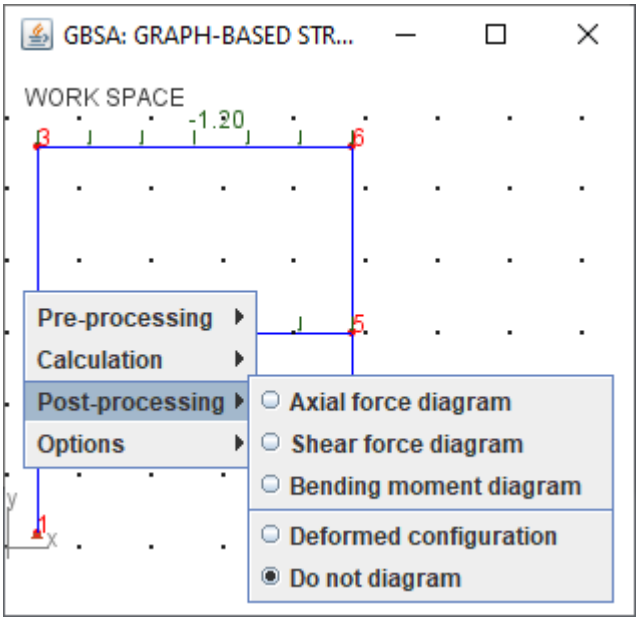

Figure 18: Options for post-processing

5. Results and Discussion

Using *GBSA*, a truss, a beam and a frame have been analyzed, whose results are compared with those of the SAP2000 program. The data are taken from Table 1.

Table 1: Data collected for member attributes

Attribute	Truss	Beam	Frame
A	0.005 m2	----------	0.075 m2
I	----------	0.0005625 m4	0.0005625 m4
ΔL	-0.01 m	----------	----------
h	----------	----------	0.30 m
E	2e8 kN/m2	2e8 kN/m2	2e8 kN/m2
Tt	----------	----------	+24 °C
Tb	----------	----------	+15 °C
α	----------	----------	1.2e-5 /°C

As a result of the analysis of each model, the nodal displacements, the deformed shape of the models and the internal force diagrams are presented. To visualize the internal force diagrams, only some members of the model have been selected, in order to avoid overloading the diagrams.

5.1 Analysis of a truss

Figure 21 shows a truss loaded at node 4, point loads at member 6-7, and trapezoidal distributed loading at member 1-2. Member 3-5 is considered to have been made 0.01 shorter. The truss was drawn on a grid with 1m spacing.

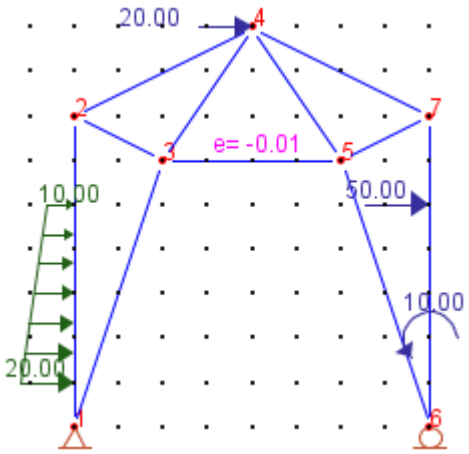

Figure 19: Truss model subjected to nodal load, point loads, trapezoidal distributed load and member length error

Table 2 presents the nodal displacements from the truss analysis.

Table 2: Comparison of nodal displacements with SAP2000

Node	Direction	DISPLACEMENTS (m)		
		GBSA	SAP2000	% Error
2	x	0.00598211	0.005982	0.000000
	y	-0.00145667	-0.001457	0.000000
3	x	0.00706941	0.007069	0.000000
	y	-0.00037879	-0.000379	0.000000
4	x	0.00206919	0.002069	0.000000
	y	0.00390949	0.003909	0.000000
5	x	-0.00225948	-0.002259	0.000000
	y	0.00113656	0.001137	0.000000
6	x	-0.00467529	-0.004675	0.000000
7	x	-0.00088487	-0.000885	0.000000
	y	-0.00084333	-0.000843	0.000000

Table 2 shows that when rounding the GBSA values to 6 decimal places, an error of 0.000000% is obtained for the displacements.

