Iraqi Journal of Science, 2026, Vol. 67, No. 4, pp: 2356- 2371
DOI: 10.24996/ijs.2026.67.4.36

ISSN: 0067-2904

Novel Conjugate Gradient Method in Optimization and Training Neural Networks

Basim A. Hassan[1], Alaa Luqman Ibrahim[2*], Bayda Ghanim Fathi[2]
[1]*Department of Mathematics, College of Computer Sciences and Mathematics, University of Mosul*
[2]*Department of Mathematics, College of Science, University of Zakho, Zakho, Iraq*

Received: 18/ 9/2024 Accepted: 13/4/2025 Published: 30/4/2026

Abstract

This work examines novel conjugate gradient methods. To address unconstrained optimization problems and optimize the training of neural networks. The methodologies employ advanced derivative-based techniques to enhance optimization outcomes. The novel approach employs any line search to guarantee adequate descent. Moreover, we prove that, given specific assumptions, our proposed method converges universally. Experimental evidence has demonstrated that our proposed approach is superior in terms of efficiency and robustness compared to traditional conjugate gradient approaches for training neural networks and solving unconstrained optimization issues.

Keywords: Optimization; Gradient Descent; Artificial Neural Networks; Convergence Properties.

طرائق التدرج المترافق الجديدة في الامثلية وتدريب الشبكات العصبية

باسم عباس[1] ، علاء لقمان[2] ، بيداء غانم[2]
[1]قسم علوم الرياضيات، كلية علوم الحاسوب و الرياضيات، جامعة الموصل
[2]قسم علوم الرياضيات، كاية العلوم، جامعة زاخو، زاخو ، العراق

الخلاصة

يتناول هذا العمل طرائق التدرج المترافق جديدة، لمعالجة مسائل الامثلية غير المقيدة وتحسين تدريب الشبكات العصبية. تستخدم المنهجيات تقنيات متقدمة تعتمد على المشتقات لتعزيز نتائج الامثلية. يستخدم النهج الجديد أي بحث خطي لضمان الانحدار الكافي. علاوة على ذلك، أثبتنا أنه في ظل فرضيات معينة، تتقارب طريقتنا المقترحة تقارباً شاملاً. وقد أثبتت الأدلة التجريبية أن نهجنا المقترح متفوق من حيث الكفاءة والمتانة مقارنة بأساليب التدرج المترافق التقليدية لتدريب الشبكات العصبية وحل مسائل الامثلية غير المقيدة.

1. Introduction

Artificial neural networks (ANNs) are robust computational models specifically developed to replicate the cognitive functions of the human brain. These models comprise

*Email: alaa.ibrahim@uoz.edu.krd

interconnected processing units that automatically adjust and acquire knowledge from experience, rendering them very efficient at identifying patterns and developing new information. ANNs are well-known for tackling complex real-world problems since they naturally can learn and adapt independently [1], [2]. In fact, other categorization methods [3] usually show better accuracy and efficiency. Neural networks (NNs) use two main steps for classification. First, the network goes through training using a dataset to understand the relationships between inputs and outputs. After training, the network uses these learned connections to classify fresh data. Feedforward neural networks (FNNs) find great use in many fields among the multiple neural network models. Mathematically, the problem of training FNN can be formulated as the minimization of an error function E; that is to find a minimizer

$w^* = (w_1^*, w_2^*, \dots, w_n^*) \in \mathbb{R}$ such that

$$w^* = \min_{w \in \mathbb{R}} E(w), \quad (1)$$

where E is the batch error estimate computed by aggregating square differences over all training set samples, essentially

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

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 is 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. An iterative gradient-based training approach producing a sequence of weights $\{w_k\}$ starting from an initial point $w_1 \in \mathbb{R}$ uses the iterative formula to address this problem traditionally.

$$w_{k+1} = w_k + \alpha_k d_k. \quad (3)$$

Usually termed epoch, k is the current iteration; $\alpha_k > 0$ is the learning rate; d_k is a descent search direction. Since the development of backpropagation [4], various techniques using second-order information have been introduced to improve the efficiency of minimizing the error.

However, because of its frequently large dimensionality and numerous local minima in the accompanying nonconvex multimodal objective function, together with wide flat zones bordered by tiny step ones, this optimization issue is particularly challenging. As a result, a number of strategies, including conjugate gradient methods, have been proposed for enhancing the effectiveness of the minimization error process. These strategies are based on the well-established unconstrained optimization theory and make use of second order derivative related information. Because conjugate gradient methods are straightforward and require relatively little memory neither the assessment of the Hessian matrix nor the unrealistic storage of an approximate version of it, they are generally useful for effectively training neural networks. These techniques have been effectively used in a variety of applications with steady and dependable convergence in the neural network literature [5], [6]. In conjugate gradient algorithms, the current gradient and the prior search direction are combined to determine the search direction d_k:

$$d_{k+1} = \begin{cases} -g_1, & k = 0 \\ -g_{k+1} + \beta_k d_k, & k > 0' \end{cases} \quad (4)$$

where $g_k = g(x_k) = \nabla E$ is the gradient of the error function, and β_k is a parameter that influences the search direction. For the first iteration, $k = 0$, the method reduces to the steepest descent approach.

To determine the step size α_k, two types of line searches can be used: exact and inexact. Exact line search aims to find the optimal step size that minimizes the error along the search direction:

$$f(x_k + \alpha_k d_k) = \min f(x_k + \alpha d_k), \ \alpha \geq 0, \quad (5)$$

but it is usually too expensive to compute [7]. Therefore, inexact line searches, such as Wolfe-Powell (WP) line search, are preferred. WP line search includes weak Wolfe-Powell (WWP) conditions [8], [9]:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{6}$$

$$g(x_k + \alpha_k d_k)^T d_k \geq \sigma g_k^T d_k. \tag{7}$$

The second part is the strong Wolfe-Powell (SWP) line search, which is given by the following equation:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{8}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \leq \sigma |g_k^T d_k|, \tag{9}$$

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

Some of the most widely used classical CG methods include the Polak-Ribière-Polyak (PR) [10], [11], Hestenes-Stiefel (HS) [12], and Liu-Storey (LS) [13] formulas:

$$\beta_k^{PR} = \frac{g_{k+1}^T y_k}{\|g_k\|^2}, \tag{10}$$

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{y_k^T d_k}, \tag{11}$$

$$\beta_k^{LS} = \frac{g_{k+1}^T y_k}{-g_k^T d_k}. \tag{12}$$

Although these methods may not always converge, they often produce good results. Other methods, like the Fletcher-Reeves (FR) [14], Dai-Yuan (DY) [15], and Conjugate Descent (CD) [16] methods, offer strong convergence properties but may face practical issues like jamming [17].

$$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \tag{13}$$

$$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{y_k^T d_k}, \tag{14}$$

$$\beta_k^{CD} = \frac{\|g_{k+1}\|^2}{-g_k^T d_k}. \tag{15}$$

The global convergence of the FR method was first demonstrated by Zoutendijk [18] under exact line search, and later Al-Baali [19] proved its global convergence with inexact line search.

