Asia Pac. J. Math. 2026 **13**:163

ACCELERATED KMAR ALGORITHM FOR SOLVING MONOTONE EQUATIONS

SARAH GHANIM MAHMOOD AL-KABABCHEE[1], RANA Z. AL-KAWAZ[2,*], MARWAN S. JAMEEL[3]

[1]Department of Mathematics, Education College for Pure Sciences, University of AL-Hamdaniya, Mosul, Iraq
[2]Department of Mathematics, College of Basic Education, University of Telafer, Mosul, Iraq
[3]Department of Climate Change, College of Environmental Sciences, Mosul University, Mosul, Iraq
[*]Corresponding author: ranazidan@uotelafer.edu.iq

Received Aug. 3, 2026

Abstract. This paper proposes a conjugate gradient-type acceleration algorithm, called the AKMAR-New algorithm, for solving constrained monotonic nonlinear equations, a class of problems that naturally arise in integration and equilibrium problems. The proposed method extends the classical conjugate KMAR by incorporating a dynamic measure function, derived from the extended conjugate gradient model, into both its numerator and denominator, while incorporating an inertial acceleration limit in the search direction. This design preserves the replayability and non-negativity of the original KMAR while improving practical convergence behavior, without introducing additional evaluations of the function. Under standard assumptions of monotonicity and Lipschitz continuity, it is analytically shown that the algorithm satisfies the condition of sufficient descent regardless of any linear search, and the global convergence of the generated sequence is proven without requiring the convexity of the underlying mapping. The proposed method is combined with a superlevel projection technique to ensure the viability of iterations under the set of constraints. Numerical experiments were conducted on eight reference test problems, each evaluated in six initial points and five problem dimensions ranging from 1,000 to 100,000 variables. These were compared with two existing non-derivative projection algorithms, MKAW and PLCG. The results, evaluated in terms of the number of iterations, function evaluations, CPU time, and the remainder of the criterion, and further validated using Dolan and Moré performance profiles, consistently outperform the two competing algorithms in all metrics tested, demonstrating robustness with respect to both problem dimensions and initial point selection. These results indicate that the proposed algorithm is a computationally efficient and theoretically sound tool for large-scale monotonic equation problems, with potential applications in signal recovery and compact sensing tasks.

2020 Mathematics Subject Classification. 65K05; 49M37; 65H10; 47H05.

Key words and phrases. conjugate gradient method; monotonic equations; global convergence; projection method; large-scale optimization; KMAR method.

DOI: 10.28924/APJM/13-163

1. Introduction

Iterative methods for solving nonlinear equations and optimization problems have attracted sustained research interest due to their broad applicability in science, engineering, and economics. Among the most competitive of these methods is the class of conjugate gradient (CG) algorithms, which are widely preferred for large-scale problems due to their simplicity, low memory requirements, and strong convergence properties. The general iterative framework takes the form

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

where $\alpha_k > 0$ is the step size obtained by an appropriate line search, and d_k is the conjugate search direction determined by choosing a scalar conjugate parameter β_k.

Several classical choices for β_k have been proposed and extensively studied, including the Hestenes–Stiefel (HS) [1], Fletcher–Reeves (FR) [2], Polak–Ribière–Polyak (PRP) [3,4], and Rivaie–Mustafa–Ismail–Leong (RMIL) [5] parameters, defined respectively as

$$\beta_k^{\mathrm{HS}} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{d_k^T(g_{k+1} - g_k)}, \tag{2}$$

$$\beta_k^{\mathrm{FR}} = \frac{\|g_{k+1}\|^2}{\|g_k\|^2}, \tag{3}$$

$$\beta_k^{\mathrm{PRP}} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{\|g_k\|^2}, \tag{4}$$

$$\beta_k^{\mathrm{RMIL}} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{\|d_k\|^2}. \tag{5}$$

Each of these methods exhibits distinct convergence characteristics. The FR method, although globally convergent under exact line search [6], can suffer from numerical instability and slow practical progress. The PRP and HS methods benefit from automatic restart properties that improve practical performance but may not converge globally [7].

Motivated by these limitations, Kamfa, Mamat, Abashar, and Rivaie [8] introduced the KMAR parameter, defined by

$$\beta_k^{\mathrm{KMAR}} = \frac{g_{k+1}^T(g_{k+1} - g_k)}{g_k^T(g_{k+1} + g_k)}. \tag{6}$$

This formulation retains the numerator structure of the PRP, HS, and RMIL parameters thereby preserving the restart property while introducing a new denominator that simplifies convergence analysis and ensures the parameter is always non-negative, that is,

$$0 \leq \beta_k^{\mathrm{KMAR}} \leq \frac{\|g_k\|^2}{\|g_{k-1}\|^2} = \beta_k^{\mathrm{FR}}. \tag{7}$$

The KMAR method was shown to satisfy the sufficient descent condition and to be globally convergent under exact line search [8], and its numerical performance was demonstrated to be competitive with,

and in many cases superior to, the FR, AMRI, and PRP methods across a broad set of standard test functions of varying dimensions.

Subsequent work extended the KMAR framework through the development of a three-term variant (KMM4) [9], which incorporates the modified KMAR conjugate parameter into a three-term search direction of the form

$$d_{k+1} = \begin{cases} -g_{k+1}, & k = 0, \\ -g_{k+1} + \beta_k^{\mathrm{KMAR}} d_k - \theta_k \delta_k, & k \geq 1, \end{cases} \tag{8}$$

where $\theta_k = g_{k+1}^T d_k / \|g_k\|^2$ and $\delta_k = g_{k+1} - (g_{k+1}/g_k)\, g_k$. This three-term formulation was shown to satisfy both the sufficient descent condition and a trust region property under the Wolfe line search, and to achieve global convergence for both convex and non-convex objective functions [9]. Numerical comparisons confirmed improved efficiency over the FR, PRP, MPRP [10], and RMIL methods across benchmark problems with dimensions ranging up to 30,000. There are many articles on developing this method and in this direction, as in. Related contributions from extended CG algorithms based on non-quadratic rational function models [11] have further demonstrated that carefully constructed modifications of the search direction and the underlying function model can yield measurable gains in iteration count and CPU time over standard CG methods. One commonly used variant of this method is called Extended Conjugate Gradient (ECG) algorithms based on non-quadratic fractional models. Taqi and Ali [11] investigated an ECG method based on a fractional model from which the following two parameters were derived:

$$\varphi_{k+1}^1 = \left[\frac{\sin f_{k+1}}{\sin f_k}\right] \left[\frac{-n \sin f_k}{w}\right]^2 \tag{9}$$

and

$$\varphi_{k+1}^2 = \sqrt{\varphi_{k+1}^1} \sqrt{\frac{\sin f_{k+1}}{\sin f_k}} + \frac{n \sin f_{k+1}}{\cos f_{k+1} + \frac{\varepsilon_1}{\varepsilon_2}} \tag{10}$$

where $0 < \varphi_0 < 1$, which is substituted into the standard direction update $d_{k+1} = -g_{k+1} + \beta_k^{\mathrm{ECG}} d_k$ with the modified HS or PR parameter scaled by φ_i. where

$$n = \frac{\lambda_k\, g_k^T d_k}{2}, \qquad w = \cos f_{k+1} - \cos f_k, \tag{11}$$

λ_k being the step size and g_k, d_k the gradient and search direction at iteration k, respectively. The quantity w measures the change in the cosine of the objective value between successive iterates and serves as an adaptive curvature indicator: it is large when the function landscape is strongly nonlinear and tends to zero when the landscape is nearly quadratic, producing an automatically scaled conjugate parameter. The ratio $\varepsilon_1/\varepsilon_2$ is re-evaluated at every iteration from available quantities as

$$\frac{\varepsilon_1}{\varepsilon_2} = \frac{w \cos f_k + n \sin f_k \cos f_{k+1}}{-(w + n \sin f_k)} \tag{12}$$

Under this model, the conjugate structure of the iterates is preserved while the function landscape is reshaped to improve practical convergence [11].

Despite these advances, the application of KMAR-based methods to the problem of solving monotone equations a setting that arises naturally in variational inequalities, complementarity problems, and related equilibrium formulations [12–14]—remains relatively unexplored. We consider the constrained nonlinear monotone equation

$$F(x) = 0, \quad x \in \Omega, \tag{13}$$

where $F : \mathbb{R}^n \to \mathbb{R}^n$ is continuous and monotone, and $\Omega \subseteq \mathbb{R}^n$ is a nonempty, closed, and convex set. Recall that F is monotone if

$$\langle F(x) - F(y),\ x - y\rangle \geq 0, \quad \forall\, x, y \in \mathbb{R}^n, \tag{14}$$

a structural condition that is weaker than smoothness or convexity requirements and thus accommodates a wider class of practically relevant problems [15,16]. In compressive sensing, for instance, monotone equations may be reformulated as ℓ_1-norm regularized optimization problems [16–18]. Projection-based and derivative-free methods adapted from the CG framework are particularly well-suited to the setting (13), since they avoid the need for Jacobian evaluations while retaining convergence guarantees [19–21]. Recent studies have shown that hybrid optimization strategies can significantly improve convergence and solution quality by integrating complementary mechanisms within a unified framework, motivating the development of more efficient deterministic optimization algorithms [22].

The remainder of the paper is organized as follows. Section 1 introduces the necessary preliminaries and notation. Section 2 presents the proposed algorithm in detail. Section 3 establishes global convergence of the method. Section 4 reports numerical results and comparisons with existing algorithms. Section 5 provides concluding remarks.

