AIMS Mathematics, 8(2): 4843–4861.
DOI: 10.3934/math.2023241
Received: 14 September 2022
Revised: 15 November 2022
Accepted: 05 December 2022
Published: 09 December 2022

http://www.aimspress.com/journal/Math

Research article

A scaled Polak-Ribi$\grave{e}$re-Polyak conjugate gradient algorithm for constrained nonlinear systems and motion control

Jamilu Sabi'u[1], Ali Althobaiti[2], Saad Althobaiti[3], Soubhagya Kumar Sahoo[4] and Thongchai Botmart[5,*]

[1] Department of Mathematics, Yusuf Maitama Sule University Kano, PMB 3220, Kano Nigeria

[2] Mathematics Department, College of Science, Taif University, P.O.Box 11099, Taif 21944, Saudi Arabia

[3] Department of Sciences and Technology, Ranyah University Collage, Taif University, P.O.Box 11099, Taif 21944, Saudi Arabia

[4] Department of Mathematics, Institute of Technical Education and Research, Siksha O Anusandhan University, Bhubaneswar 751030, Odisha, India

[5] Department of Mathematics, Faculty of Science, Khon Kaen University, Khon Kaen 40002, Thailand

* **Correspondence:** Email: thongbo@kku.ac.th.

Abstract: This paper proposes Polak-Ribi$\grave{e}$re-Polyak (PRP) conjugate gradient (CG) directions based on two efficient scaling strategies. The first scaling parameter is determined by approaching the quasi-Newton direction, and the second by utilizing the well-known Barzilai-Borwein approach. In addition, we proposed two directions that satisfy the sufficient descent criterion regardless of the line search strategy. The proposed directions lead to a matrix-free algorithm for solving monotone-constrained nonlinear systems. The proposed algorithm's global convergence analysis is presented using some underlying assumptions. Furthermore, a detailed numerical comparison with other existing algorithms revealed that the proposed algorithm is both efficient and effective. Finally, the proposed technique is applied to the motion control problem of a two-joint planar robotic manipulator.

Keywords: convex constrained monotone systems; scaled conjugate gradient method; sufficient descent condition; Barzilai-Borwein approach
Mathematics Subject Classification:90C26, 90C30

1. Introduction

Consider the constrained system

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

where the function $F : \Psi \longleftrightarrow \mathbb{R}^n$ is a continuous, monotone, and $\Psi \subset \mathbb{R}^n$ is nonempty, closed and convex. The monotonicity of F implies

$$(F(x) - F(y))^T(x - y) \geq 0, \qquad x, y \in \Psi. \tag{1.2}$$

This problem has applications in financial forecasting problems [1] and compressive sensing [2]. The Newton method [3], quasi-Newton methods [4], and conjugate gradient (CG) methods [5] are among the most often used approaches for solving it. When dealing with large-scale nonsmooth algebraic systems, CG methods are seen to be the most successful. The iterative technique for the CG algorithm is as follows:

$$x_0 \in \mathbb{R}^n \quad x_{k+1} = x_k + \alpha_k d_k, \tag{1.3}$$

the vector d_k is given by

$$d_k = -F_0, \quad k = 0, \qquad d_k = -F_k + \beta_k d_{k-1}, \quad k \geq 1, \tag{1.4}$$

where $F_k = F(x_k)$, and β_k is the CG formula that differentiates CG methods. One of the most efficient CG formulas with restarting feature is the Polak-Ribi$\grave{e}$re-Polyak (PRP) [6] formula defined by

$$\beta_k^{PRP} = \frac{F_{k+1}^T y_k}{\|F_k\|^2}, \tag{1.5}$$

where $y_k = F_{k+1} - F_k$. Despite the good features of the formula (1.5), it doesn't satisfy the sufficient descent condition (SDC). This opens up the possibility for future investigations on the modification of the PRP formula to satisfy the SDC. There are different PRP algorithms for solving the general monotone system of nonlinear equations in the literature. The first PRP algorithm for solving monotone algebraic systems was proposed by Cheng [7] where he extended the default PRP using the hyperplane technique discussed in [8]. Subsequently, Yu [9] proposed the derivative-free PRP for a large-scale monotone system that uses the backtracking line search procedure. Ahookhosh [10] proposed a descent three-term PRP algorithm for solving large-scale nonlinear monotone equations that combine extended PRP formula with the projection technique. Yuan and Zhang [11] proposed another descent-modified PRP CG algorithm for monotone nonlinear equations. Sabi'u and Gadu [12]proposed a projection-based hybrid FR and PRP algorithm for the monotone system. Others are; the improved three-term PRP CG method [13], spectral modified PolakRibirePolyak projection conjugate gradient method for solving monotone systems of nonlinear equations [14], the new hybrid algorithm for solving large-scale monotone nonlinear equations [15], and the accelerated CG algorithm for solving nonlinear monotone equations and image restoration problems [16].

Moreover, for the convex-constrained system (1.1), Jinkui [17] proposed a spectral PRP projection algorithm for solving nonlinear monotone equations with convex constraints. Followed by; the projection-based PRP-like algorithm [18], the modified spectral PRP conjugate gradient projection method for solving large-scale monotone equations and its application in compressed sensing [19],

the new hybrid prpfr CG method for solving nonlinear monotone equations and image restoration problems [20], the modified PRP CG projection method for solving large scale nonlinear convex constrained monotone equations [21], the modified PRP CG method with weaker conditions [22], the PRP-like algorithm for monotone operator equations [23], and the modified PRP-type conjugate gradient projection algorithm for solving large-scale monotone nonlinear equations with convex constraint [24]. In this work, we will propose a scaled PRP CG gradient algorithm for constrained nonlinear systems in which the scaling parameters are determined by approaching the quasi-Newton direction and exploiting the advantages of the well-known Barzilai-Borwein technique. The appealing features of our algorithm are: (i) derivative and matrix-free algorithm, (ii) exploiting best out of the PRP formula numerically and theoretically in contrast to conventional scaled CG algorithms, (iii) the application of the proposed algorithm in solving the two-joint planar robotic manipulator problems, and finally, (iv) the proposed directions will satisfy the SDC independent of the line search procedure.

The next section will discuss the proposed algorithm using the scaling strategy, Section 3 contains the global convergence of the method. The numerical experiment will be presented in Section 4, and the last section will contain the concluding remarks.

2. A scaled PRP CG method

The scaling strategy is proved to be an efficient strategy for enhancing numerical and theoretical performances of the CG methods, see [25, 26]. In line with this, we scaled the PRP CG (SPRP) formula to get

$$\beta_k^{SPRP} = \gamma \beta_k^{PRP}, \tag{2.1}$$

such that $|\gamma| \leq 1$. We are going to propose two effective ways of computing the parameter γ at each iteration.

2.1. Based on quasi-Newton method

We are going to propose an optimal choice for the scaling parameter γ at every iteration by tending the proposed direction to approach the general quasi-Newton direction. Starting with the quasi-Newton equation

$$y_k = B_{k+1} s_k, \tag{2.2}$$

where B_{k+1} is a positive definite and symmetric Jacobian approximate. Now, the general quasi-Newton direction is as follows:

$$d_{k+1} = -H_{k+1} F_{k+1}, \tag{2.3}$$

where the matrix H_{k+1} is the inverse of B_{k+1}. Now, using (1.4) and (2.1), the SPRPR search direction can be written as

$$d_{k+1} = -F_{k+1} + \beta_k^{SPRP} d_k, \quad k = 0, 1, \cdots. \tag{2.4}$$

In order to incorporate Jacobian approximation information in the quasi-Newton direction into our scheme, since the quasi-Newton schemes employ the actual Jacobian approximations, from (2.3), (2.4), and also by the virtue of equality relation, we get

$$-H_{k+1} F_{k+1} = -F_{k+1} + \gamma \frac{F_{k+1}^T y_k}{\|F_k\|^2} d_k. \tag{2.5}$$

Multiplying (2.5) by $-B_{k+1}s_k^T$ to obtain

$$s_k^T F_{k+1} = B_{k+1}s_k^T F_{k+1} - \gamma \frac{F_{k+1}^T y_k}{\|F_k\|^2} B_{k+1}s_k^T d_k. \tag{2.6}$$

Solving for γ after eliminating the matrix B_{k+1} in (2.6) by using (2.2) to get

$$\gamma_k = \frac{(y_k - s_k)^T F_{k+1}}{\beta_k^{PRP} y_k^T d_k}. \tag{2.7}$$

We further achieved $\left|\gamma_k^1\right| \leq 1$ by using the Polak-Ribi$\grave{e}$re-Polyak non-negative restriction to obtain

$$\gamma_k^1 = \min\{1, |\gamma_k|\}. \tag{2.8}$$

2.2. Based on Barzilai-Borwein approach

We will again propose another optimal choice for the scaling parameter γ by using some prominent features of the Barzilai-Borwein approach. The Barzilai-Borwein [27] proposed some prevalent choices for the scaling parameters given by

$$\omega_k^1 = \frac{s_k^T s_k}{y_k^T s_k}, \quad \text{and} \quad \omega_k^2 = \frac{s_k^T y_k}{y_k^T y_k}. \tag{2.9}$$

