**EUROPEAN JOURNAL OF PURE AND APPLIED MATHEMATICS**
2025, Vol. 18, Issue 2, Article Number 5932
ISSN 1307-5543 – ejpam.com
Published by New York Business Global

# A Globally Convergent Conjugate Gradient Method Incorporating Perry's Parameter for Unconstrained Optimization

Hasan Hazim Jameel[1], Alaa Luqman Ibrahim[2,*], Neven E. Zaya[3]

[1] *Department of Mathematics, College of Basic Education, University of Duhok, Kurdistan Region-Iraq*
[2] *Department of Mathematics, College of Science, University of Zakho, Kurdistan Region-Iraq*
[3] *Management Information System, Administrative Technical Institute, Duhok Polytechnic University, Kurdistan Region-Iraq*

**Abstract.** To develop a conjugate gradient method that is both theoretically robust and practically effective for solving unconstrained optimization problems, this paper introduces a novel conjugate gradient method incorporating Perry's parameter along with the gradient-difference vector, as suggested by Powell, to enhance performance. The proposed method satisfies the descent condition, and its global convergence is established under standard assumptions. To assess its effectiveness, the method was tested on a diverse set of unconstrained optimization problems and compared against well-known conjugate gradient methods. Numerical experiments indicate that the proposed method outperforms classical approaches in terms of iteration count, function evaluations, and computational time. The results confirm the robustness and efficiency of the proposed method, making it a competitive choice for large-scale optimization problems.

**2020 Mathematics Subject Classifications**: 49-XX, 93-XX, 90Bxx, 90-XX.
**Key Words and Phrases**: Unconstrained Optimization, Conjugate Gradient Methods, Global Convergence, Numerical Performance.

## 1. Introduction

Optimization involves the minimization or maximization of an objective function. A key subset of optimization is unconstrained optimization, which focuses on minimizing a function of real variables without constraints. The general unconstrained optimization problem can be formulated as:

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

*Corresponding author.
DOI: https://doi.org/10.29020/nybg.ejpam.v18i2.5932

*Email addresses:* hasan.jameel@uod.ac (H. H. Jameel), alaa.ibrahim@uoz.edu.krd (A. L. Ibrahim), neven.zaya@dpu.edu.krd (N. E. Zaya)

where $f$ is a smooth function with an available gradient [1]. Several numerical methods have been developed to solve such problems, including the Steepest Descent (SD) method, Newton's method, Conjugate Gradient (CG) methods, and Quasi-Newton (QN) methods. This paper focuses on CG methods due to their efficiency and scalability, particularly for large-scale optimization problems. CG algorithms are widely applied in various fields. In engineering, they are used for solving problems in structural design, fluid dynamics, and control systems [2]. In data science and machine learning, CG methods play a crucial role in optimizing loss functions for regression, classification, and neural network training [3–8]. Additionally, in signal and image processing, CG methods are used for tasks such as image reconstruction, denoising, and signal recovery [9]. Scientific computing also heavily relies on CG algorithms, particularly for solving large sparse linear systems in finite element analysis and computational physics.

A key advantage of CG methods is their low memory requirement [10], which makes them suitable for high-dimensional problems. Furthermore, they exhibit rapid convergence for specific classes of functions. The foundation of CG methods was laid in 1952 by Hestenes and Stiefel [11], who introduced the CG method for unconstrained linear optimization. Later, in 1964, Fletcher and Reeves extended the CG method to solve nonlinear unconstrained minimization problems [12].

In this paper, we propose a new conjugate gradient method that enhances computational efficiency while ensuring the satisfaction of the descent condition, and convergence properties. Our approach introduces a novel CG parameter that optimizes search directions and improves convergence rates, making it more effective in solving large-scale optimization problems.

CG methods generate a sequence $\{x_k\}$ defined as:

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

where $x_0$ is a randomly chosen initial starting point. The step size $\alpha_k$ is determined using a one-dimensional search known as a line search [13]. Line searches can be exact or inexact, with the latter being preferred due to computational efficiency [14, 15]. In our research, we employ the strong Wolfe line search, which satisfies the following conditions [16, 17]:

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

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

where $0 < \delta < \mu < 1$. The strong Wolfe-Powell (SWP) condition further includes:

$$|g(x_k + \alpha_k d_k)^T d_k| \leq \mu |g_k^T d_k|. \tag{5}$$

The search direction $d_k$ is determined as:

$$d_{k+1} = \begin{cases} -g_1, & k = 0, \\ -g_{k+1} + \beta_k d_k, & k \geq 1. \end{cases} \tag{6}$$

where $g_k = \nabla f(x_k)$ and $\beta_k$ is a CG parameter that varies depending on the CG method used. The search direction must satisfy the descent condition:

$$g_{k+1}^T d_{k+1} \leq 0. \tag{7}$$

Various CG methods arise from different choices of $\beta_k$, including Hestenes-Stiefel (HS) [11], Fletcher-Reeves (FR) [12], Polak-Ribière-Polyak (PRP) [18, 19], Liu-Storey (LS) [20], Dai-Yuan (DY) [21], and Conjugate-Descent (CD) [22]:

$$\beta_k^{HS} = \frac{g_{k+1}^T y_k}{y_k^T d_k}, \tag{8}$$

$$\beta_k^{FR} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \tag{9}$$

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

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

$$\beta_k^{DY} = \frac{\|g_{k+1}\|^2}{y_k^T d_k}, \tag{12}$$

$$\beta_k^{CD} = \frac{\|g_{k+1}\|^2}{-g_k^T d_k}, \tag{13}$$

where $y_k = g_{k+1} - g_k$ and $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^n$. In general, the DY and FR methods with inexact line searches demonstrate good convergence performance, though their computational efficiency is inferior to the PRP method. In recent years, many scholars have proposed improved CG methods to balance numerical performance and convergence properties [23].For more details on the various developments in CG methods, see [24–27].

