An Alternative Modified Conjugate Gradient Coefficient for Solving Nonlinear System of Equations

M.K. Dauda1*, Shehu Usman2, Hayatu Ubale3, M. Mamat4

1Department of Mathematical Sciences, Kaduna State University, Nigeria.
2Department of Mathematics and Statistics, Kaduna Polytechnic, Kaduna, Nigeria
3Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Kampus Gang Badak, 21300 Kuala Terengganu, Terengganu, Malaysia.

* Corresponding author: mkdflka@gmail.com

INTRODUCTION

In this paper, the following nonlinear system of equations of the form

\[ F(x) = 0, x \in \mathbb{R}^n; \]

where \( F: \mathbb{R}^n \rightarrow \mathbb{R}^n \) is continuously differentiable, Newton and quasi-Newton methods are the most widely used methods to solve such problems because they have very attractive convergence properties and practical application (see [1,2,3,4]). However, they are not usually suitable for large-scale nonlinear systems of equations because they require Jacobian matrix, or an approximation to it, at every iteration while solving optimization problems. The aim of this article is to propose solution to these shortcomings.

The iterative method used to solve (Eq 1) is formed by

\[ x_{k+1} = x_k + \alpha_k d_k, \quad k = 0,1,2, \ldots, \]

where \( \alpha_k > 0 \) is the step-size, \( x_k \) is the iterative point and \( d_k \) is the search direction. In this CG method, the search direction is described by

\[ d_k = \begin{cases} -F(x_k) & \text{if } k = 0 \\ -F(x_k) + \beta_k d_{k-1} & \text{if } k \geq 0 \end{cases} \]

where \( F: \mathbb{R}^n \rightarrow \mathbb{R}^n \) is continuously differentiable, while the scalar parameter \( \beta_k \) is the conjugate gradient coefficient. The following are examples of \( \beta_k \):

\[ \beta_{CD} = \frac{\nabla f(x_k)}{\nabla f(x_k)} \]

\[ \beta_{DY} = \frac{\nabla f(x_k)}{\nabla f(x_k)} \]

\[ \beta_{PR} = \frac{\nabla f(x_k)}{\nabla f(x_k)} \]

\[ \beta_{HS} = \frac{\nabla f(x_k)}{\nabla f(x_k)} \]

where \( g_{k+1} = \nabla f(x_{k+1}) \). The above methods are formula for Conjugate Descent (CD), Dai and Yuan (DY), Polak and Ribiere (PR), Liu and Storey (LS), Fletcher-Reeves (FR), Hestenes-Stiefel (HS) respectively. For further reading on \( \beta_k \) methods, refer to [5, 6, 7, 8, 9, 10, 11].

In [12], a method for solving unconstrained optimization problems is discussed. This article gives an improved version of the method by extending it to nonlinear system of equations as follows.

Initially, a nonlinear conjugate gradient method for the unconstrained optimization problem is considered.

\[ \min_{x \in \mathbb{R}^n} f(x), \]

where the function \( f \) is assumed to be continuously differentiable from \( R^n \) into \( R \), and the gradient \( \nabla f(x_k) \) at point \( x_k \) denoted as \( g_k \) is available. The nonlinear conjugate gradient method generates a sequence \( x_k \) by the recursive relation

\[ x_{k+1} = x_k + \alpha_k d_k, \quad k = 0,1,2, \ldots, \]

where \( \alpha_k \) is the step-length and the search direction \( d_k \) is updated by

\[ d_k = \begin{cases} -g(x_k) & \text{if } k = 0 \\ -g(x_k) + \beta_k d_{k-1} & \text{if } k \geq 0 \end{cases} \]

with

\[ \beta_k = \min(x_k + \alpha_k d_k) \]

where the \( \beta_k \) a scalar as is defined above[12]

objective function in problem (Eq 1) can be viewed as the first-order optimality condition of the problem (Eq 4), where \( F(x) \) is the gradient of \( f: \mathbb{R}^n \rightarrow \mathbb{R} \), with the aid of penalty function below

\[ f(x + \alpha_k d_k) = \min(x_k + \alpha d_k), \quad a > 0 \]

Now the following section described the proposed method for solving large-scale nonlinear systems of equations (Eq 1). The first section of this paper gives the introduction while the new CG method and its Algorithm are given in Section 2. In section 3, sufficient descent condition and global Convergence analysis are

© Readers Insight Publication

ARTICLE INFORMATION

Received: 22-Jul-2019
Revised: 12-Aug-2019
Accepted: 31-Aug-2019
DOI: Web Address

© 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
presented. Implementation of new method and discussion on Numerical results are presented in the last section, conclusion was drawn based on the performance profile.