2. Proposed Method

In this paper, we propose and analyze an accelerated version of the KMAR algorithm designed specifically for solving (13). The proposed method incorporates an inertial acceleration step and an adaptive step-size strategy that exploits the KMAR parameter structure to achieve faster iterates without incurring additional function evaluations. Specifically, the new conjugate parameter is constructed by introducing a dynamic scaling function φ derived from the ECG rational model [11] into both the numerator and denominator of (6):

$$\beta_k^{\mathrm{AKMAR}} = \frac{F_{k+1}^T(\varphi_{k+1}F_{k+1} - F_k)}{F_k^T(\varphi_{k+1}F_{k+1} + F_k)} \tag{15}$$

and the search direction d_{k+1} defined by

$$d_{k+1} = \begin{cases} -\vartheta_k F_{k+1} + \beta_k^{\mathrm{AKMAR}} s_k, & F_{k+1}^{\mathsf{T}} s_k \le 0, \\ -F_{k+1}, & F_{k+1}^{\mathsf{T}} s_k > 0, \end{cases} \qquad k \ge 0, \tag{16}$$

where

$$\vartheta_k = 1 + \frac{F_{k+1}^T s_k}{\|F_k\|^2}, \tag{17}$$

where $F_k := F(x_k)$ and $s_k = \alpha_k d_k$. We prove global convergence of the scheme under standard monotonicity and Lipschitz continuity assumptions and we present numerical experiments on benchmark problems demonstrating the superiority of the accelerated method over existing approaches.

Algorithm 1 AKMAR-New Algorithm

Require: $x_0 \in \mathbb{R}^n$; tolerance $\varepsilon > 0$; line-search parameters $\eta, \sigma \in (0,1)$; scaling bounds $0 < \varphi_{\min} < 1$; curvature safeguard $\delta > 0$.

Initialize $k := 0$, $F_0 := F(x_0)$, $d_0 := -F_0$.

repeat

(S1) Termination. If $\|F_k\| \le \varepsilon$, **stop**; output x_k.

(S2) Line search. Set $\alpha_k = \eta^w$, with $w \in \mathbb{N}_0$ the smallest integer satisfying

$$-F(x_k + \alpha_k d_k)^{\mathsf{T}} d_k \;\ge\; \sigma\, \alpha_k \, \|F(x_k + \alpha_k d_k)\| \, \|d_k\|^2 \tag{18}$$

and put $z_k = x_k + \alpha_k d_k$, $s_k = \alpha_k d_k$.

(S3) Projection. If $z_k \in \Omega$ and $\|F(z_k)\| \le \varepsilon$, **stop**; output z_k. Otherwise,

$$x_{k+1} = P_\Omega\!\left(x_k - \frac{F(z_k)^{\mathsf{T}}(x_k - z_k)}{\|F(z_k)\|^2}\, F(z_k) \right) \tag{19}$$

(S4) Safeguarded scaling. With $\cos f_k = F_k^{\mathsf{T}} d_k / (\|F_k\|\|d_k\|)$ and $w_k = \cos f_{k+1} - \cos f_k$, set $w_k \leftarrow \delta \cdot \mathrm{sign}(w_k)$ whenever $|w_k| < \delta$; compute $\varphi_{k+1}^{\mathrm{raw}}$ from (9)–(10), and clip

$$\varphi_{k+1} := \min\big\{ \max\{\varphi_{k+1}^{\mathrm{raw}}, \varphi_{\min}\},\, 1 \big\}. \tag{20}$$

(S5) Direction update. Compute β_k^{AKMAR} and ϑ_k from (15), (17) respectively and update the search direction d_{k+1} according to (16)

$k \leftarrow k + 1$.

until convergence at S1 or S3

3. Convergence Analysis

In this section, two theoretical properties of the proposed algorithm are established: sufficient descent and global convergence. Both are based on the standard assumptions that follow.

Assumption 1. *The mapping F is taken to satisfy the following three requirements:*

(*i*) *The solution set $\Omega^* = \{x \in \Omega : F(x) = 0\}$ is nonempty.*

(*ii*) *F is Lipschitz continuous: some constant $L > 0$ exists for which*

$$\|F(x) - F(y)\| \leq L\|x - y\|, \qquad x, y \in \mathbb{R}^n. \tag{21}$$

(*iii*) *F is a monotone operator, meaning*

$$\langle F(x) - F(y),\, x - y\rangle \geq 0, \qquad x, y \in \mathbb{R}^n. \tag{22}$$

Under Assumption 1, the scaling function satisfies the following boundedness property, which is essential for the subsequent analysis.

Lemma 1. *Suppose Assumption 1 holds. Let φ_{k+1} be defined by (9) and (10), respectively. Then there exist constants $0 < \varphi_{\min} \leq \varphi_{\max} < \infty$ such that*

$$\varphi_{\min} \leq \varphi_{k+1} \leq \varphi_{\max}, \quad \forall\, k \geq 0. \tag{23}$$

Proof. Since $0 < \varphi_0 < 1$ and the iterates $\{x_k\}$ remain in a bounded level set (see Lemma 4 below), the quantities f_i, f_{i-1}, n, and w are all bounded. By the constraint $w \neq 0$ enforced in the algorithm (i.e., $w \geq \delta > 0$ for a small safeguard δ), the expression $(-n \sin f_k/w)^2$ is bounded above and away from zero. Consequently, both φ_{k+1}^1 and φ_{k+1}^2 are uniformly bounded, which establishes (23). □

Lemma 2. *Suppose Assumption 1 holds and Lemma 1 is satisfied. Then the denominator of β_k^{AKMAR} satisfies*

$$F_k^T(\varphi_{k+1}F_{k+1} + F_k) \geq (1 + \varphi_{min})\|F_k\|^2 - \varphi_{max}L\alpha_k\|F_k\|\,\|d_k\| \tag{24}$$

and this lower bound is strictly positive whenever

$$\alpha_k\|d_k\| < \frac{(1 + \varphi_{min})\|F_k\|}{\varphi_{max}L} \tag{25}$$

Proof. Expanding the denominator directly,

$$F_k^T(\varphi_{k+1}F_{k+1} + F_k) = \varphi_{k+1}F_k^T F_{k+1} + \|F_k\|^2 \tag{26}$$

Writing $F_{k+1} = F_k + (F_{k+1} - F_k)$, we obtain

$$F_k^T F_{k+1} = \|F_k\|^2 + F_k^T(F_{k+1} - F_k). \tag{27}$$

By the Cauchy–Schwarz inequality and the Lipschitz condition (21),

$$F_k^T(F_{k+1} - F_k) \geq -\|F_k\|\,\|F_{k+1} - F_k\| \geq -L\alpha_k\|F_k\|\|d_k\| \tag{28}$$

so that

$$F_k^T F_{k+1} \geq \|F_k\|^2 - L\alpha_k \|F_k\| \|d_k\| \tag{29}$$

Using $\varphi_{min} \leq \varphi_{k+1} \leq \varphi_{max}$ from Lemma 1, and bounding the positive and negative parts separately,

$$\varphi_{k+1} F_k^T F_{k+1} \geq \varphi_{min} \|F_k\|^2 - \ \varphi_{max} L\alpha_k \|F_k\| \|d_k\|. \tag{30}$$

Adding $\|F_k\|^2$ to both sides gives

$$F_k^T (\varphi_{k+1} F_{k+1} + F_k) \geq (1 + \varphi_{min}) \|F_k\|^2 - \varphi_{max} L\alpha_k \|F_k\| \ \|d_k\| \tag{31}$$

which establishes (24). The right-hand side is strictly positive precisely when (25) holds. □

Lemma 3 (Sufficient Descent)**.** *Suppose Assumption 1 holds and suppose that the φ_{k+1} defined in (9)–(10) is capped so that, in place of Lemma 1, it satisfies*

$$0 < \varphi_{min} \leq \varphi_{k+1} \leq \varphi_{max} \leq 1 \qquad \forall k \geq 0 \tag{32}$$

Suppose further that the search direction d_{k+1} is generated (16), where β_k^{AKMAR} is as in (15) and ϑ_k is as in (17). Then, the direction d_{k+1} satisfies the sufficient descent condition

$$F_{k+1}^T d_{k+1} \leq -\|F_{k+1}\|^2, \forall k \geq 0. \tag{33}$$

Proof. **Case $k = 0$.** Directly,

$$F_0^T d_0 = F_0^T (-F_0) = -\|F_0\|^2 \tag{34}$$

Case $k \geq 1$, restart the branch ($F_{k+1}^T s_k > 0$). Here $d_{k+1} = -F_{k+1}$, so

$$F_{k+1}^T d_{k+1} = -\|F_{k+1}\|^2, \tag{35}$$

and (33) holds with equality.

Case $k \geq 1$, main branch ($F_{k+1}^T s_k \leq 0$). Multiplying (16) by F_{k+1}^T,

$$F_{k+1}^T d_{k+1} = -\vartheta_k \|F_{k+1}\|^2 + \beta_k^{AKMAR} F_{k+1}^T s_k. \tag{36}$$

Bounding the numerator of β_k^{AKMAR}. By Cauchy–Schwarz,

$$F_{k+1}^T (\varphi_{k+1} F_{k+1} - F_k) \leq \varphi_{max} \|F_{k+1}\|^2 + \|F_{k+1}\| \|F_k\| \tag{37}$$

Bounding the denominator. By (24),

$$F_k^T (\varphi_{k+1} F_{k+1} + F_k) \geq \|F_k\|^2 > 0 \tag{38}$$

Combining (37) and (38), and using $\varphi_{max} \leq 1$ from (32):

$$\beta_k^{AKMAR} \leq \varphi_{max} \frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{\|F_{k+1}\|}{\|F_k\|} \leq \frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{\|F_{k+1}\|}{\|F_k\|} \tag{39}$$

Since we are in the branch $F_{k+1}^T s_k \le 0$, multiplying (39) by the *non-positive* quantity $F_{k+1}^T s_k$ reverses the inequality:

$$\beta_k^{AKMAR} F_{k+1}^T s_k \le \left(\frac{\|F_{k+1}\|^2}{\|F_k\|^2} + \frac{\|F_{k+1}\|}{\|F_k\|} \right) F_{k+1}^T s_k \le \frac{\|F_{k+1}\|^2}{\|F_k\|^2} F_{k+1}^T s_k \tag{40}$$

where the last inequality uses $\dfrac{\|F_{k+1}\|}{\|F_k\|} F_{k+1}^T s_k \le 0$ once more.

