Electronic Research Archive

https://www.aimspress.com/journal/era

ERA, 33(8): 4799–4815.
DOI: 10.3934/era.2025216
Received: 07 May 2025
Revised: 23 June 2025
Accepted: 24 July 2025
Published: 19 August 2025

Research article

An improved version of Polak-Ribière-Polyak conjugate gradient method with its applications in neural networks training

Osman Omer Osman Yousif[1], Raouf Ziadi[2], Abdulgader Z. Almaymuni [3,*] and Mohammed A. Saleh[3]

[1] Department of Mathematics, Faculty of Mathematical and Computer Sciences, University of Gezira, Wad Madani, Sudan

[2] Laboratory of Fundamental and Numerical Mathematics (LMFN), Department of Mathematics, University Setif - 1 - Ferhat Abbas, Setif, Algeria

[3] Department of Cybersecurity, College of Computer, Qassim University, Saudi Arabia

* **Correspondence:** Email: almaymuni@qu.edu.sa.

Abstract: Due to their simplicity, low memory requirements, strong convergence properties, and ability to solve problems of high dimensions, the conjugate gradient (CG) methods are widely used to solve linear and non-linear unconstrained optimization problems. The Polak-Ribière-Polyak (PRP) is considered as one of the most efficient CG methods in practical computation. However, theoretically, its convergence properties are poor. Therefore, many variants of PRP with good numerical results and good convergence properties have been developed, such as Gilbert and Nocedal method (PRP$^+$), Wei-Yau-Liu method (WYL), and Yousif et al. method (OPRP). In this paper, based on PRP$^+$ and OPRP methods, we proposed another modified version of PRP that inherits all the convergence properties of PRP$^+$ and OPRP and has improved numerical results. To show the efficiency and robustness of the new modified method in practice, it was compared with PRP$^+$, WYL, and OPRP when they are all applied under the strong Wolfe line search. At the same time, the new method was applied in deep learning to obtain ideal parameters of some neural network (NN) models during the training process.

Keywords: optimization method; Polak-Ribière-Polyak conjugate gradient method; global convergence; neural networks

1. Introduction

Conjugate gradient (CG) methods comprise a class of unconstrained optimization algorithms designed to solve the following:

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function, bounded from below. They use the following iterative formula:

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \ldots, \tag{1.1}$$

where α_k is a positive real number called step length that is obtained by a line search method, and the directions d_k are computed by the rule

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{1.2}$$

where β_k is update coefficient, and g_k is the gradient vector of $f(x)$ at the point x_k. Different formulas for β_k define different CG methods. Well-known formulas for β_k are the Hestenes-Stiefel (HS) [1], Fletcher-Reeves (FR) [2], and Polak-Ribière-Polyak (PRP) [3, 4] are given as follows:

$$\begin{aligned} \beta_k^{\text{HS}} &= \frac{g_k^T(g_k - g_{k-1})}{d_{k-1}^T(g_k - g_{k-1})}, \\ \beta_k^{\text{FR}} &= \frac{\|g_k\|^2}{\|g_{k-1}\|^2}, \\ \beta_k^{\text{PRP}} &= \frac{g_k^T(g_k - g_{k-1})}{\|g_{k-1}\|^2}, \end{aligned}$$

respectively. The HS, FR, and PRP CG methods, together with the conjugate descent (CD) [5], Liu-Storey (LS) [6], and Dai-Yuan (DY) [7], are called the classical CG methods. For more formulas for the coefficient β_k, see references [8–10].

The step length α_k is computed using exact or inexact methods called line searches. In an exact line search, α_k is obtained in the direction d_k by the following rule:

$$f(x_k + \alpha_k d_k) = \min_{\alpha\geq 0} f(x_k + \alpha d_k). \tag{1.3}$$

Since it is difficult to compute α_k using formula (1.3) in practice, the inexact line searches are introduced to compute approximate values for α_k. The Wolfe and strong Wolfe line searches are examples of the inexact line search often used in practice. In the Wolfe line search [11, 12], α_k satisfies the following two conditions:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta\alpha_k g_k^T d_k, \tag{1.4}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k. \tag{1.5}$$

Additionally, the strong Wolfe line search includes condition (1.4) and

$$|g(x_k + \alpha_k d_k)^T d_k| \leq \sigma|g_k^T d_k|, \tag{1.6}$$

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

The efficiency of CG methods is measured by two things: their global convergence is

$$\lim_{k\to\infty} \|g_k\| = 0,$$

and their numerical results in practice. Both measurements essentially depend on the well-chosen line search and the search direction d_k.

To guarantee that every search direction generated by a CG method is descent, the sufficient descent property

$$g_k^T d_k \leq -C\|g_k\|^2, \quad \forall k \geq 0, \quad C > 0 \text{ is a constant}$$

is always considered.

The CG methods exhibit exceptional local search capabilities; thus, considerable efforts have been carried out to propose new CG methods [13–15] to modify existing CG methods for improved performances [16, 17] or to combine them with other methods [18]. Zoutendijk [19] proved the global convergence of the FR method under the exact line search, which was later extended by Al-Baali [20] under the strong Wolfe line search. Polak and Ribière [4] established the global convergence of the PRP with the exact line search. However, Powell [21] later reported that the PRP does not converge globally for some non-convex problems. Despite the good results of the PRP method in practice, its global convergence under the strong Wolfe line search has not been established. The PRP addresses the jamming problem that occurs in the FR method since it possesses a sufficient descent feature, that is, when $x_k - x_{k-1}$ tends to zero, the numerator of β_k^{PRP} also tends to zero, so the next search direction is the steepest descent direction. This attribute underscores the resilience of the PRP method in addressing challenging situations frequently faced in optimization, thus facilitating more efficient convergence even in intricate environments. An additional investigation into its worldwide convergence using a robust Wolfe line search is essential to strengthen its theoretical basis and expanding its usefulness. Since the PRP and HS coefficients have the same numerator, their convergence behavior is almost similar. Therefore, the PRP and HS perform much better than the FR in practice; hence, in spite of their poor convergence properties, they are preferred over the FR method. Several efforts have been made to prove the global convergence of HS and PRP via the inexact line search or to propose convergent modified versions of them; see [22] for HS and [23–25] for PRP. Powell [21] suggested restricting the coefficient β_k^{PRP} to be nonnegative. By adopting this suggestion, Gilbert and Nocedal [26] proposed a modified PRP method whose coefficient $\beta_k^{PRP^+}$ is of nonnegative values of β_k^{PRP} and is given by

$$\beta_k^{PRP^+} = \max\left(0, \beta_k^{PRP}\right),$$

which can be equivalently expressed as

$$\beta_k^{PRP^+} = \begin{cases} \beta_k^{PRP} & \text{if } \beta_k^{PRP} > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{1.7}$$

They showed that under the sufficient descent property, the global convergence of the PRP$^+$ method is satisfied. In 2006, Wei et al. [27] proposed a modified formula of $\beta_k^{PRP^+}$ given by the following:

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

In addition to the good numerical results of the WYL method, it possesses sufficient descent property and global convergence properties under the strong Wolfe line search.