In recent years CG methods have expanded to various fields such as data estimation [20], image restoration [21], [22], as well as in signal processing [23] and [24]. Since artificial intelligence grows in importance CG methods have become essential for training neural networks helping them learn effectively through gradient based optimization [25–28].

This paper introduces a novel conjugate gradient method and evaluates its convergence properties. Our method exhibits an ample sufficient descent property. As numerical results show that the new method is more efficient and robust than the PR algorithm. Lastly, we compare the process to the HS algorithm when training neural networks.

This paper is organized as follows: Section 2 introduces the new method and its formulation, which includes the sufficient descent condition and global convergence. Section 3 applies the method to neural network training and includes numerical results. Section 4 concludes with a summary.

2. Conjugate coefficient derivation

Directly solving for the global minimizer can be excessively resource-intensive and challenging computationally. The technique can operate more efficiently and with lower computational costs by approximating the answer in real-world applications. The Taylor series used in earlier research [29], [30], to provide several helpful mathematical expressions. With the use of the Taylor formula, the following can be expressed:

$$f_k = f_{k+1} - g_{k+1}^T s_k + \frac{1}{2} s_k^T Q_{k+1} s_k - \frac{1}{6} s_k^T (T_{k+1} s_k) s_k. \quad (16)$$

Next, the derivative is calculated as:

$$g_{k+1} = -g_k + Q_{k+1} s_k - \frac{1}{2} s_k^T (T_{k+1}) s_k = 0. \quad (17)$$

This results in the following mathematical expressions:

$$s_k^T Q_{k+1} s_k = \frac{5}{6} s_k^T y_k + (f_k - f_{k+1}) - \frac{1}{3} g_k^T s_k, \quad (18)$$

and

$$s_k^T Q_{k+1} s_k = 6/5 s_k^T y_k + 6/5(f_k - f_{k+1}) + 2/5 g_k^T s_k, \quad (19)$$

to derive a new conjugate coefficient, the approach outlined in [31] is followed, where the relation is defined as:

$$-Q_{k+1}^{-1} g_{k+1} = -g_{k+1} + \beta_k s_k, \quad (20)$$

multiplying this equation by y_k, the following is obtained:

$$-Q_{k+1}^{-1} g_{k+1}^T y_k = -g_{k+1}^T y_k + \beta_k s_k^T y_k, \quad (21)$$

using Equations (18) and (19), Q_{k+1} is express as follows:

$$Q_{k+1} = \frac{5/6 s_k^T y_k + (f_k - f_{k+1}) - 1/3 g_k^T s_k}{s_k^T s_k} I_k, \quad (22)$$

and

$$Q_{k+1} = \frac{6/5 s_k^T y_k + 6/5(f_k - f_{k+1}) + 2/5 g_k^T s_k}{s_k^T s_k} I_k, \quad (23)$$

substituting Q_{k+1} into Equation (21), the new conjugate coefficients β_k are derived:

$$\beta_k^{BBK1} = \left(1 - \frac{s_k^T s_k}{5/6 s_k^T y_k + (f_k - f_{k+1}) - 1/3 g_k^T s_k}\right) \frac{g_{k+1}^T y_k}{s_k^T y_k}, \quad (24)$$

and

$$\beta_k^{BBK2} = \left(1 - \frac{s_k^T s_k}{6/5 s_k^T y_k + 6/5(f_k - f_{k+1}) + 2/5 g_k^T s_k}\right) \frac{g_{k+1}^T y_k}{s_k^T y_k}. \quad (25)$$

By performing algebraic manipulations on the above formulas and ensuring the sufficient descent condition, the result is:

$$\beta_K = \frac{1}{s_k^T y_k} \left(y_k - \omega \frac{\|y_k\|^2}{s_k^T y_k} s_k\right)^{\top} g_{k+1}, \quad (26)$$

where ω is defined as

$$\omega_k^{BBK1} = \frac{(s_k^T y_k)}{\|y_k\|^2} \left[\frac{s_k^T y_k}{s_k^T s_k} * \frac{s_k^T s_k}{5/6 s_k^T y_k + (f_k - f_{k+1}) - 1/3 g_k^T s_k}\right], \quad (27)$$

or

$$\omega_k^{BBK2} = \frac{(s_k^T y_k)}{\|y_k\|^2} \left[\frac{s_k^T y_k}{s_k^T s_k} * \frac{s_k^T s_k}{6/5 s_k^T y_k + 6/5(f_k - f_{k+1}) + 2/5 g_k^T s_k}\right]. \quad (28)$$

Algorithm: Implementation of the BBK1 and BBK2 CG methods.

Step 1 :(Initialization) Given an initial point $w_1 \in R^n$, parameters, $0 < \delta < \sigma < 1$, and $\varepsilon > 0$. Set $d_1 = -g_1$, and $k = 0$.

Step 2 : If $\|g_k\| \leq \varepsilon$ then stop.

Step 3 : Compute the step size α_k by the weak Wolfe line search,

Step 4 : Generate the next iteration by (3).

Step 5 : Compute d_{k+1} by (4) and choose an appropriate conjugate parameter β_k by (26) where ω determined by (27) or (28).

Step 6 : Set $k := k + 1$ and go to Step 1.

The required sufficient descent condition and global convergence of the new algorithm are described in detail in this section. The following theorem contains the sufficient descent condition:

Theorem 2.1 Let s_k, y_k, $g_{k+1} \in R^n$ and $\beta_k \in R$, which is defined by (26). If $s_k^T y_k \neq 0$ and $\omega > 1/4$, then:

$$d_{k+1}^T g_{k+1} \leq -\left[1 - \frac{1}{4\omega}\right] \|g_{k+1}\|^2. \quad (29)$$

Proof: Using induction. Since $d_0 = -g_0$, we have $g_0^T d_0 = -\|g_0\|^2$. Now, assume $d_k^T g_k \leq -c\|g_k\|^2$ is true and then from multiplying (4) by g_{k+1}, we have:

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + \left(\frac{g_{k+1}^T y_k}{s_k^T y_k} - \omega \frac{\|y_k\|^2}{(s_k^T y_k)^2} g_{k+1}^T s_k\right) s_k^T g_{k+1}, \quad (30)$$

which simplifies to:

$$d_{k+1}^T g_{k+1} = \frac{(g_{k+1}^T y_k)(s_k^T g_{k+1})(s_k^T y_k) - \|g_{k+1}\|^2 (s_k^T y_k)^2 - \omega \|y_k\|^2 (g_{k+1}^T s_k)^2}{(s_k^T y_k)^2}. \quad (31)$$

