A New Spectral Conjugate Gradient Method for Nonlinear Unconstrained Optimization

Ibtisam A. Masmali, Zabidin Salleh and Ahmad Alhawarat

1Department of Mathematics, Faculty of Science, Jazan University, Jazan, Saudi Arabia
2Department of Mathematics, Faculty of Ocean Engineering Technology and Informatics, University Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia
3Department of Mathematics, Faculty of Ocean Engineering Technology and Informatics, University Malaysia Terengganu, 21030 Kuala Nerus, Terengganu, Malaysia

Abstract: The conjugate gradient method is widely used to solve large scale unconstrained optimization problems. However, the rate of convergence conjugate gradient method is linear unless it restarted. In this study, we present a new spectral conjugate gradient modification formula with restart property obtains the global convergence and descent properties. In addition, we proposed a new restart condition for Fletcher-Reeves conjugate gradient formula. The numerical results demonstrated that the modified Fletcher-Reeves parameter and the new CG formula with their restart conditions are more efficient and robustness than other conventional methods.

Keywords: Conjugate Gradient, Global Convergence, Descent Condition

Introduction

We consider the following problem:

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

(1)

where, \( f: \mathbb{R}^n \rightarrow \mathbb{R} \) is continuous and differentiable function and its gradient \( g(x) = \nabla f(x) \) is available. Iterative methods are usually used to solve (1), as follows:

\[ x_{k+1} = x_k + \alpha_k d_k, k = 1, 2, ..., \]  

(2)

starting from initial point \( x_1 \in \mathbb{R}^n \), where \( \alpha_k \) is obtained by some line search. The search direction \( d_k \) is defined by:

\[ d_k = \begin{cases} -g_k, k = 1, \\ -g_k + \beta_k d_{k-1}, k \geq 2, \end{cases} \]  

(3)

where, \( g_k = g(x_k) \) and \( \beta_k \) is known as the conjugate gradient parameter.

The exact line search can be used to find the steplength \( \alpha_k \). Suppose that \( \phi(\alpha) = f(x_k + \alpha d_k) \) which is problem that departs from \( x_k \) to find a step length in the direction \( d_k \) such that \( \phi(\alpha) < \phi(0) \). If the step length is defined such that the search direction minimized i.e., this line search is called exact line search where this line search is expensive. Therefore, using the inexact line search with less computation load is better. The inexact line search in particular Strong Wolfe-Powell (SWP) line search inherits the advantages of exact line search and computationally inexpensive. Thus, to reduce the computation cost of exact line search and also to reduce evaluations of the objective function and gradient function, usually the inexact line search is employed. SWP line search is more preferable than other line searches. The SWP line search is defined by:

\[ f(x_k + \alpha d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k \]  

and:

\[ g(x_k + \alpha d_k)^T d_k \leq \sigma |g_k^T d_k| \]  

(6)

where, \( 0 < \delta < \sigma < 1 \). The Weak Wolfe-Powell (WWP) (Wolfe, 1969; 1971) line search given by (5) and:

\[ g(x_k + \alpha d_k)^T d_k \leq \sigma |g_k^T d_k| \]  

(7)

The convergence of CG method will not be linear if we restart CG method (Powell, 1977). Beale (1972)
recommended the use of the two-term CG method instead 
\(d_k = -g_k, \forall k \geq 1\) as the restart search direction. Powell 
(1984) recommended restarting \(d_k\) using Beale’s method if:

\[
\|g_t^k g_{i-1}\| > 0.2 \|g_t^k\|, \quad (8)
\]

Dai and Yuan (1998) present the following restart 
criterion:

\[
\|g_t^k g_{i-1}\| > \tau \|g_t^k\|, \tau \in (0, 1). \quad (9)
\]

The famous formulas for \(\beta_k\) are the Hestenes-Stiefel 
(HS) (Hestenes and Stiefel, 1952), Fletcher-Reeves (FR) 
(Fletcher and Reeves, 1964) and Polak-Riêrie-Polyak 
(PR) (Polak and Ribiere, 1969) formulas, which are 
defined as follows:

\[
\beta_k^{HS} = \frac{g_t^k y_k}{d_{i-1}^T y_k}, \quad \beta_k^{FR} = \frac{g_t^k g_k}{\|g_t^k\|}, \quad \beta_k^{PRP} = \frac{g_t^k y_k}{\|g_t^k\|},
\]

where, \(y_k = g_k - g_{k-1}\).

