Solving Unconstrained Minimization Problems and Training Neural Networks via Enhanced Conjugate Gradient Algorithms

Bassim A. Hassan[1], Issam A. R. Moghrabi[2], Talal M. Alharbi[3,*], Alaa Luqman Ibrahim[4]

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

[2]Department of Information Systems and Technology, Kuwait Technical College, Kuwait City, Kuwait

[3]Department of Mathematics, College of Science, Buraydah, Qassim University, Saudi Arabia

[4]Department of Mathematics, College of Science, University of Zakho, Zakho, Kurdistan Region, Iraq

Emails: basimah@uomosul.edu.iq; i.moghrabi@ktech.edu.kw; ta.alharbi@qu.edu.sa; alaa.ibrahim@uoz.edu.krd

Abstract

Artificial neural networks have become a cornerstone of modern artificial intelligence, powering progress in a wide range of fields. Their effective training heavily depends on techniques from unconstrained optimization, with iterative methods based on gradients being especially common. This study presents a new variant of the conjugate gradient method tailored specifically for unconstrained optimization tasks. The method is carefully designed to meet the sufficient descent condition and ensures global convergence. Comprehensive numerical testing highlights its advantages over traditional conjugate gradient techniques, showing improved performance in terms of iteration counts, function evaluations, and overall computational time across a variety of problem sizes. Additionally, this new approach has been successfully used to improve neural network training. Experimental results show faster convergence and better accuracy, with fewer training iterations and reduced mean squared error compared to standard methods. Overall, this work offers a meaningful contribution to optimization strategies in neural network training, displaying the method is potential to tackle the complex optimization problems often encountered in machine learning.

Keywords: Optimization; Conjugate Gradient; Quasi-Newton; conjugacy condition; Neural Networks

1. Introduction

Conjugate gradient (CG) methods have attracted significant attention as methods implemented for solving large-scale unconstrained minimization problems. Their appeal lies in their straightforward implementation and robust global convergence properties. In contrast to Newton and quasi-Newton methods, CG algorithms exhibit low memory requirements, as they do not necessitate the storage of matrices. This characteristic renders them particularly suitable for applications in diverse fields such as the reformulation of radial magnetic resonance (MR) images [1], portfolio decisions [2], [3], [4], motion control issues [2], contractive sensing and image quality restoration [1], [5], [6], and the training of neural networks [7], [8].

Generally, large-scale unconstrained optimization problems are formulated as:

$$\text{minimize } f(x), \tag{1}$$

where $f: R^n \to R$ is a continuously differentiable function, and R^n denotes an n-dimensional Euclidean space. The term "unconstrained" refers to vector x as being any number in the whole n-dimensional Euclidean space, that is, no limit, restriction, or constraint on variable values. The term "large-scale" is particularly used when the dimension n—the number of variables—is very large, usually between thousands and hundreds of thousands or even millions and more. This massive size poses daunting computational issues since common algorithms based

DOI: https://doi.org/10.54216/IJNS.270120

on manipulating large matrices become astronomically costly both in terms of memory and in terms of computation. The following figure depicts a graphical abstract to the work presented in this paper:

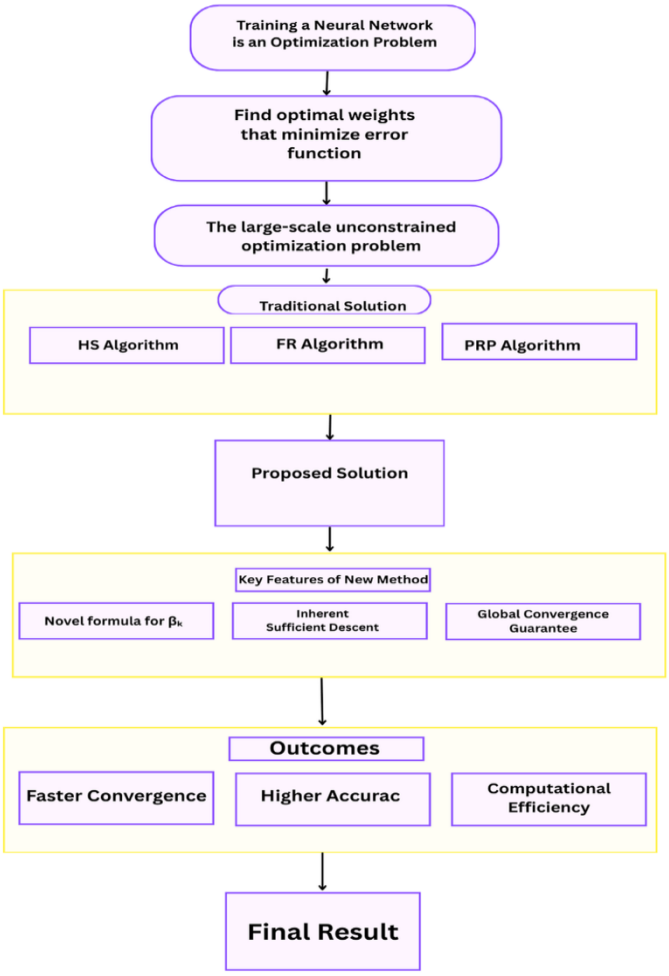

Figure 1. A graphical abstract to the work presented in this paper

Among the various methods for tackling problem (1), nonlinear CG methods stand out as well established and effective. These iterative methods proceed with an initial guess $x_0 \in R^n$ and generate a sequence of iterates $\{x_k\}$ according to the following:

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

where $\alpha_k > 0$ represents the step size determined by some line search mechanism, which can be either exact or inexact. However, in practice, inexact line search strategies are more frequently employed due to their efficiency. Two prominent inexact line search conditions are commonly utilized, referred to as Wolfe line search rules. The weak Wolfe conditions require the step size α_k to satisfy:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{3}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k, \tag{4}$$

while the strong Wolfe conditions are given by:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{5}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \leq -\sigma g_k^T d_k, \tag{6}$$

where $g_k = g(x_k) = \nabla f(x_k)$ is a gradient f at point x_k,

DOI: https://doi.org/10.54216/IJNS.270120

denotes the gradient of f at the point x_k, and the scalars δ and σ are selected to satisfy $0 < \delta < \sigma < 1$. The α_k represents the minimized value of the quadratic function f, along $x_k + \alpha d_k$, and is given by the formula [9]:

$$\alpha_k = -\frac{g_k^T d_k}{d_k^T G d_k}. \quad (7)$$

A fundamental requirement for any effective CG method is the downhill property, which ensures that the search direction leads toward a minimizer. This condition is expressed as:

$$g_k^T d_k < 0, \quad (8)$$

Building upon this, Al-Baali [10] introduced a more stringent condition known as the sufficient descent condition, which plays a crucial role in establishing the convergence of CG methods.

