Applied Mathematics and Computational Intelligence
Volume 15, No. 3, 2026 [47 – 60]

A Variant of Self-Adjusting Spectral Hybrid Dai-Liao Conjugate Gradient Method

Muhammad Aqiil Iqmal Ishak[1], Nurfaqiha Abu Salim[2*], Siti Mahani Marjugi[3]

[1,2,3]Department of Mathematics and Statistics, Faculty of Science, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia

* Corresponding author: nurfaqiha.nf@gmail.com

Received: 3 November 2025
Revised: 22 May 2026
Accepted: 25 May 2026

ABSTRACT

The Conjugate Gradient method is a popular optimization technique known for its fast convergence and low memory requirements. These advantages make it efficient for solving unconstrained problems of various scales. This study presents a spectral convex combination modification of the Liu Storey method for unconstrained optimization. The method approximates the step size while preserving descent properties, aiming to enhance efficiency and adaptability compared to existing approaches. The proposed method generates search directions that inherently satisfy the sufficient descent property, regardless of the conjugate parameter or line search choice. The study concludes by validating the impact and adaptability of various parameter combinations on the effectiveness of the spectral hybrid method for solving unconstrained optimization problems. Numerical experiments demonstrate the efficiency of the proposed methods in terms of iteration count and CPU time, comparing favorably with current approaches for addressing unconstrained optimization problems.

Keywords: Conjugate gradient method, Spectral convex combination, Unconstrained optimization

1 INTRODUCTION

Optimization refers to the process of identifying the most optimal solution among various alternatives, typically by enhancing efficiency or reducing costs. It is widely applied in fields such as engineering, business, and computer science to enhance performance and decision-making. A fundamental aspect of optimization is unconstrained optimization, where decision variables are not restricted by constraints, allowing for a more straightforward problem-solving approach.

In this research, we consider the unconstrained optimization problem:

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

where $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is continuously differentiable and its gradient is denoted by $g(x) = \nabla f(x)$.

Various optimization methods have been developed to improve efficiency, including the conjugate gradient (CG) method, which is particularly effective for solving large-scale problems due to its ability to avoid matrix

storage and ensure rapid convergence. There are variety of CG methods have been proposed in the literature. Some of them are known as standard CG methods such as Hestenes-Stiefel (HS) by [1] , Fletcher-Reeves (FR) by [2], Polak-Ribiere-Polyak (PRP) by [3], Conjugate Descent (CD) by [4], Liu-Storey (LS) by [5] and Dai-Yuan (DY) by [6]. The various coefficient β_k of CG methods are given as follows:

$$\beta_k^{\text{HS}} = \frac{g_k^T(g_k - g_{k-1})}{(g_k - g_{k-1})^T d_{k-1}}, \quad \beta_k^{\text{FR}} = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}, \quad \beta_k^{\text{PRP}} = \frac{g_k^T(g_k - g_{k-1})}{g_{k-1}^T g_{k-1}},$$

$$\beta_k^{\text{CD}} = -\frac{g_k^T g_k}{d_{k-1}^T g_{k-1}}, \quad \beta_k^{\text{LS}} = -\frac{g_k^T(g_k - g_{k-1})}{d_{k-1}^T g_{k-1}}, \quad \beta_k^{\text{DY}} = \frac{g_k^T g_k}{(g_k - g_{k-1})^T d_{k-1}}.$$

Extensions like the Spectral Conjugate Gradient Method (SCGM) further refine the CG approach by incorporating spectral properties to enhance convergence speed and numerical stability. However, these methods may be sensitive to parameter selection, necessitating careful tuning for optimal performance.The term self-adjusting denotes the adaptive update strategy applied to the spectral parameters in the proposed method, allowing them to adjust dynamically according to the optimization process.

To address these challenges, hybrid approaches have been introduced by integrating different CG methods to leverage their strengths, prevent jamming, and improve convergence properties. [7] proposed self-adjusting spectral hybrid CG methods that ensure sufficient descent and demonstrate superior efficiency compared to traditional techniques. Similarly, hybrid spectral methods have been developed to solve nonlinear equations with convex constraints, with recent studies [8] and [**?**] proving their global convergence and computational efficiency.

Line search strategies, particularly Wolfe and strong Wolfe line search methods, play a critical role in optimizing step sizes, ensuring efficient descent while maintaining numerical stability. Refinements by [9] and [10] further enhance these techniques, improving robustness and convergence speed. The integration of spectral and hybrid approaches, along with advanced line search strategies, continues to drive the development of more efficient and effective optimization algorithms.

2 ALGORITHM FRAMEWORK AND CONVERGENCE PROPERTY

This part begins with an overview of the general search direction of the proposed algorithm, followed by an analysis of the sufficient descent property.