Polak and Ribiere (1969) proved CG method with the 
PRP formula and by using exact line search is convergent. Powell 
(1984) show that the PRP fail to satisfy the 
convergence by using an example even the exact line is 
used. Powell recommended to use the non-negative of PRP 
formula and by using exact line search is convergent. Gilbert 
and Nocedal (1992) suggest to use PRP as follows:

\[
\beta_k^{PRP} = \max \{0, \beta_k^{PRP}\}
\]

Zoutendijk (1970) obtain the global convergence of 
FR formula with CG method and the exact line search. 
Al-Baali (1985) proved FR method with SWP line search 
when \(\sigma < 1/2\) and SWP line search is employed, Guanghui 
et al. (1995) extended the proof to the case for \(\sigma \leq 1/2\).

Alhawarat et al. (2017) presented the following formula:

\[
\beta_k^{zPRP} = \begin{cases} 
\frac{\|g_t^k f - \mu_t \|}{\|g_t^k f\|}, & \text{if } \|g_t^k f\| > \mu_t \|g_t^k g_{i-1}\|, \\
0, & \text{otherwise},
\end{cases} \quad (10)
\]

where, \(\mu_t\) is defined as follows:

\[
\mu_t = \frac{\|x_t - x_{t-1}\|}{\|x_t\|}
\]

Kaelo et al. (2020) proposed the following CG formula:

\[
\beta_k^{AC} = \left\{ \begin{array}{ll}
\frac{\|g_t^k f - \mu_t \|}{\|g_t^k f\|}, & \text{if } 0 < \|g_t^k g_{i-1}\| < \|g_t^k f\|
\
\max \{d_{i-1}^T y_{i-1}, -g_t^k d_{i-1}\}, & \text{otherwise.}
\end{array} \right.
\]

As we know that in the case of the function is quadratic 
i.e., \(f(x) = g^T x + (1/2)x^T H x\) and the step size obtained by exact 
line search (3), the CG method satisfy the conjugacy 
criterion: \(d_t^T H d_j = 0, \forall i \neq j\). By using the mean value 
theorem and exact line search with Eq. (2) we can obtain \(\beta_k^{AC}\). 
From quasi-Newton method, BFGS method and the limited 
memory (LBFGS) method and using (2), Dai and Liao 
(2001) present the following conjugacy condition:

\[
d_{i-1}^T y_{i-1} = -g_t^k s_{i-1},
\]

where, \(s_{i-1} = x_i - x_{i-1}\) and \(t \geq 0\). In the case of \(t = 0\) Eq. 
(8) becomes the classical conjugacy condition. By 
using (2) and (8), (Kaelo et al., 2020) proposed the following CG formula:

\[
\beta_k^{AC} = \frac{g_t^k y_k}{d_{i-1}^T y_{i-1}} - \frac{g_t^k s_{i-1}}{d_{i-1}^T y_{i-1}} 
\]

However, \(\beta_k^{AC}\) face the same problem as \(\beta_k^{PRP}\) and \(\beta_k^{AC}\) 
i.e., \(\beta_k^{AC}\) is not non-negative in general. Thus (Dai 
and Liao, 2001) replaced Eq. (9) by:

\[
\beta_k^{AC} = \max \{\beta_k^{AC}, 0\} - \frac{g_t^k s_{i-1}}{d_{i-1}^T y_{i-1}}.
\]

Hager and Zhang (2005; 2013) presented a modified 
CG parameter that satisfies the descent property for any 
inexact line search with \(g_t^k d_k \leq -(7/8)\|g_t^k\|\). This formula 
given is as follows:

\[
\beta_k^{HZ} = \max \{\beta_k^{AC}, \eta_k\}
\]

where, \(\beta_k^{AC} = \frac{1}{d_{i-1}^T y_i} \left( y_i - 2d_{i-1}^T y_i g_t^k \right) g_t^k\), 
\(\eta_k = \frac{1}{\|g_t^k\| \min (\eta, \|g_t^k\|)}\) and \(\eta > 0\) is a constant.

Notes that if \(t = 2\), then \(\beta_k^{AC} = \beta_k^{AC}\).

The positive scalar denoted by \(\theta_t\) Hence, \(d_k\) given as:

\[
d_k = -\theta_t g_t + \beta_t d_{i-1}.
\]
when, $\theta_k = 1$, the search direction is a classical CG method. If $\beta_k = 0$, then there are two possibilities of $\theta_k$. If $\theta_k = \Delta f(x_k)^{-1}$ or an approximation of it, then the search direction is Newton or Quasi-Newton, respectively.