The new CG method and its Algorithm

This section presents the new modified conjugate gradient as an alternative method for solving nonlinear system of equations. The new CG coefficient is denoted as $\beta_{DSHM}^k$, where the acronym DSHM comes from the names of the researchers and stands for Dauda, Shehu, Hayatu and Mustafa respectively. The equations is represented as follows:

$$
\beta_{DSHM}^k = \frac{\eta_k^2}{\eta_k^2 - \sum_{i=1}^{k-1} \eta_i^2} \quad (Eq \; 9)
$$

where $F_{k+1} = F_k(x_k)$ and $\| \|$ indicates the Euclidean norm of vectors. Note that, for easy representation $F_{k+1} = F_k(x_k)$ and $F_k = F_k(x_k)$. This paper considered the parameters $\alpha_i > 0, \beta_i > 0, r \in (0, 1)$ as constants and $\eta_k$ be a given positive sequence such that $\sum_{k=1}^{\infty} \eta_k < \infty$ and $\alpha_k = \max(1, r^k)$ that satisfy

$$
f(\bar{x}_k + \alpha_k \bar{d}_k) - f(x_k) \leq -\eta_k ||\alpha_k F(x_k)||^2 + \eta_k F(x_k) \quad (Eq \; 10)
$$

The line search (Eq 10) is used to calculate the stepsize. The following is the DSHM algorithm.

**Step 1:** Initialization. Given $x_0$, set $k = 0$.

**Step 2:** Compute $\beta_k^k$ based on (Eq 9).

**Step 3:** Compute $d_k$ based on (Eq 3). If $F_k = 0$, then stop.

**Step 4:** Compute $\alpha_k$ based on (Eq 10).

**Step 5:** Update the new point based on (Eq 2 or Eq 5).

**Step 6:** Convergence test and stopping criteria. If $f(x_{k+1}) < f(x_k)$ and $\|d_k\| < \varepsilon$ then stop. Otherwise go to Step 1 with $k = k + 1$.

Convergent Analysis

Every numerical analysis must satisfy convergence test, the new Modified CG coefficient is not an exception, because the algorithm satisfied the global convergence properties and the descent condition sufficiently.

**Sufficient Descent Condition of DSHM Algorithm.**

Consider the following theorem with exact line search as in (Eq 10).

**Theorem 1.** Confer DSHM method with (Eq 3), DSHM is said satisfy descent condition sufficiently if

$$
F_k^T d_k \leq -C ||F_k|| \quad \forall \; k \geq 0, \; C \in \mathfrak{R}.
$$

**Proof.** If $k = 0$, then clearly we can get $F_0^T d_0 = -C ||F_0||$ from Eq 3. Hence, condition $F_k^T d_k \leq -C ||F_k||$ holds true, at $k = 0$.

Next, for $k > 0$

Multiply (Eq 3) by $F_k^T$, then

$$
F_k^T d_k = F_k^T (-F_k + \beta_k d_{k-1}) = -F_k ||F_k||^2 + \beta_k F_k^T d_{k-1}.
$$

For exact line search, $F_k^T d_{k-1} = 0$.

