PROBABILITY-DIRECTED PROBLEM OPTIMIZATION TECHNIQUE FOR SOLVING SYSTEMS OF LINEAR AND NON-LINEAR EQUATIONS

Muhammed J. Al-Muhammed

Faculty of Information Technology American University of Madaba, Madaba, Jordan

ABSTRACT

Although many methods have been proposed for solving linear or nonlinear systems of equations, there is always a pressing need for more effective and efficient methods. Good methods should produce solutions with high precision and speed. This paper proposed an innovative method for solving systems of linear and nonlinear equations. This method transforms the problem into an optimization problem and uses a probability guided search technique for solving this optimization problem, which is the solution for the system of equations. The transformation results in an aggregate violation function and a criterion function. The aggregation violation function is composed of the constraints that represent the equations and whose satisfaction is a solution for the system of equations. The criterion function intelligently guides the search for the solution to the aggregate violation function by determining when the constraints must be checked; thereby avoiding unnecessary, time-intensive checks for the constraints. Experiments conducted with our prototype implementation showed that our method is effective in finding solutions with high precision and efficient in terms of CPU time.

KEY WORDS

Solutions for systems of linear and non-linear equations, random-guided search, optimization problem, global minimum

1. INTRODUCTION

Systems of linear or non-linear equations are ubiquitous. They frequently result from modeling problems in many important domains such as engineering, robotics, business, and many more. Therefore, finding effective and efficient methods for finding simultaneous solutions for these equations is extremely and practically important to study the properties of the problems modeled by these equations.

The desirable properties of the solution methods are the effectiveness and efficiency. The method must be effective in producing a solution with high precision. It must also produce the solution for the problem in a minimum time and resources requirements.

Researchers have proposed many methods for solving systems of equations [1][7][8] (see [5] for a good discussion of these methods). These methods either solve the equations directly by

Dhinaharan Nagamalai et al. (Eds) : CCSEIT, ICBB, DMDB, AIAP, CNSA - 2019
pp. 35-57, 2019. © CS & IT-CSCP 2019 DOI: 10.5121/csit.2019.90605

applying numerical methods [4][9][10][11][17][23] or solve them indirectly by transforming the system of equations into problem optimization [2][3][4][5][12][16][19][22].

This paper offers a technique for solving systems of linear or nonlinear equations. The technique indirectly solves a system of equations by solving an optimization problem. Each equation in the system is considered a constraint. These constraints are combined, in a way to be made precise later, into a non-negative function, called aggregate violation function. This function measures the collective amount of violation caused by some substitution to the parameters of the constraints. Thus, a substitution for which the aggregate violation function evaluates to zero is the solution for the system of equations because this substitution satisfies all the constraints (i.e. the equations).

The proposed technique goes even further by creating a function, called a criterion function, whose main objective is to identify whether some substitution is promising (results in a better value for the aggregate violation function) or not. The criterion function is created using the systems of equations in such a way that, as will show later, its computational demand is much less than the computational demand of the aggregate violation function. This function controls the process of evaluating the aggregate violation by permitting this evaluation at some substitution only when this substitution is promising. Our technique therefore saves the time required for blindly evaluating the aggregate violation function on every possible substitution and consequently increases its efficiency.

To solve the optimization problem, we propose a probability-guided algorithm that uses random numbers and biased mapping to quickly direct the search to the parts of the variables' domains where the solution resides. The algorithm also makes use of the criterion function to intelligently direct the search in such a way that the method avoids unnecessary checks of the constraints.

The paper makes the following contributions. First, it offers an effective algorithm for problem optimization. Second, it offers effective transformation from systems of equations to optimization problem. This transformation produces a very rigorous aggregate violation function whose satisfaction results in high precision solutions of the equation systems. Third, the transformation produces a light-weighted criterion function that quickly determines if the aggregate violation function must be checked or not, thereby avoiding unnecessary time consuming checks of the aggregate violation.

We present our contribution as follows. Section 2 describes the problem formalization, which transforms the solution to systems of equations to the solution of problem optimization. In section 3, we discuss the proposed algorithm and its technical details. Section 4 presents our experimental evaluation of the proposed technique. We conclude and give directions for future work in sections 5.

2. THE PROBLEM FORMALIZATION

In this section we formalize the problem of solving a system of equations. We present some fundamental concepts in subsection 2.1 and show our formalization in subsection 2.2.

2.1 PRELIMINARIES

Consider the following system of equations:

$$\begin{aligned} &a_1^1 x_1 + a_2^1 x_2 + \ldots + a_n^1 x_n = b_1 \\ &a_1^2 x_1 + a_2^2 x_2 + \ldots + a_n^2 x_n = b_2 \\ &\ldots \\ &a_1^n x_1 + a_2^n x_2 + \ldots + a_n^n x_n = b_n \end{aligned} \quad (1)$$

We rewrite the system (1) as the following set of constraints

$$C = a\,x + a\,x + \ldots + a\,x = b \quad (i = 1, 2, \ldots, n) \quad (2)$$

The terms a_j^i and b_i (i, j= 1, …n)are constants and x_i's are variables. The variables x_i'scan be of any power (not necessarily linear) or even functions (sine, log, etc.). We represent them as in (2) just to simplify the presentation. A solution to the set of equations is a substitution$X$*($x_1^*, x_2^*, \ldots, x_n^*$) for the variables x_i's such that all of the constraints C^i hold.

We associate with the constraints (2) a function $f(x_1, x_2, \ldots, x_n)$, which is defined bysumming all the constraints C^i and taking the absolute value for the resulting sum. The variables x_i'swith the same index and power are combined together. Consider for instance the following system of equations.

$$\begin{aligned} &x_1 + 3x_2^2 + x_3 = 6 \\ &3x_1 + 2x_2 + x_3 = 7 \\ &x_1^2 + x_2^2 + 2x_3 = 4 \end{aligned}$$

We sum these three constraints to yield the following function $f(x_1, x_2, x_3)$.

$$f(x_1, x_2, x_3) = | 4x_1 + x_1^2 + 2x_2 + 4x_2^2 + 4x_3 - 17 |$$

Observe that the variables x1 of powers 1 are summed to yield 4x1 and the variable x1 of power 2 are summed together and so on.

According to this, we create the function $f(x_1, x_2, \ldots, x_n)$ from the equations (1) by adding together the variables with the same index and with the same power. The addition yields the function (3).

$$f(x_1, x_2, \ldots, x_n) = | \sum_{i=1}^{n} a_1^i x_1 + \sum_{i=1}^{n} a_2^i x_2 + \ldots + \sum_{i=1}^{n} a_n^i x_n - \sum_{i=1}^{n} bi | \quad (3)$$

Let $X^i(x_1^i, x_2^i, \ldots, x_n^i)$ be a possible substitution. We define the degree of the violation caused by the substitution X^i to some constraint C^k by:

$$\rho^k(X^i) = | a_1^k x_1^i + a_2^k x_2^i + \ldots + a_n^k x_n^i - b_k |$$

It should be clear that the degree of the violation is zero (i.e. no violation) when the substitution Xisatisfies the constraint Ck. The degree of the violation is positive if Xi does not satisfy the constraint Ck. Therefore, greater values of ρk(Xi) indicate a large violation of the substitution Xito the constraint Ck.

We in addition define the aggregate violation of the substitution X^i to all of the constraints C^i as follows

$$E^c(X^i) = \sum_{k=1}^{n} \rho^k(X^i)$$

That is, the aggregate violation of some substitution X^i is the sum of the violations of X^i over all the constraints. The aggregate violation is zero only if X^i satisfies all the constraints. Greater values of the aggregate violation $E^c(X^i)$ indicate a greater violation for a constraint or more.

We emphasize that function $f(x_1, x_2, \ldots, x_n)$ is completely different from the aggregate violation $E^c(X^i)$. The following example shows the difference between them.

