AIMS Mathematics, 6(10): 10742–10764.
DOI: 10.3934/math.2021624
Received: 04 May 2021
Accepted: 14 July 2021
Published: 26 July 2021

http://www.aimspress.com/journal/Math

Research article

A decent three term conjugate gradient method with global convergence properties for large scale unconstrained optimization problems

Ibtisam A. Masmali[1], Zabidin Salleh[2,*] and Ahmad Alhawarat[3]

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

* **Correspondence:** Email: zabidin@umt.edu.my.

Abstract: The conjugate gradient (CG) method is a method to solve unconstrained optimization problems. Moreover CG method can be applied in medical science, industry, neural network, and many others. In this paper a new three term CG method is proposed. The new CG formula is constructed based on DL and WYL CG formulas to be non-negative and inherits the properties of HS formula. The new modification satisfies the convergence properties and the sufficient descent property. The numerical results show that the new modification is more efficient than DL, WYL, and CG-Descent formulas. We use more than 200 functions from CUTEst library to compare the results between these methods in term of number of iterations, function evaluations, gradient evaluations, and CPU time.

Keywords: conjugate gradient method; inexact line search; global convergence
AMS Subject Classifications: 49M37, 65K05, 90C3

1. Introduction

The conjugate gradient (CG) method is a method to solve large scale unconstrained optimization problems, we consider the following problem

$$\min f(x), x \in \mathbb{R}^n, \tag{1}$$

where $f:\mathbb{R}^n \to \mathbb{R}$ is a continuous and differentiable function and the gradient is available. The CG method generates a sequence x_k as follows:

$$x_{k+1} = x_k + \alpha_k d_k, k = 1,2,..., \tag{2}$$

where x_k is the current point (iteration) and $\alpha_k > 0$ is a steplength. The search direction d_k of the CG method is defined as follows:

$$d_k = \begin{cases} -g_k, & \text{if } k = 1, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 2, \end{cases} \tag{3}$$

where $g_k = g(x_k) = \nabla f$ and β_k is known as the CG formula.

To compute the steplength normally we use the strong Wolfe-Powell (SWP) [1,2] line search is defined as follows:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{4}$$

and

$$| g(x_k + \alpha_k d_k)^T d_k | \leq \sigma \left| g_k^T d_k \right|, \tag{5}$$

where $0 < \delta < \sigma < 1$.

The SWP line search is a strong version of the weak Wolfe-Powell (WWP) line search; the latter is given by (4) and

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k \tag{6}$$

The most famous classical formulas of CG methods Hestenses-Stiefel (HS) [3], Polak-Ribiere-Polyak (PRP) [4], Liu and Storey (LS) [5], Fletcher-Reeves (FR) [6], Fletcher (CD) [7], Dai and Yuan (DY) [8], are as follows:

$$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \quad \beta_k^{PRP} = \frac{g_k^T y_{k-1}}{\|g_{k-1}\|^2}, \quad \beta_k^{LS} = -\frac{g_k^T y_{k-1}}{d_{k-1}^T g_{k-1}}$$

$$\beta_k^{FR} = \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \quad \beta_k^{CD} = -\frac{\|g_k\|^2}{d_{k-1}^T g_{k-1}}, \quad \beta_k^{DY} = \frac{\|g_k\|^2}{d_{k-1}^T g_{k-1}},$$

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

Polak and Ribière [4] proven that PRP method with exact line search is globally converge. In the other hand Powell [9] proposed an example show that there exists a function does not global convergence even if the exact line search is employed with PRP formula. Powell suggests using non-negative value of PRP method. Gilbert and Nocedal [10] proved that if $\beta_k^{PRP+} = \max\{0, \beta_k^{PRP}\}$

with the WWP line search is employed and the sufficient descent condition (See Eq (13)) is satisfied, and then β_k^{PRP+} is globally convergent.

Zoutendijk [11] show the global convergence of FR formula with CG method and the exact line search. Al-Baali [12] show that the CG method with the FR coefficient is globally convergent when $\sigma \le 1/2$, and SWP is employed.

Dai and Liao [13] proposed the following conjugcy condition

$$d_k^T y_{k-1} = -t g_k^T s_{k-1}, \tag{7}$$

where $s_{k-1} = x_k - x_{k-1}$, and $t \ge 0$. In the case of $t = 0$, Eq (8) becomes the classical conjugcy condition. By using (2) and (7), [10] proposed the following CG formula

$$\beta_k^{DL} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} = \beta_k^{HS} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}. \tag{8}$$

However, β_k^{DL} inherits the same problem as β_k^{PRP} and β_k^{HS} i.e., β_k^{DL} is not non-negative in general. Thus [10] replaced Eq (8) by

$$\beta_k^{DL+} = \max\{\beta_k^{HS}, 0\} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}.$$

Hager and Zhang [14,15] presented a modified CG parameter that satisfies the descent property for any inexact line search with $g_k^T d_k \le -(7/8)\|g_k\|^2$. This new version of CG method is globally convergent whenever the line search satisfies the (WP) line search. This formula is given as follows:

$$\beta_k^{HZ} = \max\{\beta_k^N, \eta_k\} \tag{9}$$

where $\beta_k^N = \frac{1}{d_k^T y_k}(y_k - 2d_k\frac{\|y_k\|^2}{d_k^T y_k})^T g_k$, $\eta_k = -\frac{1}{\|d_k\|\min\{\eta, \|g_k\|\}}$, and $\eta > 0$ is a constant. Notes that if $t = 2\frac{\|y_k\|^2}{s_k^T y_k}$ then $\beta_k^N = \beta_k^{DY}$.

In 2006, Wei et al. [16]. gave a new positive CG method, which is quite similar to original PRP method which has a global convergence under exact and inexact line search that is,

$$\beta_k^{WYL} = \frac{g_k^T(g_k - \frac{\|g_k\|}{\|g_{k-1}\|}g_{k-1})}{\|g_{k-1}\|^2},$$

where $y_{k-1} = g_k - g_{k-1}$. In 2016, Alhawarat et al. [17] presented the following formula

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

where $\|\cdot\|$ represents the Euclidean norm. And μ_k is defined as follows:

$$\mu_k = \frac{\|x_k - x_{k-1}\|}{\|y_k\|}.$$

Kaelo et al. [18] proposed the following CG formula

$$\beta_k^{PKT} = \begin{cases} \dfrac{\|g_k\|^2 - g_k^T g_{k-1}}{max\{d_{k-1}^T y_{k-1}, -g_{k-1}^T d_{k-1}\}}, \text{If } 0 < g_k^T g_{k-1} < \|g_k\|^2 \\ \dfrac{\|g_k\|^2}{max\{d_{k-1}^T y_{k-1}, -g_{k-1}^T d_{k-1}\}}, \text{otherwise} \end{cases}$$

Yao et al. [19] proposed three terms of CG with a new choice of t as follows:

$$d_{k+1} = -g_{k+1} + \left(\frac{g_k^T y_k - t_k g_{k+1}^T s_k}{y_k^T d_k}\right) d_k + \frac{g_{k+1}^T d_k}{y_k^T d_k} y_k$$

Based on the SWP line search, Yao et al. [19] selected t_k to satisfy the descent condition as follows:

$$t_k > \frac{\|y_k\|^2}{y_k^T s_k}.$$

Yao et al. [19] also proposed a theorem stating that if t_k is close to $\frac{\|y_k\|^2}{y_k^T s_k}$, then the search direction results in a zigzag search path. Therefore, they selected the following choice for t_k:

$$t_k = 1 + 2\frac{\|y_k\|^2}{y_k^T s_k}.$$

For more about CG method and its application, the reader can refer to the following references [20–24].