A recent method by [7] introduced the spectral hybrid Dai–Liao (DL) CG method by using specific CG parameters. Thus, this study proposed a single new variant of the method introduced by [7], namely the Spectral LS method, which is constructed by utilizing the Liu–Storey (LS) parameter. The details of this construction are presented in Table 1. Hence, the general search direction of the proposed variant is defined as follows:

$$d_k = \begin{cases} -g_k & if\ k = 1, \\ -\theta_k^{\text{SLS}} g_k + \omega_k \cdot \beta_k^{\text{SLS}} d_{k-1} + (1 - \omega_k) \cdot \beta_k^{1} d_{k-1} & if\ k \geq 2, \end{cases} \tag{2}$$

where

$$\theta_k^{\text{SLS}} = 1 + (\omega_k \cdot \beta_k^{\text{SLS}} + (1-\omega_k) \cdot \beta_k^2) \frac{g_k^{\text{T}} d_{k-1}}{||g_k||^2}, \tag{3}$$

$$\beta_k^{\text{SLS}} = \max\{0, \min\{\beta_k^{\text{SLS}^*}, \beta_k^3\}\}, \tag{4}$$

$$\beta_k^{\text{SLS}^*} = \frac{||g_k||^2 - \frac{||g_k||}{||g_{k-1}||}|g_k^{\text{T}} g_{k-1}|}{v_1 |g_k^{\text{T}} d_{k-1}| - d_{k-1}^{\text{T}} g_{k-1}} - t_2 \cdot \frac{\max\{0, g_k^T s_{k-1}\}}{d_{k-1} y_{k-1}}. \tag{5}$$

where SLS stands for Spectral LS, which refers to the spectral-type search direction strategy employed in this method. The term "Spectral LS" is used as a short name to denote the proposed approach.

In this paper, S-hHSDLLS refers to the Spectral hybrid Hestenes-Stiefel Dai–Liao with Line Search method, while S-hLSDL denotes the Spectral hybrid Liu–Storey Dai–Liao with Line Search method. Both methods are spectral hybrid conjugate gradient algorithms that combine Dai–Liao type search directions with spectral scaling strategies and are implemented with strong Wolfe line search conditions.

Table 1 : β_k of Spectral LS method

No	Function	$\boldsymbol{\beta_k^1}$	$\boldsymbol{\beta_k^2}$	$\boldsymbol{\beta_k^3}$
1	Spectral LS	LS	LS	LS
2	S-hHSDLLS	HS	HS	LS
3	S-hLSDL	CD	CD	LS

2.1 The Spectral Liu-Storey Conjugate Gradient Method framework

Here, we initially explain the Spectral LS Conjugate Gradient Method framework thoroughly.

Step 0. Given any initial point $x_0 \in \mathbb{R}^n$, constants $0 < \rho < \sigma < 1$, $\mu \geq 0$, and tolerance $\varepsilon > 0$. Let $k := 0$.

Step 1. Compute the gradient $g_k = \nabla f(x_k)$. If $\|g_k\| \leq \varepsilon$, terminate. Otherwise, go to Step 2.

Step 2. Determine θ_k and β_k using Eq. (3) - (5).

Step 3. Compute the search direction d_k according to Eq. (2).

Step 4. Determine a steplength $\alpha_k > 0$ by applying a line search technique to satisfy the Strong Wolfe conditions given

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

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

Step 5. Update $x_{k+1} = x_k + \alpha_k d_k$ and set $k = k + 1$. Return to Step 1.

Remark 1. For simplicity, we refer to these iterative methods as Spectral LS, determined by the approaches corresponding to equations (2)–(5) within the spectral hybrid DL Conjugate Gradient method framework

presented below. Moreover, to provide a complete algorithmic structure, the procedure for computing the step length α_k (**Step 4**) is illustrated in the algorithm above, ensuring compliance with the strong Wolfe line search conditions (6) and (7).

Here, the phrase self-contained algorithm framework means that all algorithmic components, including the direction update and steplength rule, are fully described within the method, ensuring it can be implemented independently. It is different from the self-adjusting property, which specifically refers to the adaptive spectral parameter adjustment in the search direction formula.

The arrows "$\leftarrow$" in the pseudocode signify assignment operations, indicating that the expression on the right-hand side is allocated to the variable on the left-hand side. The following pseudocode describes the strong Wolfe line search algorithm, originally introduced by [11], employed to determine the steplength α_k that fulfills the strong Wolfe requirements (6) and (7).

Algorithm 1 Strong Wolfe Line Search

Input: Initial step length $\alpha_0 > 0$, parameters δ and σ satisfy $0 < \delta < \sigma < 1$.
Initialize: $\alpha_{\text{low}} \leftarrow 0$, $\alpha \leftarrow \alpha_0$, $\alpha_{\text{up}} \leftarrow +\infty$
while Wolfe conditions are not satisfied **do**
  if $f(x_k + \alpha d_k) > f(x_k) + \delta\alpha g_k^T d_k$ or ($\alpha > \alpha_{\text{low}}$ and $g(x_k + \alpha d_k)^T d_k \geq 0$) **then**
    $\alpha_{\text{up}} \leftarrow \alpha$
  else if $|g(x_k + \alpha d_k)^T d_k| \leq -\sigma g_k^T d_k$ **then**
    return α
  else
    $\alpha_{\text{low}} \leftarrow \alpha$
  end if
  if $\alpha_{\text{up}} < +\infty$ **then**
    $\alpha \leftarrow \frac{1}{2}(\alpha_{\text{low}} + \alpha_{\text{up}})$
  else
    $\alpha \leftarrow 2\alpha$
  end if
end while

To establish their convergence properties, the following standard assumptions for equation (1) are required.

