SOLUTION OF NON LINEAR EQUATIONS BY GENETIC ALGORITHMS

حل المعادلات الغير خطية بواسطة الخوارزميات المحاكية للجينات الوراثية

Dr. E. Jamal Azzam

ملخص

إن الخوارزميات المحاكية للجينات الوراثية GA مرتكزة على أفكار النشوء الطبيعي للجينات الوراثية ، والفكرة الأساسية لهذه الخوارزميات مصممة لتحاكي العمليات الطبيعية للتطور ، وخاصة تلك القائمة على مبدأ البقاء للأصلح ومن ثم فهي تمثل استغلالاً ذكيا للبحث العشوائي في مجال محدد لحل مشكلة ما . وقد درست هذه الخوارزميات وتم تجربتها وتطبيقها في العديد من المجالات الهندسية بصورة موسعة . فهي لا تقدم طرق مختلفة لحل المشاكل فقط ولكنها تتفوق بانتظام على الطرق التقليدية في معظم المشاكل ذات الصلة .

وفي هذا البحث تم تقديم مجال آخر (جديد) لتوظيف خوارزميات الجينات الوراثية وهو حل المعادلات اللاخطية متعددة المتغيرات واستخدام دالة ملاءمة مقترحة .

إن استخدامها في حل المعادلات ذات المتغير الواحد هي عملية تعظيم Optimization يسهل توظيفها فيه ، وتكمن المشكلة في المعادلات المتعددة المتغيرات حيث لم يسبق استخدام خوارزميات الجينات الوراثية لحل مثل هذه المعادلات إذ يلزم لحلها تحديد نسبة مشاركة قيمة كل متغير (في كل محاولة حل) في الخطأ الناشئ بالمعادلات ليتسنى احتساب مدى مناسبة هذه القيمة لهذا المتغير ومن ثم الاقتراب من القيمة التقريبية الصحيحة تدريجياً وذلك لكل المتغيرات آنيا . وقد تم اقتراح طريقة لاحتساب " مناسبة القيم " بهذا الخوارزم لإمكان توظيف GAs وتم مقارنة النتائج بالطرق التقليدية .

ABSTRACT:

Genetic Algorithms (GAs) are adaptive search algorithms premised on the evolutionary ideas of natural selection and genetic. The basic concept of GAs is designed to simulate processes in natural system necessary for evolution, specifically those that follow the principle survival of the Fittest. As such they represent an intelligent exploitation of a random search within a defined search space to solve a problem. GAs has been widely, studied, experimented and applied in many fields in engineering world. Not only they provide an alternative methods for solving problems . it consistently outperforms ,other traditional methods in most of the problems link.

In this paper another new field for implementing GAs is introduced, this is the solution of multivariable nonlinear equations including a proposed fitness function. Solution of a single variable equations is an optimization problem, can be solved easily by GAs. The problem lies in the multi variable equations, since it is necessary to determine how much each variable shares in the errors of equations. This is crusal to evaluate the fitness of these values. Consequently, approaching the approximate exact values gradually. A method to compute the fitness functions for this case is proposed in the algorithm to enable GAs to be implemented.

Accepted November 26, 2008

1. Introduction :

The genetic algorithm is a stochastic optimization algorithm that was originally motivated by the mechanisms of natural selection and evolutionary genetics. Over the last decade, GA has been extensively used as search and optimization tools in various problem domains, including : science, commerce and engineering. The primary reasons for their success are their broad applicability, ease of use and global perspective. There are some differences between a GA and traditional searching algorithms. They can be summarized as follows[1], [2]:

- The algorithm works with a population of strings, searching many peaks in parallel, as opposed to a single point.
- The GA works directly with strings of characters representing the parameter sets, not the parameters themselves.
- The GA uses probabilistic rules instead of deterministic rules.
- The GA uses objective function information instead of derivatives or other auxiliary knowledge.

GA is inherently parallel, because it simultaneously evaluates many points in the parameter space (search space). So, the GA has a reduced chance of converging to local optimum and would be more likely to converge to global optimum. It requires only information concerning the quality of the solution produced by each parameter set (objective function values). This differs from many optimization methods which require derivative information or, worse yet, a complete knowledge of the problem structure and parameters. Since the GA does not require such problem specific information, it is more flexible than most search methods [3-9]. Typically, the GA is characterized by the following components:

- A genetic representation (or an encoding) for the feasible solution to the optimization problem.
- A population of encoded solution.
- A fitness function that evaluates the optimality of each solution.
- Genetic operators that generate a new population from the existing population.
- Control parameters.