By considering Perry's point of view, the scaled SPRP search direction can be written as

$$d_0 = -F_0, \qquad d_{k+1} = -\hat{H_{k+1}} F_{k+1}, \quad k = 0, 1, \cdots, \tag{2.10}$$

where $\hat{H_{k+1}}$ is defined by

$$\hat{H_{k+1}} = I - \gamma \frac{y_k d_k^T}{\|F_k\|^2}. \tag{2.11}$$

Now, we aim at taking the advantage of the Barzilai-Borwein [27] approach to propose the parameter γ by minimizing

$$\min_{\gamma} \left\| \hat{H_{k+1}} - \omega_k I \right\|_F^2, \tag{2.12}$$

where $\|.\|_F$ stands for the Frobenius matrix norm. Moreover, if we assigned $Q = \hat{H_{k+1}} - \omega_k I$ and utilized the relation $\|Q\|_F^2 = \text{Trace}(Q^T Q)$, we arrived at

$$\gamma_k = (1 - \omega_k) \frac{y_k^T d_k \|F_k\|^2}{\|y_k\|^2 \|d_k\|^2}, \tag{2.13}$$

In addition, we ensure that all of the prevalent choices for the Barzilai-Borwein scaling parameters are properly integrated into our scheme by suggesting that

$$\omega_k = \max\left\{\omega_{\min}, \min\left\{\omega_k^i, \omega_{\max}\right\}\right\}, \quad i = 1, 2, \tag{2.14}$$

with $0 < \omega_{\min} < \omega_{\max} < \infty$. We proposed the following modified version of (2.13) that satisfies $\left|\gamma_k^2\right| \leq 1$ as

$$\gamma_k^2 = \min\{1, |\gamma_k|\}. \tag{2.15}$$

Furthermore, to guarantee that the sufficient descent condition is fulfilled independent of the line search process, we provided the following modified SPRP directions

$$d_{k+1}^{SPRP1} = -\zeta_k^1 F_{k+1} + \gamma_k^1 \beta_k^{PRP} d_k, \quad k = 0, 1, \cdots, \tag{2.16}$$

and,

$$d_{k+1}^{SPRP2} = -\zeta_k^2 F_{k+1} + \gamma_k^2 \beta_k^{PRP} d_k, \quad k = 0, 1, \cdots, \tag{2.17}$$

where

$$\zeta_k^1 = 1 + \gamma_k^1 \beta_k^{PRP} \frac{F_{k+1}^T d_k}{\|F_{k+1}\|^2}, \quad \zeta_k^2 = 1 + \gamma_k^2 \beta_k^{PRP} \frac{F_{k+1}^T d_k}{\|F_{k+1}\|^2}. \tag{2.18}$$

Below we present the proposed algorithm.

Algorithm 2.1. *The scaled PRP projection-based CG algorithm (SPRPCG)*
Step 0. *Start by initializing:* $\epsilon \geq 0$, $b \in (0, 1)$, $\theta > 0$, $\tau > 0$, $a > 0$, $0 < \omega_{\min} \leq \omega_{\max}$ *and* $x_0 \in \mathbf{R^n}$. *Set* $k = 0$ *and* $d_0 = -F_0$.
Step 1. *If* $\|F_k\| \leq \epsilon$, *terminate, otherwise continue with Step 2.*
Step 2. *Determine the scaled PRP CG direction* d_k *via* (2.16) *or* (2.17), *where* $s_k = u_k - x_k$, *and* $y_k = F(u_k) - F_k + bs_k$.
Step 3. *Set* $u_k = x_k + \alpha_k d_k$ *and compute* $\alpha_k = \max\left\{\tau\theta^i : i = 0, 1, 2, \cdots\right\}$ *satisfying*

$$-F(u_k)^T d_k \geq a\alpha_k \|F(u_k)\| \|d_k\|^2 \tag{2.19}$$

Step 4. *If* $u_k \in \Psi$ *and* $\|F(u_k)\| = 0$ *stop, otherwise*

$$x_{k+1} = A_\Psi[x_k - v_k F(h_k)], \tag{2.20}$$

where A is the projection operator, and

$$v_k = \frac{F(u_k)^T(x_k - u_k)}{\|F(u_k)\|^2}. \tag{2.21}$$

Step 5. *Set* $k = k + 1$ *and repeat from* ***Step 1***.

3. Convergence analysis

This section describes the global convergence of the proposed algorithm under the Lipschitz continuous and monotonicity assumptions on F.

Lemma 3.1. *For all* $k \geq 0$, *the line search* (2.19) *is well-defined.*

Proof. Suppose there exists $k_0 \geq 0$ such that for $i = 0, 1, 2, \cdots$,

$$-F(x_{k_0} + \tau\theta^i d_{k_0})^T d_{k_0} < a\tau\theta^i \|F(x_{k_0} + \tau\theta^i d_{k_0})\| \|d_{k_0}\|^2. \tag{3.1}$$

Let $i \to \infty$ in (3.1), we have

$$-F(x_{k_0})^T d_{k_0} \leq 0. \tag{3.2}$$

Since the proposed directions satisfiy the sufficient condition, we have

$$-F(x_{k_0})^T d_{k_0} \geq \|F(x_{k_0})\|^2 > 0. \tag{3.3}$$

Thus, the relations (3.2) and (3.3) yield a contradiction. Hence, the proof is complete.

Lemma 3.2. *[25] If the sequences $\{x_k\}$ and $\{v_k\}$ are produced by the SPRPCG algorithm, then for some $M > 0$ we have*

$$\|F_k\| \leq M, \quad \lim_{k\to\infty} \alpha_k \|d_k\| = 0, \quad \forall k. \tag{3.4}$$

Proof. Now, from the line search (2.19), we have

$$F(u_k)^T(x_k - u_k) = -F(u_k)^T d_k \geq a\alpha_k^2 \|F(u_k)\|\|d_k\|^2 > 0. \tag{3.5}$$

Assume that $x^* \in \Omega$ such that $F(x^*) = 0$, using the monotonicity of F, we have

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

Using (3.5) and (3.6) to get

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &= \|A_\Psi(x_k - v_k F(u_k)) - x^*\|^2 \\ &\leq \|x_k - v_k F(u_k) - x^*\|^2 \\ &= \|x_k - x^*\|^2 - 2v_k F(u_k)^T(x_k - x^*) + v_k^2 \|F(u_k)\|^2. \end{aligned} \tag{3.7}$$

By the definition of v_k, and the Cauchy Schwarz inequality in (3.7), we obtain

$$\begin{aligned} \|x_{k+1} - x^*\|^2 &\leq \|x_k - x^*\|^2 - 2v_k F(u_k)^T(x_k - u_k) + v_k^2 \|F(u_k)\|^2 \\ &\leq \|x_k - x^*\|^2 - \frac{(F(u_k)^T(x_k - u_k))^2}{\|F(u_k)\|^2} \\ &\leq \|x_k - x^*\|^2 - a^2 \|x_k - u_k\|^4, \end{aligned} \tag{3.8}$$

thus, we have

$$\|x_{k+1} - x^*\| \leq \|x_k - x^*\|, \quad \forall\ k \geq 0. \tag{3.9}$$

Therefore, $\{\|x_k - x^*\|\}$ is a decreasing sequence and hence convergent. Now, utilizing (3.9) and the Lispchitz continuity of F, we get

$$\|F(x_k)\| = \|F(x_k) - F(x^*)\| \leq L\|x_k - x^*\| \leq L\|x_0 - x^*\| = M. \tag{3.10}$$

By (2.19), monotonicity of F, and the Cauchy Schwarz inequality, we obtain

$$a\|F(u_k)\|\|x_k - u_k\|^2 \leq F(u_k)^T(x_k - u_k) \leq \|F(u_k)\|\|x_k - u_k\|, \tag{3.11}$$

which gives

$$a\|x_k - u_k\| \leq 1. \tag{3.12}$$

From (3.12), we see that the sequence $\{u_k\}$ is bounded. Also, from (3.8), we obtain

$$a^2 \sum_{k=0}^{\infty} \|x_k - u_k\|^4 \leq \sum_{k=0}^{\infty} (\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2) < \infty. \tag{3.13}$$

This implies

$$\lim_{k\to\infty} \|x_k - u_k\| = \lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.14}$$

The proof is now complete.