Assumption A.

(A1) For any $x_1 \in \mathbb{R}^n$, the objective function $f(x)$ is bounded from below on the level set $S(x_1) = x \in \mathbb{R}^n | f(x) \leq f(x_1)$;

(A2) $f(x)$ is differentiable and its gradient $g(x)$ is Lipschitz continuous in some neighborhood $\mathcal{N}$ of $S(x_1)$, i.e., there exists $L > 0$ such that $g(\cdot)$ satisfies,

$||g(x) - g(y)|| \leq L||x - y||, \forall x, y \in \mathcal{N}$.

This assumption (A2), similar to the well-established Zoutendijk condition by [12] for the classical CG method, is likewise employed in the convergence study of spectral conjugate gradient methods by [7].

The related properties of Spectral LS

In this section, we establish that the search directions of Spectral LS are sufficiently descending, and we prove the global convergence of Spectral LS under Assumption A.

Lemma 2.1 *For all $k \geq 1$, , the Spectral LS's search direction sequence $\{d_k\}$ satisfies $g_k^T d_k = -||g_k||^2$.* [7]

Proof. It follows from $d_1 = -g_1$ that $g_1^T d_1 = -||g_1||^2$. Next, we consider the case when $k \geq 2$. Based on relations (2) and (3), one has ,

$$\begin{aligned} g_k^T d_k &= -\theta_k^{\text{SLS}}||g_k||^2 + \omega_k \cdot \beta_k^{\text{SLS}} g_k^T d_{k-1} + (1-\omega_k) \cdot \beta_k^{\text{LS}} g_k^T d_{k-1} \\ &= -\left\{1 + (\omega_k \cdot \beta_k^{\text{SLS}} + (1-\omega_k) \cdot \beta_k^{\text{LS}}) \frac{g_k^T d_{k-1}}{||g_k||^2}\right\} ||g_k||^2 \\ &+ \omega_k \cdot \beta_k^{\text{SLS}} g_k^T d_{k-1} + (1-\omega_k) \cdot \beta_k^{\text{LS}} g_k^T d_{k-1} \\ &= -||g_k||^2. \end{aligned}$$

Therefore, the desired claim is proved. □

Theorem 2.2 *Assume that Assumption A holds. Let $\{x_k\}$ be generated by the Spectral LS. Then $lim_{k\to+\infty}||g_k|| = 0$.* [7].

Proof. Assume, for the sake of contradiction, that the conclusion fails to hold. Consequently, there exists a constant $\lambda > 0$ such that $||g_k|| \geq \lambda$ for any k. Let,

$$\beta_k^{\S} = \omega_k \cdot \beta^{\text{SLS}} + (1-\omega_k) \cdot \beta^{\text{LS}} = \omega_k \cdot \beta^{\text{SLS}} + (1-\omega_k) \cdot -\frac{g_k^T (g_k - g_{k-1})}{d_{k-1}^T g_{k-1}}. \tag{8}$$

To proceed, it follows from (2) that $d_k + \theta_k^{\text{SLS}} g_k = \beta_k^{\S} d_{k-1}$, which further shows

$$||d_k||^2 + 2\theta_k^{\text{SLS}} d_k^T g_k + (\theta_k^{\text{SLS}})^2 ||g_k||^2 = (\beta_k^{\S})^2 ||d_{k-1}||^2.$$

Thus,

$$\|d_k\|^2 = (\beta_k^s)^2 \|d_{k-1}\|^2 - 2\theta_k^{\text{SLS}} d_k^T g_k - (\theta_k^{\text{SLS}})^2 \|g_k\|^2. \tag{9}$$

Moreover, from the second parts of inequalities (6) and (7) and Lemma 2.1, we obtain,

$$d_{k-1}^T (g_k - g_{k-1}) \geq (\sigma - 1) g_{k-1}^T d_{k-1} = -(\sigma - 1)\|g_{k-1}\|^2 \geq 0,$$

which shows that $t_2 \cdot \frac{\max\{0, g_k^T s_{k-1}\}}{d_{k-1} y_{k-1}} \geq 0$. This, together with (5), it follows that,

$$\beta_k^{\text{SLS}^*} \leq \frac{||g_k||^2 - \frac{||g_k||}{||g_{k-1}||} |g_k^{\text{T}} g_{k-1}|}{v_1 |g_k^{\text{T}} d_{k-1}| - |d_{k-1}^{\text{T}} g_{k-1}|},$$