This paper is structured as follows: Section 2 presents the new conjugate gradient algorithm along with its details. Section 3 establishes the descent condition. Section 4 analyzes the convergence properties of the proposed method. Section 5 demonstrates its efficiency through numerical comparisons with standard methods using test problems from the CUTE collection and unconstrained optimization problems. Finally, Section 6 concludes the paper.

## 2. Proposed Method and Algorithm

In this section, we introduce a newly proposed CG method for solving nonlinear optimization problems. This method is based on the Perry parameter [28]:

$$\beta_k^{Perry} = \frac{g_{k+1}^T (y_k - v_k)}{d_k^T y_k}, \tag{14}$$

and employs a gradient-difference vector as suggested by Powell [29]:

$$\bar{y}_k = y_k + (1-\theta)(G_k s_k - y_k), \tag{15}$$

where $0 < \theta < 1$.

Additionally, to incorporate second-order information without computing the Hessian explicitly, Andrei [30] proposed a nonlinear CG algorithm where the Hessian-vector product is approximated using finite differences:

$$y_k^* = y_k + (1-\theta)\left(\frac{y_k}{\sigma} - y_k\right), \tag{16}$$

where $\sigma = \frac{2\sqrt{\epsilon_m}(1+\|x_{k+1}\|)}{\|v_k\|}$, and $\epsilon_m$ is the machine precision. Now, let us define the updated vectors as follows:

$$y_k^* = \frac{y_k}{\sigma} - \frac{\theta y_k}{\sigma} + \theta y_k, \tag{17}$$

and for the $y_k$ vector:

$$y_k^* = \left(\frac{1-\theta}{\sigma} + \theta\right) y_k. \tag{18}$$

By substituting $y_k^*$ into the Perry formula, we obtain the new expression for the conjugate gradient parameter:

$$\beta_k^{new} = \frac{g_{k+1}^T(y_k^* - v_k)}{d_k^T y_k^*} = \frac{g_{k+1}^T y_k^* - g_{k+1}^T v_k}{d_k^T y_k^*}. \tag{19}$$

After performing some algebraic manipulations, we derive the final form of $\beta_k^{new}$:

$$\beta_k^{new} = \frac{g_{k+1}^T y_k}{d_k^T y_k} - \frac{\sigma}{(1-\theta+\sigma\theta)} \cdot \frac{g_{k+1}^T v_k}{d_k^T y_k}, \tag{20}$$

where $\sigma = \frac{2\sqrt{\epsilon_m}(1+\|x_{k+1}\|)}{\|v_k\|}$, and $0 < \theta < 1$.

The newly developed conjugate gradient methods aim to improve the efficiency and convergence characteristics of traditional CG approaches, particularly for high-dimensional optimization problems.

**Note:** When $\theta = 1$, the expression in equation (20) reduces to the traditional Perry formula, meaning we revert to the original conjugate gradient method that does not use finite differences or second-order approximations.

**Algorithm 1** New Conjugate Gradient Method for Unconstrained Optimization

1: **Initialization** Choose an initial point $x_0 \in \mathbb{R}^n$, parameters $0 < \delta < \mu < 1$, and tolerance $\epsilon > 0$. Set $k = 0$ and $d_1 = -g_1$.
2: **Stopping Condition** If $\|g_k\| \leq \epsilon$, stop.
3: **Compute Step Size** Compute $\alpha_k$ using the weak Wolfe conditions (Equations (3) and (4)).
4: **Update** Update the point:

$$x_{k+1} = x_k + \alpha_k d_k. \tag{21}$$

5: **Compute Direction** Compute:

$$d_{k+1} = -g_{k+1} + \beta_k d_k. \tag{22}$$

Choose $\beta_k$ using Equation (20).
6: **Repeat** Set $k = k + 1$ and go to Step 2.

## 3. Descent Property and Sufficient Descent Condition

In this section, we establish the descent condition for the proposed algorithm.

**Theorem 1.** *If the search direction $d_{k+1}$ is generated by equation* (6) *with $\beta_k^{New}$ from equation* (20)*, then the descent condition holds:*

$$g_{k+1}^T d_{k+1} \leq 0. \tag{23}$$

*Proof.* Multiplying both sides of equation (6) by $g_{k+1}^T$ and substituting $\beta_k^{\text{New}}$ from equation (20), we obtain:

$$g_{k+1}^T d_{k+1} = -g_{k+1}^T g_{k+1} + \frac{g_{k+1}^T y_k}{d_k^T y_k} g_{k+1}^T d_k - \left(\frac{\sigma}{(1-\theta)+\sigma\theta}\right) \frac{g_{k+1}^T v_k}{d_k^T y_k} g_{k+1}^T d_k. \tag{24}$$

If the search direction is exact, then it satisfies the descent condition:

$$g_{k+1}^T d_{k+1} = -\|g_{k+1}\|^2 \leq 0. \tag{25}$$

For an inexact search direction where $g_{k+1}^T d_k \neq 0$, we note that the first two terms of equation (24) satisfy:

$$-g_{k+1}^T g_{k+1} + \frac{g_{k+1}^T y_k}{d_k^T y_k} g_{k+1}^T d_k \leq 0, \tag{26}$$

which follows from the HS method's descent property from the inequality:

$$g_{k+1}^T y_k \leq \|g_{k+1}\| \|y_k\|.$$

$$-\|g_{k+1}\|^2 + \frac{g_{k+1}^T y_k}{d_k^T y_k} d_k^T y_k \leq -\|g_{k+1}\|^2 + \|g_{k+1}\| \|y_k\|.$$

$$g_{k+1}^T d_{k+1} \leq -\left(1 - \frac{\|y_k\|}{\|g_{k+1}\|}\right) \|g_{k+1}\|.$$

Since $0 \leq \frac{\|y_k\|}{\|g_{k+1}\|} \leq 1$, we have:

$$-\|g_{k+1}\|^2 + \frac{g_{k+1}^T y_k}{d_k^T y_k} d_k^T y_k \leq 0.$$

It remains to prove that the third term of (24) is less than or equal to zero.