By applying:

$$w = \frac{1}{\sqrt{2\omega}} (s_k^T y_k) g_{k+1} \text{ and } v = \sqrt{2\omega} (g_{k+1}^T s_k) y_k$$

into the inequality:

$$w^T v \leq \frac{1}{2} (\| w \|^2 + \| v \|^2). \quad (32)$$

Obviously,

$$(g_{k+1}^T y_k)(s_k^T g_{k+1})(s_k^T y_k) \leq \frac{1}{2}\left[\frac{1}{2\omega} (s_k^T y_k)^2 \|g_{k+1}\|^2 + 2\omega (s_k^T g_{k+1})^2 \|y_k\|^2\right]. \quad (33)$$

Therefore, by (31) and (33), we have that:

$$d_{k+1}^T g_{k+1} \leq \frac{\left[\frac{1}{4\omega} - 1\right](s_k^T y_k)^2 \|g_{k+1}\|^2 + [\omega - \omega](s_k^T g_{k+1})^2 \|y_k\|^2}{(s_k^T y_k)^2}. \quad (34)$$

Hence, we have:

$$d_{k+1}^T g_{k+1} \leq -\left[1 - \frac{1}{4\omega}\right] \|g_{k+1}\|^2. \quad (35)$$

This concludes our proof. A similar basic result holds for the BBK2.

Several assumptions regarding the objective function E are necessary to establish the global convergence of the suggested optimization methods:

Assumption 1:

1. The level set $S = \{x \,|E(x) \leq E(x_k)\}$ at x_k is bounded. There exists a constant $a > 0$ such that:

$$\|x\| \leq a, \forall x \in S. \quad (36)$$

2. In some neighbourhood N of S, the function E is continuously differentiable, and its gradient is Lipschitz continuous with Lipschitz constant $L > 0$. This implies:

$$\|g(x) - g(y)\| \leq L\|x - y\| \ \forall x, y \in S. \quad (37)$$

3. There exists a positive constant b such that:

$$\|g(b)\| \leq b \quad \forall x \in S. \quad (38)$$

4. If f is strongly convex, then there exists a constant $\mu > 0$, such that:

$$\mu \|x - y\|^2 \leq (\nabla E(x) - \nabla E(y))^T (x - y), \text{ for all } x, y \in S. \quad (39)$$

Using Wolfe line search, it can be deduced that Dai et al. [32] established the prerequisite for CG approaches to converge.

Lemma 2.2: Let Assumptions (1), hold. Consider the method (2) and (3), where α_k is obtained by the Wolfe conditions and d_k is a descent direction. If:

$\sum_{k\geq 0}\frac{1}{\|d_{k+1}\|^2}=\infty,$
(40)
then:

$$\lim_{k\to\infty} inf\ \|g_{k+1}\|=0. \tag{41}$$

Theorem 2.3: Suppose the all assumptions holds. Let the sequences $\{x_k\}$ and $\{d_k\}$ given by the (4) and (26). If step size α_k satisfies Wolfe conditions, then we have:

$$\lim_{k\to\infty}\inf\|g_{k+1}\|=0. \tag{42}$$

Proof: From the search direction (4) and definition of β_k by (26) we get:

$$\begin{aligned}\|d_{k+1}\| &= \|-g_{k+1}+\beta_k^{\mathrm{BBK1}}d_k\|, \qquad (43)\\ &\leq \|g_{k+1}\|+\left|\beta_k^{\mathrm{BBK1}}\right|\|d_k\|,\\ &\leq \|g_{k+1}\|+\left\|\left(y_k-\omega\frac{\|y_k\|^2}{s_k^Ty_k}s_k\right)\right\|\frac{\|g_{k+1}\|}{\|s_k\|\|y_k\|}\|d_k\|,\\ &\leq \|g_{k+1}\|+\frac{\|y_k\|\|g_{k+1}\|+\omega\frac{\|g_{k+1}\|\|y_k\|^2\|s_k\|}{\|s_k\|\|y_k\|}}{\alpha_k\|d_k\|\|y_k\|}\|d_k\|,\\ &\leq \|g_{k+1}\|+\frac{\|y_k\|\|g_{k+1}\|+\omega\|g_{k+1}\|\|y_k\|}{\alpha_k\|d_k\|\|y_k\|}\|d_k\|,\\ &\leq \left[1+\frac{1}{\alpha_k}+\frac{\omega}{\alpha_k}\right]\|g_{k+1}\|,\\ &\leq \left[\frac{\alpha_k+1+\omega}{\alpha_k}\right]\|g_{k+1}\|. \qquad (44)\end{aligned}$$

Therefore,

$$\sum_{k\geq 1}\frac{1}{\|d_k\|^2}\geq\left(\frac{\alpha_k}{\alpha_k+1+\varpi}\right)\frac{1}{\Gamma}\sum_{k\geq 1}1=\infty, \tag{45}$$

Using Lemma 2.2 implies that $\lim_{k\to\infty}\inf\|g_k\|=0$. It is easy to test that for BBK2.

3. Numerical results

The proposed BBK1 and BBK2 methods are demonstrated in this section to be effective in unconstrained optimization problems and in the enhancement of recurrent neural network training.

3.1 Unconstrained optimization problems

In this subsection, we establish the numeral experiments, compute, and compare our method to the PR method. The comparison is created using the MATLAB 2013b run on an HP personal laptop, and the test function is constructed using the functions chosen from the CUTE library [33] as well as additional unconstrained problem collections [34], [35] with varied dimensions. We examined the computational results of our approach to the PR conjugate gradient method. The algorithms utilized the strong Wolfe line search conditions with $\delta=0.01,\text{and }\sigma=0.3$.

The iterations were ended if any of the following requirements were met: (i) $\|g_{k+1}\|<10^{-6}$, where $\|\cdot\|$ indicates the Euclidean norm; (ii) the number of iterations exceeded 2000; or (iii) the computation time surpassed 500 seconds. The performance gap between our technique and the PR conjugate gradient algorithm is clearly established.

Table 1 displays the calculation result, with number of iterations (NOI), number of function evaluation (NOF) and CPU time (CPUT), we applied the performance profile presented by Dolan and Moré [36] which are visible in Figures 1, 2, and 3.

Table 1: Comparative analysis of the performance of PR, BBK1, and BBK2 methods on unconstrained optimization cases.

