ARTIGO ORIGINAL
DOI: 10.5433/1679-0375.2018v39n1p49

Application and comparison of numerical methods in the solution of systems of linear equations in space trusses problems

Aplicação e comparação de métodos numéricos na solução de sistemas de equações lineares em problemas de treliças espaciais

Luiz Antonio Farani de Souza[1]; Rodrigo dos Santos Veloso Martins[2]; Juliana Castanon Xavier[3]; Johannes Hosp Porto[4]

Abstract

One of the interesting civil engineering applications is space truss, a three dimensional element, particularly used as roof for industrial and commercial structure, covering large open areas with few or no internal supports. This paper aims to compare different numerical methods implemented computationally for the solution of the linear equations systems generated from the Newton-Raphson iterations in the incremental process. The numerical solution of such linear systems are computationally costly, thus it is of our interest to determine which numerical methods best suit the problem. To compare the computational cost of the algorithms, a study of complexity is performed. Numerical simulations with the Matlab software are made of space truss problems with geometric nonlinear behavior. The system of nonlinear equations is solved using the Standard Newton-Raphson method associated with the Linear Arc-Length path-following technique. The structures are discretized by the Finite Element Positional Method, whose fundamental variables are the nodal positions of the finite element. For the processing time, the numerical results indicate a better performance of the iterative methods compared to direct methods, especially for large orders problems, which are most commonly found in the structural area.
Keywords: Space Truss. Finite Element Positional Method. Conjugate Gradient.

Resumo

Uma das aplicações interessantes da engenharia civil é a treliça espacial, um elemento tridimensional, particularmente utilizado como cobertura para estruturas industriais e comerciais, cobrindo grandes áreas abertas com poucos ou nenhum apoio interno. Este artigo tem como objetivo comparar diferentes métodos numéricos implementados computacionalmente para a solução do sistema de equações lineares gerado da iteração de Newton-Raphson no processo incremental. A solução numérica de tais sistemas lineares é computacionalmente onerosa, portanto é do nosso interesse determinar quais métodos numéricos melhor se adéquam ao problema. Para comparar o custo computacional dos algoritmos, um estudo da complexidade é realizado. Simulações numéricas com o programa Matlab são feitas de problemas de treliças espaciais com comportamento não linear geométrico. O sistema de equações não lineares é solucionado com o método de Newton-Raphson Padrão associado a técnica de continuação Comprimento de Arco Linear. As estruturas são discretizadas por meio do método dos Elementos Finitos Posicional, cujas variáveis fundamentais são as posições nodais do elemento finito. Para o tempo de processamento, os resultados numéricos indicam um melhor desempenho dos métodos iterativos em comparação com métodos diretos, especialmente para problemas de ordem maior, que são comumente os mais encontrados na área estrutural.
Palavras-chave: Treliça Espacial. Método Posicional de Elementos Finitos. Gradiente Conjugado.

[1] Prof. Dr. Depto. de Eng. Civil, UTFPR, Apucarana, Pr, Brasil; E-mail: lasouza@utfpr.edu.br
[2] Prof. Dr. Depto. Acadêmico de Matemática, UTFPR, Pr, Brasil; E-mail: rodrigomartins@utfpr.edu.br
[3] Profa. Dra. Depto. Acadêmico de Matemática, UTFPR, Apucarana, Pr, Brasil; E-mail: julianaxavier@utfpr.edu.br
[4] Discente, Graduação, Eng. Civil, UTFPR, Apucarana, Pr, Brasil; E-mail: johannes@alunos.utfpr.edu.br

Introduction

Trusses represent an efficient structural system that can sustain considerable loads with a smaller amount of materials. Since the beginning of its commercial use, this system has been increasingly popular, especially in large open areas with few or no intermediary support. Successful applications of lattice structural systems include residential and public buildings, bridges, tanks, television towers, industrial buildings, and a great variety of other types of structures (SEÇER, 2009). See an arc of an hangar in Figure 1.

Figure 1: Arc of a hangar.

Source: (PERELMUTER; FIALKO, 2005).

Spatial trusses combine structural efficiency and a significant cost reduction. The development and utilization of spatial trusses in Brazil had a big boost with the construction, in São Paulo city, of the Anhembí Exposition Center in the late 60's. About 48000 aluminum tubular bars compose its spatial truss, projected by the Canadian engineer Cedric Marsh to a covered area of 62500 m^2. In the upcoming decades, the use of trusses in spatial structures multiplied in Brazil with constructions of relevance importance and international repercussion. See for example the covered structure of the brewery Brahma, in Rio de Janeiro, with 132000 m^2 of covered area (30 m to 60 m of free span) and the Brasilia Fair Trade Pavilion with 57000 m^2 of covered area, assembled in only 100 days (SOUZA, 2003).

One step in the practical computation of the finite element approximation is the solution of a nonlinear algebraic equations system, frequently containing a large number of degrees of freedom. This system is commonly solved using the Newton-Raphson method, in an incremental and iterative procedure, requiring the solution of an equally large linear system of algebraic equations on each iteration (WHITELEY, 2017).

Structural analysis may involve models and schemas which are quite typical and the finite element mesh can contain from 20 to 30 thousand nodes, 30 to 50 thousand elements of various types (bars, plates, shells, elastics links), and have more than one hundred of stiffness properties sets (PERELMUTER; FIALKO, 2005).

The quick solvers in the Finite Element Method (FEM) analysis available today include direct methods for sparse matrices and highly efficient iterative methods. Should the number of degrees of freedom of a linear system of algebraic equations be sufficiently large, it is not feasible to solve the system using direct methods, such as LU factorisation. Hence it is preferable to use the iterative methods. The choice of iterative method depends on the properties of the linear system being solved. The Conjugate Gradient method (CG) is one of the most efficient iterative methods for the linear system solution, whose coefficient matrix (stiffness matrix) is symmetric and positive definite (PAPADRAKAKIS; GANTES, 1988; GREENBAUM, 1997; WHITELEY, 2017; DVORNIK; LAZAREVIĆ, 2017).

We perform in this paper a numerical analysis comparing the computational efficiency of direct and iterative methods algorithms for the solution of linear equations systems, generated at each iteration of the FEM formulation in the incremental process, from static problems of spatial trusses with geometric nonlinear behavior.

In the discretization of these structures, we use the Finite Element Positional formulation (CODA, 2003), which is based on the principle of minimum potential energy, and it is classified as a Total Lagrangian Formulation. The fundamental unknowns of the problem are the nodal position of the finite element instead of the displacements, which are the unknowns in standard Finite Element Method for solids. The material constituting the bars has a linear-elastic constitutive relation. The structural problem, described by a nonlinear equations system, is solved by the Newton-Raphson method (NR) associated to the Linear Arc-Length path-following technique.