Theorem 3.1. *The SPRPCG algorithm converges, i.e.,*

$$\liminf_{k\to\infty} \|F_k\| = 0. \tag{3.15}$$

Proof. Assume that (3.15) is not true, i.e., there exists $q > 0$ such that

$$\|F_k\| \geq q, \quad \forall k \in \mathbb{N}. \tag{3.16}$$

Again from the Lipschitz continuinity of F it is easy to see that $\|y_k\| \leq (L + a)\|s_k\| = N$, where L is assumed to be the Lipschitz constant. Thus,

$$\begin{aligned} \|d_{k+1}\| &= \left\|-\zeta_k^1 F_{k+1} + \gamma_k^1 \beta_k^{PRP} d_k\right\| \\ &= \left\|-\left(1 + \gamma_k^1 \beta_k^{PRP} \frac{F_{k+1}^T d_k}{\|F_{k+1}\|^2}\right) F_{k+1} + \gamma_k^1 \beta_k^{PRP} d_k\right\| \\ &\leq \|F_{k+1}\| + 2 + \frac{\|F_{k+1}\|\,\|y_k\|}{q^2}\|d_k\| \\ &\leq M + 2\frac{MN}{\left\|q^2\right\|}\|d_k\|. \end{aligned} \tag{3.17}$$

Now, let $h = 2\frac{MN}{\left\|q^2\right\|}$, therefore from (3.17) we get

$$\begin{aligned} \|d_{k+1}\| &\leq M + h\|d_k\| \\ &\leq M(1 + h + h^2 + \cdots + h^{k-k_0+1})\left\|d_{k_0}\right\| \\ &\leq \frac{M}{1-h} + \left\|d_{k_0}\right\|. \end{aligned} \tag{3.18}$$

Hence, for $Z = \max\{\|d_1\|, \|d_2\|, \cdots, \left\|d_{k_0}\right\|, \frac{M}{1-h} + \left\|d_{k_0}\right\|\}$, we obtain the boundedness of the proposed direction. Similarly, we can show that (2.17) is also bounded. From the sufficient condition and the Cauchy-Schwarz inequality we get

$$\|F_k\|\,\|d_k\| \geq -F_k d_k \geq \|F_k\|^2 > 0. \tag{3.19}$$

From (3.19) we obtain

$$\|d_k\| \geq \|F_k\| > q. \tag{3.20}$$

Therefore, from (3.14) and (3.20) we have

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.21}$$

Suppose α_k' doesn't safies the line search procedure (2.19), i.e.,

$$-F(x_k + \alpha_k' d_k)^T d_k < a\alpha_k' \left\|F(x_k + \alpha_k' d_k)\right\| \|d_k\|^2, \tag{3.22}$$

It is obvious that $\{x_k\}$ is bounded and has an accumulation point $\hat{x}$ and an infinite index set D_1 such that $\lim_{k\in D_1} x_k = \hat{x}$. It also follows that $\{d_k\}_{k\in D_1}$ is bounded. Therefore, there exist an infinite index set $D_2 \subset D_1$ with an accumulation point $\hat{d}$ such that $\lim_{k\in D_2} d_k = \hat{d}$. Now, taking limit in (3.22), we obtain

$$-F(\hat{x})^T \hat{d} \leq 0. \tag{3.23}$$

Also taking the limit in (3.19), we get

$$-F(\hat{x})^T \hat{d} \geq 0. \tag{3.24}$$

The last two inequalities give a contradiction and the proof is complete.

4. Numerical experiment

The proposed scaled PRP algorithm is numerically compared to the MPRPA [24], DFSP [26], and STCGM [25] algorithms, using published values in the compared papers. Furthermore, we initialized the following variables for the SPRPCG algorithm: $b = 0.2$, $a = 0.0001$, $\theta = 0.99$, $\tau = 1$, $\omega_{\min} = 0.0001$, $\omega_{\max} = 10^4$, $\Psi = R^n_+$, and $\epsilon = 10^{-10}$. Matlab14 is used to run all algorithms on an Intel Core i5 8th Generation personal computer. We carried out our experiment using the following test problems:

Problem 4.1. *Reference [25]* $F(x_i) = 2x_i - \sin|x_i|$, *for* $i = 1, 2, 3, \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 4.2. *Reference* [25] $F_i(x) = 4x_i + (x_{i+1} - 2x_i) - \frac{x^2_{i+1}}{3}$, *for* $i = 1, 2, \ldots, n-1$, $F_n(x) = 4x_n + (x_{n-1} - 2x_n) - \frac{x^2_{n-1}}{3}$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 4.3. *Reference* [26] $F_i(x) = e^{x_i} - 1, \quad i = 1, 2, \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$.

Problem 4.4. *Reference [25]* $F_1(x) = \cos(x_1) - 9 + 3x_1 + 8\exp(x_2)$ $F_i(x) = \cos(x_i) - 9 + 3x_i + 8\exp(x_{i-2})$, *for* $i = 2, 3, \ldots, n$, *and* $\Omega = \mathbf{R}^n_+$

Problem 4.5. *Reference [25]* $F_1(x) = e^{x_1} - 1$, $F_i(x) = e^{x_i} + x_{i-1} - 1, \quad i = 2, 3, \cdots, n-1$, *and* $\Omega = \mathbf{R}^n_+$.

From Tables 1–5, we used eight initial points, namely; $x^1_0 = (2, 2, \ldots, 2)$, $x^2_0 = (1, \frac{1}{2}, \frac{1}{3}, \ldots, \frac{1}{n})$, $x^3_0 = (1, 1, \ldots, 1)$, $x^4_0 = (\frac{1}{n}, \frac{2}{n}, \ldots, 1)$, $x^5_0 = (n - \frac{1}{n}, n - \frac{2}{n}, \ldots, n - 1)$, $x^6_0 = (2, \frac{2}{2}, \frac{2}{3}, \ldots, \frac{2}{n})$, $x^7_0 = (1 - 1, 1 - \frac{1}{2}, 1 - \frac{1}{3}, \ldots, 1 - \frac{1}{n})$ and $x^8_0 = (-3, -3, \ldots, -3)$. In which the terms ITER, FVAL, TIME, and NORM stand for the number of iterations, function evaluation, computing time and the norm of F_k at the stopping criteria respectively.

When compared to the MPRP, STCGM, and DFSP algorithms, the proposed approach solved the Problem 4.1 with the lowest number of ITER and FVAL. However, for the computing time, the MPRP algorithm won initially and failed to STCGM as the dimension increases. Likewise for Problems 4.2–4.5. Moreover, it is remarkable to note that the MPRP algorithm is competing with the STCGM algorithm for computing time with respect to all the considered test problems. Finally, the proposed algorithm outperforms the MPRPA, STCGM, and DFSP algorithms for ITER, and FVAL, but competes with the STCGM algorithm with respect to TIME.

Furthermore, we used the well-known Dolan and Moré [28] performance profile to produce a more exact numerical comparison using the bulk data in all the tables. Figure 1 three subfigures indicate that the SPRPCG algorithm with two directions reflects the upper left curves with regards to 1a and 1b. However, for the 1c, the SPRPCG initially won in the fractions of 0 to approximately 0.9, and later failed to STCGM algorithm. As a result, when compared to the MPRPA, STCGM, and DFSP algorithms, the SPRPCG algorithm has fewer iterations, and function evaluations on average and competes with the STCGM algorithm for CPU time.

Table 1. Numerical comparison of SPRPCG algorithm versus DFSP [26], STCGM [25] and MPRPA algorithms [24].

