axioms

MDPI

*Article*

# A Subspace Derivative-Free Conjugate Gradient Method for Solving Nonlinear Monotone Equations with Convex Constraints

**Zongxu Li, Zhuo Fang, Mingyuan Cao *, Yueting Yang *, Ruobing Mei and Siqi Liu**

School of Mathematics and Statistics, Beihua University, Jilin 132013, China; lizongxu0812@163.com (Z.L.); fz64012026@163.com (Z.F.); mrb1208@163.com (R.M.); lsq020902@163.com (S.L.)
* Correspondence: cmy0918@beihua.edu.cn (M.C.); yyt2858@beihua.edu.cn (Y.Y.)

**Abstract**

We propose a novel subspace derivative-free conjugate gradient method for solving large-scale nonlinear monotone equations with convex constraints. At each iteration, the search direction is constructed by minimizing a quadratic model within a subspace spanned by the current negative function value vector and the two most recent search directions. The algorithm incorporates a hyperplane projection technique to generate feasible iterative points. Under reasonable assumptions, we establish the global convergence and R-linear convergence rate of the proposed method. Extensive numerical experiments on benchmark problems demonstrate that the new algorithm significantly outperforms state-of-the-art derivative-free methods in terms of number of iterations, function evaluations, and CPU time. The results confirm the efficiency and robustness of the proposed approach for solving large-scale monotone systems.

**Keywords:** nonlinear monotone equations; subspace derivative-free method; global convergence; R-linear convergence rate

**MSC:** 65H10; 90C56

## 1. Introduction

Nonlinear equations have numerous practical applications, such as motion control problems [1], neural networks [2], compressive sensing [3], image restoration [4], etc. Many related issues can be transformed into

$$F(x) = 0,\ x \in \Omega, \tag{1}$$

where $\Omega \subset \mathbb{R}^n$ stands for a non-empty closed convex set, and $F : \mathbb{R}^n \to \mathbb{R}^n$ is a continuous mapping fulfilling the monotonicity condition

$$\langle F(x) - F(y),\ x - y\rangle \geq 0, \quad \forall x, y \in \mathbb{R}^n. \tag{2}$$

Among existing methods for problem (1), Newton methods [5,6], quasi-Newton methods [7,8], Levenberg–Marquardt methods and its variants [9–11], and trust region methods [12] have long attracted widespread attention from researchers due to their local superlinear convergence properties. Meanwhile, these methods require solving linear systems involving the Jacobian matrix or its approximations at each iteration, which can lead to high computational costs when dealing with large-scale nonlinear systems of equations. In practice, large-scale variants of these methods, including inexact or matrix-free



Academic Editor: Glenn Ledder

Received: 24 March 2026
Revised: 1 May 2026
Accepted: 8 May 2026
Published: 9 May 2026

Newton methods and limited-memory quasi-Newton approaches, effectively mitigate these computational costs, enabling their application to large-scale problems.

Among various derivative-free methods, subspace techniques have garnered significant interest as they can effectively balance computational cost and convergence performance. Porcelli and Toint [13] proposed a structured version of the derivative-free stochastic pattern search optimization algorithm. Hare et al. [14] provided a theoretical basis for the efficiency advantages of low-dimensional random sampling from the viewpoint of expected descent. Chen et al. [15] combined a more accurate Q-fully quadratic model with a random subspace trust region framework and analyzed its convergence rate analysis. Kimiaei et al. [16], focusing on nonlinear monotone equations, further proposed the subspace inertial line search algorithm, which improved the efficiency of solving such equations by dynamically updating the subspace structure. Xie and Yuan [17] proposed a two-dimensional-model-based subspace method, combining the two-dimensional quadratic interpolation model with the trust region technique to solve large-scale derivative-free optimization problems. In addition, subspace methods have also been widely applied to various scenarios [18–20].

In the research progress of derivative-free conjugate gradient methods, various innovative approaches have emerged, significantly expanding the technical pathways in this field [21,22]. As an important branch of such methods, the subspace minimization conjugate gradient method [23] has attracted considerable attention for its excellent numerical performance. It generates search directions by minimizing a quadratic approximation model of the objective function at the current point. In recent years, the application scope of this method has expanded. For example, Tang et al. [24] proposed a three-term subspace minimization conjugate gradient method, which generates the search direction by constructing a three-dimensional subspace and minimizing the quadratic approximation function, significantly improving the ability to solve large-scale nonlinear monotone systems of equations. Y, Arfat et al. [25,26] developed hybrid steepest-descent methods for convex minimization over split equilibrium problems and fixed point sets of multivalued mappings.