Numerical simulations were carried out in Matlab environment, and were implemented direct methods (LU Factorization and Gauss Elimination Method – GEM) and iterative methods (Conjugated Gradient – CG; Preconditioned Conjugated Gradient – PCG; and Stabilized Bi-Conjugate Gradient – Bi-CGSTAB). The algorithms are compared according to the following parameters: CPU time (in seconds); and, for CG, PCG and BI-CGSTAB, the average number of iterations necessary to solve the linear systems for each iteration of NR method (k_{av}). All methods provide the same displacement at a nodal point of the structure, and the solution of nonlinear problems is achieved with the same number of load steps and accumu-

lated iterations.

Time complexity of algorithms

The complexity study is a set of important mathematical calculations used for developing and analyzing algorithms. This study consists of counting the number of operations in an algorithm, such as additions, subtractions, multiplications, divisions, and comparisons. It allows us to compare, for the same problem, the computational cost of different algorithms and to identify their characteristics. Using this approach, we can choose a numerical method that is more practical and computationally less costly (SEDGEWICK; FLAJOLET, 2013). The addition and subtraction of real numbers are approximately equally costly, as well as multiplication and division. Therefore, in the study of the numerical methods of this text, we do the counting of the operations in these pairs.

To calculate the complexity of an algorithm, we write the computational cost of the operations as a function of n, which represents the dimension of the algorithm entry object. For our study specifically, n represents the dimension of the linear system we want to solve. Small values of n represent problems that are easy to solve by different methods and do not show a relevant difference in processing time or computational cost. Therefore, we direct time complexity to problems of large order. They represent a more critical and delicate situation, since in some cases the computational cost grows exponentially on n. This could make the problem intractable from the computational point of view. In this context, the concept of limit as n approaches infinity is an important aspect in the area, evidencing the importance of asymptotic calculations in the complexity study of algorithms (KOZEN, 2012).

The numerical algorithms may take different processing time, even for the same algorithm running multiple times on the same computer. This is due to different factors that arise from the computer itself, which can vary continuously, varying the time as well. For this reason, the complexity study is useful in the comparison of numerical methods, providing theoretical support to numerical evidence. It explains why some methods are faster than others are (SEDGEWICK; FLAJOLET, 2013).

The complexity is of great importance to determine which routine of a numerical method has the higher computational cost. By identifying this routine, it is possible to focus in this part of the algorithm, since a modification in it will result in a more effective improvement of the whole algorithm. We can also modify the most costly routine specifically for the input system, reducing unnecessary operations and increasing the algorithm speed. In some special cases, as a sparse matrix, there are many unnecessary operations that we can avoid and so optimize the algorithm. From the analysis of complexity and theoretical comparison, we can analyze whether the theoretical behavior matches the practical behavior to culminate with the choice of the best method.

Numerical methods for solution of linear system

Let us consider the linear equations set:

$$\mathbf{Ax} = \mathbf{b}, \tag{1}$$

where $\mathbf{A}$ is a symmetric positive definite sparse matrix arising when the Finite Element Method is applied to problems of structural mechanics. The direct methods and iterative methods can be applied to solve this system. The direct methods find the exact solution of the problem, disregarding rounding errors. The ones of most common use are the Gaussian Elimination and LU Factorization.

Gaussian Elimination method (GEM) is a systematic application of elementary row operations to a system of linear equations in order to convert the system to upper triangular form. Once the coefficient matrix is in upper triangular form, substitution is used to find a solution (MON; KYI, 2014). We use this algorithm frequently to solve linear systems since it can solve any system on the form equation (1) such that $det(\mathbf{A}) \neq 0$. However, this method becomes slower for the systems obtained from structural problems, since these are of great order and due to the row-by-row matrix operations.

The LU Factorization (LU) consists, in turn, of the factorization of the matrix $\mathbf{A}$ in two different matrices $\mathbf{L}$ and $\mathbf{U}$. These matrices are lower and upper triangular, respectively. This factorization occurs in such a way that the product from $\mathbf{L}$ and $\mathbf{U}$ equals the matrix $\mathbf{A} : \mathbf{A} = \mathbf{LU}$. As the resulting system and the initial system are mutually dependent, their solution is equivalent. Since $\mathbf{Ax} = \mathbf{b}$, by doing $\mathbf{y} = \mathbf{Ux}$ and $\mathbf{Ly} = \mathbf{b}$, we return to equality $\mathbf{A} = \mathbf{LU}$. An advantage of LU is that the same factorization can be used for different vectors $\mathbf{b}$ (BANDARA; RANASINGHE, 2011).

GEM and LU algorithms have similar computational cost with respect to the number of operation pairs. In both cases, if $\mathbf{A}$ is an $n \times n$ matrix, the number of multiplications and divisions is given by $n^3/3 + f(n)$, where $f(n)$ is a quadratic polynomial in n; the number of additions and subtractions is also given by $n^3/3 + f(n)$. For large

values of n the higher order terms stand out, hence both algorithms have approximately the same computational cost. Indeed, if $f(n)$ and $g(n)$ are quadratic polynomials, $f(n)$ for GEM and $g(n)$ for LU, then:

$$\lim_{n\to\infty} \frac{n^3/3+f(n)}{n^3/3+g(n)} = 1, \tag{2}$$

justifying the same approximate computational cost for large values of n.

Iterative methods are extremely important for linear algebra, since direct methods have complexity close to $O(n^3)$, and it is extremely expensive to solve them with very large values of n. Iterative methods require less memory and have lower computational cost compared to direct methods in large problems (SAAD, 2003).

The Conjugate Gradient method (CG) attempts to obtain an approximation of the vector $\mathbf{x}$ in $\mathbf{Ax} = \mathbf{b}$. The updates of the vector $\mathbf{x}^{(k-1)}$ to $\mathbf{x}^{(k)}$ are made through the equation (RODRIGUEZ, 2013; GOŚCIK, 2008):

$$\mathbf{x}^{(k)} = \mathbf{x}^{(k-1)} + t^{(k)}\mathbf{v}^{(k)}, \tag{3}$$

until finding a satisfactory approximation for it. The parameter $t^{(k)}$ and the search direction $\mathbf{v}^{(k)}$ are chosen using the fact that a vector $\mathbf{x}^*$ is a solution of a linear system $\mathbf{Ax} = \mathbf{b}$ if and only if it minimizes:

$$\mathbf{g(x)} = \langle \mathbf{x}, \mathbf{Ax} \rangle - 2\langle \mathbf{x}, \mathbf{b} \rangle, \tag{4}$$

where the notation $\langle \mathbf{a}, \mathbf{b} \rangle$ indicates the inner product between $\mathbf{a}$ and $\mathbf{b}$. Through the direction vector and with $\mathbf{v} \neq \mathbf{0}$, the CG method defines a search direction than decreases the values of g in each iteration: $g(\mathbf{x}^{(k+1)}) < g(\mathbf{x}^{(k)})$. For ill conditioned matrices, the results of the CG can diverge very easily, due to rounding errors that appear from these matrices. In these cases, we can multiply the matrix $\mathbf{A}$ by another matrix to reduce rounding errors, but in a way that we can find the original solution easily. More precisely, let $\mathbf{C}$ be a $n \times n$ matrix and let:

$$\tilde{\mathbf{A}} = \mathbf{C}^{-1}\mathbf{A}\left(\mathbf{C}^{-1}\right)^T, \tag{5}$$

where $\mathbf{C}^{-1}$ is the inverse of $\mathbf{C}$ and $\mathbf{C}^T$ is the transposed of $\mathbf{C}$. With equation (5) we find an equivalent new linear system:

$$\tilde{\mathbf{A}}\tilde{\mathbf{x}} = \tilde{\mathbf{b}}, \tag{6}$$

where $\tilde{\mathbf{x}} = \mathbf{C}^T\mathbf{x}$ and $\tilde{\mathbf{b}} = \mathbf{C}^{-1}\mathbf{b}$. Thus, if we choose $\mathbf{C}^{-1}$ so that equation (6) is a better conditioned system, we can find its solution and then find the solution of the original system, only by multiplying $\tilde{\mathbf{x}}$ by matrix $\left(\mathbf{C}^{-1}\right)^T$. By doing so, we are preconditioning the matrix and this modification shall not change its symmetry, even after applying the preconditioner (FORTES, 2008). We can solve the new system by the Conjugate Gradient method; however, we shall call it now the Preconditioned Conjugate Gradient method (PCG), since we apply the preconditioner to improve the matrix and optimize the solution (ZHANG; ZHANG, 2013).

The Stabilized Bi-Conjugate Gradient method (Bi-CGSTAB) (VORST, 1992) is an iterative method to solve linear systems that does not require any hypothesis on the matrix $\mathbf{A}$, other than $det(\mathbf{A}) \neq 0$ (FORTES, 2008; HAJARIAN, 2013). We obtain this method through the Bi-Conjugate Gradient method and it does not use the conjugated matrix (MOTTA, 2010).

We can calculate the time complexity of iterative methods similarly to direct methods. We count the operations for a single iteration and then multiply by the number k of iterations that the algorithm performs. We recall that n denotes the number of columns in $\mathbf{A}$. The number of multiplications and divisions in the Conjugated Gradient method, for example, is:

$$(k+1)(n^2+f(n)), \tag{7}$$

where $f(n)$ is a linear polynomial in n. We can combine this expression with the one from the direct methods to determine a condition involving k and n so that the Conjugate Gradient method converges faster than a direct method.

So, ignoring the terms of lower degree, we obtain:

$$\begin{aligned} (k+1)n^2 &< n^3/3, \\ k+1 &< n/3, \\ k &< n/3-1. \end{aligned} \tag{8}$$

Usually, for the linear systems in structural engineering, the number of iterations k is much smaller than the matrix dimension n. For better-conditioned matrices, we expect to find a good approximation in about $\sqrt{n}$ steps. For large values of n, $\sqrt{n}$ is much smaller than the quantity $n/3-1$ given in equation (8), as evidenced by the equation:

$$\lim_{n\to\infty} \frac{\sqrt{n}}{n/3-1} = 0. \tag{9}$$

With these concepts, we can understand clearly why the iterative methods are more effective for large values of n.

Formulation of the space truss finite element

This section describes an element in the truss structure using the Finite Element Positional formulation (CODA, 2003). Such an element transmits only axial load and has a constant cross-sectional area A. The coordinates (X_1, Y_1, Z_1) and (X_2, Y_2, Z_2) represent the initial configuration of the bar element (also known as reference coordinates). After a configuration change due to truss displacements, the bar will have new coordinates (x_1, y_1, z_1) and (x_2, y_2, z_2) (LACERDA; MACIEL; SCUDELARI, 2014). The initial (or reference) length L_0 and the current length of the bar L are calculated, respectively, by:

$$L_0 = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2 + (Z_2 - Z_1)^2}, \quad (10)$$

$$L = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}. \quad (11)$$

The tangent stiffness matrix $\mathbf{K}_{el}$ and the elementary internal forces vector $\mathbf{F}_{el}$ are given by:

$$\mathbf{K}_{el} = \frac{EA}{L_0^3}\mathbf{B} + \frac{EA \cdot \varepsilon_G}{L_0}\mathbf{C}, \quad (12)$$

$$\mathbf{F}_{el} = \frac{EA \cdot \varepsilon_G}{L_0}\mathbf{d}, \quad (13)$$

where EA is the axial stiffness and the Green strain ε_G is the given by:

$$\varepsilon_G = \frac{L^2 - {L_0}^2}{2{L_0}^2}. \quad (14)$$

In equations (12) and (13), the matrices $\mathbf{B}$ and $\mathbf{C}$ are defined as follows:

$$\mathbf{B} = \begin{bmatrix} \mathbf{I}_3 & -\mathbf{I}_3 \\ -\mathbf{I}_3 & \mathbf{I}_3 \end{bmatrix}, \quad (15)$$

$$\mathbf{C} = \mathbf{d}\mathbf{d}^T, \quad (16)$$

where $\mathbf{d} = [x_1 - x_2, y_1 - y_2, z_1 - z_2, x_2 - x_1, y_2 - y_1, z_2 - z_1]^T$ and $\mathbf{I}_3$ is the identity matrix of order 3.

Method for the solution of nonlinear problems

The basic problem of nonlinear analysis is to find the equilibrium configuration of a structure that is under the action of an applied load. In this section we present the formulation and the algorithm for Newton-Raphson method adapted to the nonlinear structural problem.

Nonlinear structural problem

The nonlinear equations system that governs the static equilibrium of a structural system can be written by:

$$\mathbf{g} = \lambda \mathbf{F}_r - \mathbf{F}_{int}(\mathbf{d}) = \mathbf{0}, \quad (17)$$

where $\mathbf{g}$ represents the unbalanced forces vector, $\mathbf{F}_r$ is a reference vector characterizing the external load direction, λ is the load parameter and $\mathbf{F}_{int}$ is the internal forces vector, which is a function of the nodal coordinates vector $\mathbf{d}$.

