Applied Mathematics and Computational Intelligence
Volume 15, No. 3, 2026 [85 – 107]

Non-Negative Generalized Parameter of the Hybrid Memoryless Three-Term Conjugate Gradient Method

Novelan Kumar Sivagumar[1*], Siti Mahani Marjugi[2], Muhammad Aqiil Iqmal Ishak[3]

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

* Corresponding author: novelankumar00@gmail.com

Received: 3 November 2025
Revised: 3 June 2026
Accepted: 8 June 2026

ABSTRACT

The Conjugate Gradient (CG) method is a well-established technique for solving optimization problems, particularly in large-scale applications. Despite its popularity, classical CG methods often face computational inefficiencies in high-dimensional tasks. This study addresses these limitations by developing the Hybrid Memoryless Three-Term Conjugate Gradient (HMTTCG) method, which introduces a non-negative generalized parameter to combine the strengths of several CG methods while reducing their weaknesses. Existing CG methods are classified into two groups: Fletcher-Reeves (FR), Dai-Yuan (DY), and Conjugate Descent (CD) in the β_1 group, and Liu-Storey (LS), Hestenes-Stiefel (HS), and Polak-Ribiere-Polyak (PRP) in the β_2 group. These methods often suffer inefficiencies due to constraints like the non-negativity of CG parameters, which can disrupt sufficient descent conditions and lead to reliance on the steepest descent direction, increasing computational costs. To overcome these challenges, the HMTTCG method proposes a hybrid parameter that maintains descent properties and enhances computational efficiency by selecting suitable CG parameters. The approach involves deriving the hybrid parameter mathematically and conducting numerical experiments across various optimization problems, comparing the HMTTCG method with existing the three-term CG methods in terms of the number of iterations (NOI) and CPU time. The method ensures descending and bounded search directions, guaranteeing global convergence under the Strong-Wolfe line search. Experimental results demonstrate that HMTTCG outperforms existing methods by solving problems with fewer iterations and computational time, proving its efficiency and robustness. In conclusion, the HMTTCG method represents a robust and efficient optimization strategy for large-scale unconstrained optimization problems.

Keywords: Unconstrained Optimization, Conjugate Gradient, Optimization Problems, Hybrid Memoryless Three-Term Conjugate Gradient, Non-Negative Generalized Parameter, Global Convergence Analysis, Strong-Wolfe Line Search Conditions.

1. INTRODUCTION

Unconstrained optimisation identifies optimal solutions without constraints on decision variables, aiming to maximize or minimize an objective function. These problems arise in engineering, physics, economics, and computer science, playing a vital role in real-world applications. Efficient solutions are crucial across industries, leading to the development of powerful optimization algorithms over the decades. For instance, machine learning frequently employs unconstrained optimization to minimize loss functions during model training.

The Conjugate Gradient (CG) method is a well-known and very efficient way to deal with a large-scale problems. The CG technique, which was first presented by [1] and [2], is still a popular optimisation algorithm.The following is the standard form of an unconstrained optimisation problem.

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

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. The CG method updates the iterations according to,

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

where $k = 0, 1, 2, \ldots$, and the search direction d_k is defined as,

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \geq 1. \end{cases} \tag{3}$$

Since the two-term search direction causes slower convergence and potential instability due to limited consideration of past information, the three-term search direction is implemented to enhance the convergence rates and provide greater stability compared to [3]. The Three-Term Conjugate Gradient (TTCG) search direction is defined as follows,

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1} + \gamma_k d_{k-2}, & \text{if } k \geq 1, \end{cases} \tag{4}$$

where, g_k represents the gradient of the objective function at iteration k, while β_k and γ_k are parameters designed to ensure the conjugacy of directions. For $k = 0$, the algorithm starts in the steepest descent direction $(-g_k)$. For $k \geq 1$, the formula incorporates previous directions (d_{k-1}, d_{k-2}) to improve efficiency and maintain the conjugacy property.

The step length α_k is obtained using the strong Wolfe line search conditions, which were first proposed by Wolfe and later refined by Powell [4, 5], as follows:

$$f(x_k + \alpha_k d_k) \leq f(x_k) + \delta \alpha_k g_k^T d_k, \tag{5}$$

$$|g(x_k + \alpha_k d_k)^T d_k| \geq \sigma |g_k^T d_k|. \tag{6}$$

Here, d_k is the search direction, g_k is the gradient (residual) vector, x_k is the current location at iteration k, and α_k is the step length that controls how far the procedure advances in the search direction. The conjugate gradient (CG) coefficient that is utilized to update the search direction is denoted by the parameter β_k. Furthermore, δ $(0 < \delta < 0.5)$ is a constant that guarantees the function value drops sufficiently, and σ $(\delta < \sigma < 1)$ is a constant that regulates the line search's curvature condition.

Additionally, the sufficient descent property by,

$$g_k^{\mathrm{T}} d_k \leq -c\|g_k\|^2, \tag{7}$$

where $c > 0$. The constant c plays a crucial role in ensuring the global convergence of the nonlinear CG method.

The parameter β_k is fundamental in conjugate gradient (CG) methods since it determines the convergence behavior and efficiency of the algorithm.Several classical formulas for β_k have been introduced in the literature, including Hestenes–Stiefel (HS) [2], Polak–Ribiere–Polyak (PRP) [6, 7], Liu–Storey (LS) [8], Dai–Yuan (DY) [9], Fletcher–Reeves (FR) [10], and Conjugate Descent (CD) [11]. Each method proposes a different expression for β_k, which explains the differences in their theoretical properties and numerical performance.

Table 1 : Formulas for the coefficient β_k in the conjugate gradient method

Formula	Description	Notation	Year
β_k^{HS}	Hestenes-Stiefel	$\frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}$	1952
β_k^{FR}	Fletcher-Reeves	$\frac{\|\|g_k\|\|^2}{\|\|g_{k-1}\|\|^2}$	1964
β_k^{PRP}	Polak-Ribiére-Polyak	$\frac{g_k^T y_{k-1}}{\|\|g_{k-1}^T\|\|^2}$	1969
β_k^{LS}	Liu-Storey	$\frac{g_k^T y_{k-1}}{-g_{k-1}^T d_{k-1}}$	1991
β_k^{DY}	Dai-Yuan	$\frac{\|\|g_k^2\|\|}{d_{k-1}^T y_{k-1}}$	1999
β_k^{CD}	Conjugate Descent	$\frac{\|\|g_k\|\|^2}{-g_{k-1}^T d_{k-1}}$	2006

Here, $y_{k-1} = g_k - g_{k-1}$ denotes the difference between two consecutive gradient (residual) vectors. In addition, r_k denotes the residual (gradient) vector at iteration k, and r_k^T represents its transpose, which is used in inner product terms in the formulas.