Definition 1. Let $f: R^n \to R$ be a continuously differentiable function. A search direction d_k is said to satisfy the sufficient descent condition if there exists a constant $C > 0$ such that:

$$g_k^T d_k \leq -C\|g_k\|^2, \forall k. \quad (9)$$

The search directions in CG methods are typically generated using the following recursive formula:

$$d_{k+1} = -g_{k+1} + \beta_k d_k, \quad (10)$$

where $g_k = \nabla f(x_k)$ and β_k is a scalar parameter that distinguishes the different CG methods. Much of the published research on CG methods explore numerous strategies for choosing the scalar β_k. Prominent among these are the classical formulas proposed by Hestenes and Stiefel (HS) [11], Fletcher and Reeves (FR) [12], Polak, Ribière, and Polyak (PRP) [13], Dai and Yuan (DY) [9], and the Alaa et al. method [14]. The formulas for β_k in these methods are given by:

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k} \quad (11)$$

$$\beta_k^{FR} = \frac{g_{k+1}^T g_k}{g_k^T g_k} \quad (12)$$

$$\beta_k^{PR} = \frac{g_{k+1}^T y_k}{g_k^T g_k} \quad (13)$$

$$\beta_k^{DY} = \frac{g_{k+1}^T g_k}{d_k^T y_k} \quad (14)$$

$$\beta_k = \frac{g_{k+1}^T y_k}{g_k^T g_k} - t\frac{g_{k+1}^T v_k}{g_k^T g_k}, t > 0. \quad (15)$$

Numerous publications have indicated that among these classical CG methods, PRP, HS, and LS often exhibit good computational performance, while FR, CD, and DY tend to possess stronger global convergence properties. Consequently, recent research efforts have focused on developing new CG algorithms that can effectively balance both robust numerical behavior and strong convergence guarantees. This has led to the emergence of various modifications and extensions, including modified CG methods [15], [16], [17], spectral CG methods [2], and three-term CG methods [25,26], among others.

Motivated by the aforementioned contributions, this paper introduces a novel CG method inspired by the structure presented in [18], intending to achieve improved numerical performance. The key contributions of this paper are as follows:

(1) A new CG method for solving unconstrained optimization problems is proposed, based on the structure of [18]. The search vector generated by the proposed method inherently satisfies the sufficient downhill trait without requiring any rigorous line search conditions. (3) The global convergence of the derived method is established under the weak Wolfe linear search stipulations [19], [20]. (4) The computational efficiency of the new method is evaluated and compared with existing methods using a set of standard benchmark test problems. (5) The practical applicability of the proposed method is demonstrated by applying it to the training of neural networks, displaying its ability to enhance training performance and reduce the error function.

The remainder of this paper is organized as follows: Section 2 details the derivation of the modified CG method and presents the algorithm. Section 3 establishes sufficient downhill property and provides the convergence analysis of the proposed method under the weak Wolfe line search constraints. Section 4 presents the numerical outcomes conducted to assess the computational performance of the new method in comparison to other existing CG methods. Section 5 demonstrates the application of the developed method to the problem of training neural networks. Finally, Section 6 concludes the paper with a summary of the findings

DOI: https://doi.org/10.54216/IJNS.270120

2. Derivation of the new method

In [18], a set of unified quasi-Newton (QN) equations was proposed, anticipated to offer improved accuracy by leveraging more available information about the function and gradient values, readily computed. These equations are given by:

$$s_k^T Q(x_k) s_k = s_k^T y_k + \omega_k [2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k], \quad (16)$$

where $\omega_k \in [0,1,2,3]$. The rationale behind these unified QN equations is that they incorporate more information regarding the objective function's curvature by considering both function and gradient values. Equation (16) implies:

$$s_k^T Q(x_k) s_k = \frac{[s_k^T g_k]^2}{s_k^T y_k + \omega_k [2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k]}. \quad (17)$$

Perry introduced one of the earliest modifications to the conjugacy condition [28], expressed as:

$$d_{k+1}^T y_k = -g_{k+1}^T s_k. \quad (18)$$

In general, the gradient difference y_k is defined as $g_{k+1} - g_k$. Multiplying this by the search direction s_k^T, we obtain:

$$s_k^T Q(x_k) s_k = s_k^T g_{k+1} - s_k^T g_k, \quad (19)$$

By relating equations (17), (18), and (19) to each other, we arrive at:

$$\beta_k = \frac{g_{k+1}^T y_k}{s_k^T y_k} - \frac{s_k^T g_k}{s_k^T y_k} - \frac{[s_k^T g_k]^2}{s_k^T y_k [s_k^T y_k + \omega_k (2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k)]}, \quad (20)$$

The new method is named as β_k^{New}(The New1 to New4 depend on ω_k).

ALGORITHM STEPS

Step 1 : Given an initial point $x_0 \in R^n$

Step 2 : Set , $k = 0$, $d_0 = -g_0$. If $\|g_k\| = 0$ (where $\epsilon > 0$ is a predefined tolerance), then terminate, otherwise jump to Step 3.

Step 3 : Find the step size α_k so that conditions (3) and (4) are satisfied.

Step 4 : Set $x_{k+1} = x_k + \alpha_k d_k$.

Step 5 : Determine g_{k+1}, if $\|g_{k+1}\| \leq 10^{-5}$ halt, else jump to Step 6.

Step 6 : Determine d_{k+1} using (10) and β_k^{New} from (20).

Step 7 : Update $k = k + 1$ and continue from Step 3.

3. Convergence analysis

In this section, we establish global convergence of the new method using the assumptions below.

Assumption 1. The level set $\mathcal{B} := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ has a bound, where x_0 is the initial iterate.

Assumption 2. In a given neighborhood $\mathcal{L}$ of $\mathcal{B}$, the gradient vector of the objective function f has Lipschitz continuity. Thus, for a scalar $L > 0$, we have, for all x,

$$\|g(x) - g(v)\| \leq L\|x - v\|, v \in \mathcal{L}. \quad (21)$$

Assumption 1 necessitates that a constant $T > 0$ exists such that:

$$\|x\| \leq T, \forall x \in \mathcal{B}. \quad (22)$$

Collectively, from Assumptions 1 and 2, we can obtain a positive constant F, such that:

$$\|g(x)\| \leq F, \forall x \in \mathcal{B}. \quad (23)$$

Next, we will present the sufficient descent condition for the New method.

Theorem 1. If d_{k+1} is created using a New Method algorithm, then the sufficient descent condition $d_{k+1}^T g_{k+1} \leq -c\|g_{k+1}\|^2$ holds for some constant $c > 0$.

DOI: https://doi.org/10.54216/IJNS.270120

Proof : Since $d_0 = -g_0$, the sufficient descent condition trivially holds for $k = 0$ with $C = 1.$. Let's assume that the $g_k^T d_k < 0$ for all $k \in n$. When we multiply (9) by g_{k+1}^T, , we get:

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + \left[\frac{g_{k+1}^T y_k}{s_k^T y_k} - \frac{s_k^T g_k}{s_k^T y_k} - \frac{[s_k^T g_k]^2}{s_k^T y_k [s_k^T y_k + \omega_k (2(f_k - f_{k+1}) + (g_k + g_{k+1})^T s_k)]}\right] s_k^T g_{k+1}. \quad (24)$$

We have the following from (24) and (20):

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + \left[\frac{g_{k+1}^T y_k}{s_k^T y_k} - \frac{s_k^T g_{k+1}}{s_k^T y_k}\right] s_k^T g_{k+1}. \quad (25)$$

We are given the Lipschitz condition on the gradient:

$$y_k^T g_{k+1} \leq L s_k^T g_{k+1}, \quad (26)$$

From equation (24) and (25), we obtained:

$$d_{k+1}^T g_{k+1} \leq -\|g_{k+1}\|^2 + \left[L\frac{s_k^T g_{k+1}}{s_k^T y_k} - \frac{s_k^T g_{k+1}}{s_k^T y_k}\right] s_k^T g_{k+1}. \quad (27)$$

Taking advantage of the inequality:

$$d_{k+1}^T g_{k+1} \leq -\|g_{k+1}\|^2 + [L-1]\frac{(s_k^T g_{k+1})^2}{s_k^T y_k}, \quad (28)$$

we get:

$$d_{k+1}^T g_{k+1} \leq -\|g_{k+1}\|^2 < 0. \quad (29)$$

This completes the proof.