The Newton-Raphson method has been one of the most utilized in the solution of equation (17). This method only provides the solution of one point in the equilibrium path. For other points, the iterations of this method are combined with an iterative and incremental procedure. The procedure is given as follows (BATHE, 2006; SOUZA et al., 2017):

$$\mathbf{K}\left(\mathbf{d}^{(j-1)}\right)\delta\mathbf{d}^{(j)} = \delta\lambda^{(j)}\mathbf{F}_r - \mathbf{g}^{(j-1)}, \quad (18)$$

$$\mathbf{d}^{(j)} = \mathbf{d}^{(j-1)} + \delta\mathbf{d}^{(j)}, \quad (19)$$

with $j = 1, 2, ...$, where $\mathbf{K}$ is the tangent stiffness matrix, $\delta\mathbf{d}^{(j)}$ is the sub-incremental nodal coordinates vector and $\delta\lambda^{(j)}$ is the load sub-increment. Note that the right superscript j is used herein to refer to the current iteration and $(j-1)$ previous iteration. The load total parameter $\lambda^{(j)}$ is update by:

$$\lambda^{(j)} = \lambda^{(j-1)} + \delta\lambda^{(j)}. \quad (20)$$

Linear Arc-Length method

The methodology for the solution of nonlinear structural problems must be able to trace the complete equilibrium path, identifying and computing the limit points. For this purpose, an incremental-iterative process is used and it consists of two steps (LEON et al., 2011):

1) From the last equilibrium configuration of the structure, an increment of load is selected (defined as initial load sub-increment - $\delta\lambda^{(0)}$), aiming to satisfy some constraint equation imposed on the problem. After selecting this parameter, the initial increment of nodal coordinates is determined $\delta\mathbf{d}^{(0)}$; and

2) The second solution step seeks, by means of a path-following strategy, to correct the incremental solution, initially proposed in the previous step, in order to restore the structure equilibrium. If the iterations involve nodal

coordinates ($\mathbf{d}$) and the load parameter (λ), then an additional constraint equation is required. The format of this equation is what distinguishes the various iteration strategies.

In the Linear Arc-Length method (RIKS, 1972; RIKS, 1979), the iteration path is always kept orthogonal to the initial tangent at each step. The expression for the initial load sub-increment (predicted solution) is given by:

$$\delta\lambda^{(0)} = \frac{\Delta l}{\|^t\delta\mathbf{d}_r\|}, \tag{21}$$

where Δl represents the arc-length increment and ${}^t\delta\mathbf{d}_r$ is the part of $\delta\mathbf{d}^{(j)}$ related to the vector $\mathbf{F}_r$ in the previous time step. The arc-length Δl can be used as the control parameter in the current time step as follows (CRISFIELD, 1991):

$$\Delta l = {}^t\Delta l \left(\frac{Nd}{{}^t j}\right)^{(1/2)}, \tag{22}$$

where ${}^t\Delta l$ represents the arc-length increment in the previous time step, Nd is the number of desired iterations for the convergence of the current iterative process, and ${}^t j$ is the number of iterations that were required to converge in the previous time step. In the subsequent iterative process, the constraint equation used to calculate $\delta\lambda^{(j)}$ is obtained making the iterative solution $(\delta\mathbf{d}^{(j)}, \delta\lambda^{(j)}\mathbf{F}_r)$ results orthogonal to the predicted incremental solution $(\Delta\mathbf{d}^{(0)}, \Delta\lambda^{(0)}\mathbf{F}_r)$. More precisely, we have:

$$\delta\mathbf{d}^{(j)^T}\Delta\mathbf{d}^{(0)} + \delta\lambda^{(j)}\Delta\lambda^{(0)}\mathbf{F}_r{}^T\mathbf{F}_r = 0. \tag{23}$$

Assuming that the inverse of the matrix $\mathbf{K}$ exists, the equation (18) can be rewritten by isolating the vector $\delta\mathbf{d}^{(j)}$:

$$\delta\mathbf{d}^{(j)} = \delta\lambda^{(j)}\mathbf{K}\left(\mathbf{d}^{(j-1)}\right)^{-1}\mathbf{F}_r - \mathbf{K}\left(\mathbf{d}^{(j-1)}\right)^{-1}\mathbf{g}^{(j-1)}. \tag{24}$$

From equations (23) and (24), an expression is obtained for the determination of the load sub-increment parameter $\delta\lambda^{(j)}$ $(j > 1)$:

$$\delta\lambda^{(j)} = -\frac{\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_g^{(j)}}{\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_r^{(j)} + \Delta\lambda^{(0)}\mathbf{F}_r^T\mathbf{F}_r}. \tag{25}$$

The vectors $\delta\mathbf{d}_g^{(j)}$ and $\delta\mathbf{d}_r^{(j)}$ are obtained by, respectively:

$$\delta\mathbf{d}_g^{(j)} = \mathbf{K}\left(\mathbf{d}^{(j-1)}\right)^{-1}\mathbf{g}^{(j-1)}, \tag{26}$$

$$\delta\mathbf{d}_r^{(j)} = \mathbf{K}\left(\mathbf{d}^{(j-1)}\right)^{-1}\mathbf{F}_r, \tag{27}$$

where $\mathbf{g}^{(j-1)} = \lambda^{(j-1)}\mathbf{F}_r - \mathbf{F}_{int}\left(\mathbf{d}^{(j-1)}\right)$. Neglecting the second term in the denominator of equation (25), namely, $\mathbf{F}_r^T\mathbf{F}_r = 0$, the parameter $\delta\lambda^{(j)}$ results:

$$\delta\lambda^{(j)} = -\frac{\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_g^{(j)}}{\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_r^{(j)}}. \tag{28}$$

The correct choice of the signal is so important in definition of the sequences of solutions $(\mathbf{d}, \lambda)$ that allow continuous advancement in load-displacement response. The displacements vector $(\mathbf{u})$ is determined as follows:

$$\mathbf{u}^{(j)} = \mathbf{d}^{(j)} - {}^0\mathbf{d}, \tag{29}$$

where ${}^0\mathbf{d}$ is the nodal coordinate vector at time 0 (undeformed structure). The signal of the initial load increment $(\Delta\lambda^{(0)})$ can be positive or negative. The procedure used in this paper consists of the analysis of the internal product between the sub-incremental nodal coordinates obtained in the previous load step $({}^t\Delta\mathbf{d})$ and the initial nodal coordinates increment $(\delta\mathbf{d}_r^{(0)})$. If ${}^t\Delta\mathbf{d}^T\delta\mathbf{d}_r^{(0)} < 0$, then $\Delta\lambda^{(0)} \leftarrow -\Delta\lambda^{(0)}$ and the predictor $\Delta\mathbf{d}^{(0)}$ will have opposite direction of $\delta\mathbf{d}_r^{(0)}$; otherwise, the predictor will have the same direction. The convergence criterion is determined by the residual load norm and the reference load norm applied by:

$$\|\mathbf{g}^{(j)}\| = tol \cdot \|\mathbf{F}_r\|, \tag{30}$$

where tol is the tolerance provided by the user. The algorithm for the standard Newton-Raphson method associated with Linear Arc-Length Method (RIKS, 1972; RIKS, 1979; WEMPNER, 1971) is shown below. The parameters considered in the algorithm input are: initial arc length $(\Delta l^{(0)})$; maximum number of iterations in each step load (j_{max}); required number of iterations (Nd); tolerance (tol); load increment (ΔP); and maximum number of load steps (ls_{max}). The algorithm outputs are: nodal coordinate vector $(\mathbf{d})$; total load parameter λ; total number of load steps (ls); total number of iterations (j); and nodal displacement vector $(\mathbf{u})$.