Problem 4.1		SPRPCG(1)				SPRPCG(2)				MPRPA				DFSP				STCGM			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.005949	0	1	14	0.005133	0	35	71	0.02165	9.62E-11	23	49	0.027654	7.9E-11	8	25	0.00943	6.42E-11
	x_0^2	2	27	0.007886	0	2	27	0.007942	0	22	189	0.033029	5.1E-11	6	56	0.013212	0	5	53	0.008826	0
	x_0^3	1	14	0.0054	0	1	14	0.006263	0	37	75	0.023115	6.37E-11	22	46	0.015948	6.25E-11	8	20	0.010007	7.57E-12
	x_0^4	2	27	0.007635	0	2	27	0.007432	0	27	221	0.039841	5.48E-11	5	43	0.010364	0	17	136	0.123572	0
	x_0^5	2	27	0.00817	0	2	27	0.007754	0	27	218	0.042077	9E-11	6	56	0.013789	0	17	130	0.023084	0
	x_0^6	2	27	0.008051	0	2	27	0.007844	0	24	206	0.086541	5.98E-11	6	57	0.01194	0	8	25	0.012607	9.09E-11
	x_0^7	2	27	0.007648	0	2	27	0.007856	0	27	218	0.044937	9E-11	6	56	0.013552	0	5	53	0.010073	0
	x_0^8	2	27	0.008063	0	2	27	0.009222	0	1	4	0.004576	0	1	6	0.004244	0	8	20	0.011038	1.07E-11
1000	x_0^1	1	14	0.007079	0	1	14	0.006871	0	36	73	0.027915	6.73E-11	24	51	0.021346	3.39E-11	17	135	0.029611	0
	x_0^2	2	27	0.009849	0	2	27	0.009229	0	21	185	0.055912	7.33E-11	6	56	0.016773	0	18	138	0.035786	0
	x_0^3	1	14	0.006306	0	1	14	0.006283	0	37	75	0.030669	9.01E-11	22	46	0.02162	8.88E-11	9	27	0.048078	5.63E-12
	x_0^4	2	27	0.010302	0	2	27	0.009833	0	11	83	0.027815	0	6	56	0.018072	0	5	53	0.082559	0
	x_0^5	2	27	0.010085	0	2	27	0.01037	0	11	83	0.023342	0	6	57	0.017067	0	8	20	0.039123	3.39E-11
	x_0^6	2	27	0.00958	0	2	27	0.012111	0	24	203	0.055708	8.24E-11	6	57	0.01826	0	18	144	0.223104	0
	x_0^7	2	27	0.009673	0	2	27	0.00937	0	11	83	0.027194	0	6	57	0.017675	0	18	143	0.12509	0
	x_0^8	2	27	0.011959	0	2	27	0.010101	0	1	4	0.004386	0	1	6	0.005709	0	9	27	0.212339	1.26E-11
10000	x_0^1	1	14	0.026098	0	1	14	0.025464	0	38	77	0.200128	5.22E-11	25	53	0.111401	3.45E-11	5	53	0.200085	0
	x_0^2	2	27	0.047774	0	2	27	0.041911	0	11	98	0.153287	0	6	56	0.089864	0	8	20	0.164896	7.57E-11
	x_0^3	1	14	0.025732	0	1	14	0.02334	0	39	79	0.246965	6.98E-11	23	48	0.096345	8.71E-11	18	144	0.601218	0
	x_0^4	2	27	0.044049	0	2	27	0.042824	0	14	110	0.20247	0	6	56	0.091267	0	19	151	1.020748	0
	x_0^5	2	27	0.047091	0	2	27	0.044061	0	14	110	0.195937	0	6	56	0.087251	0	9	27	0.268154	1.78E-11
	x_0^6	2	27	0.04558	0	2	27	0.042455	0	24	202	0.328184	8.06E-11	6	57	0.086451	0	5	53	0.831941	0
	x_0^7	2	27	0.045994	0	2	27	0.046985	0	14	110	0.201035	0	6	56	0.087745	0	9	22	0.299434	2.1E-12
	x_0^8	2	27	0.047422	0	2	27	0.048712	0	1	4	0.009599	0	1	6	0.014179	0	19	150	2.261945	0
50000	x_0^1	1	14	0.096279	0	1	14	0.101904	0	39	79	0.767104	5.77E-11	25	53	0.437284	8.8E-11	19	152	1.872535	0
	x_0^2	2	27	0.185007	0	2	27	0.184709	0	11	98	0.69978	0	6	56	0.374874	0	8	25	0.003892	7.04E-12
	x_0^3	1	14	0.095243	0	1	14	0.097266	0	40	81	0.769044	7.73E-11	24	50	0.425448	6.2E-11	5	52	0.006669	0
	x_0^4	2	27	0.389746	0	2	27	0.178131	0	16	127	0.929816	0	5	43	0.318514	0	7	18	0.00442	4.23E-11
	x_0^5	2	27	0.36199	0	2	27	0.189434	0	16	127	0.992202	0	5	43	0.308814	0	6	68	0.009472	0
	x_0^6	2	27	0.329928	0	2	27	0.178671	0	24	205	1.581707	6.03E-11	6	57	0.385743	0	5	51	0.00544	0
	x_0^7	2	27	0.295669	0	2	27	0.185862	0	16	127	0.892216	0	5	43	0.296815	0	8	25	0.007498	7.6E-12
	x_0^8	2	27	0.282256	0	2	27	0.201677	0	1	4	0.039697	0	1	6	0.046323	0	5	52	0.00674	0
100000	x_0^1	1	14	0.292199	0	1	14	0.187861	0	39	79	1.65231	8.17E-11	26	55	0.911281	4.11E-11	7	18	0.004102	4.57E-11
	x_0^2	2	27	0.454309	0	2	27	0.376562	0	11	98	1.488625	0	6	56	0.725688	0	2	18	0.005527	0
	x_0^3	1	14	0.237757	0	1	14	0.187526	0	41	83	1.776357	5.41E-11	24	50	0.861569	9.16E-11	5	51	0.005885	0
	x_0^4	2	27	0.471781	0	2	27	0.397482	0	16	126	1.741981	0	5	43	0.589476	0	8	25	0.00696	8.13E-12
	x_0^5	2	27	0.462599	0	2	27	0.370585	0	17	139	1.953327	0	5	43	0.59129	0	5	52	0.003811	0
	x_0^6	2	27	0.453023	0	2	27	0.376696	0	24	204	2.711584	6.75E-11	6	57	0.760319	0	7	18	0.006613	4.88E-11
	x_0^7	2	27	0.48348	0	2	27	0.362695	0	17	139	1.840027	0	5	43	0.583836	0	2	18	0.005547	0
	x_0^8	2	27	0.495025	0	2	27	0.42318	0	1	4	0.076396	0	1	6	0.092276	0	6	60	0.006736	0

Table 2. Numerical comparison of SPRPCG algorithm versus DFSP [26], STCGM [25], and MPRPA algorithms [24].

Problem 4.2		SPRPCG(1)				SPRPCG(2)				MPRPA				DFSP				STCGM			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.00616	0	1	14	0.006448	0	3	30	0.009564	0	9	85	0.024533	9.27E-11	1	14	0.007168	0
	x_0^2	1	14	0.006193	0	1	14	0.005646	0	2	18	0.007262	0	6	71	0.01834	0	2	27	0.00951	0
	x_0^3	1	14	0.005835	0	1	14	0.00614	0	3	32	0.009654	0	9	85	0.019681	4.01E-11	1	14	0.005623	0
	x_0^4	1	14	0.006187	0	1	14	0.006602	0	4	45	0.011538	0	5	58	0.017104	0	1	14	0.007012	0
	x_0^5	1	14	0.005661	0	1	14	0.006467	0	5	58	0.015613	0	5	58	0.016217	0	2	27	0.004613	0
	x_0^6	1	14	0.005917	0	1	14	0.006679	0	2	18	0.007484	0	6	71	0.018448	0	1	14	0.006543	0
	x_0^7	1	14	0.005769	0	1	14	0.006531	0	5	58	0.015084	0	5	58	0.016812	0	2	27	0.006565	0
	x_0^8	1	3	0.004172	0	1	3	0.004642	0	1	9	0.006094	0	1	3	0.0038	0	1	14	0.007489	0
1000	x_0^1	1	14	0.006684	0	1	14	0.007853	0	3	30	0.012287	0	10	96	0.034862	4.34E-12	1	14	0.005651	0
	x_0^2	1	14	0.00862	0	1	14	0.007312	0	4	45	0.017093	0	6	71	0.026176	0	2	27	0.011575	0
	x_0^3	1	14	0.007406	0	1	14	0.00786	0	3	32	0.013344	0	9	85	0.030241	5.69E-11	1	14	0.032168	0
	x_0^4	1	14	0.008181	0	1	14	0.008113	0	4	45	0.017142	0	5	58	0.021131	0	2	27	0.050307	0
	x_0^5	1	14	0.007682	0	1	14	0.007276	0	5	58	0.020869	0	5	58	0.022127	0	1	14	0.033315	0
	x_0^6	1	14	0.008416	0	1	14	0.008252	0	2	18	0.009431	0	6	71	0.02732	0	1	14	0.01479	0
	x_0^7	1	14	0.008132	0	1	14	0.007605	0	5	58	0.01976	0	5	58	0.021743	0	2	27	0.061964	0
	x_0^8	1	3	0.004654	0	1	3	0.004176	0	1	9	0.006268	0	1	3	0.004233	0	1	14	0.091229	0
10000	x_0^1	1	14	0.034062	0	1	14	0.034897	0	3	30	0.068186	0	10	96	0.221348	1.84E-11	2	27	0.285117	0
	x_0^2	1	14	0.035571	0	1	14	0.034198	0	4	45	0.097955	0	6	71	0.156478	0	1	14	0.092376	0
	x_0^3	1	14	0.034545	0	1	14	0.03343	0	3	32	0.071271	0	10	96	0.214021	5.82E-12	1	14	0.15221	0
	x_0^4	1	14	0.033543	0	1	14	0.033777	0	4	45	0.095412	0	5	58	0.134208	0	2	27	0.185719	0
	x_0^5	1	14	0.034247	0	1	14	0.033997	0	5	58	0.123651	0	5	58	0.126169	0	1	14	0.350673	0
	x_0^6	1	14	0.033179	0	1	14	0.033604	0	2	18	0.040918	0	6	71	0.152915	0	2	27	0.502155	0
	x_0^7	1	14	0.034517	0	1	14	0.032933	0	5	58	0.125501	0	5	58	0.127013	0	1	14	0.250523	0
	x_0^8	1	3	0.01073	0	1	3	0.010852	0	1	9	0.022272	0	1	3	0.011772	0	1	14	0.308355	0
50000	x_0^1	1	14	0.144173	0	1	14	0.146136	0	3	30	0.323007	0	10	96	1.002645	6.02E-11	2	27	0.491642	0
	x_0^2	1	14	0.144648	0	1	14	0.145859	0	4	45	0.453818	0	6	71	0.746929	0	1	14	0.004396	0
	x_0^3	1	14	0.144283	0	1	14	0.147213	0	3	32	0.319894	0	10	96	0.997627	1.36E-11	2	27	0.008702	0
	x_0^4	1	14	0.142842	0	1	14	0.139717	0	4	45	0.481684	0	5	58	0.609387	0	1	14	0.004268	0
	x_0^5	1	14	0.142552	0	1	14	0.143273	0	5	58	0.585424	0	5	58	0.598987	0	1	14	0.005546	0
	x_0^6	1	14	0.144208	0	1	14	0.141715	0	2	18	0.18099	0	6	71	0.737116	0	2	27	0.003315	0
	x_0^7	1	14	0.141234	0	1	14	0.167606	0	5	58	0.603428	0	5	58	0.605638	0	1	14	0.004384	0
	x_0^8	1	3	0.03565	0	1	3	0.036751	0	1	9	0.092186	0	1	3	0.036785	0	2	27	0.003942	0
100000	x_0^1	1	14	0.288466	0	1	14	0.281604	0	3	30	0.623989	0	11	107	2.484277	3.3E-12	1	14	0.004815	0
	x_0^2	1	14	0.281627	0	1	14	0.281598	0	4	45	0.98394	0	6	71	1.63692	0	1	14	0.002507	0
	x_0^3	1	14	0.28431	0	1	14	0.293397	0	3	32	0.670524	0	10	96	2.212571	1.98E-11	2	27	0.007047	0
	x_0^4	1	14	0.274463	0	1	14	0.279905	0	4	45	0.980961	0	5	58	1.297317	0	1	14	0.002786	0
	x_0^5	1	14	0.289894	0	1	14	0.289534	0	5	58	1.262739	0	5	58	1.323224	0	2	27	0.007023	0
	x_0^6	1	14	0.288962	0	1	14	0.289889	0	2	18	0.437525	0	6	71	1.624833	0	1	14	0.002699	0
	x_0^7	1	14	0.284459	0	1	14	0.286553	0	5	58	1.210862	0	5	58	1.331419	0	1	14	0.005008	0
	x_0^8	1	3	0.068662	0	1	3	0.066155	0	1	9	0.191548	0	1	3	0.073259	0	2	27	0.003077	0

Table 3. Numerical comparison of SPRPCG algorithm versus DFSP [26], STCGM [25], and MPRPA algorithms [24].

