Bol. Soc. Paran. Mat.
©SPM – E-ISSN-2175-1188
SPM: www.spm.uem.br/bspm

(3s.) **v. 2026 (44)** : 1–15.
ISSN-0037-8712
doi:10.5269/bspm.79860

Neural Network Training and Solution of Minimization Problems Via Novel Conjugate Gradient Methods

Basim A. Hassan[1], Alaa Luqman Ibrahim[2]*, Thaair Ameen[3]

ABSTRACT: In this study, we introduce a new conjugate gradient method designed to solve large-scale unconstrained optimization problems and train neural networks. The method was developed to satisfy the descent condition, ensuring both stability and efficiency. We also proved that the proposed method achieves global convergence under standard assumptions. To evaluate its effectiveness, we conducted numerical experiments on a variety of benchmark optimization problems, including test cases from the CUTE collection, as well as standard functions such as the penalty function, sine function, and Diagonal1 function. The performance was assessed based on the number of iterations, number of function evaluations, and computational time. The results clearly demonstrate that our proposed method outperforms the classical Hestenes-Stiefel method, offering faster convergence with fewer iterations and reduced computational cost. In addition, we applied the method to train neural networks and compared its performance with that of the standard conjugate gradient algorithm. These experiments, carried out using MATLAB and the Neural Network Toolbox, showed that our method significantly enhances the training efficiency, achieving a lower mean squared error in fewer epochs. Overall, the proposed conjugate gradient method offers a more effective and computationally efficient approach for solving unconstrained optimization problems and neural network training, making it a promising tool for future research and real-world machine-learning applications.

Keywords: Optimization algorithms, conjugate gradient methods, neural network training, gradient-based, performance analysis.

Contents

1 Introduction **1**

2 Research Methodology of the Proposed Method and Algorithm **4**
2.1 Theoretical Derivation of the Proposed Method 4
2.2 Algorithm Implementation 5

3 Descent Condition and Global Convergence **5**

4 Numerical Experiments **7**
4.1 Unconstrained Optimization Problems 7
4.2 The Proposed Method in Neural Network Training Applications 8

5 Conclusion **11**

1. Introduction

Artificial Neural Networks (ANNs) are now a key part of modern machine learning research and development. In general, machine learning concerns a computer's ability to identify and understand data patterns using mathematical or statistical models. However, ANNs use more advanced techniques to manage complex patterns within large datasets. The implementation of ANNs requires detailed methods and algorithms. ANN training can follow various learning approaches such as supervised, unsupervised, and reinforcement learning. Despite significant progress, research on neural network learning algorithms continues to grow and evolve, gaining more attention and recognition within the scientific community [1].

Recently, logic mining techniques using hybrid discrete Hopfield neural networks have shown promise for symbolic learning and attribute selection. For instance, in [2], a logic mining method combined differential evolution and swarm mutation operators to enhance binary classification, whereas in [3],

* Corresponding author.
2020 *Mathematics Subject Classification:* 65K05, 68T07, 90C30.
Submitted November 02, 2025. Published April 01, 2026

a log-line-based attribute selection mechanism was integrated into 2SATRA logic mining, improving interpretability and performance. These advances emphasize the importance of hybrid optimization and logical rule integration in learning models. Inspired by the success of hybrid optimization in logical rule integration, this work focuses on combining analytical convergence guarantees with practical performance in neural network training.

Training neural networks (NNs) can be formulated as an unconstrained nonlinear optimization problem. The training process is typically conducted by minimizing an error function $E(w)$, which is defined as the sum of the squared differences between the actual output of the feedforward neural network (FNN), denoted as o_j^h, and the desired output, represented by t_j^h. This function is mathematically expressed as follows:

$$E(w) = \frac{1}{2}\sum_{h=1}^{H}\sum_{j=1}^{N}(o_j^h - t_j^h)^2 = \sum_{h=1}^{H} E_h \tag{1.1}$$

where $w \in \mathbb{R}^n$ represents the vector of network weights and h denotes the number of patterns used in the training dataset [1].

Training and learning in neural networks inherently involve solving non-convex optimization problems. Many methods have been suggested for optimizing deep neural networks, with stochastic gradient descent (SGD) [4] being one of the most commonly used techniques. In theory, SGD can be applied to nonconvex optimization problems within deep-learning frameworks [5].

Among the iterative methods employed for efficiently training neural networks in scientific and engineering applications, the conjugate gradient (CG) method stands out because of its simplicity and minimal memory requirements [6,7,8]. The fundamental principle of CG methods is the generation of a sequence $\{x_k\}$, starting from an initial guess $x_0 \in \mathbb{R}^n$ and iteratively updating it using the recurrence relation

$$x_{k+1} = x_k + \alpha_k d_k, \tag{1.2}$$

where α_k represents the step size, which is typically determined using a line search procedure [9]. In practice, an exact line search is often computationally expensive, which makes inexact line search methods more practical [10]. The step size α_k is obtained by minimizing a quadratic function f along the search direction $x_k + \alpha d_k$, and is given by the formula [11]:

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T Q d_k}, \tag{1.3}$$

where $g_k = \nabla f(x_k)$ denotes the gradient of the objective function at iteration k, and Q is the Hessian matrix [11]. The search direction d_k is computed at the first iteration by $d_0 = -g_0$ and subsequently updated using the rule:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \tag{1.4}$$

where β_k is a scalar called the update (CG) parameter. Different CG methods mainly correspond to different choices of the CG parameter [12]. Although CG methods are equivalent in the linear case, that is, when the objective function is strictly convex and the step length is computed exactly, their numerical behavior for general functions may be quite different [13].

One of the essential CG methods was proposed by Hestenes and Stiefel [14] (HS) with the following CG parameter:

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \tag{1.5}$$

where $y_k = g_{k+1} - g_k$. Although the HS method is numerically efficient [13] and satisfies the conjugacy condition $d_{k+1}^T y_k = 0, \forall k \geq 0$, independent of the line search and objective function convexity, the method lacks the following descent property:

$$g_k^T d_k < 0, \quad \forall k \geq 0. \tag{1.6}$$

This motivated researchers to consider modifications of the HS method with the descent property (6), or its stronger version called the sufficient descent property, that is,

$$g_k^T d_k \leq -c\|g_k\|^2, \quad \forall k \geq 0, \tag{1.7}$$

where c is a positive constant, and $\|\cdot\|$ stands for the Euclidean norm. In this context, Hager and Zhang [15] (HZ) proposed the following CG parameter:

$$\beta_k^{HZ} = \beta_k^{HS} - 2\frac{\|y_k\|^2}{d_k^T y_k}\frac{g_{k+1}^T d_k}{d_k^T y_k}, \tag{1.8}$$

which guarantees (1.7) with $c = 7/8$. Subsequently, the following extension of β_k^{HZ} was proposed [12]:

$$\beta_k^{\theta} = \beta_k^{HS} - \theta_k\frac{\|y_k\|^2}{d_k^T y_k}\frac{g_{k+1}^T d_k}{d_k^T y_k}, \tag{1.9}$$

where θ_k is a non-negative parameter. It is important that if $\theta_k \geq \theta_0 > 1/4$, then the sufficient descent condition (1.7) is achieved with $c = 1 - 1/(4\theta_0)$ [16]. Recently, many advanced CG methods proposed in the literature have been modifications or enhancements of classical CG methods, aimed at improving their efficiency and robustness in practical applications [17,18,19,20].

