Al-Rafidain Journal of Computer Sciences and Mathematics (RJCSM)

www.csmj.uomosul.edu.iq

Improved Dai–Liao Conjugate Gradient Methods for Large-Scale Unconstrained Optimization

Basim A. Hassan[1], Alaa Luqman Ibrahim[2] and Mehamdia Abd Elhamid[3]

[1]Department of Mathematics, College of Computers Sciences and Mathematics, University of Mosul, Mosul, Iraq
[2]Department of Mathematics, College of Science, University of Zakho, Zakho, Kurdistan Region, Iraq
[3]Laboratory Informatics and Mathematics, Mohamed Cherif Messaadia University, Souk Ahras, Algeria
Email: basimah@uomosul.edu.iq[1] , alaa.ibrahim@uoz.edu.krd[2] and mehamdiaabdelhamid56@gmail.com[3]

Article information

Article history:
Received 03 May ,2025
Revised 17 June ,2025
Accepted 23 June ,2025
Published 26 June ,2025

Keywords:
Unconstrained Optimization,
Secant Condition,
Conjugate Gradient Method,
Dai–Liao Method,
Computational Efficiency.

Correspondence:
Basim A. Hassan
Email:
basimah@uomosul.edu.iq

Abstract

This research introduces and evaluates two enhanced conjugate gradient methods for unconstrained optimization, building upon the Dai–Liao conjugacy condition and further refined through the application of Taylor series expansion. These novel methodologies were rigorously compared against the classical Hestenes-Stiefel (HS) method using a diverse suite of benchmark test functions. The numerical results obtained unequivocally demonstrate a significant improvement in computational efficiency achieved by the proposed methods. Notably, our enhanced methods consistently outperformed the HS method across several critical performance metrics, including a reduction in the number of iterations required for convergence, a decrease in the total number of function evaluations, and an overall faster computation time.

DOI: 10.33899/csmj.2025.159774.1186,

1. Introduction

In recent years, the field of unconstrained optimization has experienced substantial progress, particularly in the development of efficient algorithms tailored for solving large-scale unconstrained optimization problems. This study focuses on the unconstrained optimization problem formulated as:

$$minf(x),\ x \in \mathbb{R}^n, \qquad (1)$$

where $f: \mathbb{R}^n \to \mathbb{R}$ is assumed to be a continuously differentiable function. Unconstrained optimization plays an important role in practical applications. For example, it is essential in machine learning, which is popular in many fields. Recently, Jandaghiet al. [1] proposed a novel machine learning approach to train a soft trunk robot model, giving a more accurate model estimation. When the scale of (1) is not large, the Newton method and quasi-Newton method are promising in solving it, and Javad Ebadi et al. [2] proposed a new BFGS method based on modified secant relations and verified its efficiency by applying it in solving (1). However, when solving (1) in large scale, the conjugate gradient (CG) method is more welcomesince it requires lower memory in practical computation. Featured in its efficiency, CG method has been applied by many researchers in a substantial number of fields such as image restoration problem [3,4], compressive signals problem [5], and signal reconstruction problem [6]. machine learning and numerical analysis [7,8,9]. In other practical fields, like disease model, optimization methods are used to solve fractional-order tuberculosis disease model [10] and fractional fascioliasis disease model [11], where the CG method can be considered to apply in the optimization problems appearing in the models.

Typically, CG methods iteratively update an initial approximation $x_0 \in \mathbb{R}^n$ through the iterative formula:

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

where α_k denotes the step length, determined via a line search technique. In this study, the strong Wolfe conditions [12] are used to define the step length, ensuring that:

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

$$\sigma\, g_k^T\, d_k \leq g_{k+1}^T\, d_k \leq -\sigma g_k^T\, d_k, \tag{4}$$

where $0 < \rho < \sigma < 1$. The search direction $d_k \in \mathbb{R}^n$ is updated using:

$$d_{k+1} = -g_{k+1} + \beta_k d_k \ \ for\ k > 0. \qquad d_0 = -g_0 \tag{5}$$

where $g_k = \nabla f(x_k)$, and the scalar β_k defines different CG variants. Extensive research has been conducted to improve CG methods by developing new formulas for β_k that enhance convergence behavior and numerical performance.