Additionally, utilizing the Cauchy–Schwarz inequality together with Lemma 2.1, it follows that

$$\frac{||g_k||^2 - \frac{||g_k||}{||g_{k-1}||}|g_k^{\mathrm{T}} g_{k-1}|}{v_1|g_k^{\mathrm{T}} d_{k-1}| - |d_{k-1}^{\mathrm{T}} g_{k-1}|} = \frac{||g_k||^2 - \frac{||g_k||}{||g_{k-1}||}|g_k^{\mathrm{T}} g_{k-1}|}{v_1|g_k^{\mathrm{T}} d_{k-1}| + ||g_{k-1}||^2} \geq 0.$$

Furthermore, from $v_1 \geq 0$ and (4), we have,

$$\begin{aligned} 0 \leq \beta_k^{\mathrm{SLS}} \leq \max\{0, \beta_k^{\mathrm{SLS}^*}\} &\leq \frac{||g_k||^2 - |g_k^{\mathrm{T}} g_{k-1}|}{v_1|g_k^{\mathrm{T}} d_{k-1}| - d_{k-1}^{\mathrm{T}} g_{k-1}} \\ &\leq \frac{g_k^T(g_k - g_{k-1})}{v_1|g_k^{\mathrm{T}} d_{k-1}| - d_{k-1}^{\mathrm{T}} g_{k-1}} \\ &\leq \frac{g_k^T(g_k - g_{k-1})}{-d_{k-1}^{\mathrm{T}} g_{k-1}} \\ &= \beta_k^{\mathrm{LS}}. \end{aligned}$$

Combining this with (8) and $\omega_k \in [0, 1]$, we obtain,

$$0 \leq \beta_k^{\S} \leq \omega_k \cdot \frac{||g_k||^2}{-d_{k-1}^{\mathrm{T}} g_{k-1}} + (1 - \omega_k) \cdot \frac{||g_k||^2}{-d_{k-1}^{\mathrm{T}} g_{k-1}} = \frac{||g_k||^2}{-d_{k-1}^{\mathrm{T}} g_{k-1}}. \tag{10}$$

Again, dividing both sides of (9) by $(g_k^T d_k)^2$, and from Lemma 2.1, (10), and $\|g_k\| \geq \lambda$, it holds that,

$$\begin{aligned} \frac{||d_k||^2}{||g_k||^4} = \frac{||d_k||^2}{(g_k^T d_k)^2} &= (\beta_k^{\S})^2 \frac{||d_{k-1}||^2}{(g_k^T d_k)^2} - \frac{2\theta_k^{\mathrm{SLS}} d_k^T g_k}{(g_k^T d_k)^2} - \frac{(\theta_k^{\mathrm{SLS}})^2 ||g_k||^2}{(g_k^T d_k)^2} \\ &= (\beta_k^{\S})^2 \frac{||d_{k-1}||^2}{(g_k^T d_k)^2} - \frac{2\theta_k^{\mathrm{SLS}}}{(g_k^T d_k)} - \frac{(\theta_k^{\mathrm{SLS}})^2 ||g_k||^2}{(g_k^T d_k)^2} \\ &\leq \Big(\frac{||g_k||^2}{-d_{k-1}^{\mathrm{T}} g_{k-1}}\Big)^2 \frac{||d_{k-1}||^2}{(g_k^T d_k)^2} - \frac{2\theta_k^{\mathrm{SLS}}}{(g_k^T d_k)} - \frac{(\theta_k^{\mathrm{SLS}})^2 ||g_k||^2}{(g_k^T d_k)^2} \\ &= \frac{||g_k||^4}{||g_{k-1}||^4} \cdot \frac{||d_{k-1}||^2}{||g_k||^4} + \frac{2\theta_k^{\mathrm{SLS}}}{||g_k||^2} - \frac{(\theta_k^{\mathrm{SLS}})^2 ||g_k||^2}{||g_k^T||^4} \\ &= \frac{||d_{k-1}||^2}{||g_{k-1}||^4} + \frac{2\theta_k^{\mathrm{SLS}}}{||g_k||^2} - \frac{(\theta_k^{\mathrm{SLS}})^2}{||g_k||^2} \\ &= \frac{||d_{k-1}||^2}{||g_{k-1}||^4} + \frac{(\theta_k^{\mathrm{SLS}} - 1)^2}{||g_k||^2} + \frac{1}{||g_k||^2} \\ &\leq \frac{||d_{k-1}||^2}{||g_{k-1}||^4} + \frac{1}{||g_k||^2} \\ &\leq \cdots \leq \sum_{i=1}^{k} \frac{1}{||g_i||^2} \leq \frac{k}{\lambda^2}. \end{aligned}$$

By summing the inequalities above, we obtain

$$\sum_{k=1}^{+\infty} \frac{||g_k||^4}{||d_k||^2} \geq \sum_{k=1}^{+\infty} \frac{\lambda^2}{k} = +\infty,$$

which contradicts to,

$$\sum_{k=1}^{+\infty} \frac{||g_k||^4}{||d_k||^2} <= +\infty,$$

Therefore, the expected result is established. □

3 RESULTS AND DISCUSSION

In this section, the computational efficiency of the proposed method Spectral Liu Storey is examined. A common approach for evaluating the efficiency of a method is to employ the test function. The test function plays a crucial role in validating and comparing newly developed optimization methods. The comparison is made between S-hHSDLLS and S-hLSDL method. The methods were coded in Matlab R2021a and run on a personal computer with AMD A9-9425 processor, Radeon R5 Graphics, 5 Compute Cores (2 CPU + 3 GPU), 3.10 GHz, and 64-bit Windows 10 operating system.

