XL CILAMCE
IBERO-LATIN AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING
NOVEMBER 11-14, 2019
Praiamar Natal Hotel & Convention
Natal, RN-BRAZIL

GEOMETRICALLY NONLINEAR ANALYSIS OF PLANE TRUSSES AND FRAMES USING AN AUTOMATIC DERIVATION ALGORITHM

Danilo Menezes Santos
Jorge Carvalho Costa
dmsantosse@gmail.com
jorgecostase@gmail.com
Universidade Federal de Sergipe
Av Marechal Rondom S/N, São Cristóvão, Sergipe, Brasil

Abstract. Structural analysis requires the derivation of complex functions usually of many variables, especially when nonlinearities are considered. From a computational standpoint, to derivate can be a great challenge due to a series of limitations inbuilt in the usual derivation techniques. Surmising both these factors, this work presents the development of an automatic differentiation (AD) algorithm in Python 3.x and its application to structural analysis, aiming at preventing possible compatibility and truncation errors inherent to other derivation processes. The software implemented both the forward and reverse modes of differentiation and is able to work with algorithms written as string or def Python classes. Later, the package was used to obtain the local stiffness (tangent) matrices for plane trusses and frames in linear and geometrically nonlinear analysis, derived through energy methods. The results were compared with benchmarks found in the literature. Excellent results were found for the stiffness matrices, nodal displacements and internal forces, supporting that this technique can be used for solid mechanics problem with ease.

Keywords: Automatic differentiation; nonlinear structural mechanics; plane structures

1 Introduction

Between the end of the 20th century and the beginning of the 21st, new technology linked to Civil Construction impacted and profoundly modified the practice on this sector. From a Structural Engineering standpoint, the need was created for engineers to evolve and try to apprehend and simulate through more profound studies and thorough analyzes the behavior of such innovations, that allowed for slenderer, more economical and complex structures.

From these studies, two types of nonlinear analysis were developed: physically nonlinear, where materials with a stress-strain relationship other then Hooke's law are considered (elastoplastic, plastic and others) and geometrically nonlinear, where the structural equilibriums is imposed in a deformed configuration so that to capture higher order effects. Nonlinear analyses are highly expensive both on theoretical development and computational effort. According to Cook, Malkus e Plesha (1989 *apud* Sales [1]), they are more difficult to formulate and their computational solution may cost from 10 to 100 times as much as a linear approximation with the same number of degrees of freedom, especially due to: the complexity of the adopted mathematical model, the need to solve nonlinear system of equations and even the use of computational numerical methods (as numerical integration and derivation).

These needs require interesting alternatives that can optimize repetitive processes used in nonlinear analyses, such as derivations. For example, using energy methods, equilibrium can be imposed as the point of minimal potential energy of the structural system, and the search for this minimum requires the evaluation of the derivatives of energy functions. Another point is relative to the equilibrium itself, may it be imposed directly or by energy minimization, that renders a series of nonlinear equations on the displacements and the techniques usually used for the solution of this system require the derivatives of the system equations.

This work aims at presenting an alternative technique for geometrically nonlinear analyses of beams and rods, using an automatic differentiation algorithm (DAALGPY) for imposing equilibrium and obtaining the local and global tangent stiffness matrices for trusses and frames.

2 Derivation techniques and automatic derivation

The most common methods for determining derivatives are: by hand, numerically, symbolically and through algorithms, the latest known as algorithmic or automatic derivation.

Hand differentiation consists in developing the necessary algebraic expressions making use of differential calculus to obtain mathematical formulae for the derivatives. It is hard to develop for long and complex expressions and susceptible to errors on calculations or implementation. Numerical methods approximate de derivative value around a point by finite differences. This allows for simpler functions to be used but accounts for truncation errors that reduce its efficiency (Haftka and Gurdal *apud* Barthelemy and Hall [2]).

Symbolic derivation consists on the use of specific mathematical analysis software the encode calculus rules to automatize the algebraic operations used to derivate. The results are precise and have no truncation error, and depending on the capabilities of the used software, the expressions can be simplified and compact. On the other hand, it presents an elevated computational cost, demanding long processing times for large scale problems. Another disadvantage appears when the platform used for symbolic derivations is not the same as for algorithm development. In these situations, the expressions must be translated into code, a step also susceptible to implementation errors.

Lastly there is Algorithmic Derivation (AD). It consists in applying derivation rules directly in a computational code. The original function is programmed and the AD algorithm is applied, rendering a new function with the derivatives. These are determined by interpreting the original function by a computer graph such as Fig. 1, dividing it into successive composite functions and latter applying the chain rule of derivations into the nodes. The strategies to analyze the graphs are divided into two methods: forward mode and reverse mode. On the first, the whole computational graph is analyzed through linear combinations between the node derivatives, and the second walks the graph in reverse order, using only the nodes where the variable being analyzed exists.

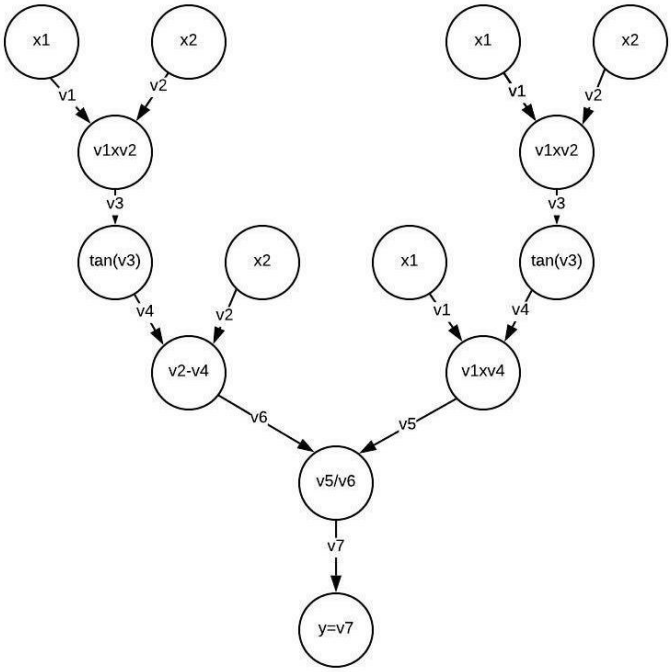

Figure 1 – Graph representation of a function of two variables

Algorithmic Differentiation, in comparison with other techniques, has the advantage to generate results with analytical precision, with no truncation errors and with computational costs similar or even lower than symbolic derivation. Due to these characteristics, AD have been used by the academic community, with applications on diverse areas as machine learning (Šrajer, Kukelova and Fitzgibbon [3]), topology optimization (Linn [4]), among others.

