Nonlinear Convex Analysis and Optimization
Vol. 1, No. 2, 2022, pp. 141–159
https://ncao.design.blog

# Modified Conjugate Gradient Method for Solving System of Nonlinear Equations

Mohammed Yusuf Waziri[a,1], Aliyu Yusuf[b,2], Kabiru Ahmed[a,3], Abubakar Sani Halilu[c,4*]

[a]Department of Mathematical Sciences, Bayero University, Kano, Nigeria
[b]Department of Science, School of Continuing Education, Bayero University, Kano, Nigeria
[c]Department of Mathematics, Sule Lamido University, Kafin Hausa, Nigeria
[1]mywaziri.mth@buk.edu.ng; [2]ayusuf.sce@buk.edu.ng; [3]kabiruhungu16@gmail.com;
[4]abubakars.halilu@slu.edu.ng
*Corresponding Author

**ABSTRACT** In this paper, we proposed a modified hybrid conjugate gradient method based on a convex combination of the Fletcher-Reeves (FR), Polak-Ribiere-Polyak (PRP) and a quasi-Newton's update. However, one of the suggested algorithm's key features is that the search direction is generated using a derivative-free line search. Under suitable assumptions, the algorithm is set up in such a way that its convergence is globally obtained. Finally, numerical outcomes on numerous benchmark test problems, show that our approach is more effective and robust than some existing methods.

**Article History**
Received 26 Feb 2022
Accepted 31 May 2022
**Keywords:**
system of nonlinear equations; conjugate gradient parameters; convex combination; global convergence; numerical experiment
**MSC:**
47H05; 47J25

## 1. Introduction

Consider a system of nonlinear equations of the form:

$$F(x) = 0, \tag{1.1}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is nonlinear map. Nonlinear system of equations have a wide range of applications in science and engineering, and a variety of methods for dealing with them have been developed. See Newton's and quasi-Newton's methods [1, 2, 3, 4], the diagonal Jacobian approximation method [5] and the derivative-free method [6], for more details. But, storage, computation, and iterative approximation of the Jacobian matrix are required, these renders both the two approaches unsuitable to solve large-scale system of nonlinear equations [7]. The conjugate gradient (CG) methods are introduced to address the disadvantages of Newton's and quasi-Newton methods. Consequently, they are effective for dealing with large-scale problems because of their convergence properties and low storage requirement, see [8, 9] for more information. It uses the recurrence relation to generate an iterative sequence $x_k$ for



**Please cite this article as:** M. Y. Waziri et al., Modified Conjugate Gradient Method for Solving System of Nonlinear Equations, Nonlinear Convex Anal. & Optim., Vol. 1 No. 2, 141–159.

a given initial guess $x_0 \in \mathbb{R}^n$ via:

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \cdots \tag{1.2}$$

where $\alpha_k > 0$, $x_{k+1}$ is the current iterate, $x_k$ is the previous iterate, and CG search direction $d_k$ is given by

$$d_k = \begin{cases} -F(x_k), & \text{if } k = 0, \\ -F(x_k) + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{1.3}$$

for which the CG update parameter is $\beta_k$. The approach changes in how $\beta_k$ is defined, and hence different choices of $\beta_k$ leads to different CG approaches with various levels of computational effectiveness and convergence.

In their survey of nonlinear CG methods, Hager and Zhang [10], divided the most important CG techniques in two major categories, for which the Fletcher-Reeves (FR) method which is in the first category, was established in 1964 (further information is available in [11]), and is given by the following update parameter:

$$\beta_k^{FR} = \frac{\|F(x_{k+1})\|^2}{\|F(x_k)\|^2} . \tag{1.4}$$

The update parameter for the Polak-Ribiere and Polyak (PRP) method, which is among the second category founded in 1969 [12], is as follows:

$$\beta_k^{PRP} = \frac{F^T(x_{k+1})y_k}{\|F(x_k)\|^2} . \tag{1.5}$$

The first category of CG methods are excellent in terms of global convergence [8], whereas the strong computational performance is exhibited by the second category of the methods [13]. So, to exploit the advantages from each category, we merge the two approaches to create hybrid CG algorithm, which are more effective and trustworthy than any other classical CG algorithm. Most importantly, the algorithm works well especially when addressing large-scale, unconstrained optimization problems of the following nature:

$$\min f(x), \ x \in \mathbb{R}^n, \tag{1.6}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable function [14]. Hybrid CG methods have been widely used to solve (1.6) as can be seen in the articles presented by Andrei that provide a variety of hybrid CG methods for convex combination, together with the information on their traits, global convergence and computational experiments, (see [15, 16, 17]) for details. However, the one described by Babaie-Kafaki et al. [18] is among the most effective of the methods.

Recently, in [14], Ioannis et al. developed a convex combination of descent hybrid CG approach using the memory-free BFGS update to solve (1.6). The hybridization parameter $\phi_k$ is determined by combining $\beta_k^{DY}$ and $\beta_k^{HS}$ updates together with the approach of Frobenius matrix norm. Because of its convergence property, ease of implementation, and cheap storage requirements, hybrid CG methods are commonly utilized, but the scheme is scarce in the literature for solving nonlinear system of equations. In this paper, we proposed a hybrid algorithm based on a convex combination of $\beta_k^{FR}$ and $\beta_k^{PRP}$ parameters, similar to the one presented in [14] by Ioannis et al. by combining the self-scaling memory-less BFGS direction with the direction of hybrid $\beta_k^{FR}$ and $\beta_k^{PRP}$ parameters given by

$$\beta_k^{MCG} = \phi_k \beta_k^{FR} + (1-\phi_k)\beta_k^{PRP}, \quad where, \quad \phi_k \in [0,1], \quad \forall k. \tag{1.7}$$