The New Formula and the Algorithm

Here, we construct the following new modification to improve the efficiency and robustness of DY CG formula and robustness of PRP CG method.

For $k = 1$, $d_k = -g_k$.

For $k \geq 2$:

$$d_k^{\text{NEW}} = \begin{cases} -\theta_k g_k + \beta_k d_{k-1}, & \text{if} \ g_k^T d_{k-1} \geq 0, \\ -g_k + \beta_k^{\text{REP}} d_{k-1}, & \text{else,} \end{cases}$$

(11)

where, $\|\cdot\|$ means the Euclidean norm and

$$\theta_k = 1 + \frac{g_k^T d_{k-1}}{g_{k-1}^T d_{k-1}} \quad \text{and:} \quad \beta_k^{\text{REP}} = \frac{\|g_k\|}{\|d_{k-1}\| (g_k - g_{k-1})}. $$

Algorithm 1

Step 1 Provide a starting point $x_0$. Set the initial search direction $d_1 = -g_1$. Let $k = 1$.

Step 2 If a stopping criteria is satisfied, then stop.

Step 3 Compute $d_k$ based on (2) with (11).

Step 4 Compute $\alpha_k$ using (4) and (6).

Step 5 Update $x_{k+1}$ based on (1).

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

In Algorithm 1, note that after the step $k = k+1$, the iterates $x_k = x_{k+1}$ takes place after every iteration. The other iterations are updated in a similar manner as $x_k$.

In following section, we present the global convergence property of the new formula (11). In case of $d_k^{\text{NEW}} = 0$, then the search direction becomes the steepest descent (negative gradient) which mean the stationary point is obtained.

Convergence of CG Algorithm with the Search Direction $d_k^{\text{NEW}}$

Assumption 1

A. The level set $\Omega = \{x | f(x) \leq f(x_1)\}$ is bounded, that is, a positive constant $M$ exists such that

$$\|x\| \leq M, \forall x \in \Omega.$$

B. In some neighbourhood $n$ of $\Omega$, $f$ is continuously differentiable and its gradient is Lipschitz continuous; that is, for all $x, y \in N$, there exists a constant such that:

$$\|g(x) - g(y)\| \leq L \|x - y\|.$$ 

This assumption implies that there exists a positive constant $B$ such that:

$$\|g(u)\| \leq B, \forall u \in N.$$ 

The descent condition:

$$g_k^T d_k \leq -\|g_k\|, \forall k \geq 1, \quad (12)$$

Al-Baali (1985) modified (12) to the following form and used it to prove the FR method:

$$g_k^T d_k \leq -c \|g_k\|, \forall k \geq 1, \quad (13)$$

where, $c \in (0,1)$. Equation (13) is the sufficient descent condition. Note that the general form of the sufficient descent condition is (14) with $c > 0$.

Descent and Convergence Properties for $d_k^{\text{NEW}}$ with the SWP Line Search

In fact, we have two types of global convergence; weak global convergence and strong global convergence both of them imply the stationary point for optimization problem. However, the convergence and the descent properties will not give any sense in terms of the efficiency for CG methods; for example, FR formula has global convergence properties with poor efficiency. Thus, to improve the efficiency when the method cycle does not reach a solution the CG algorithm should be restarted. In the following section, we will present a new CG method with restart property by using the steepest descent method.

The following lemma is called Zoutendijk condition (Al-Baali, 1985).

Lemma 3.1

Suppose assumption 1 is holds. Suppose method in the form (1), (2) and $\alpha_k$ satisfies the WWP line search (5) and (6), where the search direction satisfied. Then

$$\sum_{k=0}^{\infty} \|g_k d_k\|^2 < \infty. \quad (14)$$

Also we can extended Eq. (14) to the following form:

$$\sum_{k=0}^{\infty} \|g_k\|^2 < \infty. \quad (15)$$
Kaelo et al. (2020) present the following theorem for global convergence properties:

**Theorem 3.1**

Let assumption 1 holds. Suppose any CG method in the form (1) and (2), where \( d_k \) is a descent direction and \( a_k \) is obtained by the SWP line search. If:

\[
\sum_{i=1}^{\infty} \frac{1}{\|g_i\|^2} = \infty,
\]

then:

\[
\lim_{k \to \infty} \inf \|g_k\| = 0.
\]

**Theorem 3.2**

Let the sequence \( \{g_k\} \) and \( \{d_k\} \) are generated by the methods (2), (3) and (11), then (13) holds.

