STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **15**, January 2026, pp 380–395.
Published online in International Academic Press (www.IAPress.org)

A Novel Hybrid Conjugate Gradient Algorithm for Solving Unconstrained Optimization Problems

Romaissa Mellal*, Nabil Sellami

Laboratory of Analysis and Control of Differential Equations, Department of Mathematics, 8th May 1945 University, Guelma, Algeria

Abstract We introduce a novel hybrid conjugate gradient method for unconstrained optimization, combining the AlBayati-AlAssady and Wei-Yao-Liu approaches, where the convex parameter is determined using the conjugacy condition. Through rigorous theoretical analysis, we establish that the proposed method guarantees sufficient descent properties and achieves global convergence under the strong Wolfe conditions. Using the performance profile of Dolan and Moré, we confirm that our method, denoted as RN, consistently outperforms both classical (HS, FR, PRP and DY CG) and hybrid (BAFR and BADY) methods, particularly for large-scale problems.

Keywords Nonlinear Conjugate Gradient, Unconstrained Optimization, Strong Wolfe Line Search, Scilab.

AMS 2010 subject classifications 90C26, 90C30, 65K05.

DOI: 10.19139/soic-2310-5070-2807

1. Introduction

Optimization, defined as the systematic process of identifying extremal values of mathematical functions, constitutes a cornerstone of quantitative analysis across scientific and engineering disciplines.
When discussing unconstrained optimization, we examine problems where we aim to minimize a function without any restrictions on the variables.
In this paper, we study a general nonlinear unconstrained optimization problem.
This means we are free to explore all possible values for our variables without constraints, formulated as:

$$\min_{x\in\mathbb{R}^n} \mathcal{F}(u). \tag{1}$$

in which $\mathcal{F}:\mathbb{R}^n \mapsto \mathbb{R}$ belongs to class $\mathcal{C}^1$.
Among numerous iterative methods for solving (1), the conjugate gradient methods are considered optimal.
They generate an iterative sequence of the following form:

$$u_{k+1} = u_k + \lambda_k d_k, \tag{2}$$

where u_k represents the current iteration point, λ_k denotes the step length determined through a line search procedure along direction d_k defined by :

$$d_k = \begin{cases} -q_k, \text{ for } k = 1, \\ -q_k + \beta_k d_{k-1}, \text{ for } k \geq 2, \end{cases} . \tag{3}$$

*Correspondence to: Romaissa Mellal (Email: romaissa.mellal@yahoo.fr).

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

where q_k stands for the gradient $\nabla \mathcal{F}(u_k)$, and the scalar parameter β_k is selected to ensure that d_k constitutes the k^{th} conjugate direction, provided that the objective function is quadratic and the line search is performed exactly [4].
Different versions of this method differ in the way of selecting β_k and they are classified into three major categories: classical, hybrid and modified methods. Classical nonlinear conjugate gradient methods are HS [19], FR [13], PRP [25, 26], CD [12], BA [1], LS [23], DY [8] and Wei-Yao-Liu [28]. Their formulas of β_k are given by :

$$\beta_k^{HS} = \frac{q_{k+1}^T y_k}{d_k^T y_k},\ Hestenes - Stiefel \tag{4}$$

$$\beta_k^{FR} = \frac{||q_{k+1}||^2}{||q_k||^2},\ Fletcher - Reeves \tag{5}$$

$$\beta_k^{PRP} = \frac{q_{k+1}^T y_k}{\|q_k\|^2},\ Polak - Ribiere - Polyak \tag{6}$$

$$\beta_k^{CD} = \frac{-||q_{k+1}||^2}{q_k^T d_k},\ Fletcher \tag{7}$$

$$\beta_k^{BA} = \frac{||y_k||^2}{d_k^T y_k},\ AlBayati - AlAssady \tag{8}$$

$$\beta_k^{LS} = \frac{-q_{k+1}^T y_k}{q_k^T d_k},\ Liu - Storey \tag{9}$$

$$\beta_k^{DY} = \frac{\|q_{k+1}\|^2}{d_k^T y_k},\ Dai - Yuan \tag{10}$$

where $y_k = (q_{k+1} - q_k)$ represents the gradient difference.
It is noteworthy that these methods are equivalent when applied to strongly convex quadratic functions under exact line search.
Note that the convergence behavior of all nonlinear CG methods cited above under strong Wolfe's rule [29, 30] has been widely studied by numeros authors [4].
After that, in 2006, Wei et al. [28] introduced a novel variant of CGM, designated as the WYL method.
This approach is a modification of the PRP method, with the parameter given by the following formula:

$$\beta_k^{WYL} = \frac{q_{k+1}^T \left(q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k\right)}{\|q_k\|^2}, \quad \text{(Wei–Yao–Liu)} \tag{11}$$

In 2011, Huang et al. [20] proposed a modified version of the WYL method, further enhancing its robustness and global convergence. Hybridization of CGM via convex combinations has emerged as a significant research direction in optimization science. Recent studies have established strong theoretical foundations for these hybrid methods, demonstrating that they maintain crucial essential properties such as sufficient descent conditions and convergence under appropriate line search criteria. This line of research continues to attract considerable scholarly interest, as it advances the methodology of nonlinear optimization, with applications extending to image restoration, regression analysis, robot control and portfolio selection. [2, 5, 16, 17, 14, 18]
Various hybridization techniques that combine the WYL method with other formulations have been extensively documented in the optimization literature. For instance, in 2014, [24] discussed the adaptation of the WYL method and its variants for multiobjective optimization, enabling the computation of Pareto optimal solutions without relying on subjective weighting schemes. In 2016, [31] proposed a hybrid method combining the WYL and Dai–Liao approaches, demonstrating global convergence under Wolfe line search conditions. In 2019, [21] showed that the hybridization of PRP and WYL methods enhances convergence for a range of optimization problems. In 2024, [2] applied and further modified the WYL formula for robot control, achieving superior efficiency compared

to the classical version. Most recently, in 2024 [14] introduced a significant hybrid approach that combines the WYL and CD (Conjugate Descent) methods, where the convex parameter θ_k is determined using the conjugacy condition $d_{k+1}^T y_k = 0$.
Similarly, numerous BA-based hybrid methods as a hybrid approach that combines the BA and other CGM, in 2020 we cite BA-FR [9] and BA-DY [15] hybrid CGM.
Another notable hybridization, combining BA and HZ methods, was proposed in 2025 [6], demonstrating promising results.
Some of hese hybrid approaches employ the following parameter formulations:

$$\beta_k^{WYLPRP} = \max(\beta_k^{PRP}, \beta_k^{WYL}); \tag{12}$$

$$\beta_k^{WYLCD} = \theta_k \beta_k^{CD} + (1-\theta_k)\beta_k^{WYL}; \tag{13}$$

$$\beta_k^{BAFR} = \theta_k \beta_k^{BA} + (1-\theta_k)\beta_k^{FR}, \tag{14}$$

$$\beta_k^{BADY} = \theta_k \beta_k^{BA} + (1-\theta_k)\beta_k^{DY}, \tag{15}$$

$$\beta_k^{BAHZ} = \theta_k \beta_k^{HZ} + (1-\theta_k)\beta_k^{BA}, \tag{16}$$

All these formulations preserve the descent property under strong Wolfe conditions while demonstrating superior numerical performance compared to traditional methods. They maintain sufficient descent properties and global convergence under appropriate conditions.
To effectively combine the strengths of both BA and WYL methods and develop a more efficient algorithm, where the combination parameter is computed to satisfy the conjugacy condition. These new algorithms generate sufficient descent directions and exhibit global convergence under strong Wolfe conditions.
We evaluate the method's performance using the Dolan and Moré performance profile. These experiments include 30 standard benchmark functions selected from [3, 7] and show that RN method outperforms both classical (HS, FR, PRP and DY CG) and hybrid (BAFR and BADY) methods, particularly for large-scale problems.
The remainder of this manuscript is structured as follows: Section 2 introduces the novel formulation of the parameter and delineates the corresponding algorithmic framework. Additionally, we provide a comprehensive analysis of the descent properties exhibited by the derived search direction, followed by a rigorous demonstration of the algorithm's global convergence characteristics under strong Wolfe line search conditions.
Section 3 presents extensive numerical experiments evaluating the performance of our algorithm against established methods in the field. The final section concludes the paper with a summary of our findings.

2. New hybrid CG

In this part, a newly proposed CG method is introduced.
The approach constructs a hybrid CG parameter, β_k^{RN}, as a convex combination of the Al-Bayati & Al-Assady (BA) and Wei-Yao-Liu (WYL) parameters:

$$\beta_k^{RN} = \theta_k \beta_k^{BA} + (1-\theta_k)\beta_k^{WYL}. \tag{17}$$

Note that θ_k is a scalar parameter bounded by $0 \leq \theta_k \leq 1$ and here, β_k^{BA} and β_k^{WYL} are given by (11) .