According to Babaie-Kafaki and Ghanbari [12], optimisation techniques that make use of a common term such as $r_k^T y_{k-1}$ frequently yield positive results in real-world applications. Here, r_k denotes the residual (gradient) vector at iteration k, and r_k^T represents its transpose, which is used in inner product calculations. The term $r_k^T y_{k-1}$ therefore refers to the inner product between the current residual vector r_k and the change in residuals $y_{k-1} = r_k - r_{k-1}$. However, some researchers argue that these methods do not always lead to consistent improvements because of an issue known as jamming, and also because they behave differently compared to methods that use $\|r_k\|^2$. Here, $\|r_k\|^2$ refers to the squared length (norm) of the residual vector, which measures the size of the gradient at step k. To address these challenges, Babaie-Kafaki and Mahdavi-Amiri [13] emphasized the need to improve the effectiveness of these methods, which motivated researchers to combine ideas from both approaches to reduce their weaknesses. From a theoretical point of view, Hager and Zhang [14] explained that methods using $\|r_k\|^2$ only need the Lipschitz continuity assumption to guarantee global convergence, making them simpler compared to other update rules that require additional boundedness conditions. Powell [5] identified jamming as the main reason why the Fletcher-Reeves (FR) [11] method does not always perform well in practice. Babaie-Kafaki further explained that if a poor step occurs between two iterations, the resulting search direction d_k and step size α_k may not be effective, but this issue can often be fixed using a gradient restart technique. Interestingly, Babaie-Kafaki also pointed out that methods with a common term already have a built-in feature similar to an approximate restart, which helps reduce the jamming problem. Andrei [15, 16] also noted that when a poor step occurs, the computed search direction d_k tends to move closer to the steepest descent direction $-r_k$, since the gradient difference y_{k-1} becomes very small, leading to a small value of β_k.

Over the years, academics have worked to enhance the traditional two-term conjugate gradient (CG) methods by developing hybrid and three-term approaches. Based on Andrei [17] created a variant of the BFGS formula to ensure that the search direction meets the requirements for conjugacy and descent. The hybrid CG approach presented by Liu and Li [18] combines the LS and DY formulas through a convex combination, allowing the method to follow the Newton direction and the Dai–Liao (DL) condition while guaranteeing

convergence with a powerful Wolfe line search.According to Min Li [19], a three-term PRP scheme that functions similarly to the memoryless BFGS technique but reduces to the traditional PRP method under exact line search is still able to maintain the descent property under various methods. Later, Min Li [20] introduced another nonlinear CG technique that was shown to converge for both convex and nonconvex functions and produces directions akin to the memoryless BFGS approach. Abubakar [21] suggested a hybrid three-term strategy that uses the limited-memory BFGS algorithm to compute the search direction, improving convergence and outperforming certain earlier methods. Kumam [22] and Deepho [23] have made more recent contributions. They employed scaled preconditioners to improve performance and combined formulas like HS with LS or CD with DY. To get greater efficiency, other research has also experimented with various combinations of CG parameters.

Li et al. [24] presented an enhanced three-term CG technique in 2024 that can function without the typical Lipschitz condition and was effectively used in image processing. A new hybrid three-term CG approach based on a scaled memoryless BFGS update was proposed by Ishak and Marjugi [25] in 2025. This method demonstrated high efficiency in tackling unconstrained optimisation issues.

The structure of this paper is as follows: Section 2 presents the proposed modifications, Section 3 discusses the convergence analysis, Section 4 provides the results of numerical experiments, and Section 5 concludes the study.

2. MODIFICATIONS

Recalling the memoryless BFGS Method proposed by [26] and [27], the direction can be written as,

$$d_k^{\text{BFGS}} = -Q_k g_k, \tag{8}$$

$$Q_k = \left(I - \frac{s_{k-1}y_{k-1}^T}{s_{k-1}^T y_{k-1}} - \frac{y_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}} + \frac{s_{k-1}y_{k-1}^T - y_{k-1}^T s_{k-1}}{(s_{k-1}^T y_{k-1})^2} + \frac{s_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}} \right), \tag{9}$$

where $s_{k-1} = x_k - x_{k-1} = \alpha_{k-1} d_{k-1},$ and I is the identity matrix.

$$d_k^{\text{BFGS}} = -g_k + \left(\beta_k^{HS} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} \right) d_{k-1} + \frac{g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}} (y_{k-1} - s_{k-1}). \tag{10}$$

where the gradient at iteration k is denoted by g_k, and the term $-g_k$ represents the steepest descent direction. $\beta_k^{\text{HS}} = \frac{g_k^T y_{k-1}}{}$ for the parameter $d_{k-1}^T y_{k-1}$ is the Hestenes-Stiefel coefficient, which modifies the direction.The second term, $-\frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{(d_{k-1}^T y_{k-1})^2} d_{k-1}$, adjusts the old direction using gradient and curvature information so the new direction remains conjugate. The last part, $\frac{g_k^T d_{k-1}}{d_{k-1}^T y_{k-1}}(y_{k-1} - s_{k-1})$, provides a correction by balancing the gradient change and step change. Together, these adjustments make the search direction more stable and improve convergence.

Based on the above [20], [22] and [28] proposed variant of memoryless Three-Term (MTT) according to the HS,LS and PRP CG method with direction defined as,

$$d_k^{\text{MTT1}} = -g_k + \beta_k^1 d_{k-1} + t_k \frac{g_k^T d_{k-1}}{u_k} y_{k-1}, \tag{11}$$

where,

$$\beta_k^1 = \frac{g_k^T y_{k-1}}{u_k} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{u_k}, \tag{12}$$

and

$$u_k = \begin{cases} d_{k-1}^T y_{k-1}, & \text{HS}, \\ -d_{k-1}^T g_{k-1}, & \text{LS}, \\ \|g_{k-1}\|^2, & \text{PRP}. \end{cases} \tag{13}$$

We define Three-Term search direction based on CD,DY and FR as,

$$d_k^{\text{MTT2}} = -g_k + \beta_k^2 d_{k-1} + t_k \frac{g_k^T d_{k-1}}{u_k} g_k, \tag{14}$$

$$\beta_k^2 = \frac{g_k^T g_k}{v_k} - \frac{\|g_k\|^2 g_k^T d_{k-1}}{v_k}, \tag{15}$$

and

$$v_k = \begin{cases} d_{k-1}^T y_{k-1}, & \text{CD}, \\ -d_{k-1}^T g_{k-1}, & \text{DY}, \\ \|g_{k-1}\|^2, & \text{FR}. \end{cases} \tag{16}$$

Therefore,

$$d^{\text{H}} = -g_k + \beta_k^H d_{k-1} + \lambda_k c_k \quad k > 1, \tag{17}$$

$$\beta_k^{\text{GENERALIZED}} = \frac{g_k^T C_k}{w_k} - \frac{\|C_k\|^2 g_k^T d_{k-1}}{w_k^2}, \tag{18}$$

where,

$$\beta_k^{\text{H}} = \max\{0, \min\{\beta_k^1, \beta_k^2\}\}, \tag{19}$$

$$C_k = \begin{cases} y_{k-1}, & \text{if } \beta_k^H = \beta_k^1, \\ g_k, & \text{otherwise.} \end{cases} \tag{20}$$

Here, C_k is g_k and y_{k-1},λ_k is $t_k \frac{g_k^\top d_{k-1}}{w_k}$ and D_1, D_2, D_3 is the denominator of the desired hybrid CG parameter.

The parameter t is computed through the solution of a scalar minimization problem :

$\min \|(y_{k-1} - s_{k-1}) - tc_k\|^2$.