Proof. By using proof by induction. From (3) for \( k = 1 \), \( g_0 d_{i-1} = -\|g_i\|^2 \). Suppose that it is true until \( k - 1 \), i.e.,

\[
g_{k-1} d_{k-1} < 0, \text{ for } i = 1, 2, \ldots, k - 1 \text{ then we have the following two cases:}
\]

**Case 1** \( g_i d_{i-1} \geq 0 \):

\[
d_i = -\frac{g_i^T d_{i-1}}{g_i^T d_{i-1}} g_i - g_i + \frac{\|g_i\|^2}{d_{i-1}^T (g_i - g_{i-1})} d_{i-1},
\]

Multiply both sides by \( g_i^T \):

\[
g_i^T d_i = -\|g_i\|^2 - \|g_i\|^2 + \frac{\|g_i\|^2}{d_{i-1}^T (g_i - g_{i-1})} g_i^T d_{i-1},
\]

Since \( g_i^T d_{i-1} \geq 0 \):

\[
\begin{align*}
g_i^T d_i & = -\|g_i\|^2 \\
& \leq -\|g_i\|^2 \\
& < 0
\end{align*}
\]

**Case 2** \( g_i^T d_{i-1} < 0 \):

\[
d_i = -g_i + \beta^{\text{exp}} d_{i-1},
\]

Multiply both sides by \( g_i^T \):

\[
g_i^T d_i = -\|g_i\|^2 + \beta^{\text{exp}} g_i^T d_{i-1},
\]

By using \( g_i^T d_{i-1} < 0 \) and \( \beta^{\text{exp}} \geq 0 \), we obtain \( g_i^T d_i < 0 \).

**Theorem 3.3**

Let assumption 1 holds. Assume \( \{g_k\} \) and \( \{d_k\} \) are obtained by algorithm 1 in which \( a_k \) is obtained by the WWP line search and (13) holds. Then the

\[
\lim_{k \to \infty} \inf \|g_k\| = 0.
\]
function is plate shaped; and the Sum Squares function is bowl shaped. As the CG method is useful for small-and large-scale optimisation problems, we also select the dimensions of the functions, which varied from 2 to 10000. All of the functions are nonlinear. In Table 1, “Gen” denotes generalised, “Ext” denotes extended, “Dim” denotes dimension/s.

We employed the MATLAB programming environment (ver. 7.9). The results are shown in Fig. 1 and 2, in which a performance measure introduced by Dolan and More (2002) was employed.

The comparison include PRP+, FR, $d_k^{\text{Ext}}$ and FR* methods. $||g_k|| \leq 10^{-6}$ is used as the stopping criteria for all algorithms. To obtain the step length we used strong Wolfe-Powell line search with $\delta = 0.01$ and $\sigma = 0.1$.

Since we are interested to find the stationary point/s for optimization problems, we selected more than one initial point to test every function in Table 1 the dimension of functions between 2 and 5000. Different initial points almost will obtain different stationary points, which imply that more than one solution for multimodal functions. Hence, we obtain the best solution. In addition, we select small and large dimensions for every function. The ranges of dimensions are chosen between 2 and 10000. Thus, we conclude that using different dimension and different initial points to obtain the results will be more convince than using original initials and one dimension. However, the starting point needs more study.

Figure 1 and 2 show that the curve of the new formula $\left( \beta_k^{\text{Ext}} \right)$ is uppermost of all curves. In addition, it is clear that FR* formula is better than original FR formula which demonstrates the discussion that presented by Powell and the program is terminated by the user when the number of iterations exceeds 1000. The PRP+ formula is efficient since its curve started uppermost other curves. However, it is not satisfied the descent property with SWP line search. Thus, the program is terminated automatically.

In addition we present the following two functions the first one called Extended Beale function which given by the following formula:

$$f(x) = \sum_{i=1}^{n} \left( 1.5 - x_{2i-1} (1 - x_{2i}) \right)^2$$

$$+ \left( 2.25 - x_{2i-1} (1 - x_{2i}) \right)^2 + \left( 2.625 - x_{2i-1} (1 - x_{2i}) \right)^2$$

Number of variables (n): 500, 1000, 5000, 10000 with initial points: (-1,-1, ..., -1), (.5,.5, …, .5), (1,1,……,1), (2,2,…, 2).

This function has only one global minimum surrounded by a flat plateau. At the four corners lie four ascending steep walls that become smaller at the tip. These steep walls become higher as the value of the two variables increases. The minimum is $x^* = (3,0.5)$ and the function value is $f(x^*) = 0$ for two variable functions (note Fig. 3 for a three-dimensional graph).