To evaluate the performance of the Spectral LS and S-hLSDLLS methods, 60 test functions with respect to the test are suggested from [13] and [14]. Table 3 shows the test functions, as well as their dimensions and initial points. The dimensions of the test problems vary from 2 up to 100,000. All methods are implemented with the strong Wolfe line search, where $\delta = 0.01$, $\sigma = 0.1$ and the termination criteria was set as $||g_k|| \leq 10^{-6}$. For Spectral LS, $\omega_k = |g_k^T d_{k-1}|/(||g_k||||d_{k-1}||)$, the parameter $v_1 = 2$ and $t_2 = \alpha_k$. The symbol "–" indicates that the iteration count exceeds 10,000 or the algorithm fails to reach the optimal solution. The numerical results are summarized in Table 2, where NOI denotes the number of iterations and CPU represents the computation time.

Table 2 : Numerical results for S-hHSDLLS with S-hLSDL and Spectral LS method

		S-hHSDLLS		S-hLSDL		Spectral LS	
No	**Functions**	NOI	CPU	NOI	CPU	NOI	CPU
1	Extended White & Holst	9	0.0283	16	0.7501	8	0.0241
2	Extended White & Holst	10	0.1070	19	0.2063	8	0.1061
3	Extended White & Holst	10	0.4774	19	0.8262	9	0.6138
4	Extended Rosenbrock	10	0.0339	13	0.1635	10	0.0978
5	Extended Rosenbrock	26	0.6115	328	8.741	36	1.0058
6	Extended Rosenbrock	27	0.6562	189	4.3629	23	0.6487
7	Extended Freudenstein & Roth	10	0.0383	24	0.1149	10	0.1150
8	Extended Freudenstein & Roth	11	0.0357	22	0.0457	9	0.0466
9	Extended Beale	10	0.0438	47	0.3589	12	0.1415
10	Extended Beale	11	0.5259	12	0.5233	12	0.5839
11	Extended Beale	10	0.3811	52	1.4966	12	0.4780
12	Raydan 1	9	0.0129	19	0.0808	17	0.1024
13	Raydan 1	16	0.0076	26	0.0133	19	0.0401
14	Raydan 1	69	0.0594	26	0.1273	69	0.0696
15	Extended Tridiagonal 1	14	0.0023	146	0.2919	12	0.0664
16	Extended Tridiagonal 1	14	0.0117	439	0.4126	14	0.0465
17	Extended Tridiagonal 1	14	0.0160	521	0.733	14	0.0518
18	Extended Himmelblau	8	0.0231	13	0.1154	8	0.0387
19	Extended Himmelblau	8	0.0288	11	0.0347	7	0.1855
20	Extended Himmelblau	8	0.2522	35	0.5622	8	0.3209
21	FLETCHCR	16	0.0102	18	0.033	16	0.0340
22	FLETCHCR	18	0.0157	18	0.0148	18	0.0285
23	Extended Powel	839	1.5143	2807	4.3879	130	0.2692
24	Extended Powel	144	1.0394	5123	31.7281	233	1.5318
25	NONSCOMP	11	0.0046	13	0.0331	10	0.0624

The performance profiles in Figures 1 and 2, based on the number of iterations (NOI) and CPU time, respectively, are generated from the results in Table 2. In the profiles, the value of each curve at $\tau = 1$ (the left side) represents the percentage of test problems where the corresponding method performed best. As τ increases, the curves indicate the cumulative percentage of problems each method solves within a factor τ of the best performance. Therefore, a higher curve at any τ value indicates better performance. Figures 1 and 2 show that S-hHSDLLS performs comparably to Spectral LS but requires more iterations, consistent with [7], where LS parameter implementation improved results. Table 2 shows that Spectral LS solves 98.3% (59/60) of problems, S-hHSDLLS solves 100% (60/60), and S-hLSDL solves 95% (57/60).

Figures 1 and 2 illustrate that the Spectral LS method exhibits rapid convergence, while the S-hHSDLLS and S-hLSDL methods approach a value of 1.0 more gradually. Although S-hHSDLLS solves 100% of the problems, Spectral LS remains superior due to its consistently lower iteration count. When evaluated using CPU time, Spectral LS outperforms both S-hHSDLLS and S-hLSDL in terms of success rate and robustness. As shown in Figure 2, it solves the largest proportion of problems with minimal CPU time, indicated by its steep initial rise and highest $\rho(\tau)$ value across the range of τ. This confirms that Spectral LS is the most efficient method for this problem set.