However, we limit the values of $\phi_k$ in the expression (1.7) to the range [0,1], to obtain a convex combination. That is, $\phi_k$ is set to be zero, if it is less than zero, $\phi_k$ is equal to 1, if it is greater than 1, a suitable convex combination exists when $\phi_k$ is between 0 and 1. Throughout this paper, we denoted the Euclidean norm of vectors as $\|.\|$, $F_k = F(x_k)$, $f_k = f(x_k)$ $y_k = F_{k+1} - F_k$ and $s_k = x_{k+1} - x_k$. In addition, we assume that the Lipstchitz condition is satisfied in (1.1), and $f$ from (1.6) is defined by

$$f(x) = \frac{1}{2}\|F(x)\|^2. \tag{1.8}$$

### 1.1. Self-Scaling Memory-less BFGS

For large-scale optimization problems, the self-scaling memory-less BFGS method is typically regarded as one of the most effective approaches [17], because of its good computational performance and robust theoretical attributes. Based on the L-BFGS concept [19], the self-scaling memory-less BFGS matrices are generated, given an initial matrix $B_0 = \varrho_0 I$ with $\varrho_0 \in \mathbb{R}$ then, the formula for BFGS is:

$$B_{k+1} = B_k - \frac{B_k s_k s_k^T B_k}{s_k^T B_k s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{1.9}$$

where $B_k$ is an $n \times n$ matrix known as the Jacobian matrix's approximation at $x_k$. From (1.9), the scaled memory-less BFGS is updated as follows:

$$B_{k+1} = \varrho_k I - \varrho_k \frac{s_k s_k^T}{s_k^T s_k} + \frac{y_k y_k^T}{s_k^T y_k}, \tag{1.10}$$

where $\varrho_k \in \mathbb{R}$ is the scaling parameter. The search direction is determined by

$$d_{k+1} = -B_{k+1}^{-1} F_{k+1}, \tag{1.11}$$

where $-B_{k+1}^{-1}$ is a Jacobian inverse approximation that can be obtained using the following expression [14]

$$B_{k+1}^{-1} = \frac{1}{\varrho_k} I - \frac{1}{\varrho_k} \frac{s_k y_k^T + y_k s_k^T}{s_k^T y_k} + \left(1 + \frac{1}{\varrho_k} \frac{\|y_k\|^2}{s_k^T y_k}\right) \frac{s_k s_k^T}{s_k^T y_k}, \tag{1.12}$$

where $\varrho_k$ is to be expressed by Oren and Luenberger method [20]

$$\varrho_k^{OL} = \frac{s_k^T y_k}{\|s_k\|^2}. \tag{1.13}$$

### 1.2. Proposed Method and its Algorithm

A hybrid technique as a convex combination of the classical FR [11] and PRP [12] CG schemes is provided here. Following a similar process as in [14], the update parameter is obtained using self-scaling memory-less BFGS direction and the direction of hybrid two CG parameters. By (1.3) and (1.7), our propose descent search direction is given by:

$$d_{k+1}^{MCG} = -Q_{k+1}F_{k+1}, \tag{1.14}$$

where $Q_{k+1} = I - \phi_k \frac{d_k F_{k+1}^T}{\|F_k\|^2} - (1-\phi_k)\frac{d_k y_k^T}{\|F_k\|^2}$ is a search direction matrix. Therefore, the following minimization problem can be solved by computing the parameter $\phi_k$.

$$min\|D_{k+1}\|_F \quad \phi_k > 0, \tag{1.15}$$

where $\|.\|_F$ is the Frobenius matrix norm and $D_{k+1} = Q_{k+1}^T - B_{k+1}^{-1}$. If $\|D_{k+1}\|_F^2 = tr(D_{k+1}^T.D_{k+1})$, then we have

$$\begin{aligned}\|D_{k+1}\|_F^2 = \phi_k^2\left(\frac{(F_{k+1}^T s_k)^2 + (s_k^T y_k)^2}{\|F_k\|^4}\right) - 2\phi_k\Bigg[&\left(s_k^T y_k - F_{k+1}^T s_k\right)\frac{s_k^T y_k}{\|F_k\|^4} + \left(1 - \frac{1}{\varrho_k}\right)\frac{F_{k+1}^T s_k}{\|F_k\|^2}\\ &+ \left(\frac{1}{\varrho_k} - 1\right)\frac{s_k^T y_k}{\|F_k\|^2} + \left(1 - \frac{F_{k+1}^T s_k}{s_k^T y_k}\right)\frac{\|s_k\|^2}{\|F_k\|^2} + \left(F_{k+1}^T s_k - s_k^T y_k\right)\frac{2}{\varrho_k\|F_k\|^2}\\ &+ \left(1 - \frac{F_{k+1}^T s_k}{s_k^T y_k}\right)\frac{\|s_k\|^2\|y_k\|^2}{\varrho_k s_k^T y_k}\Bigg] + \Im,\end{aligned} \tag{1.16}$$

where $\Im$ is a constant containing the negative term of $(-\phi_k^2)$ and does not depend on $\phi_k$.

However, a second-degree polynomial with the variable $\phi_k$, where the coefficient of $\phi_k^2$ is always positive, is used to compute the value of $\|D_{k+1}\|_F^2$. Now, from (1.16), we get:

$$\begin{aligned}2\phi_k\Bigg[\phi_k\left(\frac{(F_{k+1}^T s_k)^2 + (s_k^T y_k)^2}{2\|F_k\|^4}\right) - \Bigg[&\frac{(s_k^T y_k - F_{k+1}^T s_k)s_k^T y_k}{\|F_k\|^4}\\ &+ \frac{1}{\|F_k\|^2}\left(1 - \frac{1}{\varrho_k}\right)\left(F_{k+1}^T s_k - s_k^T y_k\right) + \frac{\|s_k\|^2}{\|F_k\|^2}\left(1 - \frac{F_{k+1}^T s_k}{s_k^T y_k}\right)\\ &+ \frac{2}{\varrho_k\|F_k\|^2}\left(F_{k+1}^T s_k - s_k^T y_k\right) + \frac{\|s_k\|^2\|y_k\|^2}{\varrho_k s_k^T y_k}\left(1 - \frac{F_{k+1}^T s_k}{s_k^T y_k}\right)\Bigg]\Bigg] = 0.\end{aligned} \tag{1.17}$$

Finally, from (1.17), the unique solution of problem (1.15) is obtained as:

$$\begin{aligned}\phi_k^* = \left(\frac{2\|F_k\|^2}{(F_{k+1}^T s_k)^2 + (s_k^T y_k)^2}\right)\Bigg[&\left(F_{k+1}^T s_k - s_k^T y_k\right)\left(\frac{2}{\varrho_k} - \frac{s_k^T y_k}{\|F_k\|^2}\right)\\ &+ \left(1 + \frac{\|F_k\|^2\|y_k\|^2}{\varrho_k s_k^T y_k}\right)\left(1 - \frac{F_{k+1}^T s_k}{s^T y_k}\right)\|s_k\|^2 + \left(1 - \frac{1}{\varrho_k}\right)\left(F_{k+1}^T s_k - s_k^T y_k\right)\Bigg].\end{aligned} \tag{1.18}$$