Since $\sigma = \frac{2\sqrt{\epsilon_m}(1+\|x_{k+1}\|)}{\|v_k\|} \geq 0 \quad \text{and} \quad 0 < \theta < 1$. Thus, we conclude that:

$$\frac{\sigma}{(1-\theta) + \sigma\theta} \geq 0. \tag{27}$$

Therefore, equation (24) can be rewritten as:

$$g_{k+1}^T d_{k+1} \leq -(\frac{\sigma\alpha_k}{(1-\theta) + \sigma\theta}) \frac{(g_{k+1}^T d_k)^2}{d_k^T y_k}. \tag{28}$$

Since $\alpha_k$, $d_k^T y_k$, and $(g_{k+1}^T d_k)^2$ are non-negative, we obtain:

$$g_{k+1}^T d_{k+1} \leq 0. \tag{29}$$

This completes the proof.

## 4. Convergence Condition for the New Method

This section discusses the convergence properties of the proposed algorithm. We introduce the following mild assumptions in order to achieve global convergence.

**Assumption 1:**

- The level set $S = \{x \mid f(x) \leq f(x_0)\}$ is bounded.
- In some neighborhood $N$ of $\delta$, $f$ is continuously differentiable, and its gradient is Lipschitz continuous with a Lipschitz constant $L > 0$. That is,

$$\|g(x) - g(y)\| \leq L\|x - y\| \quad \forall x, y \in S \tag{30}$$

From these assumptions, it follows that there exists a positive constant $b$ such that

$$\|g(x)\| \leq b \quad \forall x \in S. \tag{31}$$

Based on these assumptions, the global convergence of the new algorithm can be established as follows:

**Lemma 1.** *[31] Let assumptions 1 hold. Consider methods* (1) *and* (2)*, where $d_{k+1}$ is a descent direction, and $\alpha_k$ satisfies the standard Wolfe line search. If*

$$\sum_{k\geq 1} \frac{1}{\|d_{k+1}\|^2} = \infty, \tag{32}$$

*then*

$$\lim_{k\to\infty} \inf \|g_{k+1}\| = 0. \tag{33}$$

Based on the previous discussion, we now establish the global convergence of the new algorithm:

**Theorem 2.** *If assumptions 1 is satisfied, and the sequences $\{x_k\}, \{d_k\}, \{g_k\}, \{\alpha_k\}$ are generated by Algorithm 1, then it holds that*

$$\lim_{k\to\infty} \inf \|g_{k+1}\| = 0. \tag{34}$$

*Proof.* From equations (2) and (20), we have

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \left| \frac{g_{k+1}^T y_k}{d_k^T y_k} - \frac{\sigma}{(1-\theta)+\sigma\theta} \frac{g_{k+1}^T v_k}{d_k^T y_k} \right| \|d_k\|. \tag{35}$$

Simplifying this further, we obtain

$$\|d_{k+1}\| \leq \|g_{k+1}\| + \left( \left| \frac{g_{k+1}^T y_k}{d_k^T y_k} \right| + \frac{\sigma}{(1-\theta)+\sigma\theta} \left| \frac{g_{k+1}^T v_k}{d_k^T y_k} \right| \right) \|d_k\|. \tag{36}$$

Since $g_{k+1}^T v_k \leq \alpha_k d_k^T y_k$, and using equation (28) along with the Lipschitz condition $\|y_k\| \leq L\|v_k\|$, we also have $g_{k+1}^T y_k \leq L g_{k+1}^T d_k$, where $L > 0$. Therefore,

$$\|d_{k+1}\| \leq M + \left( L + \frac{\sigma\alpha_k}{(1-\theta)+\sigma\theta} \right) \|d_k\|. \tag{37}$$

Since $\|v_k\| = \|x - x_k\|$ and $D = \max\{\|x - x_k\| \mid \forall x, x_k \in \mathbb{R}\}$, equation (37) becomes

$$\|d_{k+1}\| \leq M + \left( L + \frac{\sigma\alpha_k}{(1-\theta)+\sigma\theta} \right) D = \phi. \tag{38}$$

Thus, we have

$$\sum_{k\geq 1} \frac{1}{\|d_{k+1}\|^2} \geq \sum_{k\geq 1} \frac{1}{\phi^2} = \infty, \tag{39}$$

which implies

$$\sum_{k\geq 1} \frac{1}{\|d_{k+1}\|^2} = \infty. \tag{40}$$

By Lemma 1, we conclude that

$$\lim_{k\to\infty} \inf \|g_k\| = 0, \tag{41}$$

which completes the proof.

## 5. Numerical Experiments

In this section, we present the computational performance of a Matlab implementation of the CG algorithm on a set of unconstrained optimization test problems from the CUTE library [32, 33] and other sources of unconstrained problem sets [34]. These problems represent a wide range of problem dimensions, from small-scale cases to large-scale cases.

This comparison is based on several performance metrics: the number of iterations (NI), the total frequency of objective function evaluations (NF), and the CPU time (CPUT) in seconds. The computational performance of the proposed methods is compared with the HS and Perry methods. Each algorithm applied the strong Wolfe line search conditions, with parameters set to $\delta = 0.01$ and $\mu = 0.3$. Iterations were terminated if any of the following conditions were satisfied:

- $\|g_k\| < 10^{-6}$, where $g_k$ denotes the Euclidean norm.
- The number of iterations exceeded 2000.
- The computation time surpassed 500 seconds.

However, it is important to note that in some cases, the method failed to converge to a solution. In such instances, the results were marked as NaN to indicate that the algorithm could not produce a valid solution within the given constraints. All codes are written in Matlab, and the computational experiments were performed on a personal laptop running Windows 10, with an Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz (2.81 GHz) and 8.00 GB of RAM.

Table 1 presents the results in terms of the NI, number of NF, and CPUT. Table 2 provides a detailed comparison of the proposed method with the classical HS and Perry algorithms, highlighting the percentage improvements achieved by the new method.

