A MODIFIED DPRP CONJUGATE GRADIENT METHOD FOR UNCONSTRAINED OPTIMIZATION

Abdelrhaman Abashar\textsuperscript{1,3}, Mustafa Mamat\textsuperscript{1,4}, Mohd Rivaie\textsuperscript{2}, Muhammad Fauzi\textsuperscript{2} and Zabidin Salleh\textsuperscript{4}

\textsuperscript{1}Faculty of Informatics and Computing
Univetsiti Sultan Zainal Abidin (UniSZA)
Campus Tembila Besut, Terengganu
Malaysia
e-mail: tomcatkassala@yahoo.com
   must@unisza.edu.my

\textsuperscript{2}Department of Computer Science and Mathematics
Universiti Teknologi MARA (UiTM) Terengganu
Campus Kuala Terengganu, Malaysia
e-mail: rivaie75@yahoo.com
   fauziembong@yahoo.com

\textsuperscript{3}Faculty of Engineering
Red Sea University
Sudan

\textsuperscript{4}Pusat Pengajian Informatik dan Matematik Gunaan
Universiti Malaysia Terengganu (UMT)
Malaysia
e-mail: zabidin@umt.edu.my

Received: October 3, 2014; Accepted: December 23, 2014
2010 Mathematics Subject Classification: 65K10.
Keywords and phrases: unconstrained optimization, exact line search, global convergence.
Communicated by K. K. Azad
Abstract

Currently, Zhang [17] takes some modification of the Wei-Yao-Liu nonlinear conjugate gradient method suggested by Wei et al. [18] such that the modified method called NPRP method. Dai and Wen [19] make a simple modification to the NPRP called DPRP method. In this paper, we change denominator of DPRP method such that the modified DPRP method possesses global convergence under exact line search. Numerical results show that the proposed method is efficient for the given test functions when compared with classical formula and DPRP method.

1. Introduction

The nonlinear conjugate gradient method (CG) is considered to solve the following unconstrained optimization problem:

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

where \( f : \mathbb{R}^n \rightarrow \mathbb{R} \) is a continuously differentiable function. There are many methods for solving (1), but the conjugate gradient methods are popular method, the iterative formula of the conjugate gradient methods is given by

\[
x_{k+1} = x_k + \alpha_k d_k,
\]

where \( x_k \) is the current iterate point and \( \alpha_k \) is the step size, which is computed using exact line search

\[
f(x_k + \alpha_k d_k) = \min_{\alpha \geq 0} f(x_k + \alpha d_k),
\]

and \( d_k \) is the search direction defined by

\[
d_k = \begin{cases} 
-g_k & \text{if } k = 0, \\
-g_k + \beta_k d_{k-1} & \text{if } k \geq 1,
\end{cases}
\]

where \( g_k \) is the gradient of \( f(x) \) at the point \( x_k \) and \( \beta_k \) is a scalar. We know classical formulas for \( \beta_k \) are the Hestenes-Stiefel (HS) in 1952 [10],...
the Fletcher-Reeves (FR) in 1964 [11], the Polak-Ribiere and Polyak (PRP) in 1969 method [1], the conjugate descent method (CD) in 1987 [12], the Liu-Storey (LS) in 1992 [13] and the Dai-Yuan (DY) in 1999 [9]. The parameters of these $\beta_k$ are given as follows:

$$
\beta_k^{FR} = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\|\mathbf{g}_{k-1}\|^2},
$$

(5)

$$
\beta_k^{PRP} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{\|\mathbf{g}_{k-1}\|^2},
$$

(6)

$$
\beta_k^{HS} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{d}_{k-1}},
$$

(7)

$$
\beta_k^{LS} = \frac{\mathbf{g}_k^T (\mathbf{g}_k - \mathbf{g}_{k-1})}{-\mathbf{d}_{k-1}^T \mathbf{g}_{k-1}},
$$

(8)

$$
\beta_k^{DY} = \frac{\mathbf{g}_k^T \mathbf{g}_k}{(\mathbf{g}_k - \mathbf{g}_{k-1})^T \mathbf{d}_{k-1}},
$$