Test Function	Dimension	PR			BBK1			BBK2		
		NOI	NOF	CPUT	NOI	NOF	CPUT	NOI	NOF	CPUT
dixmaana	1500	24	133	0.287	21	67	0.129	25	72	0.121
dixmaana	3000	24	134	0.515	22	70	0.347	19	66	0.346
dixmaana	15000	22	132	2.718	23	72	1.256	24	74	1.203
dixmaana	30000	24	132	3.484	27	80	2.193	26	79	2.131
Dixmaanb	1500	24	121	0.174	20	73	0.092	24	76	0.118
Dixmaanb	3000	27	121	0.399	21	90	0.310	21	87	0.288
Dixmaanb	15000	34	174	2.339	22	83	1.107	21	79	1.077
Dixmaanb	30000	36	181	4.879	25	83	2.266	25	83	2.195
Dixmaanc	1500	26	119	0.205	24	93	0.139	16	84	0.135
dixmaanc	3000	31	172	0.576	20	92	0.303	26	95	0.320
dixmaanc	15000	27	130	1.736	29	111	1.437	26	106	1.401
dixmaand	30000	42	233	6.144	26	98	2.540	31	110	2.941
dixmaand	1500	33	164	0.228	23	78	0.101	24	93	0.122
dixmaand	3000	20	102	0.287	19	79	0.288	21	80	0.279
dixmaane	1500	216	575	0.854	182	241	0.325	185	247	0.361
dixmaane	3000	327	895	3.025	250	341	1.165	226	318	1.046
dixmaane	15000	483	1340	17.510	426	570	7.243	439	564	7.362
dixmaane	24000	569	1533	31.535	530	668	14.224	489	644	14.771
dixmaanf	1500	189	503	0.704	139	214	0.313	165	240	0.520
dixmaanf	3000	236	704	2.398	220	329	1.150	244	342	1.112
dixmaanf	15000	1745	4674	59.772	364	511	6.568	333	465	6.045
dixmaanf	24000	278	774	15.903	397	538	11.092	424	582	12.087
dixmaani	150	522	1456	0.369	494	612	0.158	535	684	0.175
dixmaani	300	579	1489	0.522	709	887	0.359	623	816	0.306
dixmaani	1500	1061	2878	3.867	708	877	1.225	727	929	1.287
dixmaani	3000	1022	2686	7.345	1045	1329	3.751	1038	1341	3.793
dixmaanj	300	122	330	0.145	258	349	0.133	286	390	0.135
dixmaanj	9000	282	831	6.386	175	243	1.875	183	253	1.979
dixmaanj	15000	822	2446	31.366	238	328	4.214	248	361	4.646
dixmaank	30000	209	560	14.595	221	333	8.751	253	357	9.252
dixmaank	1500	159	456	0.605	137	205	0.300	139	206	0.292
dixmaank	3000	155	464	1.259	129	195	0.550	163	237	0.642
dixmaank	15000	992	3343	42.388	209	319	4.085	185	292	3.804
dixmaank	30000	209	560	14.586	221	333	8.556	253	357	9.314
dixmaanl	1500	146	490	0.709	124	193	0.279	121	208	0.285
dixmaanl	3000	97	289	0.740	140	220	0.603	140	218	0.849

dixmaanl	18000	662	1764	27.479	177	275	4.190	172	271	4.131
dixmaanl	30000	206	593	15.361	203	314	8.028	238	349	9.053
dqdrtic	500	68	301	0.018	40	146	0.008	69	213	0.012
dqdrtic	1000	72	290	0.021	91	234	0.019	62	214	0.017
dqdrtic	5000	72	301	0.088	47	163	0.052	54	172	0.054
dqdrtic	10000	71	315	0.173	53	188	0.111	50	206	0.119
freuroth	4	100	361	0.021	88	244	0.008	138	322	0.019
freuroth	10	115	443	0.019	111	273	0.010	97	242	0.010
freuroth	50	152	570	0.025	132	306	0.019	146	333	0.017
freuroth	100	149	678	0.036	117	281	0.018	78	218	0.014
genrose	5	159	542	0.015	114	254	0.009	99	275	0.008
genrose	10	212	611	0.013	205	357	0.015	204	350	0.014
genrose	50	897	2453	0.060	537	831	0.034	517	830	0.033
genrose	100	1548	4191	0.125	889	1302	0.074	887	1261	0.060
himmelbg	500	2	9	0.006	2	9	0.001	2	9	0.001
himmelbg	1000	2	9	0.002	2	9	0.002	2	9	0.002
himmelbg	5000	3	21	0.015	3	21	0.015	3	21	0.013
himmelbg	10000	2	11	0.018	2	11	0.018	2	11	0.017
liarwhd	500	68	379	0.031	29	154	0.011	32	169	0.012
liarwhd	1000	51	275	0.035	37	173	0.021	37	214	0.023
liarwhd	5000	90	497	0.236	35	162	0.088	34	178	0.089
liarwhd	10000	87	471	0.446	40	198	0.178	43	213	0.184
woods	500	98	441	0.040	123	302	0.023	113	379	0.028
woods	1000	151	624	0.067	129	342	0.044	102	293	0.034
woods	5000	167	650	0.315	144	341	0.185	130	310	0.169
woods	10000	337	1307	1.227	124	344	0.340	155	404	0.398
bdexp	500	2	7	0.009	2	7	0.002	2	7	0.002
bdexp	1000	2	7	0.004	2	7	0.004	2	7	0.003
bdexp	5000	2	8	0.019	2	8	0.020	2	8	0.021
bdexp	10000	2	9	0.047	2	9	0.045	2	9	0.046
exdenschnf	500	25	127	0.023	29	134	0.017	35	166	0.022
exdenschnf	1000	32	164	0.034	29	107	0.020	33	143	0.029
exdenschnf	5000	34	185	0.159	27	115	0.097	28	130	0.112
exdenschnf	10000	30	167	0.262	32	173	0.280	27	160	0.255
exdenschnb	500	22	126	0.013	18	61	0.005	22	66	0.006
exdenschnb	1000	24	116	0.013	21	65	0.008	18	63	0.007
exdenschnb	5000	21	94	0.035	32	82	0.034	20	69	0.028
exdenschnb	10000	31	168	0.120	25	77	0.059	19	67	0.052
biggsb1	4	20	79	0.008	15	48	0.002	14	46	0.002
biggsb1	10	60	187	0.007	34	69	0.004	45	80	0.005