3 DAALGPY

The package DAALGPY (*Python Algoritmo de Diferenciação Automática*) is an application of computational differentiation algorithms in Python 3x developed by the first author using both forward and reverse modes for the determination of gradients. The algorithm accepts as inputs both a single function of a system of functions, as long as each equation is a string or an algorithmic function. The semantics for the implemented functions are the same as the native module *math*, so it is possible to use:

- Binary operations: addition $(+)$, subtraction $(-)$, division $(/)$ and multiplication $(*)$;
- Trigonometric functions: sine $(sin(\cdot))$, cossine $(cos(\cdot))$, tangent $(tan\,(\cdot))$, secant

($sec\ (\cdot)$), cossecant ($cossec\ (\cdot)$) e cotangent ($cotg(\cdot)$);

- Elementary functions: exponencial ($exp\ (\cdot)$), power ($**$), natural logaritm ($log(\cdot)$) and square roots ($sqrt\ (\cdot)$).

```
 7 import DAALGPY.Gradient as gr
 8
 9 X = 'x1 * (x2 ** 2) - log(x3 ** 2)'
10
11 def Y(x1,x2,x3):
12     return x2 **2 * x1 ** 2
13
14 def Z(x1,x2,x3):
15     f1 =  -x2 * x3 + a
16     return sqrt(log(x1 * x2) + (x2/x3)) + f1
17 variables = ['x1','x2','x3']
18 constants = ['a']
19 a = gr.GradientFM([X,Y,Z],variables,constants,'teste.py')
```

Figure 2 – Data input in DAALGPY

On Figure 2, a use example of the derivation package is shown, where:

- Line 7 imports the derivation module DAALGPY;
- Line 9 introduces equation X as a string;
- Lines 11 and 12 build function Y;
- Lines 14 to 15 build function Z;
- Line 17 defines the list of variables to be analized;
- Line 18 defines the list of constants in the problem;
- Line 19 uses the DAALGPY module in the reverse mode. From left to right: a list with all the equations to be derived, the variables, the problem constants and a string pointing to a *py* file name, generated to numerically calculate the jacobian.

As it analyzes functions or strings, the algorithm uses the native module *dis* to interpret and transform the objects bytecode into a text file in ASCII format. The *dis* module returns a simple human readable representation of how the machine interprets the working code. The file generated by the *dis* module contains 5 columns, as shown in Fig. 3.

- The first indicates the initial line for that function or string;
- The second represents the machine address of that bytecode instruction;
- The third, the bytecode instruction's name;
- The fourth column brings the index where that argument is allocated in the table of variables and constants
- The last column shows the human code name of the variable

```
 1 | 21           0 LOAD_FAST                1 (x2)
 2                2 UNARY_NEGATIVE
 3                4 LOAD_FAST                2 (x3)
 4                6 BINARY_MULTIPLY
 5                8 LOAD_GLOBAL              0 (a)
 6               10 BINARY_ADD
 7               12 STORE_FAST               3 (f1)
 8
 9  22           14 LOAD_GLOBAL              1 (sqrt)
10               16 LOAD_GLOBAL              2 (log)
11               18 LOAD_FAST                0 (x1)
12               20 LOAD_FAST                1 (x2)
13               22 BINARY_MULTIPLY
14               24 CALL_FUNCTION            1
15               26 LOAD_FAST                1 (x2)
16               28 LOAD_FAST                2 (x3)
17               30 BINARY_TRUE_DIVIDE
18               32 BINARY_ADD
19               34 CALL_FUNCTION            1
20               36 LOAD_FAST                3 (f1)
21               38 BINARY_ADD
22               40 RETURN_VALUE
```

Figure 3 – Bytecode output of equation Z.

Graphs are constructed from the interpretation of the txt file, as the data generated by the *dis* module are represented by Reverse Polish Notation. Nevertheless, for the subsequent operations, it is necessary to convert the representation to direct algebraic, where it is easier to apply the chain rule. After conversion of each function f_i ($R^n \rightarrow R$), a computational dictionary is created to store the computational graph, as shown in Fig. 4.

```
{'t0': 'x1', 't1': 'x2', 't2': 'x3', 't3': '(-1)', 't4': '(t3*t1)',
't5': '(t4*t2)', 't6': '(t5+a)', 't7': '(t0*t1)', 't8': '(log(t7))',
't9': '(t1/t2)', 't10': '(t8+t9)', 't11': '(sqrt(t10))', 't12':
'(t11+t6)'}
```

Figure 4 – Vertices of the computational graph stored in a dictionary for equation Z.

After the graph representing the operations is constructed, there is a bifurcation on how the data is treated according to the chosen differentiation method. In the forward mode (as Fig. 5), lists are created to indicate from which variables each node depends. As for the reverse mode, the lists will contain the paths walked by each variable, as shown in Fig. 6.

```
{'t2': ['t2'], 't1': ['t1'], 't0': ['t0'], 't3': ['t1'], 't4': ['t1', 't2'], 't5':
['t1', 't2'], 't6': ['t1', 't2'], 't7': ['t1', 't2'], 't8': ['t1', 't2'], 't9':
['t0', 't1'], 't10': ['t0', 't1'], 't11': ['t1', 't0', 't2'], 't12': ['t1', 't0',
't2'], 't13': ['t1', 't0', 't2']}
```

Figure 5 – List on the forward mode.

```
{'t0': [[9, 10, 11, 12, 13]], 't1': [[3, 4, 5, 6, 13], [7, 8, 11, 12, 13], [9, 10,
11, 12, 13]], 't2': [[4, 5, 6, 13], [7, 8, 11, 12, 13]]}
```

Figure 6 – Paths on the reverse mode.