(9)

$$
\beta_k^{CD} = -\frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{d}_{k-1}^T \mathbf{g}_{k-1}}.
$$

(10)

The $\mathbf{g}_k$ and $\mathbf{g}_{k-1}$ are the gradients of $f(x)$ at the points $x_k$ and $x_{k-1}$, respectively. The most studied properties of conjugate gradient methods are its global convergence properties. Zoutendijk [3] proved that the FR method with exact line searches is globally convergent on general functions. Other researchers such as Al-Baali [14], Gilbert and Nocedal [5], studied the global convergence of FR under strong Wolfe line search. In this period, many of the modifications of the original CG methods had been extensively studied. Yuan et al. [21] gave a modified PRP method which we called MPRP method. There are good comparative studies of some new CG methods by Andrei [7], Mamat et al. [15], Jusoh et al. [16], Abashar et al. [20] and Rivaie et al. [6].
In this paper, we suggest a new modified nonlinear conjugate gradient method based on Dai and Wen [19], our new modified formula and algorithm is presented in Section 2. The global convergence of the new method is proven using the exact line search and the proof is obtained in Section 3. Some exciting numerical result is presented in Section 4 by comparing our new method with other CG method. Lastly, our discussion and conclusion are presented in Section 5 and Section 6, respectively.

2. Modified Formula and Algorithm

Recently, Wei et al. [18] gave a variant of the PRP method which is called the WYL method, that is,

$$\beta_k^{WYL} = \frac{g_k^T \left( g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1} \right)}{\|g_{k-1}\|^2}. \quad (11)$$

The WYL method and PRP methods have restart properties. Zhang [17] studied and improved based on WYL conjugate gradient method, and suggested

$$\beta_k^{NPRP} = \frac{\|g_k\|^2 - \left\| \frac{g_k}{g_{k-1}} \right\| \|g_k^T g_{k-1}\|}{\|g_{k-1}\|^2}. \quad (12)$$

Zhang [17] proved that the NPRP method satisfies descent condition under strong Wolfe line search. Dai and Wen [19] proposed modified NPRP method as follows:

$$\beta_k^{DPRP} = \frac{\|g_k\|^2 - \left\| \frac{g_k}{g_{k-1}} \right\| \|g_k^T g_{k-1}\|}{\mu \|g_k^T d_{k-1}\| + \|g_{k-1}\|^2}, \quad \mu > 1. \quad (13)$$

Based on the idea of Wei et al. [18] and the discussion of the above section, we come up with the modified DPRP method below

$$\beta_k^{MDPRP} = \frac{\|g_k\|^2 - \left\| \frac{g_k}{g_{k-1}} \right\| \|g_k^T g_{k-1}\|}{d_{k-1}^T (d_{k-1} - g_k)}.$$  (14)
Algorithm 2.1 describes the steps to execute this formula.

Algorithm 2.1

Step 1: Given $x_0 \in \mathbb{R}^n$, $\varepsilon = 10^{-6}$, set $d_0 = -g_0$ if $\|g_0\| \leq \varepsilon$ then stop.

Step 2: Compute $\alpha_k$ by applying exact line search, (3).

Step 3: $x_{k+1} = x_k + \alpha_k d_k$ if $\|g_{k+1}\| \leq \varepsilon$ then stop.

Step 4: Compute $\beta_k^{MDPRP}$ based on equation (14) and generated $d_k$ by (4).

Step 5: Set $k = k + 1$ go to Step 2.

3. Global Convergence Analysis

In this section, we present global convergent properties of $\beta_k^{MDPRP}$. First, we study the sufficient descent condition. The sufficient descent condition is defined as follows:

$$g_k^T d_k \leq -c \|g_k\|^2, \quad k \geq 0, \quad c \in (1, 0).$$

(15)

The following theorem shows that our new formula MDPRP with exact line search possesses the sufficient descent condition.