The classical CG methods are typically categorized based on their convergence and numerical characteristics. The first category, known for its global convergence, includes the Fletcher–Reeves (FR) [13], Dai–Yuan (DY) [14], and Conjugate Descent (CD) [15] methods:

$$\beta_k^{FR} = \frac{g_{k+1}^T g_{k+1}}{g_k^T g_k}, \tag{6}$$

$$\beta_k^{DY} = \frac{g_{k+1}^T g_{k+1}}{d_k^T y_k}, \tag{7}$$

$$\beta_k^{CD} = \frac{g_{k+1}^T g_{k+1}}{-g_k^T d_k}, \tag{8}$$

Although globally convergent, these methods may suffer from performance degradation in practice due to issues such as jamming. The second category includes methods such as Polak–Ribière–Polyak (PRP) [16,17], Hestenes–Stiefel (HS) [18], and Liu–Storey (LS) [19], characterized by

$$\beta_k^{PRP} = \frac{g_{k+1}^T y_k}{g_k^T g_k}, \tag{9}$$

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{d_k^T y_k}, \tag{10}$$

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

These methods generally yield better numerical results but have more complex convergence analyses.

Motivated by the limitations of existing methods, researchers have explored new CG variants that aim to balance strong convergence properties with efficient numerical behavior. Among these efforts are modified CG algorithms [20–22] and three-term CG schemes [23,24].

One such advancement is the Dai–Liao (DL) method [25], which modifies the traditional conjugacy condition $d_{k+1}^\top y_k = 0$, typically valid under exact line search, to a weaker condition:

$$d_{k+1}^\top y_k = -t g_{k+1}^\top d_k, t \geq 0, \tag{12}$$

leading to the DL update formula:

$$\beta_k^{\mathrm{DL}} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - t\frac{g_{k+1}^T v_k}{d_k^T y_k}, \tag{13}$$

When if $t = 0$, the DL formula reduces to the HS method. To ensure global convergence for general (non-convex) functions, a modified version is proposed:

$$\beta_k^{\mathrm{DL}+} = \max\left\{\frac{g_{k+1}^T y_k}{d_k^T y_k}, 0\right\} - t\frac{g_{k+1}^T v_k}{d_k^T y_k}. \tag{14}$$

While the DL method offers convergence under uniform convexity, its global behavior relies heavily on the parameter t, and it may not always generate sufficient descent directions [26]. To address these concerns, Hager and Zhang [27] proposed an alternative formula incorporating the memoryless BFGS idea:

$$\beta_k^N = \frac{g_{k+1}^T y_k}{d_k^T y_k} - 2\frac{\|y_k\|^2}{(d_k^T y_k)^2} g_{k+1}^T v_k, \tag{15}$$

along with a restricted version ensuring descent:

and showed that (15) satisfies the descent condition $g_{k+1}^T d_{k+1}^T \leq \frac{7}{8}\|g_{k+1}\|^2$. To show that the method is globally convergence for general functions, Hager and Zhang [27] presented the following restricted version of (15):

$$\beta_k^{N+} = \max\left\{\beta_k^N, \frac{-1}{\|d_k\|\min\{\eta, \|g_k\|\}}\right\}. \tag{16}$$

Results from numerical computations has shown that the method is efficient and promising. Furthermore, the DL-like methods proposed in [28] and [29] happened to be globally convergent and numerically stable, but like the method in [30], they also fail to fulfil the sufficient descent condition. To overcome the defect with DL CG versions; using singular value study, Babaie-Kafaki and Ghanbari [31] and Andrei [32] proposed an adaptive optimal choice for t, which increased the numerical strength of the DL methods. Numerical experiments show that these algorithms are robust and more efficient than Hager and Zhang [33] CG method. Despite the fact that different choices of the parameter t have been suggested in [34, 35], and for nice review on recent advances on Dai-Liaomethods by Saman [37], the optimal choice of t in DL-type methods still requires moreattention, especially with hybrid CG methods.

More recently, based on the integration of DL and JHJ techniques, Aminifard and Babaie-Kafaki [21] introduced the EJHJ method with:

$$\beta_k^{EJHJ} = \frac{\|g_k\|^2 - max\left\{\frac{\|g_k\|}{\|g_{k-1}\|} g_k^\top g_{k-1}, 0\right\}}{max\{\|g_{k-1}\|^2, d_{k-1}^\top y_{k-1}\}} - t\frac{g_k^\top v_{k-1}}{max\{\|g_{k-1}\|^2, d_{k-1}^\top y_{k-1}\}}, \tag{17}$$

and numerical experiments conducted on the EJHJ method verified its promising performance. Other recent contributions include a CG variant proposed by Alaa et al. [38], defined as:

$$\beta_k = \frac{g_{k+1}^T y_k}{d_k^T y_k} - t\frac{\alpha_k \|d_k\|^2 g_{k+1}^T d_k}{(d_k^T y_k)^2},\ t > 0. \tag{18}$$

Building upon these advancements, the present work proposes a novel CG method for unconstrained optimization,

inspired by the Dai–Liao conjugacy condition and further refined using Taylor series expansion. The remainder of this article is organized as follows: Section 2 introduces the theoretical preliminaries and outlines the proposed hybrid method. Section 3 presents implementation details and computational considerations. Section 5 concludes the paper with a summary of findings and future perspectives.