The next lemma is proved in [21] for any conjugate gradient algorithm using the Wolfe conditions.

Lemma 1: Consider any conjugate gradient technique (2), where d_{k+1} is assumed to be a descent direction defined in (10) and α_k is picked by a strong Wolfe line search that satisfies (3) and (4). It is also assumed that assumptions (1) and (2) hold. If:

$$\sum_{k>1} \frac{1}{\|d_{k+1}\|^2} = \infty , \quad (30)$$

then :

$$\lim_{k\to\infty} (inf\|g_{k+1}\|) = 0. \quad (31)$$

The proof can be found in [22].

The convergence property result of the new algorithm is shown in the following theorem.

Theorem 2: Assume Assumptions 1 and 2 are correct. If f is a uniformly convex function, that is, if a constant $\mu > 0$ exists such that:

$$(\nabla f(z) - \nabla f(u))^T \geq \mu\|z-u\|^2, \forall z, u \in R^n, \quad (32)$$

then :

$$\lim_{k\to\infty} (inf\|g_{k+1}\|) = 0. \quad (31)$$

Proof: It follows from the definitions of β_k (20) and (15) that:

$$|\beta_k| = \left|\frac{g_{k+1}^T y_k}{s_k^T y_k} - \frac{s_k^T g_{k+1}}{s_k^T y_k}\right|, \quad (34)$$

For which $\|y_k\| \leq L\|s_k\|$ is obtained by using (19), while $s_k^T y_k \geq \mu\|s_k\|^2$ is obtained by using (30). When we apply the Cauchy inequality to the previous inequalities, then (34), we obtain:

$$|\beta_k| \leq \frac{L\|s_k\|\|g_{k+1}\|}{\mu\|s_k\|^2} + \frac{\|s_k\|\|g_{k+1}\|}{\mu\|s_k\|^2} \quad (35)$$

$$\leq \left(\frac{L+1}{\mu}\right)\frac{¥}{\|s_k\|} \quad (36)$$

As a result of utilizing (36) in (6), we get:

DOI: https://doi.org/10.54216/IJNS.270120

$$\|d_{k+1}\| \leq \|g_{k+1}\| + |\beta_k|\|s_k\| \leq ¥ + \left(\frac{L+1}{\mu}\right)\frac{¥}{\|s_k\|}\|s_k\| \leq ¥ + \frac{¥L+¥}{\mu} \leq \frac{¥(\mu+L+1)}{\mu}, \quad (37)$$

demonstrating that (31) is correct.

4. Numerical Results

This section presents the numerical results obtained to evaluate the effectiveness of the proposed algorithm in solving unconstrained optimization problems. The test problems utilized in this study were sourced from the CUTE library [23], [24] and other collections of unconstrained optimization problems [25], [26], encompassing a range of dimensions.

All algorithms were implemented using MATLAB 2013b and executed on an HP personal laptop. To assess the performance of our proposed method, we compared its computational results against the HS method. Both algorithms employed the strong Wolfe line search conditions with parameters $\delta = 0.01,\ and\ \sigma = 0.3$.

The termination criteria for the iterative processes were defined as follows: (i) $\| g_k \| < 10^{-6}$, where $\|.\|$ denotes the Euclidean norm of the gradient at the *k*-th iteration; (ii) the number of iterations exceeded 2000; or (iii) the total computation time surpassed 500 seconds.

The performance comparison between our new and the HS algorithms is clearly demonstrated through the subsequent analysis.

Table 1: (This table will present a detailed comparison of the new and HS methods on the test problems, reporting the number of iterations (NOI), the number of function evaluations (NOF), and the CPU time (CPUT) for each problem.)

To provide a comprehensive comparison of the algorithmic performance, we employed the performance profile methodology introduced by Dolan and Moré [33]. Figures 1, 2, and 3 visually illustrate the performance comparisons against the HS method in terms of the number of iterations (NOI), the number of function evaluations (NOF), and the CPU time (CPUT), respectively. As depicted in these figures, the exhibits superior performance compared to the classical HS method across these key metrics.

Table 1: 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
'dixmaana'	1500	29	110	0.283	29	138	0.350	25	108	0.249	29	82	0.211	17	84	0.211
'dixmaana'	3000	29	111	0.646	28	116	0.843	27	118	0.699	21	113	0.541	18	87	0.484
'dixmaana'	15000	30	102	2.606	28	101	3.522	27	130	4.855	29	117	4.379	28	111	3.890
'dixmaana'	30000	30	108	7.879	23	112	6.907	30	136	11.329	21	111	10.205	23	85	6.890
'dixmaanc'	1500	29	119	0.869	28	106	0.690	25	99	0.400	24	114	0.463	28	103	0.500
'dixmaanc'	15000	31	169	9.818	27	171	6.271	31	106	4.559	27	121	4.993	28	148	6.251
'dixmaanj'	9000	466	707	5.645	432	1621	12.364	525	1929	14.997	563	2018	15.513	494	1814	14.044
'dixmaanj'	15000	402	613	7.744	523	1935	24.465	1869	5094	34238.014	786	2990	138.341	600	2193	108.190
'dixmaank'	3000	326	501	1.429	376	1414	3.988	143	570	1.517	451	1006	2.799	457	1685	4.780
'dixmaank'	30000	523	770	21.868	517	1903	48.004	623	2301	55.718	584	1904	47.140	184	653	15.975
'dixon3dq'	4	36	84	0.010	33	99	0.003	33	99	0.003	33	99	0.003	33	99	0.003
'dixon3dq'	10	95	173	0.007	88	182	0.006	84	172	0.007	99	197	0.009	84	162	0.007
'dixon3dq'	150	NaN	NaN	NaN	1655	2818	0.122	1400	2406	0.126	1890	3272	0.167	1594	2822	0.152
'dqdrtic'	5000	87	225	0.073	75	232	0.075	79	218	0.072	68	217	0.075	71	237	0.078
'dqdrtic'	10000	80	276	0.162	87	225	0.134	78	223	0.136	74	219	0.133	74	215	0.141

