RAIRO-Oper. Res. 55 (2021) 2615–2629
https://doi.org/10.1051/ro/2021117

RAIRO Operations Research
www.rairo-ro.org

A MODIFIED PERRY-TYPE DERIVATIVE-FREE PROJECTION METHOD FOR SOLVING LARGE-SCALE NONLINEAR MONOTONE EQUATIONS

M. Koorapetse*, P. Kaelo and S. Kooepile-Reikeletseng

Abstract. In this paper, a new modified Perry-type derivative-free projection method for solving large-scale nonlinear monotone equations is presented. The method is developed by combining a modified Perry's conjugate gradient method with the hyperplane projection technique. Global convergence and numerical results of the proposed method are established. Preliminary numerical results show that the proposed method is promising and efficient compared to some existing methods in the literature.

Mathematics Subject Classification. 65K10, 90C06, 90C30, 65K05.

Received March 19, 2021. Accepted July 31, 2021.

1. Introduction

In this paper, we use a modified Perry-type derivative-free projection method to solve the nonlinear monotone equations

$$F(x) = 0, \quad x \in \Omega, \tag{1.1}$$

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

$$(F(x) - F(y))^T (x - y) \geq 0, \quad \forall x, y \in \mathbb{R}^n.$$

If $\Omega = \mathbb{R}^n$ then (1.1) is a general system of nonlinear equations problem, and when $\Omega \subseteq \mathbb{R}^n$ is a nonempty closed convex set then (1.1) is said to be constrained. Nonlinear monotone equations have many practical applications such as in chemical equilibrium systems [20] and the economic equilibrium problems [9]. Also, some monotone variational inequality problems can also be converted into nonlinear monotone equations by means of fixed point mappings or normal maps if the underlying function satisfies some coercive conditions [34].

Conjugate gradient-based projection methods are among the most famous and efficient methods for solving (1.1) and, thus, have recently received a lot of attention. This is due to their simplicity, global convergence properties and low memory requirements, which make them suitable for solving large-scale equations [8, 16, 26]. They are iterative methods, that is, given x_k, the next iterate x_{k+1} is obtained as

$$x_{k+1} = P_\Omega \left[x_k - \frac{F(z_k)^T (x_k - z_k)}{\|F(z_k)\|^2} F(z_k) \right], \tag{1.2}$$

Keywords. Derivative-free, nonlinear monotone equations, Perry's conjugate gradient method, projection technique.

Department of Mathematics, University of Botswana, Private Bag UB00704, Gaborone, Botswana
*Corresponding author: koorapetsem@ub.ac.bw

where $z_k = x_k + \alpha_k d_k$, $\alpha_k > 0$ is a step length and

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

with β_k being a conjugate gradient parameter such that

$$F_k^T d_k \leq -c\|F_k\|^2, \quad c > 0, \tag{1.3}$$

where $F_k = F(x_k)$ and $\|\cdot\|$ is the Euclidean norm. If Ω is a closed convex subset of $\mathbb{R}^n$ then the projection operator $P_\Omega[\cdot]$ is a map from $\mathbb{R}^n$ onto Ω, that is,

$$P_\Omega[x] = \arg\min\{\|x - y\| \mid y \in \Omega\}, \ \ \forall x \in \mathbb{R}^n.$$

This operator is non-expansive, that is, for any $x, y \in \mathbb{R}^n$,

$$\|P_\Omega[x] - P_\Omega[y]\| \leq \|x - y\|.$$

Conjugate gradient-based projection methods are obtained by combining conjugate gradient methods [3, 5, 6, 30, 32] with the projection technique proposed by Solodov and Svaiter [25]. Thus, they differ according to how the direction is obtained, or, more specifically, in how the parameter β_k is constructed. Recently, by employing Perry's conjugate gradient parameter [22], in which

$$\beta_k^P = \frac{F_k^T(y_{k-1} - s_{k-1})}{y_{k-1}^T d_{k-1}},$$

or the Dai-Liao conjugate gradient parameter [5], in which

$$\beta_k^{\mathrm{DL}} = \frac{F_k^T(y_{k-1} - ts_{k-1})}{y_{k-1}^T d_{k-1}}, \ t \geq 0,$$

where $y_{k-1} = F_k - F_{k-1}$ and $s_{k-1} = x_k - x_{k-1}$, a number of modified forms of the Perry and Dai-Liao methods for nonlinear systems of equations have been proposed [1, 4, 7, 27, 28]. These Perry and Dai-Liao methods are based on a quasi-Newton aspect and have been considered to be among the most effective in the context of unconstrained optimization. Note that when $t = 1$, the Dai-Liao method reduces to the Perry method.

In Dai *et al.* [7], a modified Perry method is combined with the hyperplane technique [25] to give a derivative-free method for solving large-scale nonlinear monotone equations. And in Waziri *et al.* [28], two enhanced Dai-Liao methods are presented based on two modified spectral coefficients and a revised form of the extended secant condition in [29]. All these methods were shown to be very competitive when compared to some of the existing methods for solving large-scale nonlinear monotone equations.

Abubakar *et al.*, in [2], constructed three-term conjugate gradient projection methods by proposing the direction as

$$d_k = \begin{cases} -F_k, & \text{if } k = 0, \\ -F_k + \beta_k^{CD} w_{k-1} - \lambda_k F_k, & \text{if } k \geq 1, \end{cases}$$

with $\beta_k^{CD} = \frac{\|F_k\|^2}{-d_{k-1}^T F_k}$ and $w_{k-1} = z_{k-1} - x_{k-1} = \alpha_{k-1} d_{k-1}$. In order to satisfy (1.3), they derived the parameter λ_k using three different approaches, thus proposing three different conjugate gradient projection methods with

$$\lambda_k = \frac{F_k^T w_{k-1}}{-d_{k-1}^T F_k}, \quad \lambda_k = \frac{\|F_k\|^2 \|w_{k-1}\|^2}{(-d_{k-1}^T F_k)^2} \quad \text{and} \quad \lambda_k = \frac{F_k^T w_{k-1}}{-d_{k-1}^T F_k} + \frac{\|F_k\|^2}{(-d_{k-1}^T F_k)^2},$$

and they named the resulting algorithms as M3TCD1, M3TCD2 and M3TCD3, respectively.

Another recently suggested three term derivative-free conjugate gradient-based projection method, which the authors named PDY, is that by Liu and Feng [16]. In this method, the authors proposed a three term derivative-free projection method based on the Dai-Yuan (DY) conjugate gradient parameter

$$\beta_k = \frac{\|F_k\|^2}{d_{k-1}^T w_{k-1}},$$

where

$$y_{k-1} = F_k - F_{k-1},\ w_{k-1} = y_{k-1} + t_{k-1} d_{k-1}, \text{ and } t_{k-1} = 1 + \max\left\{0, -\frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1}}\right\}.$$

This method was shown to have nice convergence properties. And Yan *et al.*, in [31], proposed two derivative-free projection methods based on the three term Hestenes-Stiefel (HS) conjugate gradient method of Zhang *et al.* [32]. They showed that their methods were also efficient for solving large-scale nonlinear systems of monotone equations. Based on a modified line search, an extension of the scaled conjugate gradient (SCG) method of [3] and the projection technique, Ou and Li [21] proposed a derivative-free SCG-type projection method for nonlinear monotone equations with convex constraints. They showed further that when F in (1.1) is a strongly monotone mapping, then the sequence $\{x_k\}$ generated by their method R-linearly converges to $x^* \in \Omega^*$, where Ω^* is the solution set of (1.1). Other gradient-based projection methods for large-scale nonlinear monotone equations can be found in [11–15, 18, 23, 26, 33, 35].