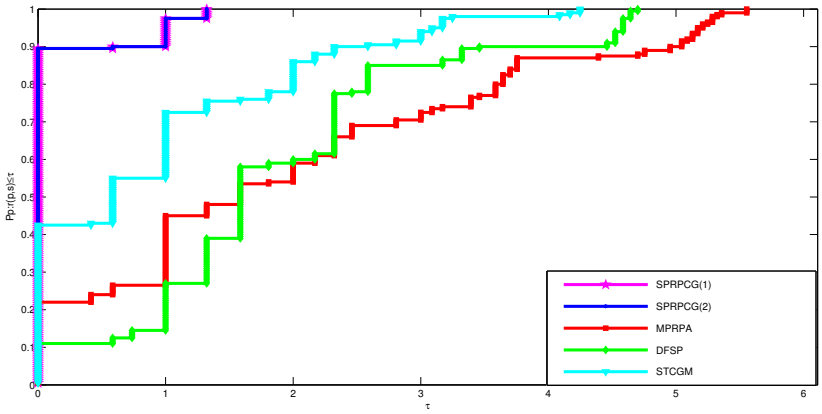
Problem 4.3		SPRPCG(1)				SPRPCG(2)				MPRPA				DFSP				STCGM			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.005065	0	1	14	0.005702	0	33	76	0.019308	5.51E-11	23	58	0.020183	3.02E-11	1	14	0.004557	0
	x_0^2	2	27	0.007402	0	2	27	0.007827	0	21	163	0.027816	6.91E-11	6	66	0.013712	0	4	50	0.009725	0
	x_0^3	1	14	0.006092	0	1	14	0.006609	0	31	63	0.017096	6.17E-11	23	49	0.016392	3.01E-11	8	26	0.008178	1.33E-11
	x_0^4	2	27	0.008335	0	2	27	0.00744	0	24	179	0.031604	5.04E-11	5	46	0.011696	0	4	49	0.011172	0
	x_0^5	2	27	0.007787	0	2	27	0.007068	0	25	180	0.032925	3.7E-11	6	60	0.013848	0	4	49	0.011364	0
	x_0^6	2	27	0.007486	0	2	27	0.008	0	9	69	0.013732	0	5	50	0.011986	0	1	14	0.006459	0
	x_0^7	2	27	0.007729	0	2	27	0.008037	0	25	180	0.029996	3.7E-11	6	60	0.012398	0	4	50	0.012351	0
	x_0^8	1	3	0.004163	0	1	3	0.003858	0	1	3	0.003841	0	1	3	0.003691	0	8	26	0.005912	1.88E-11
1000	x_0^1	1	14	0.006594	0	1	14	0.006499	0	33	76	0.025403	7.79E-11	23	58	0.022122	4.28E-11	4	49	0.012041	0
	x_0^2	2	27	0.00899	0	2	27	0.008246	0	21	161	0.0376	6.85E-11	6	66	0.017943	0	4	49	0.00792	0
	x_0^3	1	14	0.006662	0	1	14	0.005875	0	31	63	0.022674	8.72E-11	23	49	0.021127	4.27E-11	1	14	0.02137	0
	x_0^4	2	27	0.009007	0	2	27	0.009575	0	25	179	0.041317	5.58E-11	5	46	0.014358	0	4	50	0.06797	0
	x_0^5	2	27	0.009047	0	2	27	0.008923	0	25	180	0.042117	4.65E-11	6	59	0.016748	0	8	26	0.039395	5.95E-11
	x_0^6	2	27	0.00939	0	2	27	0.009113	0	9	69	0.018797	0	5	50	0.014086	0	4	49	0.057746	0
	x_0^7	2	27	0.009503	0	2	27	0.009876	0	25	180	0.041626	4.65E-11	6	59	0.015995	0	4	49	0.041586	0
	x_0^8	1	3	0.004261	0	1	3	0.004402	0	1	3	0.003657	0	1	3	0.003832	0	1	14	0.072798	0
10000	x_0^1	1	14	0.023061	0	1	14	0.023074	0	35	80	0.129582	6.03E-11	24	60	0.097674	4.11E-11	4	50	0.164054	0
	x_0^2	2	27	0.035898	0	2	27	0.037322	0	21	164	0.216364	5.35E-11	6	66	0.08125	0	9	28	0.160535	2.63E-12
	x_0^3	1	14	0.021199	0	1	14	0.021082	0	33	67	0.10905	6.76E-11	24	51	0.085409	4.15E-11	4	49	0.196973	0
	x_0^4	2	27	0.037109	0	2	27	0.037459	0	26	189	0.235984	4.6E-11	6	59	0.076381	0	4	49	0.249326	0
	x_0^5	2	27	0.036095	0	2	27	0.036294	0	26	190	0.221085	4.43E-11	6	59	0.074121	0	1	14	0.093946	0
	x_0^6	2	27	0.036085	0	2	27	0.03981	0	9	69	0.085623	0	5	50	0.06211	0	4	50	0.543017	0
	x_0^7	2	27	0.038158	0	2	27	0.041666	0	26	190	0.240258	4.43E-11	6	59	0.075446	0	9	28	0.202887	3.72E-12
	x_0^8	1	3	0.007806	0	1	3	0.007687	0	1	3	0.007488	0	1	3	0.007325	0	4	49	0.526138	0
50000	x_0^1	1	14	0.081922	0	1	14	0.083414	0	36	82	0.541356	6.68E-11	24	60	0.416089	9.4E-11	4	49	0.368258	0
	x_0^2	2	27	0.140326	0	2	27	0.144851	0	21	162	0.807446	7.06E-11	6	66	0.337831	0	1	14	0.003984	0
	x_0^3	1	14	0.075495	0	1	14	0.075355	0	34	69	0.486584	7.48E-11	24	51	0.423295	9.67E-11	4	50	0.008395	0
	x_0^4	2	27	0.144768	0	2	27	0.144126	0	26	189	0.999566	9.44E-11	6	59	0.332809	0	7	24	0.004468	7.44E-11
	x_0^5	2	27	0.142742	0	2	27	0.142328	0	27	196	1.025835	3.04E-11	6	59	0.32261	0	4	50	0.007229	0
	x_0^6	2	27	0.146161	0	2	27	0.141539	0	9	69	0.363751	0	5	50	0.252642	0	4	49	0.003732	0
	x_0^7	2	27	0.137162	0	2	27	0.142295	0	27	196	1.075529	3.04E-11	6	59	0.320723	0	1	14	0.004157	0
	x_0^8	1	3	0.022052	0	1	3	0.022394	0	1	3	0.020302	0	1	3	0.021952	0	4	50	0.008577	0
100000	x_0^1	1	14	0.170899	0	1	14	0.159038	0	36	82	1.047394	9.47E-11	25	62	0.808989	4.06E-11	7	24	0.005938	8.03E-11
	x_0^2	2	27	0.27101	0	2	27	0.276769	0	21	163	1.65595	6.97E-11	6	66	0.673843	0	4	50	0.003566	0
	x_0^3	1	14	0.155212	0	1	14	0.154447	0	35	71	0.963394	5.24E-11	25	53	0.722074	4.23E-11	4	49	0.006346	0
	x_0^4	2	27	0.283305	0	2	27	0.298966	0	27	194	2.086407	5.26E-11	6	59	0.630216	0	1	14	0.005198	0
	x_0^5	2	27	0.28082	0	2	27	0.284577	0	27	194	2.034998	5.36E-11	6	59	0.632952	0	4	50	0.003295	0
	x_0^6	2	27	0.275107	0	2	27	0.283492	0	9	69	0.73778	0	5	50	0.534591	0	7	24	0.006763	8.59E-11
	x_0^7	2	27	0.266159	0	2	27	0.284258	0	27	194	2.066981	5.36E-11	6	59	0.631261	0	4	50	0.004407	0
	x_0^8	1	3	0.039855	0	1	3	0.039435	0	1	3	0.043968	0	1	3	0.041767	0	4	49	0.005924	0

Table 4. Numerical comparison of SPRPCG algorithm versus DFSP [26], STCGM [25], and MPRPA algorithms [24].

