2019 16th International Conference on Electrical Engineering, Computing Science and Automatic Control (CCE) Mexico City, Mexico. September 11-13, 2019

A Tool to Solve Nonlinear Algebraic Equations Systems

Gerardo C. Velez-Lopez,
L. Hernandez-Martinez,
G. Diaz-Arango
INAOE Electronics Department,
Puebla, Mexico
Email: {gcvelez,luish,gerardo_diaz}@inaoep.mx

H. Vazquez-Leal
Facultad de Instrumentacion Electronica,
Universidad Veracruzana,
Cto. Gonzalo Aguirre Beltran S/N, 91000
Xalapa, VER, Mexico
Email: hvazquez@uv.mx

***Abstract*—This article explains the development of a tool (application) to solve nonlinear algebraic equations systems (NAES). Systems of nonlinear equations are usually complex and can not be solved analytically or in an exact way, therefore numerical methods are used to solve this type of systems. This paper shows the implementation of four numerical algorithms: conjugate gradient method (CGM), Newton-Raphson (N-R), Ostrowski's method (OM) and homotopic continuum methods (HCM). These methods are capable of solving systems of nonlinear algebraic equations of $n \times n$. In addition, the GiNaC library is used, which is completely written in C++ and allows the user to perform algebraic operations and interact within this language. The application has been programmed considering the following attributes: easy to use, multi-platform, that is, it must be able to operate in different operating systems (S.O.), fast in obtaining the solution, able to configure some fields to try to find the solution.**

***Index Terms*—Nonlinear algebraic equations systems (NAES), conjugate gradient method (CGM), Newton-Raphson (N-R), Ostrowski's method (OM) and homotopic continuum methods (HCM), GiNaC, C++, operating systems (S.O.).**

I. Introduction

Numerous problems of science, engineering or economics require the search for solutions of an equation. For a long time, we have tried to model present problems in nature with expressions that allow us to know a priori how a system will behave [1].

In our case, the tool is used to solve NAES in collision-free trajectories applying in robots, since the method we use is deterministic [12]–[14]. This type of systems exhibit to a greater or lesser degree a non-linear behavior that can be extraordinarily rich in possibilities or trajectories, as a response to the perturbations that it experiences. As a result, several numerical methods have been studied and developed with which you can obtain good approximations of these solutions, more efficiently and with sufficient precision [3], [4].

The numerical methods consist of finding, under certain conditions, through an iterative process, and starting from an initial approximation x_0, a sequence x_n of approximations to the solution of the equation, with the requirement that there is $\lim_{n\to\infty} x_n = \xi$, where ξ the solution of the non-linear equation (roots or zeros of the non-linear equation), under certain error criteria (ε) [4]. Nonlinear equations can be expressed as:

$$F(x) = 0, \tag{1}$$

where, $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$ and $x = \{x_1, x_2, \ldots, x_n\}$ for the multidimensional case ($n \times n$).

At present there are numerous iterative methods to solve the nonlinear equation 1 and each has an order of convergence, which refers to the speed with which a sequence converges to a solution of the corresponding equation [1]–[4]. In this work we use the Ostrowski's method that has fourth order convergence and the Newton method that has second order convergence [5]. There are four classifications of numerical methods to find the roots of 1: local methods, global methods, interval methods and graphical methods [5]. We focus on the first two methods, where the conjugate gradient method (CGM) and the homotopic continuation method (HCM) have global convergence and the Newton-Raphson method (N-R) and the Ostrowski's method (OM) are of local convergence. Currently, with the computing tools available, the analysis of these methods has undergone a revolution at the numerical level, since computers allow the simulation of these methods, but in practical realization, it is necessary to solve several problems such as the computational efficiency based on the time used by the processor and the RAM used, the selection of iterative methods that have a fast convergence to the desired solution, the control of rounding errors and the selection of a starting point (x_0 initial estimate to the root). There are computational tools (software) used to solve problems of nonlinear systems, such as Matlab®, Maple®, Wolfram Mathematica®, etc; but they have a high computational cost and the problem is not always solvable, besides you need to pay a license for use. There are other options like GiNaC, this is a C++ library that enables the user to perform algebraic operations within the C++ language, which allowed the development of our tool (application) responsible for solving systems of nonlinear algebraic equations [6], [7]. By using the C++ language, it offers a number of benefits, such as the flexibility to modify the code, portability to different operating systems, use in embedded systems, interact with various existing libraries, reduce computation times and RAM consumption. In addition, Qt Creator was used, this is a multiplatform IDE, it can be used in Linux, Mac OS, Windows. This paper is organized in the following way: in Section II, the numerical methods used in this work are explained. Section III describes the application developed. A comparison between our application and Maple 2015 is shown in Section IV. Finally, Section V presents the conclusions and future work.