Example (1): consider the following two linear equations whose solution is X (1, 1).

$$2x_1 + x_2 = 3$$
$$-x_1 + x_2 = 0$$

By summing these two equations and take the absolute value we obtain the function $f(x_1, x_2) = |x_1 + 2x_2 - 3|$. Let $X(0, 1.5)$. It is clear that $f(0,1.5)$ equals to zero while $E^c(X)$ is not equal to zero. That is, $E^c(0, 1.5) = |(2(0) + 1.5 - 3)| + |-(0) + 1.5 - 0| = 3$.

For most of the systems of equations (especially the systems of linear equations), there is a large number of common variables between the equations. For such systems the function $f(x_1, x_2, \ldots, x_n)$ is likely to have fewer number of variable substitutions than the original constraints because the common variables are combined by the summation. The function $f(x_1, x_2, \ldots, x_n)$ demands therefore less computation than the aggregate violation function does for such systems. On the other hand, even if there are few (or even none) common variables, the function $f(.)$ still demands less computation than the aggregate violation. That is because it computes the absolute value only one time while the aggregate violation function computes the absolute value for each constraint.

Proposition (1)*: Let X^i, X^j betwo possible substitutions for the aggregate violation function. If $f(X^i) > E^c(X^j)$, it cannot be the case that $E^c(X^i) < E^c(X^j)$.This is represented mathematically as $f(X^i) > E^c(X^j) \Rightarrow E^c(X^i) < E^c(X^j)$.*

Proof

The proof of this proposition is straightforward and follows directly from the definition of the function $f(.)$ and the aggregate violation function.

Suppose that $f(X^i) > E^c(X^j)$. Since $f(X^i) < E^c(X^i)$ based on the definition of both $f(.)$ and $E^c(.)$ and the properties of the absolute value, we conclude that $E^c(X^i) > E^c(X^j)$. ♦

Proposition (1) precisely defines the relationship between the aggregate violation $E^c(.)$ and the function $f(.)$. For any substitution X^i, if the value of the function $f(X^i)$ is larger than the value of the aggregate violation at some previous substitution X^j, the substitution X^i will not satisfy the constraints better than X^j; that is X^i will *not* reduce the value of the aggregate violation. Given this, the function $f(.)$ can be used as a precondition for checking the constraints at some X^i. Namely, the constraints should not be checked at any substitution X^i for which proposition (1)

holds. Observe if the substitution X^i does *not* satisfy proposition (1) (i.e. $f(X^i) \leq E^c(X^j)$), it is not necessary that X^i reduces the constraints violations (reduce the aggregate violation).

We therefore call the function $f(.)$ a *criterion function*. It is called so because it guides the process of whether we should evaluate the aggregate violation or not at some substitution X^i.

2.2 PROBLEM TRANSFORMATION

Given the aggregate violation of the constraints $E^c(X^i)$ and the criterion function $f(X^i)$, we formalize the problem of solving the system (1) as follows. Rather than solving the system of equations per se, we indirectly solve it by solving the following optimization problem.

Minimize $E^c(X^i) = \sum_{k=1}^{n} \rho^k(X^i)$

Guided by the criterion function:

$$f(x_1, x_2, ..., x_n) = \left| \sum_{i=1} a_1^i x_1 + \sum_{i=1} a_2^i x_2 + ... + \sum_{i=1} a_n^i x_n - \sum_{i=1}^{n} bi \right| \quad (4)$$

According to the definition of the aggregate violation $E^c(X^i)$, it is clear that the global minimum of $E^c(X^i)$ is zero. Therefore, if $E^c(X^i) = 0$ for some substitution X^i, this substitution is the global minimum for the optimization problem (4). This substitution X^i is also an exact solution for the system (1) because it violates no constraint (i.e. satisfies all the equations in system (1)). If, however, $E^c(X^i) > 0$, the substitution X^i is not the global minimum for the optimization problem (4). This substitution X^i is not an exact solution for the system (1) either due to the constraint violation (i.e. one or more of the equations in system (1) not satisfied). Furthermore, it can be easily shown that if the substitution X^i is an exact solution for the system (1), it is not the global minimum for the optimization problem (4) either. According to this discussion, a substitution X^i is a solution for the system (1) if and only if it is the global minimum for the optimization problem (4).

The criterion function $f(x_1, x_2, \ldots, x_n)$ plays an extremely important role in the process of optimizing the problem (4);and equivalently finding a solution for the equations (1).It intelligently instructs the solver when the constraints must be checked for some substitution X^i. Blindly checking the constraints for every possible substitution X^i has a negative impact on the performance of the solver since this check is computationally intensive especially if we have a large number of these constraints. Generally speaking, the criterion function $f(.)$ greatly improves the solver's performance by allowing this solver to check the constraints only when it should.

3. THE HIT-HIT MOVING DIRECTIVES ALGORITHM

We propose an algorithm, called Hit-Hit Moving Directives (HHMD-3), for solving unconstrained optimization problem (4). HHMD-3 is a probability-directed search technique that finds the global optimum (minimum or maximum)for unconstrained optimization problems such as the one in (4). This algorithm utilizes sequences of random numbers to search for the global optimum within the domains of the variables. It defines three directives over the unity interval [0, 1] as shown in Figure 1. These directives are a principal directive and two marginal directives. The principal directive covers part of the unity interval, which we call the effective search interval (ESI). The marginal directives cover the remaining part of the unity interval, which we call the marginal search intervals (MSI).The principal directive is centered at the point Ci of the unity interval and has a radius qi. The two marginal directives (left and right) are defined in the rest of the interval with the radius (1- 2qi).The role of these directives is to guide the search in the domains of the variables and quickly locate the global minimum for the problem. In particular, the principal directive biases the search to the parts of the domains where the global minimum is more likely to reside. The two marginal directives guarantee that if the global minimum is not within the effective search interval, the technique can escape being caught within this interval and there for missing the global minimum.

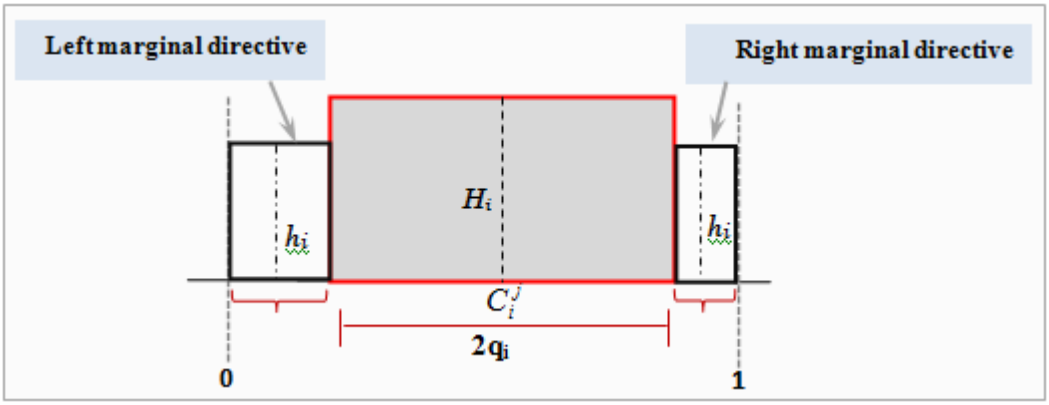


Figure 1: The principal directive (shaded) and two marginal directives on the sides.

The principal and marginal directives are fully defined by their parameters Hi and hi (heights) and the radius (qi). The heights are given by formulas (5).

$$H_i = \frac{\alpha + |\beta - q_i|}{2q_i}$$
$$h_i = \frac{\beta - |\beta - q_i|}{1 - 2(q_i - \varepsilon)} \quad (5)$$

Where α and β belong to the interval [0, 1] and α + β =1.The number ε is a sufficiently small real number (e.g. 1E−300), called the adjustment factor. Given that the value of $q_i \in (0, 0.5]$, the role of the adjustment factor is to guarantee that the denominator of h_i will never be zero and therefore h_i becomes undefined.