The S-hHSDLLS method is a competitive alternative to Spectral LS, closely following its performance while

Table 2 : *Table 2 (Continued)*

		S-hHSDLLS		S-hLSDL		Spectral LS	
No	**Functions**	NOI	CPU	NOI	CPU	NOI	CPU
26	NONSCOMP	13	0.0072	56	0.0227	11	0.0160
27	NONSCOMP	201	0.1086	186	0.1462	229	0.1576
28	Extended DENSCHNB	9	0.0121	10	0.0274	9	0.0532
29	Extended DENSCHNB	10	0.0149	31	0.0965	10	0.0177
30	Extended DENSCHNB	6	0.0218	7	0.0256	6	0.2603
31	Extended Quadratic Penalty QP1	6	0.0027	21	0.0506	6	0.0512
32	Extended Quadratic Penalty QP1	13	0.0056	46	0.0096	12	0.0231
33	Extended Penalty Function U52	6	0.0032	9	0.0289	6	0.0851
34	Extended Penalty Function U52	18	0.0166	217	0.3742	18	0.0457
35	Extended Penalty Function U52	10	0.0092	37	0.091	9	0.0390
36	Hager	12	0.0057	11	0.0231	12	0.0754
37	Hager	19	0.0249	20	0.0108	19	0.0437
38	Cube	19	0.0157	11	0.0164	13	0.0050
39	Cube	105	0.0776	35	0.0248	98	0.0738
40	Extended Maratos	12	0.006	23	0.0247	12	0.011
41	Extended Maratos	12	0.0171	23	0.0511	12	0.0599
42	Shallow	8	0.0284	19	0.1913	8	0.0393
43	Shallow	10	0.0440	53	0.3739	10	0.0812
44	Shallow	8	0.2955	15	0.324	8	0.1840
45	Generalized Quartic	6	0.1227	-	-	-	-
46	Generalized Quartic	16	0.0514	19	0.185	16	0.1003
47	Quadratic QF2	71	0.0714	102	0.1637	71	0.1148
48	Quadratic QF2	89	0.0319	181	0.1687	89	0.1015
49	Leon	16	0.0128	11	0.0115	13	0.0692
50	Leon	37	0.0211	455	0.2499	33	0.0494
51	Generalized Tridiagonal 1	22	0.0036	27	0.0408	7	0.0091
52	Generalized Tridiagonal 1	23	0.0287	26	0.0688	23	0.1578
53	Generalized Tridiagonal 1	25	0.1122	-	-	25	0.0885
54	Generalized Tridiagonal 2	16	0.0247	14	0.1034	17	0.0062
55	Generalized Tridiagonal 2	33	0.0441	32	0.0316	33	0.0584
56	Generalized Tridiagonal 2	33	0.1092	-	-	32	0.0972
57	POWER	10	0.0151	10	0.004	10	0.0424
58	POWER	141	0.1841	142	0.1295	142	0.1522
59	Dixon and Price	53	0.0243	351	0.2995	57	0.0984
60	Sum Squares	10	0.0041	10	0.004	10	0.0029

*Notes: NOI refers to the number of iterations, and CPU represents computation time in seconds.

demonstrating better efficiency than S-hLSDL, particularly in the middle range of τ values. This suggests that S-hHSDLLS is a robust option when Spectral LS is not preferable or feasible. In contrast, S-hLSDL is the least efficient among the three methods, as its slower rise in the performance profile indicates higher CPU time requirements. Nevertheless, it remains a viable, though less efficient, choice in scenarios where the other two methods are not applicable.

Overall, from Table 2 as well as Figure 1 and 2 clearly confirm that Spectral LS is the most efficient method in terms of generating the optimal search direction and achieving the most favourable steplength when

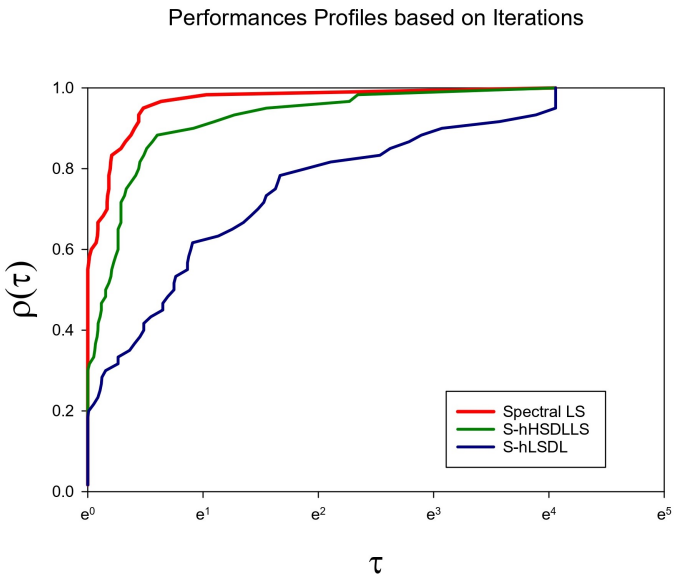

Figure 1 : Performance profiles with respect to τ based on the number of iterations.