A comprehensive analysis of these results was conducted using the performance profile tool introduced by Dolan and Moré [35]. This tool provides a cumulative distribution function $p_s(\tau)$, enabling a unified and fair comparison of the algorithms across all metrics. The performance profiles for the NI, NF, and CPUT are displayed in Figures 1a, 1b, and 1c, respectively.

Table 1: Numerical results of the proposed method compared with HS and Perry methods

| **Test Function** | **N** | **HS** | | | **Perry** | | | **New** | | | **New** ($\epsilon_m = 1$) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **NI** | **NF** | **CPUT** | **NI** | **NF** | **CPUT** | **NI** | **NF** | **CPUT** | **NI** | **NF** | **CPUT** |
| 'cosine' | 500 | 55 | 133 | 0.033755 | 31 | 85 | 0.009681 | 37 | 93 | 0.012033 | 31 | 83 | 0.016694 |
| 'cosine' | 1000 | 51 | 117 | 0.023636 | 26 | 79 | 0.01372 | 27 | 78 | 0.014678 | 26 | 79 | 0.014862 |
| 'cosine' | 5000 | 34 | 86 | 0.075612 | 31 | 97 | 0.08309 | 29 | 85 | 0.075538 | 31 | 98 | 0.080135 |
| 'cosine' | 10000 | 193 | 248 | 0.401999 | 20 | 71 | 0.130764 | 20 | 70 | 0.119581 | 20 | 71 | 0.13456 |
| 'dixmaana' | 1500 | 20 | 66 | 0.092721 | 14 | 58 | 0.084754 | 15 | 56 | 0.07047 | 14 | 58 | 0.078474 |
| 'dixmaana' | 3000 | 24 | 73 | 0.22983 | 18 | 66 | 0.21372 | 14 | 61 | 0.17963 | 18 | 66 | 0.218552 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 'dixmaana' | 15000 | 21 | 73 | 0.90865 | 19 | 71 | 0.827259 | 15 | 69 | 0.909347 | 19 | 71 | 0.938866 |
| 'dixmaana' | 30000 | 16 | 70 | 1.910883 | 16 | 69 | 1.713247 | 18 | 72 | 2.13092 | 16 | 69 | 1.629957 |
| 'dixmaanb' | 1500 | 10 | 53 | 0.090052 | 15 | 63 | 0.101415 | 16 | 64 | 0.07883 | 15 | 63 | 0.072512 |
| 'dixmaanb' | 3000 | 13 | 58 | 0.17841 | 19 | 66 | 0.192647 | 20 | 68 | 0.216327 | 19 | 66 | 0.187179 |
| 'dixmaanb' | 15000 | 14 | 65 | 0.844221 | 18 | 69 | 0.937515 | 19 | 69 | 0.903137 | 18 | 69 | 0.841963 |
| 'dixmaanb' | 30000 | 21 | 74 | 1.728917 | 18 | 70 | 1.732772 | 19 | 73 | 1.791905 | 18 | 70 | 1.777622 |
| 'dixmaanc' | 3000 | 23 | 72 | 0.215691 | 19 | 68 | 0.200968 | 13 | 61 | 0.183541 | 19 | 68 | 0.214485 |
| 'dixmaanc' | 15000 | 19 | 72 | 0.901367 | 19 | 72 | 0.887783 | 18 | 69 | 0.859604 | 19 | 72 | 0.918407 |
| 'dixmaane' | 1500 | 373 | 422 | 0.526614 | 186 | 298 | 0.327686 | 154 | 250 | 0.341556 | 158 | 266 | 0.314185 |
| 'dixmaane' | 3000 | 456 | 506 | 1.58176 | 208 | 361 | 1.002094 | 209 | 360 | 1.099917 | 231 | 378 | 1.140235 |
| 'dixmaane' | 15000 | 744 | 797 | 10.1206 | 343 | 554 | 6.977925 | 315 | 531 | 6.760724 | 430 | 682 | 8.448322 |
| 'dixmaane' | 24000 | 999 | 1054 | 20.09337 | 394 | 629 | 12.0246 | 445 | 760 | 14.85598 | 451 | 706 | 13.26736 |
| 'dixmaanf' | 15000 | 494 | 547 | 7.602266 | 277 | 469 | 5.851253 | 250 | 442 | 5.774908 | 428 | 568 | 7.05392 |
| 'dixmaanf' | 24000 | 1356 | 1418 | 28.5487 | 316 | 491 | 9.888144 | 254 | 429 | 8.428413 | 310 | 496 | 9.647825 |
| 'dixmaang' | 1500 | NaN | NaN | NaN | 167 | 271 | 0.382597 | 152 | 268 | 0.315178 | 153 | 245 | 0.301801 |
| 'dixmaang' | 3000 | NaN | NaN | NaN | 150 | 257 | 0.756952 | 203 | 327 | 0.945721 | 156 | 253 | 0.73545 |
| 'dixmaang' | 15000 | 373 | 427 | 5.115567 | 223 | 358 | 4.527224 | 265 | 424 | 5.191343 | 246 | 406 | 4.989654 |
| 'dixmaang' | 30000 | 1415 | 2595 | 62.24052 | 122 | 212 | 4.965631 | 168 | 296 | 7.095215 | 139 | 248 | 5.888025 |
| 'dixmaanh' | 1500 | NaN | NaN | NaN | 114 | 194 | 0.229408 | 133 | 244 | 0.36558 | 114 | 214 | 0.287352 |
| 'dixmaanh' | 3000 | NaN | NaN | NaN | 188 | 288 | 0.834066 | 159 | 279 | 0.814102 | 171 | 293 | 0.87006 |
| 'dixmaani' | 300 | 935 | 980 | 0.385063 | 407 | 638 | 0.207102 | 352 | 602 | 0.219785 | 402 | 642 | 0.211295 |
