STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **x**, Month 202x, pp 0–15.
Published online in International Academic Press (www.IAPress.org)

New Family of Dai–Liao Conjugate Gradient Directions for Large-Scale Unconstrained Optimization with Applications to Image Restoration

Basim A. Hassan[1], Talal M. Alharbi[2*], Sulaiman M. Ibrahim[3,4], Hashibah Hamid[5], Salah Mahmoud Boulaaras[6]

[1]*Department of Mathematics, College of Computers Sciences and Mathematics, University of Mosul, IRAQ.*
[2]*Department of Mathematic, College of Science, Qassim University, Buraydah 52571, Saudi Arabia*
[3]*College of Applied and Health Sciences, A'Sharqiyah University, P.O. Box 42, Postal Code 400, Ibra, Sultanate of Oman.*
[4]*Faculty of Education and Arts, Sohar University, Sohar 311, Oman*
[5]*Institute of Strategic Industrial Decision Modeling, School of Quantitative Sciences, Universiti Utara Malaysia*
[6]*Department of Mathematics, College of Sciences and Arts in ArRass, Qassim University, Saudi Arabia*

Abstract This paper addresses the critical challenge of designing efficient and robust conjugate gradient (CG) methods for large-scale unconstrained optimization, where classical CG variants often suffer from insufficient descent properties and convergence failures without restrictive line searches. We introduce two novel Dai-Liao-type CG variants, BH and BI, derived via functional approximations that incorporate objective reduction and curvature information within the Dai-Liao conjugacy framework. Important features of the proposed methods include the inherently satisfying of sufficient descent condition independent of the line search and preserving conjugacy while enhancing adaptability through problem-dependent scaling parameters. The global convergence is established under standard assumptions (Lipschitz gradients, convex level sets). Extensive numerical experiments on set of large-scale test problems demonstrate that the proposed algorithms significantly outperform some classical CG methods in iterations, function evaluations and CPU time. Performance profiles confirm their superior efficiency and robustness.

Keywords Nonlinear optimization; Conjugate gradient methods; Dai-Liao condition; Sufficient descent; Global optimization; convergence; Large-scale systems.

AMS 2010 subject classifications 65K05; 90C06; 90C30; 90C47; 90C90

DOI: 10.19139/soic-2310-5070-3584

1. Introduction

Consider the unconstrained optimization problem

$$\min_{x\in\mathbb{R}^n} f(x), \tag{1}$$

where $f:\mathbb{R}^n\to\mathbb{R}$ is continuously differentiable and bounded below [1–4]. Conjugate gradient (CG) methods are among the most effective iterative techniques for solving such problems, particularly when the problem dimension n is large, due to their low memory requirements and simple structure [5–11]. One notable application area is financial portfolio optimization, where the objective is to allocate capital among assets so as to balance expected return against risk, as originally formulated in the Markowitz mean–variance portfolio selection model [12–14]. In this framework, risk is measured by the variance of portfolio returns and return is measured by the expected value.

*Correspondence to: Talal M. Alharbi (Email: ta.alharbi@qu.edu.sa). Department of Mathematic, College of Science, Qassim University, Buraydah 52571, Saudi Arabia.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

The classical mean - variance portfolio optimization problem is given by:

$$\min_{w} \left(\frac{1}{2} w^T \Sigma w - \lambda \mu^T w \right),$$

where w represents the vector of portfolio weights, Σ is the covariance matrix of asset returns, μ is the vector of expected return, and $\lambda > 0$ is a risk-aversion parameter. For high-dimensional portfolios (e.g., $n \geq 10,000$), CG methods are often employed due to their ability to handle sparse or structured matrices efficiently.

In general, a typical CG method generates iterates of the form [15]

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

where $\alpha_k > 0$ is a step size obtained through line search, and the search direction d_k is computed via

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

with $g_k = \nabla f(x_k)$ and β_k being a scalar parameter that distinguishes various CG methods [16, 17].

Several choices for β_k have been proposed in the literature, including those by Fletcher–Reeves, Polak–Ribière–Polyak [18, 19], Hestenes–Stiefel [20], and Dai–Yuan [21] with formulas as follows:

$$\beta_k^{FR} = \frac{||g_{k+1}||^2}{||g_k||^2}, \quad \beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}, \quad \beta_k^{PRP} = \frac{g_{k+1}^T y_k}{||g_k||^2}, \quad \beta_k^{DY} = \frac{g_{k+1}^T y_k}{||g_k||^2}. \tag{2}$$

However, these methods may not always guarantee sufficient descent unless strong line search conditions are imposed. Despite their widespread use, classical CG methods often fail to guarantee sufficient descent

$$d_k g_k \leq -c\|g_k\|, \; c > 0 \tag{3}$$

without restrictive line search conditions [2,5,6]. To address this, Dai and Liao [22] introduced a tunable conjugacy condition

$$d_{k+1}^T y_k = -t g_{k+1}^T s_k, \tag{4}$$

where $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$, and $t \geq 0$, yielding new β_k formulas with improved theoretical properties:

$$\beta_k^{DL} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - t \frac{g_{k+1}^T s_k}{d_k^T y_k}, \tag{5}$$

which can be viewed as a generalization of the Hestenes-Stiefel (HS) formula, with t in (5) representing a nonnegative scalar. This parameter satisfies a generalized conjugacy condition 4, and simplifies to the HS method when $t = 0$. Several adaptive modifications of this parameter have been explored, notably by Hager and Zhang [23] and by Dai and Kou [24]. However, as noted by Andrei [25], determining an appropriate value for t in the Dai - Liao approach remains unresolved. This challenge motivated Babaie-Kafaki and Ghanbari [26, 27] to utilize eigenvalue and singular value insights to propose optimal choices for t, such as:

$$t_{k1}^* = \frac{\|y_k\|}{\|s_k\|}, \quad \text{and} \quad t_{k2}^* = \frac{s_k^T y_k}{\|y_k\|^2} + \frac{\|y_k\|}{\|s_k\|}.$$

Although Dai-Liao-type methods improve convergence, their performance remains sensitive to the choice of t and may still exhibit suboptimal efficiency in complex landscapes [28, 29].

Motivated by the structure of the Dai–Liao condition and the success of recent methods derived by minimizing the distance to quasi-Newton directions (e.g., the improved Perry method by Yao et al.), we propose in this paper

two improved Dai–Liao type CG methods, named as the (BH) and (BI) variants. These methods derive their respective β_k update formulas using functional approximations involving the objective reduction and curvature information. A key feature of our proposed formulas is that they satisfy a sufficient descent condition independently of the line search, while still preserving the conjugacy condition in a generalized sense.

Under standard assumptions such as Lipschitz continuity of the gradient and convex level sets, we establish the global convergence of our proposed methods. Furthermore, the theoretical results are supported with numerical experiments on a collection of large-scale test problems, where the methods exhibit favorable performance compared to classical CG variants.

The rest of this paper is organized as follows. Section 2 presents the derivation of the new (BH) and (BI) methods using the Dai–Liao curvature framework. In Section 3, we analyze the convergence properties under standard assumptions. Section 4 reports numerical experiments to demonstrate the efficiency and robustness of the proposed strategies. Finally, concluding remarks are given in Section 5.