2. The new formula and the algorithm

Since β_k^{DL} method in (8) has negative values for some times similar to β_k^{HS} and β_k^{PRP}, we construct new method depend on β_k^{WYL} and β_k^{DL} to be nonnegative and inherits the advantages of β_k^{HS}, β_k^{WYL}, and β_k^{DL}. The new method constructed as follows:

$$\beta_k^{DL-WYL} = \frac{g_k^T\left(g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1}\right)}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \tag{10}$$

Algorithm 1 shows that the steps to find the optimal solution of optimization function.

Algorithm 1. The steps of CG methods with Eq (10) to obtain the optimum method.

Step 1: Provide a starting point x_1. 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 (10).
Step 4: Compute α_k using (4) and (5).
Step 5: Update x_{k+1} based on (1).
Step 6: Set $k := k + 1$ and go to Step 2.

3. Global convergence analysis of the CG algorithm with the coefficient β_k^{DL-WYL}

To establish the convergence properties of the new formula, the following assumption is required.

Assumption 1. A. The level set $\Omega = \{x | f(x) \le f(x_1)\}$ is bounded, that is, a positive constant τ exists such that

$$\|x\| \le \tau, \quad \forall x \in \Omega.$$

B. In some neighbourhood Q of Ω, f is continuously differentiable, and its gradient is Lipschitz continuous; that is, for all $x, y \in Q$, there exists a constant $L > 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\| \tag{11}$$

This assumption implies that there exists a positive constant *B* such that

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

The descent condition (downhill condition)

$$g_k^T d_k < 0, \quad \forall k \ge 1, \tag{12}$$

is useful in the study of CG method and serves important rule in the proof of global convergence analysis. Abaali [12] modified (12) to the following form and used it to prove the FR method

$$g_k^T d_k \le -c\|g_k\|^2, \quad \forall k \ge 1 \tag{13}$$

where $c \in (0,1)$. Equation (14) is the sufficient descent condition. Note that the general form of the sufficient descent condition is (14) with $c > 0$. Moreover, using (13) is better than (12) since we can control the quantity of $g_k^T d_k$ by using $\|g_k\|^2$.

3.1. Global convergence for β_k^{DL-WYL} with the SWP line search

The following theorem demonstrates that β_k^{DL-WYL} ensures that the sufficient descent condition (13) is satisfied with the SWP line search.

Theorem 3.1. Let the sequences $\{g_k\}$ and $\{d_k\}$ are generated using (1), (2), and (10), where α_k is computed by the SWP line search (4) and (5). If $\sigma \in (0, \frac{1}{4})$, then the sufficient descent condition (14) holds.

Proof. By Multiplying (2) by g_k^T, we obtain

$$g_k^T d_k = g_k^T(-g_k^T + \beta_k d_{k-1}) = -\|g_k\|^2 + \beta_k g_k^T d_{k-1} \tag{14}$$

Substitute β_k^{DL-WYL} instead of β_k

$$g_k^T d_k = g_k^T(-g_k^T + \beta_k d_{k-1}) = -\|g_k\|^2 + \left(\frac{g_k^T(g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_{k-1})}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right) g_k^T d_{k-1} \tag{15}$$

Then we have the following two cases:

Case 1.

$$g_k^T g_{k-1} > 0.$$

Then

$$g_k^T d_k = -\|g_k\|^2 + \left(\frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \right) g_k^T d_{k-1}$$

$$g_k^T d_k = -\|g_k\|^2 + \frac{\|g_k\|^2}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1}$$

$$g_k^T d_k \le -\|g_k\|^2 - \frac{\|g_k\|^2}{(\sigma-1)g_{k-1}^T d_{k-1}} \sigma\, g_{k-1}^T d_{k-1} - t\frac{\alpha_k \|g_k^T d_{k-1}\|^2}{d_{k-1}^T y_{k-1}}$$

$$g_k^T d_k \le -\|g_k\|^2 - \frac{\|g_k\|^2}{(\sigma-1)g_{k-1}^T d_{k-1}} \sigma\, g_{k-1}^T d_{k-1} - t\frac{\alpha_k \|g_k^T d_{k-1}\|^2}{d_{k-1}^T y_{k-1}}.$$

Since

$$d_{k-1}^T y_{k-1} > 0$$

$$g_k^T d_k \le -\|g_k\|^2 - \sigma\frac{\|g_k\|^2}{(\sigma-1)}.$$

Divide both sides by $\|g_k\|^2$

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

Let $c = (1+\frac{\sigma}{\sigma-1})$, thus we obtain the result

$$g_k^T d_k \leq -c\|g_k\|^2.$$

Case 2.

$$g_k^T g_{k-1} < 0.$$

Then

$$g_k^T d_k \leq -\|g_k\|^2 + \left(\frac{2\|g_k\|^2}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}})\right) g_k^T d_{k-1}$$

$$g_k^T d_k = -\|g_k\|^2 + \frac{2\|g_k\|^2}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} g_k^T d_{k-1}$$

$$g_k^T d_k \leq -\|g_k\|^2 - \frac{2\|g_k\|^2}{(\sigma-1)g_{k-1}^T d_{k-1}} \sigma\, g_{k-1}^T d_{k-1} - t\frac{\alpha_k \|g_k^T d_{k-1}\|^2}{d_{k-1}^T y_{k-1}}$$

$$g_k^T d_k \leq -\|g_k\|^2 - \frac{2\|g_k\|^2}{(\sigma-1)g_{k-1}^T d_{k-1}} \sigma\, g_{k-1}^T d_{k-1} - t\frac{\alpha_k \|g_k^T d_{k-1}\|^2}{d_{k-1}^T y_{k-1}}.$$

Since

$$d_{k-1}^T y_{k-1} > 0$$

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

Divide both sides by $\|g_k\|^2$

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

Let $c = (1+\frac{2\sigma}{\sigma-1})$, then if $\sigma \leq \frac{1}{4}$ we obtain

$$g_k^T d_k \leq -c\|g_k\|^2.$$

The following lemma, which is referred to as the Zoutendijk condition [11], is useful for analysing the global convergence property of the CG method.

Lemma 3.1. Consider CG method with SWP line search and $g_k^T d_{k-1} < 0$ then the CG formula β_k^{DL-WYL} in non-negative.

Proof. By using SWP line search we obtain that

$$d_{k-1}^T y_{k-1} \geq 0.$$

By using Theorem 1 we have

$$-t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} > 0.$$

Since

$$\frac{g_k^T g_k - \frac{\|g_k\|}{\|g_{k-1}\|} g_k^T g_{k-1}}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \geq \frac{g_k^T g_k - \frac{\|g_k\|}{\|g_{k-1}\|}\|g_k\|\|g_{k-1}\|}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}} \geq 0.$$

Thus we obtain the result.