Projection methods, especially hyperplane projection, are widely used for convex-constrained monotone equations as they ensure feasible iterates and global convergence without derivative information. Conjugate gradient directions are efficient for large-scale problems due to low memory cost and fast convergence. Combining subspace CG directions with hyperplane projection inherits both merits: strong descent for acceleration and projection for feasibility and convergence. This motivates our work.

Inspired by the subspace conjugate gradient framework for large-scale unconstrained optimization in Yang et al. [27], we develop the subspace derivative-free conjugate gradient algorithm with well-justified modifications. We extend its subspace structure to convex-constrained monotone equations for feasible iterates, redesign the three-term search direction to enhance descent and stability, integrate hyperplane projection and a safeguard parameter to ensure sufficient descent and global convergence, and adopt derivative-free line search to avoid Jacobian calculation. These targeted improvements result in the proposed SDCG algorithm for convex-constrained nonlinear monotone equations. The main innovations of the algorithm include the following:

- The method constructs search directions by combining the current negative gradient vector with the two most recent search directions, solving a quadratic subproblem within a low-dimensional subspace. This balances computational efficiency and algorithmic performance.
- The coefficients for the subspace directions are derived explicitly through minimizing a quadratic approximation, with three specific cases based on the subspace dimension. A safeguard parameter ensures the sufficient descent condition is always satisfied.

The algorithm incorporates a hyperplane projection technique to ensure global convergence and iterate feasibility, while preserving derivative-free properties.

- Under standard assumptions, the method is proven to be globally convergent and achieves an R-linear convergence rate. Extensive numerical experiments on large-scale test problems show that the proposed SDCG method significantly outperforms existing derivative-free conjugate gradient methods in terms of iterations, function evaluations, and CPU time, especially for high-dimensional systems.

The rest of this paper is organized as follows. In Section 2, we present the specific steps of the fundamentals of the proposed subspace derivative-free conjugate gradient algorithm and a sufficient descent proof. In Section 3, we give a proof of boundedness of the search direction and global convergence of the algorithm. In Section 4, we analyze the numerical performance of the algorithm, including numerical results for solving large-scale nonlinear systems of monotone equations and comparisons with other algorithms.

## 2. Algorithm

In this section, we propose a new method for (1) with the projection technique, in which the search directions are determined by minimizing the quadratic approximation of the objective function in the subspace $\Omega_k$.

In order to avoid zigzagging and accelerate convergence, we choose the subspace spanned by the current residual and previous two search directions, capturing both the first-order information and the memory of the iterative path, i.e., $\Omega_k \triangleq \text{span}\{-g_{k+1}, s_k, s_{k-1}\}$, and consider the approximation of $f(x)$ as

$$\Phi_{k+1}(d) = g_{k+1}^T d + \frac{1}{2} d^T B_{k+1} d, \tag{3}$$

where $g_{k+1} = g(x_{k+1})$ is the gradient of $F(x)$ at $x_{k+1}$, $B_{k+1}$ is an approximation of the Hessian $\nabla^2 f(x_{k+1})$, and $d \in \Omega_k$. Suppose $B_{k+1}$ is always strictly positive definite. Since $\nabla^2 f(x_{k+1}) s_k \approx y_k$, we would like to choose $B_{k+1}$ satisfying the quasi-Newton equation $B_{k+1} s_k = y_k$. For the sake of brevity, we denote $F(x_k)$ as $F_k$ throughout this paper.

Specifically, let

$$\bar{d} = -F_{k+1} + a s_k + b s_{k-1}, \tag{4}$$

where $(a, b)$ is the solution of the following minimizing problem:

$$\min_{a,b \in \mathbb{R}} \Phi_{k+1}(\bar{d}), \bar{d} \in \Omega_k. \tag{5}$$

First, we show the solution for $a_k, b_k$, which can be separated into the following three cases:

Case I. If $\dim(\Omega_k) = 2$ and $\Omega_k \triangleq \text{span}\{-F_{k+1}, s_k\}$, then the solution can be expressed as

$$a_k = \frac{F_{k+1}^T B_{k+1} s_k - F_{k+1}^T s_k}{s_k^T B_{k+1} s_k} = \frac{F_{k+1}^T y_k - F_{k+1}^T s_k}{y_k^T s_k}, \quad b_k = 0, \tag{6}$$

and

$$\bar{d} = -F_{k+1} + a_k s_k.$$

Case II. If $\dim(\Omega_k) = 2$ and $\Omega_k \triangleq \text{span}\{-F_{k+1}, s_{k-1}\}$, then the solution can be expressed as