In recent years, several important developments in conjugate gradient algorithms have further strengthened the theoretical foundation and practical performance of CG-based optimization. Comprehensive surveys such as the work of Hager and Zhang [21] have outlined the evolution of nonlinear CG methods and highlighted key convergence principles. More recent contributions have introduced hybrid and modified CG formulations that enhance robustness and efficiency for large-scale unconstrained optimization [22,23,24,25]. In parallel, several studies have demonstrated the effectiveness of improved CG techniques in machine learning and signal-processing applications, including neural network training, heart disease prediction, image restoration, and ECG signal classification [26,27,28]. Additionally, recent advances in quasi-Newton structures [29,30,31] provide further motivation for developing optimization schemes that balance low memory requirements with strong convergence guarantees. These studies collectively underscore the growing importance of designing CG variants that are theoretically sound, computationally efficient, and well-suited for modern large-scale applications.

Building upon existing CG methods, the proposed approach introduce a new CG update formula that combines efficiency and robustness, particularly under non-convex error surfaces common in neural network training.

The key contributions of this study are summarized as follows:

- We propose a new CG formula that guarantees descent and improves the global convergence without imposing restrictive conditions.
- Theoretical analysis confirms the descent and convergence properties rigorously.
- We empirically validate the method on multiple test functions and was used to train feedforward neural networks, showing significant performance gains.
- Our approach is especially suitable for large-scale optimization, with lower memory requirements compared to second-order methods.

The structure of this paper is as follows. Section 2 presents the new method and its algorithm. Section 3 describes the descent and global convergence properties of the proposed method. Finally, Section 4 provides numerical results to illustrate the performance and practical benefits of the method for solving optimization problems and training neural networks.

2. Research Methodology of the Proposed Method and Algorithm

In this section, we introduce the proposed CG method for solving unconstrained optimization problems and training neural networks.

2.1. Theoretical Derivation of the Proposed Method

In [32], a series of conjugate gradient optimization algorithms was presented, forming the foundation of this study. Researchers have proposed the selection of β_k as follows:

$$\beta_k = \frac{g_{k+1}^T Q s_k}{d_k^T Q s_k}, \tag{2.1}$$

where Q represents the Hessian matrix, ensuring that the conjugacy condition is satisfied:

$$d_{k+1}^T Q d_k = 0. \tag{2.2}$$

To enhance this method, we introduce a unified quasi-Newton (QN) equation that includes both the standard QN equation and those presented in [33]:

$$s_k^T Q(x_k) s_k = s_k^T y_k + \omega_k \left[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k\right], \tag{2.3}$$

The motivation for modifying the QN equation is to integrate the function values within the gradient difference vector. Using equation (2.3) and applying further transformations, we derive:

$$s_k^T Q(x_k) s_k = \frac{(s_k^T g_k)^2}{s_k^T y_k + \omega_k \left[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k\right]}, \tag{2.4}$$

From this, we obtain:

$$d_k^T Q(x_k) s_k = \frac{\alpha_k (d_k^T g_k)^2}{s_k^T y_k + \omega_k \left[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k\right]}, \tag{2.5}$$

Substituting equation (2.5) into (2.1), we get:

$$\beta_k = \frac{g_{k+1}^T y_k}{\frac{\alpha_k (d_k^T g_k)^2}{s_k^T y_k + \omega_k [2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k]}}, \tag{2.6}$$

By applying an exact line search ($g_{k+1}^T g_k = 0$) to Equation (2.6), we derive:

$$\beta_k = \frac{\|g_{k+1}\|^2}{\frac{\alpha_k (d_k^T g_k)^2}{s_k^T y_k + \omega_k [2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k]}}, \tag{2.7}$$

where $\omega_k \in \{0, 1, 2, 3\}$.
We designate this new method as **New**, with specific variations:

- New1 when $\omega_k = 0$,
- New2 when $\omega_k = 1$,
- New3 when $\omega_k = 2$,
- New4 when $\omega_k = 3$.

2.2. Algorithm Implementation

The implementation steps of the proposed method for solving optimization problems are presented in the following algorithm:

Algorithm: (optimization problems)

1. **Initialization:** Given an initial point $x_0 \in \mathbb{R}^n$, parameters $0 < \delta < \sigma < 1$, and $\varepsilon > 0$. Set $d_0 = -g_0$, and $k = 0$.
2. If $\|g_k\| \leq \varepsilon$ then stop.
3. Compute the step size α_k by the weak Wolfe line search.
4. Determine the next iteration by (1.2).
5. Compute d_{k+1} using (1.4) and choose an appropriate conjugate parameter β_k by (2.7).
6. Set $k := k + 1$ and go to Step 2.

Regarding completeness, the algorithm terminates once the gradient norm satisfies $\|g_k\| \leq \varepsilon$, ensuring convergence to a stationary point. While global optimality cannot be guaranteed owing to the nonconvex nature of the problems considered, the method is theoretically guaranteed to converge under standard assumptions, such as Lipschitz continuity of the gradient and boundedness of the level set. If these conditions are violated, the termination condition in Step 2 may never be satisfied, and the algorithm may fail to converge within the maximum number of iterations.

3. Descent Condition and Global Convergence

In this section, we analyze the key properties of the proposed CG algorithm, focusing specifically on the descent condition and global convergence.

Theorem 1. Let $\{x_k\}$ and $\{d_k\}$ be the sequences generated using the proposed method. Then, the following condition holds:

$$d_{k+1}^T g_{k+1} < 0 \quad \text{and} \quad d_{k+1}^T g_{k+1} = \beta_k d_k^T g_k. \tag{3.1}$$

Proof: Clearly, if $d_k = -g_k$ then $d_1^T g_1 < 0$. Suppose that $d_k^T g_k < 0$ for any k. From equations (1.4) and (2.7), we obtain:

$$\begin{aligned} d_{k+1}^T g_{k+1} &= -g_{k+1}^T g_{k+1} + \beta_k d_k^T g_{k+1} \\ &= -\beta_k \frac{\alpha_k (d_k^T g_k)^2}{s_k^T y_k + \omega_k \left[2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k\right]} + \beta_k d_k^T g_{k+1} \\ &= \beta_k \left[d_k^T g_{k+1} - d_k^T Q s_k\right], \end{aligned} \tag{3.2}$$

Using equations (2.6) and (3.2), we further obtain:

$$d_{k+1}^T g_{k+1} = \beta_k d_k^T g_k, \tag{3.3}$$

It is clear that $d_k^T g_k < 0$, it follows that:

$$d_{k+1}^T g_{k+1} < 0. \tag{3.4}$$

The proof is complete.