**Table 1: A list of test functions**

| Function                          | Initial points                                      |
|-----------------------------------|-----------------------------------------------------|
| Ext. White & Holst function,      | (-1.2,1,-1.2,…), (5.5,…,5), (10,10,…,10), (15,15,…,15) |
| Ext. Rosenbrock function,         | (-1.2,1,-1.2,…), (5.5,…,5), (10,10,…,10), (15,15,…,15) |
| Six hump function                 | (1,1), (5,5), (10,10), (15,15)                      |
| Ext. Beale function               | (-1,-1,…,-1), (5,5,…,5), (10,10,…,10), (15,15,…,15) |
| Three hump function               | (1,1), (5,5), (10,10), (15,15)                      |
| Ext. Himmelblau function          | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Diagonal 2 function               | (0.2,0.2,…,0.2), (0.25,0.25,…,0.25), (0.5,…,0.5), (1,1,…,1) |
| NONSCOMP function                 | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Ext. DENSCHNB function            | (2,2,…,2), (2,2,…,2), (5,5,…,5), (10,10,…,10)       |
| Shallow function                  | (1,1), (5,5), (10,10), (15,15)                      |
| Booth function                    | (2,2,…,2), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| DIXMAANA function, [26]           | (2,2,…,2), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| DIXMAANB function                 | (2,2,…,2), (-1,-1,…,-1), (0,0,…,0), (1,1,…,1)       |
| NONDIA function                   | (2,2,…,2), (-1,-1,…,-1), (0,0,…,0), (1,1,…,1)       |
| Ext. Tridiagonal 1 function       | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| DQDRTIC function                  | (-1,-1,…,-1), (1,1,…,1), (2,2,…,2), (3,3,…,3)       |
| Diagonal 4 function               | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Raydan 2 function                 | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Ext. DENSCHNB function            | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| A Quadratic function QF2           | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Zettl function                    | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Extended Cliff                    | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Ext. Powell function              | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Generalized Quartic GQ1 function   | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
| Ext. Block Diagonal BD1 function   | (1,1,…,1), (5,5,…,5), (10,10,…,10), (15,15,…,15)    |
Fig. 1: Performance profile based on the number of iteration

Fig. 2: Performance profile based on the CPU time

Fig. 3: Extended Beale function in 3D
And the second function called
Perturbed quadratic function:

\[ f(x) = \left( \sum_{i=1}^{n} x_i \right)^2 + \sum_{i=1}^{n} \frac{i}{100} x_i^2 \]

Initial points: (0.5, 0.5, ..., 0.5)

This function has a smooth curve, look like dish shape and has a minimum value at \( x^* = (0,0) \) and the function value is \( f(x^*) = 0 \), where lies at the bottom for two variable functions (Fig. 4 for a three-dimensional graph).

Moreover we present another strong comparison between ATAZ and CG-Descent is given with benchmark functions in Table 2. The numerical results in Fig. 5, 6 and 7 show that the new modification ATAZ is better than CG-Descent in term of number of iterations, number of function evaluations and CPU time. The test functions can be downloaded from (Bongartz et al., 1995).

![Fig. 4: Perturbed Quadratic function in 3D](image)

![Fig. 5: Performance profile based on the number of iteration](image)
Fig. 6: Performance profile based on the function evaluation

Fig. 7: Performance profile based on CPU time
Table 2: A list of problem functions