The basic flow chart of the GA is illustrated in Fig. 1 where ($\varepsilon > 0$) a small number to check convergence.

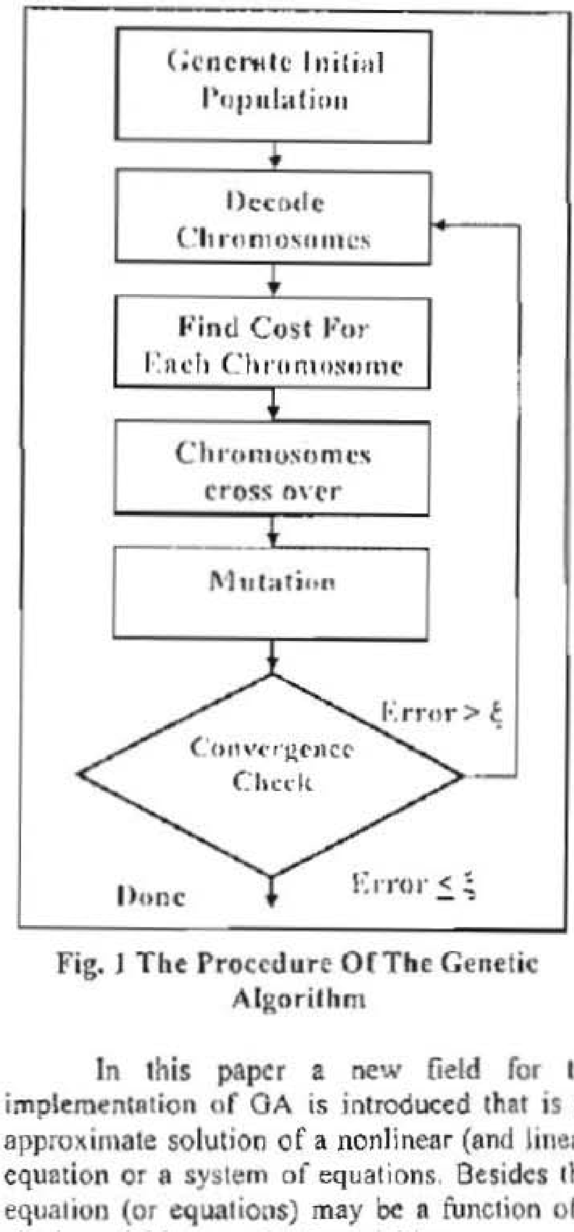


Fig. 1 The Procedure Of The Genetic Algorithm

In this paper a new field for the implementation of GA is introduced that is an approximate solution of a nonlinear (and linear) equation or a system of equations. Besides this equation (or equations) may be a function of a single variable or multiple variables.

There are a lot of conventional iterative in [10-15]. They are collected in Table 1. A comparison of most of them shown in [14].

The proposed algorithm shows a considerable success where the traditional iterative methods costs a lot computation wise.

A specific obstacle of using GA to be considered for multi variable functions that is : what is the share of each variable in the error function (E). That effect of each variable on the error is crusal for the GA to compute the fitness function. A proposed solution for this problem (used in this Algorithm) is introduced as the rate of derivatives of functions (f_j) w.r.t to a single virable , where (n_f) is the nonlinear functions.

The summation of the parial derivatives of the functions w.r.t. this variable. Although GAs don't necessitate or use the derivatives of the functions, the proposed algorithm does not necessitate to know it either, but it computes it by causing a small perturbation in the variables (in separate) and computes the corresponding change of the function .