Lemma 3.2. Let Assumption 1 be holds. Consider any CG method in the form (1), (2), and α_k satisfies the WWP line search (5) and (6), in which the search direction is descent. Then, the following condition holds:

$$\sum_{k=0}^{\infty}\frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty \tag{16}$$

3.2. *Global convergence for β_k^{HS*} with the SWP line search*

The following property, which is referred to as Property*, was presented by Gilbert and Nocedal in [10]. This property is useful to obtain the global convergence properties of CG methods related to PRP or HS family. The property is given as follows:

Property* Consider a method of the form (1) and (2). Assume that

$$0 < \gamma \leq \|g_k\| \leq \bar{\gamma} \tag{17}$$

for all $k \geq 1$. Then, the CG method has Property* if there exist constants $b > 1$ and $\lambda > 0$ such that for all k whereby for $|\beta_k| \leq b$ and $\|s_k\| \leq \lambda$, we obtain $|\beta_k| \leq \frac{1}{2b}$.

Lemma 3.3. Consider the CG method of the form (1), and (2) with the new formula β_k^{DL-WYL}. If Eq (17) holds true then β_k^{DL-WYL} has Property*.

Proof. Set $b = \frac{2\bar{\gamma}^2 + t\bar{\gamma}B}{c(1-\sigma)\gamma^2} \geq 1$, and $\lambda = \frac{c(1-\sigma)\gamma^2}{L(L+1)\bar{\gamma}}$. Using (10) and (17)

$$\left|\beta_k^{HS*}\right| = \left|\frac{g_k^T(g_k - \frac{\|g_k\|}{\|g_{k-1}\|}g_{k-1})}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}\right|$$

$$\leq \frac{\|g_k\|(\|g_k\| + \|g_{k-1}\|) + t\|g_k\|B}{c(1-\sigma)\|g_{k-1}\|^2} \leq \frac{2\bar{\gamma}^2 + t\bar{\gamma}B}{c(1-\sigma)\gamma^2} = b > 1.$$

To obtain $\left|\beta_k^{DL-WYL}\right| \leq 1/2b$,

$$\left|\beta_k^{DL-WYL}\right| \leq \left|\frac{g_k^T(g_k - \frac{\|g_k\|}{\|g_{k-1}\|}g_{k-1})}{d_{k-1}^T y_{k-1}} - t\frac{g_k^T s_{k-1}}{d_{k-1}^T y_{k-1}}\right| \leq \frac{\|g_k\|(\|g_k - g_{k-1}\| + \left\|g_{k-1} - \frac{\|g_k\|}{\|g_{k-1}\|}g_{k-1}\right\| + t\|s_{k-1}\|)}{d_{k-1}^T y_{k-1}}$$

$$\leq \frac{\|g_k\|(\|g_k - g_{k-1}\| + \|\|g_{k-1}\| - \|g_k\|\| + t\|s_{k-1}\|)}{d_{k-1}^T y_{k-1}} \leq \frac{\|g_k\|(\|g_k - g_{k-1}\| + \|\|g_{k-1}\| - \|g_k\|\| + t\|s_{k-1}\|)}{d_{k-1}^T y_{k-1}}$$

$$\leq \frac{\|g_k\|(\|g_k - g_{k-1}\| + \|g_{k-1} - g_k\| + t\|s_{k-1}\|)}{d_{k-1}^T y_{k-1}} \leq \frac{\|g_k\|(L\lambda + L\lambda + t\lambda)}{c(1-\sigma)\gamma^2} = \frac{\lambda\bar{\gamma}(2L+t)}{c(1-\sigma)\gamma^2}$$

Thus we obtain

$$\left|\beta_k^{DL-WYL}\right| \leq \frac{1}{2b}.$$

The forthcoming lemmas correspond to Lemmas 4.1 and 4.2 in [10].

Lemma 3.4. Let Assumption 1 holds. Let the sequences $\{g_k\}$ and $\{d_k\}$ are generated by Algorithm 1 in which α_k is computed by the WWP line search in which the sufficient descent condition (16) holds, and assume that the method has Property*. Suppose that (17) holds. Then there exists $\lambda > 0$ such that for any $\Delta \in N$ and any index k_0, there exist an index $k > k_0$ that satisfies

$$\left|\kappa_{k,\Delta}^{\lambda}\right| > \frac{\lambda}{2},$$

where $\kappa_{k,\Delta}^{\lambda} = \{i \in N : k \leq i \leq k + \Delta - 1, \|s_i\| > \lambda\}$, N denotes the set of positive integers, and $\left|\kappa_{k,\Delta}^{\lambda}\right|$ denotes the number of elements in $\kappa_{k,\Delta}^{\lambda}$.

Lemma 3.5. Let Assumption 1 holds. Let the sequences $\{g_k\}$ and $\{d_k\}$ are generated by Algorithm 1 in which α_k is computed by the WWP line search and the sufficient descent condition (13) holds. If $\beta_k \geq 0$ and (17) hold, then $d_k \neq 0$ and

$$\sum_{k=0}^{\infty}\|u_{k+1}-u_k\|^2<\infty \text{, where } u_k=\frac{d_k}{\|d_k\|}.$$

From Lemmas 3.1 and 3.3–3.5, the global convergence of Algorithm 1 with the WWP line search can be established in a manner that is similar to that of Theorem 4.3 in [10]; therefore, the proof of the following theorem is omitted.

Theorem 3.2. Let the sequences $\{g_k\}$ and $\{d_k\}$ be generated by (1) and (2) with the CG formula β_k^{DL-WYL} and the step size satisfies (4) and (6). If Lemmas 3.1 and 3.3–3.5 are true, then $\liminf_{k\to\infty}\|g_k\|=0$.

4. Numerical results and discussion

To analyse the efficiency of the new formula, we selected several test problems in Table 1 from CUTEr [25]. We performed a comparison with other CG coefficients, including CG-Descent, DY, and WYL coefficients based on the CPU time, number of iterations, number of function evaluations, and number of gradient evaluations. In Table 1 we define the following abbreviations as follows:

No. Iter: Number of iterations;

No. fun.ev: Number of function evaluations;

No. grad.ev: Number of gradient evaluations;

CPU time: central processing unit time in seconds.

We employed the SWP line search with $\delta=0.01$ and $\sigma=0.1$ for all algorithms except CG-Descent we use approximate WWP line search. The norm of the gradient was employed as the stopping criterion, specifically, $\|g_k\|\le 10^{-6}$ for all algorithms. The host computer is AMD A4-7210 APU Radeon R3 Graphics where the installed memory is 4 GB with operating system Ubuntu 20.04.2.0 LTS. The results are shown in Figures 1–4 in which a performance measure introduced by Dolan and More [26] was employed.

As shown from Figure 1 which present the number of iterations, we note β_k^{DL-WYL} strongly out perform all methods. Figure 2 presents the CPU time, we note that β_k^{DL-WYL} also outperform WYL, DL, and CG-Descent methods. From Figure 3 which present the number and gradient evaluations we note that the new method β_k^{DL-WYL} is completive with CG-Descent in terms of number of gradient evaluations since the later use approximation Wolfe-Powell line search for more about this line search the reader can refer to [14,15] in the other hand we can note that β_k^{DL-WYL} out perform all other methods in terms of gradient evaluations since we SWP line search for WYL, DL, and β_k^{DL-WYL}. Figure 4 presents the number of function evaluations we note that the new modification outperform all methods.

Table 1. Test functions.