Motivated by the works of [19, 30], we propose a descent Perry-type derivative-free projection method for solving large-scale nonlinear monotone equations. This method is presented in the next section. In Section 3, we prove the global convergence of the proposed method followed by the convergence rate in Section 4. Numerical results follow in Section 5. Finally, conclusion is presented in Section 6.

2. Motivation and the algorithm

In this section, we describe the details of the proposed method. But first, we briefly review the work of Livieris and Pintelas [19] and that of Yao and Ning [30] which motivated this work.

Livieris and Pintelas, in [19], proposed a modified Perry's conjugate gradient method for the unconstrained optimization problem

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

with $f : \mathbb{R}^n \to \mathbb{R}$ being a continuously differentiable function bounded from below. The method generates iterations $x_{k+1} = x_k + \alpha_k d_k$ using the direction

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -\left(1 + \beta_k^{MP} \frac{g_k^T d_{k-1}}{\|g_k\|^2}\right) g_k + \beta_k^{MP} d_{k-1}, & \text{if } k \geq 1, \end{cases}$$

where $g_k = \nabla f(x_k)$ is the gradient of f at x_k, and

$$\beta_k^{MP} = \frac{g_k^T (w_{k-1} - s_{k-1})}{w_{k-1}^T d_{k-1}}$$

with

$$w_{k-1} = y_{k-1} + h_k \|g_{k-1}\|^r s_{k-1}.$$

The authors suggested h_k be given as

$$h_k = t + \max\left\{-\frac{s_{k-1}^T y_{k-1}}{\|s_{k-1}\|^2}, 0\right\} \|g_{k-1}\|^{-r},$$

where t and r are positive constants, and $y_{k-1} = g_k - g_{k-1}$ and $s_{k-1} = x_k - x_{k-1}$. The parameter β_k^{MP} was shown to satisfy the sufficient descent property

$$d_k^T g_k \leq -c\|g_k\|^2, \quad \forall\, k \geq 0,$$

where $c > 0$ is a positive constant and that the method is globally convergent.

Yao and Ning [30], using a modified symmetric Perry matrix

$$Q_k = I - t_k \frac{s_{k-1}y_{k-1}^T + y_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}} + \frac{s_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}},$$

where t_k is a positive parameter to be determined, defined their search direction as

$$d_k = -Q_k g_k,\ \forall k \geq 1.$$

By minimizing the distance, in the Frobenius norm, between the above Perry matrix and the self-scaling memoryless BFGS matrix [24]

$$H = \xi_k I - \xi_k \frac{s_{k-1}y_{k-1}^T + y_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}} + \left(1 + \xi_k \frac{\|y_{k-1}\|^2}{s_{k-1}^T y_{k-1}}\right) \frac{s_{k-1}s_{k-1}^T}{s_{k-1}^T y_{k-1}},$$

they determined an optimal parameter t_k^* as

$$t_k^* = \frac{1}{1 + a_k},\ \ a_k = \frac{\|s_{k-1}\|^2 \|y_{k-1}\|^2}{(s_{k-1}^T y_{k-1})^2}.$$

Hence, using this optimal parameter t_k^*, the authors suggested an adaptive three term Perry's conjugate gradient method

$$d_k = -g_k + \beta_k d_{k-1} + \delta_k y_{k-1},$$

where

$$\beta_k = \frac{g_k^T(t_k y_{k-1} - s_{k-1})}{d_{k-1}^T y_{k-1}} \ \text{ and } \ \delta_k = \frac{t_k g_k^T s_{k-1}}{s_{k-1}^T y_{k-1}},$$

with

$$t_k = \min\left\{\frac{1}{1 + a_k}, \frac{s_{k-1}^T y_{k-1}}{\|y_{k-1}\|^2}\right\}.$$

The method was shown to be effective numerically.

Now, motivated by the above discussed modified Perry's conjugate gradient methods of [19, 30], we propose a modified Perry-type derivative-free projection method for solving (1.1). By taking a careful look at the search direction presented in [19] and the β_k parameter in [30], we propose our search direction as

$$d_k = \begin{cases} -F_k, & \text{if } k = 0, \\ -\left(\lambda_k + \beta_k^M \frac{F_k^T d_{k-1}}{\|F_k\|^2}\right) F_k + \beta_k^M d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{2.1}$$

where

$$\beta_k^M = \frac{F_k^T(\lambda_k w_{k-1} - s_{k-1})}{w_{k-1}^T d_{k-1}} \quad \text{and} \quad \lambda_k = \begin{cases} \lambda^*, & \text{if } \lambda^* \in [\kappa, 1], \\ 1, & \text{otherwise}, \end{cases}$$

with $\lambda^* = \frac{\|s_{k-1}\|^2}{s_{k-1}^T u_{k-1}}$, $s_{k-1} = z_{k-1} - x_{k-1}$, $u_{k-1} = y_{k-1} + \phi s_{k-1}$, $y_{k-1} = F(z_{k-1}) - F_{k-1}$, $w_{k-1} = u_{k-1} + \|F_{k-1}\| s_{k-1}$, and ϕ is a positive constant and $\kappa \in (0,1]$. With d_k given in (1.3) above, we determine the next iterate x_{k+1} using (1.2), where the step length $\alpha_k = \max\{\rho^i : i = 0,1,2,...\}$, $\rho \in (0,1]$, is such that it satisfies

$$-F(x_k + \alpha_k d_k)^T d_k \geq \sigma \alpha_k \|F(x_k + \alpha_k d_k)\| \|d_k\|^2, \quad \sigma > 0. \tag{2.2}$$

We present our proposed method below.

Algorithm 1 New Modified Perry-type Derivative-free Projection Method

1: Given $x_0 \in \Omega$, σ, κ, ϕ, $\epsilon > 0$ and $\rho \in (0,1]$, set $k = 0$.
2: **for** $k = 0, 1, \ldots$ **do**
3: If $\|F_k\| \leq \epsilon$, then stop. Otherwise, go to Step 4.
4: Compute d_k by (2.1).
5: Compute $z_k = x_k + \alpha_k d_k$, where α_k is obtained by (2.2)
6: If $z_k \in \Omega$ and $\|F(z_k)\| \leq \epsilon$, stop. Otherwise compute x_{k+1} by (1.2).
7: Set $k = k+1$ and go to Step 3.
8: **end for**

3. Global convergence

We now establish the global convergence of the presented Algorithm 1. Throughout the paper, we assume that $F_k \neq 0$ for all k, otherwise a stationary point has been found. We also assume that the following assumption holds.