Numerical simulation of spatial trusses

In this section, we apply the direct (GEM and LU) and iterative (CG, PCG and Bi-CGSTAB) methods in space trusses with geometric nonlinearity, analyze their performance and compare them to find the most effective one for the problem. The algorithms are implemented with the Matlab software version 8.6 R2015b (MATHWORKS, 2015). The computational tests were performed in a computer Core i7 with 8 GB of memory.

Algorithm - Newton-Raphson Method associated with the Linear Arc-Length Method (Riks-Wempner algorithm)

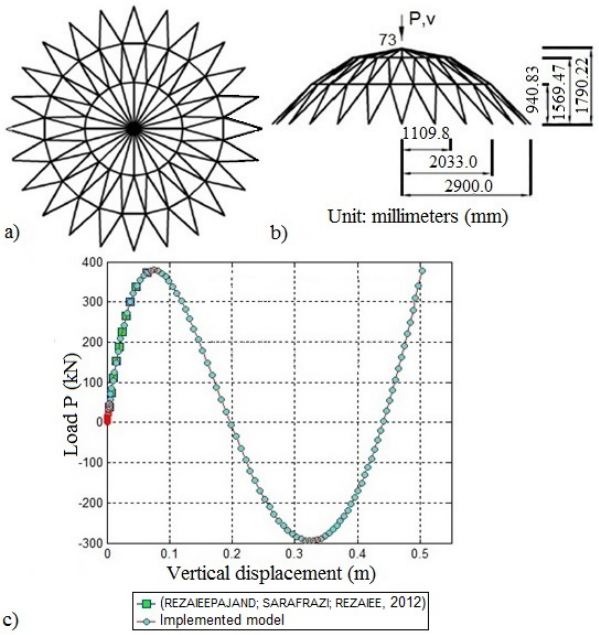
Input: $\mathbf{d}^{(0)}, ls_{max}, j_{max}, Nd, \Delta l^{(0)}, tol, \Delta P$
1. $\Delta\mathbf{d} \leftarrow \mathbf{0}$, $\lambda \leftarrow 0$
2. For $ls \leftarrow 1, ..., ls_{max}$ do
3. $\quad \delta\mathbf{d}_r \leftarrow \mathbf{K}(\mathbf{d})^{-1}\mathbf{F}_r$
4. $\quad \Delta\lambda \leftarrow \Delta l / \|\delta\mathbf{d}_r\|$
5. $\quad$ If $\Delta\mathbf{d}^T \delta\mathbf{d}_r < 0$ then
6. $\quad\quad \Delta\lambda \leftarrow -\Delta\lambda$
7. $\quad$ End-if
8. $\quad \Delta\mathbf{d}^{(0)} \leftarrow \Delta\lambda\,\delta\mathbf{d}_r$
9. $\quad \Delta\mathbf{d} \leftarrow \Delta\mathbf{d}^{(0)}$
10. $\quad \mathbf{g} \leftarrow (\lambda + \Delta\lambda)\mathbf{F}_r - \mathbf{F}_{int}(\mathbf{d} + \Delta\mathbf{d})$
11. $\quad$ For $j \leftarrow 1, ..., j_{max}$ do
12. $\quad\quad \delta\mathbf{d}_g \leftarrow \mathbf{K}(\mathbf{d} + \Delta\mathbf{d})^{-1}\mathbf{g}$
13. $\quad\quad \delta\lambda \leftarrow -\left(\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_g\right) / \left(\Delta\mathbf{d}^{(0)^T}\delta\mathbf{d}_r\right)$
14. $\quad\quad \delta\mathbf{d} \leftarrow \delta\mathbf{d}_g + \delta\lambda\,\delta\mathbf{d}_r$
15. $\quad\quad \Delta\mathbf{d} \leftarrow \Delta\mathbf{d} + \delta\mathbf{d}$
16. $\quad\quad \Delta\lambda \leftarrow \Delta\lambda + \delta\lambda$
17. $\quad\quad \mathbf{g} \leftarrow (\lambda + \Delta\lambda)\mathbf{F}_r - \mathbf{F}_{int}(\mathbf{d} + \Delta\mathbf{d})$
18. $\quad\quad$ If $\|\mathbf{g}\| \leq tol \cdot \|\mathbf{F}_r\|$ then
19. $\quad\quad\quad$ Break
20. $\quad\quad$ End-if
21. $\quad\quad \delta\mathbf{d}_r \leftarrow \mathbf{K}(\mathbf{d} + \Delta\mathbf{d})^{-1}\mathbf{F}_r$
22. $\quad$ End-for
23. $\quad \mathbf{d} \leftarrow \mathbf{d} + \Delta\mathbf{d}$
24. $\quad \lambda \leftarrow \lambda + \Delta\lambda$
25. $\quad \Delta l \leftarrow \Delta l (Nd / {}^{t} j)^{0.5}$
26. End-for
27. $\mathbf{u} \leftarrow \mathbf{d} - {}^{0}\mathbf{d}$

Output: $\mathbf{u}, \mathbf{d}, \lambda, ls, j$

Circular truss dome

Consider the circular truss dome illustrated in Figure 2a-b, constructed in steel with elasticity modulus of 206.0GPa. The transversal section area for all bars is $2.16x10^{-4}m^2$. At the base of the truss, there are pinned supports and we applied a vertical load P at its apex. This structure has 73 nodes, 168 elements and 219 degrees of freedom. The numeric results (k_{av} and t) obtained from the implementation of the methods are presented in Table 1. The parameters considered in the simulations are: $\Delta l = 0.01$; $j_{max} = 100$; $Nd = 3$; $tol = 1.0x10^{-6}$; and $\Delta P = 37.5kN$. The graph, which relates the vertical displacement in the apex and the load obtained with the algorithm, is shown in Figure 2c with two load limit points. We compare our graph with the numerical results in (REZAIEEPAJAND; SARAFRAZI; REZAIEE, 2012). The simulations performed with the codes developed in this paper were conducted beyond the equilibrium points obtained by these authors. We applied to this structure 105 load steps and obtained a vertical deflection of $-502.7091mm$.

Figure 2: The circular truss dome: a) superior view; b) lateral view; and c) equilibrium path.



Source: The Authors.

Table 1: Numeric results from the circular truss dome (number of variables: n = 219).

Methods	k_{av}	$t(s)$
LU	-	3.96
GEM	-	5.56
CG	30.7	2.70
PCG	38.7	3.31
Bi-CGSTAB	27.1	2.75

Source: The Authors.