Thus, $F_k^T d_k = -||F_k||^2$, which implies $d_k$ is sufficient descent direction. Hence, $F_k^T d_k \leq -C ||F_k||^2$ holds true, thus, the new CG coefficient DSHM possess sufficient descent condition. The proof is complete.\(\blacksquare\)

Implementation of new Method and Numerical Results

This section presents the numerical results of the implementation of the modified conjugate gradient coefficient for solving nonlinear system of equations. By solving several benchmark problems with their respective initial points using five (5) different dimensions ranging from 10 to 5000, the numerical results of the comparison between the proposed method and the result in [10] is presented in table 1 below. The Table displayed the performances result of each method in terms of number of iteration and CPU time. The meaning of each column in the table are respectively stated "Prob": Benchmark problem; "Dim": Dimension of the test problems; "Iter": the total number of iterations; "CPU": the CPU time in seconds. In case there is no available number of iterations or CPU time, it is considered as failed, and is denoted as "--". Based on Table 1, DSHM’s performance is more efficient than PRP. However, both methods failed at certain initial points as shown in the Table. Overall, DSHM manage to solve all given trigonometric functions with less time. Hence, DSHM can be defined as the best approximation method for solving large-scale systems of nonlinear equations.

**List of Benchmark Test Problem Used**

**PROBLEM 1[3]**

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

$x_0 = (0.9, 0.9, 0.9, -0.9)^T$ and $x_0 = (0.7, 0.7, 0.7, \ldots, 0.7)^T$.

**PROBLEM 2[10]**

$$
F_1(x) = x_1 - e^{x_1^2 + x_2^2},
$$

$F_2(x) = x_1 - e^{x_1^2 + x_2^2},$

$F_3(n) = x_1 - e^{x_1^2 + x_2^2};$

$i = 1, 2, 3, \ldots, n.$

**PROBLEM 3[4]**

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

and $x_0 = (0.4, 0.4, \ldots, 0.4)^T$.

All these problems are executed with four different initial points, starting from a point which is closer to the solution point till to the point further away from the solution point.

**Table** displaying the performances result of each method in terms of number of iteration and CPU time:

<table>
<thead>
<tr>
<th>Prob</th>
<th>Dim</th>
<th>DSHM</th>
<th>PRP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Iter</td>
<td>CPU</td>
<td>Iter</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>10</td>
<td>0.19289</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0.13689</td>
<td>10</td>
</tr>
<tr>
<td>100</td>
<td>17</td>
<td>0.03577</td>
<td>14</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>0.25247</td>
<td>26</td>
</tr>
<tr>
<td>5000</td>
<td>20</td>
<td>0.77096</td>
<td>21</td>
</tr>
<tr>
<td>1000</td>
<td>20</td>
<td>4.34788</td>
<td>23</td>
</tr>
<tr>
<td>5000</td>
<td>20</td>
<td>27.16918</td>
<td>32</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>9</td>
<td>0.01636</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0.19998</td>
<td>1000</td>
</tr>
<tr>
<td>1000</td>
<td>12</td>
<td>0.04573</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>10</td>
<td>7.11197</td>
<td>-</td>
</tr>
<tr>
<td>5000</td>
<td>27</td>
<td>0.93975</td>
<td>-</td>
</tr>
<tr>
<td>1000</td>
<td>26</td>
<td>5.53954</td>
<td>10</td>
</tr>
<tr>
<td>5000</td>
<td>26</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>2</td>
<td>0.44433</td>
</tr>
<tr>
<td>50</td>
<td>2</td>
<td>0.07508</td>
<td>17</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>0.21035</td>
<td>23</td>
</tr>
<tr>
<td>1000</td>
<td>2</td>
<td>0.81157</td>
<td>38</td>
</tr>
<tr>
<td>5000</td>
<td>2</td>
<td>2.12648</td>
<td>21</td>
</tr>
<tr>
<td>1000</td>
<td>3</td>
<td>16.16561</td>
<td>50</td>
</tr>
</tbody>
</table>

Copyright © 2018 Authors. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.
The code for the proposed method and all the problems stated (Benchmark Test Problem) are computed using MATLAB 7.1, R2009b programming environment and run on a personal computer 2.4GHz, Intel (R) Core (TM) i7-5500U CPU processor, 4GB RAM memory and on windows XP operator. Both the methods was implemented with the following parameters:

\[ \alpha_1 = 0.01, \tau = 0.2, \sigma_1 = 10^{-4}, \text{ and } \eta_k = \frac{1}{1 + \sqrt{k}}. \]