The deformed shape of the truss is consistent with the nodal displacements obtained as shown in Fig. 22.

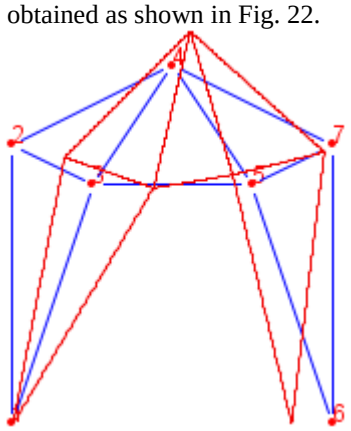

Figure 20: Deformed shape of the truss

Figure 23 shows the axial force diagram for members 1-3, 6-5, 6-7 and 4-7 according to the direction shown on such members. Members 6-7 and 4-7 are in compression (negative sign), while members 1-3 and 6-5 are in tension (positive sign).

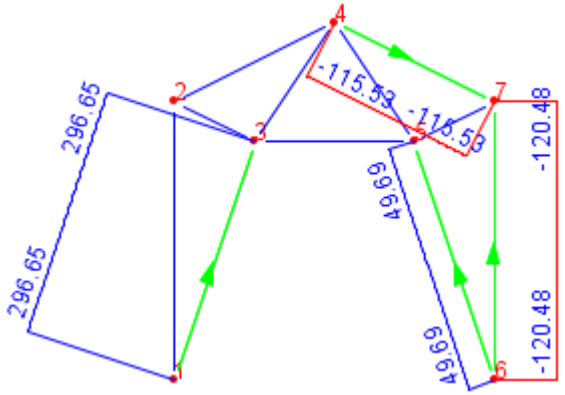

Figure 21: Axial force diagram for the truss

Figure 24 shows the shear force diagram for members 1-2 and 6-7. In all other members, the shear force is zero.

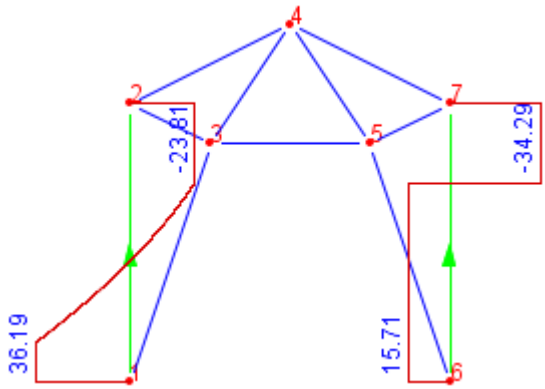

Figure 22: Shear force diagram for the truss

Figure 25 shows the bending moment diagram for members 1-2 and 6-7. In all other members the bending moment is zero.

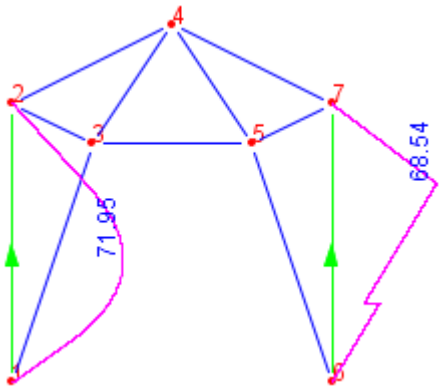

Figure 23: Bending moment diagram for the truss

5.2 Analysis of a beam

The beam in Fig. 26 is subjected to a trapezoidal distributed load in member 1-2 and a point load in member 2-3. In addition, node 2 is on an elastic support. The beam was drawn on a grid with 1m spacing.

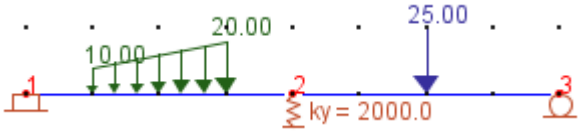

Figure 24: Beam model subjected to trapezoidal distributed load, point load and elastic support

Table 3 presents the nodal displacements for nodes 2 and 3 of the beam.

Table 3: Comparison of nodal displacements with SAP2000

Node	Direction	DISPLACEMENTS (m)		
		GBSA	SAP2000	% Error
2	y	-0.00129600	-0.001296	0.000000
	z	-0.00016066	-0.000161	0.000000
3	z	0.00067744	0.000677	0.000000

Table 3 shows that when rounding the GBSA values to 6 decimal places, an error of 0.000000% is obtained for the displacements.

