MALAYSIAN JOURNAL OF COMPUTING AND APPLIED MATHEMATICS (2020) 3 (2) 25-35

DOI: *https://doi.org/10.37231/myjcam.2020.3.2.49*

Article: Received 06.07.2020; Accepted 04.11.2020; Published 05.12.2020

A SIMPLE CONJUGATE GRADIENT METHOD VIA BROYDEN'S APPROACH FOR SOLVING LARGE-SCALE SYSTEMS OF NONLINEAR EQUATIONS

M K Dauda[a], A S Magaji[a], U S Shehu[b], M A Usman[c], M Y Waziri[d]

[a]*Department of Mathematical Sciences, Kaduna State University, Nigeria.*
[b]*Department of Mathematics and Statistics, Kaduna Polytechnic, Kaduna Nigeria.*
[c]*Department of Computer Sciences, Shehu Idris College of Health Science and Technology, Kaduna Nigeria*
[d]Department of Mathematical Sciences, Bayero University, Kano, Nigeria.
mkdfika@kasu.edu.ng

Abstract: Quasi Newton's method is among the most promising algorithm for solving systems of nonlinear equations. In this family, Broyden's method update is the famous. This paper presents a simple conjugate gradient (CG) method for solving large-scale systems of nonlinear equations via memoryless Broyden's approach. The attractive attribute of this method is due to its low memory requirements, global convergence properties and simple implementation. Under suitable conditions, the proposed method converges globally. Numerical performance of the proposed method are compared with the well-known conjugate gradients (CG) methods using benchmarks problems and are based on number of iteration and the CPU time. These are coded using MATLAB and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on an Acer Aspire with a Windows 7 operating system. The report demonstrate that the proposed method is reliable, efficient and competitive.

Keywords: Conjugate Gradient, Broyden's Update, Global Convergence, Nonlinear Equations, Approximation.

1. INTRODUCTION

Let us consider the system of nonlinear equations

$$g(x) = 0, x \in R^n \tag{1}$$

where $g: \mathrm{R}^{\mathrm{n}} \to \mathrm{R}^{\mathrm{n}}$ is continuously differentiable mapping. Suppose that for each $x \in \mathrm{R}^{\mathrm{n}}$, the Jacobian $g'(x)$ of g at x is symmetric. The prominent method for finding the solution of (1) is the classical Newton's method which generates a sequence of iterates $\{x_k\}$ from a given initial point $x_o$via

$$x_{k+1} = x_k - \left(g'(x_k)\right)^{-1} g(x_k), \quad k = 0,1,2,\cdots. \tag{2}$$

where $g'(x_k)$ is the Jacobian matrix of g at x_k.

The attractive features of this method are rapid convergence and easy to implement. Nevertheless, Newton's method requires the computation of the matrix entails the first order derivatives of the systems. In practice, computations of some functions derivatives are difficult and sometimes they are not available or could not be done precisely. In this case, Newton's methods cannot be applied directly [4,9 10,14,16].

The CG methods for solving nonlinear systems of equations generates an iterative point $\{x_k\}$ from initial given point x_o via

$$x_{k+1} = x_k + \alpha_k d_k \tag{3}$$

where $\alpha_k > 0$ is attained via line search and direction d_k are obtained using

$$d_{k+1} = \begin{cases} -g(x_k) & if\ k = 0 \\ -g(x_{k+1}) - \beta_k d_k\ if\ k \geq 0 \end{cases} \quad (4)$$

where β_k is term as conjugate gradient parameter and d_k is assumed to be a decent direction. Different conjugate gradients methods correspond to different choices for the scalar$\beta_k$exists. The following are some examples of conjugate gradients parameters [20].

$$\beta_k^{CD} = \frac{g_{k+1}^T g_{k+1}}{g_k^T d_k},\ \beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{y_k^T s_k},\ \beta_k^{PR} = \frac{g_k^T y_k}{g_k^T g_k},$$
$$\beta_k^{LS} = \frac{y_k^T g_{k+1}}{g_k^T d_k},\qquad \beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k},\ \beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k},$$