Table 1 Iterative Methods

1	Newton's Method for Multiple Roots	$X_{n+1} = X_n - \frac{f(xn)\,f'(xn)}{f'(xn)^2 - f(xn)\,f''(xn)}$
2	Convex acceleration of whittaker's method	$X_{n+1} = X_n - \frac{f(xn)}{2f'(xn)}\left(2 - L_f(xn)\right)$, $L_f(x) = \frac{f(x)f''(x)}{f'(x)^2}$
3	Double convex acceleration of whittaker's method	$X_{n+1} = X_n - \frac{f(xn)}{4f'(xn)}\left(2 - L_f(xn) + \frac{4+2L_f(xn)}{2-L_f(xn)(1-L_f(xn))}\right)$
4	Halley's method	$X_{n+1} = X_n - \frac{f(xn)}{f'(xn)}\frac{1}{1-\frac{1}{2}L_f(xn)} = X_n - \frac{1}{\frac{f'(x)}{f(x)} - \frac{f''(x)}{2f'(x)}}$
5	Chebyshev's method	$X_{n+1} = X_n - \frac{f(xn)}{f'(xn)}\left(1 + \frac{L_f(xn)}{2}\right)$
6	Convex acceleration of Newton's method or the super Halley method	$X_{n+1} = X_n - \frac{f(xn)}{2f'(xn)}\frac{2-L_f(xn)}{1-L_f(xn)} = Xn - \frac{f(xn)}{f'(xn)}\left(1 + \frac{\frac{1}{2}L_f(xn)}{1-L_f(xn)}\right)$
7	(Shifted) Stirling's method	$X_{n+1} = X_n - \frac{f(xn)}{f'(xn - f(xn))}$
8	Steffensen's method	$X_{n+1} = X_n - \frac{f(xn)}{g(xn)}$ with $g(x) = \frac{f[x+f(x)] - f(x)}{f(x)}$
9	Midpoint method	$X_{n+1} = X_n - \frac{f(xn)}{f'\left(xn - \frac{f(xn)}{2f'(xn)}\right)}$
10	Turbo Ostrowski's method	$X_{n+1} = X_n - u(xn)\frac{f(xn-u(xn)) - f(xn)}{2f(xn-u(xn)) - f(xn)}$
11	Jarrat's method	$X_{n+1} = X_n - \frac{1}{2}u(xn) + \frac{f(xn)}{f'(xn) - 3f'\left(xn - \frac{2}{3}u(xn)\right)}$
12	Inverse free Jarratt's method	$X_{n+1} = X_n - u(xn) + \frac{3}{4}u(x_n)h(x_n)\left(1 - \frac{3}{2}h(x_n)\right)$ with $U(x) = \frac{f(x)}{f'(x)}$ and $h(x) = \frac{f'\left(x - \frac{2}{3}u(x)\right) - f'(x)}{f'(x)}$

1. The Proposed Algorithm

Given a number of (linear or nonlinear) functions nf- functions of a number of variables (n_v)

$$\left.\begin{aligned} f_1(x, y, x, \ldots\ldots n_v) &= 0 \\ f_2(x, y, x, \ldots\ldots n_v) &= 0 \\ &\vdots \\ f_{nf}(x, y, x, \ldots\ldots n_v) &= 0 \end{aligned}\right\} \quad (1)$$

The variables are represented by symbols e.g. x, y, z, or x_1, x_2, x_3.
Randomly get initial values for these variables x, y, x, ... n_v

The proposed algorithm can be detected by the following steps

1. The errors : substitute the values of variables to compute the error in each function E_i where (i = 1,2... nf) and the total error function (E).

$$\left.\begin{aligned} E_1 &= f_1(x, y, x, \ldots\ldots n_v) - 0 \\ E_2 &= f_2(x, y, x, \ldots\ldots n_v) - 0 \\ &\vdots \\ E_{nf} &= f_{nf}(x, y, x, \ldots\ldots n_v) - 0 \end{aligned}\right\} \quad (2)$$

The error function $E = \sum_{i=1}^{nf} Ei$ (3)

2. The Derivatives : by a small perturbation in each variable (individually) compute the derivative of each function as :

$$\bar{f}_{1x} = \frac{\Delta_{f1}}{\Delta x}$$
$$\bar{f}_{2x} = \frac{\Delta_{f2}}{\Delta x}$$
$$\vdots$$
$$\bar{f}_{nfx} = \frac{\Delta f_{nfx}}{\Delta x}$$

The summation of derivatives w.r.t. (x) is

Similarly

$$\left.\begin{aligned} \frac{\partial F}{\partial x} &= \sum_{i=1}^{nf} \frac{\Delta_{f1}}{\Delta x} \\ \frac{\partial F}{\partial y} &= \sum_{i=1}^{nf} \frac{\Delta_{f1}}{\Delta y} \\ \frac{\partial F}{\partial z} &= \sum_{i=1}^{nf} \frac{\Delta_{f1}}{\Delta z} \\ &\vdots \\ \frac{\partial F}{\partial nv} &= \sum_{i=1}^{nf} \frac{\Delta f_{nf}}{\Delta nv} \end{aligned}\right\} \quad (4)$$