$$a_k = 0, \quad b_k = \frac{1}{2}\left[\frac{F_{k+1}^T y_k}{y_k^T s_{k-1}} - \frac{(F_{k+1}^T s_k)(y_k^T s_{k-1})(y_k^T s_k)}{(s_k^T s_k)(y_k^T s_{k-1})^2}\right], \tag{7}$$

and

$$\bar{d} = -F_{k+1} + b_k s_{k-1}.$$

Case III. If $\dim(\Omega_k) = 3$ and $\Omega_k \triangleq \text{span}\{-F_{k+1}, s_k, s_{k-1}\}$, then the solution can be expressed as

$$a_k = \frac{1}{\Delta_k}\left[\eta_k\left(F_{k+1}^T y_k - F_{k+1}^T s_k\right) - \left(y_k^T s_{k-1}\right)\left(\omega_k - F_{k+1}^T s_{k-1}\right)\right], \tag{8}$$

$$b_k = \frac{1}{\Delta_k}\left[\left(y_k^T s_k\right)\left(\omega_k - F_{k+1}^T s_{k-1}\right) - \left(y_k^T s_{k-1}\right)\left(F_{k+1}^T y_k - F_{k+1}^T s_k\right)\right], \tag{9}$$

and

$$\bar{d} = -F_{k+1} + a_k s_k + b_k s_{k-1},$$

where $\omega_k = F_{k+1}^T s_{k-1} + \frac{(F_{k+1}^T y_k)(y_k^T s_{k-1})}{y_k^T s_k} - \frac{(F_{k+1}^T s_k)(s_k^T s_{k-1})}{s_k^T s_k}, \eta_k = 2\frac{(y_k^T s_{k-1})^2}{y_k^T s_k}$, and $\Delta_k = \left(y_k^T s_{k-1}\right)^2$.

If $s_k^T y_k$ is small, it indicates that the inner product between the change in iteration point and in $F(x)$ during the previous iteration is small. By setting $\xi_1$, we ensure that the search direction satisfies the sufficient descent condition at all times. Therefore, based on (4) and selecting a parameter $\xi_1$ as a protective parameter, we have

$$d_{k+1} = \begin{cases} \bar{d}, & \text{if } s_k^T y_k \geq \xi_1 \|y_k\|^2, \xi_1 > 0, \\ -F_{k+1}, & \text{otherwise}, \end{cases} \tag{10}$$

where $\bar{d}$ is given by (4), $y_k = \bar{y}_k + r s_k$, $\bar{y}_k = F_{k+1} - F_k$, $s_k = x_{k+1} - x_k$, and $s_{k-1} = x_k - x_{k-1}$.

Now, we select an appropriate line search technique to get the step length $\alpha_k$. Zhang and Zhou [28] presented the following derivative-free line search rule, which requires $\alpha_k$ to be the largest step length of $\alpha_k = \max\{\xi\rho^i | i = 0, 1, 2, \dots\}$ to satisfy the following condition

$$-F(x_k + \alpha d_k)^{\mathrm{T}} d_k \geq \sigma\alpha \|F(x_k + \alpha d_k)\| \|d_k\|^2, \tag{11}$$

where $\sigma > 0$ and $0 < \rho < 1$.

After the search direction $d_k$ and step length $\alpha_k$ are obtained, we set auxiliary point

$$z_k = x_k + \alpha_k d_k, \tag{12}$$

which satisfies

$$F(z_k)^{\mathrm{T}}(x_k - z_k) > 0. \tag{13}$$

For any $x^*$ such that $F(x^*) = 0$, by the monotonicity of $F(x)$, we have

$$F(z_k)^{\mathrm{T}}(x^* - z_k) = -\left(F(x^*) - F(z_k)\right)^{\mathrm{T}}(x^* - z_k) \leq 0.$$

Thus, the hyperplane

$$H_k = \left\{x \in \mathbb{R}^n \mid F(z_k)^{\mathrm{T}}(x - z_k) = 0\right\},$$

strictly separates the current iterate $x_k$ from zeros of the equation system (1).

Based on the above conclusion, Solodov and Svaiter [29] projected $x_k$ onto $H_k$ to get the next iteration point $x_{k+1}$. Its iteration format is

$$x_{k+1} = x_k - \frac{F(z_k)^{\mathrm{T}}(x_k - z_k)}{\|F(z_k)\|^2} F(z_k). \tag{14}$$

This projection technique ensures that the solution generated at each iteration always lies within the given convex constraint set $\Omega$ and verifies the boundedness of the search direction. Numerical experiments show it is superior to other similar algorithms in solving large-scale nonlinear monotone equations. Next, we present the specific steps of the novel subspace derivative-free conjugate gradient method (SDCG).