Assumption 3.1.
(i) *The function $F(\cdot)$ is monotone on $\mathbb{R}^n$.*
(ii) *The solution set Ω^* is nonempty.*
(iii) *The function $F(\cdot)$ is Lipschitz continuous on $\mathbb{R}^n$, i.e. there exists a positive constant L such that*

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

Note that from the monotonicity of F, we have that

$$s_{k-1}^T u_{k-1} = (F(z_{k-1}) - F_{k-1})^T (z_{k-1} - x_{k-1}) + \phi\|s_{k-1}\|^2 \geq \phi\|s_{k-1}\|^2 > 0.$$

This indicates that λ^* is positive whenever s_{k-1} is not zero and hence λ_k in (2.1) is well-defined.

Lemma 3.2. *The search direction d_k generated by Algorithm 1 satisfies the descent condition*

$$F_k^T d_k \leq -\kappa\|F_k\|, \quad \forall\, k \geq 0 \quad \text{and} \quad \kappa > 0. \tag{3.1}$$

Proof. Since $d_0 = -F_0$, we have $F_0^T d_0 = -\|F_0\|^2$, which satisfies (3.1). Now, for $k \geq 1$, we obtain from (2.1) and the relation $\lambda_k \geq \kappa$ that

$$\begin{aligned} F_k^T d_k &= -\left(\lambda_k + \beta_k^M \frac{F_k^T d_{k-1}}{\|F_k\|^2}\right)\|F_k\|^2 + \beta_k^M F_k^T d_{k-1} \\ &= -\lambda_k \|F_k\|^2 \\ &\leq -\kappa\|F_k\|^2. \end{aligned}$$

□

Lemma 3.3. *Suppose that Assumption 3.1 holds, and let the sequences* $\{x_k\}$ *and* $\{z_k\}$ *be generated by Algorithm 1. Then, for any* $x^* \in \Omega$, *it holds that*

$$\|x_{k+1} - x^*\|^2 \le \|x_k - x^*\|^2 - \sigma^2 \|x_k - z_k\|^4. \tag{3.2}$$

Moreover, the sequences $\{x_k\}$ *and* $\{z_k\}$ *are bounded, and*

$$\sum_{k=0}^{\infty} \|x_k - z_k\|^4 < \infty.$$

Furthermore, it holds that

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.3}$$

Proof. From (2.2), we have

$$F(z_k)^T(x_k - z_k) \ge \sigma \|F(z_k)\| \|x_k - z_k\|^2 > 0. \tag{3.4}$$

For $x^* \in \Omega$, we obtain from (1.2) that

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|P_\Omega(x_k - \theta_k F(z_k)) - x^*\|^2 \\ &\le \|x_k - \theta_k F(z_k) - x^*\|^2 \\ &= \|x_k - x^*\|^2 - 2\theta_k F(z_k)^T(x_k - x^*) + \theta_k^2 \|F(z_k)\|^2, \end{aligned} \tag{3.5}$$

where $\theta_k = \frac{F(z_k)^T(x_k - z_k)}{\|F(z_k)\|^2}$. By monotonicity of F, we obtain

$$\begin{aligned} F(z_k)^T(x_k - x^*) &= F(z_k)^T(x_k - z_k) + F(z_k)^T(z_k - x^*) \\ &\ge F(z_k)^T(x_k - z_k) + F(x^*)^T(z_k - x^*) \\ &= F(z_k)^T(x_k - z_k). \end{aligned} \tag{3.6}$$

From (3.4)–(3.6), we get that

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &\le \|x_k - x^*\|^2 - 2\theta_k F(z_k)^T(x_k - z_k) + \theta_k^2 \|F(z_k)\|^2 \\ &= \|x_k - x^*\|^2 - \frac{(F(z_k)^T(x_k - z_k))^2}{\|F(z_k)\|^2} \\ &\le \|x_k - x^*\|^2 - \sigma^2 \|x_k - z_k\|^4. \end{aligned} \tag{3.7}$$

Thus, the sequence $\{\|x_k - x^*\|\}$ is decreasing and convergent, and hence $\{x_k\}$ is bounded. From (3.4), it follows that

$$\begin{aligned} \sigma \|F(z_k)\| \|x_k - z_k\|^2 &\le F(z_k)^T(x_k - z_k) \\ &\le \|F(z_k)\| \|x_k - z_k\|, \end{aligned}$$

giving

$$\sigma \|x_k - z_k\| \le 1, \tag{3.8}$$

which shows that $\{z_k\}$ is also bounded.

Furthermore, it follows from (3.7) that

$$\sigma^2 \sum_{k=0}^{\infty} \|x_k - z_k\|^4 \ \le\ \sum_{k=0}^{\infty} (\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2) < \infty,$$

which implies

$$\lim_{k\to\infty} \|x_k - z_k\| \ =\ \lim_{k\to\infty} \alpha_k \|d_k\| \ = 0.$$

□

Observe from (3.7) that

$$\|x_{k+1} - x^*\|^2 \le \|x_k - x^*\|^2$$

implies

$$\|x_k - x^*\|^2 \le \|x_0 - x^*\|^2, \ \ \forall k \ge 0.$$

Therefore, since F is continuous, by the Lipschitz condition we get that

$$\begin{aligned} \|F(x_k)\| =&\ \|F(x_k) - F(x^*)\| \\ \le&\ L\|x_k - x^*\| \\ \le&\ L\|x_0 - x^*\|. \end{aligned}$$

Taking $\gamma = L\|x_0 - x^*\|$ gives that $\|F(x_k)\| \le \gamma$. Also, (3.8) implies that there is a positive constant μ such that $\|s_k\| \le \mu, \ k \ge 0$.

Lemma 3.4. *For all $k \ge 0$, we have*

$$\kappa\|F_k\| \ \le\ \|d_k\| \ \le\ \varphi\|F_k\|. \tag{3.9}$$

where φ is a positive constant.

Proof. From (3.1) and Cauchy-Schwarz inequality , we obtain

$$\|d_k\| \ \ge\ \kappa\|F_k\|.$$

And from (2.1), we have

$$\|d_k\| \ \le\ \|F_k\| + 2|\beta_k^M|\|d_{k-1}\|. \tag{3.10}$$

From the definitions of u_{k-1}, w_{k-1} and s_{k-1} in (2.1), we get that there exist positive constants ϖ and ϱ such that

$$\|u_{k-1}\| \le \|F(z_k)\| + \|F_{k-1}\| + \phi\|s_{k-1}\| \le 2\gamma + \phi\mu = \varpi$$

and

$$\|w_{k-1}\| \le \|u_{k-1}\| + \|F_{k-1}\|\|s_{k-1}\| \le \varpi + \gamma\mu = \varrho.$$

Notice that for $F_{k-1} \ne 0$ and $s_{k-1} \ne 0$, we get

$$s_{k-1}^T w_{k-1} = s_{k-1}^T u_{k-1} + \|F_{k-1}\|\|s_{k-1}\|^2 >\ \|F_{k-1}\|\|s_{k-1}\|^2 > 0.$$