All these methods satisfy the conjugacy condition, for more details on CG, ([6,4,17]). Moreover, some substantial efforts have been made by numerous researchers [7,8,11,13,18,19,20] in order to eliminate the well-known shortcomings of Newton's method for solving nonlinear systems of equations. Most of these modifications of Newton's method still have some shortfalls as Newton's counterpart. For example, Broyden's method need to store an $n \times n$ matrix and their floating points operations are $O(n^2)$ respectively. Note that the most crucial initiative common to all these efforts is forming and storing a full matrix of Jacobian approximation (directly or indirectly), which can be very costly when handling large scale system, this leads to the idea of this paper. In this paper, we proposed a simple CG algorithm for solving large scale systems of nonlinear equations by a modification of the Broyden's inverse approximations restarted as identity matrix at every step. The method maintained low memory requirement, global convergence properties and simple to implement. Convergence results are presented in Section 3. Some numerical results are reported in Section 4. Finally, conclusions are made in Section 5.

2. A SIMPLE CONJUGATE GRADIENT METHOD VIA BROYDEN'S UPDATE (SCG)

This section presents a simple CG method for solving large-scale systems of nonlinear equations via memoryless Broyden's update. In general, quasi Newton method is an iterative method that generates a sequence of points $\{x_k\}$ from a given initial point x_o via the following

$$x_{k+1} = x_k - \alpha_k(\beta_k)^{-1} g(x_k), \quad k = 0,1,2,\ldots \quad (5)$$

where β_k is an approximation to the Jacobian which is updated at each iteration for $k = 0,1,2,\cdots$, the updated matrix β_{k+1} is chosen in such a way that it satisfies the secant equation

$$B_{k+1} s_k = y_k\ , s_k = x_{k+1} - x_k \text{ and } y_k = g(x_{k+1}) - g(x_k) \quad (6)$$

from (5), the updated formula for the Broyden matrix B_k is given as [1,8]:

$$B_{k+1} = B_k + \frac{(y_k - B_k s_k) s_k^T}{s_k^T s_k} \quad (7)$$

The inverse version of Broyden's method is represented as

$$B_{k+1}^{-1} = B_k^{-1} + \frac{(s_k - B_k^{-1} y_k) y_k^T}{y_k^T y_k} \quad (8)$$

Alternatively, (8) can be rewritten as follows, whenever $B_{k+1}^{-1} = Q_{k+1}$

$$Q_{k+1} = Q_k + \frac{(s_k - Q_k y_k) y_k^T}{y_k^T y_k} \quad (9)$$

Here we consider the matrix Q_{k+1} as Broyden's update. The ultimate motivation of using this update is due to its quality Jacobian approximation, good convergence properties and simple implementation. Recall that from equation (9) and by letting

$$Q_k \approx I \tag{10}$$

This transforms (9) to

$$Q_{k+1} = \mathrm{I} + \frac{(\mathrm{s}_k - I\mathrm{y}_k)\mathrm{y}_k^T}{y_k^T y_k} \tag{11}$$

We further multiply both side of (11) by $g(x_{k+1})$ to obtain

$$Q_{k+1}g(x_{k+1}) = g(x_{k+1}) + \frac{(\mathrm{s}_k - I\mathrm{y}_k)\mathrm{y}_k^T g(x_{k+1})}{y_k^T y_k} \tag{12}$$

Observe from (12), the quantity $Q_{k+1}g(x_{k+1})$ is the direction d_{k+1} [8]. Thus, can be written as

$$d_{k+1} = Q_{k+1}g(x_{k+1}) \tag{13}$$

Hence, we have

$$d_{k+1} = g(x_{k+1}) + \frac{(\mathrm{s}_k - \mathrm{y}_k)\mathrm{y}_k^T g(x_{k+1})}{y_k^T y_k} \tag{14}$$

Equation (14) can be rewritten as

$$d_{k+1} = g(x_{k+1}) + \theta_k \rho_k \text{ with } \theta_k = \frac{\mathrm{y}_k^T g(x_{k+1})}{y_k^T y_k} \text{ and } \rho_k = \mathrm{s}_k - \mathrm{y}_k$$

Finally,

$$x_{k+1} = x_k + \alpha_k d_k \tag{15}$$

where $\alpha_k > 0$ is attained via line search strategy

$$||g(x_k + \alpha_k d_k)|| \leq \sigma||g(x_k)||, \tag{16}$$

while the new direction is obtained via the following

$$d_{k+1} = \begin{cases} g(x_k) & if\ k = 0 \\ g(x_{k+1}) + \theta_k \rho_k & if\ k \geq 1 \end{cases} \tag{17}$$

with θ_k and ρ_k as defined in (14). Now the algorithm for the proposed method is as follows

Algorithm SCG Method

Step 1: Given $x_0, \alpha > 0, \sigma \in (0,1)$ and $\epsilon > 0$ compute $d_o = -g_o$, set $k = 0$.

