Research Article

An Efficient Modified AZPRP Conjugate Gradient Method for Large-Scale Unconstrained Optimization Problem

Ahmad Alhawarat,1,2 Thoi Trung Nguyen,1,3 Ramadan Sabra,4 and Zabidin Salleh5

1Division of Computational Mathematics and Engineering, Institute for Computational Science, Ton Duc Thang University, Ho Chi Minh City, Vietnam
2Faculty of Mathematics and Statistics, Ton Duc Thang University, Ho Chi Minh City, Vietnam
3Faculty of Civil Engineering, Ton Duc Thang University, Ho Chi Minh City, Vietnam
4Department of Mathematics, Faculty of Science, Jazan University, Jazan, Saudi Arabia
5Department of Mathematics, Faculty of Ocean Engineering Technology and Informatics, Universiti Malaysia Terengganu, Kuala Nerus 21030, Terengganu, Malaysia

Correspondence should be addressed to Zabidin Salleh; zabidin@umt.edu.my

Received 5 December 2020; Revised 30 January 2021; Accepted 19 March 2021; Published 26 April 2021

Academic Editor: Qingli Zhao

Copyright © 2021 Ahmad Alhawarat et al. 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.

To find a solution of unconstrained optimization problems, we normally use a conjugate gradient (CG) method since it does not cost memory or storage of second derivative like Newton’s method or Broyden–Fletcher–Goldfarb–Shanno (BFGS) method. Recently, a new modification of Polak and Ribiere method was proposed with new restart condition to give a so-call AZPRP method. In this paper, we propose a new modification of AZPRP CG method to solve large-scale unconstrained optimization problems based on a modification of restart condition. The new parameter satisfies the descent property and the global convergence analysis with the strong Wolfe-Powell line search. The numerical results prove that the new CG method is strongly aggressive compared with CG_Descent method. The comparisons are made under a set of more than 140 standard functions from the CUTEst library. The comparison includes number of iterations and CPU time.

1. Introduction

The conjugate gradient (CG) method aims to find a solution of optimization problems without constraint. Suppose that the following optimization problem is considered:

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

where $f: \mathbb{R}^n \rightarrow \mathbb{R}$ is continuous, the function is differentiable, and the gradient $\nabla f(x)$ is available. The iterative method is given by the following sequence:

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 1, 2, \ldots,$$

where $x_k$ is the starting point and $\alpha_k > 0$ is a step length. The search direction $d_k$ of the CG method is defined as follows:

$$d_k = \begin{cases} -\nabla f(x), & \text{if } k = 1, \\ -\nabla f(x) + \beta_k d_{k-1}, & \text{if } k \geq 2, \end{cases}$$

where $\nabla f(x) = g(x_k)$ and $\beta_k$ is a parameter.

To obtain the step length, we normally use the inexact line search, since the exact line search which is defined as follows,

$$f(x_k + \alpha_k d_k) = \min f(x_k + \alpha d_k), \quad \alpha > 0,$$

requires many iterations to obtain the step length. Normally, we use the strong version of Wolfe-Powell (SWP) [1, 2] line search which is given by

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k,$$

where $\nabla f(x_k + \alpha_k d_k)^T d_k \leq \sigma \|g_k\| d_k$, ($\delta$, $\sigma$ are parameters).
where $0 < \delta < \sigma < 1$.

The weak Wolfe-Powell (WWP) line search is defined by (5) and

$$
\nabla f(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k, \quad (7)
$$

where $\nabla f = g_k = g(x_k)$. The famous parameters of $\beta_k$ are the Hestenes-Stiefel (HS) [3], Fletcher-Reeves (FR) [4], and Polak-Rièire-Polyak (PRP) [5] formulas, which are given by

$$
\begin{align*}
\beta_k^{\text{HS}} &= \frac{g_k^T y_k}{d_k^T y_k}, \\
\beta_k^{\text{FR}} &= \frac{g_k^T g_k}{\|g_k\|^2}, \\
\beta_k^{\text{PRP}} &= \frac{g_k^T y_k}{\|g_{k-1}\|^2},
\end{align*}
$$

where $y_k = g_k - g_{k-1}$.

Powell [6] shows that there exists a nonconvex function such that the PRP method does not globally converge. Gilbert and Nocedal [7] show that if $\beta_k^{\text{PRP}+} = \max\{0, \beta_k^{\text{PRP}}\}$ with the WWP and the descent property is satisfied, then it is globally convergent.