Hence, there is a constant $\omega > 0$ such that

$$d_{k-1}^T w_{k-1} = \alpha_{k-1}^{-1} s_{k-1}^T w_{k-1} > \alpha_{k-1}^{-1} \|F_{k-1}\| \|s_{k-1}\|^2 \geq \omega \|d_{k-1}\|.$$

It then follows that

$$|\beta_k^M| \leq \frac{\|F_k\|(\|w_{k-1}\| + \|s_{k-1}\|)}{w_{k-1}^T d_{k-1}} \leq \frac{\|F_k\|(\varrho + \mu)}{\omega \|d_{k-1}\|}.$$

From (3.10) we get

$$\|d_k\| \leq \|F_k\| + 2\frac{\|F_k\|(\varrho + \mu)}{\omega \|d_{k-1}\|}\|d_{k-1}\| = \|F_k\| \left(1 + \frac{2(\varrho + \mu)}{\omega}\right) = \varphi \|F_k\|,$$

where $\varphi = 1 + \frac{2(\varrho+\mu)}{\omega}$. □

Lemma 3.5. *Let $\{x_k\}$ and $\{z_k\}$ be generated by Algorithm 1. Then*

$$\alpha_k \geq \min\left\{1, \frac{\kappa\rho}{(L + \sigma\gamma)\varphi^2}\right\} > 0. \tag{3.11}$$

Proof. From the line search procedure (2.2), if $\alpha_k \neq 1$, then $\alpha_k' = \rho^{-1}\alpha_k$ does not satisfy (2.2). This means that

$$-F(z_k')^T d_k < \sigma \alpha_k' \|F(z_k')\| \|d_k\|^2,$$

where $z_k' = x_k + \alpha_k' d_k$. This together with (2.2), (3.1) and (3.9) imply that

$$\begin{aligned}
\kappa\|F_k\|^2 &\leq -F_k^T d_k \\
&= (F(z_k') - F_k)^T d_k - F(z_k')^T d_k \\
&\leq L\alpha_k' \|d_k\|^2 + \sigma\alpha_k' \|F(z_k')\| \|d_k\|^2 \\
&= (L + \sigma\|F(z_k')\|)\alpha_k \rho^{-1} \|d_k\|^2 \\
&\leq (L + \sigma\gamma)\alpha_k \rho^{-1} \varphi^2 \|F_k\|^2.
\end{aligned}$$

Thus

$$\alpha_k \geq \frac{\kappa\rho}{(L + \sigma\gamma)\varphi^2} > 0,$$

which gives the desired result. □

Theorem 3.6. *Suppose that Assumption 3.1 holds, and let the sequence $\{x_k\}$ be generated by Algorithm 1. Then*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{3.12}$$

Proof. We assume that (3.12) does not hold, that is, $\exists\, \eta > 0$ such that $\|F_k\| \geq \eta$, $\forall\, k \geq 0$. It then follows from (3.9) that

$$\|d_k\| \geq \kappa\|F_k\| \geq \kappa\eta > 0, \qquad \forall\, k \geq 0,$$

and (3.3) implies that

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.13}$$

On the other hand, (3.11) implies that $\alpha_k > 0$, $\forall k \geq 0$, which contradicts (3.13). Thus, (3.12) holds. □

4. Convergence Rate

From the above discussions, it is evident that the sequence $\{x_k\}$ converges to a solution of problem (1.1). Therefore, we always assume that x_k converges to x^* as $k \to \infty$, where x^* belongs to the solution set Ω^* of problem (1.1). To determine the rate of convergence for the proposed algorithm, we also assume the following assumption holds.

Assumption 4.1. *For any $x^* \in \Omega^*$, there exist two positive constants ψ and δ satisfying*

$$\psi \text{dist}(x, \Omega^*) \le \|F(x)\|^2, \quad \forall\, x \in \mathcal{N}_\delta(x^*), \tag{4.1}$$

where $\mathcal{N}_\delta(x^*) = \{x \in \mathbb{R}^n : \|x - x^*\| \le \delta\}$ and $\text{dist}(x, \Omega^*)$ is the distance from x to Ω^*.

Theorem 4.2. *Let Assumptions 3.1 and 4.1 hold, and the sequence $\{x_k\}$ be generated by Algorithm 1. Then the sequence $\{\text{dist}(x_k, \Omega^*)\}$ is Q-linearly convergent to 0, and hence the sequence $\{x_k\}$ R-linearly converges to x^*.*

Proof. Let $\bar{x}_k := \arg\min\{\|x_k - \bar{x}\| : \bar{x} \in \Omega^*\}$, which implies that $\bar{x}_k$ is the closest solution to x_k, namely,

$$\|x_k - \bar{x}_k\| = \text{dist}(x_k, \Omega^*).$$

Denoting x^* by $\bar{x}_k$, it follows from (3.2) that

$$\|x_{k+1} - \bar{x}_k\|^2 \le \|x_k - \bar{x}_k\|^2 - \sigma^2 \|x_k - z_k\|^4. \tag{4.2}$$

This, together with (3.9) and (4.1), give that for $\bar{x}_k \in \Omega^*$,

$$\begin{aligned}
\text{dist}(x_{k+1}, \Omega^*)^2 &= \|x_{k+1} - \bar{x}_k\|^2 \\
&\le \|x_k - \bar{x}_k\|^2 - \sigma^2 \|x_k - z_k\|^4 \\
&\le \text{dist}(x_k, \Omega^*)^2 - \sigma^2 \|\alpha_k d_k\|^4 \\
&\le \text{dist}(x_k, \Omega^*)^2 - \sigma^2 \kappa^4 \alpha_k^4 \|F_k\|^4 \\
&\le \text{dist}(x_k, \Omega^*)^2 - \sigma^2 \psi^2 \kappa^4 \alpha_k^4 \text{dist}(x_k, \Omega^*)^2 \\
&= (1 - \sigma^2 \psi^2 \kappa^4 \alpha_k^4) \text{dist}(x_k, \Omega^*)^2.
\end{aligned}$$

Taking $\kappa^2 \le \frac{1}{\sigma\psi}$, we get that $1 - \sigma^2\psi^2\kappa^4\alpha_k^4 \in (0, 1)$ holds. This implies that the sequence $\{\text{dist}(x_k, \Omega^*)\}$ converges to 0 Q-linearly. Therefore, the whole sequence $\{x_k\}$ converges to x^* R-linearly. □

5. Numerical Experiments

In this section, we report some numerical results to test the efficacy of our proposed Algorithm 1, herein denoted as $NMPCG$. We compare it with other three term derivative-free projection methods that have recently been proposed in the literature. These are the three-term conjugate descent projection method of Abubakar *et al.* [2], denoted $M3TCD1$, the derivative-free iterative method of Liu and Feng [16], denoted PDY, and the partially symmetrical derivative-free Liu-Storey projection method of Liu *et al.* [18], denoted sLS. All algorithms are coded in MATLAB R2019b and the methods are compared using number of iterations, number of function evaluations and CPU time taken for each method to reach the optimal value or termination. We test the algorithms on eight test problems, with various dimensions, using four different starting points $x_0^1 = (-0.1, -0.1, ..., -0.1)^T$, $x_0^2 = (0.1, 0.1, ..., 0.1)^T$, $x_0^3 = (0.5, 0.5, ..., 0.5)^T$ and $x_0^4 = (2, 2, ..., 2)^T$. The eight test problems, where the mapping $F(\cdot)$ is taken as $F(x) = (F_1(x), F_2(x), F_3(x), ..., F_n(x))^T$, are as follows.