The concept of the approach in [20] was adopted to ensure that, the suggested method generates a descent search direction. Let us specify the direction of our algorithm by:

$$d_{k+1}^{MCG} = -\left(1 + \beta_k^{MCG} F_{k+1}^T d_k\right)F_{k+1} + \|F_{k+1}\|^2\beta_k^{MCG} d_k. \tag{1.19}$$

Therefore, in view of (1.19), the condition holds as follows:

$$F_{k+1}^T d_{k+1}^{MCG} \leq -\|F_{k+1}\|^2. \tag{1.20}$$

We computed the step-length $(\alpha_k)$ using the derivative-free line search method suggested in [3]. Suppose that, $\psi_1 > 0$, $\psi_2 > 0$ and $r \in (0, 1)$ are constants. Let $\{\sigma_k\}$ represent a non-negative sequence in the sense that

$$\sum_{k=0}^{\infty} \sigma_k < \sigma < \infty. \tag{1.21}$$

Hence, $\alpha_k$ is to be computed as:

$$f_{k+1} - f_k \leq -\psi_1\|\alpha_k F_k\|^2 - \psi_2\|\alpha_k d_k\|^2 + \sigma_k f_k. \tag{1.22}$$

Let $i_k$ be the lowest positive integer $i$ in the sense that, $\alpha = r^i$ satisfies in the equation (1.22). Suppose that $\alpha_k = r^{i_k}$, then the following is how we can now explain the MCG algorithm:

**Algorithm 1:** A Modified Hybrid Conjugate Gradient (MCG) Algorithm

**Step** 1: Given $x_0 \in \mathbb{R}^n$, $\alpha_k > 0$, $\epsilon = 10^{-4}$, $d_0 = -F_0$, set $k = 0$.

**Step** 2: Compute $F(x_k)$. If $\|F(x_k)\| \leq \epsilon$, stop, otherwise go to **Step** 3.

**Step** 3: Compute the step length $\alpha_k$ using (1.22).

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

**Step** 5: Compute $F(x_{k+1})$.

**Step** 6: Update $d_{k+1}^{MCG}$ from (1.19), by using (1.4), (1.5), (1.18) and (1.7).

**Step** 7: Set $k = k + 1$ and go to **Step** 2.

The remainder of this work is arranged in the following manner. In Section 2, we proved the convergence of the algorithm. In section 3, we give the numerical experiments using various test problems of nonlinear equations. It also contains a report that discusses the numerical outcomes. Section 4 is the conclusion.

## 2. Theoretical Results

We need to make the following corollaries in order to prove that, our Algorithm 1 is globally convergent to the solution of (1.1).

**Corollary 2.1.** *The following set is bounded:*

$$\Omega = \{x | \|F(x)\| \leq \|F(x_0)\|\},$$

*that is, there exists a constant* $B > 0$ *such that* $\|F(x)\| \leq B, \quad \forall x \in \Omega.$

**Corollary 2.2.** *Since* $F$ *is continuously differentiable, then, there exists* $x^* \in \mathbb{R}^n$, *such that* $F(x^*) = 0$.

**Corollary 2.3.** *If the function* $F$ *is Lipschitz continuous, then there exists a positive constant* $L$ *such that*

$$\forall \quad x,\ y \in \Omega, \qquad \|F(x) - F(y)\| \leq L\|x - y\|.$$

*But based on Corollary 2.1, it is clear that, there exists a non-negative constant $M$ such that.*

$$\|F(x)\| \leq M\ , \quad \forall \quad x \in \Omega. \tag{2.1}$$

**Lemma 2.4.** *Suppose that, $\{x_k\}$ is generated by MCG Algorithm 1, then the direction $d_k$ is a descent for $F_k$ at $x_k$. Meaning that:*

$$F_{k+1}^T d_{k+1} < 0\ , \quad \forall\ k \geq 0. \tag{2.2}$$

*Proof.* By (1.19), we can deduce that:

$$d_{k+1}^{MCG} = -\left(1 + \beta_k^{MCG} F_{k+1}^T d_k\right) F_{k+1} + \|F_{k+1}\|^2 \beta_k^{MCG} d_k. \tag{2.3}$$

Multiplying (2.3) by $F_{k+1}^T$ to obtain,

$$F_{k+1}^T d_{k+1}^{MCG} = -\|F_{k+1}\|^2 - \|F_{k+1}\|^2 \beta_k^{MCG} F_{k+1}^T d_k + \|F_{k+1}\|^2 \beta_k^{MCG} F_{k+1}^T d_k. \tag{2.4}$$

Therefore, from (2.4), we get

$$F_{k+1}^T d_{k+1}^{MCG} = -\|F_{k+1}\|^2,$$

which shows that,

$$F_{k+1}^T d_{k+1}^{MCG} < 0.$$

∎

**Lemma 2.5.** *If Corollaries 2.1 and 2.3 are met, and the sequence $\{x_k\}$ is generated by the Algorithm 1. Suppose that, $m$ is any non-negative constant, $\ni$,*

$$\|F_k\|^2 \geq m. \tag{2.5}$$

*Then,*

$$\left|\beta_k^{MCG}\right| \leq \frac{M}{m}\Big(M + 2LB\Big) := \mu.$$

*That is, our proposed $\beta_k^{MCG}$ is bounded to some positive constants.*

*Proof.* From (1.7), we have

$$\beta_k^{MCG} = \phi_k \beta_k^{FR} + (1 - \phi_k)\beta_k^{PRP}, \quad \textit{where} \quad \phi_k \in [0, 1], \quad \forall k. \tag{2.6}$$

Using (1.4) and (1.5), (2.6) becomes,

$$\beta_k^{MCG} = \phi_k \frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{F_{k+1}^T y_k}{\|F_k\|^2} - \phi_k \frac{F_{k+1}^T y_k}{\|F_k\|^2}. \tag{2.7}$$

The absolute value on each side of (2.7), is taken to obtain:

$$\left|\beta_k^{MCG}\right| \leq |\phi_k| \frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{|F_{k+1}^T y_k|}{\|F_k\|^2} + |\phi_k| \frac{|F_{k+1}^T y_k|}{\|F_k\|^2}. \tag{2.8}$$

Applying Cauchy-Schwartz inequality to (2.8), we have

$$\left|\beta_k^{MCG}\right| \leq |\phi_k| \frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{\|F_{k+1}\|\|y_k\|}{\|F_k\|^2} + |\phi_k| \frac{\|F_{k+1}\|\|y_k\|}{\|F_k\|^2}.$$

From Corollary 2.3 and (2.5), it follows that,

$$|\beta_k^{MCG}| \leq |\phi_k|\frac{M^2}{m} + \frac{LM\|s_k\|}{m} + |\phi_k|\frac{LM\|s_k\|}{m}. \tag{2.9}$$