**Theorem 1.** Suppose that the $x_k$ and $d_k$ are generated by the method of form (2), (4) and (14), and the step size $\alpha_k$ is determined by the exact line search (3). Then (15) holds for all $k \geq 0$.

**Proof.** We prove by induction, if $k = 0$, then we already have $g_0^T d_0 \leq -c \|g_0\|^2$.

Hence, condition holds true, we also need to show that for $k \geq 1$, condition will also hold true from (4) multiply by $g_{k+1}^T$, then

$$g_{k+1}^T d_{k+1} = g_{k+1}^T (-g_{k+1} + \beta_{k+1} d_k),$$

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 + \beta_{k+1} g_{k+1}^T d_k.$$
For exact line search, we know that $g_{k+1}^T d_k = 0$. Thus,

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2.$$  \hspace{1cm} (17)

Hence, this condition holds true for $k + 1$. The proof is completed. \hfill \square

In order to show that the global convergence properties of CG methods, this basic assumption is always needed.

**Assumption 1**

(i) The level set $\{x \in R^n \mid f(x) \leq f(x_0)\}$ is bounded, where $x_0$ is the starting point.

(ii) In some neighborhood $N$ of $\Omega$, the objective function is continuously differentiable, and its gradient is Lipschitz continuous, namely, there exists a constant $l > 0$ such that $\|g(x) - g(y)\| \leq l\|x - y\|$ for any $x, y \in N$.

Under this assumption, we have the following lemma, which was proved by Zoutendijk [3].

**Lemma 1.** Suppose Assumption 1 holds, let $x_k$ be generated by Algorithm 2.1 and $d_k$ satisfies (15). Then we have

$$\sum_{k=0}^{\infty} \frac{\|g_k\|^4}{\|d_k\|^2} < \infty.$$ \hspace{1cm} (18)

Note that condition (18) is known as the Zoutendijk conditions. The proof of this lemma can be seen from [3].

**Lemma 2.** Consider any CG methods in the form (2), where $d_k$ is the search direction and $\alpha_k$ satisfies the exact line search. Then the following holds:

$$\frac{1}{\|d_k\|^2} \leq \frac{1}{\|g_k\|^2}.$$ \hspace{1cm} (19)
Proof.

\[ \|g_k + d_k\|^2 = \|g_k\|^2 + \|d_k\|^2 + 2g_k^T d_k, \quad (20) \]

substitute (17) in equation (20) to obtain

\[ \|g_k + d_k\|^2 = \|d_k\|^2 - \|g_k\|^2, \quad (21) \]

\[ \|g_k + d_k\|^2 + \|g_k\|^2 = \|d_k\|^2. \quad (22) \]

Hence, from (22), we get

\[ \|d_k\|^2 \geq \|g_k\|^2, \]

\[ \frac{1}{\|d_k\|^2} \leq \frac{1}{\|g_k\|^2}. \quad (23) \]

\[ \square \]

**Theorem 2.** Let the conditions in Assumption 1 hold true, \( x_k \) be generated by Algorithm 2.1 and \( d_k \) is the descent direction. If \( \alpha_k \) is obtained by the exact line search, then

\[ \lim_{k \to \infty} \|g_k\| = 0, \quad \forall k \geq 0. \quad (24) \]

**Proof.** We will prove by contradiction. Suppose there exists a constant \( \varepsilon > 0 \) such that

\[ \|g_k\| > \varepsilon, \quad \forall k \geq 0. \quad (25) \]

Squaring both sides of (3), we obtain

\[ \|d_k\|^2 = \|g_k\|^2 - 2\beta_k g_k^T d_{k-1} + (\beta_k^MDPRP)^2 \|d_{k-1}\|^2. \]

For exact line search \( g_k^T d_{k-1} = 0 \), we get

\[ \|d_k\|^2 = \|g_k\|^2 + (\beta_k^MDPRP)^2 \|d_{k-1}\|^2. \quad (26) \]
Substituting (14) in equation (26), we get
\[ \| d_k \|^2 \leq \| g_k \|^2 + \frac{\| g_k \|^4 \| d_{k-1} \|^2}{\| g_{k-1} \|^2} . \]  
(27)