Problem 1 [2].

$$F_i(x) = e^{x_i} - 1, \quad \text{for} \quad i = 1, 2, 3, ..., n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 2 [2].

$$F_1(x) = x_1 - e^{\cos(\frac{x_1+x_2}{n+1})},$$
$$F_i(x) = x_i - e^{\cos(\frac{x_{i-1}+x_i+x_{i+1}}{n+1})}, \quad \text{for} \quad i = 2, 3, ..., n-1 \quad \text{and} \quad \Omega = \mathbb{R}^n_+,$$
$$F_n(x) = x_n - e^{\cos(\frac{x_{n-1}+x_n}{n+1})}.$$

Problem 3 [2].

$$F_i(x) = 2x_i - \sin(|x_i|), \quad \text{for} \quad i = 1, 2, 3, ..., n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 4 [13].

$$F_i(x) = \ln(|x_i| + 1) - \frac{x_i}{n}, \quad \text{for} \quad i = 1, 2, 3, ..., n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 5 [2].

$$F_i(x) = x_i - \sin(|x_i - 1|), \quad \text{for} \quad i = 1, 2, ..., n \quad \text{and} \quad \Omega = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq n, x_i \geq 0 \right\}.$$

Problem 6. [2].

$$F_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, \quad \text{for} \quad i = 1, 2, 3, ..., n \quad \text{and} \quad \Omega = \left\{ x \in \mathbb{R}^n : \sum_{i=1}^n x_i \leq n, x_i > -1 \right\}.$$

Problem 7 [11].

$$F_1(x) = x_1(2x_1^2 + 2x_2^2) - 1,$$
$$F_i(x) = x_i(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2) - 1, \quad \text{for} \quad i = 2, 3, ..., n-1 \quad \text{and} \quad \Omega = \mathbb{R}^n_+,$$
$$F_n(x) = x_n(2x_{n-1}^2 + 2x_n^2) - 1.$$

Problem 8 [17].

$$F_1(x) = x_1 - e^{\cos(\frac{x_1+x_2}{2})},$$
$$F_i(x) = x_i - e^{\cos(\frac{x_{i-1}+x_i+x_{i+1}}{i})}, \quad \text{for} \quad i = 2, 3, ..., n-1 \quad \text{and} \quad \Omega = \mathbb{R}^n_+,$$
$$F_n(x) = x_n - e^{\cos(\frac{x_{n-1}+x_n}{n})}.$$

In our experiments, the algorithms are stopped whenever the inequality $\|F_k\| \leq \epsilon = 10^{-6}$ is satisfied, or the total number of iterations exceeds 1000. The $NMPCG$ method is implemented with the parameters $\sigma = 10^{-4}$, $\rho = 0.5$, $\phi = 10^{-5}$ and $\kappa = 10^{-5}$, while parameters for the algorithms sLS, PDY and $M3TCD1$ are set as in respective papers.

The numerical results are reported in Tables 1 and 2, where $ITER$ refers to the number of iterations, FE stands for the number of function evaluations and CPU is the CPU time in seconds. We note here that all algorithms managed to solve all the eight test problems successfully. From Tables 1 and 2, we see that in terms of number of iterations, the $NMPCG$ and $M3TCD1$ methods are very competitive for most problems, and perform much better than the other methods. However, in terms of function evaluations, we see that the proposed $NMPCG$ method is the best. Overall, the results indicate that the proposed $NMPCG$ method performs better than the other competing methods, with the sLS method the weaker one among the four competing methods.

Table 1. Numerical results for problem 1–4.