II. Numerical methods implemented

A. *Method of Newton-Raphson (N-R)*

Newton method is also known as Newton-Raphson method. According to Burden and Faires (2011), Newton's method is one of the powerful and well-known numerical methods for solving linear and nonlinear equations [8] and even for solutions of complex variable equations [3]. Next, Newton's method is defined to approximate the solution of a system non-linear equations of real variables:

$$x_{i+1} = x_i - [F'(x_i)]^{-1}F(x_i),\ i = 0,1,2,\dots,k-1, \quad (2)$$

as long as $[F'(x_i)]^{-1} \neq 0$ for all $i \in \mathbb{N}$.

Newton's method has some disadvantages, for example:

- If $[F'(x_i)]^{-1} = 0$ for some $i \in \mathbb{N}$, the method can not continue iterating.
- For some functions, finding its derivative is a tedious job and even depending on the problem, there could be no information about its derivative.
- The convergence of Newton's method can be guaranteed locally, therefore we must choose a x_0 that is sufficiently close to the root x.

In these cases it is advisable to apply a modification to it or apply another approach method.

B. *Ostrowski's method (OM)*

Ostrowski's method was introduced by Alexander Markowich Ostrowski in 1960 to find the roots of a single-variable nonlinear function [5]. Ostrowski's method is an extension of Newton's method. Ostrowski and Grau-Sanchez focused on systems of nonlinear equations [9].The method employs two-step iterations using the following equations [9]:

$$\begin{aligned} y_i &= x_i - A^{-1}F(x_i), \quad i = 0,1,2,\dots,k-1,\\ x_{i+1} &= y_i - A^{-1}O_j(x_i), \quad j = 1,2,\dots,n, \end{aligned} \quad (3)$$

where, A_i is the Jacobian matrix of $\frac{\partial f_j}{\partial x_k}$ evaluated at point x and $O_j(x) = \frac{f_j(x_i)f_j(y_i)}{f_j(x_i)-2f_j(y_i)}$. The additional Ostrowski's funtion can also be written as:

$$O_j(x_i) = \begin{pmatrix} \frac{f_1(x_i)f_1(y_i)}{f_1(x_i)-2f_1(y_i)} \\ \frac{f_2(x_i)f_2(y_i)}{f_2(x_i)-2f_2(y_i)} \\ \vdots \\ \frac{f_n(x_i)f_n(y_i)}{f_n(x_i)-2f_n(y_i)} \end{pmatrix},$$

$$i = 0,1,2,\dots,k-1, \quad j = 1,2,\dots,n. \quad (4)$$

The studies have extended the Ostrowski method to one that has a convergence of sixth order and eighth order [4], [5], but in this work the method with covergency in fourth order was used.

C. *Conjugate Gradient (CGM)*

The gradient method is a descent method in which one begins to iterate at an arbitrary point x_0 and continues following the line of maximum descent of the paraboloid obtaining a succession of points $x_1, x_2, \dots$ until you get a point close enough to the solution x^*. In each iteration the direction is chosen for which f_x decreases more quickly, which is the direction opposite to $\nabla f(x_k)$, that is to say $-\nabla f(x_k)$. Below is a definition that will be useful in the rest of the sections of this work; the error ε is a vector that indicates how far you are from the solution. The residue r_k is a vector that indicates the direction of maximum descent.