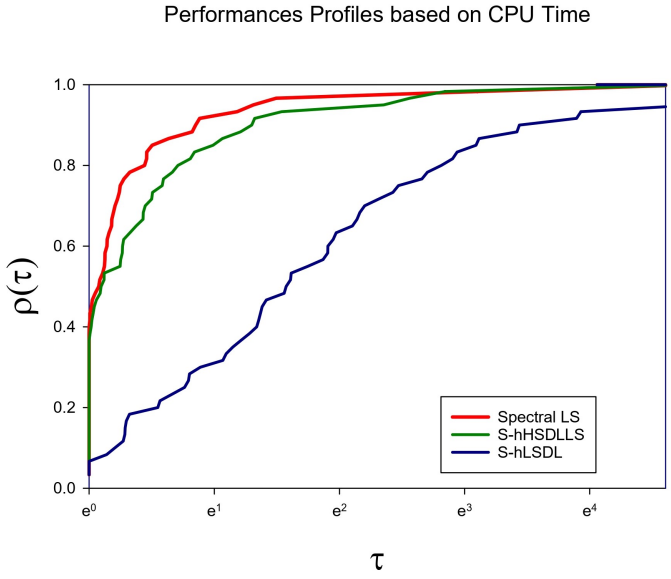

Figure 2 : Performance profiles with respect to τ based on the CPU time

considering the average outcomes, with s-hHSDLLS being a close second, while s-hLSDL demonstrates a significantly slower performance. This is because, the choosen parameter is crucial for the methods, eventhough HS can perform well, it is more sensitive to inaccuracies in gradient computations. Thus, LS is less sensitive and more stable. Other than that, LS is better due to its ability to balance robustness, and efficiency in guiding the optimization process.

4 CONCLUSION

In this study, the Spectral LS method, which is a variant of the self-adjusting spectral hybrid DL CG method, was introduced to address unconstrained optimization problems. The method successfully solved 98.3% of the test functions (59 out of 60), demonstrating its effectiveness in addressing optimization challenges. The research objectives were successfully achieved, confirming the reliability of the proposed approach.

The Spectral LS method was developed with self-adjusting spectral properties and hybrid strategies to enhance optimization performance. Numerical experiments using the strong Wolfe line search were conducted, focusing on performance metrics such as the number of iterations and CPU time.The findings indicated that the proposed method outperforms existing methods in terms of robustness and convergence speed.

Although the Spectral LS method has shown promising results, further improvements can be explored. Future studies could focus on applying this method to real-world optimization problems, such as robotic motion control, where precise and efficient search directions are essential for performance enhancement. Additionally, new variants of the Spectral LS method can be developed by integrating different combination parameters. For example, incorporating the RMIL parameter with the LS method may improve convergence properties and robustness, further expanding its application in numerical optimization. Furthermore, the findings of this study have been presented at the Mathematical Science and Statistic Seminar 2025 (MSSS2025), providing an opportunity for academic discussion and receiving constructive feedback from peers in the field.

ACKNOWLEDGEMENT

The author expresses sincere gratitude to Dr. Siti Mahani Marjugi for her invaluable guidance, advice, and constructive feedback throughout this work. Appreciation is also extended to Muhammad Aqiil Iqmal Ishak for his assistance and insightful discussions, as well as to colleagues and friends for their continuous support. The author also gratefully acknowledges the Department of Mathematics and Statistics for providing essential resources, and the anonymous reviewer for the constructive comments that greatly improved this work.

REFERENCES

[1] M. R. Hestenes, E. Stiefel *et al.*, "Methods of conjugate gradients for solving linear systems," *Journal of research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

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

[3] B. T. Polyak, "The conjugate gradient method in extremal problems," *USSR Computational Mathematics and Mathematical Physics*, vol. 9, no. 4, pp. 94–112, 1969.

[4] R. Fletcher, "Practical method of optimization (second ed.), unconstrained optimization," 1987.

[5] Y. Liu and C. Storey, "Efficient generalized conjugate gradient algorithms, part 1: theory," *Journal of Optimization Theory and Applications*, vol. 69, pp. 129–137, 1991.

[6] 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.

[7] X. W.-P. L. Hu Shao, Hang Guo, "Two families of self-adjusting spectral hybrid dl conjugate gradient methods and applications in image denoising," *Applied Mathematics Modelling*, vol. 84, no. 2, pp. 393–411, 2023.

[8] O. Mohammed, "Hybrid spectral algorithm under a convex constrained to solve nonlinear equations," *Journal of Interdisciplinary Mathematics*, vol. 41, p. 971, 2022.

[9] A. M. Awwal, L. Wang, P. Kumam, M. I. Sulaiman, S. Salisu, N. Salihu, and P. Yodjai, "Generalized rmil conjugate gradient method under the strong wolfe line search with application in image processing," *Mathematical Methods in the Applied Sciences*, vol. 46, pp. 17 544–17 556, 2023.

[10] O. Yousif, "The convergence properties of rmil+ conjugate gradient method under the strong wolfe line search," *Applied Mathematics and Computation*, vol. 367, p. 124777, 2020.