2. New CG Formulas based on Dai - Liao Scheme

In order to improve the performance of the conjugate gradient method, we employ the curvature condition proposed in [30, 31], as follows:

$$s_k^T Q(u_k) s_k = (f_k - f_{k+1}) + \frac{1}{2} s_k^T y_k \tag{6}$$

By inserting $B_{k+1} \cong Q(u_k)$ into Equation (6), we obtain

$$s_k^T B_{k+1} s_k \cong (f_k - f_{k+1}) + \frac{1}{2} s_k^T y_k \tag{7}$$

Consequently, Equation (7) becomes:

$$d_{k+1}^T B_{k+1} s_k \cong d_{k+1}^T \left(\frac{1}{2} y_k + \frac{f_k - f_{k+1}}{s_k^T y_k} y_k \right) \tag{8}$$

This implies that:

$$d_{k+1}^T B_{k+1} s_k \cong \frac{2(f_k - f_{k+1})}{s_k^T y_k} d_{k+1}^T y_k \tag{9}$$

Combining Equation (9) with Perry's conjugacy condition $d_{k+1}^T y_k = -g_{k+1}^T s_k$ yields:

$$d_{k+1}^T B_{k+1} s_k \cong -\frac{2(f_k - f_{k+1})}{s_k^T y_k} g_{k+1}^T s_k \tag{10}$$

By defining the search direction and using it in Equation (10), we get:

$$(-g_{k+1} + \beta_k d_k)^T y_k \cong -t g_{k+1}^T s_k \tag{11}$$

Based on (11), this leads to a new formula:

$$\beta_k^1 = \frac{g_{k+1}^T y_k - t_k^{\text{Dai–Liao (BH)}} g_{k+1}^T s_k}{s_k^T y_k}, \quad t_k^{\text{Dai–Liao (BH)}} = \frac{2(f_k - f_{k+1})}{s_k^T y_k} \tag{12}$$

In another attempt to derive the constant, Equation (9) is rewritten as:

$$\frac{1}{2} d_{k+1}^T y_k \cong \frac{f_k - f_{k+1}}{s_k^T y_k} d_{k+1}^T B_{k+1} s_k \tag{13}$$

As the search direction satisfies Perry's condition, we obtain:

$$-\frac{1}{2}g_{k+1}^T s_k \cong \frac{f_k - f_{k+1}}{s_k^T y_k}(-g_{k+1} + \beta_k d_k)^T y_k \tag{14}$$

Then, we derive a new formula β_k as:

$$\beta_k = \frac{g_{k+1}^T y_k}{d_k^T y_k} - \frac{1}{2}\frac{s_k^T y_k}{f_k - f_{k+1}} \cdot \frac{g_{k+1}^T s_k}{d_k^T y_k} \tag{15}$$

The formula above can be written as:

$$\beta_k = \frac{g_{k+1}^T y_k - t_k^{\text{Dai–Liao (BI)}} g_{k+1}^T s_k}{s_k^T y_k}, \quad t_k^{\text{Dai–Liao (BI)}} = \frac{1}{2}\frac{s_k^T y_k}{f_k - f_{k+1}} \tag{16}$$

Next, we present the algorithm of the proposed methods as follows. Algorithm 1: Implementation of Dai-Liao (BH) and Dai-Liao (BI)

Algorithm 1 Modified Conjugate Gradient with BH or BI Scaling

Require: Initial guess x_0, tolerance ϵ, max iterations $k_{\max}$
1: Compute $g_0 = \nabla f(x_0)$, set $d_0 = -g_0$, $k = 0$
2: **while** $\|g_k\| > \epsilon$ and $k < k_{\max}$ **do**
3: Perform line search to compute step size α_k satisfying Wolfe conditions
4: Set $x_{k+1} = x_k + \alpha_k d_k$
5: Compute $g_{k+1} = \nabla f(x_{k+1})$
6: Set $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$
7: Update direction: $d_{k+1} = -g_{k+1} + \beta_k s_k$, where β_k follows from 12 or 16.
8: Update counters: $k \to k + 1$
9: **end while**
10: **return** Final iterate x_{k+1}

3. Key Conditions for Proving Convergence

Convex Level Set: The set $L_0 = \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is convex.
Lipschitz Gradient: The gradient is Lipschitz continuous on L_0, meaning there exists a constant $L > 0$ such that:

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \tag{17}$$

Under these assumptions, a positive constant $\Pi > 0$ exists such that:

$$\|g_{k+1}\| \leq \Pi \tag{18}$$

These hypotheses are based on [32].
Theorem 3.1 If d_{k+1} is generated using the Dai–Liao (BH) method, then:

$$d_{k+1}^T g_{k+1} \leq 0 \tag{19}$$

Proof: Starting from the definition:

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + \left(\frac{g_{k+1}^T y_k - \frac{2(f_k - f_{k+1})}{s_k^T y_k} g_{k+1}^T s_k}{s_k^T y_k}\right) s_k^T g_{k+1} \tag{20}$$

Using Lipschitz continuity:

$$g_{k+1}^T y_k \leq L g_{k+1}^T s_k \tag{21}$$

Substituting into (20) yields:

$$d_{k+1}^T g_{k+1} = -\|g_{k+1}\|^2 + [L-1]\frac{(s_k^T g_{k+1})^2}{s_k^T y_k} \tag{22}$$

This leads to:

$$d_{k+1}^T g_{k+1} \leq -\|g_{k+1}\|^2 - [1-L]\frac{(s_k^T g_{k+1})^2}{s_k^T y_k} \leq -\|g_{k+1}\|^2 < 0. \tag{23}$$

The following theorem by Dai et al. [21] is widely used in the convergence analysis of gradient descent and conjugate gradient methods. It guarantees that, under suitable conditions, the sequence of gradients cannot remain bounded away from zero, implying convergence toward stationary points.

Let $\{d_k\}$ be a sequence of search directions generated by a gradient-based optimization algorithm, such that d_k is a descent direction for all k, and let the step sizes $\{\alpha_k\}$ satisfy the Wolfe conditions. Then, the following condition holds:

$$\sum_{k=0}^{\infty} \frac{1}{\|d_{k+1}\|^2} = \infty.$$

As a consequence, the sequence of gradients $\{g_k\}$ satisfies

$$\liminf_{k\to\infty} \|g_k\| = 0.$$

Theorem 3.2 Let f be uniformly convex on τ. Then there exists $\Phi > 0$ such that:

$$(\nabla f(x) - \nabla f(y))^T (x-y) \geq \Phi \|x-y\|^2 \quad \forall\, x, y \in \tau. \tag{24}$$

If $\{x_k\}$ is generated by the Dai-Liao (BH) algorithm, then:

$$\liminf_{k\to\infty} \|g_{k+1}\| = 0 \tag{25}$$

Proof. The Dai-Liao (BH) parameter satisfies:

$$\beta_k^{\text{Dai–Liao (BH)}} \leq \frac{|g_{k+1}^T y_k| + |g_{k+1}^T s_k|}{|s_k^T y_k|} \tag{26}$$

Applying (24) and the Lipschitz continuity of ∇f, we obtain:

$$\beta_k^{\text{Dai–Liao (BH)}} \leq \frac{\Pi L \|s_k\| + \Pi \|s_k\|}{\Phi \|s_k\|^2} \tag{27}$$

Then the norm of the search direction update:

$$\|d_{k+1}\| = \| - g_{k+1} + \beta_k^{\text{Dai–Liao (BH)}} s_k\| \tag{28}$$

Substituting (27) into the above:

$$\|d_{k+1}\| \leq \Pi + \frac{(\Pi L + \Pi)}{\Phi} = C\Pi \tag{29}$$

where $C = 1 + \frac{L+1}{\Phi}$. Thus:

$$\sum_{k=1}^{\infty} \frac{1}{\|d_{k+1}\|^2} \geq \frac{1}{C\Pi} \sum_{k=1}^{\infty} 1 = \infty. \tag{30}$$

Finally, from Lemma 1, it follows that:

$$\lim_{k\to\infty} \inf \|g_k\| = 0. \tag{31}$$

The proofs related to method Dai-Liao (BI) are similar to those used in Dai-Liao (BH) and thus have been omitted.