			ITER				FE				CPU			
PROB	x_0	DIM	NMPCG	PDY	M3TCD1	sLS	NMPCG	PDY	M3TCD1	sLS	NMPCG	PDY	M3TCD1	sLS
1	x_0^1	5000	1	1	1	1	3	3	3	3	0.0015	0.0011	0.0020	0.0016
		10000	1	1	1	1	3	3	3	3	0.0045	0.0010	0.0021	0.0007
		20000	1	1	1	1	3	3	3	3	0.0009	0.0009	0.0009	0.0015
		50000	1	1	1	1	3	3	3	3	0.0026	0.0033	0.0034	0.0031
	x_0^2	5000	6	16	8	22	11	45	21	58	0.0039	0.0114	0.0063	0.0125
		10000	6	16	8	22	11	45	21	58	0.0031	0.0105	0.0051	0.0177
		20000	6	16	8	23	11	45	21	60	0.0069	0.0194	0.0099	0.0301
		50000	6	17	9	23	11	48	24	60	0.0106	0.0351	0.0170	0.0573
	x_0^3	5000	7	17	9	23	13	48	26	61	0.0022	0.0071	0.0035	0.0100
		10000	7	17	9	23	13	48	26	61	0.0035	0.0103	0.0054	0.0157
		20000	7	18	9	24	13	51	26	63	0.0057	0.0233	0.0086	0.0276
		50000	7	18	9	24	13	51	26	62	0.0169	0.0546	0.0210	0.0559
	x_0^4	5000	8	19	8	23	16	57	32	63	0.0025	0.0078	0.0035	0.0105
		10000	8	19	9	24	16	57	35	65	0.0042	0.0117	0.0067	0.0147
		20000	8	21	9	24	16	69	35	64	0.0061	0.0221	0.0100	0.0239
		50000	8	22	9	25	16	76	35	67	0.0139	0.0506	0.0214	0.0569
2	x_0^1	5000	7	19	10	27	15	54	27	69	0.0085	0.0280	0.0147	0.0304
		10000	5	21	10	27	10	64	27	69	0.0084	0.0628	0.0260	0.0693
		20000	7	22	10	28	14	68	27	71	0.0239	0.1092	0.0489	0.1142
		50000	6	25	10	28	12	86	27	69	0.0452	0.3695	0.1014	0.2640
	x_0^2	5000	7	19	10	27	15	54	27	69	0.0066	0.0291	0.0114	0.0378
		10000	5	21	10	27	10	64	27	69	0.0087	0.0648	0.0217	0.0619
		20000	5	21	10	27	10	64	27	68	0.0160	0.1236	0.0452	0.1105
		50000	6	24	10	28	12	81	27	69	0.0453	0.2935	0.1106	0.3002
	x_0^3	5000	7	18	10	26	15	51	27	66	0.0065	0.0253	0.0120	0.0372
		10000	5	20	10	27	10	59	27	69	0.0105	0.0475	0.0248	0.0583
		20000	5	21	10	27	10	64	27	69	0.0192	0.1125	0.0524	0.1204
		50000	6	23	10	28	12	76	27	69	0.0465	0.2943	0.1008	0.2811
	x_0^4	5000	5	17	9	25	10	48	24	65	0.0056	0.0198	0.0116	0.0320
		10000	5	18	9	25	10	51	24	65	0.0089	0.0413	0.0196	0.0554
		20000	5	18	10	26	10	51	27	67	0.0158	0.0916	0.0417	0.1228
		50000	5	18	10	26	10	51	27	65	0.0374	0.2202	0.0981	0.2540
3	x_0^1	5000	1	1	1	1	5	5	14	5	0.0005	0.0004	0.0011	0.0005
		10000	1	1	1	1	5	5	14	5	0.0007	0.0006	0.0016	0.0006
		20000	1	1	1	1	5	5	14	5	0.0011	0.0011	0.0030	0.0011
		50000	1	1	1	1	5	5	14	5	0.0023	0.0024	0.0061	0.0023
	x_0^2	5000	5	16	8	22	9	45	21	58	0.0013	0.0051	0.0023	0.0074
		10000	5	16	8	22	9	45	21	57	0.0018	0.0076	0.0036	0.0122
		20000	5	16	9	22	9	45	24	57	0.0032	0.0136	0.0069	0.0207
		50000	5	17	9	22	9	48	24	56	0.0071	0.0292	0.0144	0.0437
	x_0^3	5000	5	17	9	24	9	48	24	62	0.0013	0.0059	0.0029	0.0088
		10000	5	17	9	25	9	48	24	64	0.0020	0.0087	0.0044	0.0136
		20000	5	18	9	25	9	51	23	64	0.0032	0.0152	0.0065	0.0230
		50000	6	18	9	25	11	51	24	63	0.0086	0.0309	0.0145	0.0483
	x_0^4	5000	6	19	10	22	11	56	31	58	0.0016	0.0066	0.0035	0.0078
		10000	6	20	10	23	11	59	30	60	0.0024	0.0112	0.0050	0.0135
		20000	6	21	10	23	11	65	30	60	0.0043	0.0192	0.0089	0.0221
		50000	6	23	10	23	11	76	30	60	0.0092	0.0488	0.0184	0.0447
4	x_0^1	5000	1	1	1	1	3	3	3	3	0.0004	0.0004	0.0004	0.0004
		10000	1	1	1	1	3	3	3	3	0.0006	0.0006	0.0006	0.0006
		20000	1	1	1	1	3	3	3	3	0.0010	0.0009	0.0009	0.0009
		50000	1	1	1	1	3	3	3	3	0.0020	0.0021	0.0020	0.0021
	x_0^2	5000	4	2	4	6	6	5	6	10	0.0010	0.0007	0.0009	0.0017
		10000	4	2	4	6	6	5	6	10	0.0016	0.0011	0.0014	0.0028
		20000	4	2	4	6	6	5	6	10	0.0027	0.0018	0.0025	0.0049
		50000	4	2	4	8	6	5	6	14	0.0060	0.0039	0.0053	0.0144
	x_0^3	5000	5	2	5	10	8	5	8	18	0.0014	0.0008	0.0014	0.0035
		10000	5	2	5	10	8	5	8	18	0.0023	0.0011	0.0021	0.0054
		20000	5	2	5	10	8	5	8	18	0.0040	0.0020	0.0036	0.0097
		50000	5	2	5	10	8	5	8	18	0.0088	0.0041	0.0076	0.0210
	x_0^4	5000	7	2	7	14	12	5	12	26	0.0024	0.0008	0.0022	0.0055
		10000	7	4	7	14	12	13	12	26	0.0035	0.0031	0.0033	0.0084
		20000	7	4	7	15	12	13	12	28	0.0059	0.0052	0.0054	0.0147
		50000	7	6	7	15	12	23	12	28	0.0131	0.0201	0.0116	0.0328

Table 2. Numerical results for problem 5–8.

			ITER				FE				CPU			
PROB	x_0	DIM	NMPCG	PDY	M3TCD1	sLS	NMPCG	PDY	M3TCD1	sLS	NMPCG	PDY	M3TCD1	sLS
5	x_0^1	5000	45	25	49	49	113	95	380	169	0.0492	0.0367	0.1657	0.0736
		10000	46	27	48	52	116	104	373	177	0.1119	0.0770	0.2895	0.1493
		20000	37	26	49	44	93	101	374	142	0.1508	0.1479	0.5446	0.2289
		50000	45	31	45	43	118	126	350	145	0.4376	0.5036	1.2382	0.6233
	x_0^2	5000	51	25	45	52	127	94	350	179	0.0537	0.0601	0.1458	0.0825
		10000	45	25	45	43	113	97	351	140	0.0951	0.0797	0.3229	0.1336
		20000	50	26	52	46	125	100	408	159	0.2223	0.1982	0.7250	0.2836
		50000	49	28	44	43	124	113	344	145	0.4518	0.3983	1.1697	0.5386
	x_0^3	5000	45	23	57	50	113	87	441	172	0.0483	0.0408	0.2055	0.0713
		10000	54	23	58	43	124	86	454	140	0.1062	0.0693	0.3843	0.1399
		20000	46	28	52	50	116	108	400	171	0.2214	0.2034	0.6746	0.2815
		50000	77	28	48	44	173	112	368	142	0.7346	0.4184	1.4102	0.5480
	x_0^4	5000	53	21	58	55	134	77	448	183	0.0564	0.0355	0.1757	0.0761
		10000	53	21	60	51	134	77	452	172	0.1029	0.0804	0.4528	0.1433
		20000	54	21	78	56	137	77	588	184	0.2121	0.1156	0.8953	0.2990
		50000	56	22	54	57	142	80	408	188	0.5549	0.3054	1.5642	0.8342
6	x_0^1	5000	15	19	5	21	41	71	31	70	0.0107	0.0104	0.0050	0.0107
		10000	15	19	5	21	41	71	31	71	0.0115	0.0151	0.0054	0.0172
		20000	11	11	5	11	31	40	31	32	0.0155	0.0161	0.0095	0.0152
		50000	15	20	5	22	41	76	31	75	0.0451	0.0886	0.0303	0.0735
	x_0^2	5000	15	18	5	19	41	67	32	66	0.0071	0.0095	0.0034	0.0107
		10000	15	18	6	19	41	67	40	66	0.0122	0.0151	0.0073	0.0154
		20000	15	19	6	20	41	71	40	70	0.0200	0.0368	0.0123	0.0407
		50000	15	19	6	20	41	71	40	70	0.0412	0.0557	0.0250	0.0649
	x_0^3	5000	4	5	4	18	7	12	24	64	0.0012	0.0017	0.0026	0.0089
		10000	4	5	4	18	7	12	24	64	0.0019	0.0025	0.0036	0.0159
		20000	4	5	4	19	7	12	24	68	0.0037	0.0043	0.0070	0.0265
		50000	4	5	4	19	7	12	24	68	0.0085	0.0114	0.0155	0.0585
	x_0^4	5000	6	19	6	23	11	70	35	77	0.0019	0.0087	0.0035	0.0115
		10000	6	21	6	22	11	80	35	70	0.0034	0.0194	0.0064	0.0182
		20000	6	21	6	23	11	80	35	74	0.0063	0.0332	0.0111	0.0337
		50000	6	21	6	23	11	81	35	72	0.0129	0.0687	0.0219	0.0712
7	x_0^1	5000	6	16	5	11	11	45	11	23	0.0026	0.0087	0.0022	0.0053
		10000	6	16	5	11	11	45	11	24	0.0041	0.0137	0.0035	0.0091
		20000	6	16	5	12	11	45	11	27	0.0072	0.0243	0.0061	0.0179
		50000	6	17	6	13	11	48	14	30	0.0165	0.0549	0.0174	0.0416
	x_0^2	5000	4	14	4	6	6	38	6	10	0.0013	0.0064	0.0012	0.0022
		10000	4	14	4	6	6	38	6	10	0.0021	0.0112	0.0020	0.0041
		20000	4	15	4	6	6	41	6	10	0.0040	0.0211	0.0038	0.0066
		50000	4	15	4	8	6	41	6	14	0.0086	0.0450	0.0077	0.0206
	x_0^3	5000	5	17	5	10	8	47	8	18	0.0017	0.0080	0.0016	0.0040
		10000	5	17	5	10	8	47	8	18	0.0029	0.0137	0.0028	0.0072
		20000	5	18	5	10	8	50	8	18	0.0055	0.0272	0.0052	0.0135
		50000	5	18	5	10	8	50	8	18	0.0127	0.0610	0.0106	0.0283
	x_0^4	5000	7	19	7	14	12	53	12	26	0.0025	0.0092	0.0023	0.0066
		10000	7	19	7	14	12	54	12	26	0.0048	0.0161	0.0044	0.0107
		20000	7	20	7	15	12	57	12	28	0.0077	0.0293	0.0071	0.0220
		50000	7	21	7	15	12	62	12	28	0.0199	0.0792	0.0183	0.0462
8	x_0^1	5000	19	15	7	15	54	68	90	61	0.0111	0.0141	0.0172	0.0116
		10000	19	15	7	15	54	68	90	61	0.0229	0.0229	0.0283	0.0234
		20000	19	16	7	16	54	73	90	65	0.0379	0.0548	0.0713	0.0603
		50000	19	17	7	16	54	79	90	65	0.0956	0.1748	0.1509	0.1233
	x_0^2	5000	18	14	6	13	51	63	76	46	0.0112	0.0148	0.0122	0.0121
		10000	18	15	6	13	51	68	76	46	0.0232	0.0255	0.0240	0.0181
		20000	18	15	6	14	51	68	76	51	0.0521	0.0525	0.0563	0.0417
		50000	18	16	6	14	51	74	76	51	0.1167	0.1108	0.1315	0.1178
	x_0^3	5000	16	13	6	14	45	59	83	58	0.0090	0.0102	0.0130	0.0143
		10000	16	13	6	14	45	59	83	58	0.0173	0.0250	0.0389	0.0393
		20000	16	13	6	13	45	59	83	47	0.0404	0.0479	0.0834	0.0419
		50000	16	14	6	14	45	64	83	52	0.0993	0.1046	0.1674	0.1036
	x_0^4	5000	9	15	7	15	21	74	118	59	0.0051	0.0202	0.0411	0.0133
		10000	9	15	7	15	21	74	118	59	0.0198	0.0249	0.0473	0.0307
		20000	9	14	7	16	21	70	118	64	0.0191	0.0672	0.1010	0.0690
		50000	9	19	7	16	21	104	118	64	0.0409	0.1871	0.1871	0.1413