Most recently, Yousif et al. [28] proposed a modified version of PRP by restricting the coefficient β_k^{PRP} in some interval depending on a parameter μ, where μ is a real number greater than 2. The modified version is called OPRP, and its coefficient is given by the following:

$$\beta_k^{\text{OPRP}} = \begin{cases} \beta_k^{\text{PRP}} & \text{if } -\mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} < \beta_k^{\text{PRP}} < \mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} \\ 0 & \text{otherwise} \end{cases}. \tag{1.8}$$

They have proven the global convergence and the sufficient descent property of the OPRP method when the line is the strong Wolfe line search.

In this paper, motivated by the good results of the PRP in practice and the convergence properties of both the PRP$^+$ and OPRP methods, we propose a new modified coefficient of PRP, and hence a new modified method in Section 2. The sufficient descent property and the global convergence of the new method are proven in Section 3. To show the efficiency and robustness of the new modified method in practice, a numerical experiment is conducted in Section 4. In Section 5, the new algorithm is applied for training some neural network (NN) problems. This is followed by a conclusion in Section 6.

2. Another new modified method

As mentioned in Section 1, the main issues when studying CG methods are their theoretical global convergence and their numerical performances. Although the PRP CG method gives good numerical results in practice when it is applied to solve unconstrained optimization problems, its global convergence has not yet been established when it is implemented under the strong Wolfe line search to solve non-convex functions. This drawback drew much attention to better variants of PRP that, at least, preserve the good numerical results of the PRP with the global convergence. One of these variants is the PRP$^+$ method, whose coefficient $\beta_k^{\text{PRP}+}$ takes the non-negative values of β_k^{PRP} and sets the other negative values of β_k^{PRP} to zero; see (1.7). Another variant is the OPRP variant, whose coefficient β_k^{OPRP} takes positive and negative values of β_k^{PRP} in the interval $\left(-\mu\frac{\|\mathbf{g}_k\|^2}{\|\mathbf{d}_{k-1}\|^2}, \mu\frac{\|\mathbf{g}_k\|^2}{\|\mathbf{d}_{k-1}\|^2}\right)$ and sets all other values to zero; see (1.8). Both modified methods have good numerical results and are globally convergent under the strong Wolfe line search. In this paper, we propose another variant of the PRP CG method by combining $\beta_k^{\text{PRP}^+}$ and β_k^{OPRP} to obtain the following:

$$\beta_k^{\text{NEW}} = \begin{cases} \beta_k^{\text{PRP}} & \text{if } \beta_k^{\text{PRP}} > -\mu\frac{\|\mathbf{g}_k\|^2}{\|\mathbf{d}_{k-1}\|^2} \\ 0 & \text{otherwise} \end{cases}, \tag{2.1}$$

where $\mu > 2$ is a real number.

Clearly, the proposed coefficient takes more values of β_k^{PRP}, so it is a better modification of β_k^{PRP}. Hence, if the modified method whose coefficient is defined by (2.1) gives good numerical results in practice and is globally convergent under the strong Wolfe line search, then it can be considered a better variant than PRP$^+$ and OPRP. Hence, the main purpose of this study is to show that the new modified method, whose coefficient is given by (2.1), gives good numerical results and is globally convergent when applied under the strong Wolfe line search.

The new modified CG method can be described by the following algorithm:

Algorithm 1: A modified PRP algorithm

Input: Choose the parameters δ and σ such that $0 < \delta < \sigma < 1$. Choose a scalar $\epsilon > 0$ sufficiently small to stop the algorithm.

// **Initialization**

- Set $k = 0$ and select a starting point $x_0 \in \mathbb{R}^n$.
- Set $g_0 = \nabla f(x_0)$ and $d_0 = -g_0$.

// **Execute the following steps while** $\|g_k\| \geq \epsilon$

while $\|g_k\| \geq \epsilon$ **do**

- Compute the steplegth α_k that meets strong Wolfe conditions (1.4) and (1.6).
- Set $x_{k+1} = x_k + \alpha_k d_k$
- Evaluate $g_{k+1} = \nabla f(x_{k+1})$
- Compute the search direction $d_{k+1} = -g_{k+1} + \beta_k^{NEW} d_k$.
- Set $k = k + 1$.

From the definition (2.1), it is clear that when $\mu \to \infty$, the new modified coefficient tends to the coefficient of the PRP. Hence, for a sufficiently large value of μ, the new method gives a good approximation to the PRP method. Additionally, since the new modified coefficient combines both the coefficients of PRP$^+$ and OPRP, and since the global convergence of both PRP$^+$ and OPRP algorithms has been proven, we expect that the global convergence of Algorithm 1 is also satisfied. We will investigate this in the next section.

3. Convergence properties

This section studies the convergence properties of Algorithm 1. To proceed, we suppose that the objective function $f(x)$ satisfies the following assumption:

Assumption 3.1.

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

(ii) In some neighborhood N *of* Ω*, the objective function is continuously differentiable and its gradient is Lipschitz continuous, namely, there exists a constant* $l > 0$ *such that*

$$\|g(x) - g(y)\| \leq l\|x - y\|, \quad \forall\, x, y \in N.$$

Note that Assumption 3.1 implies the existence of a positive constant $\bar{y}$ such that

$$\|g(x)\| \leq \bar{y}, \text{ for all } x \in N.$$

In addition to Assumption 3.1, the Zoutendijk condition [19] is usually used to analyze the global convergence of conjugate gradient methods.

Lemma 3.1. *Suppose that Assumption 3.1 holds. Consider any CG method of the form* (1.1) *and* (1.2)*, where the step length* α_k *is computed by the Wolfe line search. Then, the following condition, known as the Zoutendijk condition, is satisfied:*

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

For good convergence results, we need to require that β_k be small when the step length is small. This property is called Property*, which was defined by Gilbert and Nocedal [26].

Property*: Consider a method of the form (1.1) and (1.2), and suppose that for all $k \geq 0$,

$$0 < \gamma \leq \|g_k\| \leq \bar{\gamma},$$

where γ and $\bar{\gamma}$ are two positive constants. Then, the method has Property* if there exist two constants $b > 1$ and $\lambda > 0$ such that for all k,

$$|\beta_k| \leq b,$$

and

$$\|x_k - x_{k-1}\| \leq \lambda \Rightarrow |\beta_k| \leq \frac{1}{2b}.$$

Theorem 3.1. *Algorithm 1 possesses the Property*.*

Proof. The proof follows straightforwardly from the fact that the PRP method possesses Property* and the coefficient β_k^{NEW} of Algorithm 1 is defined as a restriction of β_k^{PRP}.

The convergence of Algorithm 1 essentially depends on the following two results:

Result 1. *In [28], Yousif et al. proved that if a CG method with a parameter β_k satisfies the condition*

$$|\beta_k| \leq \mu \frac{\|g_k\|^2}{\|d_{k-1}\|^2}, \quad \text{for } k \geq 1 \text{ and a real number } \mu \geq 1,$$

which is applied under the strong Wolfe line search with $0 < \sigma < \frac{1}{4\mu}$ to solve unconstrained optimization problems, then,

(a) $\frac{\|g_k\|}{\|d_k\|} < 2$ *for all* $k \geq 0$,
(b) the sufficient descent condition is satisfied,
(c) under Assumption 3.1, the method is globally convergent.