Using the paths or the dependencies, the derivatives are calculated for every node, both in the forward or reverse mode. For the implementation of the forward mode, the graph is run one more time to determine each derivative, similar to what Birgin [5] indicates for the cases of

gradients with few nonzero entries. Unfortunately, this strategy reduced the efficiency of the forward mode from a processing time standpoint.

The last step for the module is to create the output functions in one of two ways: a list containing the respective symbolic derivatives (as shown in Fig. 7) or a *py* file with a function to numerically compute the derivatives of the original function, according to the chosen derivation technique.

```
[['(0.5*((((log((x1*x2)))+(x2/x3)))**(-0.5)))*(1/((x1*x2)))*(x2)',
  '(x3)*((-1))+(0.5*((((log((x1*x2)))+(x2/x3)))**(-0.5)))*(1/
((x1*x2)))*(x1)+(0.5*((((log((x1*x2)))+(x2/x3)))**(-0.5)))*(1/(x3))',
  '(((-1)*x2))+(0.5*((((log((x1*x2)))+(x2/x3)))**(-0.5)))*((-1)*x2/
(x3**2))']]
```

Figure 7 – Data output as symbolic variable for the example.

4 Energy Methods

The strain energy (U) of a given body is the energy it absorbs under the influence of an external loading, neglecting the rigid body displacements. For a beam of isotropic material under normal forces and bending, the stored strain energy is connected to its normal stress and strain as

$$U = \frac{1}{2}\int_V \sigma_{xx}\varepsilon_{xx}dV. \tag{1}$$

Besides of the strain energy, the action of a given load generates an external work (Ω). It is the product of the load (Q_i) and the displacement (u_i) in its application point. As loading, it is encompassed volumetric (ρ), superficials (b), concentrated loads (P_i), couples (M_j) e torques (T_k) acting on a body

$$\Omega = \int_V \rho \cdot u\, dV + \int_S b \cdot u\, dS + \sum_n P_i \cdot u_i + \sum_n M_j \cdot u_j + \sum_n T_k \cdot u_k. \tag{2}$$

In a structural system whose strains occur in a gradual manner (quasi-static loading), kinetic energy can be neglected, so the total potential energy of the system is

$$\Pi = U - \Omega. \tag{3}$$

It is known that internal energy and external work vary with displacements, so that the total potential energy can be expressed as a function of the structure's nodal displacements. From the first theorem of Castigliano, the partial derivative of the strain energy U with relation to any displacement u_i is equal to a corresponding internal force F_i. By its turn, the derivative of the external work by the nodal displacement is the generalized load Q_i itself. With this information at hand, the equilibrium condition is imposed according to variational theorems as the point of minimal potential energy, that is, $\|\nabla\Pi\| = 0$.

$$0 = \frac{dU}{du_i} - Q_i \tag{4}$$

Deriving the strain energy function in relation to the n nodal displacements to which the

structure is subject, it is possible to establish the system of equations responsible to describe the structural equilibrium. The solution of this system of equations consists in finding the nodal displacement values, which can be determined by numerical methods such as Newton-Raphson.

4.1 Kinematics of a truss element

In order to determine the strain energy in a truss rod, an element is shown in Fig. 8 in a global *XY* system of coordinates, with its position before and after a generic loading is applied. It is imposed that the rod stays straight the whole time.

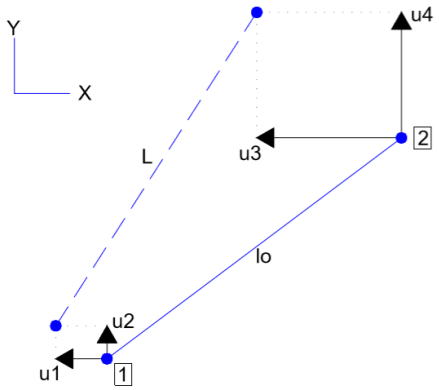

Figure 8 – Nodal displacements.

From Fig. 8, the initial nodal coordinates are

$$x_i^0 = \begin{pmatrix} x1_i^0 \\ x2_i^0 \end{pmatrix} \text{ and } y_i^0 = \begin{pmatrix} y1_i^0 \\ y2_i^0 \end{pmatrix}, \tag{5}$$

and the final coordinates are

$$x_i = \begin{pmatrix} x1_i^0 \\ x2_i^0 \end{pmatrix} + \begin{pmatrix} u1 \\ u3 \end{pmatrix} \text{ and } y_i = \begin{pmatrix} y1_i^0 \\ y2_i^0 \end{pmatrix} + \begin{pmatrix} u2 \\ u4 \end{pmatrix}. \tag{6}$$

The lengths of the bar before and after deformation are respectively

$$\begin{aligned} L_0 &= \sqrt{\left(x2_i - x1_i\right)^2 + \left(y2_i - y1_i\right)^2} \text{ and} \\ L &= \sqrt{\left(u3 - u1 + x2_i - x1_i\right)^2 + \left(u4 - u2 + y2_i - y1_i\right)^2}, \end{aligned} \tag{7}$$

so the change in length for the rod is

$$\Delta L = L - L_0. \tag{8}$$

From mechanics of materials, the axial strain on the bar is

$$\varepsilon_{xx} = \frac{\Delta L}{L_0}. \tag{9}$$

Linear elasticity is assumed, so the constitutive relation is expressed by Hooke's law and strain energy can be expressed as

$$U = \frac{1}{2}\int_V E \cdot \varepsilon_{xx}{}^2 dV. \tag{10}$$

A truss element is considered to be of constant area and straight. As no external load is applied along the bar (only at the nodes), ε_{xx} is constant, so the internal energy in an element is

$$U_e = \frac{1}{2}\int_0^{l_0} E \cdot \varepsilon_{xx}{}^2 \cdot A \cdot dx = \frac{E \cdot \varepsilon_{xx}{}^2 \cdot A \cdot l_0}{2}. \tag{11}$$