Let $E_k = (y_{k-1} - s_{k-1}) - tc_k$, then

$$\begin{aligned} E_k E_k^{\mathrm{T}} &= \big[(y_{k-1} - s_{k-1}) - tc_k\big]\big[(y_{k-1} - s_{k-1}) - tc_k\big]^T \\ &= \big[(y_{k-1} - s_{k-1}) - tc_k\big]\big[(y_{k-1} - s_{k-1})^T - tc_k^T\big] \\ &= t^2 c_k c_k^T - t(y_{k-1} - s_{k-1})c_k^T - c_k(y_{k-1} - s_{k-1})^T \\ &\quad + (y_{k-1} - s_{k-1})(y_{k-1} - s_{k-1})^T, \\ \mathrm{tr}(E_k E_k^{\mathrm{T}}) &= t^2\|c_k\|^2 - t\big[\mathrm{tr}(E_k c_k^T) + \mathrm{tr}(c_k E_k^T)\big] + \|E_k\|^2 \\ &= t^2\|c_k\|^2 - 2tc_k^T E_k + \|E_k\|^2, \\ 2t\|c_k\|^2 - 2c_k^T E_k &= 0 \\ t &= \frac{c_k^T\big(y_{k-1} - s_{k-1}\big)}{\|c_k\|^2} \\ t &= \frac{c_k^T\big(y_{k-1} - s_{k-1}\big)}{\|c_k\|^2} \\ t &= \min\Big\{\bar{t}, \max\big\{0, \frac{c_k^T\big(y_{k-1} - s_{k-1}\big)}{\|c_k\|^2}\big\}\Big\}. \end{aligned}$$

which implies $0 \le t_k \le \bar{t} \le 1$.

Motivated by the three-term CG form, the equation we propose a hybrid three-term,

$$d_0 = -g_0,\ d_k^{\mathbf{H}} = -g_k + \beta_k^H d_{k-1} + \lambda_k c_k, \tag{21}$$

where

$$\beta_k^{\text{GENERALIZED}} = \frac{g_k^\top c_k}{w_k} - \frac{\|c_k\|^2 g_k^\top d_{k-1}}{w_k^2}, \tag{22}$$

$$w_k = \max\left(\mu\|d_{k-1}\|\|c_k\|, D_1, D_2, D_3\right), \tag{23}$$

where, c_k is g_k and y_{k-1}, λ_k is $t_k \frac{g_k^\top d_{k-1}}{w_k}$ and D_1, D_2, D_3 is the denominator of the desired hybrid CG parameter.

$$\beta_k^{\mathrm{H}} = \max\{0, \min\{\beta_k^1, \beta_k^2\}\}, \tag{24}$$

where

$$\beta_k^1 = \frac{g_k^T y_{k-1}}{u_k} - \frac{\|y_{k-1}\|^2 g_k^T d_{k-1}}{u_k}, \quad \beta_k^2 = \frac{g_k^T g_k}{v_k} - \frac{\|g_k\|^2 g_k^T d_{k-1}}{v_k} \tag{25}$$

The generalized Memoryless Hybrid Three-Term Conjugate Gradient (MTTCG) method can be divided into two groups. Group 1 (β_k^1) focuses on convergence and includes methods such as FR, DY, and CD. Group 2 (β_k^2) focuses on efficiency and includes methods such as HS, LS, and PRP. The hybrid approach refers to the hybridization between Equation (25), combining elements from both groups to achieve a balance between efficiency and robustness.

2.1. Algorithm 1

The proposed parameters in this study are implemented using the steps outlined in Algorithm 1, as shown below.

Algorithm 1 Hybrid Memoryless Three-Term Conjugate Gradient Method (HMTTCG)

Step 1 Given constants $0 < \delta < \sigma < 1,\ \mu \geq 0,\ \varepsilon > 0$. Choose an initial point $x_0 \in \mathbb{R}^n$ and set $k > 0$.

Step 2 If $\|g_k\| \leq \varepsilon$, Where $\varepsilon = 10^{-6}$ then the algrorithm stops. Otherwise, continue to Step 3.

Step 3 Compute the CG parameters $\beta_k^{\text{GENERALIZED}}, w_k, c_k, \beta_k^H, \beta_k^1, \beta_k^2$.

$$\beta_k^{\text{GENERALIZED}} = \frac{g_k^\top c_k}{w_k} - \frac{\|c_k\|^2 g_k^\top d_{k-1}}{w_k^2}$$

$$w_k = \max\left(\mu\|d_{k-1}\|,\ \|c_k\|,\ d_{k-1}^\top y_{k-1},\ -g_{k-1}^\top d_{k-1},\ \|g_{k-1}\|^2\right)$$

$$c_k = \begin{cases} y_{k-1}, & \text{if } \beta_k^H = \beta_k^1, \\ g_k, & \text{otherwise.} \end{cases}$$

$$\beta_k^H = \max\{0, \min(\beta_k^1, \beta_k^2)\}$$

$$\beta_k^1 = \frac{g_k^\top y_{k-1}}{u_k} - \frac{\|y_{k-1}\|^2 g_k^\top d_{k-1}}{u_k}, \qquad \beta_k^2 = \frac{g_k^\top g_k}{v_k} - \frac{\|g_k\|^2 g_k^\top d_{k-1}}{v_k}$$

Step 4 Calculate the search direction d_k defined in (4).

Step 5 Determine a steplength $\alpha_k > 0$ using the Strong Wolfe conditions (5) and (6).

Step 6 Set $x_{k+1} = x_k + \alpha_k d_k$ and $k = k + 1$. Return to Step 2.

3. Convergence Analysis

In the subsequent analysis, the focus will be directed towards verifying the convergence of the suggested scheme. This will be accomplished by initially examining the strong Wolfe line search conditions.This study motivates and generalizes convergence in terms of lemmas and assumptions by [29].

$$\begin{aligned} f(x_k + \alpha_k d_k) - f(x_k) &\leq \delta \alpha_k g_k^T d_k, \\ g(x_k + \alpha_k d_k)^T d_k &\geq \sigma g_k^T d_k, \end{aligned} \tag{26}$$

where $0 < \delta < \sigma < 1$. In addition, we will assume that

Assumption 3.1 *The level set $\mathcal{H} = \{x : f(x) \leq f(x_0)\}$ is bounded.*
Assumption 3.2 *Suppose $\mathcal{H}$ is some neighborhood of $\bar{x}$, then f is continuously differentiable and its gradient Lipschitz continuous on $\mathcal{H}$. That is, we can find $L > 0$ such that for all x,*

$$|g(x) - g(\bar{x})| \leq L\|x - \bar{x}\|, \; x \in \mathcal{H}. \tag{27}$$

Before we establish the convergence result, we require the result of the following Lemma.
Lemma 3.1 Let $\{d_k\}$ be defined by Equation (21)–(23), then there is $\kappa > 0$ such tat $\|d_k\| \leq \kappa \|g_k\|$.
Proof. From Equation (22),

$$\begin{aligned} |\beta_k| &= \left| \frac{\|g_k\|^2}{w_k} - \frac{\|g_k\|^2 g_k^\top d_{k-1}}{w_k^2} \right| \\ &\leq \frac{\|g_k\|^2}{\lambda \|d_{k-1}\| \|g_k\|} + \frac{\|g_k\|^3 \|d_{k-1}\|}{(\lambda \|d_{k-1}\| \|g_k\|)^2} \\ &= \left(\frac{1}{\lambda} + \frac{1}{\lambda^2} \right) \frac{\|g_k\|}{\|d_{k-1}\|}. \end{aligned} \tag{28}$$