Referring to formulas (5), we can easily observe that $H_i*(2q_i) = \alpha + |\beta - q_i|$ is the area of the principal directive. Likewise, $h_i*(1-2(q_i-\varepsilon)) = \beta - |\beta - q_i|$ is the area of the two marginal directives. This means that the area of the principal directive is at least α while the area of the two marginal directives at most β. It is clear that the sum of the areas of the three directives equals 1. Note that

we can increase the area of the principal directive by increasing α and increase of the marginal directives by increasing β. Additionally, decreasing the radius q_i enlarges the area of the principal directive and reduces the area of the two marginal directives.

Definition (1)

1We make two clarifications. 1) We call the part of the interval that is covered by the principal directive the effective search interval because it has the global optimum with a high probability. 2) For the purpose of this paper, we mean by "global optimum" the minimum value (zero) of the aggregate violation.

Let P^M be the probability that some random number $\gamma \in [0, 1]$ belongs to the base of the principal directive and P^L and P^R be the probability that γ belongs to the bases of the left and right directive respectively. We define these probabilities by the following formulas. $P^M = H_i*(2q_i)$ and $P^L = h_i*(c_i-q_i)$ and $P^R = h_i*(1-c_i-q_i)$. Where c_i is the center of the principal directive and q_i is its radius.

Observe that definition (1)bases the probability on the area of the directive. That is, the larger the area of a directive is the larger the likelihood that γ will belong to the base of this directive. Since α and β represent respectively the lower and upper bounds of respectively the principal and marginal directives areas, we can increase the probability of having some random number belong to the base of principal directive by increasing α. Since α + β = 1, this means as the probability of having a random number belong the base of the principal directive increases the probability that this random number belongs to the bases of the marginal directives decreases. In other words, α and β allow us to fully bias random numbers toward the bases principal or marginal directives.

The values for α and β are determined based on the criteria of biasing the probability toward the principal directive without totally ignoring the marginal directives. Since the minimum bound of the principal directive is α, any value for α>0.5 would cause more bias toward the principal directive base (or the effective search interval). During our preliminary experiments to tune the values of α and β, we found that α= 0.75 and β =0.25 are very rational choices because they quickly derive the search to optimal solution.

Given the principal and marginal directives, we define the biased mapping between the interval [0, 1] and the bases of the directives by the mapping rules in Figure2. As the figure shows, mapping a random number $\psi \in [0, 1]$ to the bases of the principal or marginal directives is actually proportional to the area of the directives. Specifically, the algorithm compares ψ_i to the areas of the left and right directives and maps ψ_i to base of either the left directive if the comparison in lines 2, 6 succeeds or to the base of the right directive if the comparison in lines 3, 5 succeeds. It maps ψ_i to the base of the principal directive in lines 4 or 7 otherwise.

Based on the discussion, it is clear that any random number ψ_i either hits the base of the principal or the bases of the marginal directives. That is why we called the algorithm hit-hit algorithm.

(1)	IFC_i< 0.5 THEN
(2)	IF $\psi_i < P^L$ THEN $\gamma_i = \psi_i / h_i$
(3)	ELSEIf $\psi_i < (P^L + P^R)$ THEN $\gamma_i = \psi_i / h_i + 2q_i$
(4)	ELSE $\gamma_i = \frac{2(\psi_i - (Li + Ri))}{H_i} + C_i - q_i$
(5)	ElseIF $\psi_i < P^R$THEN $\gamma_i = (\psi_i / h_i) + C_i + q_i$
(6)	ELSEIF $\psi_i < (P^L + P^R)$ THEN $\gamma_i = (\psi_i / h_i) + C_i + q_i - 1$
(7)	ELSE $\gamma_i = \frac{2(\psi_i - (Li + Ri))}{H_i} + C_i - q_i$

Figure2: the rules of the biased mapping from [0, 1] to principal/marginal directives

3.1 THE TERMINATION CONDITIONS

Let and be the two substitutions of the variables x1, x2, …,x n in two consecutive rounds i and i+1. Suppose also that Ec (Xi) and Ec(Xi+1)are the best values for E c in the rounds i and i+1 respectively. We define the termination conditions as follows.

$$| E^c(X^{i+1}) - E^c(X^i) | < \delta \qquad (6)$$

Where δ is sufficiently small number (e.g. 1E−100). The termination condition means that the value of the aggregate violation does not change for two consecutive rounds *i* and *i*+1. If this condition holds, the search for the global minimum (zero) has reached an equilibrium point and no improvement can be achieved if the algorithm continues the search.

3.2 THE ALGORITHM TECHNICAL DETAILS

Figure 3 shows the technical details of the algorithm. The algorithm searches the domains of the parameters X^i for values that bring the aggregate violation function E^c (X^i)to its minimum (zero). It performs a number of rounds until the termination condition (6)holds. In any round *j*, the algorithm conducts many experiments each of which consists of *m* steps. In each step, it generates *n* random numbers $\psi_i$in the interval [0, 1] using the computer built-in random generator and uses the biased mapping rules (Figure 2) to map each random number to the bases of one of the directives (principal or marginal). The mapping yields new biased random numbers γ_i (lines 9–11). The algorithm uses the biased mapping to focus most of the search in the principal directives since these directives cover parts of the domain sin which the global minimum most likely resides. The biased mapping, however, does not ignore the marginal directives that cover the remaining parts of the domains. Thus, the biased mapping never causes the algorithm to miss the global minimum or get trapped in a local minimum.

The values γ_i's are mapped to the actual domains ([a_i, b_i]) of the variables X^i using the formula in line 12. For each randomly created substitution for the variables X^i, the algorithm computes the value of the criterion function (line 13).If the new substitution satisfies the condition in line 14, this substitution is promising and therefore, the algorithm computes the aggregate violation at this substitution (line 15). If the calculated value better than the previous minimum stored in VE^c, the algorithm stores the new minimum along with the following fundamental information: the

substitution itself and the random values γ_i's from which the substitution was generated(lines 17−19).

After performing *m* steps, the algorithm reduces the radiuses of the principal directives using the reduction formula $\forall i q_i = q_i/d$, where $d>1$ (lines 21-23). The algorithm performs another experiment if at least one of q_i's is still greater than some pre-specified threshold ε (typically ε is less than 1E−60).

We make two important points regarding q_i reduction formula and its effect on the algorithm convergence. First, the reduction factor *d* can be theoretically any real number greater than 1. Greater values of *d* cause the algorithm to converge faster because it quickly reduces the radiuses of the principal directives. We tested our algorithm for only four different values of *d*, namely 1.2, 1.5, 1.8, and 2. All these values cause the algorithm to converge quickly irrespective of the formula that defines the aggregate violation, although the time of convergence becomes relatively

[1]The best value for E^c in round *j* is the one that yields the closest value to zero in this round

shorter as d increased from 1.2 to 1.5 to 2. No tests were made for d greater than 2. Secondly, referring to formula 5, it is clear that as the algorithm further reduces qi, the areas of the principal directives increase and the areas of the marginal directives decrease. This means that more points are mapped to the bases of the principal directives and fewer points to the bases of the marginal directives. In other words, reducingqi plays the major role in greatly focusing the search to the principal directives (where the global minimum most likely is located). Therefore, the effective intervals are thoroughly searched because (1) their radiuses continuously reduce and (2) more points mapped to them