| 'dixmaani' | 3000 | NaN | NaN | NaN | 627 | 986 | 2.721861 | 620 | 1032 | 2.924245 | 637 | 1029 | 2.836656 |
| 'dixmaanj' | 300 | 1984 | 2054 | 0.65829 | 114 | 206 | 0.066094 | 109 | 211 | 0.065167 | 101 | 190 | 0.059669 |
| 'dixmaanj' | 9000 | 433 | 486 | 3.704114 | 203 | 342 | 2.639958 | 198 | 332 | 2.611461 | 198 | 337 | 2.594001 |
| 'dixmaanj' | 15000 | 500 | 553 | 6.82994 | 202 | 359 | 4.599998 | 183 | 336 | 4.205175 | 192 | 317 | 4.068118 |
| 'dixmaank' | 30000 | 640 | 703 | 17.35754 | 202 | 351 | 8.516839 | 189 | 321 | 7.615254 | 192 | 351 | 8.315288 |
| 'dixmaank' | 3000 | NaN | NaN | NaN | 135 | 231 | 0.633973 | 119 | 223 | 0.6079 | 166 | 275 | 0.731622 |
| 'dixmaank' | 30000 | 640 | 703 | 16.85895 | 202 | 351 | 8.482023 | 189 | 321 | 7.709402 | 192 | 351 | 9.147786 |
| 'dixon3dq' | 10 | 77 | 114 | 0.003808 | 60 | 117 | 0.002931 | 57 | 109 | 0.004382 | 52 | 106 | 0.003532 |
| 'dixon3dq' | 50 | 386 | 423 | 0.015569 | 458 | 733 | 0.025348 | 279 | 468 | 0.017681 | 272 | 445 | 0.018375 |
| 'eg2' | 4 | 32 | 67 | 0.007854 | 22 | 60 | 0.001858 | 26 | 67 | 0.002469 | 22 | 60 | 0.002194 |
| 'eg2' | 10 | 66 | 141 | 0.004638 | 58 | 118 | 0.003596 | 35 | 85 | 0.002841 | 58 | 118 | 0.005193 |
| 'eg2' | 50 | 71 | 206 | 0.007189 | 165 | 428 | 0.014942 | 120 | 319 | 0.01263 | 101 | 242 | 0.008975 |
| 'freuroth' | 4 | 218 | 343 | 0.023291 | 80 | 224 | 0.006497 | 51 | 159 | 0.005727 | 68 | 185 | 0.008482 |
| 'freuroth' | 10 | 301 | 395 | 0.019306 | 52 | 169 | 0.005128 | 71 | 201 | 0.009838 | 52 | 169 | 0.006593 |
| 'freuroth' | 50 | 290 | 390 | 0.027218 | 68 | 194 | 0.007977 | 64 | 188 | 0.010353 | 68 | 189 | 0.010171 |
| 'freuroth' | 100 | 275 | 351 | 0.026272 | 65 | 238 | 0.01309 | 63 | 181 | 0.010686 | 60 | 169 | 0.009422 |
| 'himmelbg' | 500 | 2 | 9 | 0.006417 | 2 | 9 | 0.000912 | 2 | 9 | 0.000876 | 2 | 9 | 0.000863 |
| 'himmelbg' | 1000 | 2 | 9 | 0.00218 | 2 | 9 | 0.001992 | 2 | 9 | 0.001696 | 2 | 9 | 0.001523 |
| 'himmelbg' | 5000 | 3 | 21 | 0.013903 | 3 | 21 | 0.012772 | 3 | 21 | 0.01348 | 3 | 21 | 0.012958 |
| 'himmelbg' | 10000 | 2 | 11 | 0.018961 | 2 | 11 | 0.017717 | 2 | 11 | 0.018844 | 2 | 11 | 0.017778 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 'liarwhd' | 500 | 88 | 227 | 0.025006 | 70 | 232 | 0.020955 | 58 | 226 | 0.021396 | 73 | 226 | 0.023267 |
| 'liarwhd' | 1000 | 164 | 389 | 0.044784 | 95 | 318 | 0.034766 | 71 | 254 | 0.027863 | 68 | 254 | 0.029333 |
| 'liarwhd' | 5000 | 144 | 371 | 0.184553 | 89 | 342 | 0.158965 | 90 | 334 | 0.15707 | 74 | 271 | 0.12338 |
| 'liarwhd' | 10000 | 145 | 472 | 0.409646 | 94 | 347 | 0.310964 | 67 | 257 | 0.224857 | 113 | 404 | 0.328639 |
| 'penalty1' | 500 | 22 | 119 | 0.178701 | 19 | 107 | 0.139866 | 20 | 113 | 0.14773 | 25 | 120 | 0.157519 |
| 'penalty1' | 1000 | 21 | 110 | 0.404508 | 21 | 111 | 0.400485 | 27 | 117 | 0.426632 | 22 | 106 | 0.385329 |
| 'penalty1' | 4000 | 23 | 138 | 6.079985 | 26 | 140 | 6.132693 | 23 | 137 | 6.028452 | 25 | 137 | 6.049148 |
| 'penalty1' | 10000 | 18 | 79 | 18.10737 | 18 | 79 | 18.12312 | 17 | 77 | 17.72842 | 17 | 79 | 18.25694 |
| 'tridia' | 500 | 1434 | 1497 | 0.116049 | 723 | 1159 | 0.080065 | 630 | 1018 | 0.07713 | 612 | 980 | 0.081127 |
| 'tridia' | 1000 | NaN | NaN | NaN | 968 | 1531 | 0.162034 | 995 | 1671 | 0.202518 | 923 | 1434 | 0.174033 |
| 'tridia' | 4000 | NaN | NaN | NaN | NaN | NaN | NaN | 1549 | 2545 | 1.03679 | 1992 | 3174 | 1.34513 |
| 'woods' | 500 | NaN | NaN | NaN | 164 | 335 | 0.023281 | 165 | 389 | 0.028255 | 169 | 374 | 0.027434 |
| 'woods' | 1000 | NaN | NaN | NaN | 207 | 433 | 0.046658 | 131 | 285 | 0.031851 | 170 | 352 | 0.039766 |