3. Parents : repeat steps 1 and 2 for a suitable number of parents P_n ($P_n > 2$).
4. Fitness function : compute the fitness function for each variable in all parents substituting the summed differention from Eqn. 4 as follows :

$$\left.\begin{aligned}
\mathrm{Fit}_x &= \frac{\partial F}{\partial x} / S_x = \sum_{i=1}^{nf} \frac{\Delta f1}{\Delta x} / S_x \\
\mathrm{Fit}_y &= \frac{\partial F}{\partial y} / S_y = \sum_{i=1}^{nf} \frac{\Delta f1}{\Delta y} / S_x \\
\mathrm{Fit}_z &= \frac{\partial F}{\partial z} / S_z = \sum_{i=1}^{nf} \frac{\Delta f1}{\Delta z} / S_x \\
\mathrm{Fit}_{nv} &= \frac{\partial F}{\partial nv} / S_{nv} = \sum_{i=1}^{nf} \frac{\Delta f1}{\Delta nv} / S_x
\end{aligned}\right\} \quad (5)$$

Where

$$S_x = \sum_{j=1}^{P} \frac{\partial F}{\partial x}$$
$$S_y = \sum_{j=1}^{P} \frac{\partial F}{\partial y}$$
$$S_z = \sum_{j=1}^{P} \frac{\partial F}{\partial z}$$
$$\vdots$$
$$S_{nv} = \sum_{j=1}^{P} \frac{\partial F}{\partial nv}$$

5. Cross-over operation : for each variable the higher fitness values in parents are repeated in the next offspring while the low fitness values are randomly crossed over to produce a new chromosome

The new values of variables are computed from the parents to produce the new generation by the real value crossover :