The CGM can be seen as a minimization algorithm for quadratic convex functions. But it can be used for non-linear functions, modifying the length of step α_k; this must be replaced by a search in one direction. Also the way to calculate the residue must be with the calculation of the gradient [10], [11].

Algorithm of the non-linear conjugated gradient $f((x), x_0)$ is as follows [10]:

1) Given a differentiable function $f(x)$ and an initial value x_0
2) Calculate the gradient $r_0 = \nabla f(x_0)$
3) Do $p_0 = -r_0$ and put $k = 0$.
4) Repeat
    a) Calculate α_k using the linear search $\alpha_k = -\frac{r_k^T p_k}{p_k^T A p_k}$
    b) To update $x_{k+1} = x_k + \alpha_k p_k$
    c) Calculate the value of the gradient $\nabla f(x_{k+1})$
    d) Calculate β_k with the Fletcher-Reeves equation $\beta_{k+1}^{FR} = \frac{\nabla f(x_{k+1})^T \nabla f(x_{k+1})}{\nabla f(x_k)^T \nabla f(x_k)}$
    e) Calculate the new address with $p_{k+1} = -r_{k+1} + \beta_{k+1}^{FR} p_k$
    f) To update $k = k+1$
5) Until $\|r_k\| \leq \varepsilon$
6) To return x_k

The application developed has two options for the calculation of β_k, with the formula of Fletcher-Reeves and that of Polak-Ribiere, this is shown below [10], [11]:

$$\beta_{k+1}^{PR} = \frac{\nabla f_{k+1}^T(\nabla f_{k+1} - \nabla f_k)}{\|\nabla f_k\|^2} \quad (5)$$

D. *Homotopic Continuation Method (HCM)*

HCMs are used to find the multiple solutions to systems of non-linear algebraic equations, which is very difficult to solve by traditional methods (example N-R) [12]–[14]. The general way to represent the homotopy is:

$$H(F(x), \lambda) = \lambda F(x) + (1-\lambda)G(x) = 0, \quad (6)$$

where, $F(x)$ is the system that you want to solve, lambda is the homotopic parameter and function $G(x)$ has well-known or trivial solutions. There are several types of homotopy, the adequate selection of $G(x)$ determines the specific homotopy.

In this work, the homotopy of Newton has been implemented, which is defined as $G(x) = f(x) - f(x_0)$ [12]–[14], now if it is replacing $G(x)$ in 6:

$$H(F(x), \lambda) = F(x) - (1 - \lambda)F(x_0) = 0, \qquad (7)$$

where, x_0 is the initial point.

On one side, if $\lambda = 0$ the homotopy system is reduced to the trivial problem $H(f(x), 0) = f(x) - f(x_0) = 0$. On the other side, if $\lambda = 1$, we obtain the original system $H(f(x), 1) = f(x) = 0$ [12]–[14].

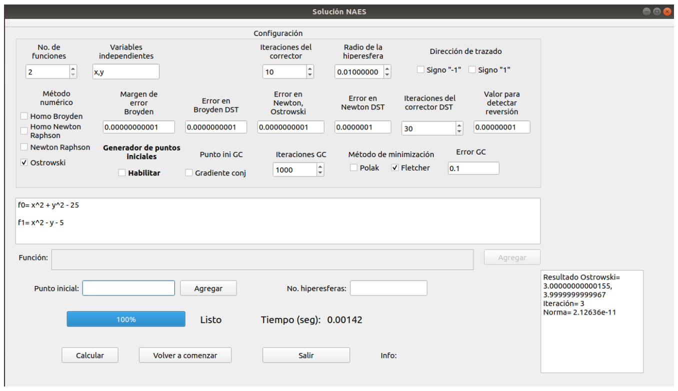
The HCM method uses the spherical tracking algorithm to follow the homotopic trajectory at all times and thus avoid falling into discontinuities or closed curves. The spherical tracking algorithm uses hyperspheres to follow the path and is based on a predictor-corrector scheme [12]–[14]. The Euler method is used as a predictor and as a corrector method, two options were implemented: N-R and Broyden.