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
AKIVA	2	8	20	15	0.02	10	21	11	0.02	2	8	20	0.02	8	20	15	0.02
ALLINITU	4	9	25	18	0.02	12	29	18	0.02	9	25	18	0.02	9	25	18	0.03
ARGLINB	200	5	67	67	0.02	5	13	13	0.02	1	3	2	0.02	5	73	72	0.09
ARGLINC	200	5	96	94	0.02	11	106	110	0.02	5	79	78	0.02	5	79	78	0.06
ARWHEAD	5000	6	16	12	0.03	7	15	8	0.02	7	16	12	0.02	6	16	12	0.02
BARD	3	12	32	22	0.02	16	33	17	0.02	12	32	22	0.02	12	32	22	0.02
BDEXP	5000	2	7	7	0.02	5	11	6	0.02	2	7	7	0.02	2	7	7	0.02
BDQRTIC	5000	198	434	389	0.66	136	273	237	0.52	140	318	280	0.47	168	363	359	0.63
BEALE	2	11	33	26	0.02	15	31	16	0.02	11	33	26	0.02	11	33	26	0.02
BIGGS3	6	79	207	144	0.02	110	231	125	0.02	79	207	144	0.02	79	207	144	0.02
BIGGS5	6	79	207	144	0.02	110	231	125	0.02	79	207	144	0.02	79	207	144	0.02
BIGGS6	6	24	64	44	0.02	27	57	31	0.02	24	64	44	0.02	24	64	44	0.02
BIGGSB1	5000	2500	2507	4995	2.37	2500	2507	4995	2.47	2500	2507	4995	2.66	8328	8335	16651	10.86
BOX2	3	10	23	14	0.02	11	24	13	0.02	10	23	14	0.02	10	23	14	0.02
BOX3	3	10	23	14	0.02	11	24	13	0.02	10	23	14	0.02	10	23	14	0.02
BOX	10000	7	25	21	0.09	8	25	19	0.08	7	24	20	0.08	7	25	21	0.09
BRKMCC	2	5	11	6	0.02	5	11	6	0.02	5	11	6	0.02	5	11	6	0.02
BROYDNBDLS	10	25	57	34	0.02	25	51	26	0.02	25	57	34	0.02	25	57	34	0.02
BROWNAL	200	3	11	9	0.03	9	25	18	0.02	9	26	20	0.02	10	29	21	0.02
BROWNBS	2	10	24	18	0.02	13	26	15	0.02	10	24	18	0.02	10	24	18	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
BROWNDEN	4	16	38	31	0.02	16	31	19	0.02	16	38	31	0.02	16	38	31	0.02
BROYDN7D	5000	58	106	80	0.34	1411	2810	1429	5.22	60	111	84	0.28	75	138	112	0.36
BRYBND	5000	72	170	107	0.25	85	174	90	0.28	38	103	77	0.2	149	317	174	0.55
CAMEL6	2	6	22	18	0.02	13	34	22	0.02	6	22	18	0.02	6	22	18	0.02
CHNROSNB	50	282	572	303	0.37	287	564	299	0.02	266	542	291	0.02	1009	1998	1180	0.01
CLIFF	2	10	46	39	0.02	18	70	54	0.02	10	46	39	0.02	10	46	39	0.01
COSINE	10000	12	51	42	0.02	11	39	32	0.19	12	54	44	0.19	12	52	43	0.2
CUBE	2	17	48	34	0.02	32	77	47	0.02	17	48	34	0.02	17	48	34	0.02
CURLY10	10000	51546	72517	82155	184.3	47808	67294	76156	171.25	58045	78735	95434	213.64	68087	88635	1E+05	240.64
CURLY20	10000	74486	98857	1E+05	402	66587	89245	1E+05	377.8	78064	1E+05	1E+05	437.5	88068	1E+05	1E+05	540.5
CURLY30	10000	84928	1E+05	1E+05	588.5	79030	102516	1E+05	635.36	83528	1E+05	1E+05	619.27	93324	1E+05	2E+05	712.27
DENSCHNA	2	6	16	12	0.02	9	19	10	0.02	6	16	12	0.02	6	16	12	0.02
DENSCHNB	2	6	18	15	0.02	7	15	8	0.02	6	18	15	0.02	6	18	15	0.02
DENSCHNC	2	11	36	31	0.02	12	26	14	0.02	11	36	31	0.02	11	36	31	0.02
DENSCHND	3	14	46	40	0.02	47	98	51	0.02	14	46	40	0.02	14	46	40	0.02
DENSCHNE	3	12	43	38	0.02	18	49	32	0.02	12	43	38	0.02	12	43	38	0.02
DENSCHNF	2	9	31	26	0.02	8	17	9	0.02	9	31	26	0.02	9	31	26	0.02
DIXMAANA	3000	7	17	12	0.02	7	15	8	0.02	6	16	13	0.02	6	15	11	0.02
DIXMAANB	3000	6	16	12	0.02	6	13	7	0.02	6	15	11	0.02	6	15	11	0.02
DIXMAANC	3000	6	14	9	0.02	6	13	7	0.02	6	14	9	0.02	6	14	9	0.02
DIXMAAND	3000	6	15	11	0.02	7	15	8	0.02	6	15	11	0.02	7	17	12	0.02
DIXMAANE	3000	248	272	482	0.22	222	239	429	0.23	265	289	516	0.3	394	428	764	0.5
DIXMAANF	3000	98	201	106	0.09	161	323	162	0.17	146	297	154	0.14	247	499	255	0.27