Result 2. *In [26], Gilbert and Nocedal proved that any CG method under the Wolfe line search (under the strong Wolfe line search with $0 < \sigma < \frac{1}{4\mu}$) that satisfies*

(a) $\beta_k \geq 0$ *for all k,*
(b) the sufficient descent condition,
(c) Property*,
(d) Assumption 3.1,

is globally convergent.

Based on the two results above, we establish the convergence of our new modified method.

Theorem 3.2. *Suppose that Assumption 3.1 holds. Then, Algorithm 1 is globally convergent when applied under the strong Wolfe line search with $0 < \sigma < \frac{1}{4\mu}$.*

Proof. By rewriting the coefficient β_k^{NEW} as follows:

$$\beta_k^{\text{NEW}} = \begin{cases} \beta_k^{\text{PRP}} & \text{if } -\mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} < \beta_k^{\text{PRP}} < \mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} \\ \beta_k^{\text{PRP}} & \text{if } \beta_k^{\text{PRP}} > \mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} \\ 0 & \text{otherwise} \end{cases},$$

we consider the following two cases:

Case I: $-\mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2} \leq \beta_k^{\text{NEW}} < \mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2}$

The proof is straightforward from Result 1.

Case II: $\beta_k^{\text{NEW}} > \mu\frac{\|g_k\|^2}{\|d_{k-1}\|^2}$

Since Algorithm 1 satisfies Property*, the proof follows from Result 2. This completes the proof.

4. Numerical experiments

In this section, to demonstrate the efficiency and robustness, we compare the new modified method with the PRP, PRP$^+$, WYL, and OPRP methods based on the following criteria:

- Most of the test problems are from [29].
- To show robustness, the test problems are implemented under low, medium, and high dimensions: 2, 3, 4, 5, 10, 50, 100, 500, 1000, 5000, and 10,000. For each dimension, two different initial points are chosen for each test problem: one of them is the initial point suggested by Andrei [29], and the other is chosen arbitrarily.
- All methods are applied under the strong Wolfe line search with parameters $\delta = 10^{-4}$ and $\sigma = 10^{-2}$.
- The termination condition for all methods is set to $\|g_k\| \leq 10^{-6}$.
- The parameter μ is set to 10.
- All algorithms were coded using MATLAB and the run was conducted on a PC computer with an Intel R Core TM i5-2520 M CPU @ 2.50 GHz processor, 4 GB of RAM memory, and a Windows 10 Professional operating system.

The numerical results are presented in Tables 1 and 2 with the following notations:

Dim The dimension of the test problem

NI The number of iterations

CPU The computation time required to solve a test problem (in seconds)

NF The number of function evaluations

NG The number of gradient evaluations

FAIL Indicates when a method failed to solve a test problem

NEW The new modified method

Table 1. Numerical results for problems of low dimensions.

Problem Number	Test Problem	Dim	Initial Vector	PRP	PRP$^+$	WYL	OPRP	NEW
				NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG
1	THREE-HUMP [30]	2	(2, 2)	13/0.04/362/86	11/0.03/245/100	14/0.04/403/127	13/0.04/362/86	13/0.04/362/86
			(5, 5)	11/0.03/306/88	FAIL	FAIL	FAIL	9/0.02/197/58
2	GENERALIZED WHITE & HOLST [29]	2	(0, 0)	FAIL	16/0.02/114/57	43/0.04/230/125	18/0.02/115/64	16/0.02/114/57
			(10, 10)	49/0.07/548/207	37/0.05/349/152	77/0.07/530/245	50/0.06/431/194	37/0.05/349/152
3	SIX-HUMP [30]	2	(1.1)	5/0.01/22/14	5/0.01/22/14	8/0.02/31/20	5/0.01/22/14	5/0.01/22/14
			(10, 10)	9/0.02/53/24	11/0.03/65/33	9/0.03/57/23	12/0.03/77/35	9/0.03/53/24
4	TRECANNI [31]	2	(1, 1)	6/0.01/22/15	5/0.01/20/13	7/0.01/26/17	5/0.01/19/13	6/0.01/22/15
			(10.10)	6/0.01/32/22	8/0.02/35/21	11/0.2/45/28	9/0.02/38/23	8/0.02/35/21
5	ZeTtLe [32]	2	(1, 1)	10/0.02/40/30	10/0.02/40/30	17/0.03/68/54	10/0.02/40/30	10/0.02/40/30
			(10, 10)	11/0.02/48/33	11/0.02/51/34	12/0.03/55/39	10/0.02/48/32	11/0.02/51/34
6	BIGGSB1 [29]	2	(0, 0)	1/0.01/3/2	1/0.01/3/2	1/0.01/3/2	1/0.01/3/2	1/0.01/3/2