**Lemma 1.** *Suppose $\{x_k\}$ is generated by Algorithm 1. Then, the search direction fulfills the sufficient descent condition*

$$d_{k+1}^T F_{k+1} \leq -C\|F_{k+1}\|^2, \tag{15}$$

*for all $k \geq 0$.*

**Algorithm 1** SDCG

Step 0. Let $x_0 \in \mathbb{R}^n$, $x_{-1} = x_0$, $\varepsilon > 0$, $0 < \sigma < 1$, $0 < \xi < 1$, $0 < \xi_1 < 1$, $0 < \rho < 1$, $0 < \kappa < 2$ and compute $F_0 = F(x_0)$. Set $d_0 := -F_0$, $k := 0$.

Step 1. Stop if $\|F_k\| \leq \varepsilon$, else compute $d_{k+1}$ by (10).

Step 2. Set $z_k = x_k + \alpha_k d_k$, compute $\alpha_k = \max\{\xi\rho^i | i = 0, 1, 2, \dots\}$ satisfying

$$-\langle F(x_k + \alpha d_k), d_k\rangle \geq \sigma\alpha\|F(x_k + \alpha d_k)\|\|d_k\|^2.$$

Step 3. If $z_k \in \Omega$ and $\|F(z_k)\| \leq \varepsilon$, set $x_{k+1} = z_k$, and stop. Otherwise compute

$$\lambda_k = \frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2}$$

and

$$x_{k+1} = P_\Omega[x_k - \kappa\lambda_k F(z_k)].$$

Step 4. Set $k := k + 1$, go to Step 1.

**Proof.** We prove the conclusion by considering the three cases of subspace dimension $\dim(\Omega_k)$ separately, where $\Omega_k \triangleq \text{span}\{-F_{k+1}, s_k, s_{k-1}\}$, $s_k = x_{k+1} - x_k$, and $s_{k-1} = x_k - x_{k-1}$.

According to the case I, $a_k$ and $b_k$ are determined by (6), so we can obtain

$$\begin{aligned} d_{k+1}^T F_{k+1} &= \left(-F_{k+1} + \frac{F_{k+1}^T y_k - F_{k+1}^T s_k}{y_k^T s_k} \cdot s_k\right)^T F_{k+1} \\ &\leq -||F_{k+1}||^2 + \left(1 - \frac{s_k^T y_k}{\|y_k\|^2}\right)\|F_{k+1}\|^2 \\ &\leq -\xi_1\|F_{k+1}\|^2. \end{aligned}$$

According to case II, $a_k$ and $b_k$ are determined by (7), so we can obtain

$$\begin{aligned} d_{k+1}^T F_{k+1} &= \left(-F_{k+1} + \frac{1}{2}\left[\frac{F_{k+1}^T y_k}{y_k^T s_{k-1}} - \frac{(F_{k+1}^T s_k)(s_k^T s_{k-1})(y_k^T s_k)}{(s_k^T s_k)(y_k^T s_{k-1})^2}\right] \cdot s_{k-1}\right)^T F_{k+1} \\ &\leq -||F_{k+1}||^2 + \frac{1}{2}\left(1 - \frac{s_k^T y_k}{\|y_k\|^2}\right)\|F_{k+1}\|^2 \\ &\leq -\frac{1}{2} - \frac{1}{2}\xi_1\|F_{k+1}\|^2. \end{aligned}$$

According to case III, $a_k$ and $b_k$ are determined by (8) and (9), so we can obtain

$$\begin{aligned} d_{k+1}^T F_{k+1} &= \left(-F_{k+1} + \frac{1}{\Delta_k}(a_k s_k + b_k s_{k-1})\right)^T F_{k+1} \\ &\leq -\frac{s_k^T y_k}{\|y_k\|^2}\|F_{k+1}\|^2 \\ &\leq -\xi_1 \|F_{k+1}\|^2. \end{aligned}$$

Combining the three cases, let $C = \min\left(1, \xi_1, \frac{1}{2} + \frac{1}{2}\xi_1\right)$. Then, $\langle d_{k+1}, F_{k+1}\rangle \leq -C\|F_{k+1}\|^2$ for all $k \geq 0$, which completes the proof. □

**Lemma 2.** *Let $\{d_k\}$ and $\{x_k\}$ be generated by Algorithm 1. Then, a step size $\alpha_k$ satisfying (11) always exists.*

**Proof.** We use contradiction to prove it. By assuming that inequality (11) does not hold, we obtain a result that contradicts (15). □

## 3. Convergence Analysis

In this section, we establish the convergence results of Algorithm 1. Some assumptions are listed.