Al-Baali [8] proved that the CG method with FR coefficient is convergent with SWP line search when $\sigma \leq 1/2$. Hager and Zhang [9, 10] presented a new CG parameter with descent property, i.e., $g_k^T d_k \leq -(7/8)\|g_k\|^2$. This formula is given as follows:

$$
\beta_k^{\text{HZ}} = \max\{\beta_k^N, \eta_k\}, \quad (9)
$$

where $\beta_k^N = (1/d_k^T y_k)(y_k - 2d_k(\|y_k\|^2/d_k^T y_k))^T g_k$; $\eta_k = -(1/\|d_k\|\min\{\eta, \|g_k\|\})$; and $\eta > 0$ is a constant. In the numerical experiments, they set $\eta = 0.01$ in (9). Al-Baali et al. [11] compared $\beta_k^{\text{HZ}}$ with a new three-term CG method (G3TCG).

Regarding the speed, memory requirements, number of iterations, function evaluations, gradient evaluations, and robustness to solve unconstrained optimization problems which have prompted the development of the CG method, the readers are advised to refer to references [10–15] for more information on these new formulas.

### 2. The New Formula and the Algorithm

Alhawarat et al. [15] presented the following simple formula:

$$
\beta_k^{\text{AZPRP}} = \begin{cases} 
\frac{\|g_k\|^2 - \mu_k |g_k^T g_k|}{\|g_{k-1}\|^2}, & \text{if } \|g_k\|^2 > \mu_k |g_k^T g_{k-1}|, \\
0, & \text{otherwise.}
\end{cases} \quad (10)
$$

Dai and Laio [12] presented the following formula:

$$
\beta_k^{\text{DL}+} = \max\{\beta_k^{\text{HS}}, 0\} - t \frac{g_k^T s_{k-1}}{d_k^T y_{k-1}}, \quad (11)
$$

where $s_{k-1} = x_k - x_{k-1}$ and $t \geq 0$.

The new formula is a modification of $\beta_k^{\text{AZPRP}}$ and $\beta_k^{\text{DL}+}$ is defined as follows:

$$
\begin{align*}
\beta_k^A &= \begin{cases} 
\frac{\|g_k\|^2 - \mu_k |g_k^T s_{k-1}|}{\|g_{k-1}\|^2}, & \text{if } \|g_k\|^2 > \mu_k |g_k^T s_{k-1}|, \\
-\frac{g_k^T s_{k-1}}{d_k^T y_{k-1}}, & \text{otherwise},
\end{cases} \\
\beta_k^A &\leq \frac{\|g_k\|^2 - \mu_k |g_k^T s_{k-1}|}{\|g_{k-1}\|^2} \leq \frac{\|g_k\|^2}{\|g_{k-1}\|^2} = \beta_k^{\text{FR}}, \quad (12)
\end{align*}
$$

where $\mu_k = (\|s_{k-1}\|/\|y_{k-1}\|)$ and $t > 0$.

We obtain the following relations (Algorithm 1):

$$
\beta_k^A \geq 0, \\
\beta_k^A \leq \frac{\|g_k\|^2 - \mu_k |g_k^T s_{k-1}|}{\|g_{k-1}\|^2} \leq \frac{\|g_k\|^2}{\|g_{k-1}\|^2} = \beta_k^{\text{FR}}. \quad (13)
$$

### 3. Convergence Analysis of Coefficient $\beta_k$ with CG Method

#### Assumption 1

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

$$
\|x\| \leq T, \quad \forall x \in \Psi. \quad (14)
$$

(B) In some neighbourhoods $N$ of $\Psi$, $f$ is continuous and the gradient is available and its gradient is Lipschitz continuous; that is, for all $x, y \in N$, there exists a constant $L > 0$ such that

$$
\|g(x) - g(y)\| \leq L \|x - y\|. \quad (15)
$$

This assumption shows that there exists a positive constant $B$ such that

$$
\|g(u)\| \leq B, \quad \forall u \in N. \quad (16)
$$

The descent condition

$$
g_k^T d_k \leq -\|g_k\|^2, \quad \forall k \geq 1. \quad (17)
$$

(17) plays an important role in the CG method. The sufficient descent condition proposed by Al-Baali [8] is a modification of (17) as follows:

$$
g_k^T d_k \leq -c \|g_k\|^2, \quad \forall k \geq 1, \quad (18)
$$

where $c \in (0, 1)$. Note that the general form of the sufficient descent condition is (18) with $c > 0$. 