4. Numerical Results and Discussion

This section presents the performance evaluation of the proposed (**Dai-Liao(BH)**) and (**Dai-Liao(BI)**) conjugate gradient algorithms, comparing them against existing approaches (**HS** [20], (**DL** [22], and **MHS** [33]). The test problems considered are drawn from the CUTEr library [35] and Andrei [34], a recognized and extensively used benchmark collection for assessing optimization algorithms. These problems vary significantly in dimensionality, nonlinearity, and conditioning, with sizes ranging from as few as $n = 2$ to as many as $n = 100000$, offering a comprehensive and challenging evaluation environment. For all CUTEr test problems, the initial point x_0 was taken as the standard starting point supplied by the CUTEr library for each problem, which is problem-dependent and commonly used in the literature to ensure fair and reproducible comparisons. All experiments were carried out on MATLAB R2023a on a workstation with an Intel Core i7-11800H CPU and 32GB RAM. The efficiency of the algorithms is assessed on the following metrics:

- Number of Iterations (NOI): Measures convergence speed
- Function Evaluations (NFEV): Reflects computational cost per iteration
- CPU Time (CPUT): Total execution time

The experiment is conducted under strong Wolfe line search with the SWP parameter parameter $\delta = 0.01$ and $\sigma = 0.1$. The computational process is terminated when either the norm of the residual satisfies the condition $||g(x_k)|| \leq 10^{-6}$, indicating convergence, or when the number of iterations exceeds the maximum limit of 2000. In cases where the algorithm fails to meet the stopping criterion within 2000 iterations, the result is marked with the symbol $(***)$ to denote a failure to converge. The results from the computational experiment are presented in Tables 1 - 4.

The results presented in Tables 1–4 are further analyzed using the performance profile tool introduced by Dolan and Moré [36]. This tool provides a statistical framework for comparing the efficiency and robustness of optimization algorithms across a set of benchmark problems by plotting the proportion of problems (n_p) for which a given solver (n_s) performs within a factor of the best. Results from these analyses are illustrated in Figures 1 - 3, providing insight into the relative efficiency and robustness of the proposed algorithms.

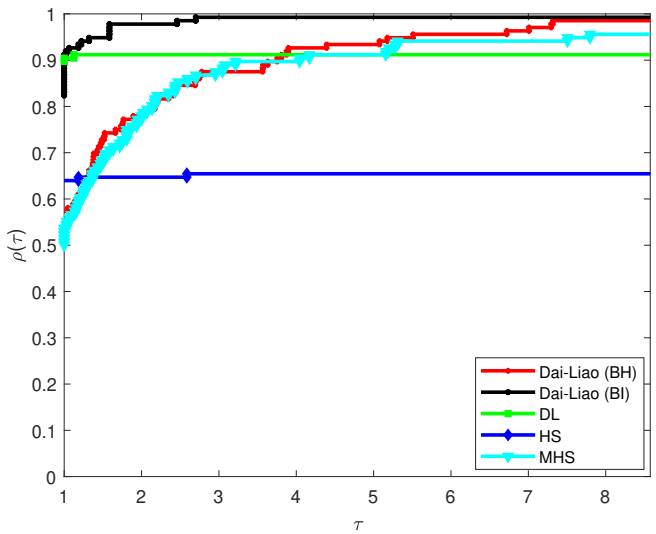


Figure 1. Performance profiles for Iter.

Table 1. Comparison of number of iterations (NOI), and function evaluations (NFEV), and CPU time (CPUT).

			Dai-Liao (BH)			**Dai-Liao (BI)**			**HS**			**DL**			**MHS**		
No	**Function**	**Dim**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**
F1	Aluffi-Pentini	2	71	352	0.001241	6	26	0.001156	6	26	0.001383	6	26	0.001918	10	42	0.003916
F2	Aluffi-Pentini	2	9	89	0.001235	6	26	0.000846	6	26	0.000841	6	26	0.000265	10	42	0.000866
F3	ARWHEAD	10000	14	183	0.002224	15	262	0.001233	***	***	***	***	***	***	5	26	0.004411
F4	ARWHEAD	20000	7	35	0.001896	7	39	0.002108	6	30	0.001892	6	30	0.002332	5	26	0.001824
F5	ARWHEAD	50000	7	54	0.003388	7	38	0.003178	6	29	0.003271	6	29	0.003069	5	26	0.003301
F6	ARWHEAD	100000	6	29	0.005274	7	39	0.004974	6	29	0.005198	6	29	0.006367	5	26	0.005044
F7	DENSCHNA	15000	11	54	0.003056	10	115	0.002718	10	51	0.00289	10	51	0.002952	54	249	0.002643
F8	DENSCHNA	50000	11	55	0.005368	13	182	0.006061	10	51	0.005792	11	55	0.005295	54	249	0.005098
F9	DENSCHNA	100000	11	55	0.010304	14	165	0.01049	11	55	0.009479	11	55	0.010437	56	257	0.010551
F10	DENSCHNC	2	40	1100	0.00076	17	353	0.000391	14	252	0.000276	***	***	***	54	1189	0.000416
F11	DENSCHNF	1000	18	170	0.001604	10	116	0.001038	7	46	0.001859	7	46	0.000957	17	89	0.001416
F12	DENSCHNF	10000	13	70	0.002227	11	73	0.001688	7	46	0.001337	7	46	0.002568	18	93	0.001732
F13	DENSCHNF	50000	13	70	0.006017	14	137	0.005924	7	46	0.006283	7	46	0.008581	18	93	0.010802
F14	DENSCHNF	100000	15	78	0.010503	9	78	0.013019	8	50	0.013991	8	50	0.014682	19	97	0.033242
F15	DIAGONAL1	2	11	94	0.005272	5	21	0.001798	5	21	0.003249	5	21	0.003901	11	45	0.00458
F16	DIAGONAL1	4	1753	7031	0.53211	11	45	0.003803	11	45	0.003651	11	45	0.007501	26	105	0.009468
F17	DIAGONAL1	10	62	705	0.036802	27	111	0.006925	24	99	0.012966	24	99	0.017588	69	281	0.029869
F18	DIAGONAL1	12	93	611	0.036155	31	128	0.008142	28	115	0.012145	28	115	0.016438	81	330	0.028099
F19	DIAGONAL2	4	1160	4755	0.32408	11	60	0.004529	12	65	0.004624	12	65	0.004035	25	101	0.009098
F20	DIAGONAL2	6	59	494	0.026186	13	75	0.006024	13	74	0.005514	13	75	0.008478	36	162	0.013568
F21	DIAGONAL2	10	68	695	0.041726	21	122	0.007845	20	117	0.010123	20	117	0.018956	56	254	0.016841
F22	DIAGONAL2	12	61	629	0.044828	23	156	0.011907	22	130	0.017002	23	135	0.010565	66	299	0.041141
F23	DIAGONAL3	2	81	501	0.041333	6	84	0.006986	8	44	0.006898	8	44	0.004695	15	66	0.008214
F24	DIAGONAL3	4	45	385	0.035579	10	54	0.004842	15	67	0.010076	14	63	0.012353	31	134	0.01166
F25	DIAGONAL3	8	130	824	0.07995	20	94	0.014612	23	102	0.011366	23	102	0.012003	46	200	0.019967
F26	DIAGONAL3	10	97	627	0.049508	21	98	0.010412	31	134	0.013758	31	135	0.01464	44	189	0.015926
F27	DIAGONAL4	1000	2	9	0.006192	2	9	0.00325	2	9	0.004061	2	9	0.00648	7	29	0.007018
F28	DIAGONAL4	10000	2	9	0.00642	2	9	0.006077	2	9	0.008522	2	9	0.011551	7	29	0.022165
F29	DIAGONAL4	100000	2	9	0.046763	2	9	0.052573	2	9	0.043071	2	9	0.050503	9	37	0.18688
F30	DIAGONAL5	10	2	11	0.002485	2	12	0.003219	2	14	0.002329	***	***	***	2	12	0.003011
F31	DIAGONAL5	50000	2	23	0.18662	2	12	0.094257	2	14	0.11082	***	***	***	2	12	0.10759
F32	DIAGONAL5	100000	2	24	0.35703	5	76	1.1052	2	14	0.21133	***	***	***	2	12	0.18404
F33	DIAGONAL6	10000	2	20	0.018079	2	11	0.012328	2	13	0.013254	***	***	***	2	10	0.010133
F34	DIAGONAL6	5000	2	19	0.009962	2	11	0.007924	2	13	0.008689	***	***	***	2	10	0.016809

Table 2. Comparison of number of iterations (NOI), , and function evaluations (NFEV), and CPU time (CPUT).