Although Equation (3.4) guarantees the theoretical descent property of the proposed method under ideal conditions, it is important to distinguish this from the empirical convergence behavior observed in practical applications. In real-world neural network training, the presence of noisy gradients, non-convex error surfaces, and inexact line searches may cause deviations from strict descent in some iterations.

Nevertheless, on average, the method retains sufficient descent to ensure progress toward a stationary point.

To establish the global convergence of the proposed CG method, we introduce the following assumptions.

Assumptions 1.

1. The level set $S = \{x : x \in \mathbb{R}^n, E(x) \leq E(x_0)\}$ is bounded, i.e. $\exists B > 0$ such that
$$\|x\| \leq B, \quad \forall x \in S. \tag{3.5}$$
2. In a neighborhood $\Omega \subseteq S$, E is differentiable and its gradient g is Lipschitz continuous, i.e. $\exists L > 0$ such that
$$\|g(x) - g(x_i)\| \leq L\|x - x_i\|, \quad \forall x, x_i \in \Omega. \tag{3.6}$$
3. From Assumptions (1) and (2), $\exists M > 0$ such that
$$\|g(x)\| \leq M, \quad \forall x \in S. \tag{3.7}$$

Lemma 1. Let all Assumption 1 hold. Consider any iteration method in which α_k is obtained using Wolfe line search. Then:
$$\sum_{k=1}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} < \infty. \tag{3.8}$$

See [34].

Theorem 2. Suppose Assumption 1 holds. If the formula for β_k satisfies Equation (19), the following condition is satisfied:
$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{3.9}$$

Proof: We proceed with induction. We assume that Equation (3.9) does not hold. Rewriting Equation (1.4) as $d_{k+1} + g_{k+1} = \beta_k d_k$, and squaring both sides, we obtain:
$$\|d_{k+1}\|^2 + \|g_{k+1}\|^2 + 2d_{k+1}^T g_{k+1} = (\beta_k)^2 \|d_k\|^2. \tag{3.10}$$

By applying Equation (3.3), we derive:
$$\|d_{k+1}\|^2 = \frac{(d_{k+1}^T g_{k+1})^2}{(d_k^T g_k)^2}\|d_k\|^2 - 2d_{k+1}^T g_{k+1} - \|g_{k+1}\|^2. \tag{3.11}$$

Dividing both sides of (3.11) by $(d_{k+1}^T g_{k+1})^2$, we get:
$$\frac{\|d_{k+1}\|^2}{(d_{k+1}^T g_{k+1})^2} = \frac{\|d_k\|^2}{(d_k^T g_k)^2} - \frac{\|g_{k+1}\|^2}{(d_{k+1}^T g_{k+1})^2} - \frac{2}{d_{k+1}^T g_{k+1}}. \tag{3.12}$$

Completing the square leads to:
$$\frac{\|d_{k+1}\|^2}{(d_{k+1}^T g_{k+1})^2} \leq \frac{\|d_k\|^2}{(d_k^T g_k)^2} - \left(\frac{\|g_{k+1}\|}{d_{k+1}^T g_{k+1}} + \frac{1}{\|g_{k+1}\|^2}\right) + \frac{1}{\|g_{k+1}\|^2} \leq \frac{\|d_k\|^2}{(d_k^T g_k)^2} + \frac{1}{\|g_{k+1}\|^2}. \tag{3.13}$$

Thus, we obtain the following inequality:
$$\frac{\|d_{k+1}\|^2}{(d_{k+1}^T g_{k+1})^2} \leq \sum_{i=1}^{k+1} \frac{1}{\|g_i\|^2}. \tag{3.14}$$

Now, suppose that there exists a constant $c_1 > 0$ such that $\|g_k\| \geq c_1$ for all $k \in \mathbb{N}$. Then,

$$\frac{\|d_{k+1}\|^2}{(d_{k+1}^T g_{k+1})^2} < \frac{k+1}{c_1^2}. \tag{3.15}$$

From our assumption and Equation (3.15), we note that:

$$\sum_{k=1}^{\infty} \frac{(g_k^T d_k)^2}{\|d_k\|^2} = \infty. \tag{3.16}$$

Based on Lemma 1, we conclude that:

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{3.17}$$

Thus, the proof is complete.

Practically, the global convergence result in Theorem 2 implies that the proposed method is robust for neural network training. As training is framed as a nonconvex optimization task, the ability to ensure that $\lim_{k\to\infty} \inf \|g_k\| = 0$ means that regardless of the starting point, the algorithm generates iterates that approach a stationary point. This supports reliable weight updates and a gradual reduction in training error, making the method suitable for large-scale learning tasks in which convergence guarantees are critical.

4. Numerical Experiments

In this section, we discuss the performance of the proposed CG method in solving optimization problems with a specific application for neural network training.

4.1. Unconstrained Optimization Problems

In this subsection, we present the numerical results obtained by implementing our new CG method with the HS conjugacy condition and search directions given in Equation (1.4). The experiments were conducted using MATLAB on a laptop with the following specifications: Windows 10 operating system, HP computer with an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz (2.81 GHz), and 8 GB of RAM. The test problems were selected from the CUTE collection [35], and additional unconstrained optimization problem sets from [36,37].

The parameters used in the implementation were set to $\delta = 0.01$ and $\sigma = 0.3$. Wolfe parameters δ and σ control the sufficient decrease and curvature conditions, respectively. In our experiments, $\delta = 0.01$ ensures adequate descent, while $\sigma = 0.3$ balances the trade-off between speed and stability of convergence. We conducted sensitivity analysis on selected problems by varying δ in $[0.001, 0.1]$ and σ in $[0.1, 0.9]$ and observed that while the method remains stable within these bounds, overly aggressive choices (e.g., very small σ) can lead to longer convergence times or instability due to weak curvature enforcement. These default settings provided a good balance across all the test cases.

A test was considered to have failed to converge if it did not satisfy the stopping criterion $\|g_{k+1}\| \leq 10^{-6}$, in which case, the result was marked as 'NaN.' The occurrence of "NaN" results typically indicates that the method failed to converge due to one of the following: divergence caused by poor search direction, failure to satisfy Wolfe conditions during the line search (resulting in undefined or excessively small step sizes), or the algorithm exceeding the maximum number of iterations without meeting the stopping criterion. These instances highlight practical limitations in handling ill-conditioned or highly nonconvex problems.

To evaluate the performance of each method, we considered several key metrics:

- Number of iterations (NOI)
- Number of function evaluations (NOF)
- CPU time (CPUT) required for convergence

A summary of the numerical results for the optimization test problems is presented in Table 1. To facilitate a more comprehensive comparison, we use the performance profiles proposed by Dolan and Moré [38]. These profiles visually represent the relative efficiency of each algorithm, with the highest curve indicating the best-performing method. Figures 1, 2, and 3 illustrate the effectiveness of the proposed CG algorithm. Each performance profile includes labeled axes for clarity: the x-axis represents the performance ratio τ, and the y-axis indicates the fraction of problems solved within that factor. A legend is included to identify the curves corresponding to each CG variant (e.g., HS, New1, New2). This ensures interpretability and allows for straightforward comparisons across methods.