Dividing both sides of (27) by $\| g_k \|^4$, we obtain
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \leq \frac{1}{\| g_k \|^2} + \frac{1}{\| g_{k-1} \|^2} , \]  
(28)

from Lemma 2, we get
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \leq \frac{1}{\| g_k \|^2} + \frac{1}{\| g_{k-1} \|^2} , \]  
(29)
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \leq \sum_{i=1}^{k} \frac{1}{\| g_i \|^2} . \]  
(30)

Then we get from (25) and (30) that
\[ \frac{\| g_k \|^4}{\| d_k \|^2} \geq \frac{\epsilon^2}{k} . \]  
(31)

This implies
\[ \sum_{k=1}^{\infty} \frac{\| g_k \|^4}{\| d_k \|^2} = \infty. \]  
(32)

This contradicts the Zoutendijk condition (18). Therefore, (24) holds. The proof is completed.

\[ \square \]

4. Numerical Results

In this section, we reveal the numerical results to test Algorithm 2.1; we use some of the test problem considered in Andrei [8] as shown in Table 1, to analyze the efficiency of our new formula as compared with other CG
methods FR, PRP and DPRP. The comparisons are based on the number of iterations and CPU time. We considered \( \varepsilon = 10^{-6} \) and \( \| g_k \| \leq \varepsilon \) as stopping criteria. All problems mentioned in Table 1 are solved by MATLAB 7.10.0 (R 2010a) subroutine programming. The CPU processor used was Core\textsuperscript{TM} i3-2328M (2.2GHZ, 3MB L3 Cache), with 6GB DDR3 RAM. The performance results shown in Figures 1 and 2, respectively, are based on the performance profile introduced by Dolan and More [2]. In this performance profile, they introduced the notion of a means to evaluate and compare the performance of the set of solvers \( S \) on a test \( P \). We assume that we have \( n_s \) solvers and \( n_p \) problem, for each problem \( p \) and solver \( s \), they defined \( t_{p,s} = \) computing time (the number of iterations, CPU time or others).