1	**FOR** $i \leftarrow$ 1 to n **DO**
2	C_i = 0.5/**initialize centers of principal directives */
3	$q_i = C_i$/** initialize radiuses of principal directives*/
4	$VE^c = \infty$/** initially the aggregate violation of the constraints is so large */
5	**REPEAT**
6	j = 1
7	**WHILE** (q_i>ε for any i) **DO**
8	**FOR** $k \leftarrow$ 1 to m **DO**/** m steps in each experiment */
9	**FOR** $i \leftarrow$ 1 to n **DO**
10	Generate a random number $\psi_i \in [0, 1]$ /**Random numbers generated*/
11	γ_i = MAP (ψi) using the logic in Figure 2
12	$x_i^j = a_i + \gamma_i (b_i - a_i)$ /**map the random number to actual variable interval*/
13	$F = f(x_1^j, x_2^j, ..., x_n^j)$ /** compute the criterion function f at x_i^j */
14	**IF** $F < VE^c$ **THEN**
15	$F = E^C(x_1^j, x_2^j, ..., x_n^j)$)/**compute the aggregate violation function*/
16	**IF** $F < VE^c$ **THEN**
17	$VE^c = F$
18	$X_{\min}^j = (x_1^j, x_2^j, ..., x_n^j)$
19	$\Omega_{\min}^j = (\gamma_1, \gamma_2, ..., \gamma_n)$
20	**ENDFOR** (K)
21	**FOR** $i \leftarrow$ 1 to n **DO**
22	$q_i = q_i/d$/**reduce the radius qi by d*/
23	Compute H_i and h_i using formulas(5)
24	**ENDWHILE** (qi >ε, for any i)
25	**FOR** $i \leftarrow$ 1 to n **DO**
26	$C_i = \gamma_i$
27	**IF** $C_i <= 0.5$ $q_i = C_i$
28	**ELSE** q_i = 1- C_i
29	j= j + 1
30	**UNTIL** Termination Condition (**6**) holds.
31	**END ALGORITHM**

Keep the so-far best minimum in round j along with the values of its variables and the random numbers that produced this value.

Move the centers of the principal directives to the new points that have resulted in the best minima in round j

Figure 3: the technical steps of the Hit-Hit moving directives algorithm (HHMD-3).

In any subsequent round j, the algorithm uses the information of the previous round j–1 to dynamically adjust the parameters of the directives. In particular, the algorithm moves the centers of the principal directives C_i's to the values γ_i's, which produced the best minimum in the previous round, and calculates the new radiuses q_i's (lines 25–28). The algorithm moves the centers to these values because there is a high probability that the search finds values for the variables that improve the minimum of the aggregate violation in the vicinity of the random numbers γ_i's.

4. PERFORMANCE ANALYSIS

We implemented our algorithm using JAVA programming language. We conducted many experiments using our prototype implementation to evaluate the effectiveness and efficiency of our algorithm. The experiments were conducted on a large number of systems of linear and non-linear equations obtained from benchmarks[5][14][15] and others.The hardware platform is Duo core processor laptop running at 1.7 GH with main memory of 2GB. The operating system is windows 7 (32 bits).

We start our analysisby studying the performance of our algorithm on samples of systems with few but challenging equations obtained from [16][17]. Table 1 shows these systems and the domains of their variables. It shows also the performance of our algorithm measured in terms of both the aggregate violation EC (i.e. the precision of the solutions) and the CPU time in milliseconds (ms).

A sample of Systems		**Performance**	
Equations	**Domain**	**Aggregate violation EC**	**CPU Time (ms)**
$e^{x_1^2} - 8x_1 \sin(x_2) = 0$ $x_1 + x_2 - 1 = 0$ $(x_3 - 1)^3 = 0$	[-10, 10]	7.54E−7	377
$3x_1 - \cos(x_2 x_3) - 0.5 = 0$ $x_1^2 - 625x_2^2 - 0.25 = 0$ $e^{-x_1 x_2} + 20x_3 + (\frac{10\pi - 3}{3}) = 0$	[-10, 10]	8.34E−8	689
$x_1^{x_2} + x_2^{x_1} - 5x_1 x_2 x_3 = 85$ $x_1^3 - x_2^{x_3} - x_3^{x_2} = 60$ $x_1^{x_3} + x_3^{x_1} - x_2 = 2$	[-10, 10]	2.19E−8	3036
$3x_1^3 + 2\sin(x_1)\cos(x_2) - 10.2 = 0$ $2x_2^2 + \cos(x_1)\sin(x_2) - 17.3 = 0$	[-10, 10]	6.35E−13	23
$x_1^2 + 3x_2 - x_3^3 + 10 = 0$ $x_1^2 + x_2^2 + x_3^2 - 6 = 0$ $x_1^3 - x_2^2 + x_3 - 2 = 0$	[-10, 10]	1.5E−12	273
$\frac{0.25}{\pi} x_2 + 0.5x_1 - 0.5\sin(x_1.x_2) = 0$ $\frac{e}{\pi} x_2 - 2.e.x_1 + (1 - \frac{0.25}{\pi}) e^{2x_1} - e) = 0$	[-10, 10]	4.45E−13	51
$4x_1^3 + 4x_1 x_2 + 2x_2^2 - 42x_1 - 14 = 0$ $4x_2^3 + 2x_1^2 + 4x_1 x_2 - 26x_2 - 22 = 0$	[-10, 10]	2.55E−11	12

Table 1: The performance of the algorithm presented in terms of aggregate violation (EC) and CPU time in milliseconds (ms).

According to the performance figures in the table, the algorithm performed really well. This is evident in both the aggregate violation (EC) and the CPU time. The highest aggregate violation is "7.5E□07". The rest of aggregate violations are much less.This means that we have solutions with really high precision. As the CPU time shows, the algorithm required roughly "0.6" second or less for almost all the cases except one case for which the algorithm required about 3 seconds. This problem took 3 seconds because it is known to be so hard to solve.

Table 2 presents benchmarks obtained from [5][14][15]. The table shows the name of the systems of equations (label), the equations themselves, the number of variables, and the domains of the variables. Note, for the economics modeling applications, we considered systemsthat consist of up to 1000 variables while other algorithms consider only up to 20.

Label	System of Equations	Variables	domain
Benchmark *i*1	$x_1 - 0.25428722 - 0.18324757x_4x_3x_9 = 0$ $x_2 - 0.37842197 - 0.16275449x_1x_{10}x_6 = 0$ $x_3 - 0.27162577 - 0.16955071x_1x_2x_{10} = 0$ $x_4 - 0.19807914 - 0.15585316x_7x_1x_6 = 0$ $x_5 - 0.44166728 - 0.19950920x_7x_6x_3 = 0$ $x_6 - 0.14654113 - 0.18922793x_8x_5x_{10} = 0$ $x_7 - 0.42937161 - 0.21180486x_2x_5x_8 = 0$ $x_8 - 0.07056438 - 0.17081208x_1x_7x_6 = 0$ $x_9 - 0.34504906 - 0.19612740x_{10}x_6x_8 = 0$ $x_{10} - 0.42651102 - 0.21466544x_4x_8x_1 = 0$	10	[-2, 2]
Neurophysiology application	$x_1^2 + x_3^2 = 1$ $x_2^2 + x_4^2 = 1$ $x_5x_3^3 + x_6x_4^3 = c1$ $x_5x_1^3 + x_6x_2^3 = c2$ $x_5x_1x_3^2 + x_6x_4^2x_2 = c3$ $x_5x_1^2x_3 + x_6x_2^2x_4 = c4$	6	[-10, 10]
Chemical equilibrium application	$x_1x_2 + x_1 - 3x_5 = 0$ $2x_1x_2 + x_1 + x_2x_3^2 + R_8x_2 - Rx_5 + 2R_{10}x_2^2 + R_7x_2x_3 + R_9x_2x_4 = 0$ $2x_2x_3^2 + 2R_5x_3^2 - 8x_5 + R_6x_3 + R_7x_2x_3 = 0$ $R_9x_2x_4 + 2x_4^2 - 4Rx_5 = 0$ $x_1(x_2 + 1) + R_{10}x_2^2 + x_2x_3^2 + R_8x_2 + R_5x_3^2 + x_4^2 - 1 + R_6x_3 + R_7x_2x_3 + R_9x_2x_4 = 0$ $R = 10, R_5 = 0.193, R_6 = 0.002597/\sqrt{40}$ $R_7 = 0.003448/\sqrt{40}, R_8 = 0.00001799/40$ $R_9 = 0.0002155/\sqrt{40}, R_{10} = 0.00003846/40$	5	[-10, 10]
Combustion Application	$x_2 + 2x_6 + x_9 + 2x_{10} = 10^{-5}$ $x_3 + x_8 = 3 \cdot 10^{-5}$ $x_1 + x_3 + 2x_5 + 2x_8 + x_9 + x_{10} = 5 \cdot 10^{-5}$ $x_4 + 2x_7 = 10^{-5}$ $0.5140437 \cdot 10^{-7}x_5 = x_1^2$ $0.1006932 \cdot 10^{-6}x_6 = 2x_2^2$ $0.7816278 \cdot 10^{-15}x_7 = x_4^2$ $0.1496236 \cdot 10^{-6}x_8 = x_1x_3$ $0.6194411 \cdot 10^{-7}x_9 = x_1x_2$ $0.2089296 \cdot 10^{-14}x_{10} = x_1x_2^2.$	10	[-10, 10]