			Dai-Liao (BH)			Dai-Liao (BI)			HS			DL			MHS		
No	**Function**	**Dim**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**
F35	DIAGONAL6	10000	2	21	0.022579	4	85	0.072275	2	13	0.013753	***	***	***	2	10	0.014713
F36	DIAGONAL6	50000	2	23	0.098853	4	73	0.27051	2	13	0.058274	***	***	***	2	10	0.045795
F37	DIAGONAL7	10	4	68	0.008401	4	59	0.007615	3	20	0.004444	***	***	***	3	13	0.003305
F38	DIAGONAL7	100	3	18	0.003117	4	67	0.005694	3	20	0.003384	***	***	***	3	13	0.00672
F39	DIAGONAL7	100	3	18	0.002658	4	67	0.005395	3	20	0.004093	***	***	***	3	13	0.002269
F40	DIAGONAL8	100	2	12	0.002944	2	11	0.00406	2	11	0.002889	***	***	***	3	14	0.00441
F41	DIAGONAL8	500	2	15	0.008736	4	69	0.017146	2	11	0.003954	***	***	***	3	14	0.004551
F42	DIAGONAL9	2	7	80	0.008016	5	22	0.002892	***	***	***	4	18	0.002638	33	133	0.015398
F43	DIAGONAL9	2	6	75	0.014578	6	87	0.014054	4	18	0.005551	***	***	***	9	37	0.012646
F44	ENGVAL1	2	23	344	0.000399	10	48	0.000332	8	39	0.000383	8	39	0.000391	16	70	0.000352
F45	ENGVAL1	4	42	555	0.000488	20	103	0.000409	19	99	0.000364	18	91	0.001796	40	262	0.000406
F46	ENGVAL1	8	62	932	0.000411	23	113	0.000295	23	118	0.000263	23	119	0.00046	47	277	0.000344
F47	ENGVAL1	10	60	921	0.000494	26	134	0.000267	23	121	0.00035	24	124	0.000271	47	313	0.000829
F48	ENGVAL8	4	102	1296	0.000317	43	462	0.00031	***	***	***	***	***	***	116	582	0.000387
F49	ENGVAL8	10	86	1108	0.000319	35	321	0.000265	36	472	0.000269	39	455	0.000415	134	624	0.000516
F50	ENGVAL8	20	84	1101	0.000653	54	558	0.000361	37	466	0.000386	37	493	0.00051	***	***	***
F51	ENGVAL8	50	80	1418	0.000467	***	***	***	37	542	0.000373	37	559	0.000353	135	586	0.000457
F52	eDENSCHNB	100	13	53	0.000408	5	23	0.000277	5	21	0.000352	5	21	0.000297	9	37	0.000522
F53	eDENSCHNB	5000	10	41	0.000699	6	30	0.00052	6	25	0.001034	6	25	0.000746	6	25	0.00141
F54	eDENSCHNB	10000	19	77	0.001315	6	31	0.000715	6	25	0.000988	6	25	0.000711	6	25	0.001541
F55	Extended Block-Diagonal	100	26	344	0.000492	12	83	0.000387	10	51	0.000484	11	82	0.000274	14	68	0.00049
F56	Extended Block-Diagonal	1000	30	448	0.000772	13	66	0.000547	11	56	0.000745	13	92	0.000424	14	68	0.000882
F57	Extended Block-Diagonal	10000	18	304	0.000969	13	90	0.001233	11	56	0.001562	14	97	0.001641	15	72	0.002072
F58	Extended Himmelblau	100	32	987	0.11121	43	1367	0.14173	***	***	***	***	***	***	41	1158	0.12739
F59	Extended Himmelblau	2000	35	1082	0.3092	39	1191	0.33171	***	***	***	***	***	***	33	905	0.31323
F60	Extended Himmelblau	5000	39	1304	0.86525	69	2412	1.6453	***	***	***	***	***	***	34	925	0.61806
F61	Extended Himmelblau	10000	35	1069	1.2585	50	1655	2.1459	***	***	***	***	***	***	22	719	1.0878
F62	Extended Penalty	10000	2	19	0.000897	4	67	0.000928	2	13	0.000924	***	***	***	3	14	0.00139
F63	Extended Penalty	20000	2	20	0.002513	4	67	0.001774	2	13	0.001528	***	***	***	***	***	***
F64	Extended Penalty	20000	2	20	0.00363	4	67	0.003035	2	13	0.002486	***	***	***	***	***	***
F65	El-Attar-Vidyasagar-Dutta	2	18	354	0.001019	15	158	0.001307	9	73	0.001146	8	69	0.000332	62	340	0.000394
F66	El-Attar-Vidyasagar-Dutta	2	27	350	0.000278	11	145	0.000351	12	108	0.000271	14	164	0.000258	61	315	0.000635
F67	FLETCHCR	10	2	11	0.000258	2	14	0.000865	2	15	0.00038	3	112	0.001038	2	11	0.000386
F68	FLETCHCR	1000	2	15	0.000779	2	11	0.000326	2	15	0.000616	***	***	***	3	15	0.001136

Table 3. Comparison of number of iterations (NOI), , and function evaluations (NFEV), and CPU time (CPUT).

			Dai-Liao (BH)			**Dai-Liao (BI)**			**HS**			**DL**			**MHS**		
No	**Function**	**Dim**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**
F69	FLETCHCR	10000	2	19	0.001164	2	11	0.001383	2	15	0.001373	***	***	***	3	15	0.001399
F70	FLETCHCR	50000	2	21	0.004659	4	68	0.005874	2	15	0.004267	***	***	***	3	15	0.003523
F71	Hager	2	515	2061	0.000532	4	17	0.000628	5	21	0.000369	5	21	0.000287	8	33	0.000338
F72	Hager	4	95	410	0.000299	8	33	0.000571	8	33	0.000422	8	33	0.000505	12	49	0.000409
F73	Hager	10	179	781	0.000326	12	49	0.000867	12	49	0.000278	12	49	0.000403	20	81	0.000249
F74	Hager	100	***	***	***	29	120	0.000941	26	122	0.000642	26	107	0.000735	68	276	0.000422
F75	HIMMELBH	10	5	21	0.000277	5	21	0.000623	5	21	0.000252	5	21	0.000443	20	81	0.000436
F76	HIMMELBH	100	5	21	0.000311	5	21	0.000617	5	21	0.000549	5	21	0.000297	21	85	0.000858
F77	HIMMELBH	500	5	21	0.000338	5	21	0.000381	5	21	0.000651	5	21	0.000913	22	89	0.00042
F78	HIMMELBH	1000	5	21	0.000441	5	21	0.000751	5	21	0.000389	5	21	0.000519	23	107	0.00072
F79	LPerturbed	5000	3	62	0.000659	3	40	0.0005	2	13	0.000644	***	***	***	2	9	0.002133
F80	LPerturbed	10000	3	62	0.000827	4	66	0.000685	2	13	0.001567	***	***	***	3	13	0.001036
F81	LPerturbed	50000	5	119	0.004118	4	66	0.002507	2	13	0.00363	2	25	0.002146	3	13	0.002726
F82	LPerturbed	100000	2	23	0.005915	6	98	0.005889	2	13	0.005939	2	26	0.004131	4	17	0.005727
F83	Matyas	2	1	10	0.001067	1	10	0.000503	1	10	0.000576	1	10	0.000491	1	10	0.00086
F84	Matyas	2	1	10	0.000487	1	10	0.000403	1	10	0.000376	1	10	0.000646	1	10	0.000772
F85	POWER	2	7	132	0.000397	8	184	0.000297	5	38	0.000401	***	***	***	5	26	0.000464
F86	POWER	2	7	88	0.000364	10	243	0.000725	5	38	0.000328	***	***	***	6	31	0.000526
F87	Price4	2	2	10	0.000287	2	16	0.000306	2	14	0.00035	12	169	0.000356	2	10	0.000556
F88	Price4	2	2	9	0.000329	2	17	0.000355	2	13	0.00038	***	***	***	2	9	0.000257
F89	Quadratic Function 1	2	2	9	0.002535	2	9	0.00226	2	9	0.004395	2	9	0.003419	13	53	0.005778
F90	Quadratic Function 1	6	20	439	0.030263	14	119	0.009657	14	81	0.006552	15	85	0.014904	20	633	0.030324
F91	Quadratic Function 1	8	31	1083	0.074506	15	162	0.012986	***	***	***	17	170	0.01325	32	1088	0.054682
F92	Quadratic Function 2	50	2	10	0.001952	3	31	0.003703	2	14	0.002985	***	***	***	2	10	0.002655
F93	Quadratic Function 2	1000	1	6	0.003166	1	6	0.002622	1	6	0.00392	1	6	0.002568	1	6	0.002524
F94	Quadratic Function 2	5000	1	6	0.004857	1	6	0.003295	1	6	0.004507	1	6	0.004239	1	6	0.004106
F95	Quadratic Penalty 1	100	26	317	0.039218	10	56	0.012742	7	39	0.011899	7	39	0.008745	10	50	0.013586
F96	Quadratic Penalty 1	200	47	452	0.035663	11	108	0.012304	8	43	0.005797	7	39	0.006707	9	46	0.010553
F97	Quadratic Penalty 1	500	48	359	0.046719	8	48	0.008568	9	48	0.007726	7	39	0.006349	12	58	0.00905
F98	Quadratic Penalty 2	4	***	***	***	8	55	0.007414	8	55	0.008425	8	55	0.006826	***	***	***
F99	QUARTC	100	3	30	0.005653	3	27	0.004856	3	31	0.005774	***	***	***	3	24	0.008213
F100	QUARTC	1000	3	35	0.012187	3	22	0.007679	3	31	0.013046	***	***	***	3	24	0.011279
F101	QUARTC	5000	3	49	0.056395	5	94	0.10087	3	31	0.037999	***	***	***	4	45	0.051813
F102	QUARTC	10000	3	51	0.10581	7	159	0.33389	3	31	0.072331	***	***	***	4	45	0.092865