3.1. Global Convergence for $\beta^k$ with the SWP Line Search. The following theorem demonstrates that $\beta^k$ ensures that the sufficient descent condition (21) is satisfied with the SWP line search.

The following theorem shows that $\beta^k$ satisfies the descent condition. The proof is similar to that presented in [8].

**Theorem 1.** Let $\{g_k\}$ and $\{d_k\}$ be generated using (2), (3), and $\beta^k = (\|g_k\|^2 - \mu_k g_k^T d_{k-1}/\|g_{k-1}\|^2)$, where $\mu_k$ is computed by the SWP line search (5) and (6). If $\sigma \in (0, 1/2)$, then the sufficient descent condition (18) holds.

Algorithm 1 shows the steps to obtain the solution of optimization problem using strong Wolfe-Powell line search.

Descent condition is (18) with $c > 0$.

**Proof.** By multiplying (1) by $g_k^T$, we obtain

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

Divide (19) by $\|g_k\|^2$; using

$$\|\nabla f(x_k + \alpha k d_k)\| \leq \|g_k d_k\|, \quad (20)$$

and (12), we obtain

$$-1 + \frac{\|g_{k-1} d_k\|^2}{\|g_{k-1}\|^2} \leq \frac{\|g_k d_k\|^2}{\|g_k\|^2} \leq -1 - \sigma \frac{\|g_{k-1} d_{k-1}\|^2}{\|g_{k-1}\|^2}. \quad (21)$$

From (3), we obtain $g_i^T d_i = -\|g_i\|^2$. Assume that it is true until $k-1$, i.e., $g_i^T d_i < 0$, for $i = 1, 2, \ldots, k-1$. Repeating the process for (21), we obtain

$$- \sum_{j=0}^{k-1} \sigma^j \frac{g_k^T d_k}{\|g_k\|^2} \leq -2 + \sum_{j=0}^{k-1} \sigma^j. \quad (22)$$

As

$$\sum_{j=0}^{k-1} (\sigma^j < \frac{1 - (\sigma)^k}{1 - \sigma}), \quad (23)$$

hence,

$$- \frac{1 - (\sigma)^k}{1 - \sigma} \leq \frac{g_k^T d_k}{\|g_k\|^2} \leq -2 + \frac{1 - (\sigma)^k}{1 - \sigma}. \quad (24)$$

and when $\sigma \leq (1/2)$, we obtain $1 - (\sigma)^k/1 - \sigma < 2$. Let $c = 2 - (1 - (\sigma)^k/1 - \sigma)$, then

$$c - 2 \leq \frac{g_k^T d_k}{\|g_k\|^2} \leq - c. \quad (25)$$

The proof is complete. □

**Theorem 2.** Let $\{g_k\}$ and $\{d_k\}$ be obtained by using (2), (3), and $\beta^k_{DL-HS} = -\mu_k g_k^T s_{k-1}/d_{k-1}^T y_{k-1}$ where $\mu_k$ is computed by SWP line search (5) and (6), then the descent condition holds.

**Proof.**

$$\beta^k_{DL-HS} = -\mu_k \frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}. \quad (26)$$

By multiplying (3) by $g_k^T$, and substituting $\beta^k$, we obtain

---

**Algorithm 1: Steps of CG method with new modification to obtain the stationary point of functions.**

- **Start**
  - Set a starting point $x_1$.
  - Set the initial search direction $d_1 = -g_1$. Let $k := 1$.
  - Yes: If a stopping criteria is satisfied, compute $d_k$ based on (2) with (10). Compute $a_k$ using (4) and (5). Set $k := k + 1$.
  - No: If a stopping criteria is satisfied, set $k := k + 1$.

- **End**
\[ g_k^T d_k = \| g_k \|^2 - \mu_k \frac{\alpha_{k-1} g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} \]
\[ = \| g_k \|^2 - \mu_k \frac{\alpha_{k-1} \| g_k^T d_{k-1} \|^2}{d_{k-1}^T y_{k-1}} < 0, \]
which completes the proof.

Zoutendijk [16] presented a useful lemma for global convergence property of the CG method. The condition is given as follows.