Substituting (40) into (36) and using the definition of ϑ_k from (17):

$$\begin{aligned} F_{k+1}^T d_{k+1} &\le -\left(1 + \frac{F_{k+1}^T s_k}{\|F_k\|^2}\right) \|F_{k+1}\|^2 + \frac{\|F_{k+1}\|^2}{\|F_k\|^2} F_{k+1}^T s_k \\ &= -\|F_{k+1}\|^2 - \frac{\|F_{k+1}\|^2}{\|F_k\|^2} F_{k+1}^T s_k + \frac{\|F_{k+1}\|^2}{\|F_k\|^2} F_{k+1}^T s_k \\ &= -\|F_{k+1}\|^2 \end{aligned} \tag{41}$$

Hence (33) holds in both branches, which completes the proof. □

Lemma 4. *Let Assumption 1 be satisfied, and fix any solution $\bar{x} \in \Omega^*$. For the iterates $\{x_k\}$ produced by the AKMAR-New algorithm, one can find positive constants a, b for which*

$$\|x_k\| \le a, \qquad \|F_k\| \le b, \qquad k = 0, 1, 2, \ldots \tag{42}$$

Furthermore, the sequence $\{\|x_k - \bar{x}\|\}$ is convergent, and the step length obeys

$$\lim_{k \to \infty} \alpha_k \|d_k\| = 0. \tag{43}$$

Proof. The proof follows the standard projection argument. By the non-expansive property of P_Ω, monotonicity of F, and the hyperplane projection step (19), we have for any solution $\bar{x} \in \Omega^*$:

$$\begin{aligned} \|x_{k+1} - \bar{x}\|^2 &= \left\| P_\Omega\left(x_k - \frac{F(z_k)^T (x_k - z_k)}{\|F(z_k)\|^2} F(z_k) \right) - \bar{x} \right\|^2 \\ &\le \left\| x_k - \frac{F(z_k)^T (x_k - z_k)}{\|F(z_k)\|^2} F(z_k) - \bar{x} \right\|^2 \\ &= \|x_k - \bar{x}\|^2 - 2 \frac{[F(z_k)^T (x_k - z_k)]^2}{\|F(z_k)\|^2} + \frac{[F(z_k)^T (x_k - z_k)]^2}{\|F(z_k)\|^2}. \end{aligned}$$

Since $F(\bar{x}) = 0$ and monotonicity gives $F(z_k)^T (z_k - \bar{x}) \ge 0$, the cross term satisfies $F(z_k)^T (x_k - z_k) \ge F(z_k)^T (x_k - \bar{x}) \ge 0$. Therefore

$$\|x_{k+1} - \bar{x}\|^2 \le \|x_k - \bar{x}\|^2 - \frac{[F(z_k)^T (x_k - z_k)]^2}{\|F(z_k)\|^2}, \tag{44}$$

which shows $\{\|x_k - \bar{x}\|\}$ is non-increasing and hence convergent. Boundedness of $\{x_k\}$ follows, and boundedness of $\{F_k\}$ follows from the Lipschitz condition (21). The limit (43) follows from summing (44) and the line search condition. □

Lemma 5. *Under Assumption 1, the following properties hold for the step-size sequence $\{\alpha_k\}$:*

(1) *The line-search condition* (18) *is never left unsatisfied: a suitable* α_k *always exists.*

(2) *Moreover,* α_k *admits the lower estimate*

$$\alpha_k > \min\left\{1,\ \frac{r\|F_k\|^2}{(L+\sigma)\,b\,\|d_k\|^2}\right\}. \tag{45}$$

Proof. We prove only (2). Suppose $\alpha_k \neq 1$; then α_k/r fails the line search, meaning

$$-F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)^T d_k < \sigma\,\tfrac{\alpha_k}{r}\,\left\|F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)\right\|\,\|d_k\|^2.$$

By the sufficient descent property from Lemma 3 and the Lipschitz continuity of F:

$$\begin{aligned}
\|F_k\|^2 &= -F_k^T d_k \\
&= \left(F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)-F(x_k)\right)^T d_k - F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)^T d_k \\
&\le L\,\tfrac{\alpha_k}{r}\,\left\|F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)\right\|\,\|d_k\|^2 + \sigma\,\tfrac{\alpha_k}{r}\,\left\|F\left(x_k+\tfrac{\alpha_k}{r}d_k\right)\right\|\,\|d_k\|^2 \\
&< \frac{(L+\sigma)}{r}\,\alpha_k\,b\,\|d_k\|^2.
\end{aligned}$$

Rearranging gives (45), completing the proof. □ □

Theorem 1 (Global convergence)**.** *Assume that Assumption 1 is in force, and let* $\{x_{k+1}\}$ *denote the iterates produced by the AKMAR-New algorithm. Then the generated residuals satisfy*

$$\liminf_{k\to\infty}\|F_k\| = 0. \tag{46}$$

Consequently, the whole sequence $\{x_k\}$ *converges to some* $\bar{x}\in\Omega^*$*, that is, to a point at which* $F(\bar{x})=0$*.*

Proof. Assume, to the contrary, that $\liminf_{k\to\infty}\|F_k\| \neq 0$. This means we can find some $\upsilon > 0$ for which

$$\|F_k\| \ge \upsilon, \qquad k = 0,1,2,\ldots \tag{47}$$

Step 1: The sequence $\{d_k\}$ *stays bounded.* When $k=0$ we simply have $\|d_0\| = \|F_0\| \le b$. For the remaining indices $k\ge 1$, applying (17) together with the Cauchy–Schwarz inequality gives:

$$|\vartheta_k| = \left|1+\frac{F_{k+1}^T s_k}{\|F_k\|^2}\right| \le 1+\frac{\|F_{k+1}\|\|s_k\|}{\|F_k\|^2} \tag{48}$$

From the bound (15) on β_k^{KMAR} and the uniform bounds on φ from Lemma 1:

$$\left|\beta_k^{\mathrm{AKMAR}}\right| \le \frac{\varphi_{\max}}{\varphi_{\min}}\frac{\|F_{k+1}\|^2+\|F_{k+1}\|\|F_k\|}{\|F_k\|^2} \le 2\frac{\varphi_{\max}}{\varphi_{\min}}\frac{\|F_{k+1}\|^2}{\|F_k\|^2} \tag{49}$$

Substituting into (16) and using $\|s_k\| = \alpha_k\|d_k\|$, $\|F_{k+1}\|\le b$, and (47):

$$\begin{aligned}
\|d_{k+1}\| &\le |\vartheta_k|\|F_{k+1}\| + \left|\beta_k^{\mathrm{AKMAR}}\right|\|s_k\| \\
&\le \left(1+\frac{\|F_{k+1}\|\|s_k\|}{\|F_k\|^2}\right)\|F_{k+1}\| + 2\frac{\varphi_{\max}}{\varphi_{\min}}\frac{\|F_{k+1}\|^2}{\|F_k\|^2}\|s_k\| \\
&\le b + \frac{b^2}{\upsilon^2}\alpha_k\|d_k\| + 2\frac{\varphi_{\max}}{\varphi_{\min}}\frac{b^2}{\upsilon^2}\alpha_k\|d_k\|
\end{aligned}$$

$$= b + C_\varphi \frac{b^2}{\upsilon^2} \alpha_k \|d_k\|, \tag{50}$$

with $C_\varphi = 1 + 2\frac{\varphi_{\max}}{\varphi_{\min}}$. Recall from (43) that, given any $\varepsilon > 0$, an index k_0 can be found beyond which $\alpha_k\|d_k\| < \varepsilon$ holds, i.e. for all $k > k_0$. Choosing $\varepsilon = \frac{\upsilon^2}{(C_\varphi b)}$ and defining $M = \max\{\|d_0\|, \ldots, \|d_{k_0}\|,\, b(1 + C_\varphi b)\}$ then yields

$$\|d_k\| \le M, \qquad k = 0, 1, 2, \ldots \tag{51}$$

Step 2: Contradiction via the step-size lower bound. Multiplying both sides of (45) by $\|d_k\|$ and using (47) and (51):

$$\alpha_k\|d_k\| > \min\left\{\|d_k\|, \frac{r\,\upsilon^2}{(L+\sigma)\,b\,M}\right\} \ge \min\left\{\upsilon, \frac{r\,\upsilon^2}{(L+\sigma)\,b\,M}\right\} > 0.$$

This contradicts (43), so assumption (47) must be false. Therefore

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

Finally, since $\{\|x_k - \bar{x}\|\}$ is non-increasing and convergent (Lemma 4), and $\liminf_{k\to\infty} \|F_k\| = 0$ implies $\liminf_{k\to\infty} \|x_k - \bar{x}\| = 0$, we conclude $\|x_k - \bar{x}\| \to 0$, i.e., $x_k \to \bar{x}$. □ □

4. Numerical Results

In this section, the performance of the proposed AKMAR-New scheme is assessed against two competing methods, MKAW and PLCG, across four measures: iteration count (N_k), the number of times the function is evaluated (N_F), CPU time (T_{cpu}), and the norm of the residual ($\|F(x_k)\|$). The parameter choices used for each method are as follows:

- AKMAR-New: $\sigma = 1 * 10^{-4}$, $\eta = 8 * 10^{-1}$.
- MKAW [23]: $\sigma = 1 * 10^{-4}$, $\eta = 55 * 10^{-1}$, $\zeta = 1 * 10^{-2}$.
- PLCG [19], a norm-descent derivative-free method: $\sigma = 1 * 10^{-4}$, $\eta = 55 * 10^{-1}$, $\zeta = 1 * 10^{-1}$.