**Assumption 1.** *The problem admits a solution $x^* \in \Omega$ such that $F(x^*) = 0$.*

**Assumption 2.** *The mapping $F(x)$ is continuous and monotone.*

**Lemma 3.** *Let Assumptions 1 and 2 hold. For the sequences $\{x_k\}$ and $\{z_k\}$ generated by Algorithm 1, the following inequality holds*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \kappa(2-\kappa)\|x_k - z_k\|^4, \quad \forall x^* \in \Omega.$$

*Furthermore, $\{x_k\}$ is bounded and*

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

**Proof.** From (2), we have

$$\langle F(z_k), x_k - x^*\rangle \geq \langle F(z_k), x_k - z_k\rangle, \quad \forall x^* \in \Omega,$$

which, together with (11), implies that

$$\langle F(z_k), x_k - z_k\rangle \geq \sigma\alpha_k^2 \|F(z_k)\|\|d_k\|^2 \geq 0.$$

As a result, we have

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &= \|P_\Omega[x_k-\kappa\lambda_k F(z_k)]-x^*\|^2\\ &\le \|x_k-\kappa\lambda_k F(z_k)-x^*\|^2\\ &= \|x_k-x^*\|^2-2\kappa\lambda_k\langle F(z_k),x_k-x^*\rangle+\|\kappa\lambda_k F(z_k)\|^2\\ &\le \|x_k-x^*\|^2-2\kappa\lambda_k\langle F(z_k),x_k-z_k\rangle+\|\kappa\lambda_k F(z_k)\|^2\\ &= \|x_k-x^*\|^2-\kappa(2-\kappa)\frac{\langle F(z_k),x_k-z_k\rangle^2}{\|F(z_k)\|^2}\\ &\le \|x_k-x^*\|^2-\kappa(2-\kappa)\sigma^2\|x_k-z_k\|^4.\end{aligned}$$

This completes the proof. □

Thus, $\{\|x_k-x^*\|\}$ is non-increasing, so $\{x_k\}$ is bounded. We have

$$\kappa(2-\kappa)\sigma^2\sum_{k=0}^{\infty}\|x_k-z_k\|^4<\|x_0-x^*\|^2<+\infty.$$

According to the definition of $\{z_k\}$,

$$\lim_{k\to\infty}\|x_k-z_k\|=\lim_{k\to\infty}\alpha_k\|d_k\|=0. \tag{16}$$

**Lemma 4.** *Let the sequence $\{d_k\}$ be generated by Algorithm 1. Then, for all $k\ge 0$, we have*

$$C\|F_{k+1}\|\le\|d_{k+1}\|\le C_3\|F_{k+1}\|, \tag{17}$$

*where $0<C<C_3$.*

**Proof.** By Lemma 1 and the Cauchy–Schwarz inequality, we obtain

$$\|d_{k+1}\|\ge C\|F_{k+1}\|. \tag{18}$$

By utilizing (2), we can obtain

$$s_k^Ty_k=s_k^T(F_{k+1}-F_k)+rs_k^Ts_k\ge r\|s_k\|^2>0. \tag{19}$$

In the following, we consider three cases:
According to case I, $a_k$ and $b_k$ are determined by (6), so we can obtain

$$\begin{aligned}\|d_{k+1}\| &= \left\|-F_{k+1}+\frac{F_{k+1}^{\mathrm{T}}y_k-F_{k+1}^{\mathrm{T}}s_k}{y_k^{\mathrm{T}}s_k}\cdot s_k\right\|\\ &\le \|F_{k+1}\|+\left\|\frac{F_{k+1}^{\mathrm{T}}y_k-F_{k+1}^{\mathrm{T}}s_k}{y_k^{\mathrm{T}}s_k}\cdot s_k\right\|\\ &\le \left(2-\frac{1}{r}\right)\|F_{k+1}\|.\end{aligned}$$

According to case II, $a_k$ and $b_k$ are determined by (7), so we can obtain

$$\begin{aligned}\|d_{k+1}\| &= \left\|-F_{k+1}+\frac{1}{2}\left[\frac{F_{k+1}^{\mathrm{T}}y_k}{y_k^{\mathrm{T}}s_{k-1}}-\frac{(F_{k+1}^{\mathrm{T}}s_k)(s_k^{\mathrm{T}}s_{k-1})(s_k^{\mathrm{T}}y_k)}{(s_k^{\mathrm{T}}s_k)(y_k^{\mathrm{T}}s_{k-1})^2}\right]\right\| \\ &\leq \|F_{k+1}\|+\frac{1}{2}\left\|\frac{F_{k+1}^{\mathrm{T}}y_k}{y_k^{\mathrm{T}}s_{k-1}}-\frac{(F_{k+1}^{\mathrm{T}}s_k)(s_k^{\mathrm{T}}s_{k-1})(s_k^{\mathrm{T}}y_k)}{(s_k^{\mathrm{T}}s_k)(y_k^{\mathrm{T}}s_{k-1})^2}s_{k-1}\right\| \\ &\leq \left(\frac{3}{2}-\frac{1}{2r}\right)\|F_{k+1}\|.\end{aligned}$$