If Eq. (8) is considered in its original form, the length of the bar is nonlinear in relation to nodal displacements and consequently the strain energy function and the nodal forces. This is said to be a geometrically nonlinear analysis and its linear counterpart comes from the linearization of Eq. (8) around the initial length with a Taylor series as

$$L(u1,u2,u3,u4) = \frac{u4\cdot(y2_i - y1_i)}{l_0} + \frac{u2\cdot(y1_i - y2_i)}{l_0} + \frac{u3\cdot(x2_i - x1_i)}{l_0} + \frac{u1\cdot(x1_i - x2_i)}{l_0} + l_0. \tag{12}$$

Linear analyzes are easier to formulate and render a linear system of equations, much faster to compute. On the other hand, only with geometrically nonlinear analyzes, can higher order effects be captured, what is important in structures with large displacements.

4.2 Kinematics of a frame element

A beam of length L_o, Young modulus E, moment of inertia I and cross-section area A is shown in a local frame of reference in Fig. 9. Each beam node has 3 degrees of freedom (vertical, horizontal and rotation).

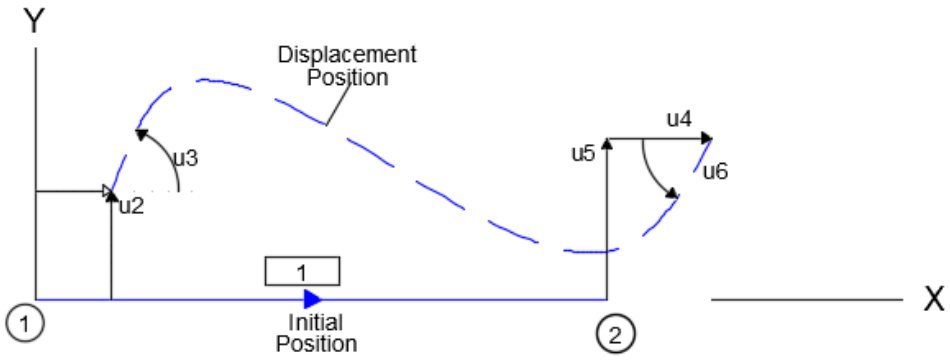

Figure 9 – Kinematics of a frame beam

The basic hypotheses for the beam elements are:

- The body is continuous and deformable, it has length much larger than cross-section dimensions;
- In each discretized element, the cross-section area is constant and the material is

homogeneous and isotropic;

- Euler-Bernoulli kinematics is assumed, so cross-sections remain plane and rigid and deformations due to shear forces are neglected;
- Moderate rotations are assumed and small horizontal strains, so for the deformed configuration, normal forces increase the bending stiffness of the element.

Displacements are given by

$$u_x = u_0 - d \cdot \frac{dv}{dx}, \text{ and } u_y = v. \tag{13}$$

Considering moderate rotations and small displacements, the Green-Lagrange deformation tensor can be expressed in an additive form:

$$\varepsilon_{xx} = \varepsilon_L + \varepsilon_{NL} \tag{14}$$

The linear and nonlinear terms of the normal strain are stated in Eqs. (16) and (17):

$$\varepsilon_L = \frac{du_0}{dx} - d \cdot \frac{d^2v}{dx^2} \tag{15}$$

$$\varepsilon_{NL} = \frac{1}{2}\left(\frac{dv}{dx}\right)^2 \tag{16}$$

The beam's strain energy function is

$$U = \frac{1}{2}\iint E \cdot \left(\varepsilon_L + \varepsilon_{NL}\right)^2 \cdot dA \cdot dx. \tag{17}$$

A mapping is used for easier computer implementation, from a variable $\xi \in [-1,1]$ to $x \in [0, l]$.

$$\xi = \frac{2x - x1 - x2}{\left(x2 - x1\right)} = \frac{2x - x1 - x2}{l} \tag{18}$$

The displacements are approximated with Lagrange and Hermite polynomials for a two-node element, the displacement fields are represented as

$$\begin{aligned} u &= N1 \cdot u1 + N2 \cdot u4 \text{ and} \\ v &= H1 \cdot u2 + H2 \cdot u3 \cdot \frac{l}{2} + H3 \cdot u5 + H4 \cdot u6 \cdot \frac{l}{2} \cdot \end{aligned} \tag{19}$$

The strain energy in an element is than

$$\begin{aligned} U_e = & \\ & \frac{2 \cdot A \cdot \varepsilon_0 \cdot E}{l^2} \int_{-1}^{1} \left(\frac{dv}{d\xi}\right)^2 d\xi + E \int_{-1}^{1} \frac{A}{l} \cdot \left(\frac{du_0}{d\xi}\right)^2 d\xi \\ & + E \int_{-1}^{1} \frac{4 \cdot I}{l^3} \cdot \left(\frac{d^2v}{d\xi^2}\right)^2 d\xi + \frac{E}{l^3} \int_{-1}^{1} A \cdot \left(\frac{dv}{d\xi}\right)^4 d\xi \end{aligned} \tag{20}$$

Using a moderate rotations assumption, $(dv/d\xi)^4$ can be neglected as the derivatives are

to the fourth power, thus

$$U_e = \frac{2 \cdot A \cdot \varepsilon_0 \cdot E}{l^2} \int_{-1}^{1} \left(\frac{dv}{d\xi} \right)^2 d\xi + E \int_{-1}^{1} \frac{A}{l} \cdot \left(\frac{du_0}{d\xi} \right)^2 d\xi + E \int_{-1}^{1} \frac{4 \cdot I}{l^3} \cdot \left(\frac{d^2 v}{d\xi^2} \right)^2 d\xi \tag{21}$$

The linear equivalent is obtained by neglecting the higher order terms, specifically ε_{NL}, which depends on the square of the displacements,

$$U_e = E \int_{-1}^{1} \frac{A}{l} \cdot \left(\frac{du_0}{d\xi} \right)^2 d\xi + E \int_{-1}^{1} \frac{4 \cdot I}{l^3} \cdot \left(\frac{d^2 v}{d\xi^2} \right)^2 d\xi. \tag{22}$$

5 Newton-Raphson's method

From the many methods for solving a system of nonlinear equations, Newton-Raphson's is probably the more broadly used. Ruggiero and Lopes [6] detail its deduction and presents a complete algorithm, adapted here for the nonlinear elasticity problem.