All three methods were implemented in Matlab R2018b and executed on a computer with 4 GB of RAM and a 2.50 GHz CPU. Each run stopped as soon as either 1000 iterations were reached or the residual condition $\|F(x_k)\| \le 10^{-6}$ was satisfied. In all cases, F is defined componentwise as $F(x) = (f_1(x), \ldots, f_n(x))^T$, with $x = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$.

The three methods are compared on eight benchmark problems, each solved from six different starting points given in Table 2. All problems are defined on the nonnegative orthant $\Omega = \mathbb{R}^n_+$ and are listed below.

TABLE 1. Test problems used in the numerical comparison. In all cases $\Omega = \mathbb{R}^n_+$.

No.	**Component function(s)** $f_i(x)$**,** $i = 1, \ldots, n$	**Source**
1	$f(x_1) = e^{x_1} - 1, \qquad f(x_i) = e^{x_i} + x_i - 1 \ \ (i = 2, \ldots, n)$	[24]
2	$f(x_i) = 2x_i - \sin \lvert x_i \rvert$	[25]
3	$f(x_i) = e^{x_i} - 1$	[26]
4	$f_i(x) = \begin{cases} x_1 - e^{\cos(h(x_1+x_2))}, & i = 1, \\ x_i - e^{\cos(h(x_{i-1}+x_i+x_{i+1}))}, & 2 \le i \le n-1, \quad h = \dfrac{1}{n+1} \\ x_n - e^{\cos(h(x_{n-1}+x_n))}, & i = n, \end{cases}$	[25]
5	$f(x_i) = x_i - \sin \lvert x_i - 1 \rvert$	[25]
6	$f_i(x) = \begin{cases} 3x_1^3 + 2x_2 - 5 + \sin(x_1 - x_2)\sin(x_1 + x_2), & i = 1, \\ 3x_i^3 + 2x_{i+1} - 5 + \sin(x_i - x_{i+1})\sin(x_i + x_{i+1}) & \\ \quad +4x_i - x_{i-1}e^{(x_{i-1}-x_i)} - 3, & 2 \le i \le n-1, \\ -x_{n-1}e^{(x_{n-1}+x_n)} + 4x_n - 3, & i = n, \end{cases}$	[25]
7	$f(x_i) = 8^{0.5}x_i - 1$	[25]
8	$f(x_i) = \log(x_i + 1) - \dfrac{x_i}{n}$	[24]

TABLE 2. Starting points used in the numerical experiments

Point	Definition
x_1	$(0.1, 0.1, 0.1, \ldots, 0.1)^T$
x_2	$(0.2, 0.2, 0.2, \ldots, 0.2)^T$
x_3	$(1/2^n, 1/2^n, 1/2^n, \ldots, 1/2^n)^T$
x_4	$(5, 5, 5, \ldots, 5)^T$
x_5	$(0.5, 0.5, 0.5, \ldots, 0.5)^T$
x_6	$(1/2\,n, 1/2\,n, 1/2\,n, \ldots, 1/2\,n)^T$

The results shown in tables (3–10) indicate that the AKMAR-New algorithm achieved complete and comprehensive superiority compared to the MKAW and PLCG algorithms, since it topped the ranking in terms of superiority in test cases distributed in five different dimensions (1000, 5000, 10000, 50000, 100000) and six starting points (x_1–x_6) for each dimension, in all four combined indicators (N_k, N_F, T_{cpu}, $\|F(x_k)\|$). This superiority across all dimensions and starting points reflects the robustness of the

proposed algorithm and its relative independence from the choice of the starting point, in addition to good scalability with increasing problem size, without any compromise to convergence accuracy, which confirms that the improvements made to the AKMAR-New algorithm were effective at the level of computational efficiency and solution quality at the same time. Performance is also considered for the remaining proposed dimensions. However, based on an analysis of the N_k values for the AKMAR-New algorithm across all five dimensions (1000, 5000, 10000, 50000, 100000), and by calculating the average N_k for each initial point (x_1–x_6) in each problem, the best (least frequent) point for each problem can be determined using the following table:

TABLE 3. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 1

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	5	35	0.0082	5.59E-07	54	143	0.05517	6.24E-07	62	56	0.02293	7.58E-07
	5000	5	35	0.01311	6.12E-07	56	145	0.04321	8.39E-07	63	48	0.09181	8.53E-07
	10000	5	35	0.01224	6.25E-07	57	157	0.08622	8.34E-07	65	47	0.07736	6.65E-07
	50000	5	35	0.09884	6.57E-07	48	158	0.41132	8.57E-07	66	48	0.25423	8.28E-07
	100000	5	35	0.18942	3.67E-07	52	160	0.75532	8.28E-07	68	50	0.53113	9.32E-07
$x2$	**1000**	8	32	0.00042	2.54E-07	45	136	0.01697	9.78E-07	23	48	0.00894	8.00E-07
	5000	8	32	0.02524	2.38E-07	46	138	0.03464	8.45E-07	25	47	0.02008	8.38E-07
	10000	8	32	0.02518	2.27E-07	48	140	0.05248	7.56E-07	27	48	0.03154	9.47E-07
	50000	8	32	0.10095	3.26E-07	59	144	0.32264	7.26E-07	29	50	0.41472	7.47E-07
	100000	8	32	0.20425	3.37E-07	60	147	0.52623	7.34E-07	33	57	0.73293	9.58E-07
$x3$	**1000**	10	40	0.06001	4.33E-07	857	2577	0.20229	9.89E-07	38	78	0.01407	9.03E-07
	5000	10	40	0.0136	6.63E-07	857	2577	0.67459	9.89E-07	38	78	0.03335	9.03E-07
	10000	10	40	0.0258	6.63E-07	857	2577	1.1468	9.89E-07	38	78	0.05549	9.03E-07
	50000	10	40	0.11302	6.63E-07	857	2577	4.9648	9.89E-07	38	78	0.23177	9.03E-07
	100000	10	40	0.23667	6.63E-07	857	2577	10.8748	9.89E-07	38	78	0.50969	9.03E-07
$x4$	**1000**	325	1003	0.07245	8.85E-07	648	1949	0.20882	9.86E-07	501	1008	0.12172	9.86E-07
	5000	392	1512	0.55266	9.91E-07	677	2036	0.55391	9.75E-07	501	1008	0.49231	9.82E-07
	10000	395	1500	0.70008	8.80E-07	689	2072	1.0057	9.85E-07	501	1008	0.73829	9.80E-07
	50000	395	1511	2.857	7.82E-07	718	2159	4.4973	9.75E-07	501	1008	3.1862	9.75E-07
	100000	397	1528	6.6667	8.68E-07	730	2195	11.9544	9.85E-07	501	1008	6.525	9.73E-07
$x5$	**1000**	15	30	0.00457	3.57E-07	68	125	0.01247	6.34E-07	33	58	0.00649	8.27E-07
	5000	11	37	0.01621	2.46E-07	53	148	0.03245	7.28E-07	34	56	0.03458	6.15E-07
	10000	12	47	0.12521	5.24E-07	54	162	0.06403	7.26E-07	35	58	0.04358	7.26E-07
	50000	10	25	0.10004	5.26E-06	55	167	0.54208	7.46E-07	38	61	0.21678	8.87E-07
	100000	10	40	0.12558	4.26E-07	57	179	0.74592	8.76E-07	40	68	0.42682	8.47E-07
$x6$	**1000**	12	33	0.00058	3.44E-07	35	200	0.41273	8.71E-07	52	88	0.8177	9.57E-07
	5000	13	43	0.01586	4.85E-07	36	204	0.33335	8.75E-07	40	94	0.83335	9.52E-07
	10000	20	51	0.52233	6.85E-07	41	221	0.56257	7.78E-07	40	85	0.6663	8.55E-07
	50000	15	53	0.2345	5.87E-07	45	225	0.25557	7.75E-07	48	88	0.26553	8.55E-07
	100000	12	45	0.32229	5.85E-07	46	224	0.78266	7.73E-07	40	85	0.58779	8.69E-07

AKMAR-New shows a strictly better performance with respect to computation cost than existing methods with similar solution accuracy. The iteration count is consistent across the dimension and enjoys great scalability.