The search direction d_{k+1}^{RN} is defined recursively:

$$d_{k+1}^{RN} = \begin{cases} -q_{k+1}, & \text{for } k = 0, \\ -q_{k+1} + \beta_k^{RN} d_k, & \text{for } k \geq 1, \end{cases} \tag{18}$$

where $q_k = \nabla \mathcal{F}(u_k)$ denotes the gradient.
The step size λ_k in the iteration $u_{k+1} = u_k + \lambda_k d_k$ is determined using the strong Wolfe conditions SWC [29, 30]:

$$\begin{aligned} \mathcal{F}(u_k + \lambda_k d_k) &\leq \mathcal{F}(u_k) + \rho \lambda_k q_k^T d_k, \\ |q_{k+1}^T d_k| &\leq \sigma |q_k^T d_k|, \end{aligned} \tag{19}$$

with constants $0 < \rho < \sigma < 1$.
It is evident that, if $\theta_k = 0$ then $\beta_k^{RN} = \beta_k^{WYL}$ and if $\theta_k = 1$ then $\beta_k^{RN} = \beta_k^{BA}$.
We consider two possibilities for selecting the parameter θ_k:
In this approach, θ_k is selected to ensure that the conjugacy condition is satisfied at each iteration.
From the recurrence relation in (18), we have:

$$\begin{aligned} d_{k+1}^{BAWYL} = -q_{k+1} + \beta_k^{BAWYL} d_k &= -q_{k+1} + \theta_k \beta_k^{BA} d_k + (1-\theta_k)\beta_k^{WYL} d_k, \\ &= -q_{k+1} + \theta_k \frac{||y_k||^2}{d_k^T y_k} d_k + (1-\theta_k) \frac{q_{k+1}^T (q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k)}{\|q_k\|^2} d_k. \end{aligned}$$

Multiplying both sides of the recurrence relation above by y_k and imposing the conjugacy condition $d_{k+1}^T y_k = 0$ yields

$$0 = -q_{k+1}^T y_k + \theta_k \frac{||y_k||^2}{d_k^T y_k} d_k^T y_k + (1-\theta_k) \frac{q_{k+1}^T (q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k)}{\|q_k\|^2} d_k^T y_k,$$

Simplifying this expression yields:

$$\theta_k = \frac{q_{k+1}^T y_k - \dfrac{q_{k+1}^T \left(q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k\right)}{\|q_k\|^2} d_k^T y_k}{\|y_k\|^2 - \dfrac{q_{k+1}^T \left(q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k\right)}{\|q_k\|^2} d_k^T y_k}$$

$$\theta_k = \frac{q_{k+1}^T y_k - \left(\dfrac{\|q_{k+1}\|^2 - \frac{\|q_{k+1}\|}{\|q_k\|} (q_{k+1}^T q_k)}{\|q_k\|^2} d_k^T y_k \right)}{\|y_k\|^2 - \left(\dfrac{\|q_{k+1}\|^2 - \frac{\|q_{k+1}\|}{\|q_k\|} (q_{k+1}^T q_k)}{\|q_k\|^2} d_k^T y_k \right)}$$

$$= \frac{\|q_k\|^2 \; q_{k+1}^T y_k - \left(\|q_{k+1}\|^2 - \frac{\|q_{k+1}\|}{\|q_k\|} q_{k+1}^T q_k \right) d_k^T y_k}{\|q_k\|^2 \; \|y_k\|^2 - \left(\|q_{k+1}\|^2 - \frac{\|q_{k+1}\|}{\|q_k\|} q_{k+1}^T q_k \right) d_k^T y_k}$$

We obtain :

$$\theta_k^{rn} = \frac{\|q_k\|^3 \; q_{k+1}^T y_k - \left(\|q_{k+1}\|^2 \|q_k\| - \|q_{k+1}\| q_{k+1}^T q_k \right) d_k^T y_k}{\|q_k\|^3 \; \|y_k\|^2 - \left(\|q_{k+1}\|^2 \|q_k\| - \|q_{k+1}\| q_{k+1}^T q_k \right) d_k^T y_k}$$

To ensure numerical stability, we regularize θ_k^{RN} as:

$$\theta_k^{RN} = \begin{cases} \theta_k^{rn}, & \text{if } 0 < \theta_k^{rn} < 1, \\ 0, & \text{if } \theta_k^{rn} \leq 0, \\ 1, & \text{if } \theta_k^{rn} \geq 1. \end{cases} \tag{20}$$

3. RN ALGORITHM

In this part, we present the RN algorithm.

Algorithm 1: RN CG Algorithm

Step 1.
Select $u_0 \in \mathbb{R}^n$ and $0 < \rho < \sigma < 1$. Calculate $\mathcal{F}(u_0)$ and $q_0 = \nabla\mathcal{F}(u_0)$. Define $d_0 = -q_0$.
Step 2.
If $\|q_k\|_\infty \leq \epsilon$, then stop. Otherwise, go to step 3.
Step 3.
Using SWC (19), calculate the stepsize λ_k, then compute $u_{k+1} = u_k + \lambda_k d_k$. Calculate $\mathcal{F}(u_{k+1})$ and $q_{k+1} = \nabla\mathcal{F}(u_{k+1})$
Step 4. Calculate $\theta_k = \theta_k^{RN}$ (20).
Step 5. Calculate $\beta_k^{RN} = \theta_k\beta_k^{BA} + (1-\theta_k)\beta_k^{WYL}$.
Step 6: : If $|q_{k+1}^T q_k| \geq 0.2\|q_{k+1}\|^2$ then set
$d_{k+1}^{RN} = -q_{k+1}$ and $\lambda_{k+1} = 1$, else calculate $d_{k+1}^{RN} = -q_{k+1} + \beta_k d_k$.
Step 7: Put $k = k+1$ and go to Step 2.

4. Convergence Analysis

Global convergence is established by initially demonstrating the theorem that follows, which proves the sufficient descent of RN direction.
In this section, we assume that:

Assumption 1
1) Let

$$\Gamma = \{\, u \in \mathbb{R}^n \mid \mathcal{F}(u) \leq \mathcal{F}(u_1) \,\}$$

be bounded.

2) In some neighborhood N of Γ, $\mathcal{F} \in \mathcal{C}^1$ and $q = \bigtriangledown\mathcal{F}$ is Lipschitz continuous, i.e.,

$$\exists\, L > 0 \text{ such that } \|q(u_1) - q(u_2)\| \leq L\|u_1 - u_2\|, \quad \forall\, u_1, u_2 \in \Gamma. \tag{21}$$

Under Assumption 1, we obtain :

$$\exists\, B,\ M > 0 : \|u\| \leq B, \qquad and \qquad ||q\,(u)\,|| \leq M, \qquad \forall u \in \Gamma \tag{22}$$

Theorem 1
The direction generated by RN CG algorithm satisfies the sufficient descent condition:

$$q_k^T d_k^{RN} \leq -c||q_k||^2,\ \ \forall k\ \geq 0. \tag{23}$$

Proof
We will demonstrate this result using mathematical induction.
If $k = 0$, then $q_0^T d_0^{RN} = -\|q_0\|^2$, so (23) holds.
On the other hand, for $k > 0$, we have :

$$\begin{aligned} d_{k+1}^{RN} &:= -q_{k+1} + \beta_k^{RN} d_k \\ &= -q_{k+1} + \left((1-\theta_k)\beta_k^{WYL} + \theta_k\beta_k^{BA}\right) d_k \\ &= (1-\theta_k)(-q_{k+1} + \beta_k^{WYL} d_k) + \theta_k(-q_{k+1} + \beta_k^{BA} d_k). \end{aligned}$$

It follows that:

$$d_{k+1}^{RN} = \theta_k d_k^{BA} + (1-\theta_k) d_k^{WYL}.$$

Multiplying by q_{k+1}^T from the left, we get

$$q_{k+1}^T d_{k+1}^{RN} = \theta_k q_{k+1}^T d_{k+1}^{BA} + (1-\theta_k)\, q_{k+1}^T d_{k+1}^{WYL}. \tag{24}$$

Firstly if $\theta_k = 0$, the direction d_k^{RN} becomes identical to the descent direction of WYL, yielding:

$$d_{k+1}^{RN} = d_{k+1}^{WYL} = -q_{k+1} + \beta_k^{WYL} d_k,$$

where they proved in [28] that

$$\begin{aligned} q_{k+1}^T d_{k+1}^{WYL} &= -q_{k+1}^T q_{k+1} + \beta_k^{WYL} q_{k+1}^T d_k \\ &= -\|q_{k+1}\|^2 + \frac{q_{k+1}^T (q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k)}{\|q_k\|^2} q_{k+1}^T d_k \\ &\leq -c_1 \|q_{k+1}\|^2, \text{ for all } k. \end{aligned} \tag{25}$$

Secondly, if $\theta_k = 1$, we have :

$$d_{k+1}^{RN} = d_{k+1}^{BA} = -q_{k+1} + \beta_k^{BA} d_k.$$

The descent property of the BA method was previously established in the literature [9], however, a complete proof is provided here for completeness and clarity. where they proved in [9] that

$$q_{k+1}^T d_{k+1}^{BA} \leq -c_2 \|q_{k+1}\|^2 \tag{26}$$

where $c_2 = 1 - \omega > 0$.