Also,

$$\begin{aligned} |\gamma_k| &= \left| -t_k \frac{g_k^T d_{k-1}}{w_k} \right| \\ &= t_k \left| \frac{g_k^T d_{k-1}}{w_k} \right| \\ &\leq \bar{t} \frac{\|g_k\| \|d_{k-1}\|}{w_k} \\ &\leq \bar{t} \frac{\|g_k\| \|d_{k-1}\|}{\lambda \|d_{k-1}\| \|g_k\|} \\ &= \frac{\bar{t}}{\lambda}. \end{aligned} \tag{29}$$

Therefore, from Equation (21)–(24), (26) and (27), we have

$$\begin{aligned}\|d_k\| &= \| - g_k + \beta_k d_{k-1} + \gamma_k g_k\| \\ &\leq \|g_k\| + |\beta_k|\|d_{k-1}\| + |\gamma_k|\|g_k\| \\ &\leq \|g_k\| + \left(\frac{1}{\lambda} + \frac{1}{\lambda^2}\right)\frac{\|g_k\|}{\|d_{k-1}\|}\|d_{k-1}\| + \frac{\bar{t}}{\lambda}\|g_k\| \\ &= \left(1 + \frac{1+\bar{t}}{\lambda} + \frac{1}{\lambda^2}\right)\|g_k\|.\end{aligned} \tag{30}$$

Letting

$$\kappa = \left(1 + \frac{1+\bar{t}}{\lambda} + \frac{1}{\lambda^2}\right),$$

we have

$$\|d_k\| \leq \kappa\|h_k\|. \tag{31}$$

The following theorem is similar to the one in [30].

Theorem 3.1 *Let Assumptions 3.1 – 3.2 hold and the sequences $\{x_k\}$ and $\{d_k\}$ are defined by Equation (2) and Equation (21), respectively. If $\{\alpha_k\}$ satisfies (23), then*

$$\lim_{k\to\infty} \|g_k\| = 0. \tag{32}$$

Proof. Using (7) and (23), we have

$$\begin{aligned}f(x_k + \alpha_k d_k) &\leq f(x_k) + \theta\alpha_k g_k^T d_k \\ &\leq f(x_k) - \theta\alpha_k c\|g_k\|^2,\end{aligned}$$

which implies

$$\theta\alpha_k\|g_k\|^2 \leq f(x_k) - f(x_k + \alpha_k d_k).$$

Taking the infinite sum of the above inequality together with Assumption 3.1, we get

$$c\theta\sum_{k=0}^{\infty}\alpha_k\|g_k\|^2 < \infty,$$

which implies

$$\alpha_k\|g_k\|^2 \to 0,\ k \to \infty. \tag{33}$$

Also from Equation (7), (23) and Assumption 3.2, we have

$$\begin{aligned} L\|\alpha_k d_k\|\|d_k\| &\geq \|g(x_k + \alpha_k d_k) - g_k\|\|d_k\| \\ &\geq [g(x_k + \alpha_k d_k) - g_k]^T d_k \\ &\geq -(1-\sigma) g_k^T d_k \\ &\geq (1-\sigma) c \|g_k\|^2. \end{aligned}$$

So, we have from Equation (29) and the above inequality that

$$\alpha_k \geq \frac{(1-\sigma)c\|g_k\|^2}{L\|d_k\|^2} \geq \frac{(1-\sigma)c}{L\kappa^2}. \tag{34}$$

Combining Equation (31) and (32), we have the desired result (30).

4. NUMERICAL EXPERIMENTS

4.1. *Number of Iterations (NOI) and Central Processing Unit (CPU).*

In this study, the efficiency of the proposed HMTTCG algorithm is examined. For performance testing, a collection of 20 benchmark functions from [31] is employed, as these functions are widely used to evaluate and compare TTCG methods. The HMTTCG approach is tested against several established TTCG variants, including TTCGFR, TTCGCD, and TTCGDY from [21, 23, 32], as well as TTCGHS, TTCGLS, and TTCGPRP from [21, 22, 33]. The comparison is based on two main criteria: the Number of Iterations (NOI) and the Central Processing Unit (CPU) time.The experiments were carried out on problems of various sizes, ranging from dimension 1 up to $1,000,000$, ensuring that both small-scale and large-scale cases are considered. To make the evaluation consistent, the results of HMTTCG were compared with classical TTCG schemes such as FR, CD, DY, HS, LS, and PRP. For the HMTTCG algorithm, the parameters were set as $\theta = 0.0001$, $\sigma = 0.009$, $t = 0.3$, and $\mu = 0.01$. All implementations were done in MATLAB R2023a and executed on a personal computer with the following specifications: Acer Aspire A315-42 (LAPTOP-UN7BMPOL), 4 GB RAM, and a 64-bit Windows 11 operating system.

Figures 1 and 2 show the performance profiles of the tested methods. In these profiles, t represents the performance ratio, which is the ratio of a method's result to the best result obtained among all methods for each problem. The function $P(t)$ denotes the proportion of problems that a method can solve within a factor t of the best performance. In other words, t measures relative efficiency, while $P(t)$ indicates the probability of achieving that efficiency level across all test problems.The numerical results are compared using two key metrics, the NOI and CPU time in seconds. Convergence was assessed in this study using the stopping criterion $||g_k|| \leq 10^{-6}$. The numerical findings for each strategy examined using various test function variants are displayed in Tables 3 and 4. A "-" in the table indicates that the algorithm is deemed unsuccessful if the number of iterations (NOI) surpasses 10,000 or if the approach is unable to achieve the intended value.

Table 2 : List of test functions, dimensions & initial points

No	Functions	Dimension	Initial Points
1	Extended Rosenbrock	(50,000 , 100,000 , 1,000,000)	(0.1,...,0.1)
2	Extended Freudenstein & Roth	(10 , 50)	(0.5,...,0.5)
3	Raydan 1	(10 , 50)	(1.1,...,1.1)
4	Diagonal 4	(1,000 , 5,000 , 50,000)	(0.1,...,0.1)
5	Extended Himmelblau	1,000	(5,...,5)
6	FLETCHCR	100	(-5,...,-5)
7	Extended Powel	1,000	(8,...,8)
8	NONSCOMP	(2 , 4 , 10)	(10,...,10)
9	Extended Penalty Function U52	5	(5,...,5)
10	Extended Penalty Function U52	10	(10,...,10)
11	Hager	(5 , 10)	(1,...,1)
12	Cube	2	(4,...,4)
13	Cube	3	(2,...,2)
14	Extended Maratos	(10 , 50 , 10)	(-0.5,...,-0.5)
15	Six Hump Camel	2	(-1.5,-2)
16	Six Hump Camel	2	(-5,-10)
17	Trecanni	2	(-5,10)
18	Zettl	2	(0,0)
19	Zettl	2	(10,10)
20	Shallow	(1,000 , 100,000)	(1.001,...,1.001)
21	Generalized Quartic	(100 , 5,000)	(1.001,...,1.001)
22	Quadratic QF2	(10 , 100)	(0.5,...,0.5)