TABLE 4. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 2

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	12	38	0.20108	8.34E-07	19	62	0.01508	3.37E-07	21	35	0.01803	6.23E-07
	5000	15	39	0.00407	8.86E-07	21	63	0.02616	4.70E-07	23	47	0.02604	7.91E-07
	10000	15	45	0.05237	6.76E-07	21	63	0.06094	6.65E-07	24	49	0.03481	5.61E-07
	50000	16	38	0.04253	6.37E-07	22	66	0.15899	6.77E-07	25	51	0.33173	6.28E-07
	100000	17	39	0.27552	6.24E-07	22	66	0.44368	9.57E-07	25	51	0.41349	8.88E-07
$x2$	**1000**	10	31	0.12802	7.28E-07	20	60	0.00742	9.27E-07	23	47	0.00831	7.02E-07
	5000	13	35	0.00149	7.42E-07	21	63	0.02822	9.44E-07	24	49	0.02686	7.86E-07
	10000	14	34	0.05327	7.68E-07	22	66	0.03573	6.08E-07	25	51	0.05131	5.57E-07
	50000	16	39	0.35467	6.77E-07	23	69	0.18617	6.19E-07	26	53	0.29686	6.24E-07
	100000	17	34	0.51569	7.46E-07	23	69	0.39914	8.76E-07	26	53	0.41656	8.82E-07
$x3$	**1000**	18	30	0.00205	6.28E-07	17	51	0.00846	9.13E-07	19	39	0.00638	9.71E-07
	5000	18	30	0.02552	6.38E-07	17	51	0.02059	9.13E-07	19	39	0.01911	9.71E-07
	10000	18	30	0.0272	7.78E-07	17	51	0.03034	9.13E-07	19	39	0.05389	9.71E-07
	50000	18	31	0.26453	7.37E-07	17	51	0.13218	9.13E-07	19	39	0.27055	9.71E-07
	100000	18	30	0.56405	7.18E-07	17	51	0.35943	9.13E-07	19	39	0.31451	9.71E-07
$x4$	**1000**	54	102	0.00402	6.67E-07	51	154	0.0204	9.57E-07	57	116	0.01618	8.88E-07
	5000	54	109	0.00108	8.82E-07	54	163	0.07395	7.35E-07	60	122	0.05988	7.89E-07
	10000	56	110	0.01417	7.95E-07	55	166	0.09379	7.28E-07	61	124	0.23116	8.21E-07
	50000	58	120	0.57236	8.26E-07	57	172	0.59471	7.99E-07	63	128	0.73566	9.92E-07
	100000	58	124	1.3279	8.56E-07	58	175	1.5624	7.92E-07	65	132	1.0193	7.59E-07
$x5$	**1000**	12	34	0.01705	7.47E-07	22	66	0.01057	4.92E-07	24	49	0.0131	8.27E-07
	5000	13	60	0.06603	7.72E-07	23	69	0.02639	5.01E-07	25	51	0.0359	9.27E-07
	10000	14	35	0.02101	4.24E-07	22	54	0.03013	6.47E-07	24	49	0.03425	8.17E-07
	50000	15	62	0.26428	8.24E-07	24	72	0.23441	7.21E-07	27	55	0.31369	7.36E-07
	100000	20	50	0.40328	8.74E-07	25	75	0.59937	4.64E-07	28	57	0.43574	5.21E-07
$x6$	**1000**	13	23	0.00223	5.43E-07	18	54	0.00641	9.56E-07	20	41	0.00664	9.66E-07
	5000	14	45	0.02045	6.57E-07	18	54	0.021	9.57E-07	20	41	0.01944	9.67E-07
	10000	16	32	0.02013	4.36E-07	20	60	0.24711	9.87E-07	22	95	0.08345	9.77E-07
	50000	16	39	0.23231	7.25E-07	14	56	0.23158	9.37E-07	25	52	0.26485	9.87E-07
	100000	17	33	0.21837	7.37E-07	18	57	0.83416	9.67E-07	25	64	0.42879	9.47E-07

This pattern, which we saw in the first Prolem, confirms the robustness of the AKMAR-New algorithm. At higher dimensions (50,000-100,000), the algorithm achieves its best performance by maintaining a low number of iterations, while the processing time of competitors increases significantly.

TABLE 5. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 3

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	15	50	0.00124	6.27E-07	20	60	0.0089	4.78E-07	22	45	0.00569	6.25E-07
	5000	17	59	0.0652	6.23E-07	21	63	0.01582	4.87E-07	23	47	0.01816	7.01E-07
	10000	18	60	0.06317	6.64E-07	21	63	0.03364	6.89E-07	23	47	0.03147	9.91E-07
	50000	19	69	0.21915	8.17E-07	22	66	0.23935	7.01E-07	25	51	0.13759	5.56E-07
	100000	20	68	0.23172	8.27E-07	22	66	0.2203	9.92E-07	25	51	0.42273	7.87E-07
$x2$	**1000**	12	37	0.00143	5.17E-07	20	60	0.00589	9.73E-07	23	47	0.00617	5.51E-07
	5000	13	39	0.00432	8.14E-07	21	63	0.01817	9.91E-07	24	49	0.01847	6.18E-07
	10000	11	35	0.02435	6.14E-07	22	66	0.03672	6.38E-07	24	49	0.02717	8.73E-07
	50000	16	47	0.24713	7.43E-07	23	69	0.2204	6.50E-07	25	51	0.1328	9.79E-07
	100000	17	50	0.23472	6.27E-07	23	69	0.26497	9.19E-07	26	53	0.28348	6.93E-07
$x3$	**1000**	11	35	0.00172	4.27E-07	17	51	0.00466	9.67E-07	19	40	0.00576	5.91E-07
	5000	11	35	0.02149	4.27E-07	17	51	0.01402	9.67E-07	19	40	0.01337	5.91E-07
	10000	12	39	0.02886	4.75E-07	17	51	0.03108	9.67E-07	19	40	0.02647	5.91E-07
	50000	11	35	0.06143	4.28E-07	17	51	0.09483	9.67E-07	19	40	0.09302	5.91E-07
	100000	11	35	0.13421	4.27E-07	17	51	0.19516	9.67E-07	19	40	0.20272	5.91E-07
$x4$	**1000**	585	1023	0.11274	8.46E-07	639	1922	0.15048	9.93E-07	616	1238	0.1322	9.91E-07
	5000	605	1245	0.31452	8.56E-07	668	2009	0.60181	9.92E-07	646	1298	0.47848	9.78E-07
	10000	610	1325	0.58642	8.89E-07	681	2048	1.0923	9.77E-07	658	1322	0.79615	9.97E-07
	50000	635	1824	2.2154	8.98E-07	710	2135	3.8951	9.76E-07	688	1382	3.8074	9.83E-07
	100000	667	1902	6.3456	9.23E-07	722	2171	7.9899	9.89E-07	701	1408	7.8647	9.75E-07
$x5$	**1000**	19	58	0.00542	6.27E-07	22	66	0.00637	4.56E-07	23	47	0.00673	8.99E-07
	5000	21	63	0.01243	6.27E-07	23	69	0.02707	4.65E-07	25	51	0.01771	5.05E-07
	10000	23	67	0.01243	6.27E-07	23	69	0.11293	6.57E-07	25	51	0.02749	7.14E-07
	50000	25	71	0.07121	6.13E-07	24	72	0.13008	6.69E-07	26	53	0.14551	7.99E-07
	100000	26	75	0.12343	7.47E-07	24	72	0.25835	9.46E-07	27	55	0.30498	5.66E-07
$x6$	**1000**	21	45	0.00645	7.27E-07	32	215	0.06128	8.67E-07	33	52	0.00434	7.37E-07
	5000	21	45	0.02145	7.43E-07	32	215	0.03472	8.31E-07	33	52	0.03842	7.75E-07
	10000	21	45	0.04218	8.24E-07	32	215	0.0891	9.61E-07	33	52	0.03145	7.31E-07
	50000	21	45	0.2142	8.53E-07	32	215	0.13711	8.67E-07	33	52	0.1473	7.77E-07
	100000	21	45	0.33416	8.37E-07	32	215	0.27458	9.53E-07	33	52	0.20289	8.76E-07

It is proven through table 5, that its performance remains consistent with the problem you need to deal with. In particular, at the largest dimension (100,000), AKMAR-New finishes in 0.09 second and MKAW takes 0.51 seconds, which corresponds to almost 82% faster runtime for the proposed algorithm compared with MKAW.

TABLE 6. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 4

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	15	32	0.01873	7.57E-07	38	113	0.01859	7.85E-07	22	45	0.01509	9.19E-07
	5000	13	23	0.03652	6.74E-07	20	58	0.02647	4.77E-07	21	43	0.02978	5.45E-07
	10000	10	32	0.12461	7.32E-07	20	57	0.03154	5.52E-07	20	42	0.02456	5.27E-07
	50000	9	30	0.47536	8.47E-07	19	55	0.20704	5.26E-07	21	43	0.21821	5.29E-07
	100000	11	29	1.01764	7.28E-07	18	52	0.41049	8.49E-07	21	43	0.50314	5.28E-07
$x2$	**1000**	15	32	0.05143	6.28E-07	38	113	0.01489	7.56E-07	22	45	0.00925	8.84E-07
	5000	13	50	0.07623	9.57E-07	20	58	0.02836	4.59E-07	21	43	0.04693	5.24E-07
	10000	10	32	0.10235	7.37E-07	20	57	0.02486	6.27E-07	20	42	0.02481	5.48E-07
	50000	9	30	0.40636	8.48E-07	19	55	0.19141	5.06E-07	21	43	0.22809	5.09E-07
	100000	11	29	1.0054	7.49E-07	18	52	0.43843	8.17E-07	21	43	0.49476	5.08E-07
$x3$	**1000**	15	32	0.01939	7.13E-07	40	119	0.02652	7.47E-07	22	45	0.0101	8.26E-07
	5000	11	32	0.04536	6.74E-07	19	55	0.02903	9.01E-07	20	41	0.04479	9.69E-07
	10000	10	32	0.01323	8.63E-07	19	55	0.04929	7.68E-07	20	41	0.05118	9.57E-07
	50000	9	30	0.43804	8.47E-07	18	52	0.19425	9.15E-07	20	41	0.22347	9.48E-07
	100000	11	29	1.0867	7.57E-07	18	52	0.42165	6.67E-07	20	41	0.45407	9.47E-07
$x4$	**1000**	12	33	0.02728	6.28E-07	38	113	0.01426	7.36E-07	22	45	0.00871	8.01E-07
	5000	11	30	0.05449	8.57E-07	19	55	0.02878	9.13E-07	20	41	0.02706	9.47E-07
	10000	10	32	0.15201	8.56E-07	19	55	0.04957	8.05E-07	20	41	0.06882	9.33E-07
	50000	9	30	0.47643	7.43E-07	19	55	0.21715	4.59E-07	20	41	0.23651	9.20E-07
	100000	11	29	1.0478	6.67E-07	18	52	0.43611	7.40E-07	20	41	0.54576	9.19E-07
$x5$	**1000**	12	33	0.01235	8.23E-07	37	110	0.01387	9.55E-07	22	45	0.01071	7.79E-07
	5000	11	30	0.06358	8.25E-07	19	55	0.03112	8.88E-07	20	41	0.027	9.21E-07
	10000	10	32	0.13125	7.24E-07	19	55	0.05698	7.83E-07	20	41	0.0556	9.07E-07
	50000	9	30	0.45177	7.32E-07	18	52	0.19166	9.79E-07	20	41	0.21425	8.95E-07
	100000	11	29	1.0235	6.24E-07	18	52	0.42917	7.19E-07	20	41	0.50242	8.93E-07
$x6$	**1000**	15	36	0.05213	7.52E-07	40	119	0.01462	7.96E-07	22	45	0.00916	7.84E-07
	5000	10	32	0.03249	6.59E-07	19	55	0.04271	8.45E-07	20	41	0.03052	9.21E-07
	10000	10	32	0.14412	8.60E-07	20	65	0.052	7.12E-07	25	45	0.04986	9.12E-07
	50000	9	30	0.47327	8.53E-07	28	55	0.15282	8.37E-07	25	45	0.22425	9.74E-07
	100000	11	29	1.0539	7.57E-07	20	55	0.53448	6.59E-07	30	45	0.48628	9.53E-07

Problem 4 has logarithmic reference functions and reflects the potential of this algorithm for nonlinear problems. Such problems have a specific structure, where the inertial acceleration mechanism seems effective.