Let $F(x) = 0$ be a system of nonlinear equations in multiple variables (gathered in vector x) and $J(x) = \nabla F$ its Jacobian. If η is a limiting error measure and x^k is an initial guess, a iterative procedure is taken as

- Calculate $J(x^k)$ and $F(x^k)$;
- Obtain s^k as the solution to the linear system of equations $J(x^k) \cdot s^k = -F(x^k)$;
- Update the solution as $x^{k+1} = x^k + s^k$;
- If $\sqrt{\frac{\sum (s^k)^2}{\sum (x^k)^2}} < \eta$, then x^{k+1} is the solution; else, $k = k + 1$ and repeat.

The adopted convergence criterium is connected to the change in nodal displacements in each iteration. According to Kassimali [7], this criterion usually renders precise results for structures whose stiffness decrease with an increasing load.

6 Validation

The package DAALGPY was validated by applying it in a program for the structural analysis of trusses and plane frames, developed in Python, both for linear elastic and geometrically nonlinear elastic analysis, and its results were compared to benchmarks available on the literature.

The analyses were carried using an incremental and iterative process with load steps, thus obtaining more expressive results for the structural behavior. For each load step, the code uses Newton-Raphson's method to solve the system of equations, either linear or nonlinear. As the iterative procedure require an initial guess, the nodal displacements for the last converged load step is used as initial guess for the following. If the previous load step did not converge, a linear analysis around the undeformed state is taken as predictor.

It is necessary for as input parameters: the nodal coordinates, the applied loads, the structure's support conditions and nodal connectivity, area, moment of inertia and Young's modulus for each element. Optional parameters are the maximum number of iterations in a load step and convergence tolerance.

Automatic differentiation was used in both steps needed for this analysis. The strain energy

function is given as described by Eqs. (12) and (13) and its derivatives renders the internal force vector for the structure. By its turn, the derivative of the internal force vector gives the tangent stiffness matrix, used as the Jacobian in Newton-Raphson's method.

For the frame implementation, it was necessary to make use of numerical integration, using Gauss quadrature to evaluate the strain energy function, thus also for the derived internal force vector and tangent matrix. A quadrature with 3 points was used as it suffices for both linear and nonlinear versions of the strain field.

The stiffness matrices generated with the AD package were compared with implementations by Kassimali [7] and Zermiani [8]. The results for nodal displacements and internal forces were also compared as a general indicator for the solution process.

7 Results and discussion

7.1 Plane truss of 5 rods – linear and nonlinear analysis

This example proposed by Segnini [9] consists of a plane truss buit out of five bars, all made of a linear elastic material with Young's modulus $E = 20\ GPa$ and cross-sectional area of $A = 0{,}001\ m^2$. The structure is subject to a concentrated downward load $Pl = 180\ kN$ at node 1. Figure 10 below presents a schematic drawing of the truss.

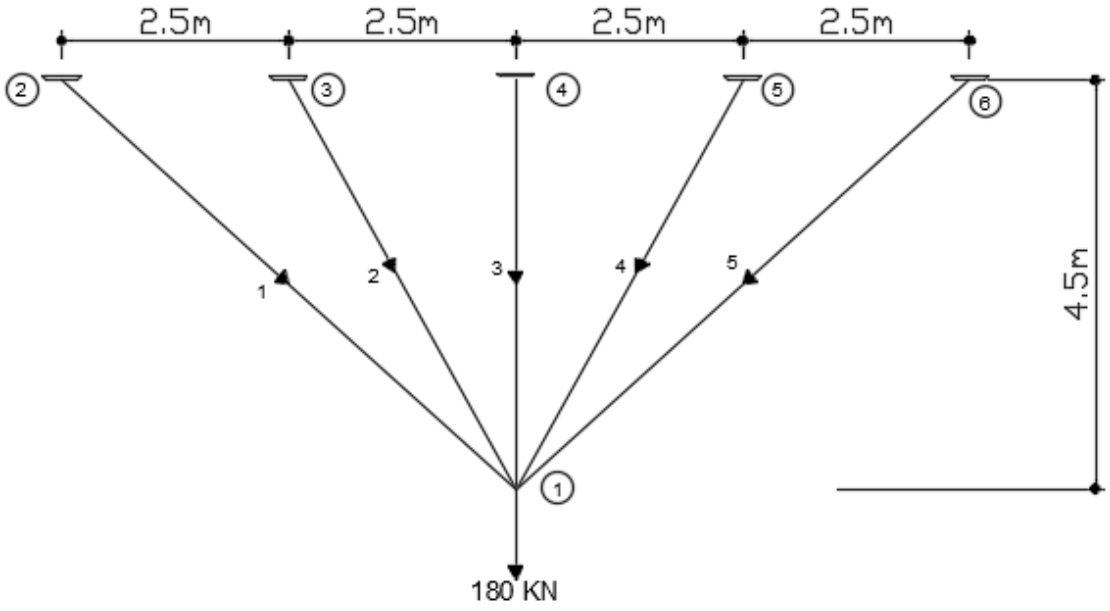

Figure 10 – Five bar truss.

The analysis was carried accounting for geometrical nonlinearity. Ten load steps were used with a displacement tolerance of $\eta = 0{,}001$. The results were compared with the nonlinear analysis in Segnini [9]. Table 1 below shows the final displacements for the single degree of freedom for each load step.