According to case III, $a_k$ and $b_k$ are determined by (8) and (9), so we can obtain

$$\begin{aligned}\|d_{k+1}\| &= \|-F_{k+1}+a_ks_k+b_ks_{k-1}\| \\ &\leq \|F_{k+1}\|+\|a_ks_k\|+\|b_ks_{k-1}\|. \\ \|a_ks_k\| &= \left\|\frac{s_k}{(y_k^Ts_{k-1})^2}\left[2(y_k^Ts_{k-1})^2\cdot\frac{F_{k+1}^Ty_k-F_{k+1}^Ts_k}{y_k^Ts_k}\right.\right. \\ &\quad \left.\left.-y_k^Ts_{k-1})\left(\frac{(F_{k+1}^Ty_k)(y_k^Ts_{k-1})}{y_k^Ts_k}-\frac{(F_{k+1}^Ts_k)(s_k^Ts_{k-1})}{s_k^Ts_k}\right)\right]\right\| \\ &\leq \left(1+\frac{1}{r}\right)\|F_{k+1}\|. \\ \|b_ks_{k-1}\| &= \left\|\frac{1}{(y_k^Ts_{k-1})^2}\left[(y_k^Ts_k)\left(\frac{(F_{k+1}^Ty_k)(y_k^Ts_{k-1})}{y_k^Ts_k}-\frac{(F_{k+1}^Ts_k)(s_k^Ts_{k-1})}{s_k^Ts_k}\right)\right.\right. \\ &\quad \left.\left.-(y_k^Ts_{k-1})\cdot(F_{k+1}^Ty_k-F_{k+1}^Ts_k)\right]s_{k-1}\right\| \\ &\leq \left(2-\frac{2}{r}\right)\|F_{k+1}\|, \\ \|d_{k+1}\| &\leq \left(4-\frac{1}{r}\right)\|F_{k+1}\|.\end{aligned}$$

So, let $C_3=\max\left\{1,\,2-\frac{1}{r},\,\frac{3}{2}-\frac{1}{2r},4-\frac{1}{r}\right\}$, completing the proof. □

**Theorem 1.** *Let Assumption 2 hold, with the sequences $\{x_k\}$ and $\{z_k\}$ generated by Algorithm 1, then*

$$\lim_{k\to\infty}\|F_{k+1}\|=0. \tag{20}$$

**Proof.** We proceed by contradiction. Assume (20) is false, i.e., there exists $r>0$ such that $\|F_k\|\geq r$ for all $k\geq 0$. Together with (18), this gives

$$\|d_k\|\geq Cr,\quad \forall k\geq 0. \tag{21}$$

By (16) and (21), $\lim_{k\to\infty}\alpha_k=0$. Since $\alpha_k=\xi\rho^{i_k}$, the line search (11) implies for sufficiently large $k$ that

$$-\langle F(x_k+\xi\rho^{i_k}d_k),d_k\rangle<\sigma\xi\rho^{i_k}\|F(x_k+\xi\rho^{i_k}d_k)\|\|d_k\|^2. \tag{22}$$

By (17) and $\|F_k\|\geq r$, $\{d_k\}$ is bounded. We may assume $\{x_k\}$ and $\{d_k\}$ are convergent. Letting $k\to\infty$ in (22) yields

$$-\langle F(\bar{x}),\bar{d}\rangle<0, \tag{23}$$

where $\bar{x}$, $\bar{d}$ are the limit points. Taking limits in (15) leads to

$$-\langle F(\bar{x}), \bar{d}\rangle \geq C\|F(\bar{x})\|^2. \tag{24}$$

Combining (23) and (24) gives $\|F(\bar{x})\| = 0$, which contradicts $\|F_k\| \geq r$. Hence, (20) holds. The proof is completed. □

The local convergence rate of Algorithm 1 is listed.