In fact, let

$$d_{k+1}^{BA} = -q_{k+1} + \beta_k^{BA} d_k.$$

Upon left multiplication by q_{k+1}^T, it follows that

$$\begin{aligned} q_{k+1}^T d_{k+1}^{BA} &= -q_{k+1}^T q_{k+1} + \beta_k^{BA} q_{k+1}^T d_k \\ &= -\|q_{k+1}\|^2 + \frac{\|y_k\|^2}{d_k^T y_k} q_{k+1}^T d_k \quad \text{for all } k. \end{aligned}$$

By invoking the strong Wolfe condition (19), we obtain:

$$q_{k+1}^T d_k \leq \frac{\sigma}{1-\sigma} \frac{\|y_k\|^2}{d_k^T y_k},$$

Combining this with the Lipschitz condition (21):

$$q_{k+1}^T d_{k+1}^{BA} \leq -\|q_{k+1}\|^2 + \frac{\sigma L^2}{1-\sigma} \|s_k\|^2$$

Under Assumption 1 ($\|s_k\| \to 0$), there exists $\omega \in (0,1)$ such that:

$$\frac{\sigma L^2}{1-\sigma} \|s_k\|^2 \leq \omega \|q_{k+1}\|^2$$

Consequently:

$$q_{k+1}^T d_{k+1}^{BA} \leq -(1-\omega)\|q_{k+1}\|^2 = -c_2 \|q_{k+1}\|^2.$$

where $c_2 = 1 - \omega > 0$.

Finally, if $0 < \theta_k < 1$, then :

$$\exists\, \eta_1, \eta_2 \in \mathbb{R} : 0 < \eta_1 \leq \theta_k \leq \eta_2 < 1, \tag{27}$$

From formulas (24) and (27), we conclude

$$q_{k+1}^T d_{k+1}^{RN} \leq \eta_1 q_{k+1}^T d_{k+1}^{BA} + (1-\eta_2)\, q_{k+1}^T d_{k+1}^{WYL},$$

Let $c = \eta_1 c_2 + (1-\eta_2)\, c_1$, then from (25) and (26) we finally get

$$q_{k+1}^T d_{k+1}^{RN} \leq -c \left\| q_{k+1} \right\|^2.$$

□

Lemma 1
Under Assumption 1, consider any CGM defined by (2) and (3), with step length λ_k determined through the SWL (19). If

$$\sum_{k\geq 0} \frac{1}{\left\| d_k \right\|^2} < \infty,$$

Then

$$\lim_{k\to\infty} \inf \left\| q_k \right\| = 0.$$

Proof
For the proof, see [4] . □

Lemma 2
Assuming that Assumption 1 is satisfied, if d_k is a descent direction and λ_k satisfies

$$q_{k+1}^T d_k \geq \sigma q_k^T d_k, : \text{where} : 0 < \sigma < 1,$$

then

$$\lambda_k \geq \frac{(1-\sigma) q_k^T d_k}{L |d_k|^2}. \tag{28}$$

Proof
For the proof of Lemma 2, refer to the work in [22]. □

The following theorem establishes the global convergence properties of our proposed method.

Theorem 2
Consider the RN CG algorithm and suppose that assumption 1 holds. Then either $q_k = 0$, for some k, or

$$\lim \inf_{k\to\infty} \left\| q_k \right\| = 0. \tag{29}$$

Proof
Consider the RN conjugate gradient method and suppose that Assumption 1 holds. Suppose that $q_k \neq 0$, for all k. We prove (29) by contradiction. Assume that (29) does not hold, then

$$\exists\, t > 0 : \left\| q_k \right\| \geq t,\ \forall\, k, \tag{30}$$

according to the relation (19) and (23), we obtain:

$$q_k^T d_k > c(1-\sigma) \|q_k\|^2 > c(1-\sigma) t^2. \tag{31}$$

By using the Lipschitz condition, we get:

$$\|y_k\| = \|q_{k+1} - q_k\| \leq LB, \tag{32}$$

where $B = \max\{\|x-y\|, x, y \in \Gamma\}$ is the diameter of Γ.
In other hand, let :

$$\begin{aligned}\|Y_k\| &= \left\| q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|} q_k \right\| \\ &= \left\| q_{k+1} - q_k + q_k - \frac{\|q_{k+1}\|}{\|q_k\|} q_k \right\| \\ &\leq 2\|q_{k+1} - q_k\| \\ &\leq 2LB\end{aligned}$$

We have

$$d_{k+1}^{RN} = -q_{k+1} + \theta_k \beta_k^{BA} d_k + (1-\theta_k)\beta_k^{WYL} d_k,$$

since, $0 < \theta_k < 1$, we obtain:

$$\|d_{k+1}^{RN}\| \leq \|q_{k+1}\| + (|\beta_k^{BA}| + |\beta_k^{WYL}|)\|d_k\|.$$

We have :

$$\left|\beta_k^{BA}\right| = \frac{||y_k||^2}{\left|d_k^T y_k\right|} \leq \frac{L^2\|s\|^2}{(1-\sigma)ct^2} \leq \frac{L^2B^2}{(1-\sigma)ct^2}. \tag{33}$$

Otherwise

$$\left|\beta_k^{WYL}\right| = \frac{\left|q_{k+1}^T(q_{k+1} - \frac{\|q_{k+1}\|}{\|q_k\|}q_k)\right|}{\|q_k\|^2} \leq \frac{2MLB}{t^2}. \tag{34}$$

Using the above relations (33) and (34), we obtain:

$$\begin{aligned}\|d_{k+1}^{RN}\| \leq \|q_{k+1}\| + (|\beta_k^{BA}| + |\beta_k^{WYL}|)\|d_k\| &\leq M + \left(\frac{L^2B^2}{(1-\sigma)ct^2} + \frac{2MLB}{t^2}\right)\frac{\|s_k\|}{\lambda} \\ &\leq M + \left(\frac{LB\,(LB + 2M(1-\sigma)c)}{(1-\sigma)ct^2}\right)\frac{B}{\lambda} \\ &\leq \zeta, \quad \text{for all } k\end{aligned}$$

where

$$\zeta = M + \frac{LB^2\,(LB + 2M(1-\sigma)c)}{(1-\sigma)c\lambda t^2} \tag{35}$$

Therefore we obtain

$$\sum_{k\geq 0} \frac{1}{\|d_{k+1}^{RN}\|^2} = +\infty. \tag{36}$$

By applying Lemma 1, we conclude that:

$$\liminf_{k\to\infty} \|q_k\| = 0. \tag{37}$$

This contradicts (30) , and thus we have proved (29) . □

5. Experimental results

In this section, we evaluate the performance of our proposed RN algorithm for solving unconstrained optimization problems of the form presented in equation (1).
To assess the effectiveness and convergence characteristics of our RN algorithm, we conducted comprehensive numerical experiments on a benchmark suite of 30 well-established test problems drawn from [3, 7] given in table 1 . The evaluation encompasses problems of varying complexity across low, medium, and high dimensions, with

problem sizes ranging from $n = 2$ to $n = 10000$.
We compare the performance of our RN method against two classes of conjugate gradient algorithms: classical conjugate gradient methods (HS, FR, PRP, and DY) [19, 13, 25, 26, 8] and hybrid conjugate gradient methods (BADY and BAFR) [15, 9], all implemented under the strong Wolfe conditions (19).

Table 1. List of Test Problems with Initial Points Employed in the Computational Framework