Space trusses with pyramidal structures

Figure 3 shows the schematic drawing of one square-on-square truss module, with pyramidal structures measuring $2.5m \times 2.5m$ and height of $1.5m$. This truss is subjected to concentrated load of intensity P at the top. The bars have tubular cross-section $\Phi 76x2$ for the bottom bars and $\Phi 60x2$ for the remaining bars, and all bars have elasticity modulus $E = 200.0GPa$. In the analysis, three structures were considered by varying the number of modules in the directions x and y:

(a) 1 module in the axis x and 1 module in the axis y (61 nodes, 200 elements and 183 degrees of freedom);

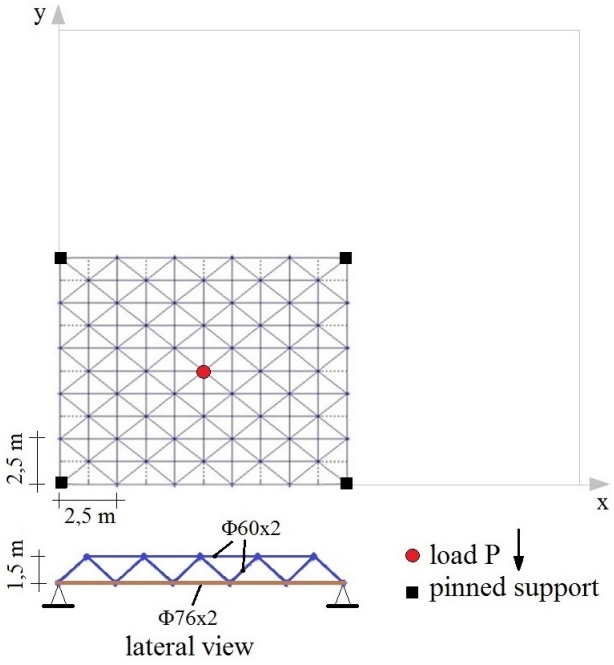
(b) 2 modules in the axis x and 2 modules in the axis y (221 nodes, 800 elements and 663 degrees of freedom); and

(c) 3 modules in the axis x and 3 modules in the axis y (481 nodes, 1800 elements and 1443 degrees of freedom).

As described above, we consider structures of in-

creasing size in (a), (b) and (c). This generates increasing linear systems, which allow us to verify the performance of the methods in problems of different dimension. We consider for the simulations the following parameters: $\Delta l = 0.5$; $j_{max} = 100$; $Nd = 3$; $tol = 1.0x10^{-6}$; and $\Delta P = 0.01N$. The numerical results (k_{av} and t) are presented in Table 2.

Figure 3: Representation of one truss module with pyramidal structures.



Source: The Authors.

Table 2: Numerical results for the spatial trusses (number of variables: $n = 183$ (1x1); $n = 663$ (2x2); $n = 1443$ (3x3)).

Methods	Modules					
	1x1		2x2		3x3	
	k_{av}	$t(s)$	k_{av}	$t(s)$	k_{av}	$t(s)$
LU	-	0.46	-	13.74	-	333.02
GEM	-	0.61	-	29.07	-	716.65
CG	27.9	0.30	60.9	1.68	86.1	10.01
PCG	25.3	0.34	52.3	2.63	73.9	20.89
Bi-CGSTAB	24.0	0.31	46.9	1.87	65.1	13.56

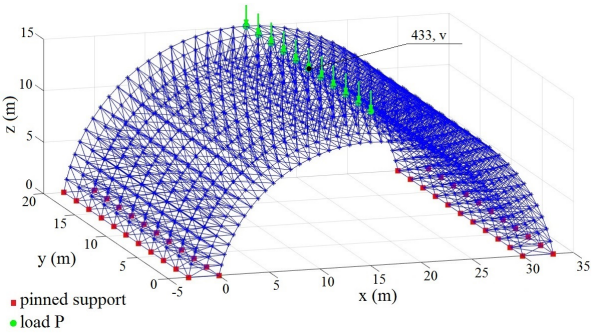
Source: The Authors.

Arc-shaped space truss

Figure 4 shows an arc-shaped space truss with 902 nodes and 3801 bars. This structure is analyzed under loads P applied vertically (in the z direction) at its top, and at its base ($z = 0$) the displacements are restricted. The axial stiffness (EA) of its members is $1.0 \times 10^4 kPa$. The parameters adopted in the analyzes are: $\Delta l = 5.0$; $j_{max} = 100$; $Nd = 3$; $tol = 1.0x10^{-6}$; and $\Delta P = 100kN$.

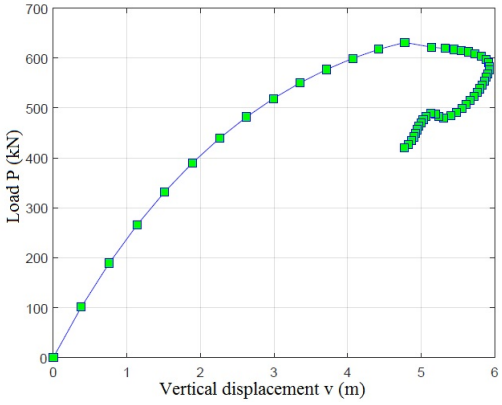
The numerical results (k_{av} and t) obtained from the analysis with the PCG and CG methods are shown in Table 3. Figure 5 shows the equilibrium path (curve deflection in the direction z versus load at node 433). The incremental-iterative process with the Newton-Raphson method was completed with 50 load steps and 165 iterations. Figure 6 illustrates the arc-shaped space truss deflections and indicates the compression (red color) and traction (black color) in the bars .

Figure 4: Structural model of the arc-shaped space truss.



Source: The Authors.

Figure 5: The load–deflection curve at top of the arc-shaped space truss.



Source: The Authors.

Table 3: Number of iterations and processing time (number of variables: $n = 2706$).

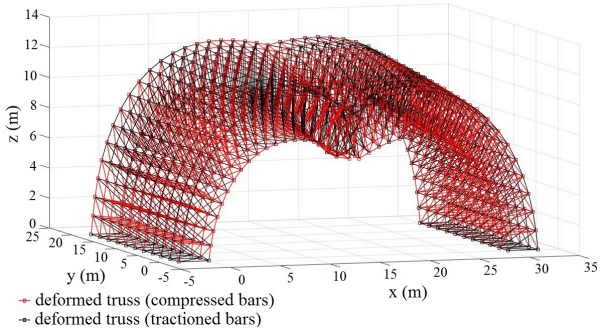
Methods	k_{av}	$t(s)$
CG	387.8	314.67
PCG	299.3	827.46

Source: The Authors.

Analysis of the numerical results

For large values of n, the iterative methods (CG, PCG and BI-CGSTAB) performed much better than the direct

Figure 6: Deflections of the arc-shaped space truss.



Source: The Authors.

methods (LU and GEM), as shown in Tables 1 and 2. Despite the first simulation in Table 2 (1x1 module) showing us some difference in the processing time between the direct and iterative methods, it is in the second (2x2 modules) and third (3x3 modules) simulations that this difference becomes evident, due to the higher value of n. The iterative methods require less memory and, as our simulations suggest, have lower computational cost in large dimension problems.