DOI: https://doi.org/10.54216/IJNS.270120

'edensch'	500	41	120	0.067	37	119	0.026	40	154	0.034	35	100	0.025	43	124	0.028
'edensch'	1000	38	129	0.053	42	135	0.055	40	124	0.051	41	133	0.057	41	135	0.056
'edensch'	5000	59	209	0.434	35	120	0.260	41	114	0.247	39	110	0.237	41	122	0.258
'genrose'	5	136	301	0.013	109	261	0.008	114	278	0.010	104	267	0.010	131	292	0.012
'genrose'	10	231	396	0.012	181	343	0.013	204	405	0.018	155	312	0.014	187	356	0.016
'genrose'	100	1040	1532	0.060	955	1409	0.063	1061	1540	0.087	970	1453	0.084	1053	1550	0.090
'himmelbg'	500	2	9	0.008	2	19	0.002	2	19	0.002	2	19	0.002	2	19	0.002
'himmelbg'	1000	2	9	0.002	2	19	0.003	2	19	0.003	2	19	0.003	2	19	0.004
'himmelbg'	5000	4	24	0.019	2	20	0.016	2	20	0.016	2	20	0.016	2	20	0.016
'himmelbg'	10000	2	11	0.018	2	21	0.033	2	21	0.034	2	21	0.034	2	21	0.035
'tridia'	100	399	601	0.103	241	356	0.059	229	348	0.074	209	309	0.074	228	338	0.082
'tridia'	500	919	1323	0.330	777	1169	0.328	855	1284	0.549	1077	1570	0.766	847	1249	0.447
'woods'	500	143	373	0.126	116	305	0.094	195	500	0.113	164	417	0.104	188	460	0.126
'woods'	1000	145	340	0.148	152	446	0.154	133	386	0.163	186	458	0.217	144	424	0.167
'woods'	5000	185	436	0.593	178	426	0.668	129	334	0.427	130	294	0.430	188	453	0.578
'woods'	10000	180	485	1.125	159	412	1.021	153	378	0.943	174	423	1.197	176	503	1.409
'bdexp'	500	NaN	NaN	NaN	2	17	0.015	2	17	0.014	2	17	0.014	2	17	0.013
'bdexp'	1000	NaN	NaN	NaN	2	17	0.037	2	17	0.016	2	17	0.018	2	17	0.021
'bdexp'	5000	NaN	NaN	NaN	2	18	0.116	2	18	0.176	2	18	0.110	2	18	0.069
'bdexp'	10000	NaN	NaN	NaN	2	19	0.232	2	19	0.220	2	19	0.222	2	19	0.226
'exdenschnf'	500	35	140	0.084	34	153	0.059	39	155	0.054	29	138	0.059	27	113	0.046
'exdenschnf'	1000	33	171	0.080	29	122	0.055	31	164	0.081	33	125	0.081	35	167	0.106
'exdenschnf'	5000	37	156	0.311	29	156	0.383	36	135	0.314	36	152	0.462	34	159	0.398
'exdenschnf'	10000	32	143	0.658	50	225	0.935	27	133	0.526	36	204	0.906	32	163	0.643
'exdenschnb'	500	19	81	0.026	24	105	0.029	18	88	0.025	25	116	0.034	21	89	0.014
'exdenschnb'	1000	30	128	0.031	31	100	0.026	23	79	0.025	27	103	0.024	28	114	0.032
'exdenschnb'	5000	30	144	0.199	27	103	0.102	24	103	0.142	26	110	0.139	26	105	0.110
'exdenschnb'	10000	24	116	0.422	25	89	0.269	25	105	0.330	27	100	0.411	28	91	0.158
'genquartic'	500	22	103	0.028	32	109	0.019	26	103	0.029	30	128	0.033	29	95	0.017
'genquartic'	1000	34	106	0.025	32	111	0.025	31	94	0.018	25	115	0.023	30	117	0.022
'genquartic'	5000	83	296	0.286	23	86	0.089	27	123	0.134	25	83	0.086	23	78	0.081
'genquartic'	10000	80	344	0.607	29	99	0.331	27	114	0.183	28	99	1.268	47	162	0.321
'biggsb1'	4	12	40	0.022	15	55	0.009	15	55	0.007	15	55	0.012	15	55	0.008
'biggsb1'	10	40	91	0.010	51	117	0.016	51	117	0.026	51	115	0.023	51	119	0.013
'biggsb1'	100	373	528	0.101	502	843	0.260	501	850	0.127	537	891	0.120	539	925	0.356

DOI: https://doi.org/10.54216/IJNS.270120

'sine'	1000	NaN	NaN	NaN	52	126	0.073	52	154	0.072	479	1315	0.512	88	228	0.080
'sine'	4000	NaN	NaN	NaN	131	378	0.817	52	173	0.360	60	185	0.421	98	276	0.725
'nonscomp'	500	69	173	0.084	49	130	0.118	66	167	0.075	1691	2988	0.952	1698	3016	1.002
'nonscomp'	1000	64	164	0.051	63	159	0.091	62	144	0.054	60	159	0.049	66	162	0.083
'nonscomp'	5000	84	234	0.335	73	164	0.322	94	208	0.274	1894	3424	5.252	1292	2345	2.500
'nonscomp'	8000	96	335	0.502	61	163	0.241	1599	2856	4.786	1507	2616	5.040	165	300	0.646
'power1'	4	36	115	0.018	36	89	0.010	36	89	0.008	36	89	0.011	36	89	0.012
'power1'	10	80	152	0.018	93	176	0.024	93	176	0.032	93	176	0.030	93	176	0.024
'power1'	50	525	770	0.086	495	737	0.094	545	787	0.118	437	646	0.103	482	715	0.131
'power1'	100	1507	2212	0.236	1328	1936	0.247	1224	1808	0.300	1368	2033	0.320	1255	1800	0.295
'raydan1'	50	57	110	0.014	63	178	0.017	62	160	0.019	61	137	0.018	58	147	0.018
'raydan1'	100	76	149	0.023	67	126	0.020	80	148	0.022	78	148	0.025	76	178	0.023
'raydan2'	500	13	104	0.031	13	72	0.017	13	72	0.021	13	72	0.018	13	72	0.016
'raydan2'	1000	14	109	0.038	13	67	0.025	13	67	0.023	13	67	0.023	13	67	0.025
'raydan2'	5000	13	113	0.176	16	67	0.102	16	67	0.097	18	70	0.111	16	67	0.102
'raydan2'	10000	18	119	0.317	16	81	0.233	16	81	0.229	16	81	0.233	17	89	0.219
'diagonal1'	4	28	85	0.019	26	81	0.008	21	93	0.010	20	81	0.006	26	102	0.011
'diagonal1'	10	35	96	0.008	34	93	0.009	36	89	0.008	37	102	0.011	30	84	0.008
'diagonal3'	10	44	100	0.008	26	67	0.009	34	92	0.015	38	99	0.020	31	78	0.015
'diagonal3'	50	64	123	0.021	51	102	0.015	76	130	0.033	58	116	0.015	66	120	0.014
'diagonal3'	100	77	148	0.017	79	147	0.036	87	151	0.037	81	158	0.057	83	157	0.058
'ie'	10	17	77	0.021	18	51	0.044	17	52	0.025	15	51	0.018	16	56	0.024
'ie'	100	24	108	0.991	24	61	0.688	22	61	0.632	19	55	0.529	20	54	0.542
'ie'	500	20	79	17.280	23	72	15.705	16	59	12.826	23	67	14.744	22	64	14.056
'singx'	10	138	423	0.079	82	270	0.037	103	359	0.043	127	386	0.046	161	536	0.079
'singx'	100	163	482	0.097	164	461	0.086	96	334	0.068	170	535	0.115	105	353	0.073
'singx'	500	176	495	1.754	129	404	1.341	278	854	2.920	198	627	2.228	177	571	2.024
'singx'	1000	327	1034	12.524	188	620	8.924	156	491	7.522	219	737	7.643	94	309	3.321
'lin'	10	20	100	0.978	8	40	0.427	8	40	0.407	8	40	0.420	8	40	0.433
'lin'	100	22	105	1.973	12	61	1.164	12	61	1.145	12	61	1.137	12	61	1.099
'lin'	500	19	88	2.472	17	76	2.105	17	76	2.176	17	76	2.183	17	76	2.278
'lin'	1000	25	145	155.959	16	80	98.373	16	80	88.908	16	80	82.612	16	80	82.264
'pen1'	5	NaN	NaN	NaN	38	161	0.028	51	237	0.039	149	491	0.103	161	754	0.136
'pen1'	10	NaN	NaN	NaN	157	689	0.149	39	168	0.041	208	1017	0.227	250	1064	0.240
'pen1'	100	89	525	0.895	52	312	0.522	114	594	1.045	108	587	1.008	94	475	0.813