			(10, 10)	1/0.01/3/2	1/0.01/3/2	1/0.1/3/2	1/0.01/3/2	1/0.01/3/2
7	LEON [32]	2	(0, 0)	FAIL	16/0.03/114/57	43/0.06/230/125	18/0.04/115/64	16/0.03/114/57
			(10.10)	49/0.07/548/207	37/0.05/349/152	74/0.08/515/236	50/0.06/431/194	37/0.05/349/152
8	DIXON & PRICE [32]	3	(1, 1, 1)	10/0.01/40/26	10/0.01/40/26	14/0.02/51/33	10/0.01/40/26	10/0.01/40/26
			(10, 10, 10)	45/0.03/165/101	23/0.02/108/60	53/0.04/229/145	21/0.02/102/56	21/0.02/102/56
9	CUBE [29]	3	$(-1.2, 1. - 1, 2)$	840.0.43/3167/1941	170/0.12/875/455	1512/.75/5495/3589	84/0.07/487/275	123/0.09/650/330
			(0, 0, 0)	528/0.28/2049/1222	1510/76/5607/3581	120/0.08/574/302	75/0.07/440/245	968/0.48/3537/2221
10	NONDIA [29]	3	$(-1, -1, -1)$	515/0.24/1783/1162	97/0.06/399/234	288/0.14/1007/643	46/0.04/238/135	97/0.06/399/234
			(10, 10, 10)	984/0.49/3653/2226	131/0.10/728/350	223/0.15/1171/590	85/0.07/489/276	131/0.10/728/350
11	Extended WOOD [29]	4	(0, 0, 0, 0)	106/0.06/428/243	106/0.06/428/243	73/0.04/282/170	57/0.03/250/142	106/0.06/428/243
			(5, 5, 5, 5)	157/0.09/697/383	114/0.07/527/278	240/0.13/991/570	85/0.06/458/233	139/0.08/576/332
12	LIARWHD [29]	4	(4, 4, 4, 4)	FAIL	18/0.09/502/172	22/0.12/660/210	25/0.15/685/251	26/0.15/794/287
			(10, 10, . . .)	19/0.06/464/148	22/0.08/588/180	16/0.05/477/194	14/0.05/362/138	19/0.06/464/148
13	COLVILLE [29]	4	(0, 0, 0, 0)	106/0.06/428/243	106/0.06/428/243	73/0.04/282/170	57/0.03/250/142	106/0.06/428/243
			(10, 10, . . .)	177/0.10/793/432	391/0.19/1609/909	305/0.15/1197/694	78/0.06/428/217	391/0.19/1609/909
14	EXTENDED POWELL [29]	4	(1, 1, . . .)	70/0.04/279/172	1303/49/3957/2635	256/0.11/835/557	56/0.03/259/178	35/0.03/173/105
			(10, 10, . . .)	1290/48/3964/2623	1290/48/3964/2623	301/0.12/988/643	59/0.04/314/212	1290/48/3964/2623
15	NONSCOMP [29]	5	(0, 0, . . .)	690/0.24/2157/1464	690/0.24/2157/1464	474/0.28/2413/1636	68/0.04/273/200	690/0.24/2157/1464
			(10, 10, . . .)	185/0.09/738/449	185/0.09/738/449	400/0.16/1414/910	227/0.11/881/621	185/0.09/738/449
16	ENGVAL1 [29]	5	(2, 2, . . .)	22/0.02/76/50	23/0.02/80/53	24/0.02/83/54	22/0.02/76/50	22/0.02/76/50
			$(-1, -1, \ldots)$	17/0.02/63/41	17/0.02/63/41	20/0.02/73/48	17/0.02/63/41	17/0.02/63/41
17	GENERALIZED TRIDIAGONAL 2 [29]	5	(1, 1, . . .)	14/0.02/47/36	7/0.01/28/21	9/0.01/34/25	7/0.02/28/21	7/0.01/28/21
			(10, 10, . . .)	48/0.05/191/136	21/0.02/94/65	29/0.03/114/81	25/0.03/106/76	22/0.02/96/68
18	GENERALIZED TRIDIAGONAL 1 [29]	10	(2, 2, . . .)	23/0.02/76/50	23/0.02/76/50	23/0.02/76/50	23/0.02/76/50	23/0.02/76/50
			(10, 10, . . .)	27/0.03/112/68	27/0.03/112/68	26/0.03/110/67	27/0.03/112/68	27/0.03/112/68
19	EDENSCH [29]	10	(0, 0, . . .)	24/0.02/80/52	24/0.02/80/52	24/0.02/80/52	24/0.02/80/52	24/0.02/80/52
			$(-1, -1, \ldots)$	20/0.02/72/47	20/0.02/72/47	21/0.02/74/48	23/0.02/80/52	20/0.02/72/47
20	SUM SQUARE [29]	10	$(-1, -1, \ldots)$	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20
			(10, 10, . . .)	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20	10/0.01/30/20
21	POWER [29]	50	$(-1.2, \ldots)$	67/0.05/201/134	67/0.05/201/134	66/0.04/198/132	1486/87/4458/2972	67/0.05/201/134
			(10, 10, . . .)	67/0.05/201/134	67/0.05/201/134	67/0.05/201/134	1668/96/5004/3336	67/0.05/201/134
22	HAGER [29]	50	(1, 1, . . .)	19/0.02/65/41	19/0.02/65/41	19/0.02/59/40	19/0.02/65/41	19/0.02/65/41
			(10, 10, . . .)	960/1.03/6456/3909	897/0.96/5982/3635	435/0.46/2823/1724	729/0.69/4106/2810	870/0.98/5835/3499
23	FLETCHER [29]	50	(0, 0, . . .)	274/0.21/1319/664	274/0.21/1319/664	292/0.23/1375/699	600/0.39/2322/1334	274/0.21/1319/664
			(10, 10, . . .)	217/0.14/845/493	222/0.15/866/511	215/0.13/836/488	556/0.33/1899/1199	222/0.15/866/511
24	RAYDAN 1 [29]	50	(1, 1, . . .)	47/0.03/144/137	47/0.03/144/137	47/0.03/144/138	47/0.03/144/137	47/0.03/144/137
			(5, 5, . . .)	93/0.06/325/286	93/0.06/302/286	78/0.05/276/238	133/0.08/445/392	93/0.06/325/286
25	ARWHEAD [29]	50	(1, 1, . . .)	5/0.02/23/13	5/0.02/23/13	6/0.03/26/15	5/0.02/23/13	5/0.02/23/13