Benchmark *i*4	$0 = x_1^2 - 0.25428722 - 0.18324757x_4^2x_3^2x_9^2$ $0 = x_2^2 - 0.37842197 - 0.16275449x_1^2x_{10}^2x_6^2$ $0 = x_3^2 - 0.27162577 - 0.16955071x_1^2x_2^2x_{10}^2$ $0 = x_4^2 - 0.19807914 - 0.15585316x_7^2x_1^2x_6^2$ $0 = x_5^2 - 0.44166728 - 0.19950920x_7^2x_6^2x_3^2$ $0 = x_6^2 - 0.14654113 - 0.18922793x_8^2x_5^2x_{10}^2$ $0 = x_7^2 - 0.42937161 - 0.21180486x_2^2x_5^2x_8^2$ $0 = x_8^2 - 0.07056438 - 0.19612740x_1^2x_7^2x_6^2$ $0 = x_9^2 - 0.34504906 - 0.19612740x_{10}^2x_6^2x_8^2$ $0 = x_{10}^2 - 0.42651102 - 0.21466544x_4^2x_8^2x_1^2$	10	[-1, 1]
Benchmark *i*5	$x_1^2 - 0.25428722 - 0.18324757x_4^3x_3^3x_9^3 + x_3^4x_9^7 = 0$ $x_2^2 - 0.37842197 - 0.16275449x_1^3x_{10}^3x_6^3 + x_{10}^4x_6^7 = 0$ $x_3^2 - 0.227162577 - 0.16955071x_1^3x_2^3x_{10}^3 + x_2^4x_{10}^7 = 0$ $x_4^2 - 0.19807914 - 0.15585316x_7^3x_1^3x_6^3 + x_1^4x_6^7 = 0$ $x_5^2 - 0.44166728 - 0.19950920x_7^3x_6^3x_3^3 + x_6^4x_3^7 = 0$ $x_6^2 - 0.14654113 - 0.18922793x_8^3x_5^3x_{10}^3 + x_5^4x_{10}^7 = 0$ $x_7^2 - 0.42937161 - 0.21180486x_2^3x_5^3x_8^3 + x_5^4x_8^7 = 0$ $x_8^2 - 0.07056438 - 0.19612740x_1^3x_7^3x_6^3 + x_7^4x_6^7 = 0$ $x_9^2 - 0.34504906 - 0.19612740x_{10}^3x_6^3x_8^3 + x_6^4x_8^7 = 0$ $x_{10}^2 - 0.42651102 - 0.21466544x_4^3x_8^3x_1^3 + x_8^4x_1^7 = 0$	10	[-1, 1]
Benchmark *i*2	$x_1 - 0.24863995 - 0.19594124 x_7 x_{10} x_{16} = 0$ $x_2 - 0.87528587 - 0.05612619 x_{18} x_8 x_{11} = 0$ $x_3 - 0.23939835 - 0.20177810 x_{10} x_7 x_{11} = 0$ $x_4 - 0.47620128 - 0.16497518 x_{12} x_{15} x_1 = 0$ $x_5 - 0.24711044 - 0.20198178 x_8 x_9 x_{16} = 0$ $x_6 - 0.33565227 - 0.15724045 x_{16} x_{18} x_{11} = 0$ $x_7 - 0.13128974 - 0.12384342 x_{12} x_{13} x_{15} = 0$ $x_8 - 0.45937304 - 0.18180253 x_{19} x_{15} x_{18} = 0$ $x_9 - 0.46896600 - 0.21241045 x_{13} x_2 x_{17} = 0$ $x_{10} - 0.57596835 - 0.16522613 x_{12} x_9 x_{13} = 0$ $x_{11} - 0.56896263 - 0.17221383 x_{16} x_{17} x_8 = 0$ $x_{12} - 0.70561396 - 0.23556251 x_{14} x_{11} x_4 = 0$ $x_{13} - 0.59642512 - 0.24475135 x_7 x_{16} x_{20} = 0$ $x_{14} - 0.46588640 - 0.21790395 x_{13} x_3 x_{10} = 0$ $x_{15} - 0.10607114 - 0.20920602 x_1 x_9 x_{10} = 0$ $x_{16} - 0.26516898 - 0.21037773 x_4 x_{19} x_9 = 0$ $x_{17} - 0.20436664 - 0.19838792 x_{20} x_{10} x_{13} = 0$ $x_{18} - 0.56003141 - 0.18114505 x_6 x_{13} x_8 = 0$ $x_{19} - 0.92894617 - 0.04417537 x_7 x_{13} x_{16} = 0$ $x_{20} - 0.57001682 - 0.17949149 x_1 x_3 x_{11} = 0$	20	[-1, 2]
Benchmark *i*3	The same as Benchmark *i*2, but different interval for the variables.	20	[-2, 2]
Economics modeling application	$(x_k + \sum_{i=1}^{n-k-1} x_i x_{i+k}) x_n - c_k = 0,\ (1 \le k \le n-1)$ $\sum_{l=1}^{n-1} x_l + 1 = 0$	**5, 20, 50, 100, 200, 300, 500, 1000**	[-10, 10]
Brown	$2x_1+x_2+x_3+x_4+x_5-6=0$ $x_1+2x_2+x_3+x_4+x_5-6=0$ $x_1+x_2+2x_3+x_4+x_5-6=0$ $x_1+x_2+x_3+2x_4+x_5-6=0$ $x_1+x_2+x_3+x_4+2x_5-1=0$	5	[0, 3]

Table 2: Systems of equations benchmarks.

Solutions		Performance	
System	**Variable values**	**EC**	**Time (ms)**
Benchmark -*i*1	x0=0.2638927436500653 x1=0.380317066392982 x2=0.2804250886510391 x3=0.2132412203761027 x4=0.44438239437353655 x5=0.14944883430013256 x6=0.433092499920527 x7=0.06746428319148512 x8=0.3469624642044846 x9=0.39763460651053917	0.0081	212
Neurophysiology application	x0=0.773073454805388 x1=0.5765805512504851 x2 = 0.6343165089155978 x3 =–0.8170403098469485 x4=3.7481129311345285E-13 x5=3.822719918389339E-12	1.79E–11	609
Chemical equilibrium application	x0=0.050958721835229426 x1=1.748330139399421 x2=0.2715118958827709 x3=-0.8464707181498703 x4=0.03582437318962839	0.0325	670
Combustion Application	x0=7.885401512197632E-6 x1=-6.287224145751225E-7 x2=1.7216365890249108E-5 x3=-5.111503458721245E-6 x4=7.830654080720478E-6 x5=1.526760066106192E-5 x6=7.555224350141998E-6 x7=1.2783895904533438E-5 x8=-1.2757429127319142E-5 x9=-3.574492433600085E-6	2.66E–9	297
Benchmark i4	x0=-0.5077723973445423 x1=0.615256474526175 x2=-0.5279178631279993 x3=0.44796092038237223 x4=0.667271196118725 x5=0.38625078626268783 x6=0.6572653126601917 x7=-0.2712236420769336 x8=0.5881911765836603 x9=-0.6537033315232845	0.00281	203
Benchmark i5	x0=-0.501902791893752 x1=0.6146257966533131 x2=-0.5293855157439512 x3=0.44456590561843945 x4=0.6644074507239301 x5=0.39586363376169453 x6=0.6550747173991653 x7=-0.2642942057821429 x8=0.5874639200667668 x9=-0.6531478452401633	9.52E–5	190
Benchmark i2	x0=0.22065654006016544 x1=0.6679777509246896 x2=0.25308693662292403	0.0655	834