| 'woods' | 5000 | NaN | NaN | NaN | 237 | 506 | 0.243439 | 124 | 286 | 0.151039 | 245 | 546 | 0.283006 |
| 'woods' | 10000 | 515 | 653 | 0.626043 | 176 | 383 | 0.350396 | 139 | 345 | 0.317029 | 125 | 287 | 0.276205 |
| 'bdexp' | 500 | 2 | 7 | 0.011732 | 2 | 7 | 0.002199 | 2 | 7 | 0.001729 | 2 | 7 | 0.001717 |
| 'bdexp' | 1000 | 2 | 7 | 0.003878 | 2 | 7 | 0.004103 | 2 | 7 | 0.004569 | 2 | 7 | 0.003686 |
| 'bdexp' | 5000 | 2 | 8 | 0.01844 | 3 | 19 | 0.032291 | 3 | 19 | 0.029448 | 3 | 19 | 0.028148 |
| 'bdexp' | 10000 | 2 | 9 | 0.042049 | 2 | 9 | 0.042081 | 2 | 9 | 0.042312 | 2 | 9 | 0.044224 |
| 'exdenschnf' | 500 | 46 | 103 | 0.019623 | 23 | 75 | 0.007377 | 22 | 78 | 0.011364 | 23 | 75 | 0.009551 |
| 'exdenschnf' | 1000 | 56 | 113 | 0.020576 | 25 | 80 | 0.015406 | 29 | 84 | 0.016264 | 28 | 85 | 0.020477 |
| 'exdenschnf' | 5000 | 43 | 102 | 0.088943 | 23 | 83 | 0.066448 | 29 | 90 | 0.078213 | 24 | 84 | 0.068349 |
| 'exdenschnf' | 10000 | 35 | 98 | 0.153769 | 27 | 90 | 0.141202 | 23 | 96 | 0.15179 | 27 | 90 | 0.139626 |
| 'biggsb1' | 4 | 17 | 47 | 0.006482 | 26 | 56 | 0.001628 | 17 | 49 | 0.001473 | 26 | 56 | 0.001858 |
| 'biggsb1' | 10 | 36 | 73 | 0.003433 | 35 | 77 | 0.003496 | 46 | 93 | 0.005471 | 35 | 77 | 0.003913 |
| 'biggsb1' | 100 | 364 | 396 | 0.017962 | 337 | 521 | 0.026132 | 231 | 369 | 0.016808 | 376 | 630 | 0.028446 |
| 'biggsb1' | 500 | 1692 | 1726 | 0.107611 | 1495 | 2408 | 0.148393 | 1128 | 1800 | 0.117736 | 1437 | 2255 | 0.144348 |
| 'power1' | 4 | 36 | 73 | 0.005881 | 33 | 80 | 0.002146 | 28 | 65 | 0.002168 | 33 | 80 | 0.002568 |
| 'power1' | 10 | 95 | 141 | 0.005946 | 76 | 128 | 0.003989 | 81 | 155 | 0.006034 | 74 | 125 | 0.005361 |
| 'power1' | 50 | 398 | 458 | 0.019117 | 473 | 778 | 0.028975 | 440 | 708 | 0.02847 | 460 | 698 | 0.02841 |
| 'power1' | 100 | 933 | 1001 | 0.040213 | 999 | 1637 | 0.058882 | 817 | 1326 | 0.057257 | 1030 | 1649 | 0.072432 |
| 'diagonal1' | 4 | 19 | 51 | 0.0055 | 16 | 46 | 0.001338 | 15 | 47 | 0.001364 | 16 | 46 | 0.001381 |
| 'diagonal1' | 10 | 31 | 69 | 0.003252 | 22 | 62 | 0.003238 | 28 | 65 | 0.002222 | 22 | 62 | 0.002042 |
| 'diagonal1' | 50 | 223 | 276 | 0.014341 | 50 | 105 | 0.005477 | 48 | 100 | 0.006636 | 49 | 104 | 0.004142 |
| 'diagonal1' | 100 | 204 | 258 | 0.017461 | 70 | 140 | 0.008903 | 73 | 139 | 0.007528 | 68 | 137 | 0.006523 |
| 'singx' | 100 | 170 | 366 | 0.029554 | 185 | 440 | 0.03583 | 150 | 380 | 0.037192 | 124 | 293 | 0.021211 |
| 'singx' | 500 | 238 | 502 | 0.798809 | 205 | 474 | 0.753916 | 152 | 365 | 0.605223 | 94 | 240 | 0.386519 |
| 'lin' | 10 | 8 | 40 | 0.150211 | 8 | 40 | 0.149828 | 8 | 40 | 0.165787 | 8 | 40 | 0.149329 |
| 'lin' | 100 | 10 | 51 | 0.362836 | 10 | 51 | 0.348565 | 10 | 51 | 0.352339 | 10 | 51 | 0.356541 |
| 'lin' | 500 | 14 | 55 | 0.604231 | 14 | 55 | 0.596905 | 14 | 55 | 0.609829 | 14 | 55 | 0.620677 |
| 'lin' | 1000 | 12 | 61 | 37.77361 | 12 | 61 | 37.48226 | 12 | 61 | 37.69565 | 12 | 61 | 37.33486 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 'osb2' | 11 | NaN | NaN | NaN | 465 | 797 | 0.072565 | 362 | 636 | 0.060535 | 460 | 772 | 0.076487 |
| 'rosex' | 50 | 127 | 275 | 0.017377 | 64 | 213 | 0.011807 | 41 | 138 | 0.009282 | 54 | 179 | 0.011496 |
| 'rosex' | 100 | 114 | 269 | 0.021787 | 58 | 166 | 0.010547 | 48 | 165 | 0.010602 | 56 | 151 | 0.0093 |
| 'trid' | 500 | 154 | 201 | 0.271871 | 30 | 76 | 0.094568 | 31 | 76 | 0.097215 | 30 | 76 | 0.107171 |
| 'trid' | 1000 | 206 | 256 | 0.964584 | 31 | 77 | 0.285915 | 31 | 81 | 0.297505 | 31 | 77 | 0.290174 |
| 'trid' | 5000 | 404 | 457 | 38.05434 | 35 | 86 | 7.110678 | 34 | 84 | 6.892447 | 35 | 86 | 6.994909 |
| 'trid' | 10000 | 579 | 635 | 196.8889 | 76 | 154 | 47.65638 | 76 | 169 | 32.48472 | 71 | 149 | 36.09446 |