No.	Test Functions	Dimensions	Initial Points
1	Rosenbrock	2, 5, 10	$(-1.2, -1.2)$ for $n = 2$, (1.2,...,1.2) for $n > 2$
2	Sphere	2	(0.5, 0.5)
3	Sum of Squares	2, 5, 10, 100, 1500, 5000, 10000	(1, 1,...,1)
4	Zakharov	2, 5, 10, 100, 1500, 5000, 10000	(1, 1,...,1)
5	Dixon-Price	2, 5, 10, 1500	(2, 2,...,2)
6	Quadratic Function QF1	2, 5, 10, 100, 1000, 1500, 5000	(1, 1,...,1)
7	Raydan 1	2, 5, 10, 100, 1500, 5000	(0.5, 0.5,...,0.5)
8	Raydan 2	2	(1, 1)
9	Extended Rosenbrock	2, 5, 10, 100, 1000, 1500, 5000	(-1.2, -1.2,...,-1.2)
10	Extended DENSCHNF	2, 5, 10, 100, 1000, 1500, 5000	(1, 1,...,1)
11	Extended Tridiagonal	2, 5, 10, 1500	(0, 0,...,0)
12	Extended Himmelblau	2, 5, 10, 100, 1000, 1500	(1, 1,...,1)
13	DBVF	2, 5, 10	(0.1, 0.1,...,0.1)
14	BRYBND	2, 5, 10, 100	(-1, -1,...,-1)
15	Perturbed Quadratic	2, 5, 10, 100	(0.5, 0.5,...,0.5)
16	TRIDIA	2, 5, 10, 100	(1, 1,...,1)
17	Extended Penalty	2, 5, 10, 100, 1000	(1, 1,...,1)
18	BALF	2, 5, 10, 100, 1000	(0.5, 0.5,...,0.5)
19	Diagonal 1	2, 5, 10, 100	(2, 2,...,2)
20	Diagonal 2	2, 5, 10, 100	(2, 2,...,2)
21	Diagonal 3	2, 5, 10, 100, 1000	(0, 0,...,0)
22	Diagonal 4	2, 5, 10, 100, 1000	(1, 1,...,1)
23	Extended Diagonal	100, 1000	(1, 1,...,1)
24	Beale	2, 1500	(1, 1)
25	Booth	2, 1500	(0, 0)
26	Ackley	2, 5, 10	(2, 2,...,2)
27	Rastrigin	2	(1.5, 1.5)
28	Griewank	2, 5, 10	(10, 10,...,10)
29	Matyas	2, 5	(1, 1) for n=2, modified for n¿2
30	Schwefel	2, 5, 10, 1500	(400, 400,...,400)

The codes are written in Python 3.13 and run on a Lenovo Thinkpad PC with AMD Ryzen 7 PRO 5850U Processor with Radeon Graphics 1.90 GHz and 16.0 GB RAM and Windows 10 Professional 64 bits operating system.
We stop the program when $\|q_k\|_\infty < \epsilon$ holds or the number of iterations atteind 5000.
We restart by taking the direction of the steepest descent if either the denominator of β is zero or the Powell restart criterion [27] $\mid q_{k+1}^T g_k \mid \geq 0.2 \parallel q_{k+1} \parallel^2$ is satisfied.
We used the parameters values of $\rho = 10^{-4}$ and $\sigma = 0.9$ for the SWC.
Numerical results are compared based on the number of iterations, CPU time and the number of evaluation of the function and they are given in Tables 2, 3 and 4 whose columns have the following meanings:
Function : the name of the function test problem;
Dim : the dimension of the problem;
ϵ : stop criteria;
NOI : the number of iterations;
NFEV : denotes the number of function evaluation.
CPU : denotes the total CPU time which should be taken to compute all of these problems.
To rigorously assess and compare the performance of the tested optimization methods, we adopt the performance profiles of Dolan and Moré [11] as our primary evaluation metric. This approach provides a comprehensive and

statistically robust means of comparing the relative efficiency and reliability of a set of solvers over a collection of benchmark problems. Let S denote the set of solvers, P the set of test problems, n_s the number of solvers, and n_p the number of problems. For each problem $p \in P$ and each solver $s \in S$, let $a_{p,s}$ represent the numerical performance measure of interest, such as the number of iterations or CPU time required to solve problem p with solver s. The performance ratio $r_{p,s}$ for solver s on problem p is defined as:

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

By definition, $r_{p,s} \geq 1$ for all p and s, and a value of 1 indicates that solver s achieved the best performance for that particular problem.

The performance profile for solver s is the cumulative distribution function:

$$\sigma_s(\lambda) = \frac{1}{n_p} \left|\{p \in P : r_{p,s} \leq \lambda\}\right|,$$

where $\lambda \geq 1$ is a scaling factor and $\sigma_s(\lambda)$ represents, for each λ, the fraction of problems where the performance ratio of solver s does not exceed λ. When plotted, these profiles visually convey the comparative strengths of the solvers: a curve higher and to the left indicates that the solver was able to solve a larger proportion of problems closer to the best possible performance.

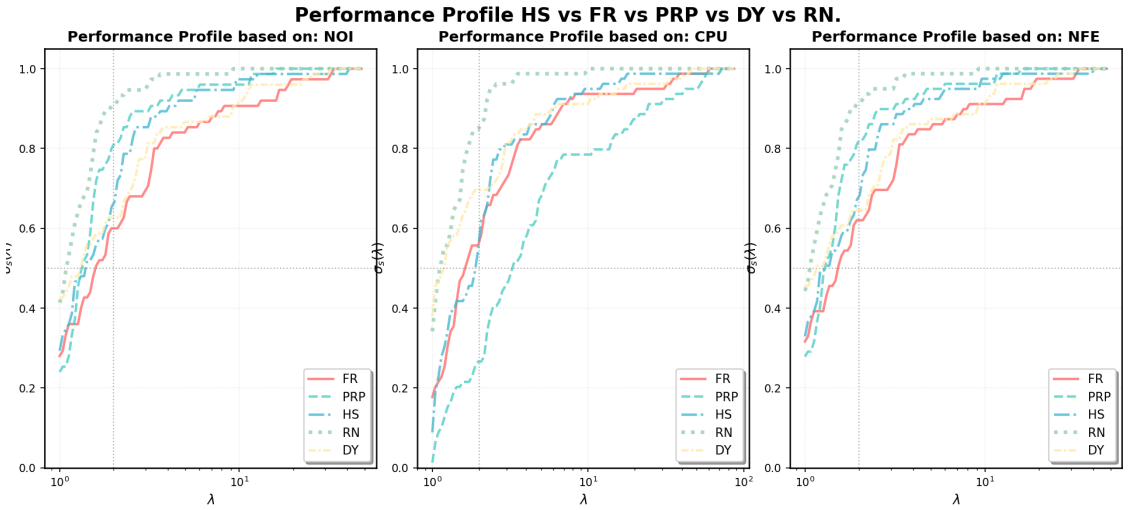
6. Commentaires

Our numerical experiments conducted on 30 test functions listed in table (1) and selected from [3, 7], are summarized in in Tables (2), (3), (4) and (5).

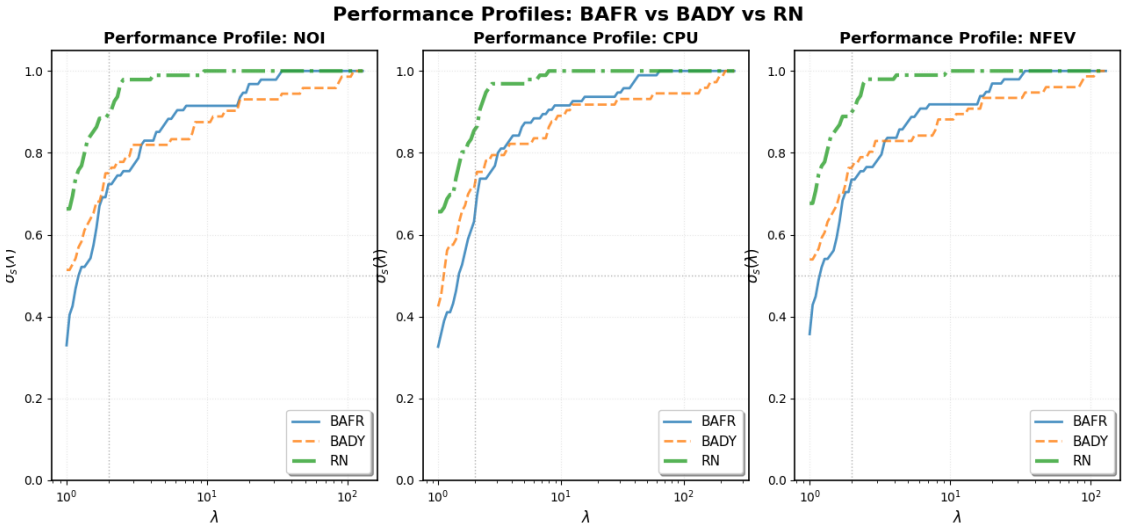
The results show that the RN method demonstrates a remarkable competitive advantage over all tested optimization algorithms.

When compared to classical conjugate gradient methods, RN achieves a success rate of 97.3%, significantly outperforming Fletcher-Reeves (FR) at 82.1%, Polak-Ribiere-Polyak (PRP) at 85.5%, Hestenes-Stiefel (HS) at 89.0%, and Dai-Yuan (DY) at 88.2%. The performance profiles reveal that RN's curve consistently maintains the highest position across all performance ratios λ, indicating superior efficiency in terms of number of iterations, CPU time and function evaluations.