Table 2. Numerical results for problems of medium and high dimensions.

Problem Number	Test Problem	Dim	Initial Vector	PRP	PRP$^+$	WYL	OPRP	NEW
				NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG	NI/CPU/NF/NG
26	EDENSCH [29]	100	$(0, 0, \ldots)$	25/0.04/101/56	25/0.04/101/56	25/0.04/101/56	25/0.04/101/56	25/0.04/101/56
			(10.10,...)	28/0.06/183/68	26/0.05/109/63	28/0.06/128/70	26/0.05/109/63	26/0.05/109/63
27	GENERALIZED ROSENBROCK [29]	100	$(-1.2, 1, \ldots)$	840/1.20/4751/2191	841/1.32/4751/2197	834/1.13/4691/2165	FAIL	840/1.18/4751/2191
			$(5, 5, \ldots)$	854/1.29/4896/2241	855/1.19/4889/2234	851/1.21/4866/2221	FAIL	854/1.29/4896/2241
28	HIMMELH [29]	100	$(0, 0, \ldots)$	5/0.02/15/10	6/0.02/23/13	10/0.03/30/20	6/0.02/18/12	6/0.02/23/13
			$(0.5, 0.5, \ldots)$	5/0.02/15/10	5/0.02/15/10	8/0.03/38/17	5/0.02/15/10	5/0.02/15/10
29	GENERALIZED QUARTIC [29]	100	$(2, 2, \ldots)$	9/0.03/89/44	6/0,02/55/36	8/0.03/118/34	5/0.01/18/13	6/0.02/55/36
			$(5, 5, \ldots)$	19/0.09/387/190	19/0.09/432/170	FAIL	FAIL	19/0.09/387/190
30	EXTENDED MARATOS [29]	100	$(1.1, 0.1, \ldots)$	FAIL	FAIL	FAIL	50/0.07/314/155	37/0.05/270/122
			$(1, 1, \ldots)$	15/0.03/104/48	17/0.03/120/52	33/0.05/207/97	21/0.04/122/67	17/0.03/120/51
31	EXTENDED PENALTY [29]	500	$(1, 1, \ldots)$	26/0.11/175/81	28/0.16/208/95	20/0.08/111/53	12/0.05/65/34	21/0.10/152/70
			$(-1, -1, \ldots)$	9/0.04/38/21	9/0.06/96/22	FAIL	9/0.04/42/21	9/0.06/96/22
32	TRIDIA [29]	500	$(1, 1, \ldots)$	239/0.48/717/478	239/0.48/717/478	240/0.49/720/480	FAIL	239/0.48/717/478
			$(10, 10, \ldots)$	251/0.52/753/502	251/0.52/753/502	251/0.52/753/502	FAIL	251/0.52/753/502
33	QF2 [29]	500	$(0.5, 0.5, \ldots)$	253/0.56/897/563	253/0.56/897/563	254/0.61/900/564	207/0.49/767/479	253/0.56/897/563
			$(10, 10, \ldots)$	227/0.60/880/511	227/0.60/880/511	233/0.67/897/527	227/0.60/880/511	227/0.60/880/511
34	QP2 [29]	500	$(1, 1, \ldots)$	40/0.33/481/142	42/ 0.35/471/134	98/0.68/976/291	57/0.47/613/190	40/0.33/465/135
			$(10, 10, \ldots)$	42/0.34/469/141	40/0.33/458/131	110/0.74/1059/320	57/0.43/610/188	39/0.32/452/131
35	QF1 [29]	500	$(1, 1, \ldots)$	131/0.27/393/262	131/0.27/393/262	131/0.27/393/262	131/0.27/393/262	131/0.27/393/262
			$(10, 10, \ldots)$	140/0.30/420/280	140/0.30/420/280	140/0.30/420/280	140/0.30/420/280	140/0.30/420/280
36	QP1 [29]	1000	$(1, 1, \ldots)$	9/0.06/42/24	8/0.05/39/22	FAIL	10/0.07/63/25	8/0.05/39/22
			$(3, 3, \ldots)$	FAIL	9/0.07/73/22	13/0.08/67/31	9/0.07/50/21	9/0.07/73/22
37	PERTURBED QUADRATIC [29]	1000	$(0.5, 0.5, \ldots)$	187/0.61/561/374	187/0.61/561/374	187/0.61/561/374	523/1.71/1569/1046	187/0.61/561/374
			$(10, 10, \ldots)$	203/0.67/609/406	203/0.67/609/406	203/0.67/609/406	629/2.12/1887/1258	203/0.67/609/406
38	DIXON3DQ [29]	1000	$(-1, -1, \ldots)$	500/1.35/1508/1009	500/1.35/1508/1009	500/1.35/1508/1009	FAIL	500/1.35/1508/1009
			$(10, 10, \ldots)$	500/1.35/1508/1009	500/1.35/1508/1009	500/1.35/1508/1009	FAIL	500/1.35/1508/1009
39	DQDRTIC [29]	1000	$(3, 3, \ldots)$	18/0.06/54/36	16/0.05/48/32	15/0.04/45/30	11/0.03/33/22	18/0.06/54/36
			$(10, 10, \ldots)$	23/0.07/69/46	18/0.06/54/36	16/0.05/48/32	11/0.03/33/22	23/0.07/69/46
40	EXTENDED DENSCHNF [29]	1000	$(2, 0, \ldots)$	16/0.32/335/149	21/0.39/417/109	26/0.49/535/129	19/0.44/482/186	16/0.32/335/149
			$(10.10, \ldots)$	FAIL	20/0.40/365/120	23/0.43/393/120	18/0.36/391/198	18/0.36/391/198
41	FREUDENSTEIN & ROTH [29]	5000	$(0.5, -2, \ldots)$	FAIL	7/0.15/36/19	10/0.19/45/24	7/0.15/36/19	7/0.15/36/19
			$(5, 5, \ldots)$	10/0.21/52/25	8/0.18/47/22	FAIL	FAIL	8/0.18/47/22
42	EXTENDED TRIDIAGONAL 1 [29]	5000	$(2, 2, \ldots)$	14/0.40/72/58	14/0.40/72/58	10/0.26/44/32	12/0.34/61/50	14/0.40/72/58
			$(10, 10, \ldots)$	13/0.39/73/62	19/0.47/84/67	8/0.23/42/34	8/0.25/47/38	13/0.39/73/62
43	DIAGONAL 4 [29]	5000	$(1, 1, \ldots)$	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5
			$(10, 10, \ldots)$	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5	2/0.03/6/5
44	EXTENDED DENSCHNB [29]	5000	$(1, 1, \ldots)$	6/0.10/22/16	5/ 0.08/19/14	10/0.13/34/24	5/ 0.08/19/14	5/ 0.08/19/14
			$(10, 10, \ldots).$	8/0.12/34/21	10/0.15/45/29	12/0.18/50/32	9/0.13/37/23	8/0.12/34/21
45	EXTENDED ROSENBROCK [29]	5000	$(-1.2, 1, \ldots)$	21/0.40/134/67	20/0.37/123/69	55/0.88/287/155	28/0.50/167/88	23/0.43/142/72
			$(10, 10, \ldots)$	25/0.54/183/72	34/0.67/228/106	41/0.74/247/115	33/0.65/218/101	32/0.63/216/102
46	EXTENDED HIMMELBLAU [29]	10^4	$(1, 1, \ldots)$	8/0.22/32/19	8/0.22/32/19	13/0.33/47/29	8/0.22/32/19	8/0.22/32/19
			$(10, 10, \ldots)$	7/ 0.21/32/17	8/0.23/34/18	10/0.28/40/22	8/0.23/35/19	8/0.23/35/19
47	STRAIT [32]	10^4	(0.0...)	18/0.53/90/51	18/0.53/90/51	26/0.70/116/70	15/0.47/80/44	18/0.53/90/51
			$(5, 5, \ldots)$	20/1.01/126/59	16/0.77/143/54	36/1.40/246/107	18/0.97/182/67	20/1.01/189/72
48	SHALLOW [33]	10^4	$(0, 0, \ldots)$	7/0.18/27/21	7/ 0.18/28/21	11/0.26/40/30	8/0.20/31/24	7/ 0.18/27/20
			$(10, 10, \ldots)$	14/0.40/66/39	11/0.34/56/33	19/0.50/80/54	12/0.38/62/39	13/0.41/68/43
49	EXTENDED BEALE [29]	10^4	$(1, 0.8, \ldots)$	14/1.01/70/43	14/1.01/72/46	20/1.15/82/54	14/1.01/72/46	14/1.01/72/46
			(3.3. ...)	9/0.7/49/28	11/0.81/56/35	12/ 0.86/61/36	13/0.89/66/40	9/0.7/49/28
50	EXTENDED WHITE & HOLST [29]	10^4	$(-1.2, 1, \ldots)$	15/1.00/98/47	17/1.05/106/56	24/1.41/139/68	15/0.99/95/50	17/1.05/106/56
			$(10, 10, \ldots)$	FAIL	38/3.38/352/154	83/5.49/551/262	50/4.17/431/194	38/3.38/352/154

Now, based on the numerical results in Tables 1 and 2, we use the performance profile introduced by Dolan and Moré [34] to show the performance of each method. The Dolan and Moré performance profile provides solver efficiency, robustness, and the probability of success by plotting the fraction P of the test problems for which any given method is within a factor t of the best. Hence, the top curved shape of the method is a winner. Additionally, the right side of the plot is a measure of a method's robustness. The results are shown in Figures 1–4.

It is clear from the left sides of Figures 1–4 that the new method and the PRP$^+$ methods seem to be similar, and they are the best solvers. However, despite the similarity, there are some problems that failed to be solved by PRP$^+$ which were successfully solved using the new method. Additionally, the right side of all figures shows that the percentage of the test problems that are successfully solved by

the new method is higher than the other, which reflects its robustness. Additionally, it is clear that the curve of the new method is above all other curves. Therefore, we conclude that the new method performs better than the PRP, PRP$^+$, WYL, and OPRP methods.