biggsb1	100	420	1189	0.040	317	418	0.028	334	404	0.022
nonscomp	500	48	173	0.018	52	134	0.009	51	120	0.009
nonscomp	1000	80	297	0.027	50	133	0.012	47	135	0.013
nonscomp	5000	881	2424	0.936	71	194	0.078	47	133	0.055
nonscomp	8000	53	186	0.110	50	152	0.093	56	159	0.096
power1	4	36	102	0.007	36	87	0.003	29	67	0.003
power1	10	99	314	0.013	75	144	0.009	77	145	0.006
power1	50	486	1316	0.036	355	522	0.027	403	611	0.034
power1	100	1243	3260	0.109	987	1431	0.081	1106	1596	0.080
raydan1	4	19	92	0.005	17	40	0.001	15	37	0.001
raydan1	50	52	172	0.007	50	88	0.004	47	84	0.003
raydan1	100	74	230	0.009	58	101	0.005	57	104	0.005
raydan1	5000	675	1882	0.826	575	855	0.471	564	856	0.460
raydan2	500	11	66	0.010	7	38	0.004	7	38	0.003
raydan2	1000	9	56	0.007	13	61	0.009	13	61	0.010
raydan2	5000	11	74	0.041	8	61	0.035	8	61	0.036
raydan2	10000	13	71	0.075	11	57	0.061	11	57	0.060
diagonal1	4	25	108	0.008	19	51	0.003	19	51	0.002
diagonal1	10	25	100	0.004	26	66	0.003	31	71	0.002
diagonal1	50	63	302	0.011	49	112	0.006	51	109	0.007
diagonal1	100	82	231	0.011	68	131	0.008	68	121	0.007
ie	10	12	66	0.009	13	40	0.006	11	38	0.005
ie	100	25	125	0.506	20	49	0.202	15	44	0.177
ie	500	15	66	6.491	18	48	4.748	15	38	3.737
Lin	10	11	69	0.249	8	40	0.121	8	40	0.158
Lin	100	11	65	0.498	10	51	0.367	10	51	0.416
Lin	500	11	71	0.897	14	55	0.688	14	55	0.665
Lin	1000	14	86	59.038	12	61	42.242	12	61	41.768

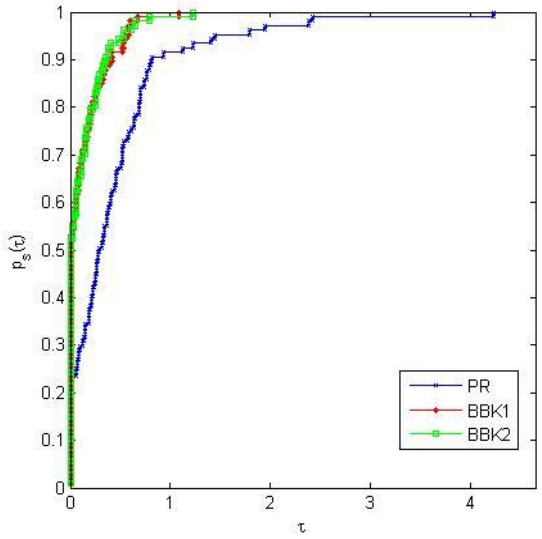

Figure 1: (NOI).

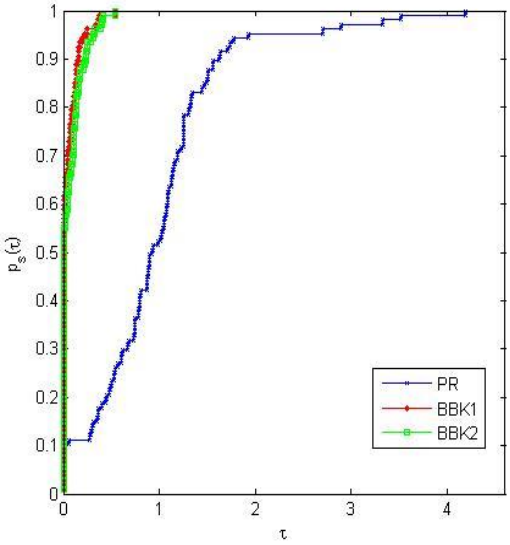

Figure 2: (NOF).

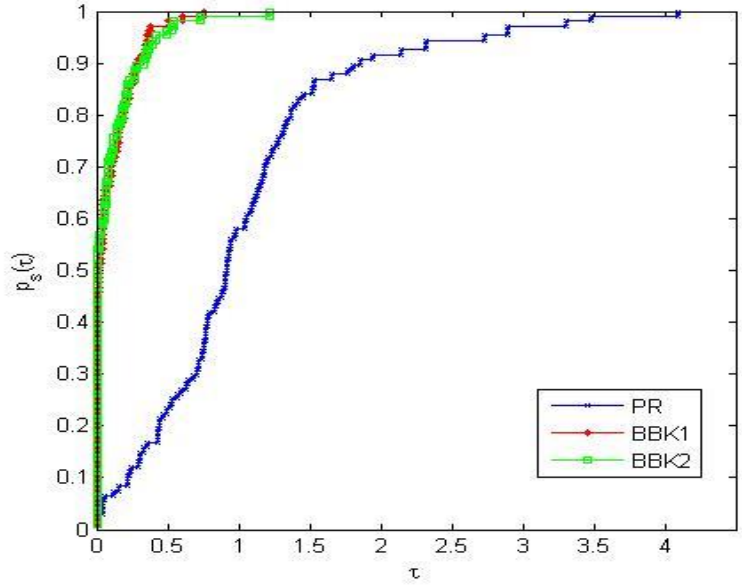

Figure 3: CUP Time.

Figures 1, 2, and 3 illustrate the performance profiles of the three conjugate gradient methods, namely PR, BBK1, and BBK2, across different metrics: the number of iterations (NOI), the number of function evaluations (NOF), and CPU time. These metrics provide crucial insights into the comparative efficiency and reliability of the algorithms under consideration.

Figure 1: This figure evaluates the methods based on the number of iterations required to reach a predefined convergence criterion. It is evident that BKK1 and BKK2 demonstrate superior performance, achieving faster convergence with fewer iterations compared with PR. This suggests that the modified BBK methods are more efficient in terms of iteration count that making them potentially more appealing for computationally intensive problems.

Figure 2: the performance in terms of NOF further highlights the efficiency of BBK1 and BBK2 Those methods consistently require fewer function evaluations than PR for indicating

a reduced computational burden during optimization. That advantage is particularly important when the function of evaluations is costly as in large scale optimization problems.

One can notice that Figure 3 focuses on the total computational time required by each algorithm here BBK1 and BBK2 again outperform PR by showcasing their ability to achieve faster Solutions. The reduced CPU time can be attributed to their lower number of iterations and function evaluations. That efficiency makes them practical for time sensitive applications, like real-time systems and signal processing significance of the results.

The consistent performance of BBK1 and BBK2 across all three metrics underlines their robustness and applicability to a wide range of optimization problems. Those results emphasize the value of adopting those methods particularly, in scenarios where computational resources and time are constrained. Furthermore, the comparison highlights the trade-offs among the algorithms providing a basis for selecting the most suitable method based on the specific requirements of the application.