$$C = R * P_{parent} A + (1 - R) * P_{parent} B \qquad (6)$$

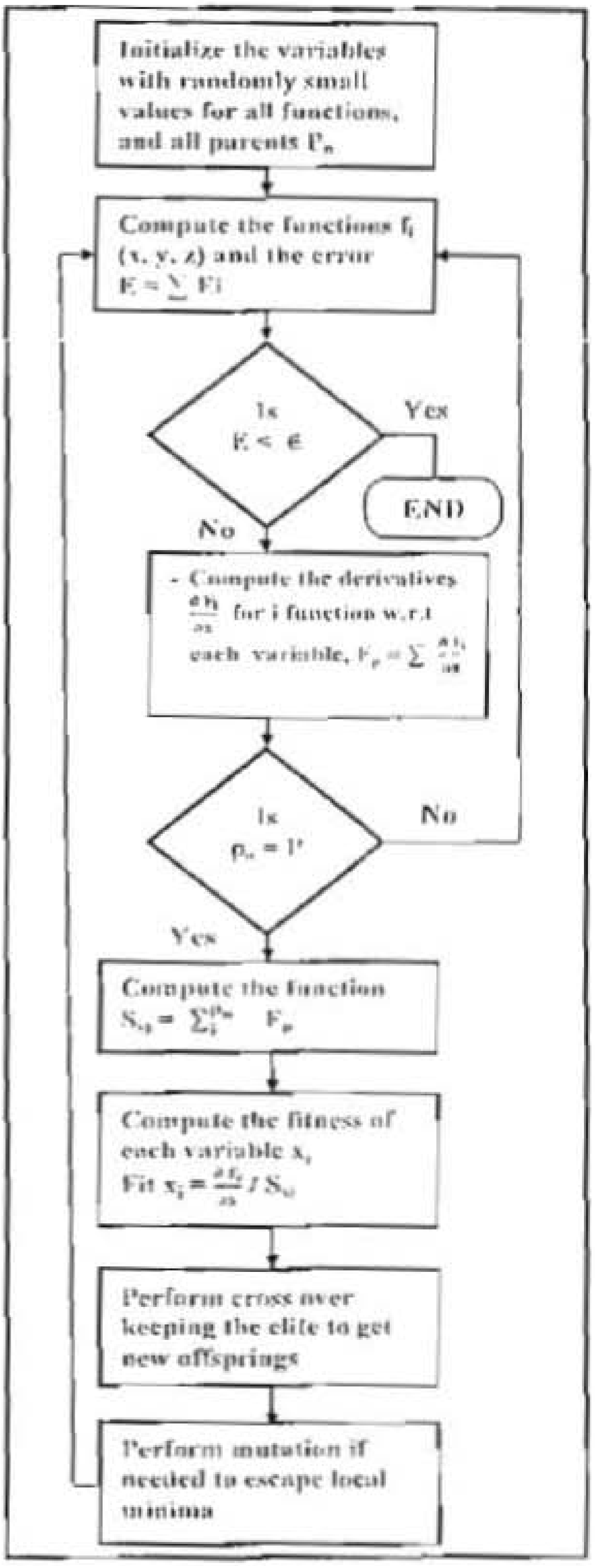
Where

C is the child of two parents A and B

R is random number between 0 and 1

This is repeated for each variable per each child. The number of generated Childs is equal to the number of parents (P_n) to be used in the next generation.

6. Mutation : check if the algorithm felled in a local minima, if so, perform a mutation process by exchanging the variables values between two random parent numbers.
7. Repeat the steps from 1 to 6 till the total error (E) is less than a specified small value6 . The algorithm is explained in Fig.2.



Initialize the variables with randomly small values for all functions, and all parents P_n
Compute the functions f_i (x, y, z) and the error $E = \sum Fi$
Is $E < \epsilon$
Yes
END
No
- Compute the derivatives $\frac{\partial f_i}{\partial x}$ for i function w.r.t each variable, $F_p = \sum \frac{\partial f_i}{\partial x}$
Is $p_n = P$
No
Yes
Compute the function $S_q = \sum_1^{P_n} F_p$
Compute the fitness of each variable x_i Fit $x_i = \frac{\partial f_i}{\partial x} / S_q$
Perform cross over keeping the elite to get new offsprings
Perform mutation if needed to escape local minima

Fig. 2 The Proposed Algorithm

Results :

Example 1 : for explanations purposes the following equation of a single variable is considered :

$F_1(x) = x^4 + 6x^3 + 7x^2 - 6x - 8 = 0$

The algorithm steps (1-7) are represented (for one trial only) in table 2. The consequent trials are repeated in a similar way till $E < \varepsilon$.

The resulted values are :

$X = -1.00154$

The error value = 0.00925441

Notes :

- $\frac{\partial F}{\partial x} = \acute{f}(x) + \acute{f}(y) + \acute{f}(x)$ ... and because there is one variable (x), $\frac{\partial F}{\partial x} = \acute{f}(x) + \acute{f}(y) + \dots \acute{f}(nv) = \acute{f}(x)$.

- R can be randomized for each parent instead of being the same for all parents. Its value is changing from trial to trial.

The parents numbers to be crossed over are randomized and are different in each trial. In this step P_2, which has the highest fitness is repeated in the new generation, other parent pairs are crossed over randomly (P_1, P_2), (P_3, P_3), (P_4, P_3), (P_5, P_2) by the ratios (R) & (1-R).

Example 2 : consider the system of linear equations

$4X_1 - 26X_2 + X_3 = 4$
$X_1 + 6X_2 - 45X_3 = 9$
$-X_1 - 17X_2 + 5X_3 = 2$

The approximated values resulted by the proposed algorithm are :

$X_1 = -0.100790$
$X_2 = -0.179341$
$X_3 = -0.2271\,10$

The total error = 0.0890370 No. of iterations is 273 computation time 3.45 sec.

Example 3 : consider the two non linear functions :
$F_1(x,y) = 2x^2 - xy - 5x + 1 = 0$
$F_2(x,y) = x + 3\log_{10} x - y^2 = 0$
The approximate solution using the proposed algorithm is :

$X = 3.39805$ & $y = 2.12373$ and the error is 0.594821

The same two equations are solved by the method of iteration, the results are :
$X = 3.487$ & $y = 2.262$ its error is 0.007137

It is clear that it is more accurate.

Example 4 : to solve the following simultaneous non linear equations
$F_1(x, y, z) = -0.1 + x + x^2 - 2yz = 0$
$F_2(x, y, z) = 0.2 + y - y^2 + 3xz = 0$
$F_3(x, y, z) = -0.3 + z + z^2 + 2xy = 0$

The results are :

$X = -0.0028$ & $Y = -0.16806$ & $Z = 0.24115$
The minimum error of the three functions is 0.0234

The proposed GA algorithm produces approximate solutions less accurate compared to the conventional methods (e.g. method of iteration, Newton method, steepest descent, ...) which result more accurate solutions. The algorithm has the following disadvantages :
The solutions are less accurate and the algorithm is sensitive to some factors e.g. the initial values, the cross over random value R, the number of parents to be crossed over. Attention has to be paid for the range of randomized values get to escape a local minima. But the algorithm has the following advantages :
Swiftness, parallism in approximating the variables, its computation burden is simpler, does not necessitate the derivatives of the functions to be introduced, and the computation time does not increase significantly with the increase in the number of attributes (variables, functions, and number of parents per each variable).