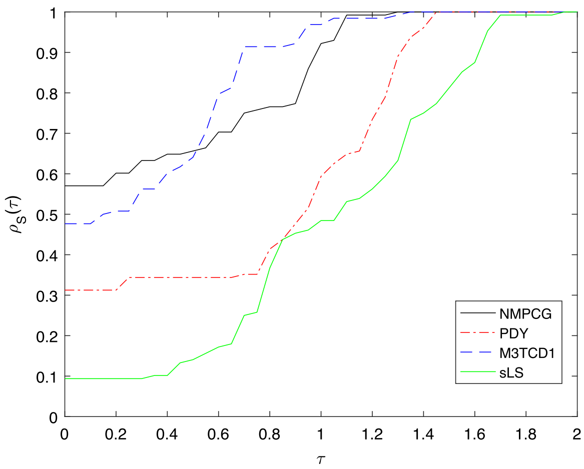

FIGURE 1. Iterations performance profile.

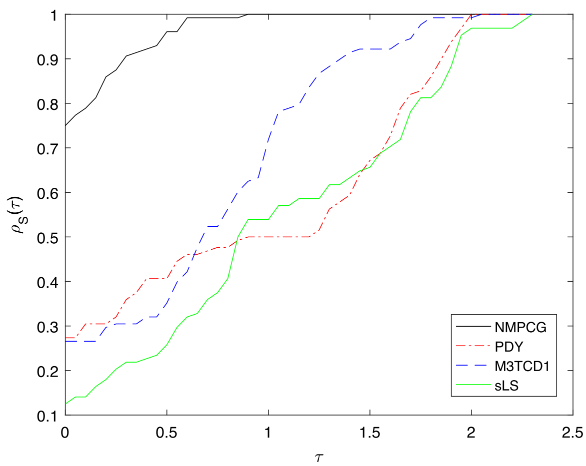

FIGURE 2. Function evaluations performance profile.

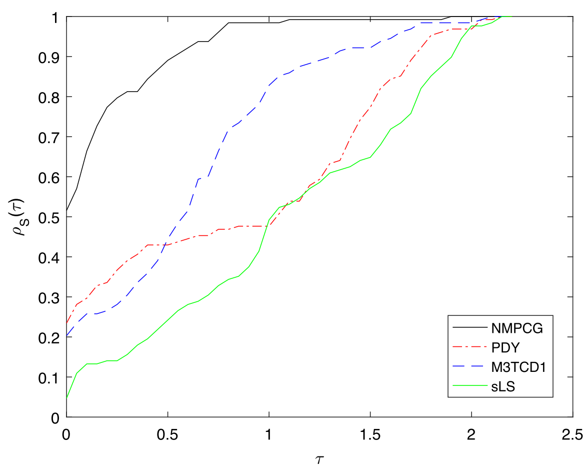

FIGURE 3. CPU time performance profile.

For a comprehensive performance of the methods, we use the performance profiles tool proposed by Dolan and Moré [10] to represent the results in Tables 1 and 2 in figures, based on the number of iterations, number of function evaluations, and CPU time. These are presented in Figures 1–3, respectively. With the Dolan and Moré performance profiles, the graph that is above the others for the most part is regarded as the best solver. We can see from Figure 1 that in terms of number of iterations, the $NMPCG$ and $M3TCD1$ methods are very competitive, and perform much better than both the PDY and sLS methods. Figure 2 shows clearly that in terms of function evaluations, the proposed $NMPCG$ method outperforms all the other methods, followed by the $M3TCD1$ method. Figure 3, for the CPU performance profiles, shows that the proposed $NMPCG$ is equally efficient.

6. Conclusion

In this paper, we proposed a new modified derivative-free Perry's conjugate gradient-based projection method for solving systems of large-scale nonlinear monotone equations. Its global convergence and rate of convergence were established. The proposed algorithm was tested on some benchmark problems, with different starting points and dimensions, and the numerical results show that the method is efficient as compared to other methods from the literature. Future work includes extending the new method to solve other kinds of problems like robotic motion control, signal restoration and image deblurring.