The efficiency of the iterative methods observed in Table 2 is yet supported by the time complexity of the algorithms. See for example the third problem in Table 2 (3x3 modules), where the size n of the matrix is given by the product between the number of nodes and the freedom degree: $n = 1443$; the CG reached the solution with a mean number of iterations equal to 86.1. Since equation (8) determines that for this problem $k < 480$, the average number of iterations satisfies this condition and supports our results. The numerical results of the CG method present the shortest processing time (CPU time) in Tables 1 and 2. Despite the fact that the BI-CGSTAB method requires fewer average iterations (k), the processing time was higher. Hence, the reduction in the iteration number does not compensate the higher computational cost in the BI-CGSTAB method, when compared to CG. Indeed, the number of multiplications and divisions in the BI-CGSTAB method is:

$$k_2(2n^2 + h(n)), \tag{31}$$

where k_2 is the number of iterations in the execution and $h(n)$ a linear polynomial in n. Let k_1 be the number of iterations in the execution of the CG method. Equations (7) and (31) suggest that CG is more efficient than BI-CGSTAB if:

$$(k_1 + 1)(n^2 + f(n)) < k_2(2n^2 + h(n)). \tag{32}$$

Ignoring the terms of lower degree, we obtain:

$$\begin{aligned} (k_1 + 1)n^2 &< 2k_2n^2, \\ k_1 &< 2k_2 - 1. \end{aligned} \tag{33}$$

Table 2 shows that this condition is verified in our experiments. Once again, we observe the time complexity of the algorithms providing theoretical support for our numerical results. We can accelerate even more the convergence of the CG method by using preconditioners. In the simulation with the PCG, we use the diagonal preconditioner or Jacobi's preconditioner, which is constructed with the elements of the main diagonal of the stiffness matrix **K** (ALMEIDA; PAIVA, 1999):

$$M_{ij} = \begin{cases} K_{ij}, \ if \ i = j \\ 0, \quad if \ i \neq j \end{cases}. \tag{34}$$

In the second problem (Table 2), the solution with the PCG was reached with less iterations when compared to the CG, but with higher CPU time. We highlight the low cost in the construction of the diagonal preconditioner, once there are only used the elements of the main diagonal of the stiffness matrix. It is important to emphasize that the construction of a preconditioner can be expensive. A careful implementation shall be realized to obtain competitive numerical results, both in terms of speed and robustness.

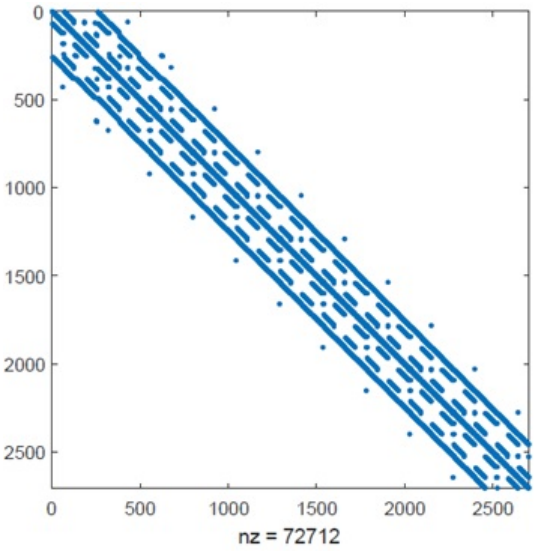
The problem of the spatial trusses in Figures 2 and 3 are enough to reach the conclusion that the iterative methods perform far better than the direct methods (Tables 1 and 2). Hence only iterative methods were used in the solution of the problem of the spatial truss in Figure 4. It is interesting to note that in this problem we also have better performance from the CG method. One can use the pre-conditioned version of the Bi-CGSTAB method in order to improve the matrix conditioning, increasing the convergence rate.

Figure 7 shows the structure of the stiffness matrix **K** (obtained with the "spy" function of Matlab) for the arc-shaped space truss problem. Note that this matrix is sparse of the band type, whose non-zero elements appear on the main diagonal and on parallel diagonals, with a sparsity of 99.007 %. Large matrices require a large storage space and, even if there are computers with the greatest capacity memory currently, it is usually not enough to store square matrix. In order to make the storage of the matrix and the operations on it less computationally costly, we can use techniques based on the storage of non-zero values, such as CSC (Compressed Sparse Column) and CSR (Compressed Sparse Row) (FILHO; XAVIER, 2015).

In the program developed we used the "sparse" function, which stores the non-null elements of the original matrix, disregarding the elements equal to zero.

The main attraction in the use of iterative methods for the solution of sparse linear equations system is that these procedures perform on the matrix **K** only two operations: multiplication of **K** and $\mathbf{K}^T$ by a vector. The CG method seeks the solution of the system by minimizing the error from the solution found in each iteration k.

Figure 7: Structure of the stiffness matrix **K** with nz = 72712 non-zero elements.



Source: The Authors.

Final considerations

The increasing simulation of complex structural models through the Finite Elements Method requires handling large amounts of data. As well as attempting to decrease the response time for solving the structural problem, handling these data is fundamental in the method, although with today's computers, this remains a challenge.

Solving the generated linear systems for each iteration is, usually, the step that requires greater computational effort during the nonlinear simulations of structural problems. If the linear systems in hand are small, then one may use whatever method one feels more comfortable with. However, if one is handling larger matrices, then iterative methods appear to be more efficient from the perspective of processing time. In particular, the time complexity analysis of the algorithms and our numerical simulations suggest that the CG method represents a robust and efficient method.

The stiffness matrix in a structural problem is characterized by a high sparsity index. It is possible to obtain in this case a better performance of the methods through algorithms that can store the nonzero coefficients in the matrix and make operations between matrices and vectors with these coefficients. This avoids the redundant calculations with null elements.

We indicate as a possibility of future work the identification of the sparsity pattern of the matrices generated in the solution of the problems treated in this work. In addition, we consider an interesting problem to study trusses with physical nonlinearity and to adapt the implemented code for studies in dynamic analysis. We also suggest the implementation of other preconditioners, for example, LU incomplete factorization or Cholesky incomplete factorization.

Acknowledgements

The authors thank the UTFPR for the support to carry out this research.

References

ALMEIDA, V. S.; PAIVA, J. B. Aplicação do método dos gradientes conjugados com o uso de pré-condicionadores em problemas do mef. In: XX IBERIAN LATIN-AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING, SÃO PAULO, 1999, São Paulo. *Anais...* São Paulo, 1999.

BANDARA, H. M. D. M.; RANASINGHE, D. N. Effective gpu strategies for lu decomposition. In: HIGH PERFORMANCE COMPUTING – HIPC 2005, 12., 2005, Goa. *Proceedings...* India, 2011. Available from internet: <http://www.hipc.org/hipc2011/studsym-papers/1569512927.pdf>. Acess in: June 08, 2018.