Table 2

	X	$\lvert E_i\rvert = f_i(x)$	$f'(x)$	$\frac{\partial F}{\partial x}$	Fitness	R	New x
P1	-0.00125126	7.99248	-6.02649	-6.02649	0.002659	0.89082	-0.0894129
P2	$\boxed{-0.808741}$	1.31513	-1141.71	-1141.71	0.50682	0.89082	$\boxed{-0.808741}$
P3	-0.350291	5.28216	-51.184	-51.184	0.022586	0.89082	-0.350291
P4	-0.746605	1.8047	-594.826	-594.826	0.262488	0.89082	-0.703335
P5	-0.710501	2.10049	-472.356	-472.356	0.208444	0.89082	-0.721227
		Total E = 18.495		Sx = 2266.1			

Conclusions :

This paper shows that the Genetic Algorithms (GAs) can be used to optimize a set of functions by minimizing the error.
Although, implementing GAs to optimize a function of a variable or set of functions (of a single variable) is a known task, it is not used for multiple variables functions so far. The reason lies on difficulty in determining the shareness of each variable in the error of the function (or summed errors of functions). This paper introduces a proposed GA algorithm explaining a solution for this problem. It opens the way to implementing GAs for multivariable functions optimization. Although the results are less accurate compared to the conventional methods, it has these advantages : swiftness, doesn't necessitate the derivatives of the functions and the increment in computation time with the increase in attributes (i.e. number of variables and (or) number of functions) is trival.

References :

[1] Armingol J.M., Moreno L., Escalera A.d., and Salichs M.A., "A Genetic Algorithm For Mobil Robot Localixation Using Ultrasonic Sensors", Journal of Intelligent and Robotic Systems, Vol. 34, N.2, PP 135-154, 2002.

[2] Jamal A.F., Hassan E.D., Soliaman M.S., " Genetic Algorithm For Dynamic Task Allocation of Multi Autonomous UAVs"., MEJ, Vol. 32, No. 2 June 2007.

[3] Jose B., Cruex J. "Genetic Algorithm for Task allocation in UAV Cooperative Control, "Genshe C., AIAA Guidance, Navigation, and Control Conference, Austin, Texas, USA, 2003.

[4] Frank H., Thomas B.,"Genetic Self-Leaning", Dortmund University, Computer Science Dept., Germany, 1992.

[5] Heinx M., Dirk S., "Analysis and Selection, Mutation and Recombination in Genetic Algorithm", GMD Schlos, Birling hoven, Germany, 2006.

[6] Heinx M.," Evolutionary Algorithms : Theory and applications:, GMD Schlos Birlioghoven Augustin, Genermay, 1994.

[7] Fleming P.J., Purshouse R.C., "Genetic Algorithms In Control Systems Engineering", IEE Colloquium on Genetic Algorithms for Control Systems Engineering, 2001.

[8]Yew S.O."Artificial Intelligence Technologies in Complex Engineering Design", P.H.D. Thesis, South Hampton University, U.K., 2001.

[9] Katya Q.V., Carlos M.F., Peter J.F., "Multi Objective Genetic Programming : A Non Einear System Identification Application", Genetic Programming Conference, Stanford, CA. 207-212,1997.

[10] Randy E., Sue E., "Practical Genetic Algorithms", John Wiley & Sons Inc., USA, 2004.

[11] Cheng L., "Numerical Solution of Nonlinear Equations", Dept. of Chemical Eng., National Taiwan, 2005.

[12] Christopher J. X, "An Introduction to Numerical Analysis for Electrical and Computer Engineers",John Wiley & Sons, Inc., USA, 2004.

[13] Demidovich B.P, Maron I.A.," Computational Mathematics ", Mir publishers,1981.

[14] Juan E.V., "Graphic and Numerical Comparison Between Iterative Methods",Mathematical Intelligencer,no.1,37-46, 2002.

[15] Daniel N.K, Jose M.M., Sandra A.S., "Solving Non Einear Systems of Equations with Simple Constraints", Applied Mathematics 16,PP. 215-235, 1997.