From Table 1, we compare the performance of the new methods (New 1, New 2, New 3, and New 4) with that of the Hestenes-Stiefel (HS) method using key metrics: NOI, NOF, and CPUT. The results show that the new methods perform significantly better across all metrics.

- The NOI was reduced by approximately 36.64% compared to the HS method, indicating that it requires less effort to reach the solution. This shows that New 1 is better at finding the optimal solution in less time.
- For NOF, New 1 showed a 39.12% improvement over HS, indicating that fewer function evaluations are needed, which makes the optimization process faster and more efficient.
- Regarding CPU time, New 1 is 45.26% faster than HS, meaning it uses less computing time to reach the solution.
- Other new methods, such as New 2, New 3, and New 4, also showed improvements over the HS. For example, New 2 reduced the NOI by 33.48%, New 3 improved by 36.10%, and New 4 showed a 32.94% improvement. These results show that the new methods are much more effective in achieving a faster optimization.

Overall, the new methods, particularly New 2, performed better than the classic HS method in all areas, suggesting that they can solve large optimization problems more efficiently.

4.2. The Proposed Method in Neural Network Training Applications

This subsection presents the numerical results that evaluate the effectiveness of the new CG methods for training neural networks. This study compared the performance of the HS method with that of the proposed new methods. The implementation was carried out using MATLAB (2013a) along with the MATLAB Neural Network Toolbox version 8.1 for the CG computations.

During the experiments, network training proceeded until the mean squared error (MSE) fell below the designated error target of 10^{-6}, effectively minimizing the error function. The maximum number of epochs was set to 1000. Consistent initial weights were used across all tested algorithms, with weights randomly initialized within the range $(0, 1)$.

The tested problems included:

- Input $P = [-1, -1, 2, 2; 0, 5, 0, 5]$
- Target $T = [-1, -1, 1, 1]$

The neural network architecture consisted of a two-layer feedforward network with three neurons in the hidden layer and one output neuron, configured as:

```
net = newff(minmax(p), [3,1], {'tansig','purelin'}, 'traincgp');
```

The hidden layer employed the tangent sigmoid activation function (`tansig`), whereas the output layer used a linear function (`purelin`). To assess the generalization capability and mitigate overfitting, early stopping was optionally enabled in preliminary tests using validation splits; however, because the training error consistently decreased without divergence, no overfitting was observed in the reported runs.

This configuration allowed for a comprehensive evaluation of the capabilities of the methods in optimizing the neural network performance, providing valuable insights into their practical applications. The results of these training methods are summarized in Table 2 and illustrated in Figures 4–8.

Table 1: Summary of numerical results for optimization test problems.