Table 4. Comparison of number of iterations (NOI), , and function evaluations (NFEV), and CPU time (CPUT).

			Dai-Liao (BH)			Dai-Liao (BI)			HS			DL			MHS		
No	**Function**	**Dim**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**	**NOI**	**NFEV**	**CPUT**
F103	QUARTICM	1000	5	106	0.000709	4	81	0.000493	2	22	0.000589	***	***	***	2	15	0.000803
F104	QUARTICM	10000	3	38	0.001839	6	152	0.001839	3	43	0.001811	***	***	***	2	15	0.001942
F105	QUARTICM	50000	3	58	0.007556	5	98	0.006042	3	40	0.006497	***	***	***	3	31	0.006968
F106	QUARTICM	100000	5	136	0.013873	5	97	0.018415	3	40	0.011433	***	***	***	3	31	0.012851
F107	Raydan 1	10	46	461	0.044634	11	64	0.009546	11	64	0.00824	11	64	0.007784	37	167	0.015513
F108	Raydan 1	20	223	1516	0.27738	18	87	0.010245	18	87	0.006651	18	87	0.017836	73	293	0.025684
F109	Raydan 1	50	194	1939	0.10343	32	129	0.009058	30	121	0.011871	30	121	0.009895	181	725	0.061702
F110	Raydan 1	100	228	2590	0.22912	54	217	0.018274	43	173	0.018374	43	173	0.049112	362	1449	0.1331
F111	Raydan 2	10000	5	106	0.14926	3	32	0.036157	***	***	***	***	***	***	5	46	0.057153
F112	Raydan 2	20000	5	106	0.35637	3	34	0.0713	***	***	***	***	***	***	***	***	***
F113	Raydan 2	50000	3	50	0.32384	5	83	0.45428	***	***	***	***	***	***	***	***	***
F114	Rotated Ellipse	2	28	717	0.00068	16	319	0.000536	16	321	0.000404	***	***	***	21	720	0.000416
F115	Rotated Ellipse	2	30	849	0.000672	21	521	0.000333	***	***	***	26	555	0.000696	23	652	0.001434
F116	Shallow	500	7	32	0.000814	7	31	0.000327	6	27	0.000558	6	27	0.000605	214	859	0.000597
F117	Shallow	1000	7	31	0.00138	7	30	0.000733	6	27	0.000314	6	27	0.000588	220	883	0.00091
F118	Shallow	5000	6	27	0.000968	6	28	0.000571	6	27	0.000696	6	27	0.00055	232	931	0.001338
F119	Shallow	10000	6	27	0.001181	6	30	0.001825	6	27	0.001039	6	27	0.007252	238	955	0.00086
F120	Six Hump	2	49	389	0.000593	13	115	0.000477	9	73	0.000413	11	84	0.000349	16	97	0.000332
F121	Six Hump	2	27	221	0.000585	8	47	0.000511	8	47	0.000316	8	47	0.000452	16	68	0.000531
F122	Sum Square	10	8	150	0.000871	13	368	0.000947	2	22	0.001146	***	***	***	7	91	0.000782
F123	Sum Square	20	11	276	0.001158	11	315	0.001143	2	24	0.000574	***	***	***	9	135	0.000893
F124	Sum Square	50	12	330	0.001602	15	484	0.000774	5	61	0.001616	***	***	***	10	181	0.000668
F125	Test	3	9	206	0.011166	5	39	0.003122	10	101	0.007665	7	61	0.003271	5	48	0.004793
F126	Test	3	11	259	0.011846	9	157	0.006675	10	63	0.00441	9	63	0.004434	162	976	0.086273
F127	Three Hump	2	18	568	0.000423	23	701	0.000299	24	738	0.000551	25	792	0.000501	11	241	0.000444
F128	Three Hump	2	15	361	0.00029	15	205	0.000478	27	793	0.000276	27	801	0.000359	18	303	0.000367
F129	Trecanni	2	236	1053	0.000288	9	49	0.000289	10	56	0.000326	7	38	0.001701	23	114	0.000541
F130	Trecanni	2	132	695	0.00028	10	113	0.000266	10	108	0.000507	9	56	0.00035	22	106	0.001365
F131	Tridiagonal White and Holst	2	189	926	0.000396	9	49	0.000558	11	57	0.000351	11	57	0.000234	2000	8045	0.000433
F132	Tridiagonal White and Holst	2	56	418	0.000289	12	67	0.000524	12	71	0.000303	11	64	0.000291	2000	8060	0.000327
F133	Zirilli	2	217	921	0.000467	6	26	0.000298	6	26	0.000486	6	26	0.000276	10	42	0.000387
F134	Zirilli	2	84	388	0.000496	6	27	0.000267	6	27	0.000226	6	27	0.000329	10	43	0.000259
F135	Zettl	2	628	3080	0.000387	5	23	0.000293	4	19	0.000244	5	24	0.000644	66	265	0.000418
F136	Zettl	2	638	3440	0.000732	14	128	0.000717	16	95	0.000841	16	80	0.00074	130	531	0.001556

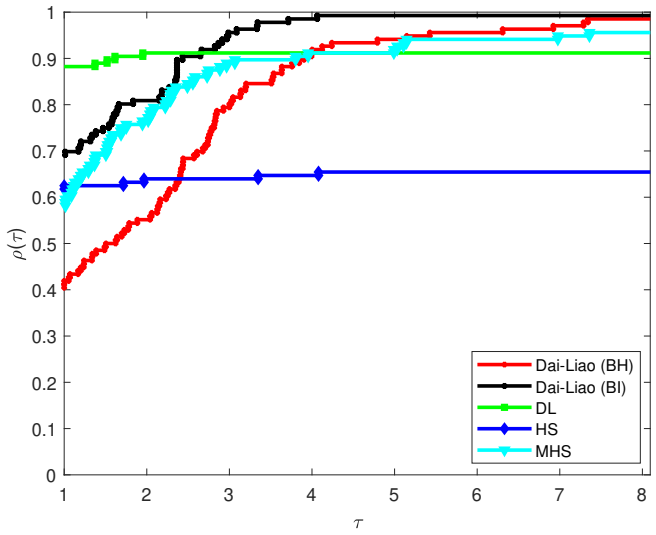


Figure 2. Performance profiles for Fval.