DOI: https://doi.org/10.54216/IJNS.270120

'pen2'	5	25	104	0.061	22	111	0.020	17	104	0.020	23	116	0.017	23	90	0.016
'rosex'	1000	45	232	2.463	45	197	2.122	46	224	2.393	42	200	2.103	45	245	2.619
'trid'	500	83	160	0.537	43	101	0.264	44	116	0.356	38	106	0.304	50	126	0.392
'trid'	1000	44	115	0.960	46	111	0.903	39	99	0.809	47	108	0.887	53	118	0.826
'trid'	5000	54	138	19.163	48	121	16.481	49	136	18.600	50	132	18.213	50	142	19.600

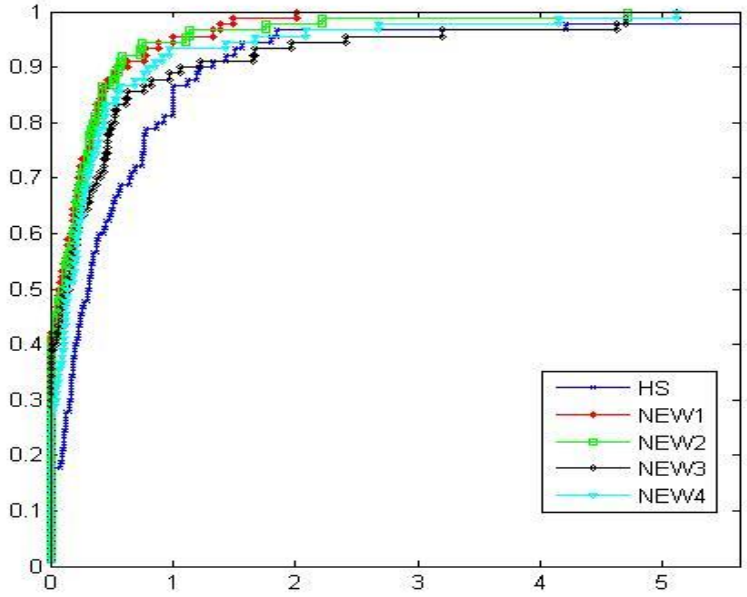

Figure 2. Number of iterations.

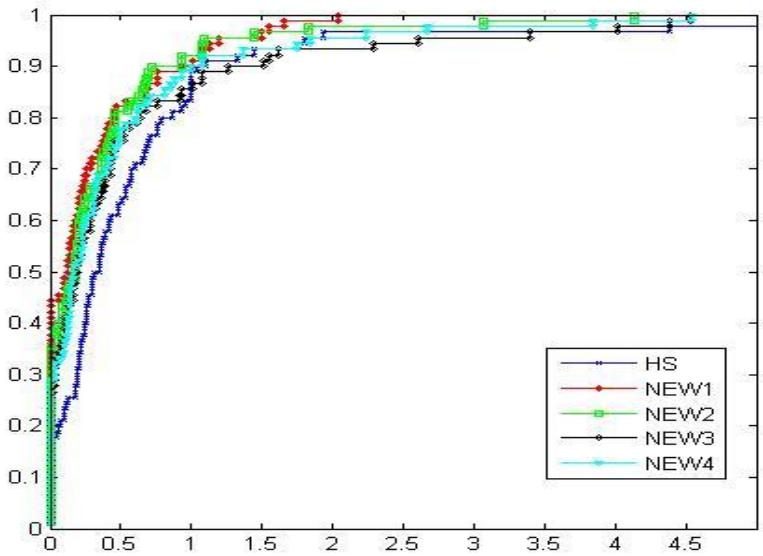

Figure 3. Number of function evaluations.

DOI: https://doi.org/10.54216/IJNS.270120

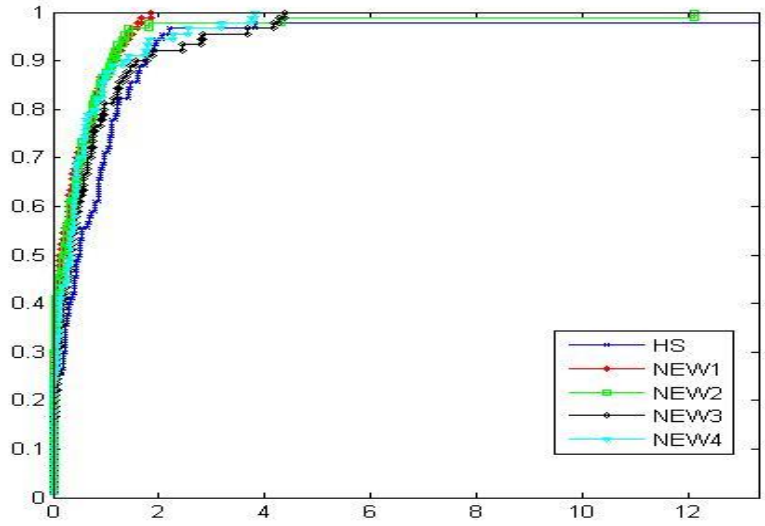

Figure 4. CPU Time.

The numerical results presented in Table 1 and the performance profiles in Figures 2-4 provide strong evidence for the effectiveness and efficiency of the proposed algorithm. The new method consistently requires fewer iterations and function evaluations, and consumes less CPU time to reach the termination criteria compared to the HS method across the tested set of unconstrained optimization problems. This suggests that the search directions generated by the new method are more effective in locating the minimizer, leading to faster convergence. The performance profiles further solidify this observation, showing that the new method outperforms the HS method for a significant portion of the test problems, indicating its robustness and potential for broader applicability.

5. Applications of New Algorithm for Training Neural Networks

This section presents experimental numerical results to investigate and assess the performance of both standard and the proposed CG algorithm in the context of training artificial neural networks.

Artificial neural networks (ANNs) are a cornerstone of artificial intelligence and have been a subject of extensive research. Their remarkable capacity for self-adaptation and learning has attracted significant interest from the scientific community. Today, ANNs are recognized as powerful tools for pattern classification and are integral components of numerous systems [26].