Step 2: Compute $g(x_k)$ and test the stopping criterion, i.e. $||g(x_k)|| \leq \epsilon$, If yes, then stop, otherwise continue with step 3

Step 3: Compute α_k by using the line search condition (16) that: if $||g(x_k + \alpha_k d_k)|| \leq \sigma||g(x_k)||$, retain α_k and goto 4. Else set $\alpha_{k+1} = \frac{\alpha_k}{2}$ and repeat 3.

Step 4: Compute $x_{k+1} = x_k + \alpha_k d_k$.

Step 5: Compute search direction using (17)

Step 6: Set $k = k + 1$ and go to step 2

3. CONVERGENCE RESULT OF THE PROPOSED SCG ALGORITHM

In this section, the global convergence result of the proposed algorithm is established using the line search $||g(x_k + \alpha_k d_k)|| \leq \sigma||g(x_k)||$ above.

Definition
Let Ω be the level set defined by

$$\Omega = |x| \parallel g(x) \parallel \leq \sqrt{\parallel g(x) \parallel^2 + \gamma}$$

where γ is a positive constant.

Assumption 3.1
In order to get global convergence of the proposed SCG algorithm, the following assumptions are needed.

(i) The level set $\Omega = |x|\|g(x)\| \leq \sqrt{\parallel g(x) \parallel^2 + \gamma}$ is bounded;

(ii) In some neighborhood of Ω, $g(x)$ is Lipschits continuous, i.e. there exists a constant $L > 0$ such that for any $x, y \in \Omega$,

$$\parallel g(x) - g(y) \parallel \leq L(\parallel x - y \parallel) \quad (18)$$

(iii) There exists a positive constant m_1, such that

$$\parallel g(x_k) \parallel \leq m_1, \forall x \in \Omega. \quad (19)$$

This shows that the sequence $\parallel g(x_k) \parallel$ is bounded.

Lemma 3.1
Supposed that assumption 3.1 holds and $\{x_k\}$ is generated by the SCG algorithm, then

$$\text{(i) } \lim_{k\to\infty} ||\alpha_k d_k|| = 0 \text{ and (ii) } \lim_{k\to\infty} ||\rho_k|| = 0 \quad (20)$$

where $\rho_k = s_k - y_k$ as defined in (15)

Proof
(i) By using Lipschitz continuity (18):

$$\parallel g(x_{k+1}) - g(x_k) \parallel \leq L(\parallel x_{k+1} - x_k \parallel = L \parallel \alpha_k d_k \parallel \quad (21)$$

Using triangular inequality,

$$\parallel g(x_{k+1}) - g(x_k) \parallel \leq L(\parallel g(x_{k+1}) \parallel + \parallel g(x_k) \parallel) \leq L \parallel \alpha_k d_k \parallel \quad (22)$$

From (15) and the line search strategy (16), we have

$$\parallel g(x_{k+1}) \parallel \leq \sigma \parallel g(x_k) \parallel \quad (23)$$

Using (23) in (22) leads to

$$\sigma \parallel g(x_k) \parallel + \parallel g(x_k) \parallel \leq L \parallel \alpha_k d_k \parallel \quad (24)$$

Thus, we have,

$$\frac{(\sigma+1)\|g(x_k)\|}{L} \geq \parallel \alpha_k d_k \parallel \quad (25)$$

Hence $\parallel \alpha_k d_k \parallel \leq \theta$

Where $\theta = \frac{(\sigma+1)\|g(x_k)\|}{L}$ (26)

This indicates that $\parallel \alpha_k d_k \parallel \leq \theta$ is monotone and bounded, hence converges. The proof is complete.