Against hybrid methods, RN establishes clear dominance with its 97.3% success rate compared to BAFR's 91.6% and BADY's 78.4%. The method's exceptional performance is particularly evident on challenging problems such as the Rosenbrock function, where RN successfully converged across all dimensions while PRP completely failed, and on high-dimensional instances of Sum of Squares and Zakharov functions where classical methods frequently exceeded the 5000 iteration limit. Analysis using the performance profile of More and Dolan given by figures (1a) and (1b) which evaluates methods based on the number of iterations, number of evaluation functions and CPU, reveals that RN consistently maintains the leading curve and successfully solves the majority of test functions, across both classical and hybrid conjugate gradient methods. When compared with classical methods HS, FR, PRP and DY, RN demonstrates the highest curve position across all metrics, achieving optimal performance on the largest proportion of BAFR demonstrates intermediate performance with approximately 91.6 % success rate, while BADY shows the poorest performance among hybrid methods at 78% success rate. The performance profiles validate that RN achieves the optimal combination of efficiency and reliability, confirming its 97.3% success rate and establishing it as the most effective method for unconstrained optimization across diverse problem landscapes. Despite its exceptional performance, the RN method presents certain limitations including increased computational complexity per iteration due to the dynamic calculation of the convex parameter θ through the conjugacy condition, and potential sensitivity to parameter selection in specific problem instances. Additionally, the hybrid approach requires greater implementation complexity and marginally higher memory overhead compared to classical conjugate gradient methods.



(a) Performance Profile RN vs Classical CG



(b) Performance Profile RN vs Hybrid CG

Figure 1. Performance Profile of RN based on NOI, CPU and NFEV

7. Conclusion

In this work, we have introduced and analyzed a novel hybrid conjugate gradient method denoted RN CGM. The RN method is based on a convex combination of the BA and WYL methods, with its parameter specifically chosen to satisfy the conjugacy condition. We provided a rigorous convergence analysis under SW conditions and demonstrated that the proposed algorithm satisfy the sufficient descent property, thereby ensuring global convergence. Extensive numerical experiments on a diverse set of test functions revealed that the RN method, in particular, outperforms several established algorithms including classical (HS, FR, PRP and DY) and hybrid CGM (BAFR and BADY) in terms of both efficiency and robustness. Performance profiles further highlighted RN's superiority with respect to success rate, iteration count, and computational time.

Table 2. Comparison of: FR vs PRP vs HS vs DY vs RN Methods on 30 Test Functions

Function	Dim	ϵ	FR			PRP			HS			DY			RN		
			NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU
Rosenbrock	2	10^{-6}	212	21202	0.0567	Failed	500001	1.2545	67	6702	0.0260	151	15102	0.0358	104	10402	0.0246
	5	10^{-6}	218	21802	0.0854	Failed	500001	1.6564	118	11802	0.0366	320	32002	0.1057	187	18702	0.0582
	10	10^{-6}	899	89902	0.3721	Failed	500001	1.7546	177	17702	0.0560	489	48902	0.2097	180	18002	0.0685
Sphere	2	10^{-6}	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001
SumSquares	2	10^{-6}	33	3302	0.0014	2	202	0.0001	19	1902	0.0009	18	1802	0.0007	2	202	0.0001
	5	10^{-6}	54	5402	0.0026	32	3202	0.0028	68	6802	0.0053	35	3502	0.0027	32	3202	0.0038
	10	10^{-6}	113	11302	0.0105	60	6002	0.0040	70	7002	0.0048	44	4402	0.0029	34	3402	0.0024
	100	10^{-6}	Failed	20001	0.1359	Failed	20001	0.1126	100	10002	0.0546	99	9902	0.0547	141	14102	0.0807
	1500	10^{-1}	353	35302	4.6787	Failed	500001	458.0429	177	17702	2.0455	157	15702	1.8062	229	22902	2.7073
	5000	10^{-1}	464	46402	32.52	Failed	100001	671.23	335	33502	21.24	326	32602	22.46	386	38602	24.49
	10000	10^{-1}	814	81402	211.10	Failed	100001	1686.48	543	54302	147.02	472	47202	133.46	530	53002	145.42
Zakharov	2	10^{-6}	32	3202	0.0054	10	1002	0.0022	15	1502	0.0025	18	1802	0.0029	10	1002	0.0020
	5	10^{-6}	38	3802	0.0075	111	11102	0.0230	145	14502	0.0336	25	2502	0.0065	90	9002	0.0208
	10	10^{-6}	67	6702	0.0191	675	67502	0.2066	57	5702	0.0124	102	10202	0.0296	154	15402	0.0429
	100	10^{-6}	Failed	20001	0.1316	Failed	20001	0.1249	Failed	20001	0.7076	Failed	20001	0.1323	152	15202	0.0998
	1500	10^{-1}	222	22202	0.3224	Failed	500001	26.1667	30	3002	0.0593	3469	346902	4.6746	195	19502	0.2659
	5000	10^{-1}	99	9902	0.42	Failed	100001	13.19	Failed	100001	12.44	Failed	100001	4.04	220	22002	0.87
	10000	10^{-1}	169	16902	1.83	Failed	100001	53.92	Failed	100001	33.51	83	8302	1.05	574	57402	23.71
Dixon-Price	2	10^{-6}	91	9102	0.0181	99	9902	0.0169	25	2502	0.0094	51	5102	0.0100	30	3002	0.0059
	5	10^{-6}	1027	102702	0.3053	140	14002	0.0220	97	9702	0.0178	196	19602	0.0440	53	5302	0.0075
	10	10^{-6}	134	13402	0.0328	168	16802	0.0317	75	7502	0.0200	225	22502	0.0639	81	8102	0.0173
	1500	10^{-1}	Failed	500001	269.31	Failed	500001	2162.79	1801	180102	73.02	619	61902	26.89	539	53902	21.93
Quadratic-QF1	2	10^{-6}	10	1002	0.0011	9	902	0.0008	17	1702	0.0016	10	1002	0.0011	9	902	0.0018
	5	10^{-6}	11	1102	0.0011	21	2102	0.0020	32	3202	0.0036	10	1002	0.0010	21	2102	0.0023
	10	10^{-6}	28	2802	0.0041	13	1302	0.0028	41	4102	0.0077	18	1802	0.0027	13	1302	0.0029
	100	10^{-6}	67	6702	0.1131	62	6202	0.1167	142	14202	0.2639	38	3802	0.0641	59	5902	0.1102
	1000	10^{-3}	127	12702	4.2392	Failed	20001	6.2574	Failed	20001	6.8460	111	11102	3.1244	150	15002	4.6312
	1500	10^{-1}	194	19402	26.9728	Failed	20001	19.7245	151	15102	15.7204	149	14902	15.0424	127	12702	12.4793
	5000	10^{-1}	213	21302	84.17	137	13702	103.59	118	11802	39.54	124	12402	45.51	167	16702	59.52
Raydan1	2	10^{-6}	28	2802	0.0016	8	802	0.0005	99	9902	0.0045	14	1402	0.0007	8	802	0.0005
	5	10^{-6}	92	9202	0.0089	27	2702	0.0021	65	6502	0.0050	27	2702	0.0019	27	2702	0.0022
	10	10^{-6}	38	3802	0.0061	52	5202	0.0079	73	7302	0.0106	29	2902	0.0041	30	3002	0.0046
	100	10^{-6}	187	18702	0.4737	Failed	20001	0.4320	94	9402	0.1927	70	7002	0.1326	131	13102	0.2788
	1500	10^{-1}	186	18602	9.36	177	17702	14.26	134	13402	6.16	133	13302	6.01	154	15402	7.09
	5000	10^{-1}	496	49602	193.80	Failed	100001	5557.29	256	25602	67.67	283	28302	77.68	230	23002	63.30
Raydan2	2	10^{-6}	1	102	0.0002	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0002
Extended Rosenbrock	2	10^{-6}	212	21202	0.0368	Failed	500001	0.7873	67	6702	0.0196	151	15102	0.0244	104	10402	0.0149
	5	10^{-6}	85	8502	0.0113	Failed	500001	0.7033	178	17802	0.0302	224	22402	0.0358	778	77802	0.1169
	10	10^{-6}	509	50902	0.1286	Failed	500001	1.0247	69	6902	0.0232	156	15602	0.0326	104	10402	0.0202
	100	10^{-6}	Failed	20001	0.3212	Failed	20001	0.2462	67	6702	0.1393	151	15102	0.2010	104	10402	0.1266
	1000	10^{-3}	179	17902	2.5359	Failed	20001	2.5217	69	6902	1.4614	165	16502	2.2148	104	10402	1.2666
	1500	10^{-1}	Failed	20001	10.5158	200	20001	8.3525	69	6902	4.8984	104	10402	4.2498	162	16202	7.1407
	1500	10^{-1}	24	2402	0.46	23	2302	1.40	51	5102	1.94	127	12702	2.58	59	5902	1.04
	5000	10^{-1}	24	2402	2.18	29	2902	7.98	51	5102	10.44	128	12802	13.53	59	5902	7.13
Extended DENSCHNF	2	10^{-6}	19	1902	0.0014	35	3502	0.0022	23	2302	0.0031	214	21402	0.0203	30	3002	0.0029
	5	10^{-6}	19	1902	0.0013	37	3702	0.0025	23	2302	0.0033	220	22002	0.0222	32	3202	0.0025
	10	10^{-6}	22	2202	0.0024	37	3702	0.0042	23	2302	0.0056	232	23202	0.0426	32	3202	0.0039
	100	10^{-6}	16	1602	0.0125	35	3502	0.0306	21	2102	0.0169	Failed	20001	0.2570	30	3002	0.0263
	1000	10^{-3}	22	2202	0.2045	37	3702	0.3557	23	2302	0.4286	Failed	20001	2.7050	32	3202	0.2829
	1500	10^{-1}	10	1002	0.12	12	1202	0.86	9	902	0.12	67	6702	1.30	8	802	0.11
	5000	10^{-1}	11	1102	0.65	12	1202	4.83	9	902	0.64	75	7502	7.36	8	802	0.55
Extended Tridiagonal	2	10^{-6}	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001
	5	10^{-6}	53	5302	0.0074	121	12102	0.0258	225	22502	0.0320	39	3902	0.0050	121	12102	0.0171
	10	10^{-6}	1288	128802	0.6668	1638	163802	0.6375	121	12102	0.0405	100	10002	0.0330	217	21702	0.0757
	1500	10^{-1}	123	12302	1346.25	37	3702	1535.48	71	7102	610.83	88	8802	723.28	35	3502	317.18
Extended Himmelblau	2	10^{-6}	28	2802	0.0032	21	2102	0.0024	32	3202	0.0049	62	6202	0.0068	18	1802	0.0023
	5	10^{-6}	29	2902	0.0032	22	2202	0.0026	32	3202	0.0049	62	6202	0.0070	18	1802	0.0022
	10	10^{-6}	31	3102	0.0055	22	2202	0.0042	32	3202	0.0080	62	6202	0.0113	20	2002	0.0039
	100	10^{-6}	28	2802	0.0356	21	2102	0.0292	30	3002	0.0546	60	6002	0.0825	18	1802	0.0255
	1000	10^{-3}	31	3102	0.5807	22	2202	0.4318	32	3202	0.8910	20	2002	0.4193	62	6202	1.2571
	1500	10^{-1}	16	1602	0.25	14	1402	0.43	26	2602	0.59	27	2702	0.43	11	1102	0.18
DBVF	2	10^{-6}	27	2702	0.0068	66	6602	0.0190	38	3802	0.0152	38	3802	0.0101	62	6202	0.0162
	5	10^{-6}	304	30402	0.1744	1583	158302	0.8913	116	11602	0.0748	184	18402	0.0966	189	18902	0.0975
	10	10^{-6}	812	81202	0.9963	Failed	500001	5.8602	480	48002	0.5284	705	70502	0.8860	679	67902	0.7837
BRYBND	2	10^{-6}	12	1202	0.0020	15	1502	0.0025	28	2802	0.0097	8	802	0.0014	15	1502	0.0029
	5	10^{-6}	Failed	500001	3.0214	118	11802	0.0331	209	20902	0.0653	2884	288402	1.5261	118	11802	0.0346
	10	10^{-6}	Failed	500001	8.0405	124	12402	0.0849	173	17302	0.1351	Failed	500001	7.7655	119	11902	0.0866
	100	10^{-6}	Failed	20001	14.5154	120	12002	4.4273	125	12502	5.0962	Failed	20001	14.7362	103	10302	3.9309
PerturbedQuad	2	10^{-6}	28	2802	0.0014	4	402	0.0002	18	1802	0.0012	27	2702	0.0013	4	402	0.0003
	5	10^{-6}	38	3802	0.0026	30	3002	0.0018	64	6402	0.0052	35	3502	0.0038	30	3002	0.0059
	10	10^{-6}	40	4002	0.0051	56	5602	0.0088	66	6602	0.0115	31	3102	0.0032	35	3502	0.0044
	100	10^{-6}	Failed	20001	0.2613	Failed	20001	0.2246	117	11702	0.1357	92	9202	0.0931	135	13502	0.1482
TRIDIA	2	10^{-6}	Failed	500001	1.2554	39	3902	0.0035	25	2502	0.0048	164	16402	0.0171	29	2902	0.0046
	5	10^{-6}	312	31202	0.0865	111	11102	0.0245	57	5702	0.0124	256	25602	0.0715	52	5202	0.0113
	10	10^{-6}	1447	144702	1.5322	358	35802	0.2010	79	7902	0.0465	2006	200602	1.8281	85	8502	0.0441
	100	10^{-6}	Failed	20001	12.2863	59	5902	2.1308	69	6902	2.6164	Failed	20001	11.5398	40	4002	1.4651
ExtendedPenalty	2	10^{-6}	1	102	0.0002	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001
	100	10^{-6}	2	202	0.0010	2	202	0.0010	3	302	0.0146	2	202	0.0009	2	202	0.0010
	1000	10^{-3}	9	902	0.0800	5	502	0.0329	9	902	1.0982	5	502	0.0493	10	1002	0.0704