	x3=0.47593660216437894 x4=0.3108760076620436 x5=0.3420931417977098 x6=0.14056743503862323 x7=0.4777925690182352 x8=0.49065449430132757 x9=0.6113946257954175 x10=0.5808035003270997 x11=0.7524629539375702 x12=0.5991667120015798 x13=0.48572097680262116 x14=0.12027902783234179 x15=0.304471847407779 x16=0.31235184663156956 x17=0.5779178263400047 x18=0.9308157797622583 x19=0.5225894624268062		
Benchmark i3	x0=0.25379644520356415 x1=0.883043903677712 x2=0.24808502738095983 x3=0.4768828006356207 x4=0.2618446169814348 x5=0.3500860164569821 x6=0.1379779091289688 x7=0.47131924360702326 x8=0.49682215346828507 x9=0.6122399213932219 x10=0.5097531602011571 x11=0.7334022329597554 x12=0.6024825313683912 x13=0.48527132307332055 x14=0.12222150380249319 x15=0.3115283407139193 x16=0.24650097840355523 x17=0.5780392185924179 x18=0.9300901851505055 x19=0.5757777139250799	`0.06911`	452
Economics modeling application(5)	x0=-0.35442260333596565 x1=-0.23692868742035955 x2=-0.14071376443385475 x3=-0.26793494480927116 x4=-5.844214001626824E-13	9.21E−13	97
Economics modeling application(20)	x0=2.6989965817847406E-11 x1=-1.0128786698260228E-11 x2=7.263523116307624E-12 x3=2.5082158572331537E-12 x4=-2.169464607959526E-11 x5=7.506884003305458E-12 x6=-9.341860618405917E-12 x7=-6.4712679659351124E-12 x8=3.375077994860476E-12 x9=-2.107469754264457E-11 x10=-1.8065549056700547E-12 x11=-2.1234569658190594E-11 x12=-6.339817559819494E-12 x13=1.8850698779715458E-11 x14=-1.1876721828230075E-11	7.10E−15	163

	x15=-3.858247055177344E-12 x16=-1.8186341321779764E-11 x17=3.597122599785507E-12 x18=-1.4916068380443903E-11 x19=9.329426120530115E-12		
Economics modeling application(30)	x0=1.438849039914203E-13 x1=-3.126388037344441E-13 x2=6.288303211476887E-13 x3=5.258016244624741E-13 x4=-6.057376822354854E-13 x5=-3.979039320256561E-13 x6=4.156675004196586E-13 x7=3.0020430585864233E-13 x8=-6.963318810448982E-13 x9=1.2789769243681803E-13 x10=6.394884621840902E-14 x11=2.7000623958883807E-13 x12=6.750155989720952E-14 x13=-1.5276668818842154E-13 x14=-5.080380560684716E-13 x15=1.1723955140041653E-13 x16=-6.377121053446899E-13 x17=-3.2152058793144533E-13 x18=-4.4231285301066237E-13 x19=1.6697754290362354E-13 x20=6.359357485052897E-13 x21=-5.595524044110789E-13 x22=-6.856737400084967E-13 x23=-3.126388037344441E-13 x24=-1.1546319456101628E-13 x25=3.907985046680551E-14 x26=-1.580957587066223E-13 x27=-7.105427357601002E-15 x28=5.666578317686799E-13 x29=3.019806626980426E-13	1.77E−15	249
Economics modeling application(100)	-	1.77E−15	645
Economics modeling application(200)	-	7.68E−14	2,046
Economics modeling application(300)	-	3.36E−14	2,938
Economics modeling application(500)	-	3.16E−14	11,099
Economics modeling application(1000)	-	5.13E−14	32,971
Brown	x0=1.0000000000000413 x1=0.9999999999999537 x2=0.9999999999999963 x3=1.0000000000003915 x4=0.9999999999995891	5.11E−13	87

Table 3: The performance numbers of our algorithm for the benchmarks in Table 2.

Table 3 shows the performance of our algorithm for the systems in Table 2. The performance is presented in terms of values of the variables, the aggregate violations, and the CPU time in milliseconds.We did not show the values of the variables for the economics modeling application

when the number of the variables exceeds 30. Basically the values for these variables are pretty the same as for 30 variables or fewer.

Based on the figures in Table 3, our algorithm produced solutions with high precision. The aggregate violations arexE−9, xE−11, xE−14,and xE−15 (x some real number) for most of the systems. For some of the systems, the accuracy was ranging from 7.43E-4 (Benchmark i5) to 0.06911 (Benchmark i3). It is worth mentioning that our algorithm achieved highprecision for hard systems of equations: Chemical equilibrium application and Benchmark i2.

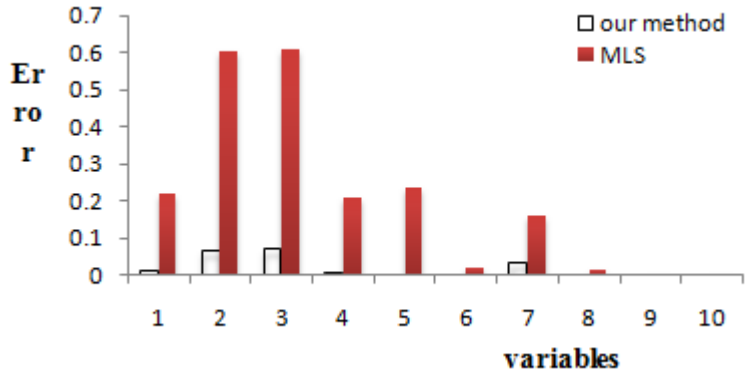
To further discus the performance of our algorithm, we compared itwith highly effective algorithms in literature. The comparison includes both the CPU time and the precision of the solution (in terms of aggregate violation). Table 4 compares our algorithm and the Modified Line Search (MLS) [4]. The entry "-" in some of the table's cells means that there is no reported performance for the corresponding system of equations.

Benchmarks (variables)	**Our algorithm**		**MLS (Modified Line Search)**	
	EC	**Time (ms)**	***En***	**Time (ms)**
Benchmark *i*1 (**10**)	0.0081	212	0.22084	516
Benchmark *i*2 (**20**)	0.0655	834	0.60634	1,297
Benchmark *i*3 (**20**)	0.06911	452	0.61134	1,016
Benchmark *i*4 (**10**)	0.00281	203	0.20734	953
Benchmark *i*5 (**10**)	9.52E-5	190	0.23610	1,000
Neurophysiology application (**6**)	1.79E−11	609	0.01998	922
Chemical equilibrium application (**5**)	0.0325	662	0.16072	922
Combustion Application (**10**)	2.66E-9	297	0.01506	860
Brown (**5**)	5.11E−13	87	-	-
Economics modeling application(**5**)	9.21E−13	97	-	-
Economics modeling application(**10**)	5.10E−15	132	0.00294	266
Economics modeling application(**20**)	7.10E−15	163	0.00459	1,078
Economics modeling application(**30**)	1.77E−15	249	-	-
Economics modeling application(**100**)	1.77E−15	645	-	-
Economics modeling application(**200**)	7.68E−14	2,046	-	-
Economics modeling application(**300**)	3.36E−14	2,938	-	-
Economics modeling application(**500**)	3.16E−14	11,099	-	-
Economics modeling application(**1000**)	5.13E−14	32,971	-	-

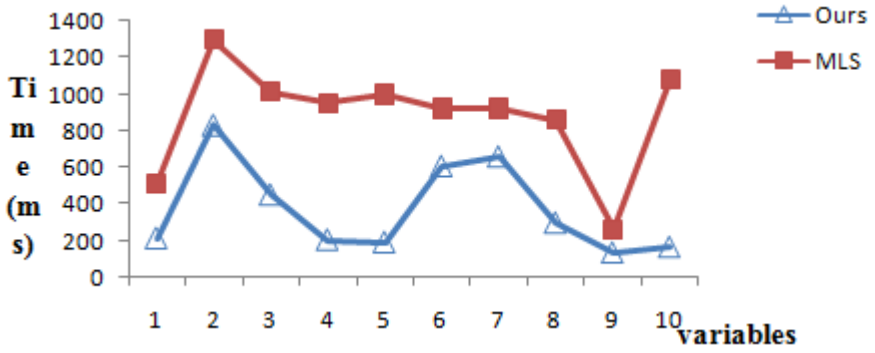
Table 4: The performance of our technique versus MLS.