**Lemma 1.** Let Assumption 1 hold and consider any method in the form of (2) and (3), where \( \alpha_k \) is obtained by the WWP line search (6) and (7), in which the search direction is descent. Then, the following condition holds:
\[ \sum_{k=0}^{\infty} \frac{(g_k^T d_k)^2}{\| d_k \|^2} < \infty. \]  

**Theorem 3.** Suppose Assumption 1 holds. Consider any form of equations (2) and (3), with the new formula (12), in which \( \alpha_k \) is obtained from the SWP line search (5) and (6) with \( \sigma \leq 1/2 \). Then,
\[ \liminf_{k \to \infty} g_k = 0. \]

**Proof.** We will prove the theorem by contradiction. Assume that the conclusion is not true, then a constant \( \varepsilon > 0 \) exists such that
\[ \| g_k \| \geq \varepsilon, \quad \forall k \geq 1. \]  

Squaring both sides of equation (3), we obtain
\[ \| d_k \|^2 = \| g_k \|^2 - 2 \beta_k g_k^T d_{k-1} + \beta_k^2 \| d_{k-1} \|^2. \]

Divide (31) by \( \| g_k \|^4 \), we get
\[ \frac{\| d_k \|^2}{\| g_k \|^4} = \frac{1}{\| g_k \|^2} - \frac{2 \beta_k g_k^T d_{k-1}}{\| g_k \|^4} + \frac{\beta_k^2 \| d_{k-1} \|^2}{\| g_k \|^4}. \]  

Using (6), (12), and (32), we obtain
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \geq \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} \]  

Repeating the process for (33) and using the relationship \( 1/\| g_1 \| = (1/\| d_1 \|) \) yields
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \leq \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} + \frac{1}{\| g_k \|^2} \frac{1}{\| g_k \|^2} \]  

From (33), we obtain
\[ \frac{\| d_k \|^2}{\| g_k \|^4} \geq \frac{\varepsilon^2}{k (1 + 2 \sigma (2 - c))}, \]  

Therefore,
\[ \sum_{k=0}^{\infty} \frac{\| g_k \|^4}{\| d_k \|^2} = \infty. \]

This result contradicts (32), thus \( \liminf_{k \to \infty} \| g_k \| = 0 \).

The proof is complete.

**4. Numerical Results**

To investigate the effectiveness of the new parameter, several test problems in Table 1 from CUTEst [17] are chosen. We performed a comparison with the CG_Descent 5.3 based on the CPU time and the number of iterations. We employed the SWP line search with the line as mentioned in [1, 2] with \( \delta = 0.01 \) and \( \sigma = 0.1 \). The modified CG_Descent 6.8 where the memory (mem) equals zero is employed to obtain all results. The code can be downloaded from Hager web page http://users.clas.ufl.edu/hager/papers/Software/.

The CG_Descent 5.3 results are obtained by run CG_Descent 6.8 with memory which equals zero. The host computer is an AMD A4-7210 with RAM 4 GB. The results are shown in Figures 1 and 2 in which a performance measure introduced by Dolan and More [18] was employed. As shown in Figure 1, formula A strongly outperforms over CG_Descent in number of iterations. In Figure 2, we notice that the new CG formula A is strongly competitive with CG_Descent.

**4.1. Multimodal Function with Its Graph.** In this section, we present six-hump camel back function, which is a multimodal function to test the efficiency of the optimization algorithm. The function is defined as follows:
\[ f(x) = \left( 4 - 2.1 x_1^2 + \frac{x_1^4}{3} \right) x_1^2 + x_1 x_2 \]
\[ + (4 - 4 x_2^2) x_2^2. \]  

The number of variables \( n \) equals 2. This function has six local minima, with two of them being global. Thus, this function is a multimodal function usually used to test global minima. Global minima are \( x_1^* = (-0.0898, 0.7126) \) and \( x_2^* = (0.0898, -0.7126) \). The function value at \( f(x^*) = -1.0316 \). As its name describes, this function looks like the back of an upside down camel with six humps (see Figure 3 for a three-dimensional graph); for more information about two-dimensional functions, the reader can refer to [19].