Table 2: Comparison of the proposed method with HS and Perry algorithms, showing percentage improvements in NI, NF, and CPUT.

| **Comparison with HS** | | | |
|---|---|---|---|
| **Metric** | **HS** | **New** | **New ($\epsilon_m = 1$)** |
| **NI** | 100% | 39.30% | 43.95% |
| **NF** | 100% | 51.86% | 55.36% |
| **CPUT** | 100% | 37.72% | 39.03% |
| **Comparison with Perry** | | | |
| **Metric** | **Perry** | **New** | **New ($\epsilon_m = 1$)** |
| **NI** | 100% | 81.35% | 90.98% |
| **NF** | 100% | 84.30% | 90.00% |
| **CPUT** | 100% | 93.08% | 96.32% |

The comparison shown in Table 2 highlights the performance of the proposed method compared to the classical HS and Perry algorithms, focusing on three key metrics (NI, NF and CPUT):

## Comparison with HS

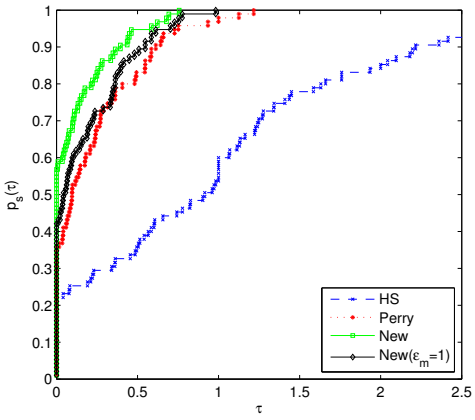
- **NI**: The new method achieves an improvement of 60.70%, and with the specific case ($\epsilon_m = 1$), the improvement slightly decreases to 56.05%.
- **NF**: The new method shows a 48.14% improvement, which further increases to 44.64% in the specific case.
- **CPUT**: The time is reduced by 62.28%, and in the specific case, it is reduced by 60.97%.

## Comparison with Perry

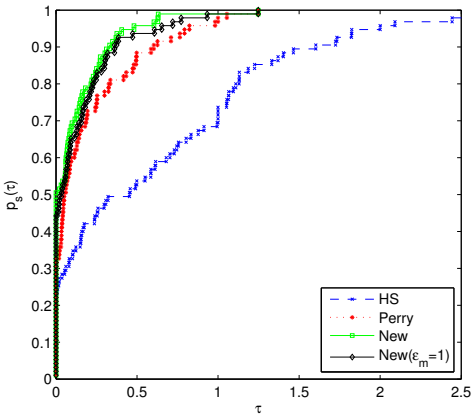
- **NI**: The proposed method shows a significant improvement of 18.65%, and in the specific case, it improves further to 9.02%.
- **NF**: An improvement of 15.70% is observed, which reaches 10.00% in the specific case.

- **CPUT**: The computational time decreases by 6.92%, and in the specific case, it improves even further to 3.68%.

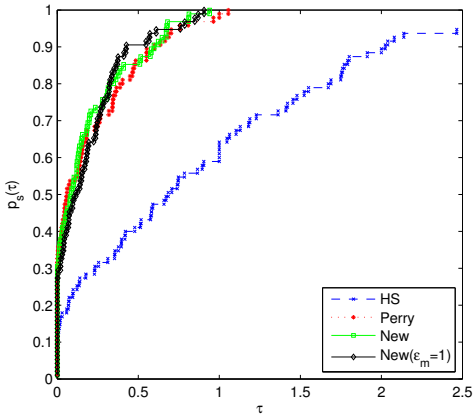
From **Table 2**, it is evident that the proposed method outperforms the HS and Perry algorithms in terms of efficiency, requiring fewer iterations, fewer function evaluations, and less computation time. The specific case ($\epsilon_m = 1$) further enhances the performance, particularly in reducing the computation time. This demonstrates the robustness and efficiency of the new method compared to traditional approaches.




(a) Performance Profile for NI.




(b) Performance Profile for NF.




(c) Performance Profile for CPUT.

Figure 1: Performance profiles for (a) NI, (b) NF, and (c) CPUT.

## 6. Conclusion

In this paper, a new method for solving unconstrained optimization problems based on the CG algorithm was proposed. To evaluate its efficiency, the new method was compared with the classical HS and Perry methods using three performance metrics: the NI, NF, and CPUT. As shown in Table 2, the new method demonstrated significant improvements over the classical methods. Compared to the HS method, the proposed approach achieved a 60.70% reduction in NI, 48.14% in NF, and 62.28% in CPUT. Similarly, when compared to the Perry method, it showed improvements of 18.65% in NI, 15.7% in NF, and 6.92% in CPUT. Furthermore, the proposed method satisfied the descent condition while ensuring global convergence. These characteristics confirm that the method provides not only accurate solutions but also stability and strong performance across a wide range of small- and large-scale optimization problems. Additionally, the special case of the proposed method ($\epsilon_m = 1$) showed further performance enhancements, underscoring the flexibility and robustness of the proposed approach in addressing diverse optimization challenges. These results indicate that the new method is not only more efficient in terms of reducing iterations and processing time but also serves as a competitive and reliable option for solving unconstrained optimization problems, adding a valuable contribution to the family of CG methods.

## References

[1] J. K. Liu and S. J. Li. New hybrid conjugate gradient method for unconstrained optimization. *Applied Mathematics and Computation*, 245:36–43, 2014.