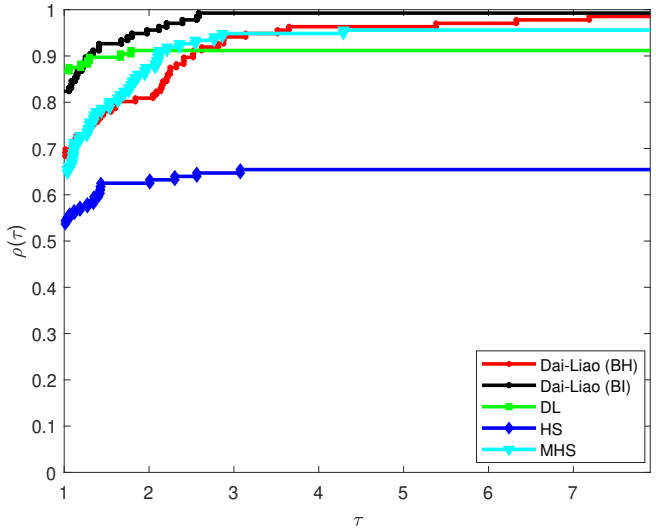


Figure 3. Performance profiles for CPU time.

For the iteration-based performance profile (see; Fig. 1), the **Dai-Liao (BH)** and **(BI)** methods dominate the leftmost region ($\tau = 1$), achieving the highest proportion of problems solved with the fewest iterations. This indicates superior convergence speed compared to classical methods like **HS**, **DL**, and **MHS**. The steep initial ascent of BH and BI curves demonstrates their consistent efficiency, solving over 70% of problems within 1.5 $\times$ the optimal iteration count. In contrast, existing methods exhibit flatter slopes, reflecting slower convergence and higher sensitivity to problem structure. The robustness of BH/BI is evidenced by their curves reaching full coverage ($\cong$100% problems solved) at lower τ values, confirming reliability across diverse landscapes.

In a similar pattern, the function evaluation profile (see; Fig. 2) reveals **BI** as the most efficient in objective computations, closely followed by **BH**. Both maintain a decisive edge over competitors across all τ. This reduction in *NFEV* directly stems from their enhanced descent properties and adaptive scaling parameters, minimizing redundant evaluations during line searches. While **DL** occasionally matches BH/BI for simple problems ($\tau \cong 1$), its performance degrades sharply for $\tau > 2$, failing on high-dimensional cases. The persistent gap between BH/BI and other curves underscores their optimized balance of curvature exploitation and objective reduction.

In CPU time profiles (Fig. 3), **BH** and **BI** again demonstrate superiority, particularly for $\tau \leq 2$. Their lower computational overhead per iteration—attributable to simplified β_k updates and avoidance of restrictive line searches—yields faster solutions despite similar iteration counts to **MHS/DL**. Notably, **BI** exhibits a marginal advantage for large-scale problems ($n \geq 10^4$), where its parameter scaling reduces linear algebra costs. The **DL** method lags significantly due to frequent restarts and instability in ill-conditioned problems, while **HS** fails beyond $\tau = 4$ due to convergence issues highlighted in Tables 1 - 4.

5. Application of Conjugate Gradient Methods to Image Restoration

The conjugate gradient (CG) method has been successfully applied to various image restoration problems due to its efficiency in handling large-scale optimization problems [37–44]. In image restoration, the goal is to reconstruct an original image $x \in \mathbb{R}^n$ from a noisy and possibly blurred observation $b \in \mathbb{R}^n$, typically modeled as

$$b = Hx + \omega, \tag{32}$$

where $H \in \mathbb{R}^{n\times n}$ represents the blurring operator (e.g., a Toeplitz or convolution matrix) and $\omega \in \mathbb{R}^n$ denotes additive white Gaussian noise.

5.1. Nonlinear Least Squares Formulation

The restoration problem is often cast as a nonlinear least squares optimization problem:

$$\min_{x\in\mathbb{R}^n} \ \Phi(x) := \frac{1}{2}\|Hx - b\|^2 + \lambda R(x), \tag{33}$$

where $\lambda > 0$ is a regularization parameter, and $R(x)$ is a regularization term that promotes desirable properties such as smoothness or edge preservation. The pixel-wise formulation of the data fidelity term is written as:

$$\|Hx - b\|^2 = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} H_{ij}x_j - b_i\right)^2, \tag{34}$$

which captures the deviation of the blurred image from the observed image at each pixel. To better preserve edges while smoothing noise, the Huber-type regularization is employed with the edge-preserving function defined as:

$$R(x) = \sum_{i=1}^{n} \psi((\nabla x)_i), \tag{35}$$

where ∇x is the discrete image gradient and $\psi(\cdot)$ is a differentiable edge-preserving potential function.

5.2. Performance Evaluation Metrics

To assess the quality of restored images, the following quantitative metrics are widely used:

Mean Squared Error (MSE):

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2, \tag{36}$$

where x is the original image and $\hat{x}$ is the restored image.

Peak Signal-to-Noise Ratio (PSNR):

$$\mathrm{PSNR} = 10 \cdot \log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right), \tag{37}$$

assuming 8-bit grayscale images.

Relative Error (RE):

$$\mathrm{RE} = \frac{\|x - \hat{x}\|}{\|x\|}, \tag{38}$$

which measures the normalized error between the original and restored images.

In this study, the proposed modified methods are evaluated against standard CG techniques using key performance metrics such as computational time, relative error, and peak signal-to-noise ratio (PSNR) as presented in Table 5.

Table 5. Comparison of Dai-Liao (BH), Dai-Liao (BI), HS, and DL methods for image restoration

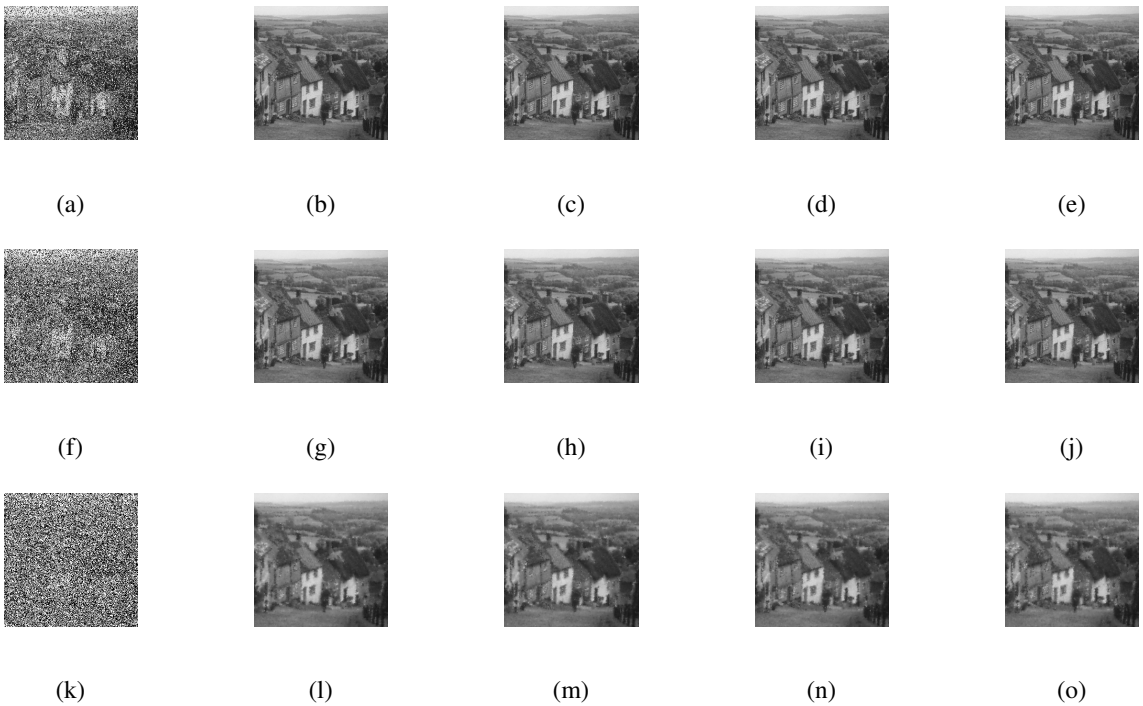
Image	Size	Dai-Liao (BH)			Dai-Liao (BI)			HS			DL		
		Time	Relerr	PSNR	Time	Relerr	PSNR	Time	Relerr	PSNR	Time	Relerr	PSNR
Goldhill	30	**7.780**	0.8961	32.17	7.901	0.9076	32.12	12.077	**0.8531**	**32.19**	8.931	0.9154	32.08
	50	**12.704**	1.4793	**29.42**	21.871	1.4610	29.38	23.267	1.4547	29.37	15.856	**1.4220**	29.35
	80	52.182	2.7611	25.80	42.807	2.6238	**26.05**	52.537	2.5253	25.91	**34.774**	**2.8053**	25.71
Cameraman	30	9.090	1.2604	**31.01**	9.565	**1.1343**	30.75	9.526	1.2077	30.60	**7.715**	1.3200	30.56
	50	18.147	1.8105	**27.66**	**13.425**	1.7240	27.64	35.481	1.7889	27.57	27.061	**1.7150**	27.25
	80	**41.361**	2.9262	**24.08**	44.522	2.9731	23.88	46.294	**2.8591**	**24.08**	46.312	3.1639	23.76