Figure 27 shows the deformed shape of the beam, which is consistent with the displacements obtained for nodes 2 and 3.

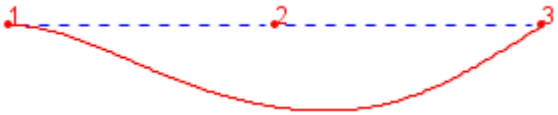

Figure 25: Deformed shape of the beam

Figure 28 shows the shear force diagram for the members of the beam. Positive shear forces are drawn upwards.

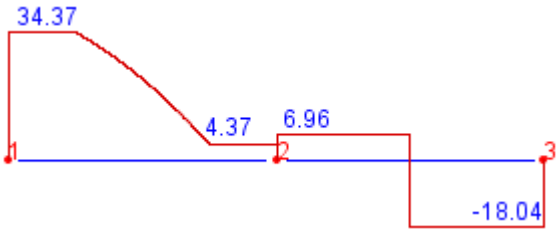

Figure 26: Shear force diagram for the beam

Figure 29 shows the bending moment diagram for the beam members. Positive bending moments are drawn downward.

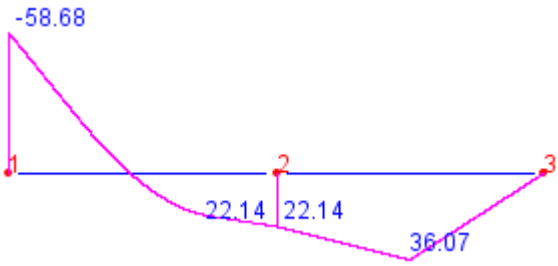

Figure 27: Bending moment diagram for the beam

5.3 Analysis of a frame

Figure 30 presents a frame subjected to trapezoidal distributed loading at member 3-4 and loads at nodes 2 and 3. Member 2-5 is subjected to temperature variation and point loading. In addition, node 6 is subject to a settlement of 0.01 m. The frame was drawn on a grid with 1m spacing.

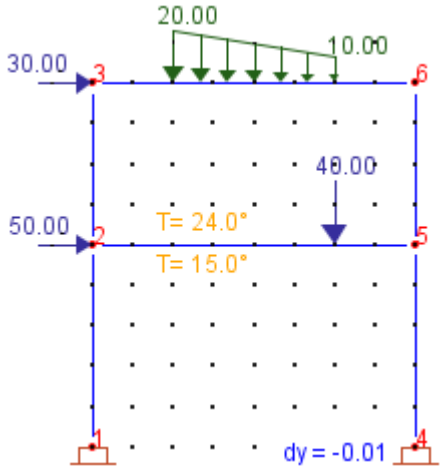

Figure 28: Frame model subjected to trapezoidal distributed load, nodal loads, point load, temperature variation and settlement

Table 4 presents the nodal displacements from the frame analysis.

Table 4: Comparison of nodal displacements with SAP2000

Node	Direction	DISPLACEMENTS (m)		
		GBSA	SAP2000	% Error
2	x	0.00776380	0.007764	0.000000
	y	-0.00000430	-0.000004	0.000000
	z	-0.00183260	-0.001833	0.000000
3	x	0.01686238	0.016862	0.000000
	y	-0.00001025	-0.000010	0.000000
	z	-0.00246757	-0.002468	0.000000
5	x	0.00962186	0.009622	0.000000
	y	-0.01002904	-0.010029	0.000000
	z	-0.00215871	-0.002159	0.000000
6	x	0.01685199	0.016852	0.000000
	y	-0.01003909	-0.010039	0.000000
	z	-0.00099437	-0.000994	0.000000

Table 4 shows that when rounding the GBSA values to 6 decimal places, an error of 0.000000% is obtained for the displacements.

Figure 31 shows the deformed shape of the frame, which is consistent with the displacements obtained for nodes 2, 3,5 and 6 of the frame.