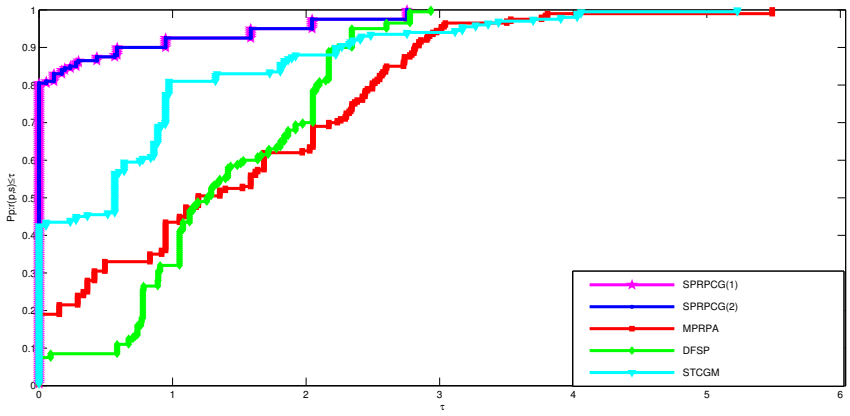
Problem 4.4		SPRPCG(1)				SPRPCG(2)				MPRPA				DFSP				STCGM			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	1	14	0.006826	0	1	14	0.007826	0	1	14	0.006633	0	2	26	0.019218	0	1	14	0.00379	0
	x_0^2	1	14	0.006307	0	1	14	0.006454	0	2	27	0.009176	0	2	24	0.008969	0	1	14	0.007432	0
	x_0^3	1	14	0.007101	0	1	14	0.007151	0	1	14	0.006891	0	2	24	0.00945	0	1	14	0.004323	0
	x_0^4	1	14	0.00711	0	1	14	0.007319	0	2	27	0.010489	0	5	63	0.019216	0	1	14	0.004346	0
	x_0^5	1	14	0.006932	0	1	14	0.00683	0	1	14	0.00652	0	5	63	0.020158	0	1	14	0.004034	0
	x_0^6	1	14	0.007527	0	1	14	0.007411	0	1	14	0.005606	0	2	24	0.009091	0	1	14	0.009196	0
	x_0^7	1	14	0.007439	0	1	14	0.007114	0	1	14	0.006337	0	5	63	0.018762	0	1	14	0.004103	0
	x_0^8	2	27	0.049849	0	2	27	0.01603	0	1	12	0.006616	0	1	9	0.005235	0	1	14	0.008791	0
1000	x_0^1	1	14	0.009025	0	1	14	0.009648	0	1	14	0.008993	0	2	26	0.014945	0	1	14	0.003916	0
	x_0^2	1	14	0.008027	0	1	14	0.007585	0	2	27	0.011886	0	2	24	0.011934	0	1	14	0.009405	0
	x_0^3	1	14	0.009139	0	1	14	0.009137	0	1	14	0.008549	0	2	24	0.013415	0	1	14	0.026243	0
	x_0^4	1	14	0.009153	0	1	14	0.008709	0	2	27	0.013689	0	5	63	0.029391	0	1	14	0.032894	0
	x_0^5	1	14	0.00978	0	1	14	0.009065	0	1	14	0.010278	0	5	63	0.029139	0	1	14	0.020722	0
	x_0^6	1	14	0.007148	0	1	14	0.008364	0	1	14	0.007888	0	2	24	0.011702	0	1	14	0.037939	0
	x_0^7	1	14	0.008703	0	1	14	0.010072	0	1	14	0.008224	0	5	63	0.028993	0	1	14	0.033949	0
	x_0^8	2	27	0.023566	0	2	27	0.0236	0	1	12	0.007582	0	1	9	0.007211	0	1	14	0.116722	0
10000	x_0^1	1	14	0.035678	0	1	14	0.035677	0	1	14	0.034545	0	2	26	0.059988	0	1	14	0.125283	0
	x_0^2	1	14	0.033449	0	1	14	0.032304	0	2	27	0.058071	0	2	24	0.052061	0	1	14	0.096024	0
	x_0^3	1	14	0.035411	0	1	14	0.037456	0	1	14	0.033769	0	2	24	0.057893	0	1	14	0.136805	0
	x_0^4	1	14	0.03611	0	1	14	0.037482	0	2	27	0.073244	0	5	63	0.149776	0	1	14	0.096786	0
	x_0^5	1	14	0.037252	0	1	14	0.038376	0	2	27	0.062793	0	5	63	0.149361	0	1	14	0.292711	0
	x_0^6	1	14	0.03319	0	1	14	0.034843	0	1	14	0.029509	0	2	24	0.052367	0	1	14	0.209517	0
	x_0^7	1	14	0.035739	0	1	14	0.035515	0	2	27	0.061489	0	5	63	0.144616	0	1	14	0.300057	0
	x_0^8	2	27	0.097333	0	2	27	0.091716	0	1	12	0.030227	0	1	9	0.023266	0	1	14	0.208775	0
50000	x_0^1	1	14	0.153393	0	1	14	0.158395	0	1	14	0.143519	0	2	26	0.264726	0	1	14	0.332781	0
	x_0^2	1	14	0.131391	0	1	14	0.127049	0	2	27	0.239204	0	2	24	0.215807	0	1	14	0.003875	0
	x_0^3	1	14	0.149158	0	1	14	0.153346	0	1	14	0.138068	0	2	24	0.255097	0	1	14	0.003652	0
	x_0^4	1	14	0.148252	0	1	14	0.146291	0	2	27	0.285022	0	5	63	0.634745	0	1	14	0.00406	0
	x_0^5	1	14	0.149375	0	1	14	0.14397	0	2	27	0.284079	0	5	63	0.656865	0	1	14	0.00492	0
	x_0^6	1	14	0.134224	0	1	14	0.133645	0	1	14	0.127057	0	2	24	0.217751	0	1	14	0.003563	0
	x_0^7	1	14	0.142002	0	1	14	0.141666	0	2	27	0.287637	0	5	63	0.646074	0	1	14	0.004908	0
	x_0^8	2	27	0.383604	0	2	27	0.388343	0	1	12	0.117212	0	1	9	0.093164	0	1	14	0.004126	0
100000	x_0^1	1	14	0.290397	0	1	14	0.278313	0	1	14	0.275099	0	2	26	0.572796	0	1	14	0.004362	0
	x_0^2	1	14	0.258255	0	1	14	0.251781	0	2	27	0.496492	0	2	24	0.515692	0	1	14	0.003318	0
	x_0^3	1	14	0.284833	0	1	14	0.289966	0	1	14	0.270982	0	2	24	0.557129	0	1	14	0.003945	0
	x_0^4	1	14	0.296112	0	1	14	0.283181	0	4	42	0.847157	0	5	63	1.408381	0	1	14	0.003111	0
	x_0^5	1	14	0.282097	0	1	14	0.286251	0	2	27	0.546095	0	5	63	1.457335	0	1	14	0.003803	0
	x_0^6	1	14	0.249396	0	1	14	0.256651	0	1	14	0.254254	0	2	24	0.489583	0	1	14	0.00303	0
	x_0^7	1	14	0.281713	0	1	14	0.294178	0	2	27	0.547392	0	5	63	1.384502	0	1	14	0.004455	0
	x_0^8	2	27	0.761841	0	2	27	0.780406	0	1	12	0.239859	0	1	9	0.182963	0	1	14	0.002743	0

Table 5. Numerical comparison of SPRPCG algorithm versus DFSP [26], STCGM [25], and MPRPA algorithms [24].