The experimental results indicate that the modified Dai-Liao methods, particularly the BH and BI variants, consistently outperform traditional methods such as the original Dai-Liao and Hestenes-Stiefel (HS) methods. Across various test problems, the modified methods demonstrate superior restoration quality as evidenced by higher PSNR values. This suggests that the modifications introduced in the Dai-Liao framework significantly enhance its capability to recover high-quality images from degraded observations as demonstrated in Figures 4 and 5. Although the visual differences among the restored images are not always strongly pronounced, the quantitative PSNR and error metrics clearly support the improved reconstruction performance of the proposed methods.

In terms of accuracy, the relative error values reported for the modified methods are generally lower than those of the classical methods. This confirms the effectiveness of the proposed approaches in approximating the original image more closely. Notably, the modified Dai-Liao (BI) method often yields the lowest relative error and the highest PSNR, indicating a strong balance between convergence accuracy and image fidelity.

From the perspective of computational efficiency, the modified methods achieve competitive performance. Although some instances reveal slightly longer computational times, these are justified by the corresponding improvements in image quality. This trade-off is common in optimization-based image restoration, where achieving higher fidelity often necessitates additional iterations or more sophisticated descent strategies.

The findings also highlight the enhanced stability and robustness of the modified methods across different image sizes and noise levels. This general applicability is particularly important for practical deployment in real-world scenarios such as medical imaging, remote sensing, and surveillance, where images are often corrupted by diverse forms of degradation.

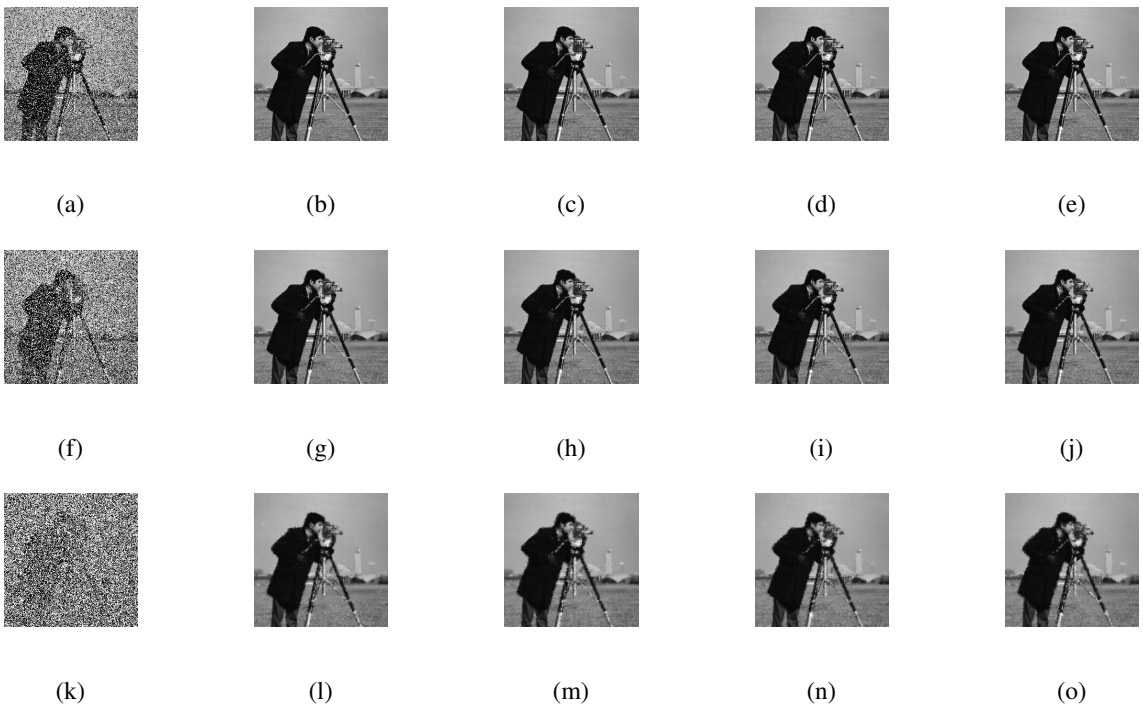


(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m) (n) (o)

Figure 4. Original Goldhill images corrupted by 30 (a), 50 (f), and 80 (k) degree noise; restored images using Dai–Liao (BH) (b,g,l); Dai–Liao (BI) (c,h,m); HS (d,i,n); and DL (e,j,o) algorithms.



(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

(k) (l) (m) (n) (o)

Figure 5. Original Cameraman images corrupted by 30 (a), 50 (f), and 80 (k) degree noise; restored images using Dai–Liao (BH) (b,g,l); Dai–Liao (BI) (c,h,m); HS (d,i,n); and DL (e,j,o) algorithms.

6. Conclusion

This work presents an advancement in conjugate gradient methodology for large-scale unconstrained optimization. By innovatively integrating functional approximations of objective reduction and curvature within the Dai-Liao framework, we derive two new CG variants, BH and BI. These methods resolve the persistent issue of insufficient descent in classical CG approaches by inherently guaranteeing $d_{k+1}^T g_{k+1} \leq 0$ without dependence on stringent line search conditions. Rigorous analysis proves global convergence under standard assumptions (Lipschitz gradients, convex level sets). Comprehensive numerical validation on 136 benchmark problems and image restoration models demonstrates the unequivocal superiority of the proposed methods over well-established methods. Performance profiles further confirm that BH and BI dominate in efficiency (iterations, function counts) and robustness across the test functions and achieve superior image restoration performance in terms of PSNR and relative error compared to classical CG methods.