(ii) Recall from (14) that, $\rho_k = s_k - y_k$

$\parallel \rho_k \parallel \leq \parallel s_k \parallel + \parallel -y_k \parallel$ from triangular inequality

$\Rightarrow \parallel \rho_k \parallel \leq \parallel y_k \parallel + \parallel s_k \parallel$

But $y_k = g(x_{k+1}) - g(x_k)$ and $s_k = \alpha_k d_k$

$\Rightarrow \parallel \rho_k \parallel \leq \parallel g(x_{k+1}) - g(x_k) \parallel + \parallel \alpha_k d_k \parallel$

$\leq L \parallel x_{k+1} - x_k \parallel + \parallel \alpha_k d_k \parallel$

$= L \parallel \alpha_k d_k \parallel + \parallel \alpha_k d_k \parallel = (L + 1) \parallel \alpha_k d_k \parallel$

Since $\parallel \alpha_k d_k \parallel \leq \theta$

Therefore, $\parallel \rho_k \parallel \leq (L + 1)\theta$

So we have $\parallel \rho_k \parallel \leq \omega$ where $\omega = (L + 1)\theta$, thus monotone and bounded, hence converges. Thus, $\lim_{k\to\infty} \parallel \rho_k \parallel = 0$.

Lemma 3.2
Supposed that assumption 3.1 holds and $\{x_k\}$ is generated by the SCG algorithm. If there exists a constant $\epsilon > 0$ such that for all k

$$||g(x_{k+1})|| \geq \epsilon \tag{27}$$

then there exists a constant $m_2 > 0$ such that for all k

$$||d_{k+1}|| \leq m_2 \tag{28}$$

Proof

Recall from (14), $\theta_k = \frac{g(x_{k+1})^T y_k}{y_k^T y_k}$ and $y_k = g(x_{k+1}) - g(x_k)$

$$\therefore |\theta_k| = \frac{|g(x_{k+1})^T y_k|}{||y_k||^2}$$

$$= \frac{g(x_{k+1})^T |(g(x_{k+1}) - g(x_k))|}{||y_k||^2}$$

using (18) from assumption 3.1, yields

$$< \frac{L\|g(x_{k+1})\|\|x_{k+1} - x_k\|}{||y_k||^2}$$

$$\frac{L\|g(x_{k+1})\|\|\alpha_k d_k\|}{||y_k||^2} \to 0 \; as \; k \to 0$$

Without lost of generality, let $\epsilon > 0$ be $\epsilon_o = 0.5$, there exist an integer $k_o > 0$, such that for all $k > k_o, |\theta_k| < \epsilon > 0$ holds. Recall that

$$d_{k+1} = -g(x_{k+1}) + \theta_k \rho_k$$

$$\| d_{k+1} \| \leq \| g(x_{k+1}) \| + |\theta_k| \| \rho_k \|$$

$$\cdots$$

$m_2 = max(\| d_1 \|, \| d_2 \|, \cdots, \| d_k \|, m_1 + \epsilon\omega)$ we can deduce that $\| d_{k+1} \| \leq m_2$
i.e. $\| d_{k+1} \|$ is uniformly bounded. Now we are going to state the following global convergence theorem. Under some suitable assumptions, there exist an accumulation point of x_k, which is a solution of equation (1).

Theorem 3.1
Supposed that the assumption 3.1 holds and $\{x_k\}$ is generated by SCG algorithm. Assume further that for all $k > 0$

$$\alpha_k \geq \frac{c|g(x_{k+1})^T d_{k+1}|}{\|\rho_k\|\|d_k\|} \tag{29}$$

where *c* is some positive constant. Then

$$\lim_{k\to\infty} ||g(x_{k+1})|| = 0, \tag{30}$$

Proof
We prove by contradiction, supposed that the conclusion does not hold, then there exists a constant $\epsilon > 0$ such that for all k,

$$g(x_{k+1}) \geq \epsilon$$

holds. Moreover, from Lemma 3.2, we have the boundedness of $d(x_{k+1})$ i.e. $||d(x_{k+1})|| \leq m_2$ and

$$\lim_{k\to\infty} |g(x_{k+1})^T d_{k+1}| = 0 \tag{31}$$

On the other hand, from (17)