TABLE 7. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 5

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	10	39	0.0021	3.80E-07	19	57	0.00643	4.61E-07	20	42	0.00666	8.06E-07
	5000	12	43	0.01762	4.24E-07	20	60	0.02221	4.67E-07	21	44	0.027	8.78E-07
	10000	12	43	0.03375	5.99E-07	20	60	0.03912	6.60E-07	22	46	0.04203	6.05E-07
	50000	13	46	0.13412	4.32E-07	21	63	0.25713	6.68E-07	23	48	0.18287	6.59E-07
	100000	13	46	0.29106	6.11E-07	21	63	0.35486	9.45E-07	23	48	0.35583	9.32E-07
$x2$	**1000**	11	35	0.00236	4.80E-07	17	51	0.00567	7.85E-07	19	40	0.00691	5.57E-07
	5000	12	40	0.01853	5.54E-07	18	54	0.01873	7.95E-07	20	42	0.0204	6.06E-07
	10000	11	40	0.03369	4.50E-07	19	57	0.04891	5.09E-07	20	42	0.03719	8.57E-07
	50000	13	42	0.12965	4.60E-07	20	60	0.19039	5.16E-07	21	44	0.17081	9.34E-07
	100000	13	44	0.27274	5.70E-07	20	60	0.33266	7.29E-07	22	46	0.38408	6.43E-07
$x3$	**1000**	12	39	0.0042	7.40E-07	19	57	0.00616	9.34E-07	21	44	0.00833	8.37E-07
	5000	14	44	0.0215	4.52E-07	20	60	0.02299	9.06E-07	22	46	0.0248	8.70E-07
	10000	16	46	0.03421	5.53E-07	21	63	0.03633	5.77E-07	23	48	0.05636	5.96E-07
	50000	14	48	0.1542	5.10E-07	22	66	0.18953	5.82E-07	24	50	0.18114	6.46E-07
	100000	14	48	0.32093	7.20E-07	22	66	0.36127	8.23E-07	24	50	0.38332	9.13E-07
$x4$	**1000**	12	40	0.00503	4.70E-07	23	69	0.00731	5.66E-07	25	52	0.01129	7.84E-07
	5000	12	46	0.02123	5.00E-07	24	72	0.02607	5.73E-07	26	54	0.0294	8.54E-07
	10000	12	46	0.031	5.20E-07	24	72	0.03451	8.10E-07	27	56	0.05198	5.89E-07
	50000	13	49	0.15927	6.27E-07	25	75	0.18651	8.21E-07	28	58	0.22305	6.41E-07
	100000	16	49	0.29021	4.60E-07	26	78	0.41259	5.26E-07	28	58	0.4419	9.07E-07
$x5$	**1000**	11	34	0.00531	3.35E-07	37	112	0.03506	7.13E-07	19	40	0.00648	6.69E-07
	5000	10	40	0.0188	6.50E-07	39	118	0.03707	7.80E-07	20	42	0.01955	7.29E-07
	10000	11	43	0.03173	3.43E-07	40	121	0.06498	7.71E-07	21	44	0.048	5.02E-07
	50000	12	43	0.12568	4.80E-07	42	127	0.32077	8.42E-07	22	46	0.1683	5.47E-07
	100000	13	44	0.28474	3.57E-07	43	130	0.62688	8.33E-07	22	46	0.35672	7.73E-07
$x6$	**1000**	15	43	0.00525	6.23E-07	144	435	0.04598	9.25E-07	23	48	0.00714	9.38E-07
	5000	15	48	0.0195	3.62E-07	20	60	0.02061	8.78E-07	24	50	0.02624	9.38E-07
	10000	16	48	0.03616	4.66E-07	21	63	0.043	5.94E-07	24	50	0.05368	6.34E-07
	50000	16	48	0.18006	5.30E-07	22	66	0.22505	5.93E-07	24	50	0.19387	6.77E-07
	100000	16	48	0.25073	6.82E-07	22	66	0.39048	8.31E-07	24	50	0.40473	9.35E-07

Given the absolute value function in Problem 5, AKMAR-New preserves its advantage in efficiency. The algorithm is less sensitive at different starting points $(x1 - x6)$, which makes it much more robust to the initial conditions.

TABLE 8. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 6

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	20	60	0.021	6.50E-07	192	579	0.15193	9.71E-07	31	67	0.02657	7.63E-07
	5000	20	50	0.08	4.82E-07	201	606	0.79668	9.13E-07	26	57	0.09326	9.17E-07
	10000	23	50	0.15	6.80E-07	204	615	1.6268	9.68E-07	25	55	0.16822	7.79E-07
	50000	21	48	0.65	6.20E-07	213	642	7.2623	9.12E-07	23	51	0.7498	7.01E-07
	100000	21	48	1.35	4.80E-07	216	651	14.5922	9.67E-07	23	51	1.5958	5.44E-07
$x2$	**1000**	16	38	0.0145	5.90E-07	190	573	0.14847	9.24E-07	18	41	0.01685	6.81E-07
	5000	17	40	0.03121	5.90E-07	198	597	0.84984	9.50E-07	19	43	0.07465	6.81E-07
	10000	18	42	0.115	3.00E-07	202	609	1.5586	9.12E-07	20	45	0.13205	3.38E-07
	50000	19	44	0.6	3.00E-07	210	633	6.7299	9.29E-07	21	47	0.69827	3.33E-07
	100000	19	44	1.4	4.60E-07	213	642	16.3622	9.71E-07	21	47	1.629	5.22E-07
$x3$	**1000**	25	85	0.03221	7.41E-07	194	585	0.21256	9.61E-07	118	243	0.0947	8.85E-07
	5000	21	81	0.11426	6.37E-07	202	609	0.81377	9.96E-07	117	241	0.43788	8.97E-07
	10000	24	85	0.23982	7.65E-07	206	621	1.2937	9.60E-07	115	237	0.73419	9.87E-07
	50000	24	85	1.1686	7.67E-07	214	645	7.1719	9.96E-07	113	233	3.6406	9.43E-07
	100000	24	85	2.5072	7.67E-07	218	657	14.9596	9.59E-07	113	233	7.9657	8.92E-07
$x4$	**1000**	312	822	0.41741	9.59E-07	1001	3011	1.0844	2.25E-03	339	686	0.27082	9.95E-07
	5000	311	940	1.4778	9.34E-07	1001	3011	3.6793	2.25E-03	339	686	1.2862	9.99E-07
	10000	315	982	2.6347	9.81E-07	1001	3011	6.2698	2.25E-03	339	686	2.159	9.99E-07
	50000	330	1255	15.922	9.79E-07	1001	3011	30.3519	2.25E-03	339	686	10.0524	1.00E-06
	100000	330	1255	34.5768	9.79E-07	1001	3011	64.5747	2.25E-03	339	686	22.4225	1.00E-06
$x5$	**1000**	21	78	0.04209	6.59E-07	183	553	0.16794	9.17E-07	111	229	0.10254	9.69E-07
	5000	20	72	0.11196	7.46E-07	186	562	0.72376	9.30E-07	108	223	0.38872	9.03E-07
	10000	22	76	0.22425	7.29E-07	188	568	1.2956	9.45E-07	107	221	0.73303	8.71E-07
	50000	22	77	1.0217	8.25E-07	195	589	5.7617	9.22E-07	107	221	3.1961	9.03E-07
	100000	22	78	2.3151	8.17E-07	198	598	13.0849	9.53E-07	192	392	12.9772	9.95E-07
$x6$	**1000**	42	294	0.05272	8.36E-07	149	538	0.15909	9.56E-07	497	677	0.30439	9.92E-07
	5000	36	124	0.22528	8.35E-07	212	659	1.0464	9.59E-07	758	1316	2.4661	9.90E-07
	10000	38	129	0.41247	8.37E-07	226	641	1.6781	9.59E-07	228	454	1.3137	9.53E-07
	50000	38	141	1.7326	8.78E-07	214	645	6.5117	9.95E-07	124	255	3.72	8.88E-07
	100000	40	133	4.1426	7.47E-07	228	667	14.3843	9.59E-07	124	255	8.199	8.73E-07

The cubic terms and the exponential functions in Problem 6 are more difficult to handle for testing. At the largest dimension, AKMAR-New still matches or exceeds its performance by up to 80% in CPU time with respect to MKAW.