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
DIXMAANG	3000	170	345	178	0.16	157	315	158	0.12	171	347	179	0.14	348	701	356	0.38
DIXMAANH	3000	175	355	183	0.16	173	347	174	0.2	172	351	183	0.2	332	671	343	0.45
DIXMAANI	3000	3215	3311	6344	0.66	3856	3926	7644	4.09	3115	3205	6150	3.33	3522	3623	6953	4.66
DIXMAANJ	3000	332	669	340	0.28	327	655	328	0.25	360	723	365	0.36	476	957	486	0.56
DIXMAANK	3000	304	613	312	0.25	283	567	284	0.22	309	622	316	0.25	425	854	432	0.49
DIXMAANL	3000	249	505	260	0.23	237	475	238	0.2	250	507	261	0.2	320	647	331	0.3
DIXMAANP	3000	618	1241	626	0.5	686	1373	687	0.5	620	1244	627	0.52	857	1721	872	0.89
DIXON3DQ	10000	10000	10007	19995	19.48	10000	10007	19995	19.48	10000	10007	19995	19.48	15258	15265	30511	37.63
DJTL	2				0.02	82	917	880	0.02	75	1163	1148	0.02	75	1163	1148	0.02
DQDRTIC	5000	5	11	6	0.02	5	11	6	0.02	15	32	18	0.02	15	32	18	0.02
ECKERLE4LS.SIF	3	3	7	4	0.03	3	7	4	0.02	2	6	4	0.02	2	6	4	0.02
EDENSCH	2000	26	56	50	0.02	26	52	38	0.02	27	62	53	0.08	27	66	54	0.03
EG2	1000	3	8	5	0.02	5	11	6	0.02	3	13	10	0.02	3	13	10	0.02
EIGENALS	2550	9318	16870	11105	156.5	10083	18020	12244	172.67	7876	13863	9786	139.67	9534	18450	18540	185.64
EIGENBLS	2550	30617	61246	30632	453	15301	30603	15302	225.23	16792	33591	16800	251.25	22540	45340	24700	350.43
EIGENCLS	2652	10020	18835	11248	168.4	10136	19292	11118	167.52	9928	18702	11106	164.84	13450	26740	18450	203.45
ELATVIDU	2	11	25	15	0.02	11	25	15	0.02	8	32	29	0.02	8	32	29	0.02
ENGVAL1	5000	25	49	42	0.06	27	50	36	0.06	23	48	40	0.06	21	48	37	0.06
ENGVAL2	3	26	73	55	0.02	26	61	37	0.02	26	73	55	0.02	26	73	55	0.02
ENSOLS	9	23	45	26	0.02	23	45	26	0.02	22	47	27	0.02	22	47	27	0.02
EXPFIT	2	9	29	22	0.02	13	29	16	0.02	9	29	22	0.02	9	29	22	0.02
EXTROSNB	1000	1130	2671	1686	0.41	3808	7759	3982	1.05	2636	5854	3402	0.83	7182	12662	10680	2.39
exp2	2	9	18	10	0.02	8	17	9	0.02	7	16	9	0.02	7	16	9	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
FBRAINLS	2	8	20	12	0.02	10	23	14	0.02	9	27	21	0.03	9	27	21	0.02
FBRAIN2LS	4	114	305	222	0.53	118	339	248	0.66	79	259	204	0.67	79	259	204	0.47
FLETCBV2	5000	1	1	1	0.02	1	1	1	0.02	1	1	1	0.02	1	1	1	0.02
FLETCHCR	1000	228	449	258	0.16	152	290	176	0.05	268	530	297	0.11	199	412	217	0.08
FMINSRF2	5625	286	577	293	0.98	346	693	347	0.97	316	637	323	0.95	733	1545	826	2.34
FMINSURF	5625	409	823	416	1.28	473	947	474	1.42	410	825	417	1.34	1245	2567	1342	4.08
GENHUMPS	5000	6849	14296	7489	21.42	6475	12964	6493	19.89	7789	16244	8490	22.52	4938	14763	10180	25.27
GROWTHLS	3	109	431	369	0.02	156	456	319	0.02	109	431	369	0.02	109	431	369	0.02
GULF	3	33	95	72	0.02	37	84	48	0.02	33	95	72	0.02	33	95	72	0.02
HAHN1LS	7	5	56	53	0.02	37	121	86	0.02	5	56	53	0.02	5	56	53	0.02
HAIRY	2	17	82	68	0.02	36	99	65	0.02	17	82	68	0.02	17	82	68	0.02
HATFLDD	3	17	49	37	0.02	20	43	24		17	49	37	0.02	17	49	37	0.02
HATFLDE	3	13	37	30	0.02	30	72	45	0.02	13	37	30	0.02	13	37	30	0.02
HATFLDFL	3	21	68	54	0.02	39	92	55	0.02	21	68	54	0.02	21	68	54	0.02
HATFLDFLS	3	48	156	125	0.02	64	155	97	0.02	48	156	125	0.02	48	156	125	0.02
HEART6LS	6	375	1137	876	0.02	684	1576	941	0.02	375	1137	876	0.02	375	1137	876	0.02
HEART8LS	8	253	657	440	0.02	249	524	278	0.02	253	657	440	0.02	253	657	440	0.02
HELIX	3	23	60	42	0.02	23	49	27	0.02	23	60	42	0.02	23	60	42	0.02
HIELOW	3	13	30	21	0.03	14	30	16	0.02	13	30	21	0.03	13	30	21	0.05
HILBERTA	2	2	5	3	0.02	2	5	3	0.02	2	5	3	0.02	2	5	3	0.02
HILBERTB	10	4	9	5	0.02	4	9	5	0.02	4	9	5	0.02	4	9	5	0.02
HIMMELBB	2	4	28	18	0.02	10	28	21	0.02	4	18	18	0.02	4	18	18	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
HIMMELBF	4	23	59	46	0.02	26	60	36	0.02	23	59	46	0.02	23	59	46	0.02
HIMMELBG	2	7	22	17	0.02	8	20	13	0.02	7	22	17	0.02	7	22	17	0.02
HIMMELBH	2	5	13	9	0.02	7	16	9	0.02	5	13	9	0.02	5	13	9	0.02
HUMPS	2	45	223	202	0.02	52	186	146	0.02	45	223	202	0.02	45	223	202	0.02
HYDCAR6LS.SIF	29	53	107	54	0.09	14401	29028	14875	0.45	923866	9E+05	2E+06	39.11	1001	2027	1174	0.03
INDEF	5000	1	46	147	0.36	1	46	147	0.44	1	46	147	0.41	1	46	147	0.42
INTEQNELS.SIF	12	6	13	7	0.02	6	13	7	0.02	6	13	7	0.02	6	13	7	0.02
JENSMP	2	12	47	41	0.02	15	33	22	0.02	12	47	41	0.02	12	47	41	0.02
JIMACK	3549	8331	16664	8333	1172	8314	16629	8315	1169.4	8305	16612	8307	1167.42	11978	23971	12235	1732.9
JUDGE	2	9	24	18	0.02	10	23	13	0.02	9	24	18	0.02	9	24	18	0.02
KOWOSB	4	16	46	32	0.02	17	39	23	0.02	16	46	32	0.02	16	46	32	0.02
KSSLS	1000	6	19	16	0.47	10	25	16	0.58	5	18	16	0.47	6	19	16	0.55
LANCZOS1LS	6	61	177	135	0.02	148	325	181	0.02	61	177	135	0.02	61	177	135	0.02
LANCZOS2LS	6	60	169	125	0.02	169	379	215	0.02	60	169	125	0.02	60	169	125	0.02
LANCZOS3LS	6	61	164	118	0.02	179	392	219	0.02	61	164	118	0.02	61	164	118	0.02
LIARWHD	5000	15	46	34	0.08	21	45	25	0.06	16	47	37	0.05	15	41	31	0.03
LOGHAIRY	2	26	196	179	0.02	27	81	58	0.02	26	196	179	0.02	26	196	179	0.02
LSC1LS	3	31	108	89	0.02	36	101	71	0.02	31	108	89	0.02	31	108	89	0.02
LSC2LS	3	37	106	86	0.02	54	119	67	0.02	37	106	86	0.02	37	106	86	0.02
LUKSAN11LS	100	928	1887	962	0.03	955	1912	957	0.03	926	1894	973	0.03	2434	5355	3048	0.13
LUKSAN12LS	98	162	352	263	0.02	160	302	233	0.02	154	340	268	0.02	252	529	407	0.01
LUKSAN13LS	98	82	168	128	0.02	84	158	121	0.02	85	171	142	0.02	142	279	243	0.02
LUKSAN14LS	98	151	325	211	0.02	98	122	156	0.02	170	370	247	0.02	157	313	201	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