III. Description of the Application

The application has been developed in Ubuntu 16.04 and this first stage has only been tested under this S.O. The application uses the GiNaC analyzer to capture systems of equations and can solve systems of two or more equations; it has four numerical methods and each of them has configuration parameters, which the user can choose to obtain the solution to the system of equations to solve. Only HCM has the option of creating two text files, where the solutions and the trajectory found are stored. The configuration that needs to be placed is: the error ε, initial points of the functions (x_0), the number of equations, the independent variables and the numerical method that you want to execute (N-R, OM, CGM, HCM). Each numerical method has control parameters, which help expert users to find the solution in less time. The CGM is the only method that can collaborate with the other methods (N-R, OM, HCM), since it is capable of being used as a generator of initial points (x_0) when the systems of equations are very complex. It also has a generator of initial points that helps the user when he ignores the values x_0.

The program is able to display flags or messages indicate the error that occurred during the evaluation of the system to be solved and the evaluation will stop. These messages will help the user to modify the configuration parameters, to be successful in the next attempt. The following image shows the user interface implemented; where it shows a system of equations of two functions ($f0 = x^2 + y^2 - 25$, $f1 = x^2 - y - 5$), the independent varibles are x and y, the numeric method selected is OM. OM was configured with a maximum of 10 iterations, initial points $(x_0, y_0) = (10, 10)$ and a stop criterion $\|F(x)\| \leq 10^{-10}$. The solutions found are $(x, y) = (3.00000000000155, 3.9999999999967)$, it was obtained in iteration 3, with an error of 2.12636^{-11} and consuming a compute time of 0.00142 seconds. The result is shown in the lower window on the left side and the compute time is shown at the bottom of the application.

The following picture shows the flowchart of program operation.


Solución NAES

Configuración

No. de funciones: 2
Variables independientes: x,y
Iteraciones del corrector: 10
Radio de la hiperesfera: 0.01000000
Dirección de trazado: ☐ Signo "-1" ☐ Signo "1"

Método numérico:
☐ Homo Broyden
☐ Homo Newton Raphson
☐ Newton Raphson
✓ Ostrowski

Margen de error Broyden: 0.00000000001
Error en Broyden DST: 0.0000000001
Error en Newton, Ostrowski: 0.0000000001
Error en Newton DST: 0.0000001
Iteraciones del corrector DST: 30
Valor para detectar reversión: 0.00000001

Generador de puntos iniciales: ☐ **Habilitar**
Punto ini GC: ☐ Gradiente conj
Iteraciones GC: 1000
Método de minimización: ☐ Polak ✓ Fletcher
Error GC: 0.1

```
f0= x^2 + y^2 - 25

f1= x^2 - y - 5
```

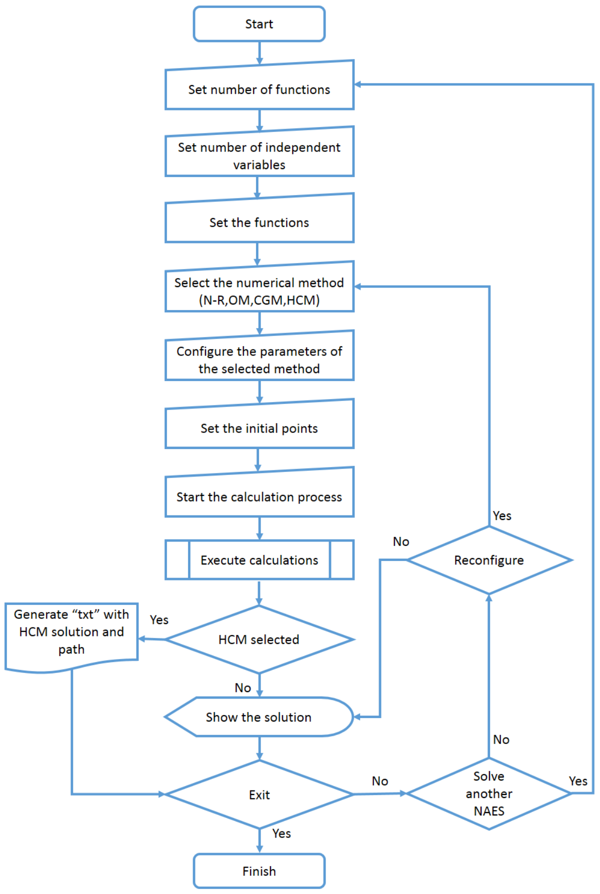
Función:
Punto inicial:
No. hiperesferas:

100% Listo Tiempo (seg): 0.00142

```
Resultado Ostrowski=
3.00000000000155,
3.9999999999967
Iteración= 3
Norma= 2.12636e-11
```

Info:

Fig. 1. User interface



Start
Set number of functions
Set number of independent variables
Set the functions
Select the numerical method (N-R,OM,CGM,HCM)
Configure the parameters of the selected method
Set the initial points
Start the calculation process
Execute calculations
Reconfigure
Yes
No
Generate "txt" with HCM solution and path
Yes
HCM selected
No
Show the solution
No
Exit
No
Solve another NAES
Yes
Yes
Finish

Fig. 2. Flowchart of program operation

IV. Study Cases

This section shows a comparison between the developed application (DA) and Maple 2015. Maple is selected as comparison software, because it incorporates symbolic algorithms, powerful numerical solvers, as well as a set of tools that are sufficiently flexible to fit all the calculation needs. This software is widely used in all areas of science and technology.

The comparison was made with the evaluation of twelve sets of systems of equations of the polynomial type, where each set has at least two independent variables and it has up to twelve independent variables. In addition, each set has from 10 to 250,000 terms (real numbers, trigonometric functions, exponentials, logarithms, etc.). To obtain the execution time of table I, each NAES was evaluated 20 times, the execution time was measured and an average was made. The stopping criterion used was $\|F(x)\| \leq 10^{-9}$. In Maple the commands "fsolve" and "solve" were used to find the solutions of the systems of equations. The following link: https://github.com/gcvelez/NAES, shows a video of the operation of DA and the study cases evaluated in the table I. The tests are performed on a laptop with Intel (R) Core (TM) i5-4210U @ 1.7 GHz processor with 8 GB in RAM and a S.O Ubuntu 16.04. The table shows the comparison.

Study cases	Solution found in DA	Time consumed in DA	Solution found in Maple	Time consumed in Maple
NAES 1	Yes	3.7 ms	Yes	8 ms
NAES 2	Yes	29 ms	Yes	128 ms
NAES 3	Yes	14 ms	Yes	28 ms
NAES 4	Yes	400 ms	Yes	1.05 s
NAES 5	Yes	860 ms	Yes	870 ms
NAES 6	Yes	3.8 s	Yes	33 s
NAES 7	Yes	253 s	Yes	1.9 s
NAES 8	Yes	14 ms	Yes	20 ms
NAES 9	Yes	9 s	Yes	18.4 s
NAES 10	Yes	20 s	Yes	30 s
NAES 11	Yes	19.7 s	Yes	30.05 s
NAES 12	Yes	10 min	No	Indeterminate

TABLE I
COMPARISON TABLE

The results obtained clearly show that the DA is faster than Maple in obtaining the solution, except for the NAES 7. It is also observed that DA solved the NAES 12 and Maple can not solve it. Here we can comment that during the evaluation of the NAES in DA, the numeric methods were used individually with an initial point in $x_0 = 1$, in cases where no solution was obtained (more complex cases) the generator of initial points and CGM was used in conjunction with the other numerical methods (NR, OM, HCM) to obtain the solution. As well as, in the study case 7 (NAES 7) roots were obtained that are different from those obtained in Maple. In addition, the minimum RAM used in DA is of the order of 12 MB and the minimum RAM used in Maple is of the order of 150 MB.