$$d_{k+1} = g(x_{k+1}) + \frac{g(x_{k+1})^T y_k (y_k - s_k)}{\|y_k\|^2}$$

$$d_{k+1} = g(x_{k+1}) + \frac{g(x_{k+1})^T y_k \rho_k}{\|y_k\|^2} \tag{32}$$

multiplying (32) by $g(x_{k+1})^T$

$$g(x_{k+1})^T d_{k+1} = g(x_{k+1})^T g(x_{k+1}) + \frac{g(x_{k+1})^T y_k g(x_{k+1})^T \rho_k}{\|g(x_{k+1}) - g(x_k)\|}$$

by Lipschitz condition

$$\|g(x_{k+1})\|^2 \leq |g(x_{k+1})^T d_{k+1}|) + \frac{\|g(x_{k+1})\|^2 L \|\alpha_k d_k)\| \|\rho_k\|}{\|g(x_{k+1}) - g(x_k)\|} \tag{33}$$

by the equation (19) i.e. $\| g(x_k) \| \leq m_1$ for all $x \in \Omega$, Lemma 3.1 and Lemma 3.2 yields (33). Taking the limit of inequality (33) we have $\lim_{k\to\infty} \| g(x_{k+1}) \| = 0$, which contradict equality

$$\| g(x_{k+1}) \| \geq \epsilon \text{ as } k \to \infty.$$

This completes the proof.

4. NUMERICAL RESULTS

In this section, the performance of the SCG method denoted as $M1$ is compared with the methods for nonlinear equations [2] denoted as $M2$. The numerical results were respectively reported by solving several benchmark problems with seven (7) different dimensions ranging from 10 to 50000.

Problem 1: (Spares 1 Function of Byeong [2])

$$F_i(x) = x_i^2 - 1;\ i = 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

Problem 2: (System of Nonlinear Equations)[8]

$$f_i(x) = e^{x_i^2 - 1} - cos(1 - x_i^2);\ i = 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

Problem 3: (System of n Nonlinear Equations)[12]

$$f_i(x) = (1 - x_i^2) + x_i(1 + x_i x_{n-2} x_{n-1} x_n) - 2; i = 1,2,3,\cdots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

Problem 4 (System of *n* Nonlinear Equations)[8]

$$f_i(x) = x_i - 0.1x_{i+1}^2; i = 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

Problem 5: (System of *n* Nonlinear Equations)[2]

$$f_i(x) = \ln(x_i)\cos(1 - (1 + (x^T x)^2)^{-1})\, e^{1-\left(1+(x^T x)^2\right)^{-1}}; i = 1,2,3,\cdots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

Problem 6: (System of *n* Nonlinear Equations)[20]

$$F_i(x) = e^{x_i} - 1;\ i = 1,2,3,\ldots,n.$$

$x_0 = (0.5,0.5,0.5,\ldots,0.5)^T$ and $x_0 = (0.2,0.2,0.2,\ldots,0.2)^T$

The code for the proposed method was done using MATLAB 7.1 [15]. Using the programming environment R2009b and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on windows XP operator. The search is stopped if:

$$\big||s_k|\big| + \big||g(x_k)|\big| \leq 10^{-4} \tag{34}$$

We further design the codes to terminate whenever the number of iterations is at most 1000 but no point of x_k that satisfy (34) is obtained. The performance of these methods is compared in terms of number of iterations and CPU times measured in seconds. We present all the results using the performance profile proposed by Dolan and More [3]. This profile presents a descriptive measure providing a wealth of information such as solver efficiency and probability of success in the general form. The performance profile

$P : R \rightarrow [0,1]$ is defined as follows: Let P and S be the set of problems and set of solvers respectively. For n_s solvers and n_p problems, and for each problem $p \in P$ and for each solver $s \in S$, we define $t_{p,s}$:=(number of iterations required to solve problem p by solver s. The performance ratio is given by

$$r_{p,s} := t_{p,s}/min\{t_{p,s}\}.$$

Then the performance profile is defined by

$$P(\tau) := \frac{1}{n_p} size\big(p \in P : r_{p,s} \leq \tau\big),$$

for all $\tau \in R$ where $P(\tau)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor $\tau \in R$ of the best possible ratio. The numerical results are presented in table 1-6. The meaning of each column in the tables are stated as follows. "*Prob*" stands for problem solved, "*Dim*" stands for dimension of the test problems, "Iter": the total number of iterations. We say that in the particular problem, the $M1$ performs better than the others if the number of iteration (*iter*) or the CPU Time of $M1$ is less than the number of iteration or the CPU Time corresponding to the other method respectively. By the given results, the proposed method performs effectively with the given benchmark problems as in table 1-6.