LUKSAN15LS	100	25	57	42	0.02	28	59	44	0.02	26	57	42	0.02	27	60	45	0.02
LUKSAN16LS	100	28	57	41	0.02	31	57	38	0.02	28	58	42	0.02	35	72	53	0.02
MANCINO	100	12	30	19	0.09	11	23	12	0.06	11	23	12	0.06	11	23	12	0.06
MARATOSB	2	589	2885	2585	0.02	1145	3657	2779	0.02	589	2885	2585	0.02	589	2885	2585	0.02
MEXHAT	2	14	59	55	0.02	20	56	39	0.02	14	59	55	0.02	14	59	55	0.02
MEYER3	3	19	76	63	0.02	19	67	52	0.02	19	76	63	0.02	19	76	63	0.02
MGH09LS	4	25	82	72	0.02	57	137	86	0.02	25	82	72	0.02	25	82	72	0.02
MGH10LS	3	1082	4052	4968	0.02	1134	4464	5357	0.03	1082	4052	4968	0.03	1082	4052	4968	0.02
MGH10SLS	3	19	112	102	0.02	146	505	401	0.03	19	112	102	0.03	19	112	102	0.02
MGH17LS	5	84	323	265	0.02	228	564	363	0.02	19	112	102	0.02	84	323	365	0.02
MISRA1BLS.SIF	2	26	113	101	0.02	35	139	117	0.02	84	323	265	0.02	26	113	101	0.02
MISRA1CLS.SIF	2	26	145	121	0.02	26	110	91	0.02	26	145	121	0.02	26	145	121	0.02
MISRA1DLS.SIF	2	22	90	84	0.02	24	74	75	0.02	22	90	84	0.02	22	90	84	0.02
MODBEALE.SIF	20000	47	108	65	1.22	517	1157	1010	13.17	158	340	218	3.53	224	473	304	4.89
MOREBV	5000	161	168	317	0.31	161	168	317	0.42	161	168	317	0.3	117	124	229	0.23
MSQRTALS	1024	2889	5787	2900	8.36	2905	5815	2911	8.89	2845	5699	2856	8.5	8953	17316	9581	28.81
MSQRTBLS	1024	2354	4336	2744	7.48	2280	4525	2326	6.91	2359	4726	23369	7.63	5786	11558	5818	17.72
NCB20	5010	1186	2748	1757	14.72	879	1511	1463	11.38	975	2265	1484	12.5	11026	20505	15341	129.2
NELSONLS	3	1101	5415	7690	0.23	1118	5692	7331	0.17	1101	5415	7690	0.25	1101	5415	7690	0.23
NONCVXU2	5000	6980	13302	7644	16.25	6610	12833	6999	15	7100	13344	7966	16.56	54585	94397	84907	182.92
NONDIA	5000	7	25	19	0.01	7	25	20	0.03	7	25	19	0.03	7	25	19	0.03
NONDQUAR	5000	2354	4770	2454	2.92	1942	3888	1947	2.3	2843	5758	2959	3.28	2349	4787	2488	2.83
OSBORNEA	5	82	230	174	0.02	94	213	124	0.02	82	230	174	0.02	82	230	174	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
OSBORNEB	11	57	134	84	0.02	62	127	65	0.02	57	134	84	0.02	57	134	84	0.02
OSCIPATH	10	3E+05	8E+05	5E+05	2.19	3E+05	670953	4E+05	1.91	295029	8E+05	5E+05	2.3	3E+05	8E+05	5E+05	2.42
PALMER1C	8	12	27	28	0.02	11	26	26	0.02	12	27	28	0.02	12	27	28	0.02
PALMER1D	7	10	24	23	0.02	11	25	25	0.02	10	24	23	0.02	10	24	23	0.02
PALMER2C	8	11	21	22	0.02	11	21	21	0.02	11	21	22	0.02	11	21	22	0.02
PALMER3C	8	11	21	21	0.02	11	20	20	0.02	11	21	21	0.02	11	21	21	0.02
PALMER4C	8	11	21	21	0.02	11	20	20	0.02	11	21	21	0.02	11	21	21	0.02
PALMER5C	6	6	13	7	0.02	6	13	7	0.02	6	13	7	0.02	6	13	7	0.02
PALMER6C	8	11	24	24	0.02	11	24	24	0.02	11	24	24	0.02	11	24	24	0.02
PALMER7C	8	11	20	20	0.02	11	20	20	0.02	11	20	20	0.02	11	20	20	0.02
PALMER8C	8	11	19	19	0.02	11	18	17	0.02	11	19	19	0.02	11	19	19	0.02
PARKCH	15	263	592	361	9.31	672	1385	1128	29	740	1560	1359	34.86	412	982	611	16.02
PENALTY1	1000	20	63	50	0.02	28	69	44	0.02	14	51	43	0.02	14	51	43	0.02
PENALTY2	200	189	223	359	0.03	191	221	354	0.03	192	226	365	0.03	337	480	758	0.06
PENALTY3	200	24	77	60	0.45	99	285	219	1.74	80	295	247	1.94	102	346	290	2.19
PENALTY3	200	24	77	60	0.5	99	285	219	1.75	80	295	247	1.92	102	346	290	2.22
POWELLBSLS	2	50	211	234	0.02	61	247	246	0.02	50	211	234	0.02	50	211	234	0.02
POWELLSG	5000	33	89	64	0.06	26	53	27	0.03	27	68	47	0.03	36	92	65	0.05
POWER	10000	357	731	382	0.58	372	754	384	0.58	359	734	384	0.61	356	733	391	0.58
POWERSUM	4	4	10	6	0.02	5	11	6	0.02	4	10	6	0.02	4	10	6	0.02
PRICE3	2	10	25	17	0.02	11	23	12	0.02	10	25	17	0.02	10	25	17	0.02
PRICE4	2	9	30	23	0.02	16	32	18	0.02	9	30	23	0.02	9	30	23	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
QING	100	66	140	76	0.02	68	135	84	0.02	68	136	86	0.02	85	179	96	0.02
QUARTC	5000	24	72	57	0.05	17	37	21	0.02	15	32	18	0.02	15	32	18	0.02
RAT43LS.SIF	4	44	156	122	0.02	54	145	97	0.02	44	156	122	0.02	44	156	122	0.02
RECIPELS.SIF	3	16	49	38	0.02	27	58	32	0.02	16	49	38	0.02	16	49	38	0.02
ROSENBR	2	28	84	65	0.02	34	77	44	0.02	28	84	65	0.02	28	84	65	0.02
ROSENBRTU.SIF	2	37	175	153	0.02	49	156	113	0.02	37	175	153	0.02	37	175	153	0.02
S308	2	7	21	17	0.02	8	19	12	0.02	7	21	17	0.02	7	21	17	0.02
SCHMVETT	5000	42	73	59	0.19	43	73	60	0.02	40	71	53	0.17	59	103	88	0.28
SENSORS	100	26	67	48	0.47	21	50	34	0.02	27	70	49	0.41	24	71	53	0.47
SINEVAL	2	46	181	153	0.02	64	144	88	0.02	46	181	153	0.02	46	181	153	0.02
SINQUAD	5000	14	52	46	0.09	14	40	33	0.08	13	41	33	0.06	13	46	38	0.09
SISSER	2	5	19	19	0.02	6	18	14	0.02	5	19	19	0.02	5	19	19	0.02
SNAIL	2	61	251	211	0.02	100	230	132	0.02	61	251	211	0.02	61	251	211	0.02
SPMSRTLS	4999	390	744	461	0.94	203	412	210	0.61	391	747	463	0.87	310	633	327	0.81
SROSENBR	5000	9	23	15	0.02	11	23	12	0.02	9	23	15	0.02	9	23	15	0.02
SSI	3	307	1162	990	0.02	345	948	657	0.02	307	1162	990	0.02	307	1162	990	0.02
STREG	4	60	218	180	0.02	96	224	139	0.02	60	218	180	0.02	60	218	180	0.02
STRATEC	10	170	419	283	6.61	462	1043	796	19	170	419	283	6.83	170	419	283	6.3
STRTCHDV.SIF	10	12	38	32	0.02	16	35	20	0.02	12	38	32	0.02	12	38	32	0.02
TESTQUAD	5000	1573	1580	3141	1.27	1577	1584	3149	1.28	1573	1580	3141	1.25	20325	20361	40674	21.61
THURBERLS	7	105	259	216	0.02	102	232	175	0.02	105	259	216	0.02	105	259	216	0.02
TOINTGOR	50	115	213	146	0.02	135	233	174	0.02	124	222	164	0.02	192	348	270	0.02
TOINTGSS	5000	4	9	5	0.02	4	9	5	0.02	4	9	5	0.02	4	9	5	0.02