V. Conclusions and future work

The combination of numerical methods with local and global convergence was a success, since it brought as a consequence the solution to all the proposed systems. In addition, the HCM has the characteristic of finding multiple solutions to the systems of nonlinear equations and this was observed in the system 7; since the solutions found with DA were not obtained with Maple. In this first version of DA has shown an excellent performance in consumption of RAM and compute time, this is due to the use of the C++ language and GiNaC libraries. Some libraries or functions developed have been used in robotics to solve problems such as discontinuities and indeterminacy, as well as reduce execution times in the trajectory planning of a robot. As a consequence this type of libraries can be easily implemented in an embedded system.

As future work, it is proposed to continue exploring other numerical methods that could help even more to improve the application, finding solutions for more complex systems with less calculation time; as well as is to improve the graphical interface, making it more intuitive, allowing to save worksheets and include a help section for the user. Another objective is to consolidate libraries for general use with the purpose of supporting users interested in these issues, as well as consolidating the application to work in different operating systems.

References

[1] López, F. I. C. (2017). Análisis dinámico y aplicaciones de métodos iterativos de resolución ecuaciones no lineales (Doctoral dissertation, Universitat Politécnica de Valéncia).

[2] Gutiérrez, L., 2016. Métodos numéricos para resolver ecuaciones y problemas de optimización no lineales. Tesis presentada para obtener el título de Ingeniero petrolero. Universidad Nacional Autónoma de México.

[3] Tarapués, J. A. T., & Álvarez, C. M. R. (2017). Fractales en la Aplicación del Método de Newton al Plano Complejo. Sigma, 13(1), 1-14.

[4] Vassileva, M. P. (2011). Métodos iterativos eficientes para la resolución de sistemas no lineales (Doctoral dissertation, Tesis Doctoral, Universidad Politécnica de Valencia).

[5] Nor, H. M., Rahman, A., Ismail, A. I. M., & Majid, A. A. (2014). Numerical Solution of Polynomial Equations using Ostrowski Homotopy Continuation Method. Matematika, 30, 47-57.

[6] Bauer, C., Frink, A., & Kreckel, R. (2002). Introduction to the GiNaC framework for symbolic computation within the C++ programming language. Journal of Symbolic Computation, 33(1), 1-12.

[7] Vollinga, J. (2006). GiNaC—Symbolic computation with C++. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, 559(1), 282-284.

[8] Ismail, W. Z. A. W. (2016). Newton Homotopy Continuation Method for Solving Nonlinear Equations using Mathematica. Journal of Science and Technology, 7(1).

[9] Nor, H. M., Rahman, A., Ismail, A. I. M., & Majid, A. A. (2016). Super Ostrowski homotopy continuation method for solving polynomial system of equations. Matematika, 32(1), 53-67.

[10] Solorio, F. C. (2012). OPTIMIZACION NO–LINEAL (VER 1.10).

[11] Kindelán, U. (2007). Resolución de sistemas lineales de ecuaciones: Método del gradiente conjugado. Universidad Politécnica de Madrid.

[12] G. C. V. López, Tesis de Maestría: Planeación de Trayectorias Homotópicas Libres de Colisión para Brazos Robóticos. Universidad Veracruzana, Xalapa, Veracruz, Mexico, 2017.

[13] Diaz-Arango, G., Vázquez-Leal, H., Hernandez-Martinez, L., Pascual, M. T. S., & Sandoval-Hernandez, M. (2017). Homotopy path planning for terrestrial robots using spherical algorithm. IEEE Transactions on Automation Science and Engineering, 15(2), 567-585.

[14] Vazquez-Leal, H., Marin-Hernandez, A., Khan, Y., Yıldırım, A., Filobello-Nino, U., Castañeda-Sheissa, R., & Jimenez-Fernandez, V. M. (2013). Exploring collision-free path planning by using homotopy continuation methods. Applied Mathematics and Computation, 219(14), 7514-7532.