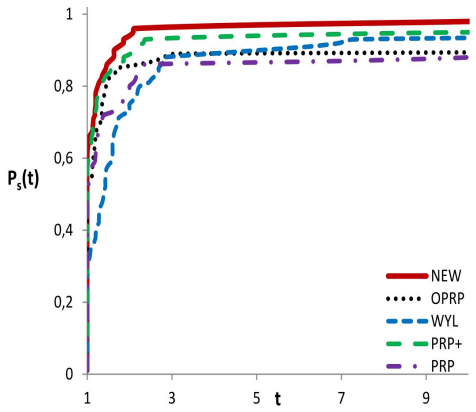

Figure 1. The performance in terms of NI.

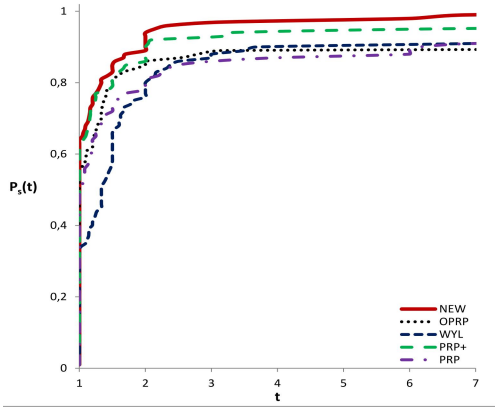

Figure 2. The performance in terms of CPU.

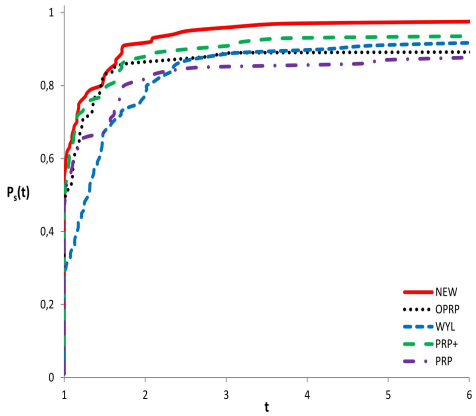

Figure 3. The performance in terms of NF.

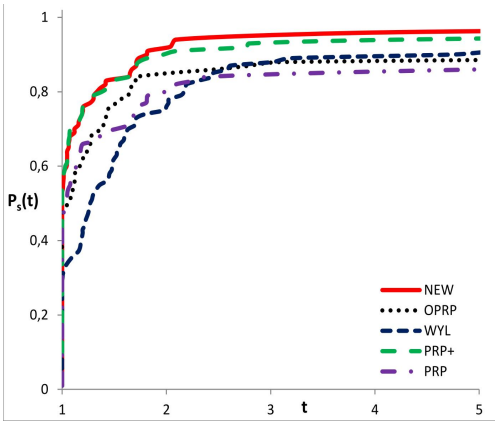

Figure 4. The performance in terms of NG.

5. Application in neural network training

An NN is a machine learning (ML) model that is inspired by the structure and function of biological networks in the human brain. It consists of connected units or nodes called artificial neurons that represent the input, hidden, and output layers. These are connected by edges, and each has a weight value. In the training process, the inputs are multiplied by the corresponding weights and summed at each neuron.Then, the sum transfers to an activation function, which is chosen such that it has a mathematically favorable derivative, which makes it easier to compute partial derivatives of the error function with respect to individual weights. Therefore, in most cases, a sigmoid function, hyperbolic tangent, or rectified linear unit (ReLU) is preferred. A sigmoid function is a bounded, differentiable, real function that is defined for all real input values and has a non-negative derivative at each point and exactly one inflection point.They most often return values either in the range 0 to 1 or in the range from -1 to 1. In addition to sigmoid functions, data scientists use linear activation functions, also known as identity functions, when they want the output to be the same as the input. It is a linear function that has the following:

$$y_{\text{out}} = \sigma(x) = x, \quad \forall x.$$

Then, the output of the activation function is fed as the input to the subsequent unit in the next layer. The result of the final output layer is used as the solution for the problem.

NNs can be used in various problems, including pattern recognition, classification, clustering, dimensionality reduction, computer vision, natural language processing (NLP), regression, predictive analysis, etc.

The implementation of NNs consists of the following steps:

i) Acquire a training and testing data set.
ii) Train the network.
iii) Make a prediction with test data.

In NN, most studies are concerned with developing optimization algorithms to obtain better performance results (to find the optimal values for the weights) by improving the parameters of the

NN-based model during the training process. The task of these algorithms is to obtain ideal parameters of the NN model, which is usually referred as the cost function [35–37].

The conjugate gradient method has gained popularity in training NNs due to its speed and simplicity. Unlike a basic gradient descent, which may oscillate or converge slowly, the conjugate gradient method leverages past gradient information to determine the search directions, which leads to faster convergence. The training time can be significantly reduced when feedforward networks are trained using these second-order methods, thus making them highly suitable for large-scale and complex problems [38].

In this study, we consider the model illustrated by Figure 5, where $x_1, \ldots, x_n$ with $x_i \in \mathbb{R}^n$, $i = 1, 2, \ldots, n$ is the input that represents the features, $w = [w_1, \ldots, w_n]^T$ represents the weights to be evaluated during the training process, b_m represents the bias, and $y = [y_1, \ldots, y_m]^T$ is the output to be predicted.

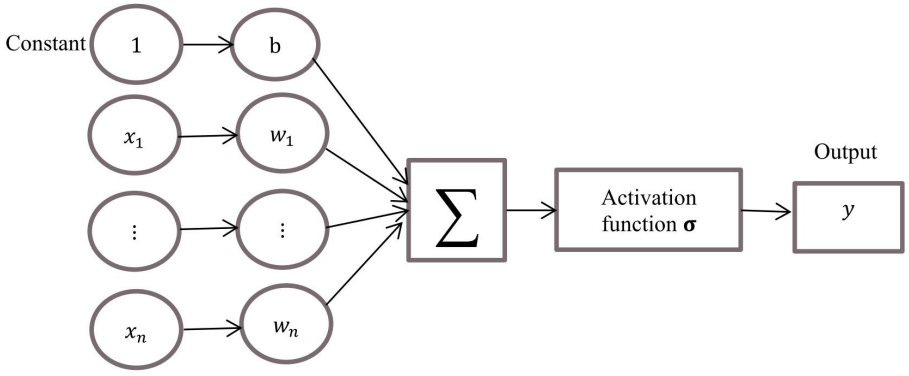

Figure 5. Neural network training process.

From Figure 5, the model can be given using the finite sum as follows:

$$y_i = \sigma\left(\left(\sum_{k=1}^{n} x_{ik} w_k\right) + b_i\right), \quad i = 1, 2, \ldots, m, \tag{5.1}$$

where σ is the activation function.

Since the task in the training process stage is to obtain minimum values of the weight vectors, then the error function MSE or $E(w)$ is as follows:

$$E(w) = \frac{1}{2} \sum_{i=1}^{m} \left(y_i - \sigma\left(\left(\sum_{k=1}^{n} x_{ik} w_k\right) + b_i\right)\right)^2,$$

which is a continuous and differentiable function in the weights w that can be optimized to obtain the minimum values of w. Hence, the conjugate gradient methods can be applied.