Table 3. Comparison of: FR vs PRP vs HS vs DY vs RN Methods on 30 Test Functions

Function	Dim	ϵ	FR			PRP			HS			DY			RN		
			NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU
BALF	2	10^{-6}	9	902	0.0006	10	1002	0.0007	18	1802	0.0057	23	2302	0.0012	10	1002	0.0007
	5	10^{-6}	35	3502	0.0031	12	1202	0.0013	23	2302	0.0022	18	1802	0.0016	12	1202	0.0012
	10	10^{-6}	9	902	0.0012	11	1102	0.0018	20	2002	0.0150	24	2402	0.0028	11	1102	0.0020
	100	10^{-6}	9	902	0.0080	10	1002	0.0095	18	1802	0.1193	22	2202	0.0171	10	1002	0.0095
	1000	10^{-3}	9	902	0.1901	11	1102	0.1993	20	2002	3.4014	11	1102	0.2447	24	2402	0.4223
Diagonal1	2	10^{-6}	33	3302	0.0011	2	202	0.0001	19	1902	0.0008	18	1802	0.0006	2	202	0.0001
	5	10^{-6}	54	5402	0.0032	32	3202	0.0016	68	6802	0.0035	35	3502	0.0017	32	3202	0.0019
	10	10^{-6}	113	11302	0.0107	60	6002	0.0051	70	7002	0.0056	44	4402	0.0033	34	3402	0.0028
	100	10^{-6}	Failed	20001	0.1500	Failed	20001	0.1274	100	10002	0.0642	99	9902	0.0604	125	12502	0.0800
Diagonal2	2	10^{-6}	18	1802	0.0010	8	802	0.0006	21	2102	0.0035	14	1402	0.0008	8	802	0.0007
	5	10^{-6}	78	7802	0.0119	31	3102	0.0042	33	3302	0.0051	24	2402	0.0031	31	3102	0.0044
	10	10^{-6}	60	6002	0.0182	56	5602	0.0152	66	6602	0.0185	32	3202	0.0086	37	3702	0.0183
	100	10^{-6}	152	15202	0.7901	Failed	20001	0.7612	118	11802	0.4500	121	12102	0.6386	144	14402	0.5248
Diagonal4	2	10^{-6}	66	6602	0.0058	2	202	0.0001	80	8002	0.0181	64	6402	0.0093	2	202	0.0002
	5	10^{-6}	71	7102	0.0074	81	8102	0.0151	81	8102	0.0198	68	6802	0.0080	81	8102	0.0155
	10	10^{-6}	73	7302	0.0103	86	8602	0.0250	86	8602	0.0213	68	6802	0.0111	86	8602	0.0214
	100	10^{-6}	63	6302	0.0283	76	7602	0.0676	73	7302	0.0655	57	5702	0.0251	76	7602	0.0700
	1000	10^{-3}	69	6902	0.7038	84	8402	1.6597	75	7502	1.5925	82	8202	1.6060	66	6602	0.7537
ExtendedDiag	100	10^{-6}	64	6402	0.0748	72	7202	0.1885	74	7402	0.1899	62	6202	0.0748	72	7202	0.1893
	1000	10^{-3}	74	7402	2.2958	82	8202	5.2970	84	8402	5.0847	82	8202	5.3388	72	7202	2.3409
Beale	2	10^{-6}	55	5502	0.0057	708	70802	0.0588	63	6302	0.0131	54	5402	0.0043	117	11702	0.0104
	1500	10^{-1}	9	902	0.0027	21	2102	0.0137	25	2502	0.0129	9	902	0.0011	7	702	0.0012
Booth	2	10^{-6}	44	4402	0.0030	39	3902	0.0019	17	1702	0.0008	29	2902	0.0021	25	2502	0.0018
	1500	10^{-1}	7	702	0.0009	6	602	0.0007	6	602	0.0007	10	1002	0.001	10	1002	0.0011
Ackley	2	10^{-6}	18	1802	0.0416	25	2502	0.0297	45	4502	0.0765	12	1202	0.0079	25	2502	0.0208
	5	10^{-6}	8	802	0.0032	11	1102	0.0054	21	2102	0.0411	8	802	0.0036	11	1102	0.0033
	10	10^{-6}	8	802	0.0024	11	1102	0.0034	21	2102	0.0564	7	702	0.0041	11	1102	0.0035
Rastrigin	2	10^{-6}	1	102	0.0002	1	102	0.0001	1	102	0.0001	1	102	0.0001	1	102	0.0001
Griewank	2	10^{-6}	602	60202	0.1511	25	2502	0.0027	26	2602	0.0032	Failed	Failed	Failed	26	2602	0.0033
	5	10^{-6}	830	83002	0.2685	52	5202	0.0073	50	5002	0.0080	696	69602	0.1685	43	4302	0.0065
	10	10^{-6}	Failed	Failed	Failed	125	12502	0.0483	124	12402	0.0265	Failed	Failed	Failed	127	12702	0.0368
Matyas	2	10^{-6}	52	5202	0.0014	87	8702	0.0071	87	8702	0.0102	42	4202	0.0018	87	8702	0.0126
	5	10^{-6}	52	5202	0.0021	87	8702	0.0106	87	8702	0.0075	42	4202	0.0010	87	8702	0.0075
Schwefel	2	10^{-6}	24	2402	0.0023	55	5502	0.0039	55	5502	0.0052	14	1402	0.0008	55	5502	0.0056
	5	10^{-6}	25	2502	0.0026	57	5702	0.0047	57	5702	0.0049	14	1402	0.0016	57	5702	0.0051
	10	10^{-6}	25	2502	0.0039	58	5802	0.0078	58	5802	0.0079	14	1402	0.0020	58	5802	0.0085
	1500	10^{-1}	12	1202	0.19	27	2702	0.41	27	2702	0.41	7	702	0.11	27	2702	0.43