3.2 Neural networks training application

This subsection presents the results of training neural networks using various conjugate gradient methods on the Fisher's dataset that based on the characteristics of iris flowers are classified into one of three species Setosa or, Versicolor or, Virginica. That dataset used to evaluate the performance of many training methods with a specific emphasis on efficiency and robustness.

The iris flowers dataset which is accessible through MATLAB, that comprises 150 samples each of which is characterized by four attributes:

- Sepal Length: The length of the sepal in centimeters.
- Sepal Width: The width of the sepal in centimeters.
- Petal Length: The length of the pedal in centimeters
- Pedal Width: The width of the pedal in centimeters.

Using the dataset, a comprehensive assessment of many training techniques was carried out, focusing mainly on their efficiency and robustness. Continuous training was conducted until the mean squared error (MSE) reached a predetermined threshold, guaranteeing the reduction of the error function. Uniform initialization was upheld for all algorithms, where weights were randomly chosen within intervals of (0, 1) to provide an equitable comparison.
Maximum Epochs: 1000.
Training Goal: High precision target (e.g., error less than 1×10^{-6}).
Training Display: Progress shown every 5 epochs.
The structural design of the neural network is seen in Figure 4.

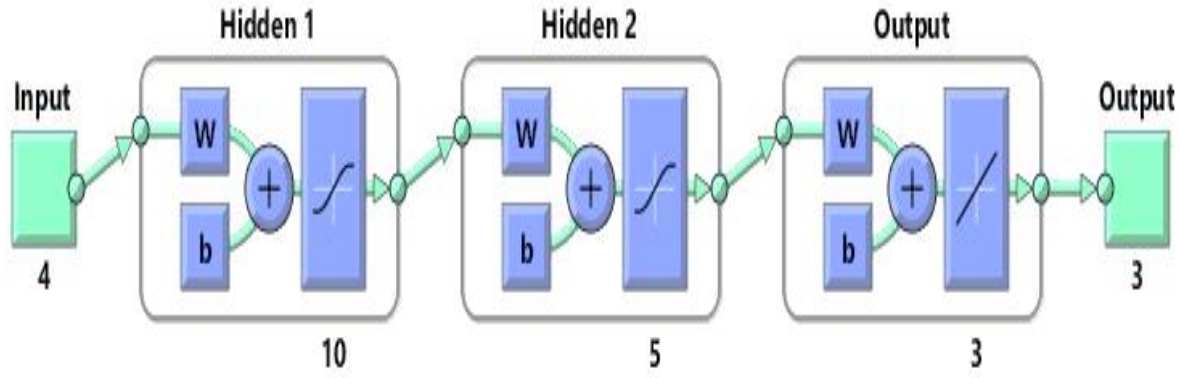

Figure 4: The architecture of neural network

Figure 4 illustrates a neural network architecture comprising an input layer with 4 features, two hidden layers with 10 and 5 neurons respectively, and an output layer with 3 neurons. Each layer processes data through weighted connections (W), bias terms (b), and nonlinear activation functions, enabling the model to extract and represent complex patterns for multi-output prediction tasks.

An overview of the training techniques' results is provided in Table 2 and visually shown in Figures 5, 6 and 7.

Table 2: Performance Comparison of HS, BBK1, and BBK2 Methods in Neural Network Training

Method	Epochs	MSE	Gradient	Step size
HS	134	0.00999	0.00260	0.00848
BBK1	76	0.00923	0.00474	0.0695
BBK2	85	0.00918	0.00207	0.0145

Table 2: BBK1 exhibits superior efficiency by completing training in 76 epochs, the fewest among the approaches, while maintaining a low mean squared error (MSE) of 0.00923. Additionally, it has the highest step size of 0.0695, which enables quicker convergence. In this comparison, BBK1 emerges as the most efficient approach for neural network training, with a much shorter duration of 85 epochs compared to HS's 134 epochs.