| Function      | Dimension | No. function | CPU time | No. function | CPU time |
|---------------|-----------|--------------|----------|--------------|----------|
| AKIVA         | 2         | 8            | 0.02     | 10           | 0.02     |
| ALLINITI      | 4         | 9            | 0.02     | 12           | 0.02     |
| ARGELINA      | 200       | 1            | 0.02     | 1            | 0.02     |
| ARWEIRD       | 200       | 6            | 0.02     | 7            | 0.02     |
| BARD          | 3         | 12           | 0.02     | 16           | 0.02     |
| BEALE         | 2         | 11           | 0.02     | 15           | 0.02     |
| BIGGS6        | 6         | 24           | 0.02     | 27           | 0.02     |
| BOX3          | 3         | 10           | 0.02     | 11           | 0.02     |
| BRKMCC        | 2         | 5            | 0.02     | 5            | 0.02     |
| BROWNAL       | 200       | 9            | 0.02     | 9            | 0.02     |
| BROWNBS       | 2         | 10           | 0.02     | 13           | 0.02     |
| BROWNEN        | 4        | 16           | 0.02    | 16           | 0.02     |
| CHAINWOO      | 4000      | 352          | 0.8      | 318          | 0.86     |
| CHNROSNB      | 50        | 269          | 0.02    | 287          | 0.02     |
| CLIFF         | 2         | 10           | 0.02     | 18           | 0.02     |
| CUBE          | 2         | 17           | 0.02     | 32           | 0.02     |
| CURLY10       | 10000     | 52849        | 197     | 47808        | 173.7    |
| CURLY20       | 10000     | 79446        | 437     | 66587        | 383.94   |
| CURLY30       | 10000     | 81281        | 644     | 79030        | 639.63   |
| DECONVU       | 63        | 390          | 806     | 400          | 2.00E-02 |
| DENSCHNA      | 2         | 6            | 0.02    | 9            | 0.02     |
| DENSCHNB      | 2         | 6            | 0.02    | 7            | 0.02     |
| DENSCHNC      | 2         | 11           | 0.02    | 12           | 0.02     |
| DENSCHND      | 3         | 14           | 0.02    | 47           | 0.02     |
| DENSCHNE      | 3         | 12           | 0.02    | 18           | 0.02     |
| DENSCHNF      | 2         | 9            | 0.02    | 8            | 0.02     |
| DIXMAANA      | 3000      | 6            | 0.02    | 7            | 0.02     |
| DIXMAANB      | 3000      | 6            | 0.02    | 6            | 0.02     |
| DIXMAAND      | 3000      | 6            | 0.02    | 6            | 0.02     |
| DIXMAANE      | 3000      | 13           | 0.23    | 222          | 0.33     |
| DIXMAANF      | 3000      | 17           | 0.02    | 161          | 0.13     |
| DIXMAANH      | 3000      | 44           | 0.08    | 173          | 0.22     |
| DIXMAANJ      | 3000      | 415          | 0.52    | 327          | 0.36     |
| DIXON3DQ      | 10000     | 10000        | 19.12   | 10000        | 19.12    |
| DJTL          | 2         | 75           | 1163    | 82           | 917      |
| DQDRTIC       | 5000      | 5            | 0.02    | 5            | 0.02     |
| DQRTIC        | 5000      | 15           | 0.01    | 17           | 37       |
| EDENSC        | 200       | 26           | 0.08    | 26           | 52       |
| EG2           | 1000      | 3            | 0.02    | 5            | 11       |
| EGENALS       | 2550      | 9111         | 164     | 10083        | 178.36   |
| ENGVIAL1      | 5000      | 24           | 47      | 27           | 50       |
| ENGVIAL2      | 3000      | 136          | 0.02    | 26           | 61       |
| EXPF          | 2         | 9            | 0.02    | 13           | 29       |
| EXTROSNB      | 1000      | 23           | 71      | 3808         | 7759     |
| FLETCBV2      | 5000      | 1            | 1       | 1            | 1        |
| FLETCHCR      | 1000      | 252          | 497     | 152          | 290      |
| FMINSRF2      | 5625      | 23           | 83      | 346          | 693      |
| FMINSURF      | 5625      | 27           | 86      | 473          | 947      |
| GENHUMPS      | 5000      | 5740         | 15736   | 6475         | 12964    |
| GENROSE       | 500       | 1376         | 2820    | 1078         | 2167     |
| GROWTHLS      | 3         | 109          | 431     | 156          | 456      |
| GULF          | 3         | 33           | 95      | 37           | 84       |
| HAIRY         | 2         | 17           | 82      | 36           | 99       |
| HATFLDD       | 3         | 17           | 49      | 20           | 43       |
| HATFLEDE      | 3         | 13           | 37      | 30           | 72       |
| HATFLEFL      | 3         | 21           | 68      | 39           | 92       |
| HEART6LS      | 6         | 375          | 1137    | 684          | 1576     |
| HEART8LS      | 8         | 253          | 657     | 249          | 524      |
| HELIX         | 3         | 23           | 60      | 23           | 49       |
| HIELOW        | 2         | 13           | 30      | 14           | 30       |
| HILBERTA      | 3         | 2            | 5       | 2            | 5        |
| HILBERTB      | 10        | 4            | 9       | 4            | 9        |
Table 2: Continue