2. An Improved Dai–Liao Method and Its Algorithm

Building upon the classical Dai–Liao framework, we propose an enhanced class of CG methods derived through a refined approximation of the secant equation. Specifically, Basim et al. [39,40] extended the conventional secant condition by employing Taylor series expansions to obtain more accurate representations of the Hessian approximation. Two alternative secant relations were introduced:

$$v_k^T Q(x_k) v_k = 5/6 v_k^T y_k + (f_k - f_{k+1}) - 1/3 g_k^T v_k, \quad (19)$$

and

$$v_k^T Q(x_k) v_k = 6/5 v_k^T y_k + 6/5 (f_k - f_{k+1}) + 2/5 g_k^T v_k, \quad (20)$$

where $Q(x_k)$ denotes an approximation to the Hessian matrix at iteration k, and $v_k = x_{k+1} - x_k$. These formulations incorporate both gradient and function value information to improve the quality of the Hessian approximation.

In parallel, Perry [41] proposed a modified conjugacy condition, based on the second-order information:

$$d_{k+1}^T y_k = -v_k^T g_{k+1}, \quad (21)$$

which provides a more flexible framework for updating the search direction.

Motivated by the above, we incorporate relations (19), (20), and (21) into a modified Dai–Liao update formula. The proposed conjugate gradient parameter is defined as:

$$\beta_k^{DL} = \frac{g_{k+1}^T y_k - t_k g_{k+1}^T v_k}{d_k^T y_k}, \quad (22)$$

where the scalar t_k is adaptively computed based on one of two strategies:

- **Dai-Liao (BA1) method:**

$$t_k = \frac{5}{6} + \frac{(f_k - f_{k+1}) - 1/3 g_k^T v_k}{d_k^T y_k}. \quad (23)$$

- **Dai-Liao (BA2) method:**

$$t_k = \frac{6}{5} + \frac{6/5 (f_k - f_{k+1}) + 2/5 g_k^T v_k}{d_k^T y_k}. \quad (24)$$

These adaptive parameters enable the proposed methods to better reflect the underlying curvature information, enhancing the descent property and overall robustness of the search direction.

The full procedure of the improved Dai–Liao methods is outlined in the following algorithm.

ALGORITHM: IMPROVED DAI–LIAO METHODS (BA1 AND BA2)

Step 1 : (Initialization) Given an initial point $w_0 \in R^n$, parameters $0 < \rho < \sigma < 1$, and $\varepsilon > 0$. Set $d_0 = -g_0$, 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 : Compute x_{k+1} by (2).

Step 5 : Compute the search direction by (5) and β_k from (22), where t_k from (13), or (24).

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

3. Numerical Results

This section presents a detailed numerical evaluation of the proposed improved Dai–Liao methods for solving unconstrained optimization problems. The performance of the proposed methods is compared against the well-known HS method, which is widely recognized in the optimization literature for its robustness and efficiency.

To ensure fairness and comprehensiveness, a variety of test problems were selected from the CUTE problem set [42] and additional benchmark collections [43,44], covering a broad spectrum of problem types and dimensions. All experiments were implemented in MATLAB R2013b and executed on a standard HP laptop.

To maintain consistency, identical line search parameters and Wolfe condition settings were used across all methods. Specifically, the line search parameters were set to ρ=0.01 and σ=0.3. The algorithm terminates under any of the following conditions:

- The gradient norm satisfies $\| g_k \| < 10^{-6}$,
- The number of iterations exceeds 2000,
- Or the CPU time exceeds 500 seconds.

If the algorithm fails to meet the convergence criteria, the outcome is recorded as NaN, indicating numerical instability or divergence. The variable N denotes the dimensionality of the problem under consideration.

To quantify performance, three key metrics were recorded for each algorithm:

- **NOI:** Number of iterations,
- **NOF:** Number of function evaluations,
- **CUPT:** CPU time in seconds.