Table 4. Performance Comparison: BADY vs BAFR vs RN Methods (30 Test Functions)

Function	Dim	ϵ	BADY			BAFR			RN		
			NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU
Rosenbrock	2	10^{-6}	Failed	500001	2.81	434	43402	0.12	104	10402	0.02
	5	10^{-6}	Failed	500001	3.83	218	21802	0.09	187	18702	0.08
	10	10^{-6}	Failed	500001	3.91	899	89902	0.35	180	18002	0.06
Sphere	2	10^{-6}	1	102	0.0001	1	102	0.0001	1	102	0.0001
SumSquares	2	10^{-6}	34	3402	0.001	33	3302	0.002	2	202	0.0001
	5	10^{-6}	254	25402	0.02	54	5402	0.003	32	3202	0.003
	10	10^{-6}	99	9902	0.009	113	11302	0.01	34	3402	0.003
	100	10^{-6}	251	25102	0.16	255	25502	0.19	155	15502	0.11
	1500	10^{-1}	311	31102	9.99	217	21702	6.99	229	22902	7.28
	2000	10^{-1}	418	41802	18.7606	426	42602	19.4882	381	38102	17.9099
	10000	10^{-1}	603	60302	174.94	623	62302	185.14	530	53002	147.20
Zakharov	2	10^{-6}	1071	107102	0.35	32	3202	0.004	10	1002	0.002
	5	10^{-6}	Failed	500001	2.60	38	3802	0.009	90	9002	0.02
	10	10^{-6}	63	6302	0.02	69	6902	0.02	154	15402	0.04
	100	10^{-6}	Failed	500001	5.94	145	14502	0.09	163	16302	0.12
	2000	10^{-1}	257	25702	1.0337	203	20302	0.8637	520	52002	2.0222
	10000	10^{-1}	170	17002	2.20	460	46002	5.41	574	57402	9.41
Dixon-Price	2	10^{-6}	Failed	500001	1.77	91	9102	0.01	30	3002	0.004
	5	10^{-6}	4560	456002	1.88	1027	102702	0.32	53	5302	0.01
	10	10^{-6}	617	61702	0.21	134	13402	0.03	81	8102	0.03
	100	10^{-6}	Failed	500001	25.58	Failed	500001	19.47	1413	141302	2.89
	1500	10^{-1}	743	74302	81.50	424	42402	46.38	539	53902	58.86
Quadratic-QF1	2	10^{-6}	10	1002	0.0008	10	1002	0.001	9	902	0.0007
	5	10^{-6}	12	1202	0.001	11	1102	0.001	21	2102	0.002
	10	10^{-6}	29	2902	0.004	28	2802	0.004	13	1302	0.003
	100	10^{-6}	67	6702	0.14	77	7702	0.15	71	7102	0.14
	1500	10^{-1}	27	2702	1.19	27	2702	1.16	27	2702	1.16
	1500	10^{-1}	62	6202	7.66	60	6002	7.46	59	5902	7.19
Raydan1	2	10^{-6}	380	38002	0.05	28	2802	0.001	8	802	0.0004
	5	10^{-6}	145	14502	0.03	92	9202	0.01	27	2702	0.004
	10	10^{-6}	35	3502	0.006	38	3802	0.006	30	3002	0.005
	100	10^{-6}	Failed	500001	151.27	Failed	500001	141.48	191	19102	1.64
	1500	10^{-1}	203	20302	27.45	183	18302	24.82	154	15402	20.31
Raydan2	2	10^{-6}	1	102	0.0001	1	102	0.0002	1	102	0.0002
Extended Rosenbrock	2	10^{-6}	Failed	500001	1.81	434	43402	0.06	104	10402	0.02
	5	10^{-6}	Failed	500001	2.66	85	8502	0.02	778	77802	0.14
	10	10^{-6}	Failed	500001	3.12	493	49302	0.15	104	10402	0.02
	100	10^{-6}	Failed	500001	22.13	550	55002	0.98	106	10602	0.13
	1500	10^{-1}	74	7402	3.50	122	12202	5.79	59	5902	2.74
Extended DENSCHNF	2	10^{-6}	25	2502	0.002	19	1902	0.002	30	3002	0.003
	5	10^{-6}	25	2502	0.002	19	1902	0.001	32	3202	0.002
	10	10^{-6}	26	2602	0.003	22	2202	0.003	32	3202	0.004
	100	10^{-6}	28	2802	0.03	33	3302	0.03	36	3602	0.04
	1500	10^{-1}	8	802	0.28	8	802	0.29	8	802	0.29
Extended Tridiagonal	2	10^{-6}	1	102	0.0002	1	102	0.0002	1	102	0.0001
	5	10^{-6}	30	3002	0.004	53	5302	0.008	121	12102	0.02
	10	10^{-6}	Failed	500001	4.88	1256	125602	0.67	217	21702	0.08
	100	10^{-6}	Failed	500001	342.92	2920	292002	123.67	2891	289102	85.87
	1500	10^{-1}	33	3302	3455.82	31	3102	593.45	35	3502	638.09
Extended Himmelblau	2	10^{-6}	34	3402	0.003	28	2802	0.003	18	1802	0.002
	5	10^{-6}	34	3402	0.01	29	2902	0.005	18	1802	0.004
	10	10^{-6}	35	3502	0.007	31	3102	0.006	20	2002	0.004
	100	10^{-6}	37	3702	0.07	32	3202	0.06	21	2102	0.04
	1500	10^{-1}	11	1102	0.61	11	1102	0.61	11	1102	0.60
DBVF	2	10^{-6}	40	4002	0.008	27	2702	0.007	62	6202	0.01
	5	10^{-6}	Failed	500001	4.50	310	31002	0.24	189	18902	0.12
	10	10^{-6}	Failed	500001	8.03	807	80702	1.06	679	67902	0.84

Table 5. Performance Comparison: BADY vs BAFR vs RN Methods (30 Test Functions)