Acknowledgements. The authors are grateful to the Editor and the anonymous referees for their valuable suggestions and comments that improved the quality of this work.

References

[1] A.B. Abubakar and P. Kumam, A descent Dai-Liao conjugate gradient method for nonlinear equations. *Numer. Algor.* **81** (2019) 197–210.

[2] A.B. Abubakar, J. Rilwan, S.E. Yimer, A.H. Ibrahim and I. Ahmed, Spectral three-term conjugate descent method for solving nonlinear monotone equations with convex constraints. *Thai J. Math.* **18** (2020) 501–517.

[3] N. Andrei, Scaled conjugate gradient algorithms for unconstrained optimization. *Comput. Optim. Appl.* **38** (2007) 401–416.

[4] A.M. Awwal, P. Kumam, H. Mohammad, W. Watthayu and A.B. Abubakar, A Perry-type derivative-free algorithm for solving nonlinear system of equations and minimizing l_1 regularized problem. *Optimization* **70** (2021) 1231–1259.

[5] Y.H. Dai and L.Z. Liao, New conjugacy conditons and related nonlinear conjugate gradient methods. *Appl. Math. Optim.* **43** (2001) 87–101.

[6] Y.H. Dai, Y.K. Huang and X.W. Liu, A family of spectral gradient methods for optimization. *Comput. Optim. Appl.* **74** (2019) 43–65.

[7] Z. Dai, X. Chen and F. Wen, A modified Perry's conjugate gradient method-based derivative-free method for solving large-scale nonlinear monotone equations. *Appl. Math. Comput.* **270** (2015) 378–386.

[8] Y. Ding, Y. Xiao and J. Li, A class of conjugate gradient methods for convex constrained monotone equations. *Optimization* **66** (2017) 2309–2328.

[9] S.P. Dirkse and M.C. Ferris, A collection of nonlinear mixed complementarity problems. *Optim. Math. Softw.* **5** (1995) 319–345.

[10] E.D. Dolan and J.J. More, Benchmarking optimization software with performance profiles. *Math. Program.* **91** (2002) 201–213.

[11] P. Gao, C. He and Y. Liu, An adaptive family of projection methods for constrained monotone equations with applications. *Appl. Math. Comput.* **359** (2019) 1–16.

[12] A.S. Halilu, A. Majumder, M.Y. Waziri and K. Ahmed, Signal recovery with convex constrained nonlinear monotone equations through conjugate gradient hybrid approach. *Math. Comput. Simulat.* **187** (2021) 520–539.

[13] P. Kaelo and M. Koorapetse, A global convergent projection method for systems of nonlinear monotone equations. *Int. J. Comput. Math.* **98** (2021) 421–434.

[14] M. Koorapetse, P. Kaelo and E.R. Offen, A scaled derivative-free projection method for solving nonlinear monotone equations. *Bull. Iran. Math. Soc.* **45** (2019) 755–770.

[15] M. Koorapetse, P. Kaelo, S. Lekoko and T. Diphofu, A derivative-free RMIL conjugate gradient projection method for convex constrained nonlinear monotone equations with applications in compressive sensing. *Appl. Numer. Math.* **165** (2021) 431–441.

[16] J. Liu and Y. Feng, A derivative-free iterative method for nonlinear monotone equations with convex constraints. *Numer. Algor.* **82** (2019) 245–262.

[17] P. Liu, J. Jian and X. Jiang, A new conjugate gradient projection method for convex constrained nonlinear equations. *Complexity* **2020** (2020) 1–14.

[18] J.K. Liu, J.L. Xu and L.Q. Zhang, Partially symmetrical derivative-free Liu Storey projection method for convex constrained equations. *Int. J. Comput. Math.* **96** (2019) 1787–1798.
[19] I.E. Livieris and P. Pintelas, Globally convergent modified Perry's conjugate gradient method. *Appl. Math. Comput.* **218** (2012) 9197–9207.
[20] K. Meintjes and A.P. Morgan, A methodology for solving chemical equilibrium systems. *Appl. Math. Comput.* **22** (1987) 333–361.
[21] Y. Ou and J. Li, A new derivative-free SCG-type projection method for nonlinear monotone equations with convex constraints. *J. Appl. Math. Comput.* **56** (2018) 195–216.
[22] A. Perry, A modified conjugate gradient algorithm. *Oper. Res.* **26** (1978) 1073–1078.
[23] J. Sabi'u, A. Shah and M.Y. Waziri, Two optimal Hager-Zhang conjugate gradient methods for solving monotone nonlinear equations. *Appl. Numer. Math.* **153** (2020) 217–233.
[24] D. Shanno, On the convergence of a new conjugate gradient algorithm. *SIAM J. Numer. Anal.* **15** (1978) 1247–1257.
[25] M.V. Solodov and B.F. Svaiter, A Globally Convergent Inexact Newton Method for Systems of Monotone Equations, edited by M. Fukushima, L. Qi. In: Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods, Vol. 22 of *Applied Optimization*. Springer, Boston, MA (1998), 355–369.
[26] M. Sun and J. Liu, New hybrid conjugate gradient projection method for the convex constrained equations. *Calcolo* **53** (2016) 399–411.
[27] M.Y. Waziri, K. Ahmed and J. Sabi'u, A Dai-Liao conjugate gradient method via modified secant equation for system of nonlinear equations. *Arab. J. Math.* **9** (2020) 443–457.
[28] M.Y. Waziri, K. Ahmed, J. Sabi'u and A.S. Halilu, Enhanced Dai-Liao conjugate gradient methods for systems of nonlinear equations. *SeMA J.* **78** (2021) 15–51.
[29] Z. Wei, G. Li and L. Qi, New quasi-Newton methods for unconstrained optimization problems. *Appl. Math. Comput.* **175** (2006) 1156–1188.
[30] S. Yao and L. Ning, An adaptive three-term conjugate gradient method based on self-scaling memoryless BFGS matrix. *J. Comput. Appl. Math.* **332** (2018) 72–85.
[31] Q.-R. Yan, X.-Z. Peng and D.-H. Li, A globally convergent derivative-free method for solving large-scale nonlinear monotone equations. *J. Comput. Appl. Math.* **234** (2010) 649–657.
[32] L. Zhang, W.J. Zhou and D.H. Li, A descent modified Polak-Ribiére-Polyak conjugate gradient method and its global convergence. *IMA J. Numer. Anal.* **26** (2006) 629–640.
[33] L. Zheng, L. Yang and Y. Liang, A modified spectral gradient projection method for solving non-linear monotone equations with convex constraints and its application. *IEEE Acess* **99** (2020) 1–1.
[34] W.Y. Zhao and D. Li, Monotonicity of fixed point and normal mapping associated with varriational inequality and its application. *SIAM. J. Optim.* **11** (2000) 962–973.
[35] L. Zheng, L. Yang and Y. Liang, A conjugate gradient projection method for solving equations with convex constraints. *J. Comput. Appl. Math.* **375** (2020) 399–411.