Table 1 – Vertical displacement for the five-bar truss

P/Pl	Vertical displacement of node 1 (*cm*)		%
	Segnini [9]	AD	
0,1	-0,001380	-0,001380	0,00%
0,2	-0,002760	-0,002760	0,00%
0,3	-0,004139	-0,004139	0,00%
0,4	-0,005518	-0,005518	0,00%
0,5	-0,006897	-0,006897	0,00%
0,6	-0,008275	-0,008275	0,00%
0,7	-0,009653	-0,009653	0,00%
0,8	-0,011031	-0,011031	0,00%
0,9	-0,012409	-0,012409	0,00%
1,0	-0,013786	-0,013786	0,00%

It can be noticed that convergency was achieved for every value. Besides the vertical displacement, the tangent stiffness matrix analytically developed by Kassimali [7] is compared to the one derived algorithmically by DAALGPY, as is shown in Fig. 11 for beam 2 in the last iteration of the last load step.

```
BAR
2
DAALGPY MATRIX
[[  918.97714384 -1642.83710524  -918.97714384  1642.83710524]
 [-1642.83710524  2975.24377547  1642.83710524 -2975.24377547]
 [ -918.97714384  1642.83710524   918.97714384 -1642.83710524]
 [ 1642.83710524 -2975.24377547 -1642.83710524  2975.24377547]]
KASSIMALI ANALYTICAL MATRIX
[[  918.97714384 -1642.83710524  -918.97714384  1642.83710524]
 [-1642.83710524  2975.24377547  1642.83710524 -2975.24377547]
 [ -918.97714384  1642.83710524   918.97714384 -1642.83710524]
 [ 1642.83710524 -2975.24377547 -1642.83710524  2975.24377547]]
DIFFERENCE
[[ 2.27373675e-13 -2.27373675e-13 -2.27373675e-13  2.27373675e-13]
 [-2.27373675e-13  0.00000000e+00  2.27373675e-13  0.00000000e+00]
 [-2.27373675e-13  2.27373675e-13  2.27373675e-13 -2.27373675e-13]
 [ 2.27373675e-13  0.00000000e+00 -2.27373675e-13  0.00000000e+00]]
```

Figure 11 – Comparison of the tangent stiffness matrices for rod 2.

For this example, the divergency between the stiffness matrices was found in the 13th decimal, possibly unnoticeable within Python's float point arithmetic system.

The verify the computations for the internal force vector, a comparison was carried to the results in Sales [1] for a linear analysis of the structure. Equation (15) was used for the linear approximation of the strain energy, so that a linear system of equations was obtained.

Table 2 – Comparison of axial forces.

Bar	Axial force (kN)		Difference %
	Sales [1]	AD	
1	27,4436	27,4481	0,02%
2	46,8386	46,8690	0,06%
3	61,2728	61,3347	0,10%
4	46,8386	46,8690	0,06%
5	27,4436	27,4481	0,02%

The results for displacements and axial forces agree with the literature, corroborating to validate this implementation.

7.2 Single beam truss – nonlinear analysis

This example proposed by Segnini [9] consists of a plane truss with a single rod (as to simulate a symmetrical triangular configuration). The rod is mad of elastic linear material with elasticity modulus $E = 20500\, KN/cm^2$ and cross-section area of $A = 6.526\, cm^2$. The structure is subject to a concentrated downward load $Pl = 10\, kN$ at node 2. Figure 12 below presentes a schematic drawing for the truss.

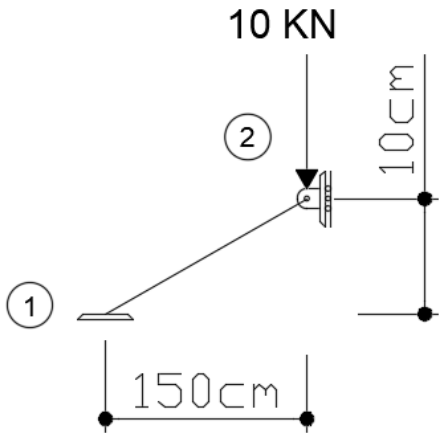

Figure 12 – Single rod truss.

Equilibrium was imposed by the minimization of the strain energy function taking into account geometrical nonlinearity. Ten load steps were used with a tolerance of $\eta = 0.001$. Results were compared to those obtained by Segnini [9] and Sales [1] also for nonlinear analysis. Table 3 contains the final displacements for the vertical displacement of node 2.

Table 3 – Results for single rod truss.

Load (kN)	Vertical displacement of node 2 (cm)				
	AD	Segnini[9]	Difference %	Sales[1]	Difference %
1	-0,264	-0,264	0,00%	-0,264	0,00%
2	-0,553	-0,553	0,00%	-0,553	0,00%
3	-0,872	-0,872	0,00%	-0,872	0,00%
4	-1,234	-1,234	0,00%	-1,234	0,00%
5	-1,658	-1,658	0,00%	-1,658	0,00%
6	-2,187	-2,187	0,00%	-2,187	0,00%
7	-2,957	-2,957	0,00%	-2,957	0,00%
8	-21,619	-2,902	644,97%	-21,619	0,00%
9	-21,783	-21,783	0,00%	-21,783	0,00%
10	-21,941	-21,941	0,00%	-21,941	0,00%

For this example, a large discrepancy was found for the load of $8\,KN$. The value of $-21{,}619\ cm$ was found in the present work with AD as well as by Sales [1] with the stiffness matrix obtained analytically, both with a large number of iterations. The value of $-2{,}902\ cm$ obtained by Segnini [9] can be reproduced ignoring convergency and limiting the Newton-Raphson process to 25 iterations (as described by the author). Thus, it is possible to admit that the correct value is close to $-21{,}619\ cm$ and that the present application is satisfactory for geometrically linear and nonlinear trusses.

7.3 Column under eccentric compression

This example proposed by Zermiani [8] consists of a balance column made of elastic linear material with Young's modulus $E = 943\ KN/cm^2$, cross-section area $A = 225{,}00\ cm^2$ and moment of inertia $I = 4218{,}75\ cm^4$. The structure is subject to a concentrated downward load $Pl = 39{,}24\ kN$ and a couple $Ml = -220{,}43\ kNcm$. Three meshes were used: 2, 3 and 11 nodes. Figure 12 below shows a schematic representation of the problem.