Rearrange (2.9), to get

$$|\beta_k^{MCG}| \leq \frac{M}{m}\Big(|\phi_k|M + L\|s_k\|\Big(1 + |\phi_k|\Big)\Big).$$

Thus, by Corollary 2.1, we have

$$|\beta_k^{MCG}| \leq \frac{M}{m}\Big(M + 2LB\Big) := \mu. \tag{2.10}$$

∎

**Lemma 2.6.** *Assume that, Corollaries 2.1 and 2.1 are true, and the direction $\{d_k\}$ is generated by the MCG Algorithm 1. Then,*

$$\|d_{k+1}^{MCG}\| \leq \rho^k M^{k+1}, \quad \textit{where} \quad \rho > 0, \quad \forall\ k.$$

*Proof.* We have by Lemma 2.4, $\forall k > 0$:

$$d_{k+1}^{MCG} = -\left(1 + \beta_k^{MCG} F_{k+1}^T d_k^{MCG}\right) F_{k+1} + \|F_{k+1}\|^2 \beta_k^{MCG} d_k^{MCG}. \tag{2.11}$$

Now, equation (2.11) can be expressed as

$$\left\|d_{k+1}^{MCG}\right\| = \left\| -\left(1 + \beta_k^{MCG} F_{k+1}^T d_k^{MCG}\right) F_{k+1} + \|F_{k+1}\|^2 \beta_k^{MCG} d_k^{MCG}\right\|. \tag{2.12}$$

By triangular inequality, (2.12) becomes

$$\left\|d_{k+1}^{MCG}\right\| \leq \left\|\left(1 + \beta_k^{MCG} F_{k+1}^T d_k\right) F_{k+1}\right\| + \left|\beta_k^{MCG}\right|\left\|F_{k+1}\right\|^2\left\|d_k^{MCG}\right\|. \tag{2.13}$$

From which, we expand (2.13) and obtain

$$\left\|d_{k+1}^{MCG}\right\| \leq \left\|F_{k+1} + \beta_k^{MCG}\left\|F_{k+1}\right\|^2 d_k\right\| + \left|\beta_k^{MCG}\right|\left\|F_{k+1}\right\|^2\left\|d_k^{MCG}\right\|. \tag{2.14}$$

By triangular inequality, (2.14) can written as

$$\left\|d_{k+1}^{MCG}\right\| \leq \left\|F_{k+1}\right\| + \left|\beta_k^{MCG}\right|\left\|F_{k+1}\right\|^2\left\|d_k\right\| + \left|\beta_k^{MCG}\right|\left\|F_{k+1}\right\|^2\left\|d_k^{MCG}\right\|. \tag{2.15}$$

From, (2.15), we have

$$\left\|d_{k+1}^{MCG}\right\| = \left\|F_{k+1}\right\| + 2\left|\beta_k^{MCG}\right|\left\|F_{k+1}\right\|^2\left\|d_k^{MCG}\right\|. \tag{2.16}$$

By apply (2.1) and (2.10), (2.16) becomes

$$\|d_{k+1}^{MCG}\| \leq M + 2\mu M^2\|d_k^{MCG}\|. \tag{2.17}$$

However, (2.17) can be expressed as

$$\|d_{k+1}^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\|d_k^{MCG}\|. \tag{2.18}$$

When $k = 1$, (2.18) becomes

$$\|d_2^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\|d_1^{MCG}\|.$$

But,

$$\|d_1^{MCG}\| = \| - F_1^{MCG}\| = \|F_1^{MCG}\|.$$

Therefore,

$$\|d_2^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\|F_1^{MCG}\|. \tag{2.19}$$

When $k = 2$, then, from (2.18), we have

$$\|d_3^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\|d_2^{MCG}\|.$$

In which from (2.19), we have

$$\|d_3^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\Big(M\Big(1 + 2\mu M\Big)\|F_1^{MCG}\|\Big). \tag{2.20}$$

From (2.20), we get

$$\|d_3^{MCG}\| \leq M^2\Big(1 + 2\mu M\Big)^2\|F_1^{MCG}\|. \tag{2.21}$$

When $k = 3$, then, from (2.18), we have

$$\|d_4^{MCG}\| \leq M\Big(1 + 2\mu M\Big)\|d_3^{MCG}\|. \tag{2.22}$$

Using (2.21), (2.22) becomes

$$\|d_4^{MCG}\| \leq M\Big(1 + 2\mu M\Big)M^2\Big(1 + 2\mu M\Big)^2\|F_1^{MCG}\|. \tag{2.23}$$

From (2.23), we obtain

$$\|d_4^{MCG}\| \leq M^3\Big(1 + 2\mu M\Big)^3\|F_1^{MCG}\|. \tag{2.24}$$

Using (2.1), we can write (2.24) as

$$\|d_4^{MCG}\| \leq \Big(1 + 2\mu M\Big)^3 M^4.$$

In general, $\forall k \geq 0$, we have

$$\|d_{k+1}^{MCG}\| \leq \rho^k M^{k+1},$$

where

$$\rho = 1 + 2\mu M.$$

∎

**Lemma 2.7.** *[1] If Corollaries 2.3 is holds, let the sequence $\{x_k\}$ be generated by the Algorithm 1. Then, we have*

$$\lim_{k\to\infty} \|\alpha_k d_k\|^2 = 0,$$

*and*

$$\lim_{k\to\infty} \|\alpha_k F(x_k)\|^2 = 0. \tag{2.25}$$

**Theorem 2.8.** *If Corollaries 2.1, 2.2 and 2.3 are satisfied, let $\{x_k\}$ sequence be generated by the Algorithm 1. Then,*

$$\lim_{k\to\infty} inf\|F_k\| = 0.$$

*Proof.* **Case 1.** If $\lim_{k\to\infty} inf\|d_k\| = 0$. Then, based on the definition of the direction, we have:

$$\lim_{k\to\infty} inf\|F_k\| = 0.$$

**Case 2.** If $\lim_{k\to\infty} inf\|d_k\| > 0$. Then, we have

$$\lim_{k\to\infty} inf\|F_k\| > 0.$$

By (2.25), we obtain

$$\lim_{k\to\infty} \alpha_k = 0.$$

Using (1.8) and (1.22), we obtain:

$$\|F_{k+1}\|^2 - \|F_k\|^2 \leq \omega_1\|\alpha_k F_k\|^2 - \omega_2\|\alpha_k d_k\|^2 + \sigma_k\|F_k\|^2. \tag{2.26}$$