Continued on next page

function	DIM	No. Iter DL-WYL	No. fun.ev DL-WYL	No. grad.ev DL-WYL	CPU-time DL-WYL	No. iter CG-Descent	No. Fun.ev Descent	No. grad.ev CG DESC	CPU-time CG-Descent	No. Iter WYL	No. fun.ev WYL	No. grad.ev WYL	CPU-time WYL	No. Iter DL	No. fun.ev DLL	No. grad.ev DL	CPU-time DL
TOINTPSP	50	159	339	361	0.02	143	279	182	0.02	117	258	196	0.02	145	313	250	0.02
TOINTQOR	50	29	36	53	0.02	29	36	53	0.02	29	36	53	0.02	49	56	93	0.02
TQUARTIC	5000	12	44	36	0.02	14	40	27	0.02	12	43	35	0.02	11	41	34	0.03
TRIDIA	5000	781	788	1557	0.87	782	7889	1559	0.89	782	789	1559	0.91	4699	4721	9408	6.38
TRIGON1	10	19	41	22	0.02	22	45	23	0.02	19	41	22	0.02	19	41	22	0.02
TRIGON2	10	22	57	43	0.02	26	52	28	0.02	22	57	43	0.02	22	57	43	0.02
VANDANMSLS.	22	5	11	6	0.02	5	12	7	0.02	5	11	6	0.02	5	11	6	0.02
VARDIM	200	10	24	19	0.02	10	21	11	0.02	9	20	15	0.02	9	20	15	0.02
VAREIGVL	50	24	51	29	0.02	23	47	21	0.02	24	51	29	0.02	28	71	51	0.02
VESUVIALS	8	1262	1954	3155	1.22	1519	2317	4111	1.56	1262	1954	3155	1.45	1262	1954	3155	1.22
VESUVIOULS	8	79	211	173	0.09	80	180	131	0.06	79	211	173	0.14	79	211	173	0.09
VIBRBEAM	8	98	255	174	0.02	138	323	199	0.02	98	255	174	0.02	98	255	174	0.02
WAYSEA1	2	11	55	50	0.02	18	39	22	0.02	11	55	50	0.02	11	55	50	0.02
WAYSEA2	2	9	28	23	0.02	31	68	39	0.02	9	28	23	0.02	9	28	23	0.02
WOODS	4000	22	52	33	0.02	22	51	30	0.03	24	63	45	0.03	24	62	41	0.03
YATP1CLS	1E+05	17	51	39	7.72	23	53	31	6.13	17	50	38	7.31	17	48	36	7.12
YATP2CLS	1E+05	7	28	23	2.66	8	23	17	2.09	7	29	24	2.8	7	27	22	2.67
YFITU	3	68	208	167	0.02	84	197	118	0.02	68	208	167	0.02	68	208	167	0.03
ZANGWIL2	2	1	3	2	0.02	1	3	2	0.02	1	3	2	0.02	1	3	2	0.02

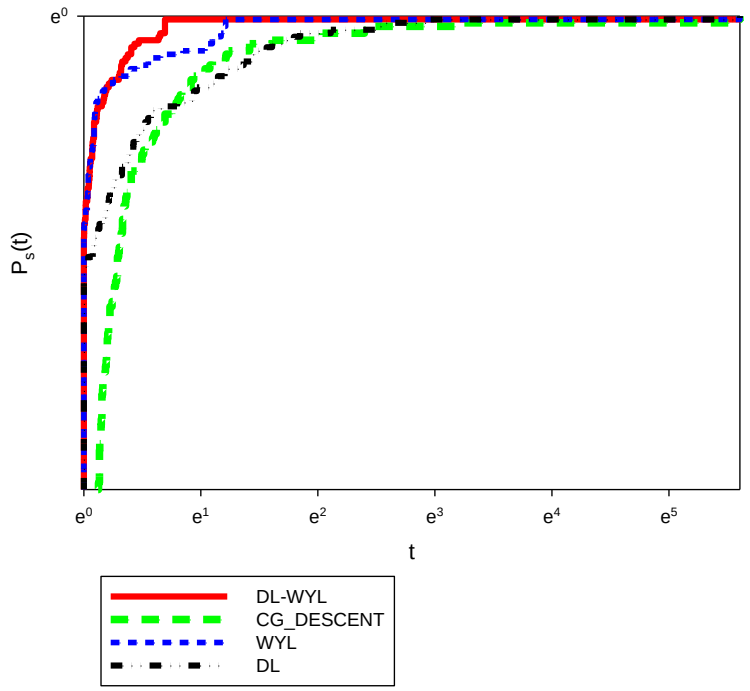

Figure 1. Performance measure based on the number of iterations.

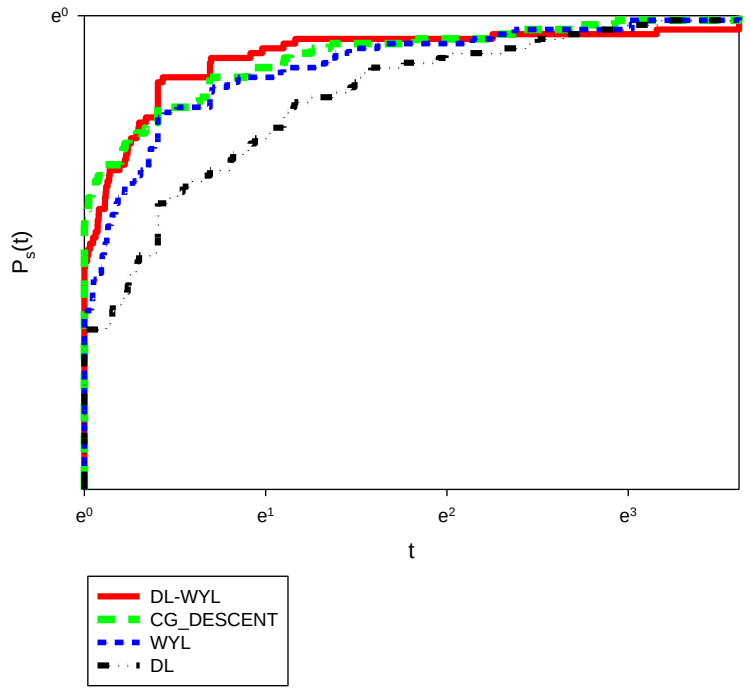

Figure 2. Performance measure based on the CPU time.