Figure 1-2 presents the graphical results of problems 1-6 relative to number of iterations and CPU Time respectively. That is, for each method, we plot the fraction $P(\tau)$ of problems for which the method is within a factor τ of the best time [5,12]. The top curve is the method that performs better in a time that was within a factor τ of the best time. From figure 1, the proposed $M1$ methods relative to the number of iterations, outperforms other method $M2$.

Similarly, Figure 2 shows the performance of $M1$ and other methods relative to CPU Time where the $M1$ is the top performer, showing that the proposed method achieved better results, thus we conclude that $M1$ method is promising.

Table 1. Numerical results of problem 1 based on dimension of problem (n), Number of iterations and CPU time

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
1	10	2	1.32263	3	0.05509
	50	2	0.17996	4	0.29727
	100	2	0.32584	4	0.26083
	1000	3	0.22476	4	0.46225
	5000	4	0.22757	4	1.23982
	10000	4	1.28413	4	12.77257
	50000	5	5.46303	4	63.81277
	10	5	0.00739	3	0.08002
	50	5	0.32214	3	0.47035
	100	5	0.03015	4	0.04157
	1000	5	0.16982	5	0.08773
	5000	5	0.51964	6	1.67228
	10000	5	2.33713	6	8.43514
	50000	7	14.64674	7	7.33695

Table 2. Numerical results of problem 2 based on dimension of problem (n), Number of iterations and CPU time

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
2	10	3	0.01289	5	0.00915
	50	4	0.00969	6	0.00915
	100	4	0.01434	6	0.01941
	1000	5	0.03282	6	0.05069
	5000	6	0.25895	5	0.31622
	10000	7	0.88493	7	1.79673
	50000	8	4.04199	7	9.43085
	10	5	0.03552	4	1.85555
	50	5	0.34212	5	0.05237
	100	5	0.14323	6	0.05184
	1000	5	0.12377	6	0.22882
	5000	5	0.15634	6	0.62009
	10000	5	0.44366	6	3.26438
	50000	5	1.09876	4	78.44583

Table 3. Numerical results of problem 3 based on dimension of problem (n), Number of iterations and CPU time

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
3	10	6	0.106	-	-
	50	6	0.10898	100	1.19026
	100	6	0.12543	30	0.2983
	1000	6	0.12322	26	0.39239
	5000	6	323432	23	0.77831
	10000	6	0.43532	21	5.06292
	50000	9	28.24723	-	-
	10	-	-	21	0.02378
	50	6	0.06803	27	0.98422
	100	7	0.38938	44	0.43931
	1000	7	0.95442	28	0.38536
	5000	8	4.50767	33	1.02821
	10000	4	16.70493	26	5.38295
	50000	56	85.32403	20	26.683997

Table 4. *Numerical results of problem 4 based on dimension of problem (n), Number of iterations and CPU time*

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
4	10	5	0.20765	11	0.01779
	50	6	0.24014	11	0.05944
	100	6	0.11044	12	0.31941
	1000	6	0.1719	12	0.93839
	5000	7	0.55043	12	3.72989

	10000	10	3.50237	16	5.19109
	50000	9	25.48175	19	30.68633
	10	5	0.01148	11	0.01029
	50	8	0.01488	12	0.02641
	100	95	0.04344	12	0.03914
	1000	5	0.2454	13	0.09055
	5000	9	0.67137	14	0.59221
	10000	9	8.02988	8	32.11532
	50000	9	7.32454	9	196.84769

Table 5. Numerical results of problem 5 based on dimension of problem (n), Number of iterations and CPU time

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
5	10	10	0.19289	4	0.02573
	50	10	0.13698	10	0.05326
	100	17	0.03577	14	0.05442
	1000	20	0.25247	26	0.25592
	5000	20	0.77096	21	0.89268
	10000	20	4.34788	23	6.07403
	50000	20	27.16918	32	44.26985
	10	9	0.01636	5	0.13752
	50	12	0.19998	1000	18.42693
	100	12	0.04573	-	-
	1000	10	7.11197	-	-
	5000	27	0.93975	-	-
	10000	26	5.53954	10	5.21812
	50000	-	-	-	-