BATHE, K. *Finite element procedures*. New Jersey: Prentice-Hall Inc, 2006.

CODA, H. B. *Análise não linear geométrica de sólidos e estruturas: uma formulação posicional baseada no MEF*. Tese (Doutorado) — Texto complementar para concurso de professor titular. Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo, 2003.

CRISFIELD, M. *Non-linear finite element analysis of solids and structures*. New York, John Wiley & Sons: Google Scholar, 1991.

DVORNIK, J.; LAZAREVIĆ, D. Iterated ritz method for solving systems of linear algebraic equations. *Građevinar*, Hrvatski savez građevinskih inženjera, v. 69, n. 7., p. 521–535, 2017. Https://doi.org/10.14256/JCE.2036.2017.

FILHO, A. A.; XAVIER, A. C. C. Solução de sistemas lineares esparsos utilizando cuda: uma comparação de desempenho em sistemas windows e linux. *Revista de Ciências Exatas e Tecnologia*, v. 8, n. 8, p. 181–195, 2015.

FORTES, W. R. *Precondicionadores e solucionadores para resoluçao de sistemas lineares obtidos de simulaçao de enchimento de reservatórios*. Tese (Doutorado) — Dissertaçao de Mestrado, PPG-Eng. Mec. UERJ, 2008.

GOŚCIK, J. Numerical efficiency of iterative solvers for the poisson equation using equation using computer cluster. *Zeszyty Naukowe Politechniki Białostockiej. Informatyka*, p. 39–52, 2008.

GREENBAUM, A. *Iterative methods for solving linear systems*. [S.l.]: Siam, Journal on Applied Mathematics, Philadelphia, 1997. Https://doi.org/10.1137/1.9781611970937.

HAJARIAN, M. Matrix iterative methods for solving the sylvester-transpose and periodic sylvester matrix equations. *Journal of the Franklin Institute*, Philadelphia, v. 350, n. 10, p. 3328–3341, 2013. Https://doi.org/10.1016/j.jfranklin.2013.07.008.

KOZEN, D. C. *The design and analysis of algorithms*. [S.l.]: Springer Science & Business Media, 2012.

LACERDA, E.; MACIEL, D. N.; SCUDELARI, A. C. Geometrically static analysis of trusses using the arc-length method and the positional formulation of finite element method. In: XXXV IBERIAN LATIN-AMERICAN CONGRESS ON COMPUTATIONAL METHODS IN ENGINEERING, 35., 2014, Fortaleza. *Anais...* [S.l.], 2014.

LEON, S. E.; PAULINO, G. H.; PEREIRA, A.; MENEZES, I. F.; LAGES, E. N. A unified library of nonlinear solution schemes. *Applied Mechanics Reviews*, American Society of Mechanical Engineers, v. 64, n. 4, p. 040803, 2011. Https://doi.org/:10.1115/1.4006992.

MATHWORKS. *MATLAB version 8.6.0 (R2015b)*. Natick, Massachusetts: The MathWorks Inc., 2015.

MON, Y.; KYI, L. L. W. Performance comparison of gauss elimination and gauss-jordan elimination. *IJCCER*, v. 2, n. 2, p. 67–71, 2014.

MOTTA, V. S. *Soluçao de sistemas lineares de grande porte com múltiplos lados direitos*. Tese (Doutorado) — Universidade Federal do Rio de Janeiro, 2010.

PAPADRAKAKIS, M.; GANTES, C. Truncated newton methods for nonlinear finite element analysis. *Computers & Structures*, Elsevier, v. 30, n. 3, p. 705–714, 1988.

PERELMUTER, A. V.; FIALKO, S. Y. Problems of computational mechanics relate to finite-element analysis of structural constructions. Begel House Inc., 2005. Available from internet: <https://scadsoft.com/download/211P.pdf>. Acess in: May 24, 2018.

REZAIEE-PAJAND, M.; SARAFRAZI, S. R.; REZAIEE, H. Efficiency of dynamic relaxation methods in nonlinear analysis of truss and frame structures. *Computers & Structures*, Elsevier, v. 112, p. 295–310, 2012. Https://doi.org/10.1016/j.compstruc.2012.08.007.

RIKS, E. The application of newton's method to the problem of elastic stability. *Journal of Applied Mechanics*, American Society of Mechanical Engineers, v. 39, n. 4, p. 1060–1065, 1972. Https://doi.org/10.1115/1.3422829.

RIKS, E. An incremental approach to the solution of snapping and buckling problems. *International journal of solids and structures*, v. 15, n. 7, p. 529–551, 1979.

RODRIGUEZ, N. L. R. *Resoluçao de sistemas de equaçoes lineares de grande porte em clusters multi-GPU utilizando o método do gradiente conjugado em OpenCLTM*. Tese (Doutorado) — PUC-Rio, 2013.

SAAD, Y. *Iterative methods for sparse linear systems*. [S.l.]: siam, 2003.

SEÇER, M. Inelastic and large deformation analyses of plane trusses. *Technology*, v. 12, n. 3, p. 175–184, 2009.

SEDGEWICK, R.; FLAJOLET, P. *An introduction to the analysis of algorithms*. [S.l.]: Pearson Education India, 2013.

SOUZA, A. S. C. d. *Análise teórica e experimental de treliças espaciais*. Tese (Doutorado) — Universidade de São Paulo, 2003.

SOUZA, L. A. F. de; CASTELANI, E. V.; SHIRABAYASHI, W. V. I.; MACHADO, R. D. Métodos iterativos de terceira e quarta ordem associados à técnica de comprimento de arco linear. *Ciência & Engenharia*, Uberlândia, v. 26, n. 1, p. 39–49, 2017.

VORST, H. A. Van der. Bi-cgstab: A fast and smoothly converging variant of bi-cg for the solution of nonsymmetric linear systems. *SIAM Journal on scientific and Statistical Computing*, SIAM, Philadelphia, v. 13, n. 2, p. 631–644, 1992.

WEMPNER, G. A. Discrete approximations related to nonlinear theories of solids. *International Journal of Solids and Structures*, New York, v. 7, n. 11, p. 1581–1599, 1971.

WHITELEY, J. A preconditioner for the finite element computation of incompressible, nonlinear elastic deformations. *Computational Mechanics*, Springer, Berlin, v. 60, n. 4, p. 683–692, 2017. Https://doi.org/10.1007/s00466-017-1430-3.

ZHANG, J.; ZHANG, L. Efficient cuda polynomial preconditioned conjugate gradient solver for finite element computation of elasticity problems. *Mathematical Problems in Engineering*, Hindawi, London, v. 2013, 2013. Http://dx.doi.org/10.1155/2013/398438.

Received on: Jan. 11, 2018
Accepted on: Jun. 19, 2018