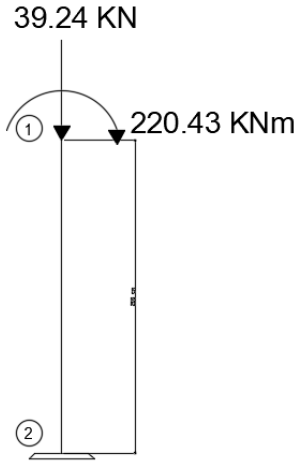

Figure 12 – Column under eccentric load.

The column was analyzed with a nonlinear implementation and later with the linear formulation for comparison. The nonlinear approach used ten load steps and convergence tolerance of $\eta = 0{,}001$. Results were compared for the displacement on the top of the column

and the moment on the bottom.

Table 4 – Results for the column under eccentric load – nonlinear analysis.

Mesh	Horizontal displacement (cm)		Difference %	Reaction moment $(kNcm)$		Difference %
	Zermiani	AD		Zermiani	AD	
2	2,323334	2,323334	0,00%	311,5974	311,5976	0,00%
3	2,324826	2,324824	0,00%	311,6566	311,6561	0,00%
11	2,324733	2,324926	0,01%	311,6391	311,6601	0,01%

The difference between the analyses is neglectable, so the results can be regarded as valid. As for the trusses, also the local stiffness matrix was compared to the one Zermiani [8] derives analytically. Figure 13 compares the local stiffness matrix for element 10 in its last iteration, obtained by AD and analytically. Once again, the difference in results are inside the precision for Python's float point arithmetic.

```
BAR
10
DAALGPY MATRIX
[[  8487.           0.           0.       -8487.           0.
       0.     ]
 [     0.        3053.43648  38187.576        0.       -3053.43648
   38187.576  ]
 [     0.       38187.576   636394.2          0.      -38187.576
  318295.2    ]
 [ -8487.           0.           0.        8487.           0.
       0.     ]
 [     0.       -3053.43648 -38187.576        0.        3053.43648
  -38187.576  ]
 [     0.       38187.576   318295.2          0.      -38187.576
  636394.2    ]]
ZERMIANI MATRIX
[[  8487.           0.           0.       -8487.           0.
       0.     ]
 [     0.        3053.43648  38187.576        0.       -3053.43648
   38187.576  ]
 [     0.       38187.576   636394.2          0.      -38187.576
  318295.2    ]
 [ -8487.           0.           0.        8487.           0.
       0.     ]
 [     0.       -3053.43648 -38187.576        0.        3053.43648
  -38187.576  ]
 [     0.       38187.576   318295.2          0.      -38187.576
  636394.2    ]]
DIFFERENCE
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  7.27595761e-12 -1.16415322e-10  0.00000000e+00
  -7.27595761e-12 -5.82076609e-11]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  7.27595761e-12 -5.82076609e-11  0.00000000e+00
  -7.27595761e-12 -1.16415322e-10]]
```

Figure 13 – Local stiffness matrices for elemento 10 in nonlinear analysis.

For the linear analysis, the results are summarized in Table 5.

Table 5 – Results for the column under eccentric load – linear analysis.

Mesh	Horizontal Displacement (cm)		Difference %	Moment at support (kNcm)		Difference %
	Zermiani	AD		Zermiani	AD	
2	1,73151	1,73151	0,00%	220,4299	220,4300	0,00%
3	1,73151	1,73151	0,00%	220,4301	220,4300	0,00%
11	1,73143	1,73151	0,00%	220,4298	220,4300	0,00%

The small fluctuations on the results are also probably to numerical accuracy, as those agree also with hand calculations. The linear stiffness matrix is shown in Fig. 14.

```
BAR
10
DAALGPY MATRIX
[[  8487.        0.        0.    -8487.        0.        0.  ]
 [     0.     3055.32  38191.5       0.    -3055.32  38191.5 ]
 [     0.    38191.5  636525.        0.   -38191.5  318262.5 ]
 [ -8487.        0.        0.     8487.        0.        0.  ]
 [     0.    -3055.32 -38191.5       0.     3055.32 -38191.5 ]
 [     0.    38191.5  318262.5       0.   -38191.5  636525.  ]]
ZERMIANI MATRIX
[[  8487.        0.        0.    -8487.        0.        0.  ]
 [     0.     3055.32  38191.5       0.    -3055.32  38191.5 ]
 [     0.    38191.5  636525.        0.   -38191.5  318262.5 ]
 [ -8487.        0.        0.     8487.        0.        0.  ]
 [     0.    -3055.32 -38191.5       0.     3055.32 -38191.5 ]
 [     0.    38191.5  318262.5       0.   -38191.5  636525.  ]]
DIFFERENCE
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00 -4.54747351e-13 -7.27595761e-12  0.00000000e+00
   4.54747351e-13 -7.27595761e-12]
 [ 0.00000000e+00  0.00000000e+00 -1.16415322e-10  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  4.54747351e-13  7.27595761e-12  0.00000000e+00
  -4.54747351e-13  7.27595761e-12]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00 -1.16415322e-10]]
```

Figure 14 – Linear stiffness matrix for element 10.

7.4 Plane frame for a building

This example, proposed by Banki [10] and studied by Junges, La Rovere e Loriggio [11], consists of a frame used as bracing structure for a small building, subject to vertical and horizontal loads. The geometrical characteristics of the frame are: columns and beams made of linear elastic material with Young's modulus $E = 27\,000\,MPa$, beams cross-sections of $13 \times 35\ cm$ and columns $30 \times 20\ cm$. A mesh of 12 beam elements was used for the analysis. Figure 15 presents a schematic drawing with the proposed load.