The search is terminated if: (i) \( \| p_0 \| \leq 10^{-6} \) or (ii) The total number of iteration exceeds 2000. In particular problem \( i \), the DSHM performs better if the number of iteration or the CPU (Time) of DSHM is less than the number of iteration or the CPU time corresponding to the performance in PRP method respectively. To further evaluate the performance of DSHM method relative to other CG method, the results are compiled in two graphs using performance profile based on Dolan and More [13]. Let \( P \) be the set of benchmark problems and let \( S \) be the set of algorithms. We define \( t_{ps} \) to be the number of iterations (or the CPU time in seconds) required to solve the problem \( p \in P \) by algorithm \( s \in S \). The comparison of each of the two measures is based on the performance ratio given by

\[ r_{ps} := \frac{t_{ps}}{\min(t_{ps})}. \]

Then the performance profile is defined by

\[ p_s(\tau) := \frac{1}{n} \sum_{p \in P} \mathbb{1}[t_{ps} \leq \tau], \]

for all \( \tau \in R \) where \( P(\tau) \) is the probability for solver \( s \in S \) that a performance ratio \( r_{ps} \) is within a factor \( \tau \) of \( R \) of the best possible ratio, where \( P \) is the number of benchmark problems.

<table>
<thead>
<tr>
<th>5000</th>
<th>0</th>
<th>3</th>
<th>63.36318</th>
<th>50</th>
<th>78.60937</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>10</td>
<td>3</td>
<td>0.17667</td>
<td>3</td>
<td>0.17823</td>
</tr>
<tr>
<td>50</td>
<td>3</td>
<td>0.06205</td>
<td>10</td>
<td>0.03633</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>4</td>
<td>0.1121</td>
<td>10</td>
<td>0.07558</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>4</td>
<td>0.81112</td>
<td>10</td>
<td>0.25212</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>4</td>
<td>0.67543</td>
<td>10</td>
<td>1.01534</td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
<td>0.57876</td>
<td>12</td>
<td>1.08977</td>
<td></td>
</tr>
<tr>
<td>5000</td>
<td>0</td>
<td>23.8762</td>
<td>10</td>
<td>31.01261</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. Presents Performance profile of DSHM and PRP methods with respect to the number of iterations for the problems 1-4. Figure 2 presents the graphical results of problems 1-4 relative to CPU time. The right side represents the test problems that are successfully solved by each method while the left side of the figure represents the method which is fastest in solving test problems. The top curve is the method that performs better in a time that was within a factor \( \tau \) of the best time. Therefore, from Figure 1, the proposed DSHM methods relative to the number of iteration, performs relatively better. Figure 2 gives the performance of DSHM methods relative to CPU time which outperforms PRP.

### Discussion and Conclusion

Considering the benchmark problems above with their respective initial points, the new CG, DSHM has been proven to be the best method when compared to standard PRP CG methods. It also manages to solve all of the tested problems within the shortest possible time. The proposed methods are indeed capable for solving large-scale systems of nonlinear equations.

### Acknowledgement

Authors would like to thank the editor and the anonymous referee for their constructive suggestions that improved the quality of this paper greatly.

### References

Author Biography

Dauda Muhammad Kabir was born in Kaduna State, Nigeria, on 19th September, 1979. He bagged his B.Sc., M.Sc. and Ph.D. degree in Mathematics from Bayero University Kano, Ahmadu Bello University, Zaria, Nigeria and Universiti Sultan Zainal Abidin, Malaysia respectively. He also obtained his Post Graduate Diploma in Education from National Teachers’ Institute, Nigeria. M.K.D. as popularly known is married and blessed with kids. A Senior Lecturer with department of Mathematical Sciences, Kaduna State University, Kaduna. A registered teacher and member of Teachers Registration Council of Nigeria (TRCN) and a member, Mathematical Association of Nigeria (MAN). Dauda, attended conferences and published a number of articles in the field of Mathematics. His main areas of research interest are computational and applied mathematics. e-mail: mkdfika@gmail.com