A detailed comparison of these metrics is presented in

Table 1. Table 2 presents a comparative evaluation of the proposed Dai–Liao-based methods against the classical HS method. In this comparison, the HS method is considered the baseline, set at 100% for all metrics. The percentages shown for BA1 and BA2 represent their computational costs relative to HS. A lower percentage indicates improved performance and higher efficiency. In addition, **Figures 1–3** depict the performance profiles of the tested algorithms using the methodology proposed by Dolan and Moré [45]. These performance profiles provide a graphical summary of the relative efficiency of each method across the full set of test problems.

PERFORMANCE PROFILE ANALYSIS

The performance ratio for problem p∈P and algorithm s∈S is defined as:

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}: s \in S\}},$$

where $t_{p,s}$ represents a specific performance metric (e.g., NOI, NOF, or CUPT), and the denominator corresponds to the best performance achieved by any method on problem p.

The cumulative performance of algorithm sss is captured by:

$$\rho_s(\tau) = \frac{1}{n_p} \text{size}\,\{p \in P: r_{p,s} \leq \tau\},$$

where $\rho_s(\tau)$ indicates the proportion of test problems for which the performance of algorithm s is within a factor τ\tauτ of the best possible performance.

As shown in **Figures 1- 3**, the proposed methods consistently outperform the HS method across all measured criteria.

Table 1. Comparison of the proposed improved (ba1 and ba2) methods with the hs.

Test Function	N	HS			BA1			BA2		
		NOI	*NOF*	*CPUT*	*NOI*	*NOF*	*CPUT*	*NOI*	*NOF*	*CPUT*
'cosine'	500	NaN	NaN	NaN	55	231	0.02	52	195	0.02
'cosine'	1000	43	114	0.02	31	169	0.03	25	138	0.02
'cosine'	5000	NaN	NaN	NaN	34	185	0.15	53	257	0.22
'cosine'	10000	NaN	NaN	NaN	46	229	0.35	45	213	0.33
'dixmaana'	1500	41	161	0.22	27	97	0.15	27	128	0.19
'dixmaana'	3000	38	145	0.51	28	97	0.31	25	113	0.35
'dixmaana'	15000	31	137	1.75	24	96	3.91	23	119	5.77
'dixmaana'	30000	37	189	6.68	29	97	2.70	20	94	2.38
'dixmaanb'	1500	38	272	0.35	31	107	0.13	31	109	0.13
'dixmaanb'	3000	38	190	0.58	22	100	0.30	26	111	0.33
'dixmaanb'	15000	38	262	3.46	31	120	1.57	21	114	1.44
'dixmaanb'	30000	36	188	4.90	26	122	3.14	22	108	2.94
'dixmaanc'	1500	29	171	0.28	27	103	0.13	24	112	0.13
'dixmaanc'	3000	31	157	0.46	28	98	0.30	33	158	0.47
'dixmaanc'	15000	45	229	3.23	27	118	1.52	25	138	1.90
'dixmaand'	30000	44	207	5.40	37	122	3.03	28	136	3.66
'dixmaand'	1500	34	195	0.24	33	131	0.15	21	122	0.22
'dixmaand'	3000	40	185	0.55	23	95	0.27	26	126	0.41
'edensch'	500	47	163	0.05	51	243	0.05	67	356	0.08
'edensch'	1000	47	144	0.06	49	204	0.08	45	185	0.07
'edensch'	5000	71	331	0.69	61	447	0.92	NaN	NaN	NaN
'edensch'	10000	NaN	NaN	NaN	75	440	1.83	102	768	3.19
'eg2'	4	45	192	0.01	38	86	0.00	38	91	0.00