| Algorithm | Dimension | Iterations | CPU Time (s) | Memory (MB) |
|-----------|-----------|------------|-------------|-------------|
| HIMMELBB  | 2         | 4          | 18          | 0.02        |
| HIMMELBF | 4         | 23         | 59          | 0.02        |
| HIMMELDG | 2         | 7          | 22          | 0.02        |
| HIMMELBH | 2         | 5          | 13          | 0.02        |
| HUMPS     | 2         | 45         | 223         | 0.02        |
| JENSMIP   | 4         | 16         | 46          | 0.02        |
| KOWOSB    | 4         | 16         | 50          | 0.02        |
| LIARWHD   | 5000      | 19         | 50          | 0.05        |
| LOGHairy  | 2         | 26         | 196         | 0.02        |
| MANCINO   | 100       | 10         | 21          | 0.08        |
| MARATOSB  | 2         | 589        | 2885        | 0.02        |
| MEXHAT    | 2         | 14         | 59          | 0.02        |
| MOREBV    | 5000      | 25         | 102         | 0.13        |
| NCB20B    | 500       | 2044       | 2288        | 0.17        |
| NCB20     | 5010      | 303        | 747         | 4.17        |
| NONCXXU2  | 5000      | 43         | 81          | 0.17        |
| NONDIA    | 5000      | 7          | 25          | 0.02        |
| NONDQUAR  | 5000      | 66         | 216         | 0.17        |
| OSBORNA   | 5         | 82         | 230         | 0.02        |
| OSBORNEB  | 11        | 57         | 134         | 0.02        |
| OSCIPATH  | 10        | 295029     | 781729      | 2.24        |
| PALMER1C  | 8         | 12         | 27          | 0.02        |
| PALMER1D  | 7         | 10         | 24          | 0.02        |
| PALMER2C  | 8         | 11         | 21          | 0.02        |
| PALMER3C  | 8         | 11         | 21          | 0.02        |
| PALMER4C  | 8         | 11         | 21          | 0.02        |
| PALMER5C  | 6         | 6          | 13          | 0.02        |
| PALMER6C  | 8         | 11         | 24          | 0.02        |
| PALMER7C  | 8         | 11         | 20          | 0.02        |
| PALMER8C  | 8         | 11         | 19          | 0.02        |
| PENALTY1  | 1000      | 14         | 51          | 0.02        |
| PENALTY2  | 200       | 202        | 238         | 0.03        |
| PENALTY3  | 200       | 36         | 102         | 0.58        |
| POWELSG   | 5000      | 28         | 70          | 0.01        |
| POWER     | 10000     | 360        | 736         | 0.63        |
| QUARTC    | 5000      | 15         | 32          | 0.02        |
| ROSENB    | 2         | 28         | 84          | 0.02        |
| ROSENBR   | 2         | 7          | 21          | 0.02        |
| SCHMVETT  | 5000      | 41         | 72          | 0.19        |
| SENSORS   | 1000      | 27         | 73          | 0.39        |
| SINEVAL   | 2         | 46         | 181         | 0.02        |
| SINGAUD   | 5000      | 14         | 45          | 0.08        |
| SISR      | 2         | 5          | 19          | 0.02        |
| SNAIL     | 2         | 61         | 251         | 0.02        |
| SPARSE    | 5000      | 22306      | 22552       | 86          |
| SROSENB   | 5000      | 9          | 23          | 0.02        |
| STRATEC   | 10        | 170        | 419         | 6.22        |
| TESTQUAD  | 5000      | 1580       | 1587        | 1.34E+00    |
| TOINTGOR  | 50        | 118        | 215         | 0.02        |
| TOINTGSS  | 5000      | 4          | 9           | 0.02        |
| TOINTPSP  | 50        | 120        | 262         | 0.02        |
| TOINTQOR  | 50        | 29         | 36          | 0.02        |
| TQUARTIC  | 5000      | 11         | 41          | 0.03        |
| TRIDIA    | 5000      | 783        | 790         | 0.91        |
| VARDIM    | 200       | 9          | 20          | 0.02        |
| VAREIGVL  | 50        | 24         | 54          | 0.02        |
| VIBRBEAM  | 8         | 98         | 255         | 0.02        |
| WATSON    | 12        | 58         | 219         | 0.02        |
| WOODS     | 4000      | 24         | 61          | 0.03        |
| YFITU     | 3         | 68         | 208         | 0.02        |
| ZANGWIL2  | 2         | 1          | 3           | 0.02        |
Conclusion

In this study, we proposed a new modification of conjugate gradient method \( d^{alt} \) with restart condition is presented. In addition, new restart criterion proposed for FR CG method. Our numerical results had shown that the new coefficients are better than other conventional CG methods. Furthermore the efficiency of the FR method improved substantially when the new restart criterion is used.