Table 3 : Numerical result in term of NOI and CPU Time for HMTTCG, FR and DY methods

No	Functions	HMTTCG		TTCGFR		TTCGDY	
		NOI	CPU	NOI	CPU	NOI	CPU
1	Extended Rosenbrock	21	1.7322	28	1.9542	27	1.3700
2	Extended Rosenbrock	21	2.9393	28	3.8226	27	2.7489
3	Extended Rosenbrock	21	29.1213	31	41.212	27	26.2063
4	Extended Freudenstein & Roth	15	0.0450	37	0.0927	21	0.0151
5	Extended Freudenstein & Roth	18	1.0910	32	0.0790	24	0.0288
6	Raydan 1	19	0.0895	19	0.0582	19	0.0457
7	Raydan 1	47	0.0927	47	0.0243	57	0.0387
8	Diagonal 4	2	0.2127	2	0.0102	2	0.3164
9	Diagonal 4	2	0.0534	2	0.0233	2	0.0312
10	Diagonal 4	2	0.1141	2	0.1223	2	0.1286
11	Extended Himmelblau	13	0.3612	10	0.0390	10	0.0737
12	FLETCHCR	37	0.2711	41	0.0681	41	0.0531
13	Extended Powel	4195	11.3619	5814	15.9411	6301	16.032
14	NONSCOMP	11	0.2013	72	0.2467	148	0.0524
15	NONSCOMP	77	0.0470	6388	0.6379	6362	0.4781
16	NONSCOMP	8465	1.4369	9456	0.7655	-	-

(Continued on next page)

Table 3 : *(Continued from previous page)*

No	Functions	HMTTCG		TTCGFR		TTCGDY	
		NOI	CPU	NOI	CPU	NOI	CPU
17	Extended Penalty Function U52	19	0.0525	20	0.0437	20	0.0911
18	Extended Penalty Function U52	12	0.0104	142	0.0619	152	0.0321
19	Hager	9	0.0445	11	0.1216	9	0.0216
20	Hager	11	0.0070	11	0.0043	11	0.0021
21	Cube	54	0.0444	7566	6.4095	31	0.0329
22	Cube	275	0.0407	7877	6.7975	126	0.0190
23	Extended Maratos	34	0.0576	59	0.0838	38	0.0165
24	Extended Maratos	34	0.0218	42	0.0180	43	0.0336
25	Extended Maratos	35	0.0333	42	0.0394	40	0.0229
26	Six Hump Camel	4	0.1666	-	-	5	0.0127
27	Six Hump Camel	7	0.0574	-	-	10	0.0019
28	Trecanni	1	0.0866	7	0.0087	7	0.0057
29	Zettl	1	0.0322	1	0.0215	1	0.0162
30	Zettl	12	0.0582	28	0.0333	18	0.0102
31	Shallow	3	0.1411	3	0.1174	3	0.0561
32	Shallow	4	0.6151	4	5.7561	4	0.3613
33	Generalized Quartic	8	0.2635	11	0.1770	7	2.1848
34	Generalized Quartic	9	0.4222	10	0.4911	7	2.1366
35	Quadratic QF2	26	0.0827	49	0.0626	49	0.0055
36	Quadratic QF2	102	0.1005	158	0.1778	156	0.0317

Table 4 : Numerial result in term of NOI and CPU Time For HMTTCG, CD and LS

No	Functions	HMTTCG		TTCGCD		TTCGLS	
		NOI	CPU	NOI	CPU	NOI	CPU
1	Extended Rosenbrock	21	1.7322	-	-	-	-
2	Extended Rosenbrock	21	2.9393	-	-	23	4.5866
3	Extended Rosenbrock	21	29.1213	-	-	-	-
4	Extended Freudenstein & Roth	15	0.045	-	-	-	-
5	Extended Freudenstein & Roth	18	1.091	5	0.1031	-	-
6	Raydan 1	19	0.0895	52	0.0432	11	0.0217
7	Raydan 1	47	0.0927	-	-	11	0.0269
8	Diagonal 4	2	0.2127	10	0.3349	6	0.3967
9	Diagonal 4	2	0.0534	8	1.7034	6	0.5363
10	Diagonal 4	2	0.1141	-	-	6	0.7624
11	Extended Himmelblau	13	0.3612	-	-	38	0.0513
12	FLETCHCR	37	0.2711	-	-	-	-
13	Extended Powel	4195	11.3619	-	-	-	-
14	NONSCOMP	11	0.2013	-	-	13	0.0776
15	NONSCOMP	77	0.0470	-	-	98	0.0309

(Continued on next page)

Table 4 : *(Continued from previous page)*

No	Functions	HMTTCG		TTCGCD		TTCGLS	
		NOI	CPU	NOI	CPU	NOI	CPU
16	NONSCOMP	8465	1.4369	-	-	-	-
17	Extended Penalty Function U52	19	0.0525	-	-	9	0.0877
18	Extended Penalty Function U52	12	0.0104	-	-	-	-
19	Hager	9	0.0445	12	0.0172	-	-
20	Hager	11	0.007	18	0.0181	-	-
21	Cube	54	0.0444	-	-	-	-
22	Cube	275	0.0407	-	-	-	-
23	Extended Maratos	34	0.0576	-	-	20	0.0437
24	Extended Maratos	34	0.0218	-	-	22	0.0124
25	Extended Maratos	35	0.0333	-	-	-	-
26	Six Hump Camel	4	0.1666	-	-	5	0.0263
27	Six Hump Camel	7	0.0574	-	-	8	0.0233
28	Trecanni	1	0.0866	-	-	1	0.0214
29	Zettl	1	0.0322	1	0.0021	5	0.0153
30	Zettl	12	0.0582	-	-	9	0.0126
31	Shallow	3	0.1411	11	0.0210	3	0.0167
32	Shallow	4	0.6151	15	1.2099	3	0.3956
33	Generalized Quartic	8	0.2635	-	-	-	-
34	Generalized Quartic	9	0.4222	-	-	-	-
35	Quadratic QF2	26	0.0827	-	-	-	-
36	Quadratic QF2	102	0.1005	-	-	104	0.0464

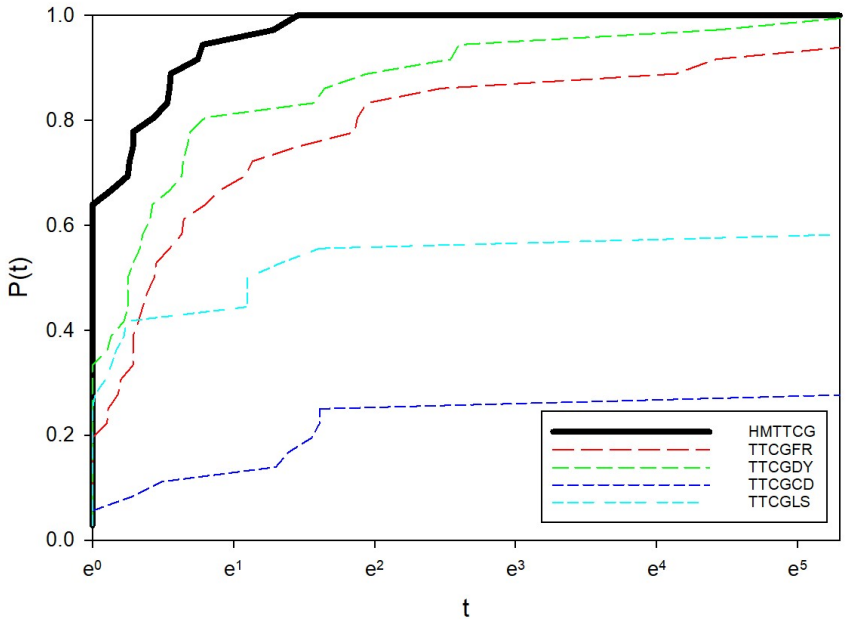

Figure 1 : The performance profile that corresponds to the NOI.