Function	Dim	ϵ	BADY			BAFR			RN		
			NOI	NFEV	CPU	NOI	NFEV	CPU	NOI	NFEV	CPU
BRYBND	2	10^{-6}	11	1102	0.001	12	1202	0.002	15	1502	0.002
	5	10^{-6}	Failed	500001	7.18	Failed	500001	3.44	118	11802	0.04
	10	10^{-6}	Failed	500001	14.54	Failed	500001	8.09	119	11902	0.08
	100	10^{-6}	Failed	500001	741.24	Failed	500001	511.04	124	12402	5.91
	1500	10^{-1}	43	4302	1021.13	33	3302	742.60	47	4702	1116.57
PerturbedQuad	2	10^{-6}	43	4302	0.002	28	2802	0.001	4	402	0.0004
	5	10^{-6}	386	38602	0.06	38	3802	0.004	30	3002	0.002
	10	10^{-6}	48	4802	0.006	40	4002	0.004	35	3502	0.004
	100	10^{-6}	244	24402	0.41	248	24802	0.38	156	15602	0.17
TRIDIA	2	10^{-6}	36	3602	0.004	Failed	500001	1.07	29	2902	0.003
	5	10^{-6}	Failed	500001	5.27	314	31402	0.10	52	5202	0.02
	10	10^{-6}	Failed	500001	10.79	1470	147002	1.51	85	8502	0.05
	100	10^{-6}	Failed	500001	685.82	1374	137402	100.42	44	4402	2.42
ExtendedPenalty	10	10^{-6}	1	102	0.0001	1	102	0.0002	1	102	0.0001
	100	10^{-6}	2	202	0.001	2	202	0.002	3	302	0.002
BALF	5	10^{-6}	32	3202	0.004	35	3502	0.004	12	1202	0.002
	10	10^{-6}	9	902	0.001	9	902	0.001	11	1102	0.002
	100	10^{-6}	9	902	0.02	10	1002	0.01	12	1202	0.02
Diagonal1	2	10^{-6}	34	3402	0.001	33	3302	0.001	2	202	0.0001
	5	10^{-6}	254	25402	0.03	54	5402	0.006	32	3202	0.003
	10	10^{-6}	99	9902	0.01	113	11302	0.01	34	3402	0.003
	100	10^{-6}	251	25102	0.24	255	25502	0.30	133	13302	0.12
Diagonal2	2	10^{-6}	16	1602	0.001	18	1802	0.001	8	802	0.0007
	5	10^{-6}	44	4402	0.006	78	7802	0.02	31	3102	0.004
	10	10^{-6}	Failed	500001	25.79	60	6002	0.02	37	3702	0.01
	100	10^{-6}	Failed	500001	350.31	Failed	500001	324.99	205	20502	3.71
Diagonal4	2	10^{-6}	67	6702	0.006	66	6602	0.006	2	202	0.0001
	5	10^{-6}	70	7002	0.01	71	7102	0.008	81	8102	0.02
	10	10^{-6}	72	7202	0.01	73	7302	0.01	86	8602	0.02
	100	10^{-6}	80	8002	0.10	83	8302	0.10	96	9602	0.21
ExtendedDiag	100	10^{-6}	83	8302	0.27	84	8402	0.31	92	9202	0.59
Beale	5	10^{-6}	Failed	500001	7.88	55	5502	0.01	117	11702	0.01
	1500	10^{-1}	13	1302	0.004	7	702	0.003	7	702	0.002
Booth	5	10^{-6}	23	2302	0.001	44	4402	0.002	25	2502	0.001
	1500	10^{-1}	10	1002	0.002	9	902	0.002	10	1002	0.002
	10000	10^{-1}	10	1002	0.008	9	902	0.005	10	1002	0.006
Ackley	2	10^{-6}	12	1202	0.01	18	1802	0.03	25	2502	0.02
	5	10^{-6}	8	802	0.003	8	802	0.003	11	1102	0.003
	10	10^{-6}	8	802	0.003	8	802	0.003	11	1102	0.004
Rastrigin	2	10^{-6}	1	102	0.0001	1	102	0.0001	1	102	0.0001
Griewank	2	10^{-6}	2275	227502	0.79	602	60202	0.14	26	2602	0.004
	5	10^{-6}	Failed	500001	3.45	830	83002	0.27	43	4302	0.007
	10	10^{-6}	Failed	500001	19.14	Failed	500001	5.18	127	12702	0.04
	1500	10^{-1}	6	602	4.21	6	602	3.98	6	602	4.06
	2000	10^{-1}	7	702	6.0804	7	702	6.1092	7	702	6.2357
	10000	10^{-1}	13	1302	57.82	13	1302	58.23	13	1302	58.02
Matyas	2	10^{-6}	51	5102	0.002	52	5202	0.002	87	8702	0.008
	5	10^{-6}	51	5102	0.002	52	5202	0.002	87	8702	0.01
Schwefel	5	10^{-6}	24	2402	0.002	25	2502	0.003	57	5702	0.006
	10	10^{-6}	25	2502	0.004	25	2502	0.004	58	5802	0.008
	1500	10^{-1}	27	2702	1.19	27	2702	1.16	27	2702	1.16

REFERENCES

1. A. Y. Al-Bayati and N. H. Al-Assady, *Conjugate gradient method.*, Tech. Research School of Comp. Studies, Leeds Univ., UK, 1986.
2. S. N. Alsuliman, N. F. Ibrahim, N. A. Hanum Aizam *A new modification of the Wei–Yao–Liu conjugate gradient method for unconstrained optimization and its application in robot control* Journal of Advanced Research in Applied Sciences and Engineering Technology, 314-327, 2024.
3. N. Andrei, *An unconstrained optimization test functions collection*, Adv. Model. Optim. **10** 147–161, 2008.
4. N. Andrei, *Nonlinear conjugate gradient methods for unconstrained optimization*, Springer Optimization and its Applications, Romania, 2020.
5. S. Babaie-Kafaki, N. Mirhoseini, and Z. Aminifard, *A Descent Extension of a Modified Polak-Ribiere-Polyak method with application in image restoration problem,* Optimization Letters, vol. 17, no. 2, pp. 351–367, 2023.
6. S. BenHanachi, B. Sellami, M. Belloufi, *Global convergence property with inexact line search for a new conjugate gradient method*, An Inter. Journal of Optimiz. and Control: Theories and Applic. **15**, 25–34, 2025.
7. I. Bongartz, A. R. Conn, N. Gould, P. L. Toint, *CUTE: constrained and unconstrained testing environments*, ACM Trans. Math. Soft. **21**, 123–160, 1995.
8. Y. H. Dai, Y. Yuan, *A nonlinear conjugate gradient method with a strong global convergenc property*, SIAM J.Opt. **10**,177-182,1999.
9. S. Delladji, M. Belloufi, B. Sellami, *New hybrid conjugate gradient method as a convex combination of FR and BA methods*, J. Inform. Optim. Sci. **42**, 591–602, 2021.
10. S. Djordjevic, *New hybrid conjugate gradient method as a convex combination of HS and FR*, Journal of Applied Mathematics and Computation, **9**, 366-378, 2018.
11. E. D. Dolan, J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program. **91**, 201-213, 2002.
12. R. Fletcher, *Practical Methods of Optimization. Unconstrained Optimization*, Wiley, New York, 1987.
13. R. Fletcher, C. M. Reeves, *Function minimization by conjugate gradients*, Comput. J. **7**, 149-154, 1964.
14. I. Guefassa, Y. Chaib and T. Bechouat *Another hybrid conjugate gradient method as a convex combination of WYL and CD methods*, Monte Carlo Methods and Applications, vol. 30, no. 3, pp. 225-234, 2024.
15. A. Hamdi, B. Sellami and M. Belloufi *A New hybrid conjugate gradient method as a convex combination of BA and DY methods* Commun. Optim. Theory, 2020.
16. B. A. Hassan and A. A. Saad, *Explaining new parameters conjugate analysis based on the quadratic model*, Journal of Interdisciplinary Mathematics, vol. 26, no. 6, pp. 1219–1229, 2023.
17. B. A. Hassan, I. A. R. Moghrabi, and I. M. Sulaiman, *New conjugate gradient image processing methods*, Asian-European Journal of Mathematics, vol. 16, no. 6, 2023.
18. B. A. Hassan and H. M. Sadiq, *Efficient New Conjugate Gradient Methods for Removing Impulse Noise Images,* European Journal of Pure and Applied Mathematics, vol. 15, no. 4, pp. 2011-2021, 2022.
19. M. R. Hestenes, E. Stiefel, *Methods of conjugate gradients for solving linear systems*, J. Res. Natl. Bur. Stand. **49**, 409–436, 1952.
20. H. Huang, S. Lin, *A modified Wei–Yao–Liu conjugate gradient method for unconstrained optimization*, Journal of Computational and Applied Mathematics, 2011.
21. X. Li *A hybrid Polak–Ribière–Polyak and Wei–Yao–Liu conjugate gradient method for unconstrained optimization* Journal of Inequalities and Applications, 2019.
22. J. K. Liu and S. J. Li, *New hybrid conjugate gradient method for unconstrained optimization.* Applied Mathematics and Computation, 245, 36-43, 2014.
23. Y. Liu, C. Storey, *Efficient generalized conjugate gradient algorithm. Part 1: Theory*, J. Optim. Theory Appl. **69**, 129-137, 1991.
24. W. Liu, Z. Wei, L. Liu *A new hybrid conjugate gradient method for multiobjective optimization* Abstract and Appl. Analysis, 2014
25. E. Polak, G. Ribiere, *Note sur la convergence des méthodes de directions conjuguées*, Rev. Fran. Infor. Rech. Oper. **16**, 35-43, 1969.
26. B. T. Polyak, *The conjugate gradient method in extreme problems*, U.S.S.R. Comput. Math. Phys. **9**, 94-112, 1969.
27. M. J. D. Powell, *Restart procedures of the conjugate gradient method*, Math. Program. **2**, 241–254, 1977.
28. Z. Wei, S. Yao, L. Liu *A new conjugate gradient method for unconstrained optimiz.* Applied Mathematics and Computation, 2006
29. P. Wolfe, *Convergence conditions for ascent methods*, SIAM Rev. **11**, 226-235, 1969.
30. P. Wolfe, *Convergence conditions for ascent methods. II: Some corrections*, SIAM Rev. **13**, 185-188, 1971.
31. S. Yao, B. Qin, *A hybrid Dai–Liao and Wei–Yao–Liu conjugate gradient method for unconstrained optimization*, Journal of Computational and Applied Mathematics, 2016.