[11] P. Wolfe, "Convergence conditions for ascent methods. ii: Some corrections," *SIAM review*, vol. 13, no. 2, pp. 185–188, 1971.

[12] G. Zoutendijk, "Nonlinear rogramming computational methods," *Integer and Nonlinear Programming*, pp. 37–86, 1987.

[13] N. Andrei, *Nonlinear Conjugate Gradient Methods for Unconstrained Optimization*, 1st ed. Springer Cham, 2020.

[14] H.-J. Z. Momin Jamil, Xin-She Yang, "Test functions for global optimization: A comprehensive survey," *Elsevier*, pp. 193–222, 2013.

APPENDIX

Table 3 : Numerical results for S-hHSDLLS with S-hLSDL and Spectral LS method

No.	Function	Dimension	Initial Points
1	Extended White & Holst	50	$(1.2, \ldots, 1.2)$
2	Extended White & Holst	1,000	$(1.2, \ldots, 1.2)$
3	Extended White & Holst	10,000	$(1.2, \ldots, 1.2)$
4	Extended Rosenbrock	1,000	$(10, \ldots, 10)$
5	Extended Rosenbrock	10,000	$(10, \ldots, 10)$
6	Extended Rosenbrock	100,000	$(5, \ldots, 5)$
7	Extended Freudenstein & Roth	50	$(0.5, \ldots, 0.5)$
8	Extended Freudenstein & Roth	1,000	$(0.1, \ldots, 0.1)$
9	Extended Beale	1,000	$(-1, \ldots, -1)$
10	Extended Beale	10,000	$(-1, \ldots, -1)$
11	Extended Beale	50,000	$(0.5, \ldots, 0.5)$
12	Raydan 1	10	$(1, \ldots, 1)$
13	Raydan 1	100	$(-1, \ldots, -1)$
14	Raydan 1	1,000	$(-1, \ldots, -1)$
15	Extended Tridiagonal 1	10	$(2, \ldots, 2)$
16	Extended Tridiagonal 1	50	$(10, \ldots, 10)$
17	Extended Tridiagonal 1	100	$(10, \ldots, 10)$
18	Extended Himmelblau	1,000	$(1, \ldots, 1)$
19	Extended Himmelblau	1,000	$(20, \ldots, 20)$
20	Extended Himmelblau	10,000	$(-1, \ldots, -1)$
21	FLETCHCR	10	$(1.1, \ldots, 1.1)$
22	FLETCHCR	100	$(1.1, \ldots, 1.1)$
23	Extended Powel	100	$(1, \ldots, 1)$
24	Extended Powel	1,000	$(2, \ldots, 2)$
25	NONSCOMP	2	$(3, \ldots, 3)$
26	NONSCOMP	2	$(5, \ldots, 5)$
27	NONSCOMP	4	$(0.1, \ldots, 0.1)$
28	Extended DENSCHNB	10	$(10, \ldots, 10)$
29	Extended DENSCHNB	100	$(-50, \ldots, -50)$
30	Extended DENSCHNB	1,000	$(1, \ldots, 1)$
31	Extended Quadratic Penalty QP1	4	$(1, \ldots, 1)$
32	Extended Quadratic Penalty QP1	4	$(10, \ldots, 10)$
33	Extended Penalty Function U52	10	$(1, \ldots, 1)$
34	Extended Penalty Function U52	50	$(-2, \ldots, -2)$
35	Extended Penalty Function U52	100	$(-10, \ldots, -10)$
36	Hager	10	$(1, \ldots, 1)$
37	Hager	50	$(1.2, \ldots, 1.2)$
38	Cube	2	$(2, 2)$
39	Cube	3	$(1.1, \ldots, 1.1)$
40	Extended Maratos	10	$(1, \ldots, 1)$

Table 3 : *Table 3 (Continued)*

No.	Function	Dimension	Initial Points
41	Extended Maratos	100	$(1,\ldots,1)$
42	Shallow	1,000	$(2,\ldots,2)$
43	Shallow	1,000	$(10,10)$
44	Shallow	10,000	$(-1,\ldots,-1)$
45	Generalized Quartic	1,000	$(5,\ldots,5)$
46	Generalized Quartic	1,000	$(20,\ldots,20)$
47	Quadratic QF2	50	$(0.5,\ldots,0.5)$
48	Quadratic QF2	50	$(30,\ldots,30)$
49	Leon	2	$(2,2)$
50	Leon	2	$(8,8)$
51	Generlized Tridiagonal 1	10	$(2,\ldots,2)$
52	Generlized Tridiagonal 1	100	$(2,\ldots,2)$
53	Generlized Tridiagonal 1	500	$(1,\ldots,1)$
54	Generlized Tridiagonal 2	10	$(1,\ldots,1)$
55	Generlized Tridiagonal 2	100	$(1,\ldots,1)$
56	Generlized Tridiagonal 2	500	$(1,\ldots,1)$
57	POWER	10	$(1,\ldots,1)$
58	POWER	100	$(2,\ldots,2)$
59	Dixon and Price	10	$(2,\ldots,2)$
60	Sum Squares	10	$(1,\ldots,1)$