Table 4 and Figure 4 indicate that our algorithm outperformed MLS in both solution precisionand the CPU time. Figure 4(a) visually compares the errors in the solutions produced by our algorithm and MLS. Our algorithm clearly produced solutions with much smaller errors than MLS. Referring to Table 4, our algorithm produced a solution for the problem Benchmark i1 with an aggregate violation of 0.0081while MLS produced a solution with accumulative error of 0.22084 for the same problem. Other examples in Table 4 show even much betterprecision in the solutions produced by our algorithm. For instance, our algorithm produced solutions for Combustion

Application (10) and Economics modeling application(10) with aggregate violations of respectively2.66E-9 and 5.10E-15 while MLS produced solutions with violations of respectively 0.01506 and 0.00294 for the same problems.



(a): Comparing the amount of error in solution of each of the problems. The higher the bar the larger the error and the worse the solution is.



(b): The time performance for our algorithm and MLS.

Figure 4: The plot of performance numbers of our technique versus MLS

Figure 4(b)visually compares the timing numbers of our algorithm versus those of MLS. Generally speaking, our algorithm performed better than MLS. Our technique required much less time to produce solutions for all the problems than MLS did. Referring to Table 4, our algorithm required only 190 milliseconds to produce a solution for Benchmark *i*5 problem while MLS required 1000 milliseconds. Other numbers in the same table clearly show that our technique required less time in all the problems.

The authors in [6] proposed an effective algorithm called Algorithm 2.4. They compared their algorithm against a large number of algorithms and showed its superiority to the others. The comparison is based on the following two problems [6].

Problem (1)

$$[E_i(x_2\sin\psi_i - x_3) - F_i(x_2\sin\phi_i - x_3)]^2 + [F_i(1+x_2\cos\phi_i) - E_i(x_2\cos\psi_i - 1)]^2 -$$
$$[(1+x_2\cos\phi_i)(x_2\sin\psi_i - x_3)x_1 - (x_2\sin\phi_i - x_3)(x_2\cos\psi_i - x_3)x_1]^2 = 0$$

where,

$$E_i = x_2(\cos\phi_i - \cos\phi_0) - x_2x_3(\sin\phi_i - \sin\phi_0) - (x_2\sin\phi_i - x_3)x_1$$
$$F_i = -x_2\cos\psi_i - x_2x_3\sin\psi_i + x_2\cos\psi_0 + x_1x_3 + (x_3 - x_1)x_2\sin\psi_0$$
$$(i = 1, 2, 3)$$

Where the values of ϕi and ϕi are given in the following table [6].

i	ψ_i	ϕ_i
0	1.3954170417470 90114	1.7461756494150842271
1	1.7444828545735749268	2.0364691127919609051
2	2.0656234369405315683	2.3390977868265978920
3	2.1600678478912500533	2.1600678409809341550

Problem (2)

The Combustion Application problem as defined in Table 2. They used the interval [0, 1] for all the variables instead of [-10, 10].

	Our algorithm		**Algorithm 2.4 [8]**	
Test Case	**EC**	**Time (ms)**	**Accuracy**	**Time (ms)**
Problem (1)	4.3E−72	62	3.6E−90	125
Problem (2)	1.6E−52	211	5.9E−89	484

Table 5: The performance of our algorithm versus Algorithm 2.4

As Table 5 shows, our algorithm outperformed Algorithm 2.4 in terms of CPU time. We used different measure to estimate the aggregate violation. Therefore, it is not possible to compare the precision.

Authors in [21] presented conjugate direction flower pollination algorithm (CDFPA)and compared the performance of this algorithm with other algorithms such as flower pollination algorithm (FPA) and conjugate direction (CD) method. Based on the reported results in [21], the CDFPA performed better than the others. Table 6 shows the cases and the performance of our algorithm versus CDFPA. The entry "-" means no reported performance numbers and "□" means close but not equal to. It is clear from the table that our algorithm performed better than CDFPA. Our algorithm found the exact solution in all cases while CDFPA found the exact solution for only one case and approximated solution for the rest. No timing figures reported for CDFPA to compare against.

Cases		**Our algorithm**		**CDFPA**	
		EC	**CPU time (ms)**	**Precision**	**CPU time**
Case1	$x_1 + 0.99x_2 = 1$ $0.99x_1 + 0.98x_2 = 1$	0	1011	0	-
Case 2	$H \cdot x = b$ $h_{ij}=1/(i+j-1)$	0	205	≈ 0	-

	$b_i = \sum_{j=1}^{5} h_{ij}.j$, i=1..5				
Case 3	$200x_1 + 101x_2 = 100$ $400x_1 + 201x_2 = -100$	0	672	≈ 0	-
Case 4	$x_1^2 - 2x_1 + 3x_2 = -1$ $2x_1^2 - 3.9999x_1$ $+6.0001x_2 = -1.9999$	0	13615	≈ 0	-

Table 6: the performance of our algorithm versus CDFPA

Finally we compare our algorithm with the one proposed in [20]. Table 7 shows the systems of equations and the performance of our algorithm compared to that of the algorithm in [20]. Clearly our approach produced better precision. No timing numbers are reported in [20] to compare against.

Equations	**Interval**	**Error in solution**	
		Our method	**Algorithm[20] "MinError"**
$2x_1 + x_2 + x_3 + x_4 + x_5 - 6 - 0$ $x_1 + 2x_2 + x_3 + x_4 + x_5 - 6 = 0$ $x_1 + x_2 + 2x_3 + x_4 + x_5 - 6 = 0$ $x_1 \mid x_2 \mid x_3 \mid 2x_4 \mid x_5 \quad 6 = 0$ $x_1x_2x_3x_4x_5 - 1 = 0$	**[-10, 10]**	4.5E-9	5.09E−05
$x_1 + \frac{x_2^2 x_4 x_6}{4} + 0.75 = 0$ $x_2 + 0.405e^{1+x_1x_2} - 1.405 = 0$ $x_3 - \frac{x_4x_6}{2} + 1.5 = 0$ $x_4 - 0.605e^{1-x_3^2} - 0.395 = 0$ $x_5 - \frac{x_2x_6}{2} + 1.5 = 0$ $x_6 - x_1x_5 = 0$	**[-5, 5]**	**0**	**0**
$x_1x_2 - (x_1 - 2x_3)(x_2 - 2x_3) - 165 = 0$ $\frac{x_1x_2^3}{12} - \frac{(x_1 - 2x_3)(x_2 - 3x_3)^3}{12} - 9369 = 0$ $\frac{2(x_2 - x_3)^2(x_1 - x_3)^2 x_3}{x_2 + x_1 - 2x_3} - 6835 = 0$	**[-40,40]**	**5.8E-10**	**8.3E−04**
$e^{x_1 - x_2} - \sin(x_1 + x_2) = 0$ $x_1^2x_2^2 - \cos(x_1 + x_2) = 0$	**[-10, 10]**	**3.1E-14**	**3.93E−07**

Table 7: The performance of our algorithm verses [20].