'fletchcr'	1000	NaN	NaN	NaN	106	730	0.07	150	1039	0.10
'fletchcr'	5000	NaN	NaN	NaN	217	1857	0.81	297	2536	1.12
'fletchcr'	10000	NaN	NaN	NaN	281	2553	2.15	199	1593	1.33
'fletchcr'	12000	NaN	NaN	NaN	313	2801	2.80	307	2712	2.69
'himmelbg'	500	7	25	0.01	3	15	0.00	3	15	0.00
'himmelbg'	1000	3	20	0.00	2	9	0.00	2	9	0.00
'himmelbg'	5000	4	27	0.02	3	21	0.01	3	21	0.01
'himmelbg'	10000	8	33	0.05	3	14	0.03	3	14	0.03
'penalty1'	500	NaN	NaN	NaN	26	130	0.16	28	160	0.20
'penalty1'	1000	NaN	NaN	NaN	19	102	0.37	17	110	0.40
'penalty1'	4000	32	261	11.77	14	81	3.70	19	108	4.82
'penalty1'	10000	NaN	NaN	NaN	74	510	116.93	143	993	225.84
'quartc'	500	37	148	0.04	46	176	0.04	44	197	0.04
'quartc'	1000	69	248	0.09	49	173	0.06	54	214	0.07
'quartc'	5000	73	281	0.51	85	293	0.53	64	274	0.50
'quartc'	10000	80	275	0.99	75	233	0.84	96	345	1.25
'bdexp'	500	NaN	NaN	NaN	3	14	0.00	3	14	0.00
'bdexp'	1000	NaN	NaN	NaN	2	7	0.00	2	7	0.00
'bdexp'	5000	NaN	NaN	NaN	3	19	0.03	NaN	NaN	NaN
'bdexp'	10000	NaN	NaN	NaN	3	13	0.11	3	13	0.10
'exdenschnf'	500	42	191	0.05	38	193	0.04	37	189	0.04
'exdenschnf'	1000	42	201	0.05	29	144	0.03	33	164	0.03
'exdenschnf'	5000	36	149	0.17	35	199	0.23	38	204	0.24
'exdenschnf'	10000	42	205	0.45	39	202	4.73	39	216	3.34
'exdenschnb'	500	32	200	0.03	22	102	0.01	26	106	0.02
'exdenschnb'	1000	44	211	0.03	26	122	0.03	23	90	0.01
'exdenschnb'	5000	27	148	0.25	29	137	0.08	18	88	0.06
'exdenschnb'	10000	32	165	0.16	28	115	0.10	36	166	0.14
'genquartic'	500	33	233	0.03	29	107	0.01	26	122	0.01
'genquartic'	1000	34	128	0.02	29	118	0.02	30	151	0.03
'genquartic'	5000	48	150	0.58	30	146	0.17	29	127	0.07
'genquartic'	10000	59	194	0.19	25	115	0.21	25	108	0.13
'biggsb1'	4	14	56	0.01	24	92	0.01	25	115	0.01
'biggsb1'	10	66	125	0.01	87	179	0.01	68	132	0.01
'nonscomp'	8000	97	247	0.17	86	277	0.17	152	398	0.25
'raydan1'	4	23	95	0.01	26	69	0.00	22	60	0.00
'raydan1'	50	71	128	0.01	62	120	0.00	63	120	0.00
'raydan1'	100	99	164	0.01	131	277	0.01	132	298	0.01
'raydan1'	5000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
'raydan2'	500	12	76	0.01	12	65	0.00	12	65	0.01
'raydan2'	1000	19	136	0.02	17	70	0.01	15	64	0.01
'raydan2'	5000	NaN	NaN	NaN	17	85	0.05	21	98	0.05

'raydan2'	10000	21	163	0.16	20	92	0.10	20	92	0.10
'diagonal3'	4	28	95	0.01	21	87	0.00	27	102	0.00
'diagonal3'	10	48	98	0.01	44	152	0.01	49	181	0.01
'bv'	500	250	430	0.53	128	338	0.41	202	416	0.63
'ie'	10	21	89	0.01	15	60	0.01	16	77	0.01
'ie'	100	26	112	0.49	24	138	0.58	25	132	0.60
'ie'	500	25	104	10.36	19	83	8.22	22	107	10.48
'lin'	10	37	198	0.96	13	56	0.24	15	93	0.38
'lin'	100	75	625	5.41	19	113	0.68	20	128	0.87
'lin'	500	47	345	4.16	17	80	0.95	16	98	1.43
'lin'	1000	32	158	135.20	19	118	92.00	19	118	98.02
'pen1'	5	NaN	NaN	NaN	335	999	0.07	275	1041	0.06
'pen1'	10	NaN	NaN	NaN	147	584	0.04	140	576	0.04
'pen1'	100	90	539	0.33	266	856	0.47	245	971	0.77
'pen1'	500	129	591	2.76	182	711	3.39	111	492	2.11

Table 2. Percentage improvement in performance of the proposed (ba1 and ba2) methods relative to the hs method across all test problems.

TOOLS	HS	BA1	BA2
NOI	100%	67.47 %	68.07%
NOF	100%	60.24 %	63.36%
CPUT	100%	57.45%	83.68%

From **Table 2** the results highlight the efficiency and effectiveness of the proposed methods.

1. Number of Iterations

- The BA1 method reduced the number of iterations to **67.47%**, indicating a **32.53% improvement** over the HS method.
- The BA2 method achieved a **31.93% reduction** in iterations, completing the tasks with only **68.07%** of the iterations required by HS.
- These results demonstrate that both proposed methods achieve faster convergence.