Test Function	N	HS			New1			New2			New3			New4		
		NOI	NOF	CPUT	NOI	NOF	CPUT	NOI	NOF	CPUT	NOI	NOF	CPUT	NOI	NOF	CPUT
'cosine'	500	342	938	0.163	15	67	0.011	18	68	0.011	17	66	0.011	18	69	0.011
'cosine'	1000	NaN	NaN	NaN	17	84	0.017	22	102	0.021	14	70	0.014	18	86	0.019
'cosine'	5000	NaN	NaN	NaN	17	83	0.081	19	85	0.077	16	79	0.073	17	86	0.097
'cosine'	10000	NaN	NaN	NaN	21	88	0.152	18	85	0.162	19	82	0.197	19	84	0.138
'dixmaana'	1500	29	110	0.290	26	122	0.313	29	139	0.332	35	139	0.353	20	102	0.251
'dixmaana'	3000	29	111	0.584	27	135	0.679	33	138	0.736	36	173	0.929	22	115	0.592
'dixmaana'	15000	30	102	2.279	24	118	2.851	28	131	3.586	22	103	2.927	28	137	3.524
'dixmaana'	30000	30	108	5.042	28	141	7.071	22	117	6.034	21	116	5.789	22	129	6.811
'dixmaanb'	1500	24	109	0.512	25	117	0.419	37	165	0.527	24	119	0.396	22	114	0.359
'dixmaanb'	3000	21	120	0.775	11	72	0.484	24	109	0.877	12	77	0.501	21	119	0.759
'dixmaanb'	15000	32	150	4.166	32	133	3.773	24	119	3.768	19	108	2.986	16	94	2.797
'dixmaanb'	30000	29	158	9.580	24	117	6.907	24	112	7.070	26	122	6.952	33	132	7.230
'dixmaanc'	1500	29	119	0.387	14	88	0.263	26	133	0.424	25	105	0.327	10	65	0.207
'dixmaanc'	3000	31	170	1.207	29	146	0.924	34	165	1.002	37	169	1.177	31	147	0.889
'dixmaanc'	3000	31	170	0.966	29	146	0.878	34	165	0.988	37	169	0.927	31	147	0.842
'dixmaanc'	15000	31	169	4.290	22	140	3.839	24	137	4.396	16	103	2.754	25	121	3.186
'dixmaand'	30000	30	144	7.216	32	144	7.569	24	118	5.806	21	117	6.011	32	167	7.536
'dixmaand'	1500	25	110	0.225	24	121	0.277	22	112	0.230	16	89	0.226	26	123	0.286
'dixmaand'	1500	25	110	0.244	24	121	0.297	22	112	0.229	16	89	0.194	26	123	0.300
'dixmaand'	3000	26	149	0.725	22	117	0.601	29	155	0.809	32	147	0.802	33	153	0.846
'dixmaank'	1500	709	1039	4.095	511	1990	6.052	426	1667	5.000	347	1344	4.168	253	1043	3.105
'dixmaank'	3000	326	501	3.385	317	1218	8.555	444	1727	11.979	449	1766	11.993	237	908	6.354
'edensch'	500	41	120	0.087	30	84	0.036	32	84	0.037	33	86	0.036	30	82	0.030
'edensch'	1000	38	129	0.084	33	88	0.062	33	89	0.070	36	98	0.077	38	91	0.059
'edensch'	5000	59	209	0.730	35	92	0.319	37	95	0.336	35	94	0.328	32	88	0.290
'edensch'	10000	66	366	2.535	26	88	0.594	48	205	1.405	38	99	0.668	37	96	0.650
'fletchcr'	1000	49	185	0.051	86	157	0.039	89	149	0.032	97	156	0.036	105	167	0.039
'fletchcr'	5000	134	1207	0.934	97	164	0.136	80	143	0.132	125	185	0.157	101	267	0.216
'fletchcr'	10000	237	1897	2.753	81	147	0.227	118	184	0.289	67	130	0.189	90	150	0.236
'fletchcr'	12000	NaN	NaN	NaN	69	133	0.234	92	159	0.291	70	134	0.242	87	148	0.270
'himmelbg'	500	2	9	0.046	2	9	0.002	2	9	0.002	2	9	0.002	2	9	0.002
'himmelbg'	1000	2	9	0.004	2	9	0.005	2	9	0.004	2	9	0.005	2	9	0.004
'himmelbg'	5000	4	24	0.042	3	21	0.025	3	21	0.028	3	21	0.028	3	21	0.024
'himmelbg'	10000	2	11	0.029	2	11	0.030	2	11	0.030	2	11	0.032	2	11	0.035
'penalty1'	500	NaN	NaN	NaN	24	135	0.387	26	146	0.515	23	130	0.355	26	144	0.389
'penalty1'	1000	NaN	NaN	NaN	25	146	1.136	23	134	1.017	25	150	1.140	27	162	1.185
'penalty1'	4000	88	586	52.119	23	129	10.993	22	123	10.026	23	138	11.903	23	130	11.581
'penalty1'	10000	NaN	NaN	NaN	18	111	30.245	15	96	25.695	18	106	28.396	19	116	30.878
'bdexp'	500	NaN	NaN	NaN	2	7	0.002	2	7	0.002	2	7	0.002	2	7	0.002
'bdexp'	1000	NaN	NaN	NaN	2	7	0.003	2	7	0.003	2	7	0.003	2	7	0.003
'bdexp'	5000	NaN	NaN	NaN	3	19	0.030	3	19	0.033	3	19	0.032	3	19	0.029
'bdexp'	10000	NaN	NaN	NaN	2	9	0.045	2	9	0.040	2	9	0.040	2	9	0.039
'exdenschnf'	500	35	140	0.024	21	85	0.011	20	82	0.010	23	86	0.010	21	82	0.010
'exdenschnf'	1000	33	171	0.033	25	90	0.016	23	89	0.016	25	93	0.017	27	95	0.017
'exdenschnf'	5000	37	156	0.127	30	97	0.079	23	89	0.073	24	81	0.065	28	91	0.077
'exdenschnf'	10000	32	143	0.229	22	92	0.147	26	94	0.151	21	91	0.147	21	87	0.136
'exdenschnb'	500	19	81	0.015	27	132	0.010	23	95	0.007	18	94	0.007	19	81	0.006
'exdenschnb'	1000	30	128	0.012	26	108	0.009	34	147	0.013	33	160	0.014	33	143	0.013
'exdenschnb'	5000	30	144	0.053	23	90	0.035	25	92	0.034	25	118	0.042	21	87	0.034
'exdenschnb'	10000	24	116	0.080	24	120	0.086	29	135	0.095	31	127	0.086	27	108	0.075
'genquartic'	500	22	103	0.019	36	140	0.011	30	132	0.010	37	158	0.012	34	133	0.009
'genquartic'	1000	34	106	0.011	30	124	0.016	25	109	0.013	32	117	0.014	28	100	0.010
'genquartic'	5000	83	296	0.134	19	96	0.040	12	69	0.029	13	85	0.036	16	84	0.035
'genquartic'	10000	80	344	0.275	32	131	0.104	24	111	0.090	23	96	0.080	32	142	0.119
'sine'	500	NaN	NaN	NaN	88	200	0.022	97	212	0.021	49	138	0.018	172	286	0.033
'sine'	1000	NaN	NaN	NaN	17	78	0.015	18	75	0.016	15	71	0.014	20	80	0.016
'sine'	3000	NaN	NaN	NaN	19	87	0.054	20	91	0.053	16	85	0.050	23	99	0.058
'sine'	4000	NaN	NaN	NaN	17	78	0.061	19	85	0.065	18	78	0.059	16	79	0.059
'raydan2'	500	13	104	0.013	14	92	0.008	11	72	0.007	11	72	0.007	11	72	0.006
'raydan2'	1000	14	109	0.015	11	71	0.009	11	71	0.009	11	71	0.010	11	71	0.009
'raydan2'	5000	13	113	0.056	18	116	0.059	13	92	0.050	14	94	0.051	14	101	0.054
'raydan2'	10000	18	119	0.119	19	116	0.114	22	141	0.140	12	77	0.075	11	70	0.075
'diagonal1'	4	28	85	0.009	22	93	0.004	35	143	0.006	16	75	0.003	23	90	0.003
'diagonal1'	10	35	96	0.004	24	68	0.003	26	70	0.003	26	67	0.002	24	65	0.002
'diagonal1'	50	60	134	0.005	91	168	0.007	81	155	0.009	96	174	0.011	86	169	0.011
'diagonal3'	4	24	79	0.013	22	68	0.003	27	86	0.004	22	80	0.004	23	75	0.004
'diagonal3'	10	44	100	0.004	28	60	0.002	33	74	0.003	28	67	0.002	24	63	0.002
'diagonal3'	50	64	123	0.006	91	167	0.012	81	143	0.007	89	160	0.008	113	185	0.011
'diagonal3'	100	77	148	0.010	126	245	0.013	139	234	0.015	129	220	0.012	125	224	0.021
'ie'	10	17	77	0.007	27	102	0.009	13	59	0.005	21	95	0.008	26	107	0.009
'ie'	100	24	108	0.424	16	68	0.266	22	85	0.335	19	83	0.331	25	77	0.323
'ie'	500	20	79	7.973	15	55	6.300	17	79	9.062	17	70	7.389	18	57	5.757
'lin'	10	20	100	0.324	11	69	0.197	11	69	0.200	11	69	0.204	11	69	0.221
'lin'	100	22	105	0.703	11	56	0.373	11	56	0.456	11	56	0.481	11	56	0.401
'lin'	500	19	88	1.609	14	82	1.322	14	82	1.327	14	82	1.358	14	82	1.254
'lin'	1000	25	145	120.651	15	71	60.588	15	71	61.397	15	71	67.118	15	71	53.701
'pen1'	5	NaN	NaN	NaN	548	3308	0.237	567	3323	0.229	693	4190	0.276	634	3626	0.289
'pen1'	10	NaN	NaN	NaN	163	966	0.093	165	968	0.093	105	640	0.062	459	2636	0.248

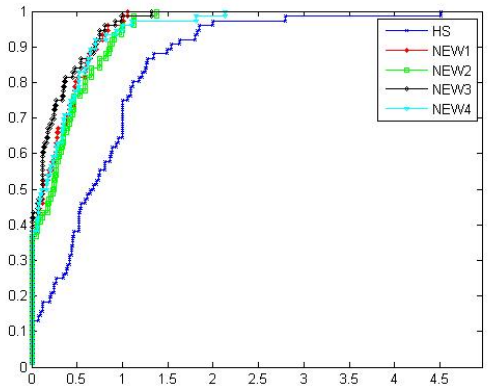

Figure 1: Performance profile in terms of NOI.