**Assumption 3.** *For any $x^* \in \Omega$, there exist constants $\omega \in (0,1)$ and $\delta > 0$ satisfying*

$$\omega \mathrm{dist}(x, \Omega) \leq \|F(x)\|^2, \quad \forall x \in N(x^*, \delta), \tag{25}$$

*where* $\mathrm{dist}(x, \Omega)$ *stands for the distance from $x$ to the solution set $\Omega$, and $N(x^*, \delta) = \{x \in \Omega \mid \|x - x^*\| \leq \delta\}$.*

**Theorem 2.** *Holding Assumptions 2 and 3, let $\{x_k\}$ be generated by Algorithm 1. Then, $dist(x_k, \Omega)$ converges Q-linearly to 0, and $\{x_k\}$ converges R-linearly to $\bar{x} \in \Omega$.*

**Proof.** Let $u_k := \arg\min\{\|x_k - u\| \mid u \in \Omega\}$. We know clearly that $u_k$ is the nearest solution point from $x_k$, i.e.,

$$\|x_k - u_k\| = \mathrm{dist}(x_k, \Omega).$$

According to (18) and (25), we can easily conclude that $\mathrm{dist}(x_k, \Omega)$ is Q-linearly convergent to 0. If $\mathrm{dist}(x_k, \Omega)$ possesses this property, $\{x_k\}$ is R-linearly convergent to $\bar{x}$. □

## 4. Numerical Experiments

In this section, we compare the performance of SDCG with that of the DFCG [30], DFPB1 [31], and MRMIL1 [32]. All test algorithms were coded in MATLAB R2018b and run on an HP desktop with an Intel(R) Core(TM) i5-10210 CPU (2.11 GHz), 8.00 GB RAM and Windows 10 OS.

In SDCG, we select the following parameters:

$$\rho = 0.7, \sigma = 0.3, \xi = 0.1, \xi_1 = 10^{-7}, \kappa = 1.9, r = 0.0001.$$

All test methods are terminated if the iteration exceeds 10,000, or if the function value of the current iteration satisfies the condition $\|F_k\| \leq 10^{-4}$. We now list the following test problems.

**Problem 1.** *The function $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, where*

$$f_i(x) = 2x_i - \sin(x_i), \quad 1 \leq i \leq n.$$

**Problem 2.** *The function $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, where*

$$f_i(x) = 2x_i - \sin|x_i|, \quad 1 \leq i \leq n.$$

**Problem 3.** *The function* $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, *where*

$$\begin{cases} f_1(x) = x_1 - \exp\left(\cos\left(\frac{x_1+x_2}{n+1}\right)\right), \\ f_i(x) = x_i - \exp\left(\cos\left(\frac{x_{i-1}+x_i+x_{i+1}}{n+1}\right)\right), \quad 2 \le i \le n-1, \\ f_n(x) = x_n - \exp\left(\cos\left(\frac{x_{n-1}+x_n}{n+1}\right)\right). \end{cases}$$

**Problem 4.** *The function* $F(x) = Ax + b(x) - e$, *where* $b(x) = (e^{x_1}, e^{x_2}, \ldots, e^{x_n})^T$, $e = (1, 1, \ldots, 1)^T$ *are the* $n \times 1$ *vectors, and*

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}$$

*is the* $n \times n$ *matrix.*

**Problem 5.** *The function* $F(x) = Ax - e$, *where* $e = (1, 1, \ldots, 1)^T$ *is the* $n \times 1$ *vector, and*

$$A = \begin{pmatrix} \frac{5}{2} & 1 & & & \\ 1 & \frac{5}{2} & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 1 \\ & & & 1 & \frac{5}{2} \end{pmatrix}$$

*is the* $n \times n$ *matrix.*

**Problem 6.** *The function* $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, *where*

$$F_i(x) = e^{x_i} - 1, \text{ for } i = 1, 2, 3, \ldots, n.$$

**Problem 7.** *The function* $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, *where*

$$\begin{cases} F_1(x) = 2.5x_1 + x_2 - 1, \\ F_i(x) = x_{i-1} + 2.5x_i + x_{i+1} - 1, \quad \text{for } i = 2, 3, \ldots, n-1, \\ F_n(x) = x_{n-1} + 2.5x_n - 1. \end{cases}$$

**Problem 8.** *The function* $F(x) = (f_1(x), f_2(x), \ldots, f_n(x))^T$, *where*

$$f_l(x) = \log(x_i + 1) - \frac{x_i}{n}, \quad i = 1, 2, \ldots, n.$$

The above problems with different dimensions ($n$ = 500, 1000, 3000, 5000 and 10,000) are used to test the four methods, and all problems are initialized with the following 6 starting points: $x_0^1 = (1, 1, \ldots, 1)^T$, $x_0^2 = -(1, 1, \ldots, 1)^T$, $x_0^3 = (0.1, 0.1, \ldots, 0.1)^T$, $x_0^4 = (1, \frac{1}{2}, \ldots, \frac{1}{n})^T$, $x_0^5 = (\frac{1}{n}, \frac{2}{n}, \ldots, 1)^T$, $x_0^6 = (\frac{n-1}{n}, \frac{n-2}{n}, \ldots, 0)^T$.