2. Number of Function Evaluations

- BA1 required only **60.24%** of the function evaluations compared to HS, showing a **39.76% improvement**.
- BA2 performed slightly more evaluations than BA1 but still required only **63.36%**, resulting in a **36.64% gain** in efficiency.
- This reduction indicates that the proposed methods are more effective in minimizing the objective function with fewer evaluations.

3. CPU Time

- BA1 significantly reduced CPU time usage to **57.45%**, reflecting a **42.55% improvement** over HS.
- BA2 also reduced the time consumption, achieving **83.68%** of HS's execution time, which corresponds to a **16.32% improvement**.
- These improvements are particularly valuable in large-scale optimization problems where computation time is critical.

The data in **Table 2** clearly highlights the superior performance of the proposed BA1 and BA2 methods compared to the classical HS method. In particular:

- **BA1** shows the most substantial improvements across all metrics, making it the most efficient among the tested methods.
- **BA2** also demonstrates strong performance, offering significant enhancements in convergence speed and computational efficiency.

The consistent improvements validate the effectiveness of

integrating modified secant conditions and enhanced conjugate parameters in the optimization process. These results confirm that the proposed Dai–Liao-based methods are well-suited for solving large-scale unconstrained optimization problems efficiently.

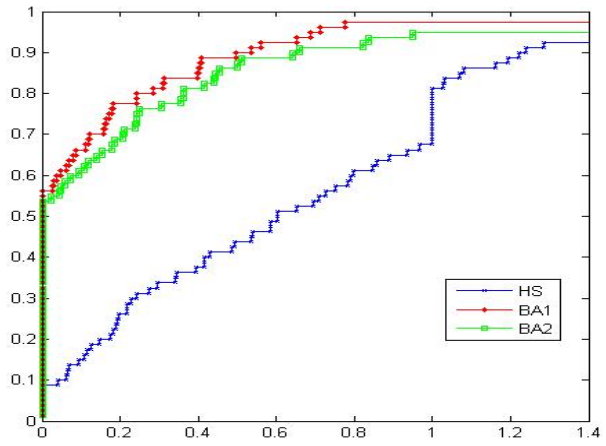

Figure 1: Performance based on NOI.

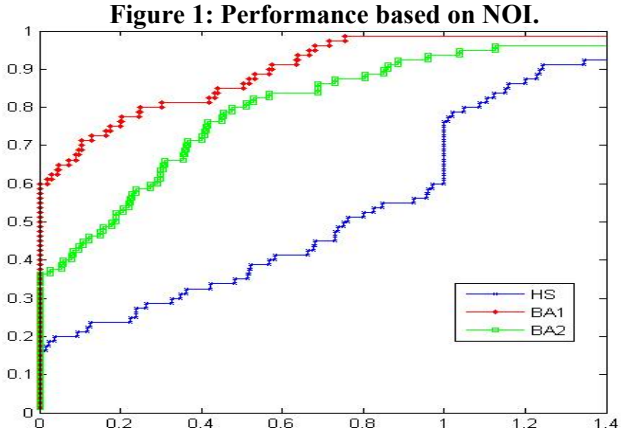

Figure 2: Performance based on NOF.

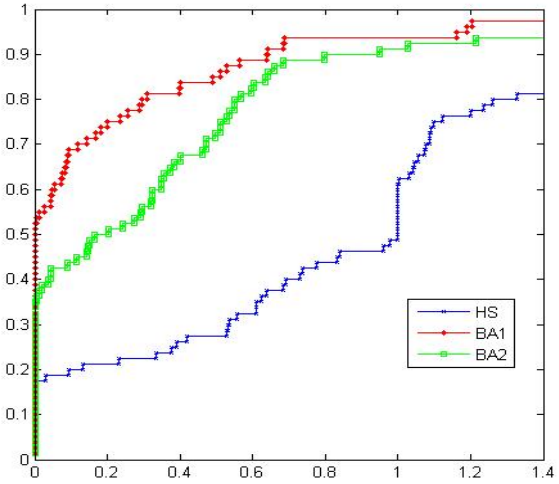

Figure 3: Performance based on CPU.

Conclusion

In this paper, we present two new methods, called BA1 and BA2, to solve large-scale optimization problems faster and better. These methods build upon the Dai–Liao conjugacy condition and are further refined through the application of Taylor series expansion.

We tested them on a lot of tough, well-known test optimization problems with different dimensions. We also compared them to another well-known conjugate gradient method across several critical performance metrics, including a reduction in the number of iterations required for convergence, a decrease in the total number of function evaluations, and an overall faster computation time.