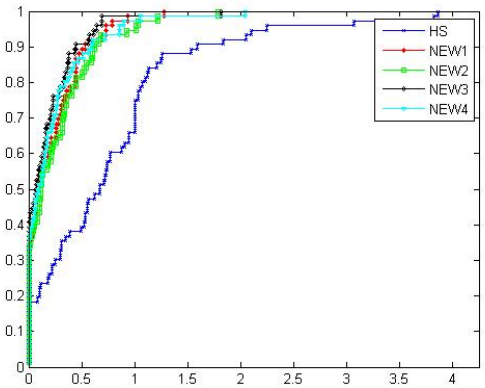

Figure 2: Performance profile in terms of NOF.

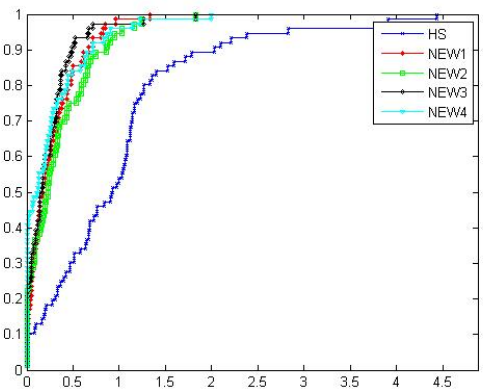

Figure 3: Performance profile in terms of CPUT.

Table 2: Performance Comparison of new methods with HS for Training Neural Networks

Method	Epochs	MSE
HS	1000	8.74e-06
New1	25	1.11e-08
New2	21	1.12e-08
New2	35	1.23e-08
New4	48	1.18e-08

Table 2 presents a comparative analysis of the performance of the proposed CG methods against the classical HS method for training neural networks. The evaluation was based on two key metrics:

- **Epochs:** The number of iterations required for the training process to converge.
- **MSE:** Final error value after training, indicating the accuracy of the method.

From the results in Table 2, it is evident that the proposed new methods significantly outperform the HS method in terms of the convergence speed. The HS method required 1000 epochs to reach an MSE of 8.74e-06, whereas the new methods achieved much lower MSE values in substantially fewer epochs.

- New1 converges in 25 epochs with an MSE of 1.11e-08.
- New2 converges even faster in 21 epochs with an MSE of 1.12e-08.
- Another instance of New2 takes 35 epochs, achieving an MSE of 1.23e-08.
- New4 completes training in 48 epochs with an MSE of 1.18e-08.

These results clearly demonstrate the efficiency of the new CG methods, achieving better accuracy (lower MSE) with significantly fewer epochs than the HS method. This highlights their potential for optimizing neural network training by reducing computational cost and improving convergence speed.

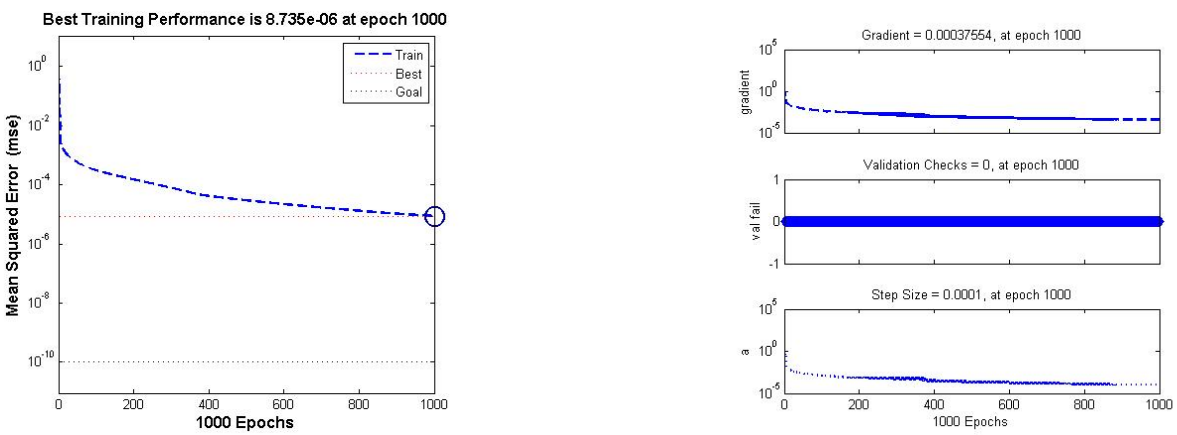

Figure 4: Performance of HS method for training neural networks.

5. Conclusion

In this study, we propose a new CG method for solving large-scale optimization problems. The proposed method satisfies the descent condition, and its global convergence has been proven theoretically. The performance of the new method was evaluated on a set of standard unconstrained optimization test problems, and it demonstrated superior performance compared to the HS method. This superiority was assessed based on key metrics, including NOI, NOF, and CPUT, confirming its efficiency in solving

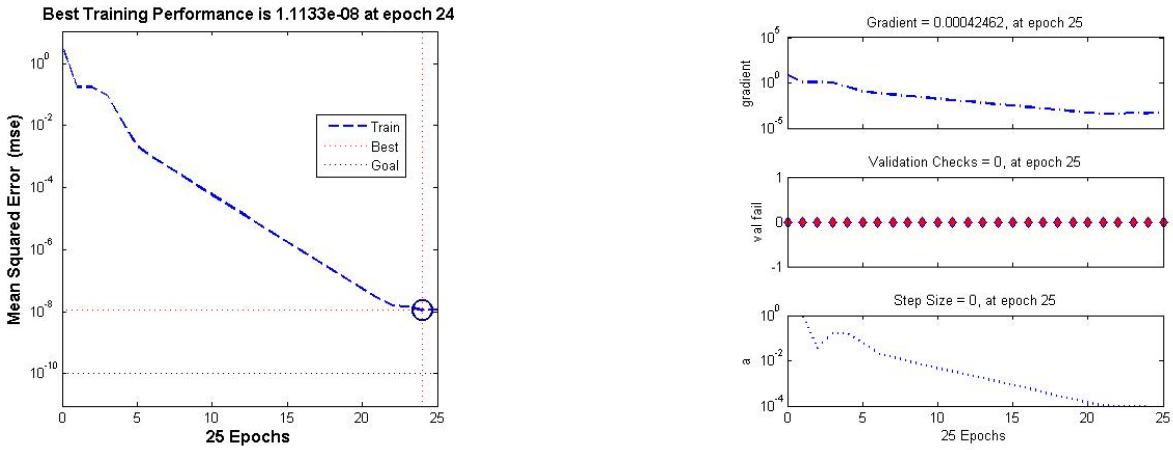

Figure 5: Performance of New1 method for training neural networks.