As shown in Figure 1, the suggested HMTTCG approach performs better than TTCGFR, TTCGDY, TTCGCD, and TTCGLS in terms of iterations, requiring a notably smaller number of iterations to converge. The HMTTCG's (black line) steep curve demonstrates its effectiveness, since it quickly achieves 100% success, making it the most dependable and effective approach among those examined. The TTCGDY method (green dashed line) follows as the second-best performer, steadily reaching approximately 80% success within a few iterations, making it a strong alternative when HMTTCG is unavailable. TTCGFR (red dashed line) and TTCGCD (blue dashed line) exhibit moderate performance, with slower growth curves compared to HMTTCG and TTCGDY, making them suitable for applications where rapid convergence is not critical. In contrast, TTCGLS (cyan dashed line) is the least effective, with its success probability increasing very slowly, reaching only 20%after several iterations, indicating that it is unsuitable for fast optimization tasks. In conclusion, HMTTCG stands out as the best method due to its rapid and near-perfect success rate, with TTCGDY as a strong alternative, while TTCGFR and TTCGCD offer moderate performance, and TTCGLS lags significantly behind, emphasizing the need for efficient methods in optimization problems.

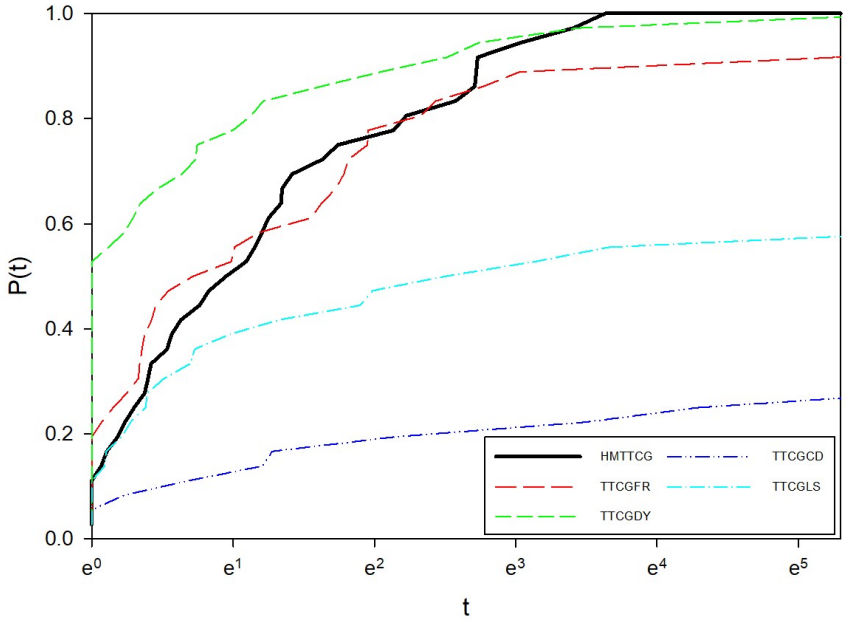

Figure 2 : The performance profile that corresponds to CPU time

Figure 2 shows that the HMTTCG method has a slower convergence time initially compared to the DY and FR methods due to its higher complexity and the limited RAM of the laptop used. DY (green dashed line) achieves the highest CPU efficiency at 96%, followed by FR (red dashed line) at 92%, making them the fastest and most reliable methods in terms of computational time. While HMTTCG (black line) is slower at first, it maintains a strong performance with 85% efficiency, proving to be a competitive option despite its resource-intensive nature. In contrast, LS (cyan dashed line) and CD (blue dashed line) exhibit lower CPU efficiency at 40% and 25%, respectively. Overall, DY and FR emerge as the best performers in early convergence, while HMTTCG remains a strong contender over time.

4.2. *Performance of Hybrid Parameter.*

The HMTTCG method introduces a new parameter, β_k^H, which integrates the strengths of β_k^1 and β_k^2 to enhance optimization efficiency. The method evaluates the frequency of positive values for these parameters, as positive values generally indicate improved optimization directions, leading to faster convergence and more accurate solutions. Table 5 presents the parameter results for β_k^1, β_k^2, and β_k^H, reflecting different performance aspects of the method, while Table 6 summarizes the overall effectiveness of β_k^H within HMTTCG.

Table 5 : Parameter result for β_k^1, β_k^2, and β_k^H by using HMTTCG Method

NO.	Functions	Negative			Zero			Positive		
		β_k^1	β_k^2	β_k^H	β_k^1	β_k^2	β_k^H	β_k^1	β_k^2	β_k^H
1	Extended Rosenbrock	2	0	0	0	0	2	19	21	19
2	Extended Rosenbrock	2	0	0	0	0	2	19	21	19

(Continued on next page)

Table 5 (continued)

NO.	Functions	Negative			Zero			Positive		
		β_k^1	β_k^2	β_k^H	β_k^1	β_k^2	β_k^H	β_k^1	β_k^2	β_k^H
3	Extended Rosenbrock	2	0	0	0	0	2	19	21	19
4	Extended Freudenstein & Roth	1	0	0	0	0	1	14	15	14
5	Extended Freudenstein & Roth	3	0	0	1	0	4	14	18	14
6	Raydan 1	0	0	0	0	0	0	19	19	19
7	Raydan 1	0	0	0	0	0	0	47	47	47
8	Diagonal 4	1	0	0	0	0	1	1	2	1
9	Diagonal 4	1	0	0	0	0	1	1	2	1
10	Diagonal 4	0	0	0	0	0	0	2	2	2
11	Extended Himmelblau	0	0	0	0	0	0	13	13	13
12	FLETCHCR	0	0	0	0	0	0	37	37	37
13	Extended Powel	1	0	0	0	0	1	4194	4195	4194
14	NONSCOMP	1	0	0	0	0	1	10	11	10
15	NONSCOMP	0	0	0	0	0	0	77	77	77
16	NONSCOMP	1	0	0	0	0	1	8464	8465	8464
17	Extended Penalty Function U52	0	0	0	0	0	0	19	19	19
18	Extended Penalty Function U52	2	0	0	0	0	2	10	12	10
19	Hager	0	0	0	0	0	0	9	9	9
20	Hager	0	0	0	0	0	0	11	11	11
21	Cube	0	0	0	0	0	0	54	54	54
22	Cube	2	0	0	0	0	2	273	275	273
23	Extended Maratos	2	0	0	0	0	2	32	34	32
24	Extended Maratos	2	0	0	0	0	2	32	34	32
25	Extended Maratos	2	0	0	0	0	2	33	35	33
26	Six Hump Camel	2	0	0	0	0	2	2	4	2
27	Six Hump Camel	0	0	0	0	0	0	7	7	7
28	Trecanni	0	0	0	0	0	1	0	1	0
29	Zettl	1	0	0	0	0	1	0	1	0
30	Zettl	2	0	0	0	0	2	10	12	10
31	Shallow	0	0	0	0	0	0	2	2	2
32	Shallow	0	0	0	0	0	0	4	4	4
33	Generalized Quartic	5	0	0	0	0	5	3	8	3
34	Generalized Quartic	5	0	0	0	0	5	4	9	4
35	Quadratic QF2	0	0	0	0	0	0	26	26	26
36	Quadratic QF2	0	0	0	0	0	0	102	102	102