If, (2.26) is not true, then, it means that, there exists a non-negative integer $i-1$ such that,

$$\|F_{k+1}\|^2 - \|F_k\|^2 > \omega_1\|r^{i-1}F_k\|^2 - \omega_2\|r^{i-1}d_k\|^2 + \sigma_k\|F_k\|^2.$$

Since, $\{\|F_k\|\}$ and $\{\|d_k\|\}$ are bounded, then, allowing $i\to\infty$, we have

$$\|F_{k+1}\|^2 - \|F_k\|^2 > \sigma_k\|F_k\|^2. \tag{2.27}$$

By rearranging (2.27), we obtain

$$\|F_{k+1}\|^2 > \Big(1+\sigma_k\Big)\|F_k\|^2. \tag{2.28}$$

Taking the summation on both sides of (2.28), we have:

$$\sum_{j=0}^{k}\|F_{j+1}\|^2 > \sum_{j=0}^{k}\Big(1+\sigma_j\Big)\|F_j\|^2. \tag{2.29}$$

From (2.29), we deduce that

$$\|F_1\|^2+\|F_2\|^2+\ldots+\|F_{k+1}\|^2 > \|F_0\|^2+\|F_1\|^2+\ldots+\|F_k\|^2+\sigma\Big(\|F_0\|^2+\|F_1\|^2+\ldots+\|F_k\|^2\Big). \tag{2.30}$$

However, (2.30) can reduce to

$$\|F_{k+1}\|^2 > \|F_0\|^2 + \sigma \sum_{j=0}^{k} \|F_j\|^2 > \|F_0\|^2.$$

Which implies that,

$$\|F_{k+1}\|^2 > \|F_0\|^2.$$

So,

$$\|F_{k+1}\| > \|F_0\|, \quad \textit{for some } k.$$

This contradicts Corollary 2.1. Thus, we finally conclude that,

$$\lim_{k\to\infty} \inf \|F_k\| = 0.$$

∎

## 3. Numerical Results

We evaluated the effectiveness of our suggested algorithm using different benchmark problems with various initial guess and dimensions by the following algorithm:

**MCG** stands for our proposed algorithm, and the following settings were made: $r = 0.2$, $\sigma_k = \frac{1}{(k+1)^2}$ and $\psi_1 = \psi_2 = 10^{-4}$.

A new derivative-free conjugate gradient **NDCG** is the method proposed in [6] and we also have the following: $\psi_1 = \psi_2 = 10^{-4}$, $r = 0.2$ and $\sigma_k = \frac{1}{(k+1)^2}$.

The codes were written in MATLAB 7.71GB (R2014a) and ran on a 2GB RAM PC with a 2.13 GHz CPU. The iterations would be terminated when the total number of 5000 are attained or $\|F(x_k)\| \leq 10^{-4}$. Twenty (20) test problems with various initial guess and dimensions (n values) were used to compare the two algorithms.

The Twenty (20) test problems used in the proposed algorithm are as follows:

**Problem 3.1.** [21]

$$\begin{aligned} F_1(x) &= exp(x_1) - 1, \\ F_i(x) &= exp(x_i) - 1, \\ F_n(x) &= exp(x_n) - 1, \\ i &= 1, 2, 3, \dots, n. \\ x_0 &= (-0.1, -0.1, \dots, -0.1)^T. \end{aligned}$$

**Problem 3.2.** [21]

$$\begin{aligned} F_1(x) &= x_1 - 3x_1(sin((x_1)/3) - 0.66) + 2, \\ F_i(x) &= x_i - 3x_i(sin((x_i)/3) - 0.66) + 2, \\ F_n(x) &= x_n - 3x_n(sin((x_n)/3) - 0.66) + 2, \\ i &= 1, 2, 3, \dots, n. \\ x_0 &= (-0.5, -0.5, \dots, -0.5)^T. \end{aligned}$$

**Problem 3.3.** [13]

$$F_i(x) = \log(x_i + 1) + \frac{x_i}{n},$$
$$i = 1, 2, \dots, n.$$
$$x_0 = (0.04, 0.04, \dots, 0.04)^T.$$

**Problem 3.4.** [1]

$$F_i(x) = x_i - (0.1)x_{i+1}^2$$
$$i = 1, 2, \dots, n-1.$$
$$x_0 = (0.25, 0.25, \dots, 0.25)^T.$$

**Problem 3.5.** [1]

$$F_i(x) = 2x_i - sin|x_i|,$$
$$i = 1, 2, \dots, n.$$
$$x_0 = (0.15, 0.15, \dots, 0.15)^T.$$

**Problem 3.6.** [1]

$$F_1 = x_1 - e^{cos\left(\frac{x_1 + x_2}{n+1}\right)},$$
$$F_i = x_i - e^{cos\left(\frac{x_{i-1} + x_i + x_{i+1}}{n+1}\right)},$$
$$F_n = x_n - e^{cos\left(\frac{x_{n-1} + x_n}{n+1}\right)},$$
$$i = 2, 3, \dots, n-1.$$
$$x_0 = (5, 5, \dots, 5)^T.$$

**Problem 3.7.** [21]

$$F_1(x) = 0.2x_1^2 - 2,$$
$$F_i(x) = 0.2x_i^2 - 2,$$
$$F_n(x) = 0.2x_n^2 - 2,$$
$$i = 1, 2, \dots, n.$$
$$x_0 = (-0.15, -0.15, \dots, -0.15)^T.$$

**Problem 3.8.** [7]

$$F_i(x) = (1 - x_i^2) + x_i(1 + x_l x_{n-2} x_{n-1} x_n) - 2,$$
$$i = 1, 2, \dots, n.$$
$$x_0 = (-0.03, -0.03, \dots, -0.03)^T.$$

**Problem 3.9.** [21]

$$F_1(x) = e^{x_1^2} - 1 - cos(1 - x_1),$$
$$F_i(x) = e^{x_i^2} - 1 - cos(1 - x_i),$$
$$F_n(x) = e^{x_n^2} - 1 - cos(1 - x_n),$$
$$i = 1, 2, \dots, n.$$
$$x_0 = (0.8, 0.8, \dots, 0.8)^T.$$

**Problem 3.10.** [7]

$$\begin{aligned} F_1(x) &= x_1 - x_2^2, \\ F_i(x) &= x_i - x_{i+1}^2, \\ i &= 1, 2, \ldots, n-1. \\ x_0 &= (0.05, 0.05, \ldots, 0.05)^T. \end{aligned}$$