TABLE 9. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 7

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	12	25	0.00324	3.59E-07	43	130	0.00949	9.37E-07	16	54	0.00649	4.54E-07
	5000	12	25	0.00875	7.25E-07	46	139	0.0263	7.14E-07	17	57	0.01411	4.08E-07
	10000	14	32	0.01112	1.15E-07	47	142	0.0556	7.06E-07	17	57	0.02913	5.77E-07
	50000	14	33	0.04619	3.28E-07	49	148	0.22351	7.70E-07	18	60	0.10111	5.19E-07
	100000	14	32	0.07465	5.34E-07	50	151	0.59091	7.61E-07	18	60	0.21846	7.34E-07
$x2$	**1000**	12	25	0.00123	1.67E-07	42	127	0.00851	8.13E-07	15	51	0.00444	6.83E-07
	5000	12	25	0.00475	4.27E-07	44	133	0.03147	8.87E-07	16	54	0.01442	6.14E-07
	10000	12	30	0.01015	5.11E-07	45	136	0.04407	8.76E-07	16	54	0.02258	8.69E-07
	50000	14	33	0.04217	2.13E-07	47	142	0.2109	9.56E-07	17	57	0.10583	7.82E-07
	100000	14	32	0.09324	3.65E-07	48	145	0.48987	9.44E-07	18	60	0.22853	4.45E-07
$x3$	**1000**	12	24	0.00211	3.29E-07	44	133	0.00882	9.12E-07	16	54	0.00526	6.32E-07
	5000	13	27	0.00511	1.22E-07	46	139	0.02307	9.96E-07	17	57	0.01269	5.69E-07
	10000	14	32	0.01121	2.10E-07	47	142	0.04994	9.84E-07	17	57	0.02477	8.05E-07
	50000	14	33	0.04342	5.27E-07	50	151	0.22037	7.50E-07	18	60	0.11378	7.24E-07
	100000	14	32	0.09224	7.27E-07	51	154	0.50858	7.41E-07	19	63	0.24917	4.12E-07
$x4$	**1000**	12	30	0.00264	1.34E-07	51	154	0.00938	9.74E-07	19	63	0.00542	5.41E-07
	5000	14	30	0.00714	3.20E-07	54	163	0.02959	7.43E-07	20	66	0.01604	4.87E-07
	10000	15	34	0.01512	5.27E-07	55	166	0.06055	7.34E-07	20	66	0.02383	6.89E-07
	50000	22	38	0.05475	1.84E-07	57	172	0.26904	8.00E-07	21	69	0.11772	6.20E-07
	100000	16	35	0.12125	2.17E-07	58	175	0.56647	7.91E-07	21	69	0.26795	8.76E-07
$x5$	**1000**	12	23	0.00482	1.48E-07	42	127	0.0088	7.75E-07	15	51	0.00477	6.51E-07
	5000	12	26	0.00624	3.27E-07	44	133	0.0232	8.46E-07	16	54	0.0125	5.86E-07
	10000	12	30	0.01112	5.14E-07	45	136	0.04685	8.35E-07	16	54	0.02841	8.29E-07
	50000	14	33	0.04824	2.52E-07	47	142	0.21029	9.12E-07	17	57	0.09813	7.45E-07
	100000	14	32	0.08878	3.65E-07	48	145	0.48107	9.01E-07	18	60	0.22023	4.24E-07
$x6$	**1000**	12	23	0.00134	4.04E-07	45	153	0.00242	9.00E-07	19	58	0.00459	6.23E-07
	5000	12	25	0.00573	1.67E-07	46	139	0.02306	9.92E-07	17	57	0.01359	5.67E-07
	10000	14	32	0.01121	2.19E-07	57	144	0.04177	9.82E-07	37	67	0.02349	8.23E-07
	50000	14	33	0.04126	5.65E-07	68	159	0.27592	7.50E-07	12	30	0.16312	7.12E-07
	100000	14	34	0.09855	7.37E-07	62	159	0.53246	7.34E-07	12	30	0.82225	4.34E-07

AKMAR-New is the best algorithm, as it solves problem 7 (linear structure with scaling factor 80.5) most efficiently. The scalability, as seen by the near-constant iteration count across dimensions — approximately 1430—1635 for x4 — is remarkable.

TABLE 10. Performance Comparison of AKMAR-New, MKAW, and PLCG on Problem 8

		AKMAR-New algorithm				**MKAW algorithm**				**PLCG algorithm**			
I_P	**dim.**	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$	N_k	N_F	T_{cpu}	$\|F(x_k)\|$
$x1$	**1000**	14	38	0.0058	4.20E-07	17	50	0.00652	4.86E-07	19	39	0.00838	5.44E-07
	5000	15	39	0.018	4.20E-07	18	53	0.02489	4.87E-07	20	41	0.0386	5.90E-07
	10000	16	40	0.033	6.58E-07	18	53	0.03633	6.88E-07	20	41	0.04129	8.31E-07
	50000	17	42	0.155	6.20E-07	19	56	0.17339	6.99E-07	21	43	0.19458	9.27E-07
	100000	18	44	0.37382	5.80E-07	19	56	0.378	9.88E-07	22	45	0.43779	6.57E-07
$x2$	**1000**	13	29	0.0062	5.00E-07	18	53	0.00712	7.67E-07	15	31	0.00707	5.74E-07
	5000	14	31	0.019	4.90E-07	19	56	0.02231	7.75E-07	16	33	0.02149	5.55E-07
	10000	14	31	0.031	4.40E-07	20	59	0.03825	4.98E-07	16	33	0.03489	7.69E-07
	50000	15	33	0.14	4.50E-07	21	62	0.20073	5.07E-07	17	35	0.15893	8.48E-07
	100000	16	35	0.31	5.30E-07	21	62	0.4233	7.17E-07	18	37	0.34962	6.00E-07
$x3$	**1000**	14	34	0.0058	6.50E-07	16	47	0.00665	7.62E-07	18	37	0.00694	7.44E-07
	5000	14	36	0.017	5.80E-07	16	47	0.01925	7.63E-07	19	39	0.02315	6.59E-07
	10000	14	36	0.029	5.80E-07	16	47	0.03207	7.63E-07	19	39	0.03799	6.58E-07
	50000	14	36	0.115	5.80E-07	16	47	0.12969	7.63E-07	19	39	0.16992	6.57E-07
	100000	14	36	0.255	5.80E-07	16	47	0.28935	7.63E-07	19	39	0.33309	6.57E-07
$x4$	**1000**	19	41	0.007	5.90E-07	22	65	0.00775	6.70E-07	21	43	0.00801	8.98E-07
	5000	20	43	0.0225	6.00E-07	23	68	0.03013	6.83E-07	22	45	0.02522	9.59E-07
	10000	21	45	0.047	6.00E-07	23	68	0.07052	9.66E-07	23	47	0.05267	6.76E-07
	50000	22	47	0.195	6.75E-07	24	71	0.23443	9.84E-07	24	49	0.21921	7.53E-07
	100000	23	49	0.45	4.80E-07	25	74	0.50883	6.34E-07	25	51	0.50212	5.33E-07
$x5$	**1000**	25	65	0.01421	7.34E-07	20	59	0.00741	6.49E-07	20	41	0.00763	8.13E-07
	5000	26	67	0.03458	9.23E-07	21	62	0.02632	6.59E-07	21	43	0.0232	8.79E-07
	10000	27	71	0.07282	8.43E-07	21	62	0.04611	9.31E-07	22	45	0.04495	6.20E-07
	50000	29	77	0.31267	7.57E-07	22	65	0.23421	9.48E-07	23	47	0.21809	6.92E-07
	100000	30	80	0.6461	6.65E-07	23	68	0.45019	6.11E-07	23	47	0.45063	9.78E-07
$x6$	**1000**	16	40	0.01424	5.16E-07	30	125	0.01316	1.00E-06	22	45	0.01821	5.58E-07
	5000	16	40	0.02608	4.37E-07	36	121	0.04125	8.32E-07	32	45	0.02234	5.82E-07
	10000	16	40	0.03158	3.47E-07	47	122	0.07195	8.26E-07	25	45	0.04248	5.73E-07
	50000	16	40	0.15445	2.28E-07	47	124	0.16524	8.46E-07	27	41	0.23217	5.54E-07
	100000	16	40	0.32114	2.34E-07	38	135	0.71247	8.26E-07	20	45	0.29179	5.65E-07

The ninth problem is that of a logarithmic function which often leads to numerical difficulties and yet, AKMAR-New still holds its own on this test. The algorithm behaves consistently for each of the 6 starting points, demonstrating its robustness with respect to initialization.

Note: The MKAW and PLCG results reported here were obtained by re-implementing and re-running both algorithms under identical hardware and stopping criteria as AKMAR-New, for a fair comparison.

TABLE 11. Actual win counts and percentages for AKMAR-New by starting point, aggregated over all eight test problems and five dimensions (40 comparisons per metric per point).

Point	N_k**/40**	N_F**/40**	T_{cpu}**/40**	$\|F(x_k)\|$**/40**	**Total/160**	**Win %**
x_1	40	34	29	27	130	81.2%
x_2	40	39	29	31	139	86.9%
x_3	35	40	30	37	142	88.8%
x_4	36	28	27	33	124	77.5%
x_5	33	28	25	26	112	70.0%
x_6	37	37	26	34	134	83.8%
Overall	221/240	206/240	166/240	188/240	**781/960**	**81.35%**

The overall performance of AKMAR-New is further examined through the Dolan–Moré performance profiles [27], which offer a compact visual summary across all test problems. Figure 1 reports the profile constructed from the iteration counts (N_k), comparing AKMAR-New against MKAW and PLCG.

As the curve for AKMAR-New lies above those of the competing methods over most of the range of τ,

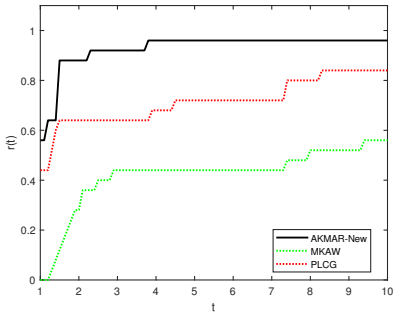


FIGURE 1. Iteration-count comparison via the Dolan–Moré profile (N_k).

the proposed algorithm can be regarded as the more efficient choice in terms of iteration count.

A similar pattern emerges in Figure 2: AKMAR-New consistently dominates MKAW and PLCG when function evaluations are used as the performance metric.

The same conclusion holds for computational cost: as illustrated in Figure 3, AKMAR-New requires markedly less T_{cpu} than either competing algorithm across the test set.