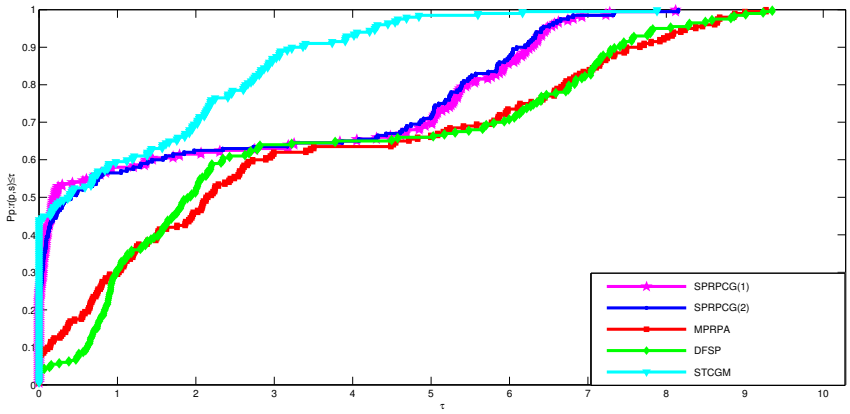
Problem 4.5		SPRPCG(1)				SPRPCG(2)				MPRPA				DFSP				STCGM			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
500	x_0^1	2	27	0.031742	0	2	27	0.04209	0	5	61	0.041802	0	7	73	0.059792	0	2	27	0.033139	0
	x_0^2	2	27	0.008489	0	2	27	0.025384	0	3	30	0.026171	0	4	45	0.017053	0	3	40	0.026812	0
	x_0^3	2	27	0.004682	0	2	27	0.006594	0	7	84	0.010716	0	6	69	0.010229	0	3	40	0.00564	0
	x_0^4	3	40	0.009244	0	3	40	0.008962	0	4	35	0.006085	0	5	57	0.011329	0	4	53	0.006817	0
	x_0^5	2	27	0.006779	0	2	27	0.007919	0	94	1214	0.155333	9.21E-11	5	56	0.011371	0	3	40	0.006959	0
	x_0^6	2	27	0.006728	0	2	27	0.008041	0	4	38	0.00796	0	6	72	0.013581	0	2	27	0.004568	0
	x_0^7	2	27	0.006478	0	2	27	0.006652	0	94	1214	0.170393	9.21E-11	5	56	0.01192	0	3	40	0.008646	0
	x_0^8	5	66	0.013512	0	5	66	0.012999	0	2	16	0.005321	0	2	17	0.006361	0	3	40	0.011563	0
1000	x_0^1	2	27	0.009778	0	2	27	0.009275	0	4	48	0.012698	0	8	84	0.023033	0	4	53	0.011952	0
	x_0^2	2	27	0.009892	0	2	27	0.008778	0	3	30	0.009737	0	4	45	0.012158	0	3	40	0.011994	0
	x_0^3	2	27	0.009887	0	2	27	0.009648	0	6	71	0.016186	0	9	78	0.019961	0	2	27	0.036469	0
	x_0^4	3	40	0.012129	0	3	40	0.011647	0	4	34	0.010774	0	5	56	0.015291	0	3	40	0.04434	0
	x_0^5	2	27	0.009335	0	2	27	0.009274	0	5	51	0.013627	0	5	56	0.014255	0	3	40	0.042221	0
	x_0^6	2	27	0.00897	0	2	27	0.014394	0	4	38	0.010787	0	6	72	0.017204	0	4	53	0.062818	0
	x_0^7	2	27	0.009645	0	2	27	0.008467	0	5	51	0.013366	0	5	56	0.016387	0	3	40	0.046493	0
	x_0^8	5	66	0.017462	0	5	66	0.018425	0	2	16	0.006788	0	2	17	0.007563	0	2	27	0.126165	0
10000	x_0^1	2	27	0.037271	0	2	27	0.039634	0	4	48	0.061572	0	8	65	0.114416	0	3	40	0.22902	0
	x_0^2	2	27	0.037149	0	2	27	0.061507	0	3	30	0.041141	0	4	45	0.070161	0	3	40	0.17597	0
	x_0^3	2	27	0.040859	0	2	27	0.044393	0	5	56	0.076301	0	9	68	0.104997	0	4	53	0.298447	0
	x_0^4	3	40	0.05355	0	3	40	0.069205	0	4	33	0.046029	0	5	56	0.087023	0	3	40	0.18673	0
	x_0^5	2	27	0.048185	0	2	27	0.049425	0	4	34	0.058705	0	5	56	0.086953	0	2	27	0.319727	0
	x_0^6	2	27	0.043537	0	2	27	0.046542	0	4	38	0.063938	0	6	72	0.106673	0	3	40	0.295781	0
	x_0^7	2	27	0.047107	0	2	27	0.048886	0	4	34	0.060947	0	5	56	0.084679	0	3	40	0.496922	0
	x_0^8	5	66	0.106679	0	5	66	0.1112	0	2	16	0.030821	0	3	30	0.046564	0	4	53	0.461636	0
50000	x_0^1	2	27	0.209091	0	2	27	0.206392	0	4	48	0.343347	0	10	89	0.61594	0	3	40	0.697509	0
	x_0^2	2	27	0.188112	0	2	27	0.188834	0	3	30	0.223882	0	4	45	0.316733	0	2	27	0.002348	0
	x_0^3	2	27	0.201281	0	2	27	0.201688	0	5	56	0.391772	0	7	71	0.493416	0	3	40	0.004563	0
	x_0^4	3	40	0.296775	0	3	40	0.287346	0	4	33	0.259131	0	5	56	0.39824	0	3	40	0.007643	0
	x_0^5	2	27	0.19805	0	2	27	0.198322	0	4	33	0.247166	0	5	56	0.395003	0	3	40	0.00774	0
	x_0^6	2	27	0.186891	0	2	27	0.188475	0	4	38	0.277752	0	6	72	0.490119	0	3	40	0.005804	0
	x_0^7	2	27	0.210921	0	2	27	0.201636	0	4	33	0.250694	0	5	56	0.397919	0	2	27	0.004488	0
	x_0^8	5	66	0.497911	0	5	66	0.476101	0	2	16	0.131588	0	3	30	0.211239	0	3	40	0.005869	0
100000	x_0^1	2	27	0.424607	0	2	27	0.401009	0	4	48	0.672367	0	9	67	1.027705	0	3	40	0.004747	0
	x_0^2	2	27	0.388978	0	2	27	0.391899	0	3	30	0.448241	0	4	45	0.689342	0	3	40	0.006402	0
	x_0^3	2	27	0.394533	0	2	27	0.401115	0	5	56	0.80191	0	11	124	1.801222	0	3	40	0.004984	0
	x_0^4	3	40	0.571176	0	3	40	0.615386	0	4	33	0.507612	0	5	56	0.86444	0	2	27	0.003807	0
	x_0^5	2	27	0.404814	0	2	27	0.389485	0	4	33	0.499288	0	5	56	0.869131	0	3	40	0.004945	0
	x_0^6	2	27	0.360131	0	2	27	0.405794	0	4	38	0.583693	0	6	72	1.080131	0	3	40	0.006149	0
	x_0^7	2	27	0.380321	0	2	27	0.382274	0	4	33	0.558948	0	5	56	0.898111	0	3	40	0.007208	0
	x_0^8	5	66	0.940601	0	5	66	0.972183	0	2	16	0.258353	0	3	30	0.494071	0	3	40	0.006065	0



(a) Number of iterations.



(b) Function evaluations.



(c) CPU time in second.

Figure 1. Comparison of the proposed algorithm with the two directions versus MPRPA and DFSP algorithms.

4.1. The motion control of robotic manipulator

IIn this part, we will tackle the motion control of a two-joint planar robotic manipulator using the SPRP projection-based algorithm. The following discrete-time kinematics equation describes the position level of the two-joint planar robot manipulator

$$f(\eta_k) = \sigma_k, \tag{4.1}$$

where the kinematics function $f(.)$ is defined by

$$f(\eta) = \begin{pmatrix} m_1 p_1 & m_2 p_2 \\ m_1 r_1 & m_2 r_2 \end{pmatrix}, \tag{4.2}$$

in which, m_j for $j = 1, 2$ is the length of the $j - th$ rod, $p_1 = \cos(\eta_1)$, $r_1 = \sin(\eta_1)$, $p_2 = \cos(\eta_1 + \eta_2)$, and $r_2 = \sin(\eta_1 + \eta_2)$. Also, the vector $\sigma_k \in \mathbb{R}^2$ stands for the end effector position vector, and the joint angle vector is represented by $\eta_k \in \mathbb{R}^2$, for more detail about the kinematics equation we refer readers to [29] and the references therein. In our experiment we aimed at solving the series of optimization problems at a time interval $t_k \in \left[0, t_f\right]$ defined as follows:

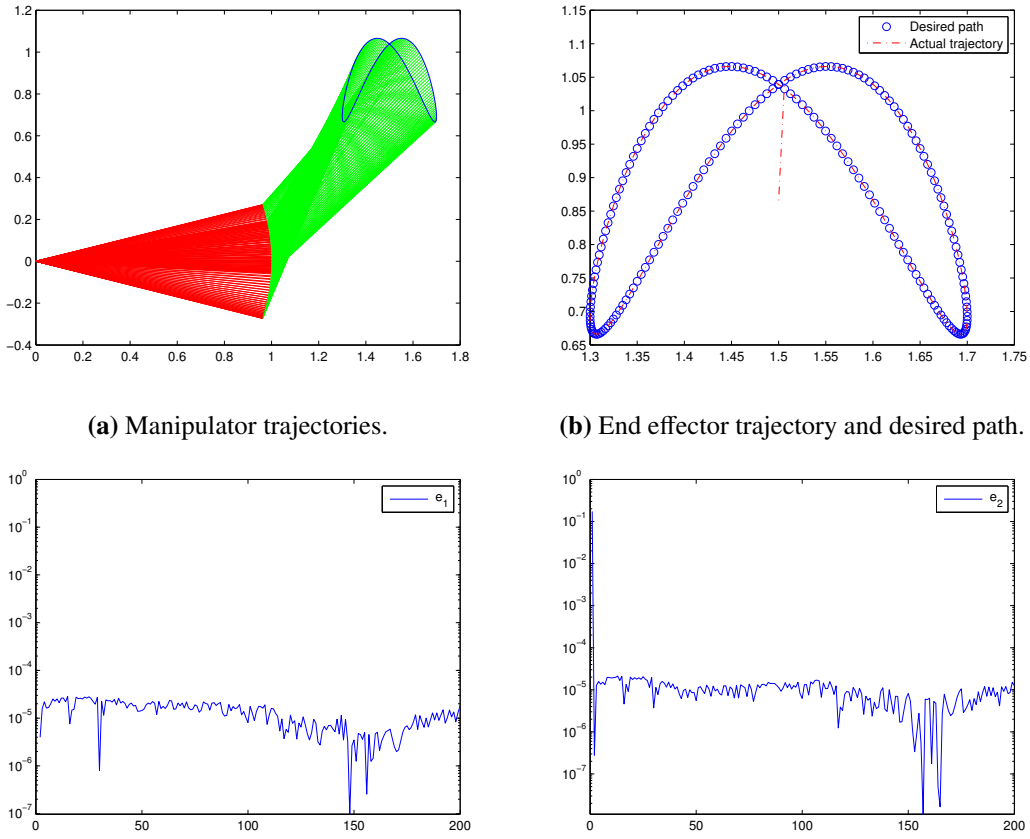
$$\min_{\sigma_k \in \mathbb{R}^2} \frac{1}{2} \|\sigma_k - \sigma_{dk}\|^2 . \tag{4.3}$$

In this experiment, $m_j = 1$ is assigned for $j = 1, 2$, and the end effector is regulated to trace a Lissajous curve as provided by

$$\sigma_{dk} = \begin{pmatrix} 1.5 + 0.2 \sin \frac{\pi t_k}{5} \\ \frac{\sqrt{3}}{2} + 0.2 \sin\left(\frac{\pi t_k}{5} + \frac{\pi}{3}\right) \end{pmatrix}. \tag{4.4}$$

For SPRP algorithm, we initialized $b = 0.2$, $a = 0.08$, $\theta = 0.2$, $\tau = 1$, $\omega_{\min} = 0.0001$, $\omega_{\max} = 10^4$, $\eta_0 = \left[0, \frac{\pi}{3}\right]$, and the end task duration $t_f = 10s$. We further divided the task duration $[0, 10]$ into 200 equal parts.