Finally, note that CG method can be applied in image restoration problems and neural network and others. For more information, the reader can refer to [20, 21].
Table 1: The test functions.

| Function     | Dimension | Number of iterations | CPU time | Number of iterations | CPU time |
|--------------|-----------|----------------------|----------|----------------------|----------|
| AKIVA        | 2         | 10                   | 0.02     | 8                    | 0.02     |
| ALLINITU     | 4         | 12                   | 0.02     | 9                    | 0.02     |
| ARG LiNA     | 200       | 1                    | 0.02     | 1                    | 0.02     |
| ARG LinB     | 200       | 5                    | 0.02     | 6                    | 0.11     |
| ARW HEAD     | 5000      | 7                    | 0.02     | 6                    | 0.03     |
| BARD         | 3         | 16                   | 0.02     | 12                   | 0.02     |
| BD QRTIC     | 5000      | 136                  | 0.58     | 161                  | 0.75     |
| BEALE        | 2         | 15                   | 0.02     | 11                   | 0.02     |
| BIGG S6      | 6         | 27                   | 0.02     | 24                   | 0.02     |
| BOX3         | 3         | 11                   | 0.02     | 10                   | 0.02     |
| BOX          | 1000      | 8                    | 0.08     | 7                    | 0.08     |
| BRKMCC       | 2         | 5                    | 0.02     | 5                    | 0.02     |
| BROWNAL      | 200       | 9                    | 0.02     | 9                    | 0.02     |
| BROWN B5     | 2         | 13                   | 0.02     | 10                   | 0.02     |
| BROWN DEN    | 4         | 16                   | 0.02     | 16                   | 0.02     |
| BRO YD N7D   | 5000      | 1411                 | 5.47     | 64                   | 0.37     |
| BRY BND      | 5000      | 85                   | 0.38     | 39                   | 0.22     |
| CHAIN WOO    | 4000      | 318                  | 0.866    | 379                  | 1.08     |
| CHN RO SNB   | 50        | 287                  | 0.02     | 340                  | 0.02     |
| CLIFF        | 2         | 18                   | 0.02     | 10                   | 0.02     |
| COSINE       | 10000     | 11                   | 0.19     | 14                   | 0.26     |
| CRA GGL VY   | 5000      | 103                  | 0.45     | 104                  | 0.48     |
| CUBE         | 2         | 32                   | 0.02     | 17                   | 0.02     |
| CUR L Y10    | 10000     | 47808                | 173.7    | 42454                | 145.16   |
| CUR L Y20    | 10000     | 66587                | 383.94   | 67279                | 366.03   |
| CUR L Y30    | 10000     | 79030                | 639.63   | 74375                | 509.59   |
| DECON VU     | 63        | 400                  | 2.00E-02 | 227                  | 0.02     |
| DENSCHNA     | 2         | 9                    | 0.02     | 6                    | 0.02     |
| DENSCH NB    | 2         | 7                    | 0.02     | 6                    | 0.02     |
| DENSCH NC    | 2         | 12                   | 0.02     | 11                   | 0.02     |
| DENSCH ND    | 3         | 47                   | 0.02     | 14                   | 0.02     |
| DENSCH NE    | 3         | 18                   | 0.02     | 12                   | 0.02     |
| DENSCH NF    | 2         | 8                    | 0.02     | 9                    | 0.02     |
| DIX MA N A   | 3000      | 7                    | 0.02     | 6                    | 0.02     |
| DIX MA N B   | 3000      | 6                    | 0.02     | 6                    | 0.02     |
| DIX MA N C   | 3000      | 6                    | 0.02     | 6                    | 0.02     |
| DIX MA N D   | 3000      | 7                    | 0.02     | 8                    | 0.02     |
| DIX MA AN E  | 3000      | 222                  | 0.33     | 218                  | 0.33     |
| DIX MA AN F  | 3000      | 161                  | 0.13     | 116                  | 0.09     |
| DIX MA AN G  | 3000      | 157                  | 0.12     | 173                  | 0.14     |
| DIX MA AN H  | 3000      | 173                  | 0.22     | 190                  | 0.2      |
| DIX MA N I   | 3000      | 3856                 | 4.25     | 3160                 | 3.34     |
| DIX MA N J   | 3000      | 327                  | 0.36     | 360                  | 0.39     |
| DIX MA NK    | 3000      | 283                  | 0.28     | 416                  | 0.36     |
| DIX MA NL    | 3000      | 237                  | 0.2      | 399                  | 0.36     |
| DIX O N3 D Q | 10000     | 10000                | 19.12    | 10000                | 19.12    |