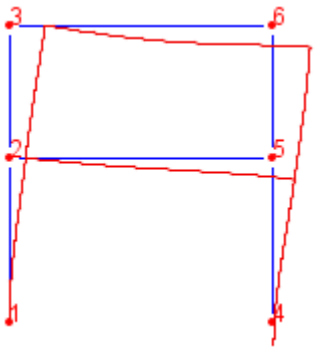

Figure 29: Deformed shape of the frame

Figure 32 shows the axial force diagram for members 1-2, 4-5 and 3-6 of the frame.

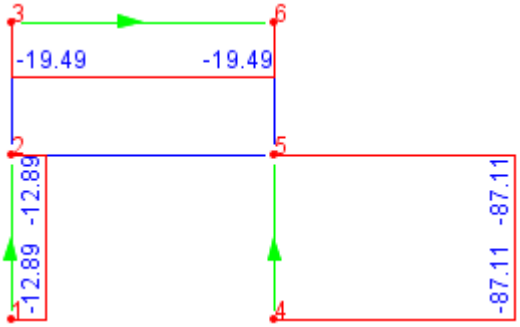

Figure 30: Axial force diagram for the frame

Figure 33 shows the shear force diagram for members 1-2, 4-5 and 3-6 of the frame.

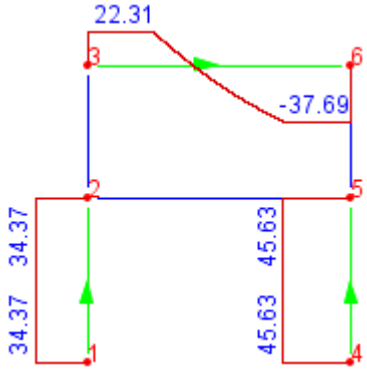

Figure 31: Shear force diagram for the frame

Figure 34 shows the bending moment diagram for members 1-2, 4-5 and 3-6 of the frame.

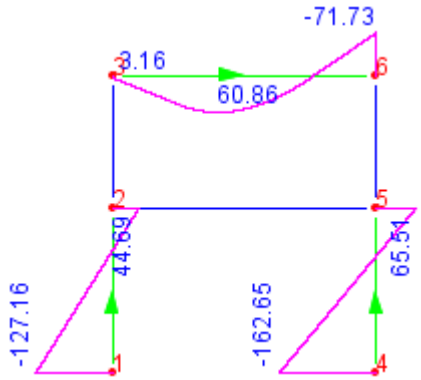

Figure 32: Bending moment diagram for the frame

6. Discussions

The analysis of three structures with different geometries and subjected to different types of loads has been presented, obtaining in all cases that the results coincide with those of the SAP2000 software.

For each model, a combination of the following topics has been included: thermal effects, fabrication errors, elastic supports, trapezoidal distributed loads, point loads, nodal loads, and support movements.

Data was quickly assigned to models by selecting their nodes and members.

In order to visualize the internal force diagrams, the user can select a member or a set of members with their respective directions.

7. Conclusions

The representation of models by directed edges is useful, since it allows greater interactivity with the user, by allowing him to choose the direction of a member to view the diagrams and assign loads and temperatures.

The Graph ADT facilitated the implementation of a canvas where the user can draw the models using the mouse, in this way the user does not need to type one by one coordinates and connectivity between nodes.

The operations of the Graph ADT allows a model to be built and modified dynamically, that is, the number of members and nodes of the model is not restricted.

The algorithms presented can analyze structures subjected to topics such as: thermal effects, fabrication errors, elastic supports, trapezoidal distributed loads, point loads, nodal loads, and support movements.

The presented algorithms produce correct results, therefore, they are useful for the development of software for the analysis of trusses, beams and frames.