Table 6. Numerical results of problem 6 based on dimension of problem (n), Number of iterations and CPU time

		SCG ($M1$)		Broyden ($M2$)	
Prob	Dim	Iter	CPU time	Iter	CPU Time
6	10	2	0.44433	23	0.08413
	50	2	0.07508	17	0.40897
	100	2	0.21035	23	0.32005
	1000	2	0.81157	38	0.55585
	5000	2	2.12648	21	0.81418
	10000	3	16.16561	50	10.27528
	50000	3	63.36318	50	78.60937
	10	3	0.17667	3	0.17823
	50	3	0.06205	10	0.03633
	100	4	0.1121	10	0.07558
	1000	4	0.81112	10	0.25121
	5000	4	0.67543	10	1.01534
	10000	4	0.57876	12	1.08977
	50000	4	23.8762	10	31.01261

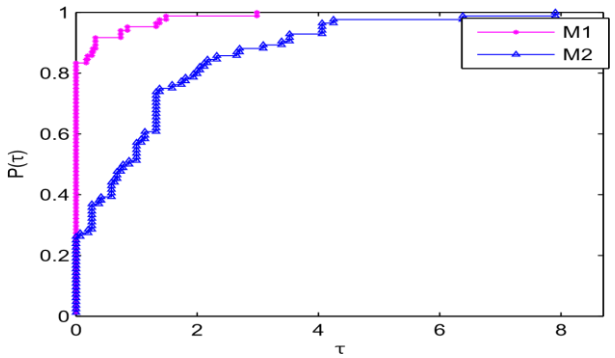

Figure 1. Performance profile of M1 and M2 methods with respect to the number of iterations for the problems 1-6

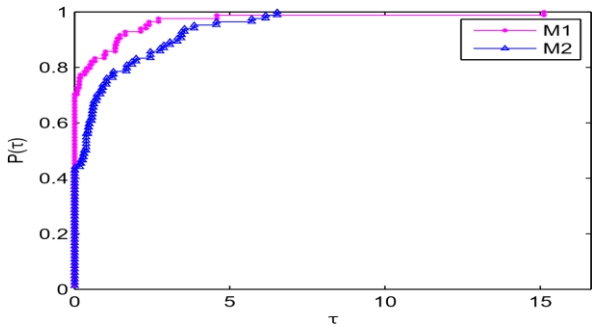

Figure 2. Performance profile of M1 and M2 methods with respect to the CPU Time for the problems 1-6

5. CONCLUSION

This paper presents a simple new conjugate gradient method, based on a modification of Broyden's algorithm via memoryless approach, for which both the descent condition and the conjugacy condition are satisfied. Numerical experiments on some test problems of different dimension shows that the proposed method is the top performer in all the cases compare to other conjugate gradient algorithm. It is also worth mentioning that the method is capable of significantly reducing the CPU Time and the number of iterations, as compared to CG-Like algorithms [2], while maintaining good accuracy of the numerical solution to some extends. Hence, we can claim that our proposed method (SCG) is a good alternative algorithm for solving large scale systems of nonlinear equations.

References

[1] C.G. Broyden, A class of methods for solving nonlinear simultaneous equations, Math. Comput., 19 (1965), 577-593.

[2] D. H. Li and M. Fukushima (2000). A Derivative-free line search and global convergence of Broyden- like methods for nonlinear equations, Optimization Methods and Software 13, 181-201.

[3] Elizabeth D. Dolan, Jorge J. Mor´e (2002). Benchmarking Optimization Software with Performance Profiles. Mathematical Programming 91, 201-213.

[4] Gongli Yuan, Xiwen Lu, (2008). A new backtracking inexact BFGS method for symmetric nonlinear equations", Journal of computers and mathematics with applications, 55, 116-129.

[5] Jamilu Sabi'u, Abdullah Shah, Mohammed Yusuf Waziri & Muhammad Kabir Dauda (2020). A New Hybrid Approach for Solving Large-scale Monotone Nonlinear Equations. J. Math. Fund. Sci., Vol. 52, No. 1, 17-26. DOI: 10.5614/j.math.fund.sci.2020.52.1.2

[6] Jinkui Liu and Shengjie Li, Spectral DY Type Projection Method for Nonlinear Monotone Systems of Equations, Journal of Computation of Mathematics, 4(2015) 341-354.