| DJ TL        | 2         | 82                   | 0.02     | 75                   | 0.02     |
| DQ DRTIC     | 5000      | 5                    | 0.02     | 5                    | 0.02     |
| DQ RTIC      | 5000      | 17                   | 0.03     | 15                   | 0.03     |
| EDEN SCH     | 2000      | 26                   | 0.03     | 32                   | 0.05     |
| E G2         | 1000      | 5                    | 0.02     | 3                    | 0.02     |
| Eigen ALS   | 2550      | 10083                | 178.36   | 7247                 | 133.4    |
| Eigen BL S  | 2550      | 15301                | 237      | 18846                | 290.3    |
| Eigen C L S | 2652      | 10136                | 174.19   | 11152                | 186.86   |
| ENG VA L1    | 5000      | 27                   | 0.06     | 23                   | 0.12     |
| ENG VA L2    | 3         | 26                   | 0.02     | 26                   | 0.02     |
| ERR INROS    | 50        | 380                  | 0.02     | 95504                | 2.36     |
| Function       | Dimension | Number of iterations | CPU time | Number of iterations | CPU time |
|----------------|-----------|----------------------|----------|----------------------|----------|
| EXPFIT         | 2         | 13                   | 0.02     | 9                    | 0.02     |
| EXTROSNB       | 1000      | 3808                 | 1.25     | 2370                 | 0.87     |
| FLETCBV2       | 5000      | 1                    | 0.02     | 1                    | 0.02     |
| FLETCHCR       | 1000      | 152                  | 0.05     | 84                   | 0.05     |
| FMINSRF2       | 5625      | 346                  | 1.09E+00 | 485                  | 1.4      |
| FMINSURF       | 5625      | 473                  | 1.51     | 542                  | 1.64     |
| FREUROTH       | 5000      | 25                   | 0.11     | 29                   | 0.19     |
| GENROSE        | 500       | 1078                 | 0.17     | 2098                 | 0.45     |
| GROWTHLS       | 3         | 156                  | 0.02     | 109                  | 0.02     |
| GULF           | 3         | 37                   | 0.02     | 33                   | 0.02     |
| HAIRY          | 2         | 36                   | 0.02     | 17                   | 0.02     |
| HATFLDD        | 3         | 20                   | 0.02     | 17                   | 0.02     |
| HATFLDE        | 3         | 30                   | 0.02     | 13                   | 0.02     |
| HATFLDFL       | 3         | 39                   | 0.02     | 21                   | 0.02     |
| HEART6LS       | 6         | 684                  | 0.02     | 375                  | 0.02     |
| HEART8LS       | 8         | 249                  | 0.02     | 253                  | 0.02     |
| HELIX          | 3         | 23                   | 0.02     | 23                   | 0.02     |
| HIELOW         | 3         | 14                   | 0.02     | 13                   | 0.05     |
| HILBERTA       | 2         | 2                    | 0.02     | 2                    | 0.02     |
| HILBERTB       | 10        | 4                    | 0.02     | 4                    | 0.02     |
| HIMMELBB       | 2         | 10                   | 0.02     | 4                    | 0.02     |
| HIMMELBF       | 4         | 26                   | 0.02     | 23                   | 0.02     |
| HIMMELBG       | 2         | 8                    | 0.02     | 7                    | 0.02     |
| HIMMELBH       | 2         | 7                    | 0.02     | 5                    | 0.02     |
| HUMPS          | 2         | 52                   | 0.02     | 45                   | 0.02     |
| JENSMMP        | 2         | 15                   | 0.02     | 12                   | 0.02     |
| JIMACK         | 35449     | 8314                 | 1182.25  | 7297                 | 1030.3   |
| KOWOSB         | 4         | 17                   | 0.02     | 16                   | 0        |
| LIARWHD        | 5000      | 21                   | 0.03     | 15                   | 0.05     |
| LOGHAIRY       | 2         | 27                   | 0.02     | 26                   | 0.02     |
| MANCINO        | 100       | 11                   | 0.08     | 11                   | 0.08     |
| MARATOSB       | 2         | 1145                 | 0.02     | 589                  | 0.02     |
| MEXHAT         | 2         | 20                   | 0.02     | 14                   | 0.02     |
| MOREBV         | 5000      | 161                  | 0.41     | 161                  | 0.38     |
| MSQRTALS       | 1024      | 2905                 | 8.64     | 2788                 | 9.08     |
| MSQRTBLS       | 1024      | 2280                 | 6.91     | 2181                 | 6.84     |