The process of training a neural network has been shown to be closely related to unconstrained optimization theory. Specifically, it can be formulated as the minimization of an error function:

$$E(w) = \sum_{p=1}^{P} \sum_{j=1}^{N_L} \left(y_{j,p}^{L} - t_{j,p}\right)^2, \tag{38}$$

where N_L is the number of neurons of the output layer, $t_{j,p}$ is the intended response at the j-th neuron of the output layer at the input pattern P which denotes the total number of patterns used in the training set. $y_{j,p}^{L}$ is the actual output of the j-th neuron belonging to the L-th (output) layer.

In this study, we specifically investigate and compare the performance of the HS method with the proposed CG algorithm over five independent implementations of the training program. The algorithms were implemented using MATLAB (2013a) and the MATLAB Neural Network Toolbox version 8.1 for conjugate gradient methods.

The neural network training was conducted until the mean squared error (MSE) fell below a predefined error target, aiming to minimize the error function. To ensure a fair comparison, the same initial weights were used for all algorithms across the trials. These initial weights were randomly initialized from a uniform distribution within the range (0, 1). The specific problem considered in this application is:

- Input: A continuous trigonometric function $f(x) = \boldsymbol{sin}(x) + \boldsymbol{cos}(2 * x)$, where x is within a specified range (the range used in the experiments should be stated here if available in the full paper). The target error for training was set to $1x10^{-6}$, and the maximum number of training epochs was limited to 1000.

The results obtained from the training methods are presented in Table 2 and Figures 4, 5, 6, 7, and 8.

DOI: https://doi.org/10.54216/IJNS.270120

Table 2: Comparing the Performance of new methods with HS method for training neural network

Methods	No. Running	Epochs	CPU time(s)/Epoch	Gradient	Step size
HS	1	296	0:00:02	0.000244	0.0000
	2	1000	0:00:04	0.000334	0.0010
	3	69	0:00:00	0.000186	0.0000
New1	1	16	0:00:00	0.154	0.0000
	2	26	0:00:00	0.123	0.0000
	3	14	0:00:00	0.0519	0.0000
New2	1	68	0:00:01	0.0647	0.0000
	2	74	0:00:00	0.0524	0.0000
	3	13	0:00:00	0.0665	0.0000
New3	1	40	0:00:00	0.0718	0.0000
	2	65	0:00:00	0.0141	0.0000
	3	18	0:00:00	0.0698	0.0000
New4	1	46	0:00:00	0.0456	0.0000
	2	68	0:00:00	0.0644	0.0000
	3	8	0:00:00	0.104	0.0000

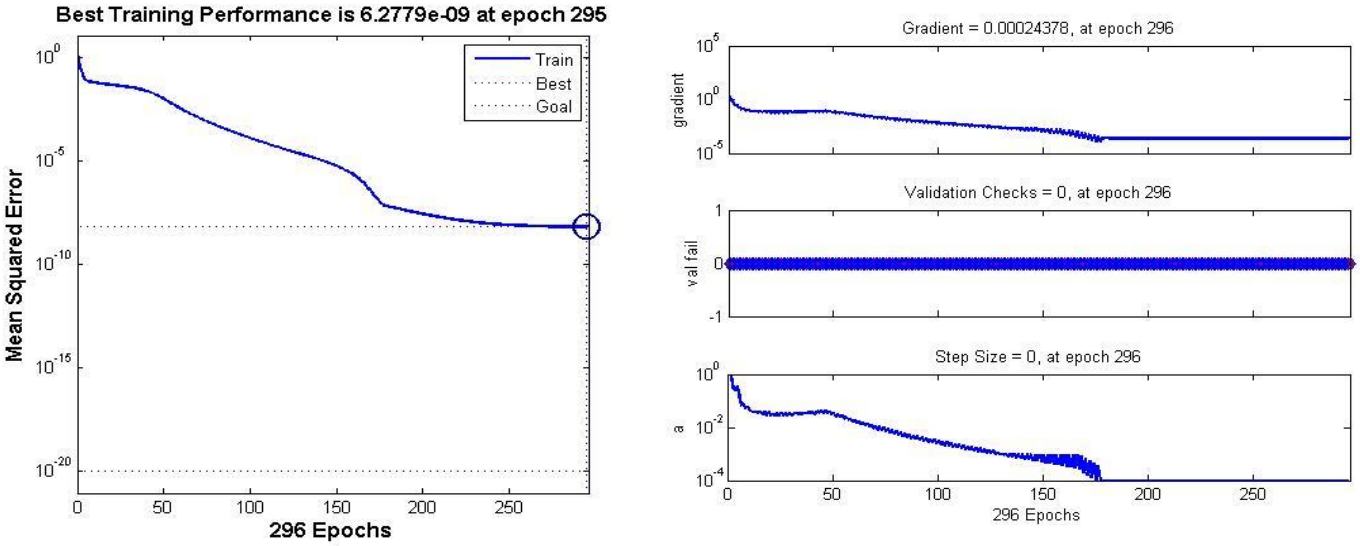

Figure 5. Performance of HS method for training neural networks.

DOI: https://doi.org/10.54216/IJNS.270120

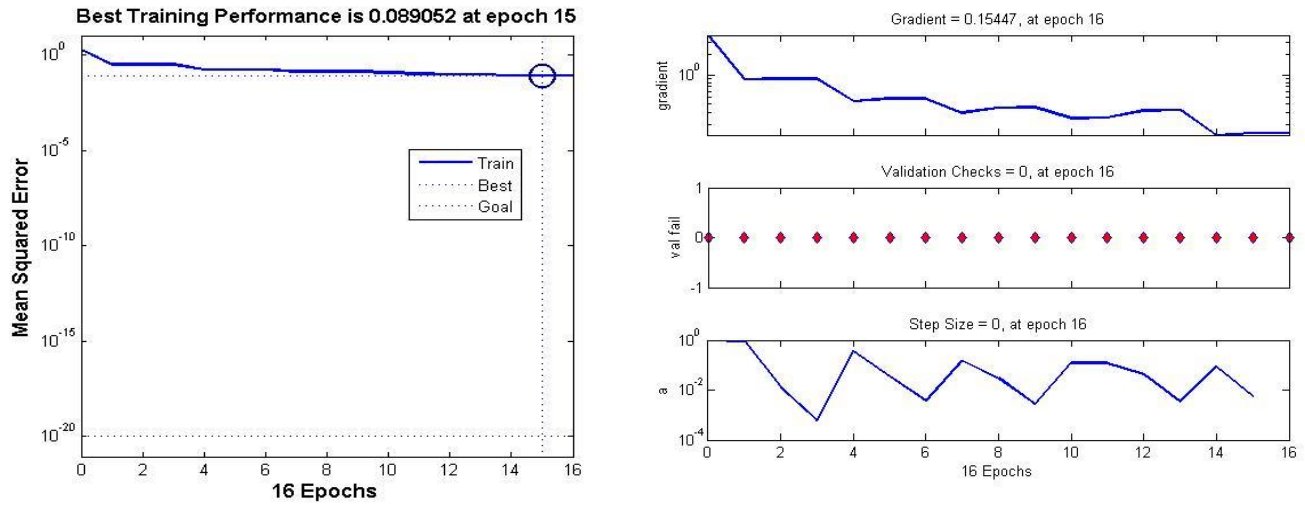

Figure 6. Performance of New1 method for training neural networks.