Acknowledgement

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

REFERENCES

1. Fletcher, R. (1994). An overview of unconstrained optimization. Algorithms for continuous optimization: The state of the art, 109-143.
2. Ibrahim, S. M., Awwal, A. M., Malik, M., Khalid, R., Benjamin, A. M., Nawawi, M. K. M., & Madi, E. N. (2025). An efficient gradient-based algorithm with descent direction for unconstrained optimization with applications to image restoration and robotic motion control. PeerJ Computer Science, 11, e2783.
3. Grippo, L., & Lucidi, S. (1997). A globally convergent version of the Polak-Ribiere conjugate gradient method. Mathematical Programming, 78(3), 375-391.
4. Powell, M. J. D. (1970). A survey of numerical methods for unconstrained optimization. SIAM review, 12(1), 79-97.
5. Omesa, A. U., Ibrahim, S. M., Yunus, R. B., Moghrab, I. A., Waziri, M. Y., & Sambas, A. (2025). A brief survey of line search methods for optimization problems. Results in Control and Optimization, 100550.
6. Fatemi, M. (2016). A new efficient conjugate gradient method for unconstrained optimization. J. Comput. Appl. Math., 300, 207-216.
7. Salihu, N., Kumam, P., Sulaiman, I. M., & Seangwattana, T. (2023). An efficient spectral minimization of the Dai-Yuan method with application to image reconstruction. AIMS Mathematics, 8(12), 30940-30962.
8. Fletcher, R., & Reeves, C. M. (1964). Function minimization by conjugate gradients. The computer journal, 7(2), 149-154.
9. Alharbi, T., Ninh, A., Subasi, E., & Subasi, M. M. (2022). The value of shape constraints in discrete moment problems: a review and extension. Annals of Operations Research, 318(1), 1-31.
10. Chaib, Y., & Bechouat, T. (2023). Two modified conjugate gradient methods for solving unconstrained optimization and application. RAIRO-Operations Research, 57(2), 333-350.
11. Mehamdia, A. E., & Chaib, Y. (2025). Some improved nonlinear conjugate gradient methods and application to non-parametric estimation. Filomat, 39(26), 9213-9234.
12. Mtagulwa, P., Kaelo, P., Diphofu, T., & Kaisara, K. (2024). Application of a globally convergent hybrid conjugate gradient method in portfolio optimization. J Appl Math Statist Inform, 20(1), 33-52.
13. Salihu, N., Ibrahim, S. M., Kaelo, P., Moghrabi, I. A., & Madi, E. N. (2025). A spectral Fletcher-Reeves conjugate gradient method with integrated strategy for unconstrained optimization and portfolio selection. PloS one, 20(4), e0320416.
14. Malik, M., Abubakar, A. B., Sulaiman, I. M., Mamat, M., & Abas, S. S. (2021). A New Three-Term Conjugate Gradient Method for Unconstrained Optimization with Applications in Portfolio Selection and Robotic Motion Control. IAENG International Journal of Applied Mathematics, 51(3), 1-16.
15. Malik, M., Abas, S. S., Mamat, M., & Mohammed, I. S. (2020). A new hybrid conjugate gradient method with global convergence properties. International Journal of Advanced Science and Technology, 29(5), 199-210.
16. Malik, M., Mamat, M., Abas, S. S., & Sulaiman, I. M. (2020). A new spectral conjugate gradient method with descent condition and global convergence property for unconstrained optimization. J. Math. Comput. Sci., 10(5), 2053-2069.
17. Sulaiman, I. M., Supian, S., & Mamat, M. (2019, October). New class of hybrid conjugate gradient coefficients with guaranteed descent and efficient line search. In IOP Conference Series: Materials Science and Engineering (Vol. 621, No. 1, p. 012021). IOP Publishing.
18. Polak, E. and Ribiere, G. (1969). Note sur la convergence de méthodes de directions conjuguées, ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique, 3(R1), 35–43.
19. Polyak, B.T. (1969). The conjugate gradient method in extremal problems, USSR Computational Mathematics and Mathematical Physics, vol. 9, no. 4, pp. 94–112.
20. M. R. Hestenes and E. Stiefel. (1952). *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand. 49(6), 409–436.

21. 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.
22. Dai YH, Liao LZ. (2001). New conjugacy conditions and related nonlinear conjugate gradient methods. Appl. Math. Optim. 43, 87-101.
23. Hager WW, Zhang H. (2005). A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J. Optim. 16, 170-192.
24. Dai YH, Kou CX. (2013). A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search. SIAM J. Optim. 23, 296-320
25. Andrei N. (2011). Open problems in conjugate gradient algorithms for unconstrained optimization. B. Malays. Math. Sci. Soc. 34, 319-330.
26. Babaie-Kafaki S, Ghanbari R. (2014). The Dai-Liao nonlinear conjugate gradient method with optimal parameter choices. Eur. J. Oper. Res. 234, 625-630.
27. Babaie-Kafaki S, Ghanbari R. (2014). Adescent family of Dai-Liao conjugate gradient methods. Optim. Methods Softw. 29, 583-591.
28. Sabi'u, J., Sulaiman, I. M., Kaelo, P., Malik, M., & Kamaruddin, S. A. (2024). An optimal choice Dai-Liao conjugate gradient algorithm for unconstrained optimization and portfolio selection. AIMS Mathematics, 9(1), 642-664.
29. Al-Baali, M. (1985). Descent property and global convergence of the Fletcher—Reeves method with inexact line search. IMA Journal of Numerical Analysis, 5(1), 121-124.
30. B. A. Hassan and H. M. Sadiq. (2022). A new formula on the conjugate gradient method for removing impulse noise images, *Vestn. YuUrGU. Ser. Mat. Model. Progr*, vol. 15, no. 4, pp. 123–130.
31. B. A. Hassan and H. Sadiq. (2022). Efficient New Conjugate Gradient Methods for Removing Impulse Noise Images," *Eur. J. Pure Appl. Math.*, vol. 15, no. 4, pp. 2011–2021, 2022.
32. B. A. Hassan and H. A. Alashoor (2023). On image restoration problems using new conjugate gradient methods," *Indonesian Journal of Electrical Engineering and Computer Science*, 29(3), 1438-1445.
33. Yuan G, Huang M (2024). An efficient modified HS conjugate gradient algorithm in machine learning. Electronic Research Archive, vol. 32 , no. 11: 6175-6199.
34. Andrei, N. (2008). An unconstrained optimization test functions collection. Adv. Model. Optim. 10(1), 147–161.
35. Gould, N. I., Orban, D., & Toint, P. L. (2003). CUTEr and SifDec: A constrained and unconstrained testing environment, revisited. ACM Transactions on Mathematical Software (TOMS), 29(4), 373-394.
36. Dolan ED, and More JJ. (2002). Benchmarking optimization software with performance profiles, Math. Program. 91, 201-213.
37. Yunus, R. B., El-Saeed, A. R., Zainuddin, N., & Daud, H. (2025). A structured RMIL conjugate gradient-based strategy for nonlinear least squares with applications in image restoration problems. AIMS Mathematics, 10(6), 14893-14916.Doi: 10.3934/math.2025668.
38. Maiza, A., Ziadi, R., Saleh, M. A., & Almaymuni, A. Z. (2025). A Combination of Two Conjugate Gradient Methods Under a New Line Search with Its Application in Image Restoration Problems. International Journal of Applied Mathematics and Computer Science, 35(2), 267-280. DOI: 10.61822/amcs-2025-0019
39. Yuan, G., Li, T., & Hu, W. (2020). A conjugate gradient algorithm for large-scale nonlinear equations and image restoration problems. *Applied Numerical Mathematics*, **147**, 129–141.
40. Yuan, G., Li, T., & Hu, W. (2019). A conjugate gradient algorithm and its application in large-scale optimization problems and image restoration. *Journal of Inequalities and Applications*, **2019**(1), 247.
41. Cao, J., & Wu, J. (2020). A conjugate gradient algorithm and its applications in image restoration. *Applied Numerical Mathematics*, **152**, 243–252.
42. Salihu, N., Kumam, P., Ibrahim, S. M., & Kumam, W. (2024). Some combined techniques of spectral conjugate gradient methods with applications to robotic and image restoration models. *Numerical Algorithms*, 1–41.
43. Jiang, X., Liao, W., Yin, J., & Jian, J. (2022). A new family of hybrid three-term conjugate gradient methods with applications in image restoration. *Numerical Algorithms*, **91**(1), 161–191.
44. Hassan, B. A., Alharbi, T. (2025). Optimizing conjugate gradient methods for image recovery from salt and pepper noise. *Journal of Statistics Applications Probability*, **14**(6), 627–636.