| NCB20B         | 500       | 2035                 | 46.36    | 4181                 | 70.16    |
| NCB20          | 5010      | 879                  | 11.83    | 959                  | 13       |
| NONCVXU2       | 5000      | 6610                 | 15.89    | 6379                 | 15.92    |
| NONDIA         | 5000      | 7                    | 0.03     | 7                    | 0.03     |
| NONDQUAR       | 5000      | 1942                 | 2.45     | 3058                 | 3.88     |
| OSBORNEA       | 5         | 94                   | 0.02     | 82                   | 0.02     |
| OSBORNEB       | 11        | 62                   | 0.02     | 57                   | 0.02     |
| PALMER1C       | 8         | 11                   | 0.02     | 12                   | 0.02     |
| PALMER1D       | 7         | 11                   | 0.02     | 10                   | 0.02     |
| PALMER2C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PALMER3C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PALMER4C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PALMER5C       | 6         | 6                    | 0.02     | 6                    | 0.02     |
| PALMER6C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PALMER7C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PALMER8C       | 8         | 11                   | 0.02     | 11                   | 0.02     |
| PARKCH         | 15        | 672                  | 29.45    | 823                  | 39.39    |
| PENALTY1       | 1000      | 28                   | 0.02     | 41                   | 0.02     |
| PENALTY2       | 200       | 191                  | 0.05     | 200                  | 0.03     |
| PENALTY3       | 200       | 99                   | 1.78     | 88                   | 1.98     |
| Function    | Dimension | Number of iterations | CPU time | Number of iterations | CPU time |
|-------------|-----------|----------------------|----------|----------------------|----------|
| POWELSG     | 5000      | 26                   | 0.02     | 27                   | 0.05     |
| POWER       | 10000     | 372                  | 0.76     | 543                  | 1.2      |
| QUARTC      | 5000      | 17                   | 0.03     | 15                   | 0.02     |
| ROSENBR     | 2         | 34                   | 0.02     | 28                   | 0.02     |
| S308        | 2         | 8                    | 0.02     | 7                    | 0.02     |
| SCHMVETT    | 5000      | 43                   | 0.23     | 40                   | 0.27     |
| SENSORS     | 100       | 21                   | 0.25     | 50                   | 0.8      |
| SINEVAL     | 2         | 64                   | 0.02     | 46                   | 0.02     |
| SINQUAD     | 5000      | 14                   | 0.09     | 15                   | 0.09     |
| SISER       | 2         | 6                    | 0.02     | 5                    | 0.02     |
| SNAIL       | 2         | 100                  | 0.02     | 61                   | 0.02     |
| SPARSINE    | 5000      | 18,358               | 73       | 21,328               | 83       |
| SPARSQUR    | 10000     | 28                   | 0.31     | 35                   | 0.98     |
| SPMSRTLS    | 4999      | 203                  | 0.59     | 219                  | 0.61     |
| SROSENBR    | 5000      | 11                   | 0.02     | 9                    | 0.03     |
| STRATEC     | 10        | 462                  | 19.98    | 170                  | 6.23     |
| TESTQUAD    | 5000      | 1577                 | 1.52E+00 | 1573                 | 1.42     |
| TOINTGOR    | 50        | 135                  | 0.02     | 120                  | 0.02     |
| TOINTGSS    | 5000      | 4                    | 0.02     | 5                    | 0.02     |
| TOINTPSP    | 50        | 143                  | 0.02     | 157                  | 0.02     |
| TOINTQOR    | 50        | 29                   | 0.02     | 29                   | 0.02     |
| TQUARTIC    | 5000      | 14                   | 0.03     | 11                   | 0.03     |
| TRIDIA      | 5000      | 782                  | 0.84     | 783                  | 1.11     |
| VAREIGVL    |           | 23                   | 0.02     | 24                   | 0.02     |
| VIBRBEAM    | 50        | 138                  | 0.02     | 98                   | 0.02     |
| WATSON      | 8         | 49                   | 0.02     | 61                   | 0.02     |
| WOODS       | 12        | 22                   | 0.06     | 22                   | 0.03     |
| YFITU       | 4000      | 84                   | 0.02     | 68                   | 0.02     |
| ZANGWIL2    | 3         | 1                    | 0.02     | 1                    | 0.02     |