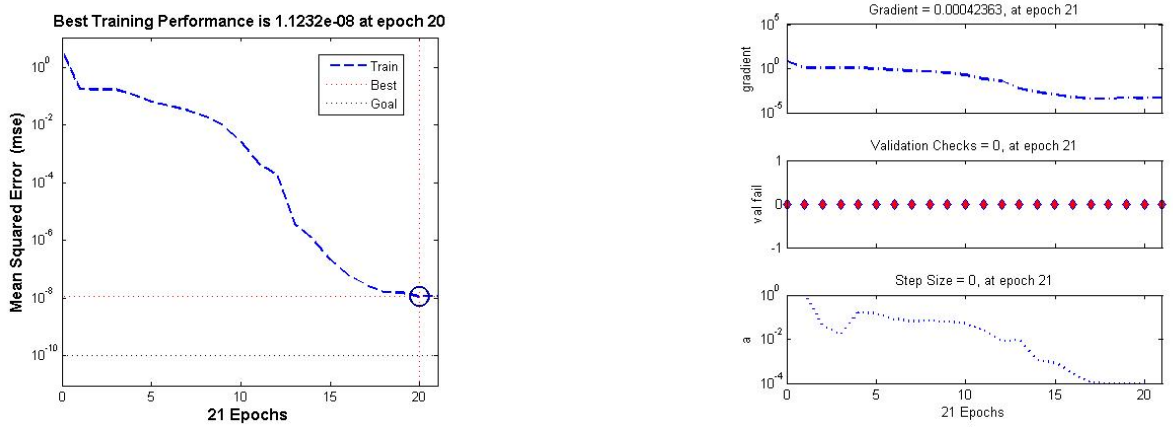

Figure 6: Performance of New1 method for training neural networks.

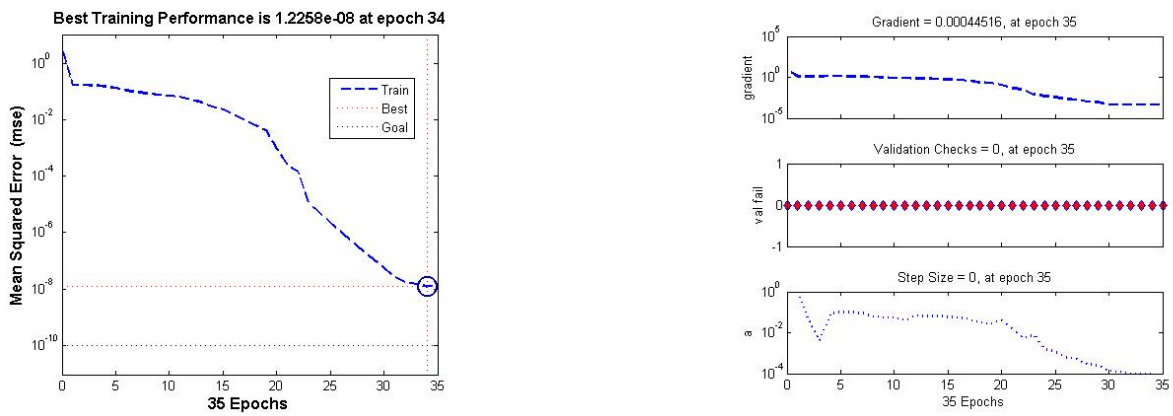

Figure 7: Performance of New1 method for training neural networks.

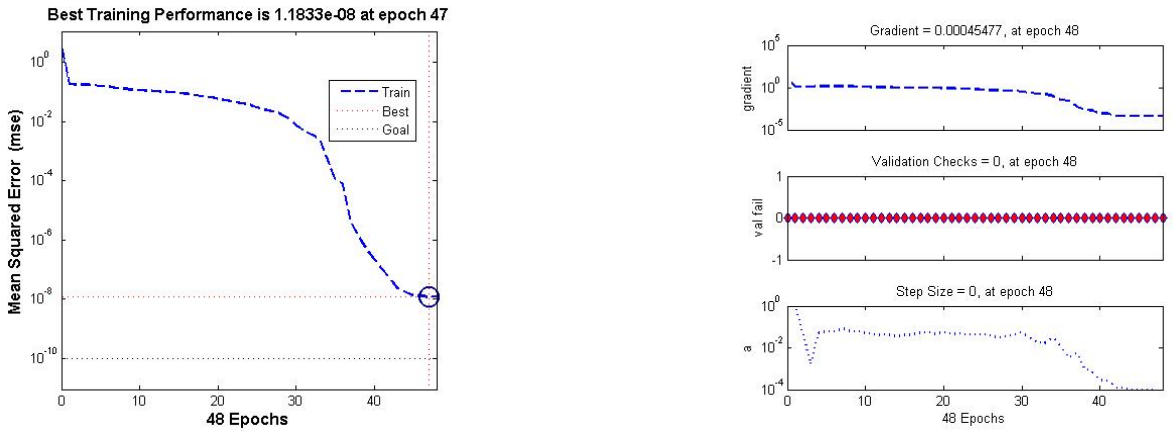

Figure 8: Performance of New1 method for training neural networks.

optimization problems. Furthermore, we applied the new CG method to train the neural networks. Numerical experiments showed that the proposed method outperformed the classical CG approach, achieving faster convergence and improved stability. These results highlight the effectiveness of the new method in both optimization and neural-network training applications, making it a promising approach for future research and practical implementation.

Funding

This research received no external funding.

Ethics Declarations

Ethical Approval: Not applicable.
Informed Consent: Not applicable.
Conflict of Interest: The authors declare no conflict of interest.

References

1. Rumelhart, D.E., Hinton, G.E., & Williams, R.J. (1986). Learning internal representations by error propagation. In D.E. Rumelhart & J. McClelland (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition (pp. 318–362). Cambridge, MA.

2. Guo, Y., Kasihmuddin, M.S.M., Zamri, N.E., Li, J., Romli, N.A., Mansor, M.A., & Ruzai, W.N.A. (2025). Logic mining method via hybrid discrete Hopfield neural network. Computers & Industrial Engineering, 206, 111200. https://doi.org/10.1016/j.cie.2025.111200

3. Romli, N.A., Jamaludin, S.Z.M., Kasihmuddin, M.S.M., Mansor, M.A., & Zamri, N.E. (2024). Modelling logic mining: A log-linear approach. AIP Conference Proceedings, 2895(1), 040002. https://doi.org/10.1063/5.0192155

4. Robbins, H., & Monro, S. (1951). A stochastic approximation method. The Annals of Mathematical Statistics, 22, 400–407.

5. Loizou, N., Vaswani, S., Laradji, I., & Lacoste-Julien, S. (2021). Stochastic Polyak step size for SGD: An adaptive learning rate for fast convergence. In Proceedings of the 24th International Conference on Artificial Intelligence and Statistics (AISTATS), 130, 1–11.

6. Charalambous, C. (1992). Conjugate gradient algorithm for efficient training of artificial neural networks. IEEE Proceedings, 139(3), 301–310.