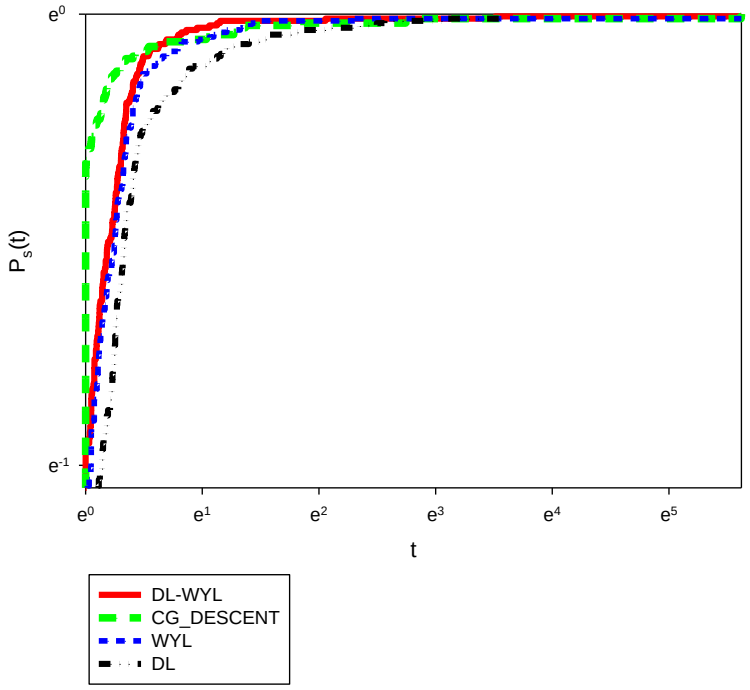

Figure 3. Performance measure based on the gradient evaluations.

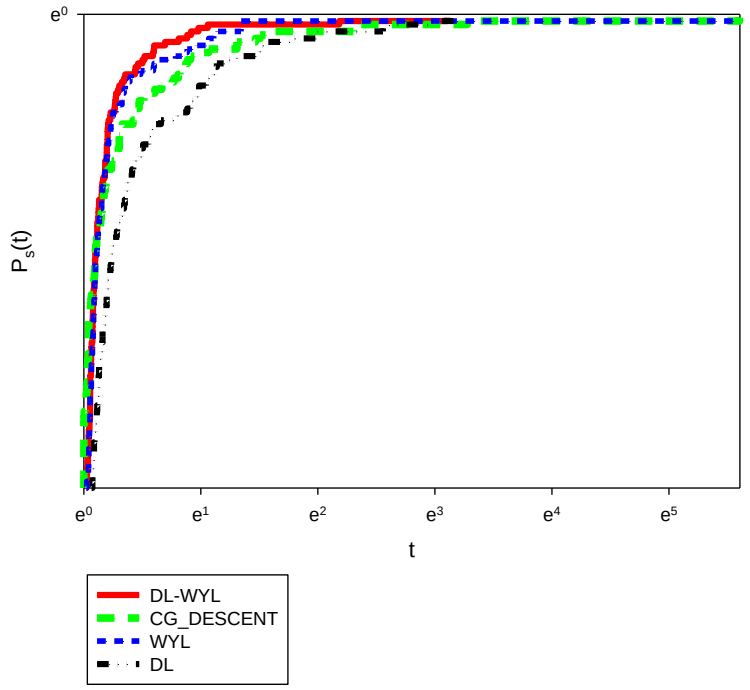

Figure 4. Performance measure based on the function evaluations.

5. Conclusions

In this paper, we propose a new three term CG method based on WYL and DL CG methods. We have also proved that the CG algorithm which employs the new coefficient has global convergence properties with the sufficient descent condition when SWP line search is employed. In numerical result part we show that the new three term CG method is more efficient than recent CG methods such as WYL, DL, and CG-Descent methods.

Availability of data and material

The data available inside the paper.

Acknowledgements

The authors are grateful for all these supported and improve our paper, also we would like to thank the University of Malaysia Terengganu (UMT) for funding this paper.

Conflict of interest

The authors declare that they have no competing interests.

References

1. P. Wolfe, Convergence conditions for ascent methods, *SIAM Rev.*, **11** (1969), 226–235.
2. P. Wolfe, Convergence conditions for ascent methods. II: Some corrections, *SIAM Rev.*, **13** (1971), 185–188.
3. M. R. Hestenes, E. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. Nat. Bur. Standards*, **49** (1952), 409–435.
4. E. Polak, G. Ribiere, Note sur la convergence de méthodes de directions conjuguées, *ESAIM: Math. Modell. Numer. Anal.*, **3** (1969), 35–43.
5. Y. Liu, C. Storey, Efficient generalized conjugate gradient algorithms, part 1: Theory, *J. Optim. Theory Appl.*, **69** (1991), 129–137.
6. R. Fletcher, C. M. Reeves, Function minimization by conjugate gradients, *Comput. J.*, **7** (1964), 149–154.
7. R. Fletcher, *Practical methods of optimization*, Vol. 1, Unconstrained Optimization, A Wiley-Interscience Publication, 1997.
8. Y. H. Dai, Y. Yuan, A nonlinear conjugate gradient method with a strong global convergence property, *SIAM J. Optim.*, **10** (1999), 177–182.
9. M. J. Powell, Nonconvex minimization calculations and the conjugate gradient method, In: *Numerical analysis*, Springer, Berlin, Heidelberg, (1984), 122–141.
10. J. C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM J. Optim.*, **2** (1992), 21–42.
11. G. Zoutendijk, Nonlinear programming, computational methods, *Integer Nonlinear Program.*, (1970), 37–86.

12. M. Al-Baali, Descent property and global convergence of the Fletcher-Reeves method with inexact line search, *IMA J. Numer. Anal.*, **5** (1985), 121–124.
13. Y. H. Dai, L. Z. Liao, New conjugacy conditions and related nonlinear conjugate gradient methods, *Appl. Math. Optim.*, **43** (2001), 87–101.
14. W. W. Hager, H. Zhang, A new conjugate gradient method with guaranteed descent and an efficient line search, *SIAM J. Optim.*, **16** (2005), 170–192.
15. W. W. Hager, H. Zhang, The limited memory conjugate gradient method, *SIAM J. Optim.*, **23** (2013), 2150–2168.
16. Z. Wei, S. Yao, L. Liu, The convergence properties of some new conjugate gradient methods *Appl. Math. Comput.*, **183** (2006), 1341–1350.
17. A. Alhawarat, Z. Salleh, M. Mamat, M. Rivaie, An efficient modified Polak-Ribière-Polyak conjugate gradient method with global convergence properties, *Optim. Methods Software*, **32** (2017), 1299–1312.
18. P. Kaelo, P. Mtagulwa, M. V. Thuto, A globally convergent hybrid conjugate gradient method with strong Wolfe conditions for unconstrained optimization, *Math. Sci.*, **14** (2020), 1–9.
19. S. Yao, L. Ning, H. Tu, J. Xu, A one-parameter class of three-term conjugate gradient methods with an adaptive parameter choice, *Optim. Methods Software*, **35** (2020), 1051–1064.
20. G. Yuan, J. Lu, Z. Wang, The PRP conjugate gradient algorithm with a modified WWP line search and its application in the image restoration problems, *Appl. Numer. Math.*, **152** (2020), 1–11.
21. G. Yuan, T. Li, W. Hu, A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems, *Appl. Numer. Math.*, **147** (2020), 129–141.
22. J. Liu, S. Du, Y. Chen, A sufficient descent nonlinear conjugate gradient method for solving M-tensor equations, *J. Comput. Appl. Math.*, **371** (2020), 112709.
23. S. Yao, Y. Wu, J. Yang, J. Xu, A three-term gradient descent method with subspace techniques, *Math. Probl. Eng.*, **2021** (2021), 8867309.
24. A. Alhawarat, N. Trung, Z. Salleh, Conjugate gradient method: A developed version to resolve unconstrained optimization problems, *J. Comput. Sci.*, **16** (2020), 1220–1228.
25. I. Bongartz, A. R. Conn, N. Gould, P. L. Toint, CUTE: Constrained and unconstrained testing environment, *ACM Trans. Math. Software*, **21** (1995), 123–160.
26. E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.*, **91** (2002), 201–213.