**Problem 3.11.** [1]

$$\begin{aligned} F_i(x) &= 0.1(1 - x_i)^2 - e^{-x_i^2}, \\ F_n(x) &= \frac{n}{10}(1 - e^{-x_n^2}), \\ i &= 1, 2, \ldots, n-1. \\ x_0 &= (0.05, 0.05, \ldots, 0.05)^T. \end{aligned}$$

**Problem 3.12.** [13]

$$\begin{aligned} F_i(x) &= x_i - \frac{1}{n}x_i^2 + \frac{1}{n}\sum_{i=1}^{n} x_i + 1, \\ i &= 1, 2, \ldots, n. \\ x_0 &= (0.5, 0.5, \ldots, 0.5)^T. \end{aligned}$$

**Problem 3.13.** [13]

$$\begin{aligned} F_i &= 2x_i + \sin(x_i) - 1, \\ i &= 1, 2, 3, \ldots, n. \\ x_0 &= (1, 1, \ldots, 1)^T. \end{aligned}$$

**Problem 3.14.** [22]

$$F_i(x) = x_i - \left(1 - \frac{c}{2n}\sum_{j=1}^{n} \frac{(\mu_i x_j)}{(\mu_i + \mu_j)}\right)^{-1}, \ \textit{for } i = 1, 2, \ldots, n \quad \textit{wth } c \in [0, 1) \textit{ and}$$

$$\mu_i = \frac{i - 0.5}{n}, \ \textit{for } 1 \leq i \leq n. (\textit{In our experiment}, \ \textit{we take } c = 0.9).$$

$$x_0 = (0.1, 0.1, \ldots, 0.1)^T.$$

**Problem 3.15.** [1]

$$F(x) = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} x + (e_1^x - 1, \ldots, e_n^x - 1)^T.$$

$$x_0 = (-0.1, -0.1, \ldots, -0.1)^T.$$

**Problem 3.16.** [23]

$$\begin{aligned} F_i &= x_i \cos(x_i - \frac{1}{n}) - x_i, \\ i &= 1, 2, 3, \ldots, n. \\ x_0 &= (0.5, 0.5, \ldots, 0.5)^T. \end{aligned}$$

**Problem 3.17.** [23]

$$F_i = \cos(x_i - 1) + x_i - 1,$$
$$i = 1, 2, 3, \dots, n.$$
$$x_0 = (1, 1, \dots, 1)^T.$$

**Problem 3.18.** [24]

$$F_1 = 5x_1^2 - 2x_1 - 3,$$
$$F_i = 5x_i^2 - 2x_i - 3,$$
$$F_n = 5x_n^2 - 2x_n - 3,$$
$$i = 1, 2, 3, \dots, n-1.$$
$$x_0 = (3, 3, \dots, 3)^T.$$

**Problem 3.19.** [1]

$$F(x) = \begin{pmatrix} 2 & -1 & & & \\ 0 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} x + (sinx_1 - 1, \dots, sinx_n - 1)^T.$$
$$x_0 = (0.5, 0.5, \dots, 0.5)^T.$$

**Problem 3.20.** [23]

$$F_1 = x_1^2 - 4,$$
$$F_i = x_i^2 - 4,$$
$$F_n = x_n^2 - 4,$$
$$i = 1, 2, 3, \dots, n-1.$$
$$x_0 = (5, 5, \dots, 5)^T.$$

**Table 1.** Experimental results of MCG and NDCG algorithms for problems 3.1 - 3.10

| | | MCG | | | NDCG | | |
|---|---|---|---|---|---|---|---|
| Problems | Dimension | NI | Time(s) | $\|F(x_k)\|$ | NI | Time(s) | $\|F(x_k)\|$ |
| 3.1 | 1000 | 3 | 0.91353 | 2.16E-09 | — | — | — |
| | 10000 | 3 | 1.002982 | 6.82E-09 | — | — | — |
| | 100000 | 3 | 3.176163 | 2.16E-08 | — | — | — |
| 3.2 | 1000 | 8 | 0.333859 | 2.40E-05 | 10 | 0.151941 | 9.36E-05 |
| | 10000 | 8 | 0.726523 | 7.60E-05 | 11 | 0.744871 | 5.88E-05 |
| | 100000 | 9 | 3.400227 | 4.82E-05 | 13 | 3.744509 | 3.73E-05 |
| 3.3 | 1000 | 2 | 0.180006 | 1.48E-05 | 44 | 0.302014 | 3.67E-05 |
| | 10000 | 2 | 0.639259 | 2.23E-05 | — | — | — |
| | 100000 | 2 | 3.027584 | 9.34E-05 | — | — | — |
| 3.4 | 1000 | 3 | 0.477494 | 7.78E-09 | 5 | 0.114819 | 1.93E-05 |
| | 10000 | 3 | 0.651447 | 7.76E-08 | 6 | 0.648342 | 3.45E-06 |
| | 100000 | 3 | 2.942724 | 7.42E-07 | 6 | 2.854309 | 1.50E-05 |
| 3.5 | 1000 | 3 | 0.170331 | 2.24E-08 | — | — | — |
| | 10000 | 3 | 0.604953 | 7.10E-08 | — | — | — |
| | 100000 | 3 | 2.803624 | 2.24E-07 | 76 | 6.367274 | 2.33E-05 |
| 3.6 | 1000 | 2 | 0.15777 | 4.51E-07 | 3 | 0.109802 | 2.58E-05 |
| | 10000 | 2 | 0.596782 | 1.43E-10 | 3 | 0.648579 | 2.59E-08 |
| | 100000 | 4 | 3.252949 | 6.81E-06 | 4 | 3.23372 | 6.81E-06 |
| 3.7 | 1000 | 8 | 0.167542 | 2.84E-05 | 19 | 0.160408 | 7.36E-05 |
| | 10000 | 8 | 0.686853 | 8.97E-05 | 21 | 0.861724 | 6.11E-05 |
| | 100000 | 9 | 3.015206 | 7.51E-05 | 23 | 3.931699 | 5.07E-05 |
| 3.8 | 1000 | 6 | 0.188879 | 3.61E-05 | 10 | 0.141725 | 4.06E-05 |
| | 10000 | 7 | 0.745648 | 2.28E-05 | 11 | 0.778653 | 2.57E-05 |
| | 100000 | 7 | 3.614148 | 7.21E-05 | 11 | 3.712357 | 8.11E-05 |
| 3.9 | 1000 | 11 | 0.204847 | 6.77E-05 | 11 | 0.200002 | 6.77E-05 |
| | 10000 | 12 | 0.952941 | 8.07E-05 | 12 | 0.899824 | 8.07E-05 |
| | 100000 | 13 | 3.996135 | 9.61E-05 | 13 | 3.980791 | 9.61E-05 |
| 3.10 | 1000 | 3 | 0.139341 | 4.24E-09 | 5 | 0.114294 | 8.72E-05 |
| | 10000 | 3 | 0.603523 | 4.08E-08 | 6 | 0.782242 | 1.45E-05 |
| | 100000 | 3 | 3.104715 | 4.01E-07 | 7 | 3.071305 | 2.23E-05 |