Additionally, to predict the optimum value of w, we can express relation (5.1) in a matrix form as follows:

$$\sigma\left[\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{pmatrix} \times \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}\right] = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix},$$

or simply

$$y = \sigma(Aw + b). \tag{5.2}$$

By taking σ as the linear activation function, we find that the solution of Eq (5.2) is equivalent to the solution of the following unconstrained optimization problem of the quadratic objective function:

$$\min_{\theta \in \mathbb{R}^n} \left(\frac{1}{2}\theta^T Q\theta + \theta^T \gamma + c\right), \tag{5.3}$$

where $Q = 2A^TA$, $\gamma = -2A^Tb$, and $c = b^Tb$.

Therefore, the new CG method can be used to solve problem (5.3) as follows:

Algorithm 2: The new method for training NNs

Input: Input the features $x_1, \ldots, x_n$ as columns of a matrix $A_{n\times m} = [x_1 : \cdots : x_n]$, bias $b \in \mathbb{R}^m$, initial guess $x_0 \in \mathbb{R}^n$. Choose a scalar $\epsilon > 0$ sufficiently small to stop the algorithm.

// **Initialization**

- Set $Q = 2A^TA$, $\gamma = -2A^Tb$, $c = b^Tb$, and $k = 0$.
- Set $g_0 = Qx_0 + \gamma$ and $d_0 = -g_0$

// **Execute the following steps whenever** $\|g_k\| \geq \epsilon$

while $\|g_k\| \geq \epsilon$ **do**

- Compute the learning rate $\alpha_k = \dfrac{-g_k^T d_k}{d_k^T Q d_k}$.
- Set $x_{k+1} = x_k + \alpha_k d_k$.
- Evaluate $g_{k+1} = Qx_{k+1} + \gamma$.
- Compute the search direction $d_{k+1} = -g_{k+1} + \beta_k^{NEW} d_k$.
- Set $k = k + 1$.

To practically, test the efficiency of Algorithm 2, we consider the following three problems:

Problem 1

Inputs		Outputs
x_1	x_2	y
3	4	1
4	2	1
7	8	1
8	3	1
5	4	0
9	2	0
0	4	0

Problem 2

Inputs				Outputs
x_1	x_2	x_3	x_4	y
6	5	8	9	1
2	3	6	7	1
9	7	5	2	1
0	1	0	3	1
5	7	9	1	0
6	9	2	1	0
7	7	8	3	0

Problem 3

Inputs								Outputs
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	y
6	5	8	9	4	5	6	7	1
2	3	6	7	9	8	7	6	1
9	7	5	2	1	1	1	1	1
0	1	0	3	3	2	1	3	1
5	7	9	1	8	9	9	6	0
6	9	2	1	8	7	6	4	0
7	7	8	3	4	5	5	6	0

Now, by applying Algorithm 2 on the above problems, we obtain the following results reported in Table 3.

Table 3. NN training results.

No.	Test Problem	Number of Training	Weights	Bias	CPU Time	MSE
1	Problem 1	3	0.0252, 0.0641	0.1946	3.3240e-04	7.8486e-27
2	Problem 2	5	0.1607, -0.2274, -0.0613, 0.0612	1.1400	3.5830e-04	3.2670e-19
3	Problem 3	8	-0.1387, 0.0470, 0.0647, 0.1901, -0.0045, -0.1417, 0.1557, -0.3408	1.5465	4.4810e-04	7.8591e-24

In addition to Problems 1–3, we implement Algorithm 2 on randomly generated low and larger datasets, which show the efficiency of Algorithm 2 and its ability to be extended to real-world datasets such as the MNIST database (Modified National Institute of Standards and Technology database) [39]. To complete this task, the `MATLAB` command *randi* $(100, m, n)$ is used to generate n vectors, each of m integers between 1 and 100. The result is in Table 4.

Table 4. The result on random Datasets.

No.	Dataset Size	Number of Training	CPU Time	MSE
1	2×5	5	$2.480e^{-4}$	$2.7623e^{-23}$
2	3×10	10	$2.865e^{-4}$	$4.6783e^{-20}$
3	5×15	15	$3.805e^{-4}$	$6.9371e^{-18}$
4	10×20	20	$6.575e^{-4}$	$7.1256e^{-25}$
5	20×25	25	0.0013	$5.7892e^{-22}$
6	50×30	30	0.0025	$7.8673e^{-25}$
7	100×35	35	0.0052	$6.6386e^{-19}$
8	500×40	40	0.044	$6.1721e^{-24}$
9	1000×50	50	0.19	$7.9034e^{-23}$
10	10000×10	10	9.28	$5.6870e^{-26}$

From Tables 3 and 4 , we can deduce that the new algorithm can be successfully used for training NNs due to the low values of the errors.

6. Conclusions

This paper proposed another modified version of the PRP conjugate gradient method. The convergence of the new method was proved when the line search satisfies the strong Wolfe conditions. To demonstrate the efficiency and robustness of the modified method in practical computations, it was compared under the strong Wolfe line search with the PRP, PRP$^+$, WYL, and OPRP methods. The comparison was based on four criteria: the number of iterations, the central processing unit (CPU) time, the number of function evaluations, and the number of gradient evaluations. It was reported that the new modified method performs better than the others. Additionally, based on some NN problems, the new method successfully estimated the parameters of NN models during the training process.

However, it is still a limitation of this study to compute the learning rate using the exact line search.

To overcome this limitation, future work could explore the development of an adaptive learning rate strategy, which would further enhance the performance and applicability of the model.

Use of AI tools declaration

We declare that we have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The Researchers would like to thank the Deanship of Graduate Studies and Scientific Research at Qassim University for financial support (QU-APC-2025).

Conflict of interest

The authors declare there are no conflicts of interest.

References

1. M. R. Hestenes, E. Steifel, Method of conjugate gradient for solving linear equations, *J. Res. Nat. Bur. Stand.*, **49** (1952), 409–436.
2. R. Fletcher, C. Reeves, Function minimization by conjugate gradients, *Comput. J.*, **7** (1964), 149–154. https://doi.org/10.1093/comjnl/7.2.149
3. B. T. Polyak, The conjugate gradient method in extreme problems, *USSR Comput. Math. Math. Phys.*, **9** (1969), 94–112. https://doi.org/10.1016/0041-5553(69)90035-4
4. E. Polak, G. Ribière, Note sur la convergence de directions conjugées, *Rev. Fr. Inf. Rech. Opér. Ser. Rouge*, **3** (1969), 35–43. https://doi.org/10.1051/m2an/196903R100351
5. R. Fletcher, Unconstrained optimization, in *Practical Method of Optimization*, Wiley, (2000), 1–136. Available from: https://onlinelibrary.wiley.com/doi/book/10.1002/9781118723203.
6. Y. Liu, C. Storey, Efficient generalized conjugate gradient algorithms part 1: Theory, *J. Optim. Theory Appl.*, **69** (1991), 129–137. https://doi.org/10.1007/BF00940464
7. Y. H. Dai, Y. Yuan, A nonlinear conjugate gradient with a strong global convergence properties, *SIAM J. Optim.*, **10** (1999), 177–182. https://doi.org/10.1137/S1052623497318992
8. A. Abdelrahman, O. O. O. Yousif, M. Mogtaba, M. K. Elbahir, Global convergence of nonlinear conjugate gradient coefficients with inexact line search, *Sci. J. King Faisal Univ. Basic Appl. Sci.*, **22** (2021), 86–91. https://doi.org/10.37575/b/sci/210058
9. A. Abdelrahman, M. Mohammed, O. O. O. Yousif, M. K. Elbahir, Nonlinear conjugate gradient coefficients with exact and strong Wolfe line searches techniques, *J. Math.*, **2022** (2022), 1383129. https://doi.org/10.1155/2022/1383129
10. O. Omer, M. Rivale, M. Mamat, Z. Amani, A new conjugate gradient method with sufficient descent without any line search for unconstrained optimization, *AIP Conf. Proc.*, **1643** (2015), 602–608. https://doi.org/10.1063/1.4907500