References

[1] Da Fonseca, Z. (2016) Análisis matricial de estructuras reticulares, Primera. Fondo Editorial Biblioteca, Maracaibo
[2] Kassimali, A. (2012) Matrix analysis of structures, 2nd ed. Cengage Learning, Carbondale
[3] Kassimali, A. (2015) Análisis estructural, 5th ed. Cengage Learning, Carbondale
[4] Cervera, M., Blanco, E. (2014) Mecánica de estructuras. Centro Internacional de Métodos Numéricos en Ingeniería, Barcelona
[5] Barhate, PG., Ladhane, KB. (2016) Development of structural analysis program for truss structure using MATLAB. Int J Technol Res Eng 3:2460–2465
[6] François, S., Schevenels, M., Dooms, D., Jansen, M., Wambacq, J., Lombaert, G., Degrande, G., De Roeck, G.(2021) Stabil: An educational Matlab toolbox for static and dynamic structural analysis. Comput Appl Eng Educ 29:1372–1389. https://doi.org/10.1002/cae.22391
[7] Martínez-Pañeda, E.(2016) MATLAB : Una herramienta para la didáctica del Método de los Elementos Finitos. Rev Iberoam Educ Matemática 242–268
[8] Neiva, PHG., Vieira, GD., Batista, S., Dos Prazeres, PGC. (2018) Desenvolvimento de programa didático para análise de vigas pelo método dos elementos finitos. In: XLVI COBENGE
[9] Villagómez, M., Calderón, RR., López, L., Arbesú, RS.(2015) SAE Software de Análisis Estructural. In: Revista del Congreso Internacional de Innovación Educativa. pp 426–430
[10] Nogueira, LGO., Bezerra, EMF. (2017) Ferramenta em ambiente excel para análise estrutural de treliças espaciais pelo método dos elementos finitos. Proc XXXVIII Iber Lat Am Congr Comput Methods Eng. https://doi.org/10.20906/cps/cilamce2017-1280

[11] Bakošová, A., Krmela, J., Handrik, M. (2020) Computing of truss structure using MATLAB. Manuf Technol 20:279–285. https://doi.org/10.21062/mft.2020.059
[12] Neves, NS., Pinheiro, VP., Camargo, RS.(2019) Desenvolvimento de uma interface gráfica educacional para ensino de elementos finitos aplicado a problemas de viga sob base elástica. In: X Encontro Científico de Física Aplicada
[13] Zotkin, SP., Blokhina, NS., Zotkina, IA.(2015) About development and verification of software for finite element analysis of beam systems. Procedia Eng 111:902–906. https://doi.org/10.1016/j.proeng.2015.07.045
[14] Barrantes, FD., Hernández, ÁG.(2020) Modelo computacional para el análisis matricial de estructuras reticulares. Universidad Peruana de Ciencias Aplicadas
[15] Barrera, JA.(2018) Desarrollo de software para el análisis de casos indeterminados y específicos de vigas, pórticos y armaduras denominado ECHELON. Universidad Distrital Francisco José de Caldas
[16] Barreto Bezerra, AA., Sousa Da Silva, LM., Lima, AW.(2018) Desenvolvimento de um programa computacional para análise de vigas Euler-Bernoulli utilizando a linguagem Pytho. Rev Principia - Divulg Científica e Tecnológica do IFPB 1:54. https://doi.org/10.18265/1517-03062015v1n38p54-60
[17] Htwe, T., Khaing, SY.(2014) Analysis of Beam Structure with Matlab Software. Int J Sci Eng Technol Res 03:2064–2069
[18] Patil, IS., Annigeri, SA.(2016) Introduction to PSA as a Free Structural Analysis Software. Bonfring Int J Man Mach Interface 4:116–120. https://doi.org/10.9756/bijmmi.8167
[19] Pallares, M., Calderón, WR., García,Deg.(2020) An educational computer program for matrix analysis of plane trusses in civil engineering. ARPN J Eng Appl Sci 15:570–576
[20] Chen, X (2020) Programming for solving plane rigid frame based on MATLAB. MATEC Web Conf 319:09003. https://doi.org/10.1051/matecconf/202031909003
[21] Pamnani, G., Rajput, DS., Tiwari, N., Gajendra, A. (2014) Beam Analysis in Matlab Specify design characteristic. Adv Phys Lett 1:27–36
[22] Godoi, R., Vanalli, L., Da Silva, S.(2017) Análise de estruturas utilizando o software Excel através do método da rigidez direta. In: 26 Encontro Anual de Iniciação Científica
[23] De Oliveira, CJ., Steffen, LO., Vogel, GM., Nunes, RB., Dos Santos, GM.(2019) Aplicação do software Excel como ferramenta de ensino para resolução de treliças planas utilizando elementos finitos de barras. Rev Tecnol 40:1–13. https://doi.org/10.5020/23180730.2019.9903
[24] Chapra, SC., Canale, RP.(2007) Métodos numéricos para ingenieros, 5th ed. McGraw-Hill, México, D.F.