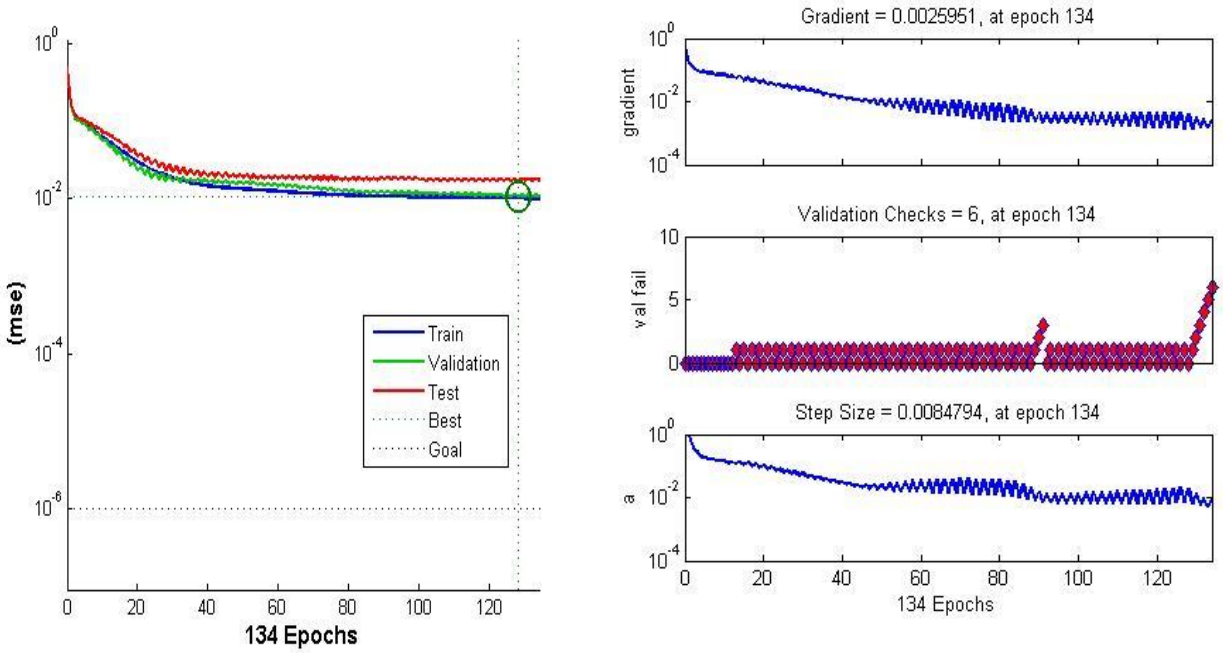

Figure 5: Performance of HS

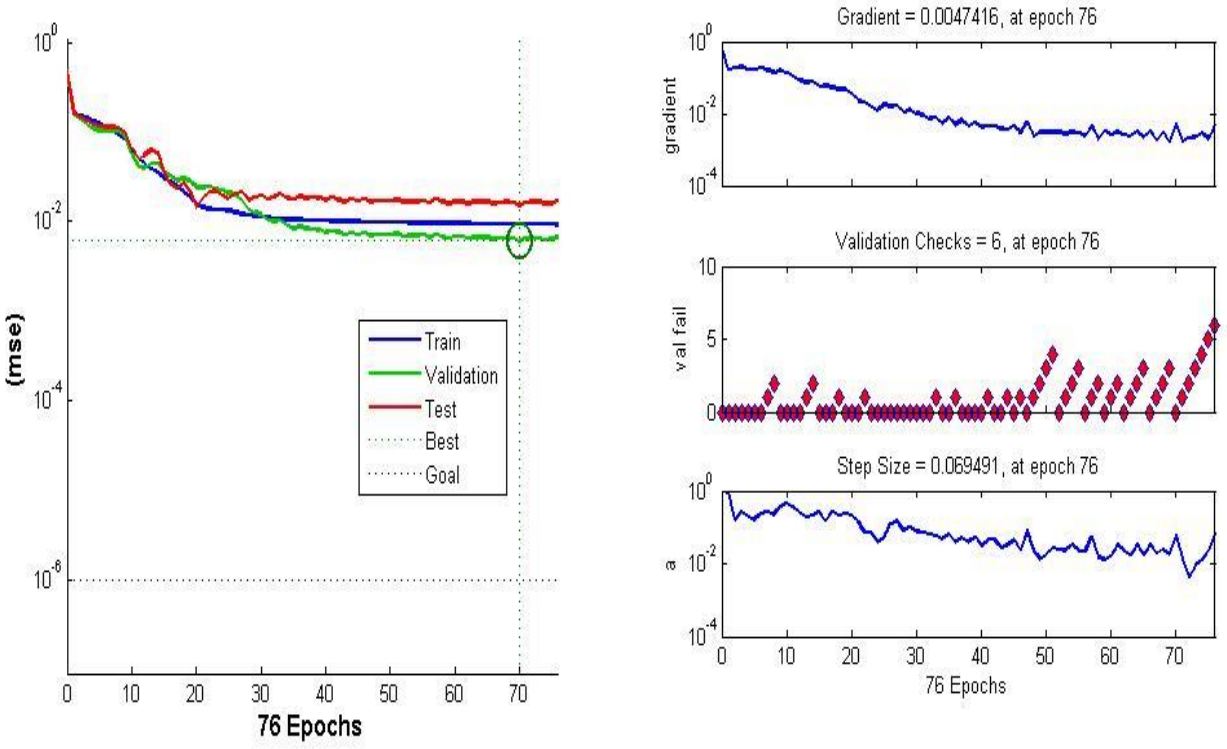

Figure 6: Performance of BBK1.

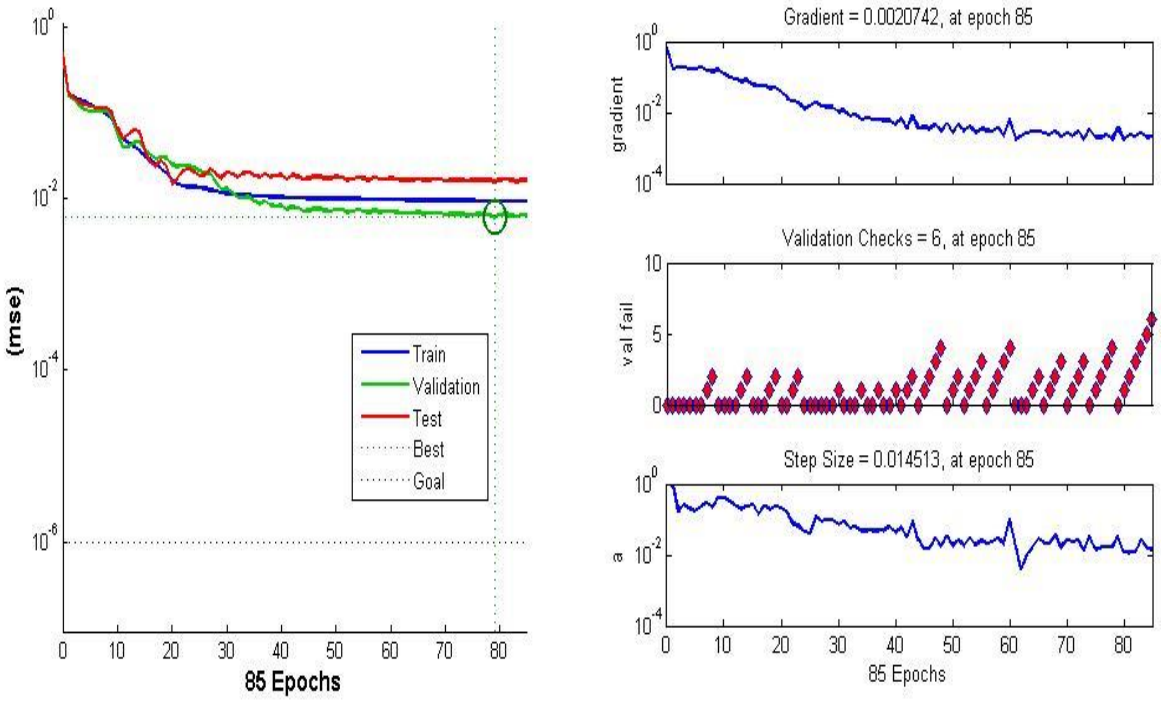

Figure 7: Performance of BBK2.

Figures 5, 6 and 7 illustrate the performance of the HS, BBK1, and BBK2 algorithms, respectively, during neural network training. Each figure displays the Mean Squared Error (MSE) for training, validation, and testing phases, alongside the gradient, validation checks, and step size over epochs.

Figure 5 shows the HS algorithm requiring 134 epics to converge, with a gradual decline in MSE and stable step size adjustments. Figure 6 highlights BBK1's faster convergence at 76 epochs, achieving similar accuracy with fewer validation checks and smaller step sizes. Figure 7 demonstrates BBK2 converging in 85 epochs, balancing efficiency and stability. These comparisons underscore the differences in convergence speed and computational efficiency among the algorithms.

4. Conclusions

In this study there are two new conjugate gradient techniques BBK1 and BBK2, which introduced to enhance the efficiency of unconstrained optimization in neural network training. These techniques adhere to Wolf line search conditions for ensuring global convergence and adequate descent without frequent restarts. The numerical experiments demonstrate that BBK1 and BBK2 provide superior performance compared to traditional methods PR and HS. Specifically, BBK1 exhibits faster convergence with lower computational costs and better MSE values that making it particularly advantages for neural network training. These findings highlight the potential of BBK1 and BBK2 to improve optimization processes in computational efficiency in machine learning applications.