[2] C. Luo, L. Wang, Y. Xie, and B. Chen. A new conjugate gradient method for moving force identification of vehicle-bridge system. *Journal of Vibration Engineering and Technologies*, 12(1):19–36, 2024.

[3] Y. Yuan, D. H. Tsang, and V. K. Lau. Combining conjugate gradient and momentum for unconstrained stochastic optimization with applications to machine learning. *IEEE Internet of Things Journal*, 2024.

[4] S. B. Hanachi, B. Sellami, and M. Belloufi. A new family of hybrid conjugate gradient method for unconstrained optimization and its application to regression analysis. *RAIRO Operations Research*, 58(1):613–627, 2024.

[5] A. L. Ibrahim and M. G. Mohammed. A new conjugate gradient for unconstrained optimization problems and its applications in neural networks. *Indonesian Journal of Electrical Engineering and Computer Science*, 33:93–100, 2024.

[6] B. A. Hassan, I. A. R. Moghrabi, A. L. Ibrahim, and H. N. Jabbar. Improved conjugate gradient methods for unconstrained minimization problems and training recurrent neural networks. *Engineering Reports*, 7:e70019, 2025.

[7] A. Ibrahim and S. Shareef. Modified conjugate gradient method for training neural networks based on logistic mapping. *Journal of University of Duhok*, 22(1):45–51, 2019.

[8] D. H. Omar, A. L. Ibrahim, M. M. Hassan, B. G. Fathi, and D. A. Sulaiman. En-

hanced conjugate gradient method for unconstrained optimization and its application in neural networks. *European Journal of Pure and Applied Mathematics*, 17(4):2692–2705, 2024.

[9] X. Jiang, L. Pan, M. Liu, and J. Jian. A family of spectral conjugate gradient method with strong convergence and its applications in image restoration and machine learning. *Journal of the Franklin Institute*, page 107033, 2024.

[10] B. Balaram, M. D. Narayanan, and P. K. Rajendrakumar. Optimal design of multi-parametric nonlinear systems using a parametric continuation-based genetic algorithm approach. *Nonlinear Dynamics*, 67(4):2759–2777, 2012.

[11] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving. *Journal of Research of the National Bureau of Standards*, 49(1):409–436, 1952.

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

[13] Z. Salleh and A. Alhawarat. An efficient modification of the hestenes-stiefel nonlinear conjugate gradient method with restart property. *Journal of Inequalities and Applications*, page 110, 2016.

[14] Z. Wei, G. Li, and L. Qi. New nonlinear conjugate gradient formulas for large-scale unconstrained optimization problems. *Applied Mathematics and Computation*, 179(2):407–430, 2006.

[15] G. Yuan, Z. Wei, and Q. Zhao. A modified polak–ribière–polyak conjugate gradient algorithm for large-scale optimization problems. *IIE Transactions*, 46(4):397–413, 2014.

[16] P. Wolfe. Convergence conditions for ascent methods. *SIAM Review*, 11(2):226–235, 1969.

[17] P. Wolfe. Convergence conditions for ascent methods. II: Some corrections. *SIAM Review*, 13(2):185–188, 1971.

[18] B. T. Polyak. The conjugate gradient method in extremal problems. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 9(4):807–821, 1969.

[19] E. Polak and G. Ribière. Note sur la convergence de méthodes de directions conjuguées. *Revue Française d'Informatique et de Recherche Opérationnelle*, 16:35–43, 1969.

[20] Y. Liu and C. Storey. Efficient generalized conjugate gradient algorithms, Part 1: Theory. *Journal of Optimization Theory and Applications*, 69:129–137, 1991.

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

[22] R. Fletcher. *Practical Methods of Optimization, Unconstrained Optimization*, volume 1. John Wiley and Sons, New York, 1987.

[23] M. L. N. Goncalves, F. S. Lima, and L. F. Prudente. A study of liu-storey conjugate gradient methods for vector optimization. *Applied Mathematics and Computation*, 425:127099, 2022.

[24] B. A. Hassan and A. A. Saad. Explaining new parameters conjugate analysis based on the quadratic model. *Journal of Interdisciplinary Mathematics*, 26(6):1219–1229, 2023.

[25] B. A. Hassan, H. N. Jabbar, and Y. A. Laylani. Upscaling parameters for conjugate gradient method in unconstrained optimization. *Journal of Interdisciplinary Mathematics*, 26(6):1171–1180, 2023.

[26] B. Hassan and A. Saad. Elastic conjugate gradient methods to solve iteration problems. https://tarupublication.s3south-1.amazonaws.com/articles/jim-1619. Accessed: Dec. 10, 2024.

[27] B. A. Hassan and H. A. Alashoor. On image restoration problems using new conjugate gradient methods. *Indonesian Journal of Electrical Engineering and Computer Science*, 29(3):1438–1445, 2023.

[28] A. Perry. A modified conjugate gradient algorithm. *Operations Research*, 26:1073–1078, 1978.

[29] M. J. D. Powell. Algorithms for nonlinear constraints that use lagrangian functions. *Mathematical Programming*, 14:224–248, 1978.

[30] N. Andrei. Accelerated conjugate gradient algorithm with finite difference hessian/vector product approximation for unconstrained optimization. *Journal of Computational and Applied Mathematics*, 230(2):570–582, 2009.

[31] Y. Dai, J. Han, G. Liu, D. Sun, H. Yin, and Y.-X. Yuan. Convergence properties of nonlinear conjugate gradient methods. *SIAM Journal on Optimization*, 10:345–358, 1999.

[32] N. I. M. Gould, D. Orban, and P. L. Toint. CUTEr and sifdec: A constrained and unconstrained testing environment, revisited. *ACM Transactions on Mathematical Software*, 29:373–394, 2003.

[33] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7:17–41, 1981.

[34] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91:201–213, 2002.

[35] N. Andrei. An unconstrained optimization test functions collection. *Advanced Modeling and Optimization*, 10:147–161, 2008.