[7] Li, Q. & Li, D.H., (2011). A Class of Derivative-free Methods for Large-Scale Nonlinear Monotone Equations, IMA Journal of Numerical Analysis, 31(4), pp. 1625-1635.

[8] M. K. Dauda, Abubakar S Magaji, Habib Abdullah, Jamilu Sabi'u and Abubakar S Halilu (2019). A New Search Direction via Hybrid Conjugate Gradient Coefficient for Solving Nonlinear System of Equations. Malaysian Journal of Computing and Applied Mathematics, Vol 2(1): 8-15.

[9] M. K. Dauda, Mustafa Mamat, M. Y. Waziri, Fadhila Ahmad and Fatma Susilawati Mohamad,Inexact CG-Method via SR1 Update for Solving Systems of Nonlinear Equations, Far East Journal of Mathematical Sciences(FJMS) Volume 100, Issue 11,(2016) Pages 1787-1804.

[10] M. K. Dauda, Mustafa Mamat, Mohamad A. Mohamed, Nor Shamsidah Amir Hamzah. (2019). Hybrid conjugate gradient parameter for solving symmetric systems of nonlinear equations. Indonesian Journal of Electrical Engineering and Computer Science Vol. 16, No. 1, October, pp. 539~543 ISSN: 2502-4752, DOI: 10.11591/ijeecs.v16.i1.pp539-543.

[11] M. K. Dauda, Mustafa Mamat, Mohamad Afendee Mohamed, Fatma Susilawati Mohamad and M.Y. Waziri, (2017). Derived Conjugate Gradient Parameter for Solving Symmetric Systems of Nonlinear Equations, Far East Journal of Mathematical Sciences (FJMS), 102 (11)2599-2610.

[12] M. Mamat, M. K. Dauda, M. A. bin Mohamed, M. Y. Waziri and F. S. Mohamad, H. Abdullah, (2018). Derivative free Davidon-Fletcher-Powell (DFP) for solving symmetric systems of nonlinear equations, IOP Conf. Series: Materials Science and Engineering 332, doi:10.1088/1757-899X/332/1/012030.

[13] M.K. Dauda, Mustafa Mamat, M.Y. Waziri, Fadhilah Ahmad and Fatma Susilawati Mohamad, (2016), Improved Quasi-Newton Method Via PSB Update for Solving Systems of Nonlinear equations, AIP Conference Proceedings (ICOQSIA 2016), 1782, 030009 (2016); doi: 10.1063/1.4966066.

[14] Mahammad Kabir Dauda, Mustafa Mamat, Mohamad Afendee bin Mohamed and Mahammad Yusuf Waziri (2019). Improved Quasi-Newton method via SR1 update for solving symmetric systems of nonlinear equations. Malaysian Journal of Fundamental and Applied Sciences Vol. 15, No.1, 117-120.

[15] Mathews J. H, Fink K. D. (1999). Numerical method using MATLAB. Prentice Hull, Upper saddle river, NJ 07458.

[16] Mustafa Mamat, Fatma Susilawati Mohamad, Abubakar S. Magaji and M.Y. Waziri (2019). Derivative Free Conjugate Gradient Method via Broyden's Update for solving symmetric systems of nonlinear equations. Journal of Physics: Conference Series, 1366. 012099 IOP Publishing doi:10.1088/1742-6596/1366/1/012099.

[17] Neculai Andrei, 40 Conjugate Gradient Algorithms for Unconstrained Optimisation, A Survey on their definition. Academy of Romanian Scientists, Romania.

[18] W. Zhou and D. Shen, (2015). Convergence properties of an iterative method for solving symmetric non-linear equations. Journal of Optimization Theory and Applications", vol. 164, no. 1, pp. 277- 289.

[19] Weijun Zhou & Dongmei Shen (2014). An Inexact PRP Conjugate Gradient Method for Symmetric Nonlinear Equations, Numerical Functional Analysis and Optimization, 35:3, 370-388, DOI: 10.1080/01630563.2013.871290.

[20] Yasushi Narushima and Hiroshi Yabe, Conjugate gradient methods based on secant conditions that generate descent search directions for unconstrained Optimization, Journal of Computational and Applied Mathematics, 236 (2012), 4303-4317.