Table 6 : Overall Performance of Parameter β_k^{H}

No.	**Functions**	β_k^{H}		
		β_k^1	β_k^2	$Zero$
1	Extended Rosenbrock	3	16	2
2	Extended Rosenbrock	3	16	2
3	Extended Rosenbrock	3	16	2
4	Extended Freudenstein & Roth	3	11	1
5	Extended Freudenstein & Roth	3	15	0
6	Raydan 1	2	17	0
7	Raydan 1	2	45	0
8	Diagonal 4	1	0	1
9	Diagonal 4	0	1	1
10	Diagonal 4	0	1	1
11	Extended Himmelblau	2	11	0
12	FLETCHCR	13	24	0
13	Extended Powel	4	4190	1
14	NONSCOMP	4	6	1
15	NONSCOMP	12	65	0
16	NONSCOMP	53	8411	1
17	Extended Penalty Function U52	3	16	0
18	Extended Penalty Function U52	2	10	0
19	Hager	1	8	0
20	Hager	1	10	0
21	Cube	5	49	0
22	Cube	13	261	1
23	Extended Maratos	3	29	2
24	Extended Maratos	3	29	2
25	Extended Maratos	3	29	3
26	Six Hump Camel	2	0	2
27	Six Hump Camel	3	4	0
28	Trecanni	0	0	1
29	Zettl	0	0	1
30	Zettl	3	7	2
31	Shallow	1	2	0
32	Shallow	1	3	0
33	Generalized Quartic	3	0	5
34	Generalized Quartic	4	1	4
35	Quadratic QF2	12	14	0
36	Quadratic QF2	32	70	0

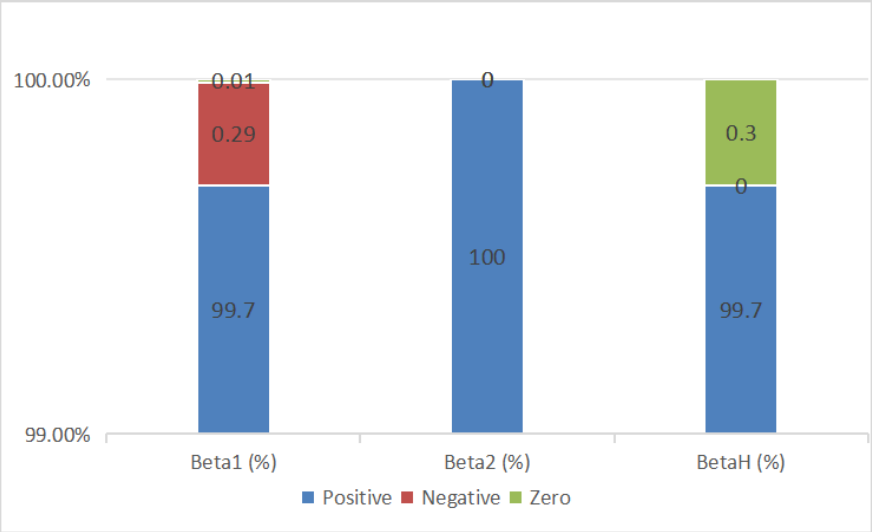

Figure 3 : Performance of β_k^1, β_k^2, and β_k^{H} by using HMTTCG

Figure 3 analysis of β_k^1, β_k^2, and β_k^{H}in the HMTTCG method shows some key findings. β_k^1 has a positive contribution of 99.7% and a small negative value of 0.29%, along with an even smaller zero value of 0.01%. This suggests that while β_k^1 generally improves the method's performance, minor inefficiencies may occur in certain cases due to the negative and zero values. Enhancing β_k^1 could help reduce these inefficiencies and boost the method's overall effectiveness. β_k^2 performs exceptionally well, with a completely positive contribution of 100% and no negative or zero values. This makes β_k^2 the most stable and reliable parameter in the method. Its consistent positive impact highlights its importance and shows that β_k^2 is already optimized, requiring little to no adjustment. For β_k^{H}, the positive contribution is 99.7%, with a zero value of 0.3% and no negative values. While it significantly enhances the method's performance, the small zero contribution indicates minimal inefficiencies in some cases. Refining β_k^{H} could help address these issues and make the algorithm more reliable.

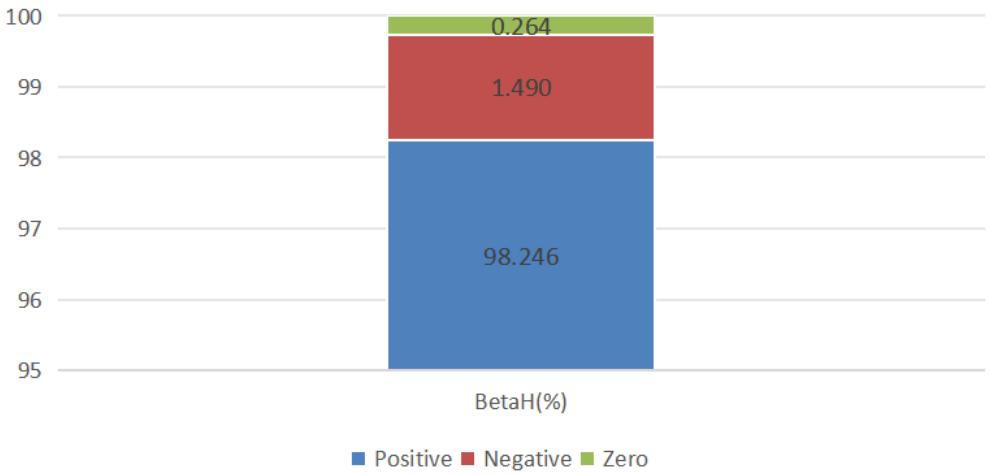

Figure 4 : Performance of β_k^{H}

Figure 4 shows that the performance of β_k^{H} has a mostly positive impact on the HMTTCG method, with 98.246% of its influence being beneficial. This shows that β_k^{H} is very important for the performance of the method. When its value is highly positive, it helps the HMTTCG algorithm remain stable and work more effectively. However, there is a small negative contribution of 1.490%, indicating that in certain situations, β_k^{H} might cause minor inefficiencies. Although this negative effect is very small, it suggests that fine-tuning could help reduce such issues and ensure smoother performance. Additionally, a zero contribution of 0.264% shows there are rare cases where β_k^{H} doesn't have any noticeable effect, which could be further investigated to understand why this happens. In conclusion, β_k^{H} shows a strong and mostly positive performance, making it a key parameter for the HMTTCG method. While the negative and zero contributions are minor, improving β_k^{H} further could help enhance the algorithm's reliability and minimize inefficiencies.