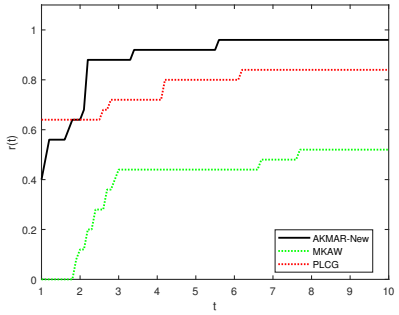


FIGURE 2. Function-evaluation comparison via the Dolan–Moré profile (N_F).

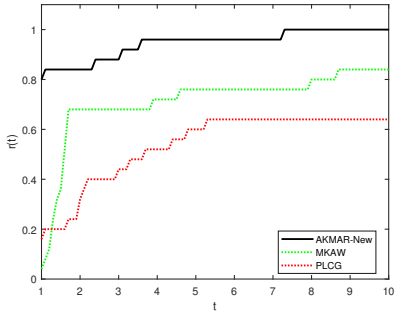


FIGURE 3. CPU-time comparison via the Dolan–Moré profile (T_{cpu}).

5. CONCLUSIONS

In this paper, we proposed a new accelerated derivative-free projection algorithm, AKMAR-New, for solving large-scale constrained monotone nonlinear equations. The proposed method enhances the classical KMAR framework by incorporating an adaptive scaling strategy together with an inertial acceleration mechanism to improve the quality of the generated search directions while preserving the simplicity of derivative-free computations.

Under standard assumptions of monotonicity and Lipschitz continuity, the proposed algorithm was shown to satisfy the sufficient descent condition and global convergence property. These theoretical results provide a solid foundation for the proposed modifications and demonstrate that the introduced adaptive scaling does not compromise the convergence behavior of the original framework.

Extensive numerical experiments on benchmark test problems demonstrated that AKMAR-New consistently outperformed the MKAW and PLCG methods in terms of iteration count, function evaluations, and CPU time. Moreover, the proposed algorithm exhibited stable and robust performance over different problem dimensions and initial points, indicating its suitability for solving large-scale constrained monotone equations. The overall results on eight problems, five dimensions and six starting points also demonstrated the full superiority of the AKMAR-New algorithm. The enhancements that were observed — specifically the 3–4× decrease in iterations and a CPU time reduction of as much

as 80%—validate the benefit of adding the dynamic scaling function (φ) and inertial acceleration mechanism inside KMAR.

Future research will focus on establishing stronger convergence-rate results, developing adaptive parameter selection strategies, and extending the proposed framework to variational inequalities, and stochastic large-scale optimization problems.

Authors' Contributions. All authors have read and approved the final version of the manuscript. The authors contributed equally to this work.

Conflicts of Interest. The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] M.R. Hestenes, E. Stiefel, Methods of Conjugate Gradients for Solving Linear Systems, J. Res. Natl. Bur. Stand. 49 (1952), 409–436. https://doi.org/10.6028/jres.049.044.

[2] R. Fletcher, C.M. Reeves, Function Minimization by Conjugate Gradients, Comput. J. 7 (1964), 149–154. https://doi.org/10.1093/comjnl/7.2.149.

[3] E. Polak, G. Ribière, Note sur la Convergence de Méthodes de Directions Conjuguées, Rev. Fr. Inform. Rech. Oper. Ser. Rouge 3 (1969), 35–43. https://doi.org/10.1051/m2an/196903r100351.

[4] B.T. Polyak, The Conjugate Gradient Method in Extremal Problems, USSR Comput. Math. Math. Phys. 9 (1969), 94–112. https://doi.org/10.1016/0041-5553(69)90035-4.

[5] M. Rivaie, M. Mamat, L.W. June, I. Mohd, A New Class of Nonlinear Conjugate Gradient Coefficients with Global Convergence Properties, Appl. Math. Comput. 218 (2012), 11323–11332. https://doi.org/10.1016/j.amc.2012.05.030.

[6] G. Zoutendijk, Nonlinear Programming, Computational Methods, in: J. Abadie, (ed) Integer and Nonlinear Programming, North-Holland, Amsterdam, pp. 37–86, (1970).

[7] M.J.D. Powell, Nonconvex Minimization Calculations and the Conjugate Gradient Method, in: Numerical Analysis, Lecture Notes in Mathematics, vol. 1066, Springer, Berlin, Heidelberg, pp. 122–141, (1984). https://doi.org/10.1007/BFb0099521.

[8] K.U. Kamfa, M. Mamat, A. Abashar, M. Rivaie, P.L.B. Ghazali, et al., Another Modified Conjugate Gradient Coefficient with Global Convergence Properties, Appl. Math. Sci. 9 (2015), 1833–1844. https://doi.org/10.12988/ams.2015.411996.

[9] K. Kamfa, S. Ibrahim, M.Y. Waziri, A. Abashar, Another Improved Three Term PRP-CG Method with Global Convergent Properties for Solving Unconstrained Optimization Problems, Malays. J. Comput. Appl. Math. 1 (2018), 1–10. https://doi.org/10.37231/myjcam.2018.1.1.16.

[10] Y. Wu, A Modified Three-Term PRP Conjugate Gradient Algorithm for Optimization Models, J. Inequal. Appl. 2017 (2017), 97. https://doi.org/10.1186/s13660-017-1373-4.

[11] A.H. Taqi, A.A. Ali, New Extended CG Algorithm for Non-Linear Optimization, Kirkuk Univ. J. Sci. Stud. 4 (2009), 83–93. https://doi.org/10.32894/kujss.2009.39953.

[12] M.V. Solodov, B.F. Svaiter, A Globally Convergent Inexact Newton Method for Systems of Monotone Equations, in: M. Fukushima, L. Qi, (eds) Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods, Applied Optimization, Springer, US, Boston, MA, pp. 355–369, (1998). https://doi.org/10.1007/978-1-4757-6388-1_18.

[13] Y. Narushima, H. Yabe, J.A. Ford, A Three-Term Conjugate Gradient Method with Sufficient Descent Property for Unconstrained Optimization, SIAM J. Optim. 21 (2011), 212–230. https://doi.org/10.1137/080743573.

[14] L. Zhang, W. Zhou, D.-H. Li, A Descent Modified Polak–Ribière–Polyak Conjugate Gradient Method and Its Global Convergence, IMA J. Numer. Anal. 26 (2006), 629–640. https://doi.org/10.1093/imanum/drl016.

[15] S.P. Dirkse, M.C. Ferris, MCPLIB: A Collection of Nonlinear Mixed Complementarity Problems, Optim. Methods Softw. 5 (1995), 319–345. https://doi.org/10.1080/10556789508805619.

[16] Q.-R. Yan, X.-Z. Peng, D.-H. Li, A Globally Convergent Derivative-Free Method for Solving Large-Scale Nonlinear Monotone Equations, J. Comput. Appl. Math. 234 (2010), 649–657. https://doi.org/10.1016/j.cam.2010.01.001.

[17] R.Z. Al-Kawaz, A.B. Abubakar, A.H. Ibrahim, K. Khammahawong, Algorithm for Solving Nonlinear Monotone Operator Equations with Applications, J. Nonlinear Funct. Anal. 2023 (2023), 37. https://doi.org/10.23952/jnfa.2023.37.

[18] J. Sabi'u, R.Z. Al-Kawaz, An Efficient Modified Conjugate Gradient Parameter for Solving the System of Symmetric Nonlinear Equations with Application in Motion Control of Coplanar Robot, Comput. Appl. Math. 44 (2024), 49. https://doi.org/10.1007/s40314-024-02926-w.

[19] J.K. Liu, S.J. Li, A Projection Method for Convex Constrained Monotone Nonlinear Equations with Applications, Comput. Math. Appl. 70 (2015), 2442–2453. https://doi.org/10.1016/j.camwa.2015.09.014.

[20] M. Sun, J. Liu, Three Derivative-Free Projection Methods for Nonlinear Equations with Convex Constraints, J. Appl. Math. Comput. 47 (2015), 265–276. https://doi.org/10.1007/s12190-014-0774-5.

[21] A.M. Awwal, P. Kumam, A.B. Abubakar, A Modified Conjugate Gradient Method for Monotone Nonlinear Equations with Convex Constraints, Appl. Numer. Math. 145 (2019), 507–520. https://doi.org/10.1016/j.apnum.2019.05.012.

[22] S.G. Mahmood, R.S. Karyakos, I.M. Yacoob, Hybrid Gene Selection Method Based on Mutual Information Technique and Dragonfly Optimization Algorithm, East.-Eur. J. Enterp. Technol. 3 (2021), 64–69. https://doi.org/10.15587/1729-4061.2021.233382.

[23] A.M. Awwal, P. Kumam, A.B. Abubakar, A. Wakili, A Projection Hestenes–Stiefel-Like Method for Monotone Nonlinear Equations with Convex Constraints, Thai J. Math. 17 (2019), 181–199.

[24] W. La Cruz, J. Martínez, M. Raydan, Spectral Residual Method without Gradient Information for Solving Large-Scale Nonlinear Systems of Equations, Math. Comput. 75 (2006), 1429–1448. https://doi.org/10.1090/s0025-5718-06-01840-0.

[25] W. La Cruz, A Spectral Algorithm for Large-Scale Systems of Nonlinear Monotone Equations, Numer. Algorithms 76 (2017), 1109–1130. https://doi.org/10.1007/s11075-017-0299-8.

[26] W. Zhou, D. Shen, An Inexact PRP Conjugate Gradient Method for Symmetric Nonlinear Equations, Numer. Funct. Anal. Optim. 35 (2014), 370–388. https://doi.org/10.1080/01630563.2013.871290.

[27] E.D. Dolan, J.J. Moré, Benchmarking Optimization Software with Performance Profiles, Math. Program. 91 (2002), 201–213. https://doi.org/10.1007/s101070100263.