The results were clear: our new methods, BA1 and BA2, were better than the HS method. They needed fewer steps to find the answer, did fewer calculations overall, and even took less time on the computer. What's even cooler is that the BA1 method was the best in everything we measured. These new methods could be good alternatives to what's already out there, and they also give us new ideas for making even better methods in the future.

Acknowledgement

None.

Conflict of interest

The author declares the following potential conflict of interest: ***Basim A. Hassan*** is a member of the editorial board of this journal. However, this manuscript was handled using the journal's standard editorial procedures, independently of the author's role, to ensure an objective and unbiased review process. No other conflicts of interest are declared.

References

[1] Jandaghi, E., Chen, X., & Yuan, C. (2023). Motion dynamics modeling and fault detection of a soft trunk robot. 2023 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM), 1324–1329.

[2] Ebadi, M. J., Fahs, A., Fahs, H., & Dehghani, R. (2023). Competitive secant (BFGS) methods based on modified secant relations for unconstrained optimization. Optimization, 72(7), 1691–1706.

[3] Yuan, G., Zhou, Y., & Zhang, M. (2023). A hybrid conjugate gradient algorithm for nonconvex functions and its applications in image restoration problems. Journal of Operational Research Society of China, 11, 759–781.

[4] Ibrahim, A. L., Fathi, B. G., & Abdulrazzaq, M. B. (2025). Conjugate gradient techniques: Enhancing optimization efficiency for large-scale problems and image restoration. Numerical Algebra, Control and Optimization. https://doi.org/10.3934/naco.2025008

[5] Yin, J., Jiang, X., & Wu, X. (2023). A family of inertial-relaxed DFPM-based algorithms for solving large-scale monotone nonlinear equations with application to sparse signal restoration. Journal of Computational and Applied Mathematics, 419, 114674.

[6] Zhu, Z., Ma, J., & Zhang, B. (2020). A new conjugate gradient hard thresholding pursuit algorithm for sparse signal recovery. Computational and Applied Mathematics, 39, 1–20.

[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 networks. Engineering Reports, 7(1), 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] 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

[10] Avazzadeh, Z., Hassani, H., Agarwal, P., Mehrabi, S., Ebadi, M. J., & Dahaghin, M. S. (2023). An optimization method for studying fractional-order tuberculosis disease model via generalized Laguerre polynomials. Soft Computing, 27(14), 9519–9531.

[11] Avazzadeh, Z., Hassani, H., Agarwal, P., Mehrabi, S., Ebadi, M. J., & Hosseini Asl, M. K. (2023). Optimal study on fractional fascioliasis disease model based on generalized Fibonacci polynomials. Mathematical Methods in the Applied Sciences, 46(8), 9332–9350.

[12] Hanachi, S. B., Sellami, B., & Belloufi, M. (2024). A new family of hybrid conjugate gradient methods for unconstrained optimization and its application to regression analysis. RAIRO-Operations Research, 58, 613–627. https://doi.org/10.1051/ro/2023196

[13] Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. The Computer Journal, 7(2), 149–154.

[14] Dai, Y. H., & Yuan, Y. (1999). A nonlinear conjugate gradient method with a strong global convergence property. SIAM Journal on Optimization, 10(1), 177–182.

[15] Fletcher, R. (2000). Practical Methods of Optimization. John Wiley & Sons.

[16] Polak, E., & Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées. Revue Française d'Information et de Recherche Opérationnelle, Série Rouge, 3(16), 35–43.

[17] Polyak, B. T. (1969). The conjugate gradient method in extremal problems. USSR Computational Mathematics and Mathematical Physics, 9(4), 94–112.

[18] Hestenes, M. R., & Stiefel, E. (1952). Methods of conjugate gradients for solving linear systems. Journal of Research of the National Bureau of Standards, 49(6), 409–435.

[19] Liu, Y., & Storey, C. (1991). Efficient generalized conjugate gradient algorithms, part 1: Theory. Journal of Optimization Theory and Applications, 69, 129–137.

[20] Hassan, B. A. (2012). Development of a special conjugate gradient algorithm for solving unconstrained minimization problems. Rafidain Journal of Computer & Mathematical Sciences, 9, 73–84.

[21] Hassan, B. A., & Sadiq, H. M. (2013). A nonlinear conjugate gradient method based on a modified secant condition. Iraqi Journal of Statistical Sciences, 24, 1–16.