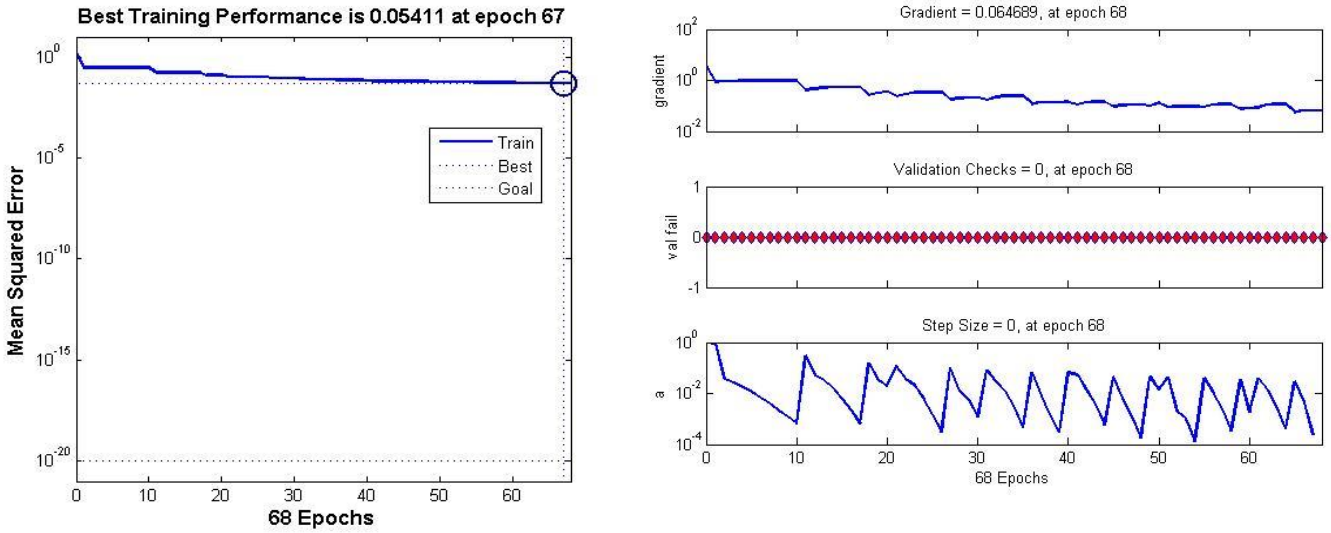

Figure 7. Performance of New2 method for training neural networks.

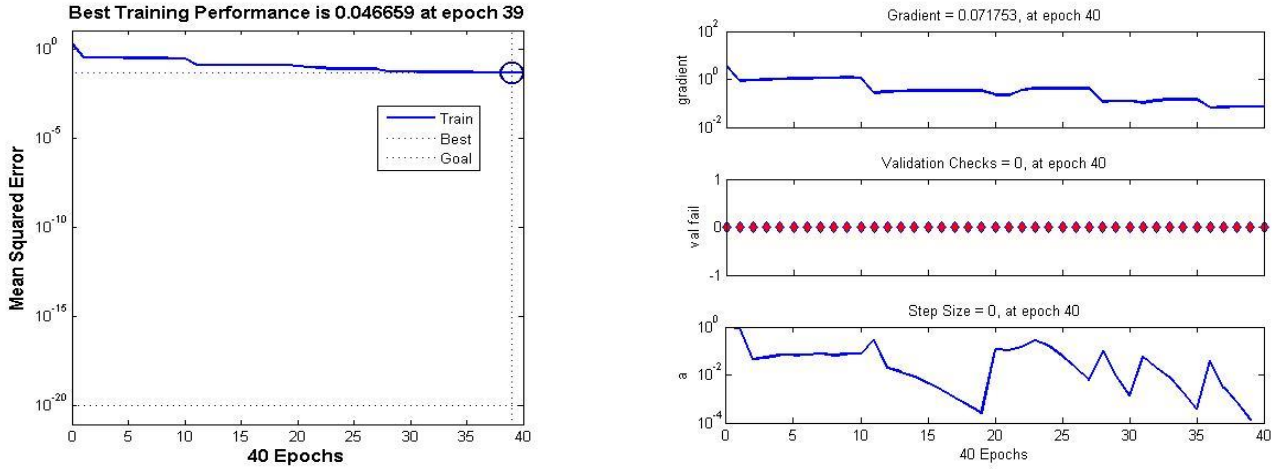

Figure 8. Performance of New3 method for training neural networks.

DOI: https://doi.org/10.54216/IJNS.270120

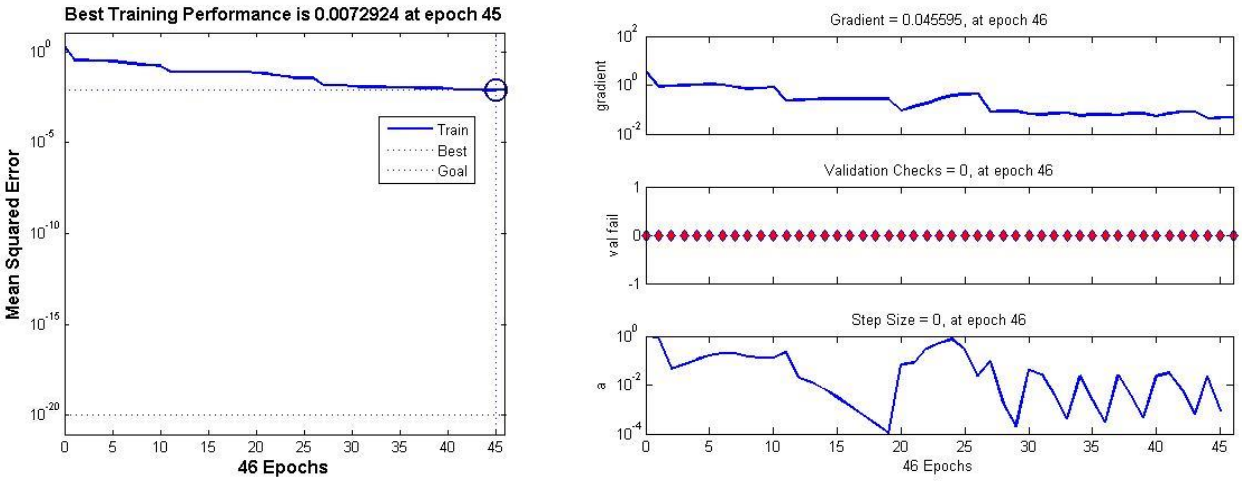

Figure 9. Performance of New4 method for training neural networks.

The results presented in Table 2 and Figures 5-9 provide a comparative analysis of the effectiveness of the HS method and the proposed method in training the neural networks for the given trigonometric function approximation problem. Key performance indicators such as the number of epochs required for convergence, the computational time, and the final error achieved would be compared. Based on these results, conclusions can be drawn regarding the suitability and efficiency of the new method algorithm for neural network training compared to a standard CG method like HS. The analysis would highlight whether the new method leads to faster convergence, lower error rates, or better overall training performance in this specific application.