5. CONCLUSION

In order to address large-scale unconstrained optimisation problems, we presented the hybrid memoryless three-term conjugate gradient (HMTTCG) approach in this study. The method uses a new parameter, β_k^{H}, which is intended to always remain non-negative, to combine concepts from other conjugate gradient algorithms. Regardless of the kind of line search used, this method's guarantee of the sufficient descent property is one of its main advantages. The strong Wolfe-Powell line search rule was used to determine the step size.

Numerical experiments were conducted on 36 test issues to assess the role of β_k^{H} in order to support the theoretical analysis. The results demonstrated that, although requiring more CPU time due to its complexity and the computer's memory limitations, the HMTTCG approach is generally more reliable and effective than a number of other approaches now in use. The findings also showed that β_k^{H} typically had positive values. All things considered, the HMTTCG approach presents a viable substitute for large-scale optimisation. It is still a strong and useful tool for resolving challenging optimisation issues, even with certain computational cost issues.

6. ACKNOWLEDGMENT

The authors of this study are grateful to the editors and referees for their valuable feedback and ideas.

7. REFERENCES

[1] M. R. Hestenes, "The conjugate-gradient method for solving linear systems," *Numerical Analysis*, no. 6, p. 83, 1956.

[2] M. R. Hestenes and E. Stiefel, "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. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/jres/049/6/V49.N06.A08.pdf

[3] L. Zhang, W. Zhou, and Y. Dai, "A descent modified polak–ribière–polyak conjugate gradient method and its global convergence," *SIAM Journal on Optimization*, vol. 20, no. 2, pp. 627–640, 2010.

[4] P. Wolfe, "Convergence conditions for ascent methods," *SIAM Journal*, vol. 11, no. 2, pp. 226–235, 1969.

[5] M. J. D. Powell, "Restart procedures for the conjugate gradient method," *Mathematical Programming*, vol. 12, pp. 241–254, 1977.

[6] E. Polak and G. Ribiere, "Note sur la convergence de méthodes de directions conjuguées," *Revue française d'informatique et de recherche opérationnelle. Série rouge*, vol. 3, no. 16, pp. 35–43, 1969.

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

[8] 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.

[9] 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.

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

[11] R. Fletcher, *Practical methods of optimization*. John Wiley & Sons, 2000.

[12] S. Babaie-Kafaki, R. Ghanbari, and N. Mahdavi-Amiri, "Two new conjugate gradient methods based on modified secant equations," *Journal of Computational and Applied Mathematics*, vol. 234, no. 5, pp. 1374–1386, 2010.

[13] S. Babaie-Kafaki, M. Fatemi, and N. Mahdavi-Amiri, "Two effective hybrid conjugate gradient algorithms based on modified bfgs updates," *Numerical Algorithms*, vol. 58, no. 3, pp. 315–331, 2011.

[14] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2004.

[15] N. Andrei, "A simple three-term conjugate gradient algorithm for unconstrained optimization,"

Journal of Computational and Applied Mathematics, vol. 241, pp. 19–29, 2013.

[16] Andrei, "Another three-term conjugate gradient algorithm for unconstrained optimization," *Numerical Algorithms*, vol. 62, pp. 401–423, 2013.

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

[18] M. Li, "A modified hestense–stiefel conjugate gradient method close to the memoryless bfgs quasi-newton method," *Optimization Methods and Software*, vol. 33, pp. 1–18, 05 2017.

[19] N. Li, "A three term polak-ribière-polyak conjugate gradient method close to the memoryless bfgs quasi-newton method," *Journal of Industrial Management Optimization*, vol. 13, pp. 1–16, 01 2017.

[20] Y. Li, P. Y. Chen, T. Du, and W. Matusik, "Learning preconditioners for conjugate gradient pde solvers," in *International Conference on Machine Learning*. PMLR, 2023, pp. 19 425–19 439.

[21] I. Abubakar, Sulaiman, M. Mamat, and Waziri, "A new hybrid conjugate gradient method for unconstrained optimization problems," *Journal of Optimization Theory and Applications*, vol. 192, no. 3, pp. 902–920, 2022.

[22] P. Kumam, A. B. Abubakar, M. Malik, A. H. Ibrahim, N. Pakkaranang, and B. Panyanaks, "A hybrid hs-ls conjugate gradient algorithm for unconstrained optimization with applications in motion control and image recovery," *Computers & Mathematics with Applications*, vol. 115, pp. 304–320, 2023.

[23] F. Deepho, "A ttcg method combining fr-dy parameters," *Journal of Optimization Techniques*, vol. 55, no. 3, pp. 789–803, 2020.

[24] D. Li, Y. Li, and S. Wang, "An improved three-term conjugate gradient algorithm for constrained nonlinear equations under non-lipschitz conditions and its applications," *Mathematics*, vol. 12, no. 16, p. 2556, 2024.

[25] M. A. I. Ishak, Y. Ayub, and S. M. Marjugi, "A new family of hybrid three-term conjugate gradient bnc-btc based on scaled memoryless bfgs update for unconstrained optimization problems," *Applied Mathematics and Computational Intelligence*, vol. 14, no. 2, pp. 19–36, 2025.

[26] D. F. Shanno, "Conjugate gradient methods with inexact searches," *Mathematics of Operations Research*, vol. 3, no. 3, pp. 244–256, 1978.

[27] J. Nocedal, "Updating quasi-newton matrices with limited storage," *Mathematics of Computation*, vol. 35, no. 151, pp. 773–782, 1980.

[28] M. Malik, I. M. Sulaiman, A. B. Abubakar, G. Ardaneswari, and Sukono, "A novel hybrid three-term conjugate gradient method for unconstrained optimization," *Mathematics*, vol. 11, no. 1, pp. 1–20, 2023.

[29] J. Deepho, A. B. Abubakar, M. Malik, and I. K. Argyros, "Solving unconstrained optimization problems via hybrid cd-dy conjugate gradient methods with applications," *Journal of Computational and Applied Mathematics*, vol. 405, p. 113823, 2022.

[30] G. Yuan, Z. Sheng, B. Wang, W. Hu, and C. Li, "The global convergence of a modified bfgs method for nonconvex functions," *Journal of Computational and Applied Mathematics*, vol. 327, pp. 274–294, 2018.

[31] W. Sun and Y.-X. Yuan, *Optimization Theory and Methods*. New York: Springer, 2006. [Online]. Available: https://doi.org/10.1007/0-387-30931-7

[32] F. Abubakar, "Solving unconstrained optimization problems via hybrid cd-dy conjugate gradient methods with applications," *Journal of Computational Optimization*, vol. 58, no. 2, pp. 345–359, 2020.

[33] S. Devila, M. Malik, and W. Giyarti, "A new hybrid prp-mmsis conjugate gradient method and its application in portofolio selection," *Jurnal Riset dan Aplikasi Matematika (JRAM)*, vol. 5, no. 1, pp. 47–59, 2021.