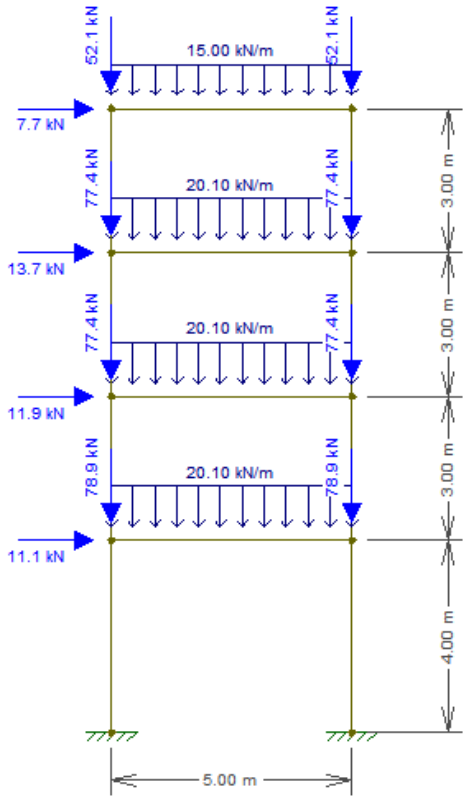

Figure 15 – Frame for a small building

A geometrically nonlinear analysis was made, with 10 load steps and a convergency tolerance of $\eta = 0.001$. The results are compared to the bibliography for the horizontal displacement at the top of the frame and are shown in Table 6.

Tabela 6 – Results for the small building

Horizontal displacement (*cm*)				
AD	Banki	Difference %	Junges et. al (SAP 2000)	Difference %
4.868	4.866	0.04%	4.872	0.08

Later, the loads were amplified as to subject the structure to its limit load, obtaining the following curve of stress as function of the displacement:

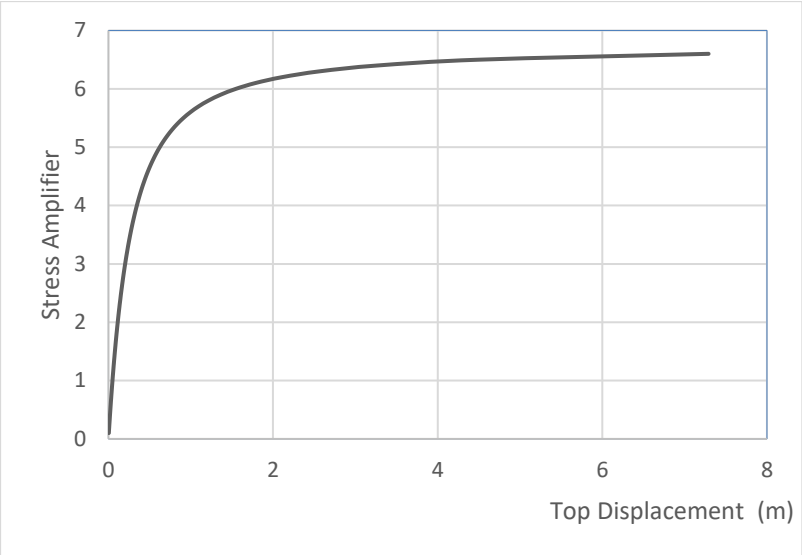

Figure 16 – Load – displacement analysis for the frame

It can be noticed from Fig. 16 that the frame's displacements are subject to nonlinear effects. The ultimate load for global buckling of the frame is approximately 6 times larger than the load in Fig. 15.

8 Conclusion

The results shown in section 7 display the structural responses obtained using the DAALGPY automatic differentiation package are satisfactory when compared with numerical results in the literature. The only discrepancy was found in section 7.2 but is most likely not related to the differentiation algorithm.

It was noticed that the values found for the local stiffness matrices for frames and trusses show small differences with the analytical expressions by the order of 10^{-16}. As stated before, this difference is most likely due to the float point arithmetic system, as the functions obtained by AD usually have many more operations than the simplified analytical expressions, thus are more susceptible to computational limitations.

An interesting point using the AD algorithm was the possibility to chance between linear and nonlinear analysis by a simple change on the expression for the strain energy function. This flexibility made possible to run different analysis inside the same computer code, each taking different considerations for the structural kinematics, generating simple but flexible programs.

The implementation can easily be extended to use more complex formulations for geometrically exact nonlinear formulations of frame structures, taking into account large deformations, physical nonlinearity for trusses and frames, application to topology optimization among others.

References

[1] D. C. Sales. *Análise Geometricamente Não Linear de Treliças Planas*. Bachelor Dissertation. University Federal of Sergipe, 2019.

[2] J. F. M. Barthelemy and L. E. Hall. *Automatic Differentiation as a Tool in Engineering Design*. NASA Technical Memorandum, 107661, 1992.

[3] F. Šrajer, Z. Kukelova and A. Fitzgibbon. *A Benchmark of Selected Algorithmic Differentiation Tools on Some Problems in Computer Vision and Machine Learning*. AD2016—7th International Conference on Algorithmic Differentiation, and in Optimization Methods and Software, Taylor and Francis, Feb 2018.
[4] R. V. Linn, *Shape optimization of shell structures based on NURBS description using automatic differentiation.* University Federal of Rio Grande do Sul, 2010.
[5] E. J. G. Birgin. *Diferenciação Computacional e Aplicações.* PhD Thesis, University State of Campinas, 1998.
[6] M. A. G. Ruggiero and V. L. R. Loppes. *Cálculo Numérico – Aspectos Teóricos e Computacionais*. Pearson, 2006.
[7] A. Kassimali. *Matrix Analysis of Structures*. Cegange Learning, 2012.
[8] F. L. Zermiani. *Contribuição à Análise Não-Linear Geométrica de Pórticos Planos*. MSc thesis, University Federal of Santa Catarina, 1998.
[9] S. C. A. Segnini. *Estudo Comparativo de Formulações para a Análise Não-Linear Geométrica de Treliças*. MSc thesis, Campinas State University, 2000.
[10] A. L. Banki. *Estudo sobre a inclusão da não linearidade geométrica em projetos de edifícios. Florianópolis.* Msc thesis, University Federal of Santa Catarina, 1999.
[11] E. Junges, H. L. La Rovere and D. D. Loriggio. *Análise de segunda ordem global de estruturas de concreto armado utilizando programas computacionais de dimensionamento.* 54º Congresso Brasileiro do Concreto, October 2012.