Note on the Constraint Violation Measure

Our technique outperformed other techniques in terms of time, precision (measured in terms of the aggregate violation EC), or both. Furthermore, we measure the aggregate violation as the sum

of absolute values of the violations in each individual constraint while others measure the violation in terms of square root of the sum of the square of the individual violation. Our measure of violation is therefore more rigorous than theirs. That is because it can be easily shown that

$$\sum_{i=1}^{n} | \rho^{c_i}(X_i) | \geq \sqrt{\sum_{i=1}^{n} (f_i(X_i))^2} \quad \forall X_i.$$

According to this, our technique would have shown even significantly higher precision if we used the same constraint violation measure as the others

5. CONCLUSION AND FUTURE WORK

This paper proposed effective approach to find solutions for systems of linear and non-linear equations. Our approach transforms the problem of solving systems of equations into an optimization problem. Our transformation results in an aggregate violation function whose global minimum is the solution for the system of equations. The transformation defines also a criterion function that effectively determines when the aggregate function must be evaluated.

The paper proposed random-guided algorithm to find the solution for the optimization problem and therefore to the corresponding system of equations. The algorithm uses one principal and two marginal directives that effectively search for the global minimum of the optimization problem. These three directives are augmented with biased mapping rules that enable the algorithm to focus the search in the parts of the domain that most likely contain the global minimum without ignoring the other parts of the domains that may contain the minimum.

We conducted many experiments to evaluate our approach. The experiments showed that our proposed technique is very effective in finding solutions with high precisionand high speed. We also compared our algorithm with state-of art algorithms. Our algorithm has better performance in terms of both the solution precision and the CPU time.

We have two directions for future work. First, we would like to conduct more experiments to analyze the effect of tuning the parameters of the algorithm (α,β, d) on its performance. Second, we would like to check the advantage of parallelizing the algorithm so that we can run more than one version of the algorithm using different centers for the directives.

REFERENCES

[1] A. Pourrajabian, R. Ebrahimi, M. Mirzaei &M.Shams ,(2013)"Applying Genetic Algorithms for Solving Nonlinear Algebraic Equations", Applied Mathematics and Computation, Vol. 219, No. 24, pp. 11483–11494.

[2] Angel Fernando Kuri-morales&Autónomo México, (2003) "Solution of simultaneous non-linear equations using genetic algorithms", WSEAS Transactions on Systems, Vol. 1, No. 2, pp. 44–51.

[3] S. Effati& A. R. Nazemi, (2005) "A New Method for Solving a System of Nonlinear Equations", Applied Mathematics and Computations, Vol. 168, No. 2, pp. 877–894.

[4] S. Abbasbandy, P.Bakhtiari, A. Cordero&T.Lotfi, (2016)"New efficient methods for solving nonlinear systems of equations with arbitrary even order", Applied Mathematics and Computation, Vol. 103, pp. 94–103.

[5] C. Grosan& A. Abraham, (2008)"A New Approach for Solving Nonlinear Equations Systems", IEEE transactions on systems, man, and cybernetics, Vol. 38, No. 3, pp. 698–714.

[6] M. Waseem, M. A. Noor, & K. I. Noor, (2016)"Efficient Method for Solving a System of Nonlinear Equations", Applied Mathematics and Computation, Vol. 275, pp. 134–146

[7] F. Awawdeh, (2010) "On New Iterative Method For Solving Systems Of Nonlinear Equations", Numerical Algorithms, Vol. 54, No.3, pp. 395–409.

[8] M.A. Noor, M. Waseem, K.I. Noor & M.A. Ali, (2015) "New Iterative Technique for Solving Nonlinear Equations", Applied Mathematics Computation, Vol. 265, pp. 1115–1125.

[9] M. Y. Waziri, W.J. Leong, M. A. Hassan &M. Monsi, (2010) "A New Newton Method with Diagonal Jacobian Approximation for Systems of Non-Linear Equations",Journal of Mathematics and Statistics Science Publication, Vol. 6, No. 3, pp. 246–252.

[10] F. S. Emmanuel, (2015)"On Some Iterative Methods for Solving Systems of Linear Equations", Computational and Applied Mathematics Journal, Vol. 1, No. 2, pp. 21–28.

[11] B.K. Ibrahim, (2010)"A Survey of three Iterative Methods for the Solution of Linear Equations", IJNM. Vol. 5, No. 1, pp. 153–162.

[12] P. Y. Nie, (2004)"A Null Space Method for Solving System of Equations", Applied Mathematics and Computations, Vol. 149, No. 1, pp. 215–226.

[13] M. K. A. Ariyaratne, T. G. I. Fernando & S. Weerakoon, (2016) "A Self-tuning Modified Firefly Algorithm to Solve Univariate Nonlinear Equations with Complex Roots", IEEE Congress on Evolutionary Computation (CEC), Vancouver, BC, pp. 1477–1484.

[14] C. Grosan, A. Abraham, & V. Snasel, (2012)"Solving Polynomial Systems Using A Modified Line Search Approach", International Journal of Innovative Computing, Information and Control, Vol. 8, No. 1, pp. 501–526.

[15] P. Van Hentenryck, D. McAllester& D. Kapur, (1997) "Solving Polynomial Systems using a Branch and Prune Approach", SIAM Journal of Numerical Analysis, Vol.34, No.2, pp.797–827.

[16] Y. Li, Y. Wei& Y. Chu, (2015) "Research on Solving Systems of Nonlinear Equations Based on Improved PSO", Mathematical Problems in Engineering, Vol. 2015, 13 pages.

[17] J. L. Hueso, E. Mart´ınez& J. R. Torregrosa, (2009)"Modified Newton's Method for Systems of Nonlinear Equations with Singular Jacobian", Journal of Computational and Applied Mathematics, Vol. 224, No. 1, pp. 77–83.

[18] C. Qu& W. He, (2015) "A Double Mutation Cuckoo Search Algorithm for Solving Systems of Nonlinear Equations", International Journal of Hybrid Information Technology Vol.8, No.12, pp. 433–448.

[19] E. Pourjafari&H. Mojallali, (2012)"Solving Nonlinear Equations Systems with a New Approach Based on Invasive Weed Optimization Algorithm and Clustering", Swarm and Evolutionary Computation, Vol. 4, pp. 33–43.

[20] K. A. Sidarto& A. Kania, (2015) "Finding All Solutions of Systems of Nonlinear Equations using Spiral Dynamics Inspired Optimization with Clustering", Journal of Advanced Computational Intelligence and Intelligent Informatics, Vol.19 No.5, pp. 697–707.

[21] M. Abdel-Baset& I. M. Hezam, (2016) "A Hybrid Flower Pollination Algorithm for Solving Ill-Conditioned Set of Equations", International Journal of Bio-Inspired Computation, Vol. 8, No. 4, pp. 215–220.

[22] E. Rushdy, M. Abdel-Baset& I. M. Hezam, (2017)"Solving Systems of Nonlinear Equations via Conjugate Direction Flower Pollination Algorithm", International Journal of Computing Science and Mathematics, Vol. 8, No. 3, pp. 201–209.

[23] X. Wang & X. Fan, (2016) "Two Efficient Derivative-Free Iterative Methods for Solving Nonlinear Systems", Algorithms Journal, Vol. 9, No. 1, pp. 9–14.

AUTHOR

Dr. Muhammed Jassem Al-Muhammed holds PhD in computer science, Brigham Young University, USA, 2007. I joined American university of Madaba (Jordan), faculty of information technology in 2013. Prior to this, I worked as a faculty member in Damascus University and international university for science and technology, Syria. Al-Muhammed has published many conference and journal papers. Most of his research interest is in computer security (including cryptography, access control, and so on), problem optimization, and semantic web. Al-Muhammed has also authored many books in different computer fields.