7. Hassan, B.A., Moghrabi, I.A.R., Ibrahim, A.L., & Jabbar, H.N. (2025). Improved Conjugate Gradient Methods for Unconstrained Minimization Problems and Training Recurrent Neural Network. Engineering Reports, 7: e70019. https://doi.org/10.1002/eng2.70019

8. Ibrahim, A.L., Fathi, B.G., & Abdulrazzaq, M.B. (2025). Improving three-term conjugate gradient methods for training artificial neural networks in accurate heart disease prediction. Neural Computing and Applications. https://doi.org/10.1007/s00521-025-11121-9

9. Salleh, Z., & Alhawarat, A. (2016). An efficient modification of the Hestenes-Stiefel nonlinear conjugate gradient method with restart property. Journal of Inequalities and Applications, 2016(1), 110. https://doi.org/10.1186/s13660-016-1049-5

10. Yuan, G., Wei, Z., & Zhao, Q. (2014). A modified Polak-Ribière-Polyak conjugate gradient algorithm for large-scale optimization problems. IIE Transactions (Institute of Industrial Engineers), 46(4), 397–413. https://doi.org/10.1080/0740817X.2012.726757

11. Nocedal, J., & Wright, S.J. (1999). Numerical Optimization. Springer Series in Operations Research. Springer Verlag, New York.

12. Hager, W., & Zhang, H. (2006). A survey of nonlinear conjugate gradient methods. Pacific Journal of Optimization, 2(1), 35–58.

13. Andrei, N. (2007). Numerical comparison of conjugate gradient algorithms for unconstrained optimization. Studies in Informatics and Control, 16(4), 333–352.

14. Hestenes, M., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49(6), 409–436.

15. Hager, W., & Zhang, H. (2005). A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM Journal on Optimization, 16(1), 170–192.

16. Babaie-Kafaki, S. (2014). On the sufficient descent condition of the Hager-Zhang conjugate gradient methods. 4OR, 12(3), 285–292.

17. Omar, D.H., Ibrahim, A.L., Hassan, M.M., Fathi, B.G., & Sulaiman, D.A. (2024). Enhanced Conjugate Gradient Method for Unconstrained Optimization and its Application in Neural Networks. European Journal of Pure and Applied Mathematics, 17(4), 2692–2705. https://doi.org/10.29020/nybg.ejpam.v17i4.5354

18. Hassan, B.A., & Alashoor, H.A. (2023). On image restoration problems using new conjugate gradient methods. Indonesian Journal of Electrical Engineering and Computer Science, 29(3), 1438–1445.

19. Hassan, B.A., & Sadiq, H.M. (2022). A new formula on the conjugate gradient method for removing impulse noise images. Bulletin of the South Ural State University. Series: Mathematical Modelling, Programming & Computer Software, 15(4), 123–130.

20. Hassan, B.A., & Alashoor, H.A. (2022). A New Type Coefficient Conjugate on the Gradient Methods for Impulse Noise Removal in Images. European Journal of Pure and Applied Mathematics, 15(4), 2043–2053.

21. Hager, W.W., and Zhang, H. (2006). A survey of nonlinear conjugate gradient methods. *Pacific Journal of Optimization*, 2(1), 35–58.

22. Mehamdia, A.E., Chaib, Y., and Tahar, B. (2025). A new hybrid conjugate gradient method as a convex combination of methods. *Bulletin of the Transilvania University of Brasov Series III: Mathematics and Computer Science*, 5, 179–192.

23. Mehamdia, A.E., and Chaib, Y. (2024). Two modified conjugate gradient methods for unconstrained optimization. *Optimization Methods and Software*, 40(2).

24. Mehamdia, A.E., and Chaib, Y. (2024). Improved conjugate gradient methods and application to nonparametric estimation. *Applicationes Mathematicae*, 51(2).

25. Mehamdia, A.E., and Chaib, Y. (2025). A modified conjugate gradient method for solving unconstrained optimization with application in conditional mode regression. *International Journal of Computer Mathematics*.

26. Ibrahim, A.L., Fathi, B.G., and Abdulrazzaq, M.B. (2025). Improving three-term conjugate gradient methods for training artificial neural networks in accurate heart disease prediction. *Neural Computing and Applications*, 37, 10381–10405. doi: 10.1007/s00521-025-11121-9.

27. Ibrahim, A.L., Fathi, B.G., and Abdulrazzaq, M.B. (2025). Conjugate gradient techniques: Enhancing optimization efficiency for large-scale problems and image restoration. *Numerical Algebra, Control and Optimization*, 15(4), 1151–1175. doi: 10.3934/naco.2025008.

28. Ibrahim, A.L., Fathi, B.G., and Abdulrazzaq, M.B. (2025). An improved three-term conjugate gradient approach for deep neural network training in ECG signal classification. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 14, 141. doi: 10.1007/s13721-025-00639-6.

29. Hassan, B.A., and Ayoob, A.R. (2022). On the New Quasi-Newton Equation for Unconstrained Optimization. In: *8th International Engineering Conference (IEC 2022): Towards Engineering Innovations and Sustainability*.

30. Hassan, B.A., and Ayoob, A.R. (2021). An Adaptive Quasi-Newton Equation for Unconstrained Optimization. In: *Proceedings of the 2nd Information Technology to Enhance E-Learning and other Application Conference (IT-ELA 2021)*. doi: 10.1109/IT-ELA52201.2021.9773580.

31. Hassan, B.A., and Sulaiman, R.M. (2021). Using a New Type Quasi-Newton Equation for Unconstrained Optimization. In: *Proceedings of the 7th International Engineering Conference "Research and Innovation Amid Global Pandemic" (IEC 2021)*. doi: 10.1109/IEC52205.2021.9476089.

32. Hassan, B.A., & Sadiq, H.M. (2022). A new formula on the conjugate gradient method for removing impulse noise images. Bulletin of the South Ural State University. Series: Mathematical Modelling, Programming & Computer Software (Bulletin SUSU MMCS), 15(4), 123–130.

33. Ataee Tarzanagh, D., & Peyghami, M.R. (2015). A new regularized limited memory BFGS-type method based on modified secant conditions for unconstrained optimization problems. Journal of Global Optimization, 63, 709–728. https://doi.org/10.1007/s10898-015-0310-7

34. Zoutendijk, G. (1970). Nonlinear programming computational methods. In J. Abadie (Ed.), Integer Nonlinear Programming (pp. 37–86). North-Holland, Amsterdam.

35. Gould, N.I.M., Orban, D., & Toint, P.L. (2003). CUTEr and sifdec: A constrained and unconstrained testing environment, revisited. ACM Transactions on Mathematical Software, 29, 373–394. https://doi.org/10.1145/962437.962439

36. Moré, J.J., Garbow, B.S., & Hillstrom, K.E. (1981). Testing unconstrained optimization software. ACM Transactions on Mathematical Software, 7, 17–41. https://doi.org/10.1145/355934.355936

37. Andrei, N. (2008). An unconstrained optimization test functions collection. Advances in Modeling and Optimization, 10, 147–161.

38. Dolan, E.D., & Moré, J.J. (2002). Benchmarking optimization software with performance profiles. Mathematical Programming, 91, 201–213. https://doi.org/10.1007/s101070100263

[1] *Department of Mathematics, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq*

[2] *Department of Mathematics, College of Science, University of Zakho, Zakho, Iraq*

[3] *Department of Mathematics, College of Computer Science and Mathematics, University of Mosul, Mosul, Iraq*