Figure 2 depicts the experimental findings generated by the SPRP algorithm, including the robot trajectories synthesized by the SPRP algorithm, the end plots of the effector trajectory and the intended route, and the error of the SPRP algorithm on both the vertical x and y axes. It is not difficult to see from Figures 2a and 2b that the SPRP algorithm completes the specified task satisfactorily. Furthermore, Figures 2c and 2d show that the resultant error is around 10^{-5}.



(a) Manipulator trajectories.

(b) End effector trajectory and desired path.

(c) Tracking errors on the horizontal x-axis.

(d) Tracking errors on the vertical y-axis.

Figure 2. Numerical results generated by SPRP algorithm for the motion control of a two-joint planar robotic manipulator problem.

5. Conclusions

A robust descent-scaled PRP projection-based technique for solving monotone nonlinear problems with convex constraints was given. The monotone and Lipschitz continuous assumptions are applied to accomplish the proposed algorithm's global convergence. A detailed numerical comparison with the related algorithms indicated that the proposed technique is much more efficient than the existing algorithms. Furthermore, the proposed technique is used to solve the motion control problem of a two-joint planar robotic manipulator with extremely little error. Finally, the provided technique may be used for various problems, including solving unconstrained optimization problems, generic algebraic nonlinear systems, and so on.

Acknowledgments

This research received funding support from the NSRF via the Program Management Unit for Human Resources & Institutional Development, Research and Innovation, (grant number B05F650018).

Conflict of interest

The authors declare no conflict of interest.

References

1. Z. Dai, H. Zhou, F Wen, S. He, Efficient predictability of stock return volatility: The role of stock market implied volatility, *N. Am. J. Econ. Financ.*, **52** (2020), 101174. https://doi.org/10.1016/j.najef.2020.101174
2. M. Figueiredo, R. Nowak, S. J. Wright, Gradient projection for sparse reconstruction application to compressed sensing and other inverse problems, *IEEE J-STSP*, **1** (2007), 586–597. https://doi.org/10.1109/JSTSP.2007.910281
3. J. M. Ortega, W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, 1970.
4. G. Zhou, K. C. Toh, Superlinear convergence of a Newton-type algorithm for monotone equations, *J. Optim. Theory Appl.*, **125** (2005), 205–221. https://doi.org/10.1007/s10957-004-1721-7
5. J. Sabi'u, A. Shah, M. Y. Waziri, A modified Hager-Zhang conjugate gradient method with optimal choices for solving monotone nonlinear equations, *Int. J. Comput. Math.*, **99** (2022), 332–354. https://doi.org/10.1080/00207160.2021.1910814
6. E. Polak, G. Ribi$\grave{e}$re, Note sur la convergence de methods de directions conjugees, *Rev. Fr. Inform. Rech. Oper.*, **16** (1969), 35–43.
7. W. Cheng, A PRP type method for systems of monotone equations, *Math. Comput. Model.*, **50** (2009), 15–20. https://doi.org/10.1016/j.mcm.2009.04.007
8. M. Fukushima, L. Qi (Eds.), *Reformulation: Nonsmooth, piecewise smooth, semismooth and smoothing methods*, Kluwer Academic Publishers, 1999.
9. G. Yu, A derivative-free method for solving large-scale nonlinear systems of equations, *J. Ind. Manag. Optim.*, **6** (2010), 149–160. https://doi.org/10.3934/jimo.2010.6.149
10. M. Ahookhosh, K. Amini, S. Bahrami, Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations, *Numer. Algor.*, **64** (2013), 21–42. https://doi.org/10.1007/s11075-012-9653-z
11. G. Yuan, M. Zhang, A three-terms Polak-Ribi$\grave{e}$re-Polyak conjugate gradient algorithm for large-scale nonlinear equations, *J. Comput. Appl. Math.*, **286** (2015), 186–195. https://doi.org/10.1016/j.cam.2015.03.014
12. J. Sabi'u, A. M. Gadu, A projected hybrid conjugate gradient method for solving large-scale system of nonlinear equations, *Malays. J. Comput. Appl. Math.*, **1** (2018), 10–20. https://doi.org/10.37231/myjcam.2018.1.2.20
13. A. B. Abubakar, P. Kumam, An improved three-term derivative-free method for solving nonlinear equations, *Comput. Appl. Math.*, **37** (2018), 6760–6773. https://doi.org/10.1007/s40314-018-0712-5

14. A. M. Awwal, P. Kumam, A. B. Abubakar, Spectral modified Polak-Ribire-Polyak projection conjugate gradient method for solving monotone systems of nonlinear equations, *Appl. Math. Comput.*, **362** (2019), 124514. https://doi.org/10.1016/j.amc.2019.06.028

15. J. Sabiu, A. Shah, M. Y. Waziri, M. K. Dauda, A new hybrid approach for solving large-scale monotone nonlinear equations, *J. Math. Fund. Sci.*, **52** (2020), 17–26. https://doi.org/10.5614/j.math.fund.sci.2020.52.1.2

16. H. Feng, T. Li, An accelerated conjugate gradient algorithm for solving nonlinear monotone equations and image restoration problems, *Math. Probl. Eng.*, **2020** (2020), 7945467. https://doi.org/10.1155/2020/7945467

17. J. K. Liu, Derivative-free spectral PRP projection method for solving nonlinear monotone equations with convex constraints (Chinese), *Math. Numer. Sin.*, **38** (2016), 113–124. https://doi.org/10.12286/jssx.2016.2.113

18. D. Feng, M. Sun, X. Wang, A family of conjugate gradient methods for large-scale nonlinear equations, *J. Inequal. Appl.*, **2017** (2017), 236. https://doi.org/10.1186/s13660-017-1510-0

19. J. Guo, Z. Wan, A modified spectral PRP conjugate gradient projection method for solving large-scale monotone equations and its application in compressed sensing, *Math. Probl. Eng.*, **2019** (2019), 5261830. https://doi.org/10.1155/2019/5261830

20. Y. Zhou, Y. Wu, X. Li, A new hybrid prpfr conjugate gradient method for solving nonlinear monotone equations and image restoration problems, *Math. Probl. Eng.*, **2020** (2020), 6391321. https://doi.org/10.1155/2020/6391321

21. Y. Hu, Y. Wang, An efficient projected gradient method for convex constrained monotone equations with applications in compressive sensing, *J. Appl. Math. Phys.*, **8** (2020), 983–998. https://doi.org/10.4236/jamp.2020.86077

22. H. Guan, S. Wang, A modified conjugate gradient method for solving large-scale nonlinear equations, *Math. Probl. Eng.* **2021** (2021), 9919595. https://doi.org/10.1155/2021/9919595

23. A. B. Abubakar, P. Kumam, H. Mohammad, A. H. Ibrahim, PRP-like algorithm for monotone operator equations, *Japan J. Indust. Appl. Math.*, **38** (2021), 805–822. https://doi.org/10.1007/s13160-021-00462-2

24. M. Y. Waziri, K. Ahmed, A. S. Halilu, A modified PRP-type conjugate gradient projection algorithm for solving large-scale monotone nonlinear equations with convex constraint, *J. Comput. Appl. Math.*, **407** (2022), 114035. https://doi.org/10.1016/j.cam.2021.114035

25. J. Sabiu, K. O. Aremu, A. Althobaiti, A. Shah, Scaled three-term conjugate gradient methods for solving monotone equations with application, *Symmetry*, **14** (2022), 936. https://doi.org/10.3390/sym14050936

26. K. Amini, P. Faramarzi, S. Bahrami, A spectral conjugate gradient projection algorithm to solve the large-scale system of monotone nonlinear equations with application to compressed sensing, *Int. J. Comput. Math.*, **99** (2022), 2290–2307. https://doi.org/10.1080/00207160.2022.2047180

27. J. Barzilai, J. M. Borwein, Two-point step size gradient methods, *IMA J. Numer. Anal.*, **8** (1988), 141–148. https://doi.org/10.1093/imanum/8.1.141

28. E. D. Dolan, J. J. Moré, Benchmarking optimization software with performance profiles, *Math. Program.*, **91** (2002), 201–213. https://doi.org/10.1007/s101070100263

29. Y. Zhang, L. He, C. Hu, J. Guo, J. Li, Y. Shi, General four-step discrete-time zeroing and derivative dynamics applied to time-varying nonlinear optimization, *J. Comput. Appl. Math.*, **347** (2019), 314–329. https://doi.org/10.1016/j.cam.2018.08.017

AIMS Press