**Table 2.** Experimental results of MCG and NDCG algorithms for problems 3.11 - 3.20

| | | MCG | | | NDCG | | |
|---|---|---|---|---|---|---|---|
| Problems | Dimension | NI | Time(s) | $\|F(x_k)\|$ | NI | Time(s) | $\|F(x_k)\|$ |
| 3.11 | 1000 | 29 | 0.26663 | 7.52E-05 | 48 | 0.282874 | 9.70E-05 |
| | 10000 | 32 | 0.928716 | 7.40E-05 | 53 | 1.025351 | 8.95E-05 |
| | 100000 | 35 | 4.827367 | 7.28E-05 | 58 | 6.165456 | 8.25E-05 |
| 3.12 | 1000 | 17 | 0.22852 | 3.23E-05 | 48 | 0.274158 | 2.03E-05 |
| | 10000 | 37 | 1.067673 | 1.01E-06 | 84 | 1.596403 | 6.30E-05 |
| | 100000 | 49 | 5.517871 | 3.17E-08 | 87 | 11.69978 | 2.38E-05 |
| 3.13 | 1000 | 13 | 0.178711 | 9.39E-05 | 13 | 0.130961 | 8.22E-05 |
| | 10000 | 13 | 0.708997 | 9.39E-05 | 13 | 0.616541 | 8.22E-05 |
| | 100000 | 13 | 2.857375 | 9.39E-05 | 13 | 2.827481 | 8.22E-05 |
| 3.14 | 1000 | 84 | 0.671679 | 7.19E-05 | 24 | 0.169855 | 3.89E-05 |
| | 10000 | 96 | 1.948775 | 8.04E-05 | 7 | 0.669502 | 8.19E-05 |
| | 100000 | 76 | 10.66796 | 8.23E-05 | 78 | 9.22312 | 3.72E-06 |
| 3.15 | 1000 | 19 | 1.132849 | 9.29E-05 | 21 | 1.372109 | 5.31E-05 |
| | 10000 | 19 | 58.18636 | 9.52E-05 | 21 | 66.01411 | 8.78E-05 |
| | 100000 | — | — | — | — | — | — |
| 3.16 | 1000 | 5 | 0.139434 | 9.49E-07 | — | — | — |
| | 10000 | 5 | 0.659638 | 3.07E-06 | — | — | — |
| | 100000 | 5 | 4.953931 | 9.72E-06 | — | — | — |
| 3.17 | 1000 | 32 | 0.277749 | 6.81E-05 | 61 | 0.384831 | 8.10E-05 |
| | 10000 | 34 | 0.976909 | 9.77E-05 | 67 | 1.426963 | 7.83E-05 |
| | 100000 | 37 | 4.873251 | 9.44E-05 | 73 | 9.98812 | 7.56E-05 |
| 3.18 | 1000 | 26 | 0.410418 | 6.20E-05 | — | — | — |
| | 10000 | 28 | 0.938922 | 7.06E-05 | — | — | — |
| | 100000 | 30 | 4.749132 | 8.04E-05 | — | — | — |
| 3.19 | 1000 | 22 | 1.244464 | 9.49E-05 | 28 | 2.033708 | 5.24E-05 |
| | 10000 | 26 | 2.013005 | 9.99E-05 | 33 | 131.454 | 5.24E-05 |
| | 100000 | — | — | — | — | — | — |
| 3.20 | 1000 | 11 | 0.177192 | 6.36E-05 | 19 | 0.564376 | 3.55E-05 |
| | 10000 | 12 | 0.730628 | 4.02E-05 | 20 | 0.851822 | 2.24E-05 |
| | 100000 | 13 | 3.393716 | 2.54E-05 | 20 | 4.004684 | 7.10E-05 |

Tables 1 and 2, contain the experimental results of the two algorithms, the total number of iterations and CPU time respectively are denoted by "NI" and "Time (s)", while "$\|F(x_k)\|$" is the magnitude of the function $F$. Both of these algorithms attempt to solve problem (1.1) according to the tables, and the efficiency of our proposed algorithm was established because it successfully solves some problems that the NDCG algorithm fails to solve. This is convincing evidence that, NDCG method fails to solve problems 3.1, 3.16 and 3.18 completely. We used (—) to represent a failure.

**Table 3.** The numerical results shown in Tables 1 and 2 are summarized.

| | ALGORITHM | | |
|---|---|---|---|
| | MCG | NDCG | Undecided |
| Number of all problems | 60 | 60 | |
| Problems solved with fewer number of iterations | 49 | 2 | 9 |
| Percentage | 81.67% | 3.33% | 15.00% |
| Problems solved with fewer CPU time | 40 | 18 | 2 |
| Percentage | 66.67% | 30.00% | 3.33% |

In the summarized Table 3, it is observed that, the MCG method is a winner compared to the NDCG method as fewer iterations and CPU time are needed to solve more problems respectively. The MCG method wins 81.67% (49 out of 60) of the problems that required fewer iterations compared to NDCG 3.33% (2 out of 60). It is also observed in the summarized result that, both MCG and NDCG algorithms solved 9 problems with equal iterations' number, equivalents to 15.00% indicated as undecided. The summary Table also shows that, in terms of CPU time, the suggested MCG algorithm performs better than the NDCG algorithm. The reported data indicates that 66.67% (40 out of 60) of the problems were solved by the MCG method Using less CPU time compared to 30.00% (18 out of 60) solved by the NDCG method. both MCG and NDCG algorithms solved 2 problems with equal CPU time, equivalents to 3.33% indicated as undecided.