References

[1] C. M. Bishop, *Neural networks for pattern recognition*. Oxford University Press, 1995.

[2] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. E. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, pp. e00938,

2018

[3] B. Lerner, H. Guterman, M. Aladjem, and I. Dinstein, "A comparative study of neural network based feature extraction paradigms," *Pattern Recognition Letters*, vol. 20, no. 1, pp. 7–14, 1999.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing*, vol. 1, pp. 318–363, 1986.

[5] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks*, vol. 6, no. 4, pp. 525–533, 1993.

[6] C. C. Peng and G. D. Magoulas, "Adaptive Nonmonotone Conjugate Gradient Training Algorithm for Recurrent Neural Networks," In *Proceedings - International Conference on Tools with Artificial Intelligence, ICTAI*, vol. 2, pp. 374-381, 2007.

[7] W. SUN and Y. X. YUAN, *Optimization Theory and Methods: Nonlinear Programming*, Springer Optimization and Its Applications, 2013.

[8] P. Wolfe, "Convergence Conditions for Ascent Methods. II: Some Corrections," *SIAM Review*, vol. 13, no. 2, pp. 185–188, 1971.

[9] P. Wolfe, "CONVERGENCE CONDITIONS FOR ASCENT METHODS*," *SIAM Review*, vol. 11, no. 2, pp. 226–235, 1969.

[10] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue française d'informatique et de recherche opérationnelle. Série rouge*, vol. 3, no. R1, pp. 35–43, 1969.

[11] B. T. Polyak, "The Conjugate Gradient Method in Extremal Problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 807–821, 1969.

[12] M. R. H. S. and E. Stiefe, "Methods for conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, pp. 409–436, 1952.

[13] Y. Liu and C. Storey, "Efficient generalized conjugate gradient algorithms, part 1: Theory," *Journal of Optimization Theory and Applications*, vol. 69, no. 1, pp. 129–137, 1991.

[14] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149–154, Jan. 1964.

[15] Y. H. Dai and Y. Yuan, "A Nonlinear Conjugate Gradient Method with a Strong Global Convergence Property," *SIAM Journal on Optimization*, vol. 10, no. 1, pp. 177–182, 1999.

[16] C. Witzgall and R. Fletcher, *Practical Methods of Optimization.*, 2nd ed., vol. 53, no. 188. New York: John Wiley & Sons, 1989.

[17] N. Andrei, "Hybrid Conjugate Gradient Algorithm for Unconstrained Optimization," *Numerical Algorithms*, vol. 47, pp. 143–156, 2008.

[18] G. Zoutendijk, "Nonlinear Programming Computational Methods," *In: Abadie, J. Ed., Integer and Nonlinear Programming, NorthHolllad, Amsterdam,* pp. 37-86, 1970.

[19] M. Al-baali, "Descent property and global convergence of the fletcher-reeves method with inexact line search," *IMA Journal of Numerical Analysis*, vol. 5, no. 1, pp. 121–124, 1985.

[20] S. Shoid et al., "The application of new conjugate gradient methods in estimating data," *International Journal of Engineering and Technology (UAE)*, vol. 7, no. 2, pp. 25–27, 2018.

[21] A. Alhawarat, Z. Salleh, H. Alolaiyan, H. El Hor, and S. Ismail, "A three-term conjugate gradient descent method with some applications," *Journal of Inequalities and Applications*, vol. 2024, no. 1, 2024.

[22] Z. Chen, H. Shao, P. Liu, G. Li, and X. Rong, "An efficient hybrid conjugate gradient method with an adaptive strategy and applications in image restoration problems," *Applied Numerical Mathematics*, vol. 204, pp. 362–379, 2024.

[23] A. B. Abubakar, P. Kumam, H. Mohammad, and A. M. Awwal, "A Barzilai-Borwein gradient projection method for sparse signal and blurred image restoration," *Journal of the Franklin Institute*, vol. 357, no. 11, pp. 7266–7285, 2020.

[24] G. Abbass, H. Chen, M. Abdullahi, and A. B. Muhammad, "An efficient projection algorithm for large-scale system of monotone nonlinear equations with applications in signal recovery," *Journal of Industrial and Management Optimization*, vol. 20, no. 11, pp. 3580–3595, 2024.

[25] W. Zhao and H. Huang, "Adaptive stepsize estimation based accelerated gradient descent algorithm for fully complex-valued neural networks," *Expert Systems with Applications*, vol. 236, pp. 121166, 2024.

[26] H. Iiduka and Y. Kobayashi, "Training deep neural networks using conjugate gradient-like methods," *Electronics (Switzerland)*, vol. 9, no. 11, pp. 1–25, 2020.

[27] H. Kim et al., "Variable three-term conjugate gradient method for training artificial neural networks," *Neural Networks*, vol. 159, pp. 125–136, 2023.

[28] W. Ali, H. H. Zuberi, X. Qing, A. Miyajan, A. Jaffar, and A. Alharbi, "Scaled Conjugate Gradient Neural Intelligence for Motion Parameters Prediction of Markov Chain Underwater Maneuvering Target," *Journal of Marine Science and Engineering*, vol. 12, no. 2, 2024.

[29] B. A. Hassan and A. Ayoob, "On the New Quasi-Newton Equation for Unconstrained Optimization, " *8th IEC 2022 - International Engineering Conference: Towards Engineering Innovations and Sustainability*, pp. 168-172, 2022.

[30] B.A. Hassan and M. Mohammed, "Extra Quasi-Newton Equation for Unconstrained Optimization," In *2022 8th International Conference on Contemporary Information Technology and Mathematics, ICCITM 2022*, pp. 375- 379, 2022.

[31] L. Nazareth, "A conjugate direction algorithm without line searches," *Journal of Optimization Theory and Applications*, vol. 23, no. 3, pp. 373–387, 1977.

[32] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, and Y. X. Yuan, "Convergence Properties of Nonlinear Conjugate Gradient Methods," *SIAM Journal on Optimization*, vol. 10, no. 2, pp. 345–358, 1999.

[33] N. I. M. Gould, D. Orban, and P. L. Toint, "CUTEr and sifdec: A constrained and unconstrained testing environment, revisited," *ACM Transactions on Mathematical Software*, vol. 29, no. 4, pp. 373–394, 2003.

[34] J. J. Moré, B. S. Garbow, and K. E. Hillstrom, "Testing Unconstrained Optimization Software," *ACM Transactions on Mathematical Software (TOMS)*, vol. 7, no. 1, pp. 17–41, 1981.

[35] N. Andrei, "An unconstrained optimization test functions collection," *Advanced Modelling and Optimization*, vol. 10, no. 1, pp. 147–161, 2008.

[36] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.