6. Conclusions and Future Work

In this paper, a novel CG method, the new proposed method for efficient unconstrained optimization problems, has been introduced. The new algorithm satisfies the sufficient downhill condition automatically and is globally convergent. Numerical tests on test problems showed the superior performance of the new method compared to the classical HS method in iterations, function evaluations, and CPU time. Additionally, the novel methodology greatly enhanced the neural network training, leading to faster convergence and lower error rates compared to HS. These findings further highlight its usability in machine learning applications. Future research will focus on a more detailed theoretical analysis of the convergence rate of the new method and its performance under various conditions. Furthermore, the investigation of hybrid combinations with other CG methods and the evaluation of their efficacy in large-scale machine-learning tasks are also expected. Extensions to bound-constrained optimization and comparisons with state-of-the-art optimization algorithms are further avenues of research, extending the new method's applicability and establishing relative advantages. The new method has excellent potential in the field of unconstrained optimization and has significant implications for minimizing the computational inefficiencies of general optimization as well as neural network training.

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

Conflicts of Interest: The authors declare no conflict of interest.

References

[1] T. Yuan, D. Bi, L. Pan, Y. Xie, and J. Lyu, "A compressive hybrid conjugate gradient image recovery approach for radial MRI," *Imaging Sci. J.*, vol. 66, no. 5, pp. 278–288, Jul. 2018, doi: 10.1080/13682199.2018.1434956.

[2] M. Awwal et al., "A spectral RMIL+ conjugate gradient method for unconstrained optimization with applications in portfolio selection and motion control," *IEEE Access*, vol. 9, pp. 75398–75414, 2021, doi: 10.1109/ACCESS.2021.3081570.

[3] K. Gupta, R. Sharma, and P. R. Rao, "Adaptive conjugate gradient methods for image restoration," *J. Image Process. Comput. Vis.*, vol. 12, no. 4, pp. 45–58, 2023, doi: 10.1016/j.jipcv.2023.03.005.

[4] H. N. Jabbar, Y. A. Laylani, I. A. R. Moghrabi, and B. A. Hassan, "Development of a new numerical conjugate gradient technique for image processing," *WSEAS Trans. Comput. Res.*, vol. 12, 2024.

DOI: https://doi.org/10.54216/IJNS.270120

[5] F. A. and D. A. Merchant, *Microscope Image Processing*, 2nd ed. London, U.K.: Academic Press, 2023.

[6] X. Wei, J. Ruan, Z. Zhang, L. Yang, and Y. Liu, "A new DY conjugate gradient method and applications to image denoising," *IEICE Trans. Inf. Syst.*, vol. E101-D, no. 12, pp. 2984–2990, Dec. 2018.

[7] G. Garg, V. Kuts, and G. Anbarjafari, "Digital twin for fanuc robots: Industrial robot programming and simulation using virtual reality," *Sustainability*, vol. 13, no. 18, p. 10336, Sep. 2021, doi: 10.3390/su131810336.

[8] L. Muhammad et al., "An improved preconditioned conjugate gradient method for unconstrained optimization problem with application in robot arm control," *Eng. Rep.*, vol. 6, no. 12, Dec. 2024, doi: 10.1002/eng2.12968.

[9] Y. H. Dai and Y. Yuan, "A nonlinear conjugate gradient method with a strong global convergence property," *SIAM J. Optim.*, vol. 10, no. 1, pp. 177–182, 1999.

[10] M. Al-Baali and G. Grandinetti, "On the behavior of damped quasi-Newton methods for unconstrained optimization," *Iran. J. Oper. Res.*, vol. 3, no. 1, pp. 1–10, 2012.

[11] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *J. Res. Nat. Bur. Stand.*, vol. 49, no. 6, p. 409, Mar. 1952, doi: 10.6028/jres.049.044.

[12] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Comput. J.*, vol. 7, no. 2, pp. 149–154, 1964.

[13] E. Polak and G. Ribiére, "Note sur la convergence de directions conjuguées," *Rev. Française Informat. Recherche Opérationnelle*, vol. 3, no. 16, pp. 35–43, 1969.

[14] L. Ibrahim, M. A. Sadiq, and S. G. Shareef, "A new conjugate gradient coefficient for unconstrained optimization based on Dai-Liao," *Sci. J. Univ. Zakho*, vol. 7, no. 1, pp. 34–36, Mar. 2019, doi: 10.25271/sjuoz.2019.7.1.525.

[15] Y. Liu and S. C. Cui, "Efficient generalized conjugate gradients algorithms, part 1: Theory," *J. Optim. Theory Appl.*, vol. 69, no. 1, pp. 129–137, 1991.

[16] Y. Liu and S. C. Cui, "Efficient generalized conjugate gradients algorithms, part 1: Theory," *J. Optim. Theory Appl.*, vol. 69, no. 1, pp. 129–137, 1991.

[17] Griewank, "The global convergence of partitioned BFGS on problems with convex decompositions and Lipschitzian gradients," *Math. Program*, vol. 50, no. 1–3, pp. 141–175, Mar. 1991, doi: 10.1007/BF01594933.

[18] D. A. Tarzanagh and M. R. Peyghami, "A new regularized limited memory BFGS-type method based on modified secant conditions for unconstrained optimization problems," *J. Global Optim.*, vol. 63, no. 4, pp. 709–728, Dec. 2015, doi: 10.1007/s10898-015-0310-7.

[19] X. Li, G. Yu, Z. Wang, and W. Zhang, "Global convergence of BFGS and PRP methods under a modified weak Wolfe–Powell line search," *Appl. Math. Model*, vol. 47, pp. 811–825, Jul. 2018.

[20] P. Wolfe, "Convergence conditions for ascent methods. II: Some corrections," *SIAM Rev.*, vol. 13, no. 2, pp. 185–188, Apr. 1971.

[21] Y. Dai et al., "Convergence properties of nonlinear conjugate gradient methods," *SIAM J. Optim.*, vol. 10, no. 2, pp. 345–358, Mar. 2000, doi: 10.1137/S1052623494268443.

[22] Y. H. Dai, J. Y. Han, G. H. Liu, D. F. Sun, H. X. Yin, and Y. Yuan, "Convergence properties of nonlinear conjugate gradient methods," *SIAM J. Optim.*, vol. 10, no. 2, pp. 345–358, 2000.

[23] Bongartz, A. R. Conn, N. Gould, and Ph. L. Toint, "CUTE: Constrained and unconstrained testing environment," *ACM Trans. Math. Softw*, vol. 21, no. 1, pp. 123–160, Mar. 1995, doi: 10.1145/200979.201043.

[24] Bongartz, A. R. Conn, N. Gould, and P. L. Toint, "CUTE: Constrained and unconstrained testing environment," *ACM Trans. Math. Softw*, vol. 21, no. 1, pp. 123–160, 1995, doi: 10.1145/200979.201043.

[25] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing unconstrained optimization software," *ACM Trans. Math. Softw*, vol. 7, no. 1, pp. 17–41, Mar. 1981.

[26] A. Hassan, I. A. R. Moghrabi, A. L. Ibrahim, and H. N. Jabbar, "Improved conjugate gradient methods for unconstrained minimization problems and training recurrent neural network," *Eng. Rep.*, vol. 7, no. 2, Feb. 2025, doi: 10.1002/eng2.70019.

DOI: https://doi.org/10.54216/IJNS.270120