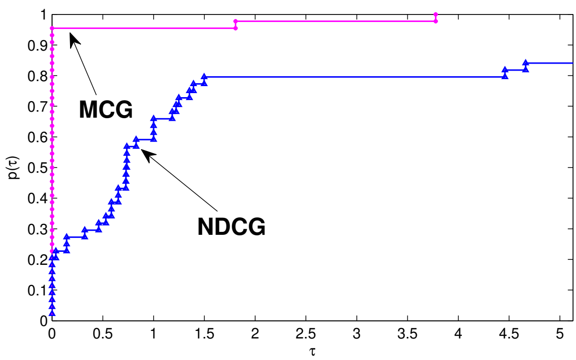


**Fig. 1.** Performance profiles for problems in relation to the number of iterations

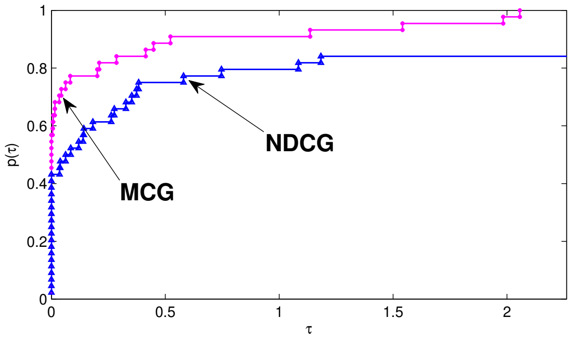


**Fig. 2.** Performance profiles for problems in relation to the CPU time (s)

Figures 1 and 2 demonstrate how our algorithm performs according to the CPU time (S) and number of iterations, as measured by Dolan and More profiles [25]. Specifically, we plot the proportion $P(\tau)$ of problems where the algorithm is within the best time for each method by a factor called $\tau$ and the top curve representing our algorithm.

## 4. Conclusion

We presented a modified conjugate gradient (MCG) method for finding approximate solution of nonlinear system of equations in this work, then we compare how well it performs with new derivative-free conjugate gradient (NDCG) method proposed in [6] by numerical tests. Using a non-monotone type line search [3], we proved that our suggested algorithm is globally converged to the solution of equation (1.1). Experimentally, our method demonstrates its robustness.

## References


[1] A.S. Halilu and M.Y. Waziri, A Transformed double steplength method for solving large-scale system of nonlinear equations, J. Numer. Math. Stoch., 9 (2017) 20–32.

[2] M.Y. Waziri, W.J. Leong, M.A. Hassan and M. Monsi, Jacobian computation-free Newton method for systems of non-linear equations, J. Numer. Math. Stoch., 2 (2010) 54–63.

[3] M. Fukushima and D. Li, A global and superlinear convergent Gauss-Newton base BFGS method for symmetric nonlinear equation, SIAM J. Numer. Anal., 37 (1999) 152–172.

[4] M.K. Dauda, M. Mamat, M.A. Mohamed and M.Y. Waziri, Improved quasi-Newton method via SR1 update for solving symmetric systems of nonlinear equations, Malay. J. Funda. Appl. Sci., 15 (2019) 177–130.

[5] M.Y. Waziri, W.J. Leong, M.A. Hassan and M. Monsi, A new Newton's method with diagonal Jacobian approximation for system of nonlinear equations, J. Math. Stat., 6 (2010) 246–252.

[6] X. Fang and Q. Ni, A new derivative-free conjugate gradient method for large-scale nonlinear systems of equations, Bull. Aust. Math. Soc., (2017).

[7] M.Y. Waziri and J. Sabiu, A new derivative-free conjugate gradient method and its global convergence for symmetric nonlinear equations, J. Numer. Math. Stoch., 2015 (2015) 8.

[8] Y.H. Dai and Y.X. Yuan, A nonlinear conjugate gradient with a strong global convergence properties, SIAM J. Optim., 10 (1999) 177–182.

[9] M.Y. Waziri, A. Yusuf and A.B. Abubakar, Improved conjugate gradient method for nonlinear system of equations, Comput. Appl. Math., 39 (2020) 1–17.

[10] W.W. Hager and H. Zhang, A survey of nonlinear conjugate gradient methods, Pacific J. Optim., 2 (2006) 35–58.

[11] R. Fletcher and C.M. Reeves, Function minimization by conjugate gradients, Comput. J., 7 (1964) 149–154.

[12] E. Polak and G. Ribiere, Note sur la convergence de methods de direction conjuguees, Revue Francais d'Informatique et de Recherche Operationnelle, 16 (1969) 35–43.

[13] M.J.D. Powell, Non-convex minimization calculations and the conjugate gradient method, In Numerical analysis, Springer, Berlin, Heidelberg, 1066 (1984) 122–141.

[14] E.L. Ioannis, T. Vassilis and P. Panagiotis, A descent hybrid conjugate gradient method based on memory-less BFGS update, Numer. Algorithms, 67 (2018) 31–48.

[15] N. Andrei, Another hybrid conjugate gradient algorithm for unconstrained optimization, Numer. Algorithms, 47 (2008) 143–156.

[16] N. Andrei, Hybrid conjugate gradient algorithm for unconstrained optimization, J. Optim. Theory Appl., 141 (2009) 249–264.

[17] N. Andrei, Scaled memory-less BFGS preconditioned conjugate gradient algorithm for unconstrained optimization, Optim. Method Softw., 22 (2007) 261–571.

[18] S.B. Kafaki and R. Ghanbari, A class of adaptive Dai-Laio conjugate gradient methods based on scaled memory-less BFGS update, 4OR, 4 (2016) 1–8.

[19] D.C. Liu and J. Nocedal, On the limited memory BFGS method for large scale optimization methods, Maths. Progm., 45 (1989) 503–528.

[20] L. Zang, W. Zhou and D. Li, Global convergence of modified Fletcher-Reeves conjugate gradient method with Armijo-type line search, Numerische mathematik, 104 (2006) 561–572.

[21] S. Jamilu, M. Y. Waziri and I. Abba, A new hybrid Dai-Yuan and Hestenes-Stiefel conjugate gradient parameters for solving system of nonlinear equations, MAYFEB J. Math., 1 (2017) 44–55.

[22] A. Perry, A modified conjugate gradient algorithm, Oper. Res., 26 (1978) 1037–1078.

[23] M.K. Dauda, M. Mustafa, M.Y. Waziri, A. Fadhila and S.M. Fatima, Inexact CG-method via SR1 update for solving systems of nonlinear equations, Far East Journal of Mathematical Science, 100 (2016) 1787–1804.

[24] M.K. Dauda, M. Mustafa, S.M. Fatima, S.M. Abubakar and M.Y. Waziri, Derivative free conjugate gradient method via Broyden's update for solving symmetric systems of nonlinear equations, In Journal of Physics: Conference Series, 1 (2019) 1366–1344.

[25] E. Dolan and J. More, Benchmark optimization software with performance profiles, Math. Program., 91 (2002) 201–213.