Acknowledgements

The authors are grateful to the editor and the anonymous reviewers for their valuable comments and suggestions, which have substantially improved this paper. This research was partially supported by the Center of Research and Innovation Management, Universiti Malaysia Terengganu.

Author’s Contributions

Ahmad Alhawarat: Methodology and software.
Zabidin Salleh: Proof theory and review the paper with the software.
Ibtisam A. Masmali: Support software and review the main CG formula with its proof.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Al-Baali, M. (1985). Descent property and global convergence of the Fletcher-Reeves method with inexact line search. IMA Journal of Numerical Analysis, 5(1), 121-124. https://doi.org/10.1093/imanum/5.1.121
Alhawarat, A., Salleh, Z., Mamat, M., & Rivaie, M. (2017). An efficient modified Polak–Ribière–Polyak conjugate gradient method with global convergence properties. Optimization Methods and Software, 32(6), 1299-1312. https://doi.org/10.1080/10556788.2016.1266354
Andre, N. (2008). An unconstrained optimization test functions collection. Advance Modelling Optima, 10(1), 147-161.
Beale, E. M. L. (1972). A deviation of conjugate gradients. Numerical Methods for Nonlinear Optimization, 39-43.
Bongartz, I., Conn, A. R., Gould, N., & Toint, P. L., (1995). CUTE: Constrained and unconstrained testing environment. ACM Transactions on Mathematical Software (TOMS), 21(1), pp, 123-160. https://doi.org/10.1145/200979.201043
Dai, Y. H., & Liao, L. Z. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods. Applied Mathematics and Optimization, 43(1), 87-101. https://doi.org/10.1007/s002450010019
Dai, Y., & Yuan, Y. (1998). Convergence properties of Beale-Powell restart algorithm. Science in China Series A: Mathematics, 41(11), 1142-1150. https://doi.org/10.1007/BF02871976
Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2), 201-213. https://doi.org/10.1007/s101070100263
Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. The Computer Journal, 7(2), 149-154. https://doi.org/10.1093/comjnl/7.2.149
Gilbert, J. C., & Nocedal, J. (1992). Global convergence properties of conjugate gradient methods for optimization. SIAM Journal on optimization, 2(1), 21-42. https://doi.org/10.1137/0802003
Guanghui, L., Jiye, H., & Hongxia, Y. (1995). Global convergence of the Fletcher-Reeves algorithm with inexact linesearch. Applied Mathematics-A Journal of Chinese Universities, 10(1), 75-82. https://doi.org/10.1007/BF02663897
Hager, W. W., & Zhang, H. (2005). A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on optimization, 16(1), 170-192. https://doi.org/10.1137/030601880
Hager, W. W., & Zhang, H. (2013). The limited memory conjugate gradient method. SIAM Journal on Optimization, 23(4), 2150-2168. https://doi.org/10.1137/120898097
Hestenes, M. R., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems (Vol. 49, No. 1). Washington, DC: NBS. https://doi.org/10.6028/jres.049.044
Kaelo, P., Mtakulwa, P., & Thuto, M. V. (2020). A globally convergent hybrid conjugate gradient method with strong Wolfe conditions for unconstrained optimization. Mathematical Sciences, 14(1), 1-9. https://doi.org/10.1007/s40096-019-00310-y
Polak, E., & Ribiere, G. (1969). Note on the convergence of conjugate direction methods. ESAIM: Mathematical Modeling and Numerical Analysis-Mathematical Modeling and Numerical Analysis, 3 (R1), 35-43. https://doi.org/10.1051/m2an/196903R100351
Powell, M. J. (1984). Nonconvex minimization calculations and the conjugate gradient method. In Numerical analysis (pp. 122-141). Springer, Berlin, Heidelberg. https://doi.org/10.1007/BFb0099521
Powell, M. J. D. (1977). Restart procedures for the conjugate gradient method. Mathematical Programming, 12(1), 241-254. https://doi.org/10.1007/BF01593790
Wolfe, P. (1969). Convergence conditions for ascent methods. SIAM Review, 11(2), 226-235. https://doi.org/10.1137/1011036

Wolfe, P. (1971). Convergence conditions for ascent methods. II: Some corrections. SIAM Review, 13(2), 185-188. https://doi.org/10.1137/1013035

Zoutendijk, G. (1970). Nonlinear programming, computational methods. Integer and Nonlinear Programming, 37-86. https://ci.nii.ac.jp/naid/10030666308/