Figure 1: Performance measure based on the number of iterations.
5. Conclusions

In this study, a modified version of the CG algorithm (A) is suggested and its performance is investigated. The modified formula is restarted based on the value of the Lipchitz constant. The global convergence is established by using SWP line search. Our numerical results show that the new coefficient produces efficient and competitive results compared with other methods, such as CG_Descent 5.3. In the future, an application of the new version of CG method will be combined with feed-forward neural network (back-propagation (BP) algorithm) to improve the training process and produce fast training multilayer algorithm. This will help in reducing time needed to train neural network when the training samples are massive.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank Universiti Malaysia Terengganu for supporting this work.

References

[1] P. Wolfe, “Convergence conditions for ascent methods,” SIAM Review, vol. 11, no. 2, pp. 226–235, 1969.
[2] P. Wolfe, “Convergence conditions for ascent methods. II: some corrections,” SIAM Review, vol. 13, no. 2, pp. 185–188, 1971.
[3] E. Stiefel, “Methods of conjugate gradients for solving linear systems,” Journal of Research of the National Bureau of Standards, vol. 49, pp. 409–435, 1952.
[4] R. Fletcher and C. M. Reeves, “Function minimization by conjugate gradients,” *The Computer Journal*, vol. 7, no. 2, pp. 149–154, 1964.

[5] E. Polak and G. Ribiere, “Note sur la convergence de méthodes de directions conjuguées,” *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, vol. 3, no. R1, pp. 35–43, 1969.

[6] M. J. Powell, “Nonconvex minimization calculations and the conjugate gradient method,” in *Numerical Analysis*, pp. 122–141, Springer, Berlin, Heidelberg, 1984.

[7] J. C. Gilbert and J. Nocedal, “Global convergence properties of conjugate gradient methods for optimization,” *SIAM Journal on Optimization*, vol. 2, no. 1, pp. 21–42, 1992.

[8] M. Al-Baali, “Descent property and global convergence of the Fletcher-Reeves method with inexact line search,” *IMA Journal of Numerical Analysis*, vol. 5, no. 1, pp. 121–124, 1985.

[9] W. W. Hager and H. Zhang, “A new conjugate gradient method with guaranteed descent and an efficient line search,” *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2005.

[10] W. W. Hager and H. Zhang, “The limited memory conjugate gradient method,” *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2150–2168, 2013.

[11] M. Al-Baali, Y. Narushima, and H. Yabe, “A family of three-term conjugate gradient methods with sufficient descent property for unconstrained optimization,” *Computational Optimization and Applications*, vol. 60, no. 1, pp. 89–110, 2015.

[12] Y.-H. Dai and L. Z. Liao, “New conjugacy conditions and related nonlinear conjugate gradient methods,” *Applied Mathematics and Optimization*, vol. 43, no. 1, pp. 87–101, 2001.

[13] M. Al-Baali, E. Spedicato, and F. Maggioni, “Broyden’s quasi-Newton methods for a nonlinear system of equations and unconstrained optimization: a review and open problems,” *Optimization Methods and Software*, vol. 29, no. 5, pp. 937–954, 2014.

[14] S. Babaie-Kafaki and R. Ghanbari, “A descent family of Dai-Liao conjugate gradient methods,” *Optimization Methods and Software*, vol. 29, no. 3, pp. 583–591, 2014.

[15] A. Alhawarat, Z. Salleh, M. Mamat, and M. Rivaie, “An efficient modified Polak-Ribière-Polyak conjugate gradient method with global convergence properties,” *Optimization Methods and Software*, vol. 32, no. 6, pp. 1299–1312, 2017.

[16] G. Zoutendijk, *Nonlinear Programming, Computational Methods*, pp. 37–86, Integer and Nonlinear Programming, North Holland, Amsterdam, 1970.

[17] I. Bongartz, A. R. Conn, N. Gould, and P. L. Toint, “CUTE,” *ACM Transactions on Mathematical Software*, vol. 21, no. 1, pp. 123–160, 1995.

[18] E. D. Dolan and J. J. Moré, “Benchmarking optimization software with performance profiles,” *Mathematical Programming*, vol. 91, no. 2, pp. 201–215, 2002.

[19] A. Alhawarat and Z. Salleh, “Modification of nonlinear conjugate gradient method with weak Wolfe-Powell line search,” *Abstract and Applied Analysis*, vol. 2017, Article ID 7238134, 6 pages, 2017.

[20] G. Yuan, T. Li, and W. Hu, “A conjugate gradient algorithm and its application in large-scale optimization problems and image restoration,” *Journal of Inequalities and Applications*, vol. 2019, no. 1, pp. 1–25, 2019.

[21] G. Yuan, J. Lu, and Z. Wang, “The modified PRP conjugate gradient algorithm under a non-descent line search and its application in the Muskingum model and image restoration problems,” *Soft Computing*, vol. 25, no. 8, pp. 5867–5879, 2021.