11. P. Wolfe, Convergence conditions for ascent methods, *SIAM Rev.*, **11** (1969), 226–235. https://doi.org/10.1137/1011036
12. P. Wolfe, Convergence conditions for ascent methods. II: Some corrections, *SIAM Rev.*, **13** (1971), 185–188. https://doi.org/10.1137/1013035
13. O. Omer, M. Mamat, A. Abashar, M. Rivale, The global convergence properties of a conjugate gradient method, *AIP Conf. Proc.*, **1602** (2014), 286–295. http://doi.org/10.1063/1.4882501
14. O. Omer, M. Rivale, M. Mamat, A. Abdalla, A new conjugate gradient method and its global convergence under the exact line search, *AIP Conf. Proc.*, **1635** (2014), 639–646. https://doi.org/10.1063/1.4903649
15. O. O. O. Yousif, A. Abdelrahman, M. Mohammed, M. A. Saleh, A sufficient condition for the global convergence of conjugate gradient methods for solving unconstrained optimisation problems, *Sci. J. King Faisal Univ. Basic Appl. Sci.*, **23** (2022), 106–112. https://doi.org/10.37575/b/sci/220013
16. O. O. O. Yousif, M. A. Saleh, Another modified version of RMIL conjugate gradient method, *Appl. Numer. Math.*, **202** (2024), 120–126. https://doi.org/10.1016/j.apnum.2024.04.014
17. O. O. O. Yousif, R. Ziadi, M. A. Saleh, A. Z. Almaymuni, Another updated parameter for the Hestenes-Stiefel conjugate gradient method, *Int. J. Anal. Appl.*, **23** (2025), 10. https://doi.org/10.28924/2291-8639-23-2025-10
18. Y. Zhao, F. Wu, J. Pang, W. Zhong, New heterogeneous comprehensive learning particle swarm optimizer enhanced with low-discrepancy sequences and conjugate gradient method, *Swarm Evol. Comput.*, **93** (2025), 101848. https://doi.org/10.1016/j.swevo.2025.101848
19. G. Zoutendijk, Nonlinear programming computational methods, *Integer Nonlinear Program.*, **1970** (1970), 37–86.
20. M. Al-Baali, Descent property and global convergence of Fletcher-Reeves method with inexact line search, *IMA J. Numer. Anal.*, **5** (1985), 121–124. https://doi.org/10.1093/imanum/5.1.121
21. M. J. D. Powell, Convergence properties of algorithm for nonlinear optimization, *SIAM Rev.*, **28** (1986), 487–500. https://doi.org/10.1137/1028154
22. 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. https://doi.org/10.1137/030601880
23. G. Yuan, X. Lu, A modified PRP conjugate gradient method, *Ann. Oper. Res.*, **166** (2009), 73–90. https://doi.org/10.1007/s10479-008-0420-4
24. L. Zhang, An improved Wei-Yao-Liu nonlinear conjugate gradient method for optimization computation, *Appl. Math. Comput.*, **215** (2009), 2269–2274. https://doi.org/10.1016/j.amc.2009.08.016
25. Z. Wei, G. Li, L. Qi, New nonlinear conjugate gradient formulas for large-scale unconstrained optimizations problems, *Appl. Math. Comput.*, **179** (2006), 407–430. https://doi.org/10.1016/j.amc.2005.11.150
26. J. C. Gilbert, J. Nocedal, Global convergence properties of conjugate gradient methods for optimization, *SIAM J. Optim.*, **2** (1992), 21–42. https://doi.org/10.1137/0802003

27. Z. Wei, S. Yao, L. Liu, The convergence properties of some new conjugate gradient methods, *Appl. Math. Comput.*, **183** (2006), 1341–1350. https://doi.org/10.1016/j.amc.2006.05.150

28. O. O. O. Yousif, M. A. Y. Mohammed, M. A. Saleh, M. K. Elbashir, A criterion for the global convergence of conjugate gradient methods under strong Wolfe line search, *J. King Saud Univ. Sci.*, **34** (2022), 1–7. https://doi.org/10.1016/j.jksus.2022.102281

29. N. Andrei, An unconstrained optimization test functions collection, *Adv. Modell. Optim.*, **10** (2008), 147–161.

30. M. Molga, C. Smutnicki, Test functions for optimization needs, **101** (2005), 32.

31. W. X. Zhu, A class of filled functions irrelevant to the number of local minimizers for global optimization (in Chinese), *J. Syst. Sci. Math. Sci.*, **22** (2004), 406–413.

32. S. Mishra, Some new test functions for global optimization and performance of repulsive particle swam method, *SSRN*, (2006), 1–24. https://doi.org/10.2139/ssrn.926132

33. I. A. R. Moghrabi, Minimization of extended quadratic functions with inexact line searches, *J. Korean Soc. Ind. Appl. Math.*, **9** (2005), 55–61.

34. E. Dolan, J. J. Moré, Benchmarking optimization software with performance profile, *Math. Program.*, **91** (2002), 201–213. https://doi.org/10.1007/s101070100263

35. M. Roohi, S. Mirzajani, A. R. Haghighi, A. Basse-O'Connor, Robust design of two-level non-integer SMC based on deep soft actor-critic for synchronization of chaotic fractional order memristive neural networks, *Fractal Fract.*, **8** (2024), 548. https://doi.org/10.3390/fractalfract8090548

36. M. Roohi, C. Zhang, M. Taheri, A. Basse-O'Connor, Synchronization of fractional-order delayed neural networks using dynamic-free adaptive sliding mode control, *Fractal Fract.*, **7** (2023), 682. https://doi.org/10.3390/fractalfract7090682

37. M. Roohi, C. Zhang, Y. Chen, Adaptive model-free synchronization of different fractional-order neural networks with an application in cryptography, *Nonlinear Dyn.*, **100** (2020), 3979–4001. https://doi.org/10.1007/s11071-020-05719-y

38. P. P. van der Smagt, Minimisation methods for training feedforward neural networks, *Neural Networks*, **7** (1994), 1–11. https://doi.org/10.1016/0893-6080(94)90052-3

39. L. Deng, The MNIST Database of handwritten digit images for machine learning research, *IEEE Signal Process. Mag.*, **29** (2012), 141–142. https://doi.org/10.1109/MSP.2012.2211477

AIMS Press