[22] Jahwar, B. H., Ibrahim, A. L., Ajeel, S. M., & Shareef, S. G. (2024). Two new classes of conjugate gradient methods based on logistic mapping. Telkomnika, 22(1), 86–94. https://doi.org/10.12928/TELKOMNIKA.v22i1.25264

[23] Ibrahim, A. L., & Jahwar, B. H. (2022). A new version coefficient of three-term conjugate gradient method to solve unconstrained optimization. New Trends in Mathematical Science, 2022. https://doi.org/10.20852/ntmsci.2022.483

[24] 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, 37, 10381–10405. https://doi.org/10.1007/s00521-025-11121-9

[25] Dai, Y. H., & Liao, L. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods. Applied Mathematics and Optimization, 43, 87–101.

[26] Babaie-Kafaki, S. (2015). On optimality of the parameters of self-scaling memoryless quasi-Newton updating formulae. Journal of Optimization Theory and Applications, 167(1), 91–101.

[27] Hager, W. W., & Zhang, H. (2006). Algorithm 851: CG descent, a conjugate gradient method with guaranteed descent. ACM Transactions on Mathematical Software, 32(1), 113–137.

[28] Arazm, M. R., Babaie-Kafaki, S., & Ghanbari, R. (2017). An extended Dai-Liao conjugate gradient method with global convergence for nonconvex functions. Glasnik Matematički, 52(2), 361–375.

[29] Narushima, Y., & Yabe, H. (2012). Conjugate gradient methods based on secant conditions that generate descent search directions for unconstrained optimization. Journal of Computational and Applied Mathematics, 236(17), 4303–4317.

[30] Dai, Y. H., & Liao, L. Z. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods. Applied Mathematics and Optimization, 43(1), 87–101.

[31] Babaie-Kafaki, S., & Ghanbari, R. (2015). Two optimal Dai–Liao conjugate gradient methods. Optimization, 64(11), 2277–2287.

[32] Andrei, N. (2018). An adaptive scaled BFGS method for unconstrained optimization. Numerical Algorithms, 77(2), 413–432.

[33] Andrei, N. (2018). A Dai-Liao conjugate gradient algorithm with clustering of eigenvalues. Numerical Algorithms, 77(4), 1273–1282.

[34] Babaie-Kafaki, S., & Ghanbari, R. (2014). A descent family of Dai–Liao conjugate gradient methods. Optimization Methods and Software, 29(3), 583–591.

[35] Salihu, N., Odekunle, M. R., Saleh, A. M., & Salihu, S. (2021). A Dai-Liao hybrid Hestenes-Stiefel and Fletcher-Reeves methods for unconstrained optimization. International Journal of Industrial Optimization, 2(1), 33–50.

[36] Babaie-Kafaki, S. (2023). A survey on the Dai-Liao family of nonlinear conjugate gradient methods. RAIRO-Operations Research, 57(1), 43–58.

[37] Aminifard, Z., & Babaie-Kafaki, S. (2022). Dai–Liao extensions of a descent hybrid nonlinear conjugate gradient method with application in signal processing. Numerical Algorithms, 89(3), 1369–1387.

[38] Shareef, S. G., & Ibrahim, A. L. (2016). A new conjugate gradient for unconstrained optimization based on step size of Barzilai and Borwein. Science Journal of University of Zakho, 4(1), 104–114.

[39] Hassan, B. A., & Mohammed, M. I. (2022). Extra quasi-Newton equation for unconstrained optimization. 8th International Conference on Contemporary Information Technology and Mathematics (ICCITM2022), Mosul University, Mosul-Iraq, 375–379.

[40] Hassan, B. A., & Ayoob, A. R. (2022). On the new quasi-Newton equation for unconstrained optimization. 8th International Engineering Conference on Advances in Computer and Civil Engineering Towards Engineering Innovations and Sustainability (IEC-2022), Erbil-Iraq, 168–172.

[41] Perry, A. (1978). Technical note – A modified conjugate gradient algorithm. Applied Mathematics Operations Research, 26(6), 1073–1078.

[42] 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(4), 373–394. https://doi.org/10.1145/962437.962439

[43] Moré, J. J., Garbow, B. S., & Hillstrom, K. E. (1981). Testing unconstrained optimization software. ACM Transactions on Mathematical Software (TOMS), 7(1), 17–41. https://doi.org/10.1145/355934.355936

[44] Andrei, N. (2008). An unconstrained optimization test functions collection. Advanced Modelling and Optimization, 10(1), 147–161.

[45] Dolan, E. D., & Moré, J. J. (2002). Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2), 201–213. https://doi.org/10.1007/s101070100263