**Table 1.** A list of problem functions

<table>
<thead>
<tr>
<th>No.</th>
<th>Functions</th>
<th>Dim</th>
<th>Initial points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Six hump camel</td>
<td>2</td>
<td>(8, 8), (10, 10)</td>
</tr>
<tr>
<td>2.</td>
<td>Booth</td>
<td>2</td>
<td>(25, 25), (100, 100)</td>
</tr>
<tr>
<td>3.</td>
<td>Treccani</td>
<td>2</td>
<td>(5, 5), (10, 10)</td>
</tr>
<tr>
<td>4.</td>
<td>Zettl</td>
<td>2</td>
<td>(20, 20), (50, 50)</td>
</tr>
<tr>
<td>5.</td>
<td>Three hump camel</td>
<td>2</td>
<td>(10, 10), (20, 20)</td>
</tr>
<tr>
<td>6.</td>
<td>Rosenbrock</td>
<td>2, 4, 10, 100, 500, 1000, 10000</td>
<td>(20, 20, ..., 20), (30, 30, ..., 30)</td>
</tr>
<tr>
<td>7.</td>
<td>Penalty</td>
<td>2, 4, 10, 100</td>
<td>(3, 3, ..., 3), (100, 100, ..., 100)</td>
</tr>
<tr>
<td>8.</td>
<td>Diagonal 2</td>
<td>2, 4, 10, 100, 500, 1000</td>
<td>(5, 5, ..., 5), (20, 20, ..., 20)</td>
</tr>
<tr>
<td>9.</td>
<td>Shallow</td>
<td>2, 4, 10, 100, 500, 1000, 10000</td>
<td>(50, 50, ..., 50), (100, 100, ..., 100)</td>
</tr>
<tr>
<td>10.</td>
<td>Extended tridiagonal 1</td>
<td>2, 4, 10, 100, 500, 1000</td>
<td>(40, 40, ..., 40), (60, 60, ..., 60)</td>
</tr>
<tr>
<td>11.</td>
<td>Raydan 1</td>
<td>2, 4, 10, 100</td>
<td>(3, 3, ..., 3), (5, 5, ..., 5)</td>
</tr>
<tr>
<td>12.</td>
<td>White and Holst</td>
<td>2, 4, 10, 100, 500, 1000, 10000</td>
<td>(3, 3, ..., 3), (9, 9, ..., 9)</td>
</tr>
<tr>
<td>13.</td>
<td>Quadratic QF2</td>
<td>2, 4, 10, 100, 500, 1000</td>
<td>(5, 5, ..., 5), (100, 100, ..., 100)</td>
</tr>
</tbody>
</table>
14. Diagonal 4 2, 4, 10, 100, 500, 1000, 10000 (2, 2, ..., 2), (15, 15, ..., 15)
15. Extended denschnb 2, 4, 10, 100, 500, 1000, 10000 (5, 5, ..., 5), (25, 25, ..., 25)
16. Hager 2, 4, 10 (21, 21, ..., 21), (23, 23, ..., 23)
17. Generalized tridiagonal 1 2, 4, 100 (23, 23, ..., 23), (35, 35, ..., 35)
18. Generalized quartic 2, 4, 10, 100 (2, 2, ..., 2), (40, 40, ..., 40)
19. Extended beale 2, 4, 10, 100, 500, 1000, 10000 (7, 7, ..., 7), (10, 10, ..., 10)
20. Himmelblau 2, 4, 10, 100, 500, 1000, 10000 (10, 10, ..., 10), (200, 200, ..., 200)
21. Quadratic penalty 2, 4, 10, 100, 500 (10, 10, ..., 10), (100, 100, ..., 100)
22. Perturbed quadratic 2, 4, 10, 100 (3, 3, ..., 3), (30, 30, ..., 30)
23. Nonscomp 2, 4, 100, 500 (13, 13, ..., 13), (15, 15, ..., 15)

**Figure 1.** Performance profile based on the number of iteration.
Figure 2. Performance profile based on the CPU time.

We require baseline for comparisons, we compare the performance on problem $p$ by solver $s$ with the best performance by any solver on this problem, that we use the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in S\}}.$$\)

We assume that parameter $r_M \geq r_{p,s}$ for all $p,s$ is chosen, and $r_{p,s} = r_M$ if and only if solver $s$ does not solve problem $p$. The performance of solver $s$ on any given problem might be of interest, but we would like to obtain overall assessment of the performance of the solver. If we define

$$p_s(t) = \frac{1}{n_p} \text{size}\{p \in P : r_{p,s} \leq t\},$$

then $p_s(t)$ is the probability for solver $s \in S$ that a performance ratio $r_{p,s}$ is within a factor $t \in R$ of the best possible ration. The function $p_s$ is the cumulative distribution for the performance ratio. The performance profile
for a solver is non-decreasing, piecewise constant function, continuous from the right at each break point, the value of \( p_s(l) \) is the probability that the solver will win over the rest of the solvers, in general, a solver with high value of \( p(t) \) or at the top right of the figure represents the best solver.

5. Discussion

From Figures 1 and 2, it is shown that the performance of these methods is relative to the number of iterations and CPU time, respectively. We show that our proposed algorithm is better when compared with FR which solves only 87% of test problem, and PRP which solves 97% of test problem; our new algorithm solved 100% of the test problem functions and faster than DPRP which can solve 97% of problems.

6. Conclusion

In this paper, we proposed a new and simple \( \beta_k \) that has a global convergence property. Numerical results have shown that the new \( \beta_k \) performs better than FR, PRP and DPRP. In the future, we intend to test our new formula using inexact line search.

Acknowledgements

The authors would like to thank The Ministry of Higher Education Malaysia (MOHE) for funding this research under Fundamental Research Grant Scheme (vot 59256).

References