We list partial numerical results with $x_0^2$ as the initial point in Table 1. *Pi* ($i = 1, 2, \cdots, 8$) means the $i$-th test problem listed above, and $Ni$ and $NF$ mean the number of iterations and the number of function evaluations. Finally, $T$ stands for the CPU time required for iterations.

As shown in Table 1, SDCG exhibits excellent performance in solving high-dimensional problems. We can see that SDCG costs less than twice as much as other algorithms in solving test problems 2, 7, and 8. When handling most test problems, SDCG can achieve optimal solutions in less time with fewer iterations and fewer function evaluations.

**Table 1.** The numerical results with $x_0^2 = -(1, 1, \ldots, 1)^T$.

| | | SDCG | DFCG | DFPB1 | MRMIL1 |
|---|---|---|---|---|---|
| **Prob.** | ***n*** | **NF/Ni/T** | **NF/Ni/T** | **NF/Ni/T** | **NF/Ni/T** |
| P1 | 500 | 60/8/0.047 | 74/10/0.017 | 46/14/0.036 | 46/15/0.036 |
| | 1000 | 63/10/0.054 | 99/13/0.050 | 70/17/0.047 | 69/18/0.045 |
| | 3000 | 75/11/0.167 | 275/22/0.350 | 142/25/0.172 | 145/27/0.178 |
| | 5000 | 84/13/0.266 | 235/27/0.450 | 200/31/0.427 | 197/32/0.407 |
| | 10,000 | 109/18/0.707 | 421/40/1.800 | 308/41/1.307 | 308/42/1.290 |
| P2 | 500 | 25/7/0.016 | 117/14/0.034 | 82/13/0.024 | 91/15/0.028 |
| | 1000 | 30/10/0.017 | 148/17/0.066 | 116/17/0.056 | 372/49/0.201 |
| | 3000 | 57/15/0.100 | 243/25/0.283 | 217/25/0.254 | 218/26/0.26 |
| | 5000 | 76/18/0.183 | 317/31/0.600 | 287/31/0.584 | 294/32/0.597 |
| | 10,000 | 114/22/0.550 | 466/42/1.850 | 436/41/1.795 | 440/43/1.792 |
| P3 | 500 | 92/15/0.048 | 251/33/0.100 | 245/36/0.110 | 242/38/0.100 |
| | 1000 | 125/24/0.100 | 413/46/0.200 | 377/48/0.101 | 372/49/0.200 |
| | 3000 | 220/30/0.452 | 844/78/1.100 | 711/80/1.100 | 769/81/1.201 |
| | 5000 | 297/39/0.864 | 1128/99/2.600 | 1058/101/2.501 | 1056/102/2.600 |
| | 10,000 | 462/60/2.584 | 1698/138/8.200 | 1618/140/7.900 | 1619/141/7.902 |
| P4 | 500 | 58/15/0.034 | 198/33/0.095 | 79/20/0.054 | 172/33/0.118 |
| | 1000 | 62/17/0.134 | 213/38/0.265 | 91/20/0.174 | 196/37/0.362 |
| | 3000 | 77/23/1.233 | 267/45/2.740 | 192/37/2.115 | 259/45/2.693 |
| | 5000 | 88/28/2.984 | 315/47/7.075 | 242/42/5.861 | 299/50/6.893 |
| | 10,000 | 117/35/13.066 | 425/60/34.242 | 328/47/27.705 | 392/57/31.735 |
| P5 | 500 | 72/14/0.066 | 195/26/0.012 | 166/23/0.017 | 177/27/0.015 |
| | 1000 | 75/16/0.183 | 248/29/0.142 | 210/26/0.250 | 230/32/0.132 |
| | 3000 | 85/20/1.467 | 369/39/3.052 | 351/37/3.104 | 377/43/3.181 |
| | 5000 | 96/26/3.380 | 495/50/9.569 | 452/47/9.283 | 479/50/9.802 |
| | 10,000 | 120/30/14.151 | 695/62/50.202 | 648/56/49.591 | 678/63/50.867 |
| P6 | 500 | 64/10/0.050 | 81/12/0.023 | 39/15/0.015 | 34/11/0.020 |
| | 1000 | 65/13/0.067 | 82/14/0.053 | 61/18/0.033 | 59/17/0.033 |
| | 3000 | 69/15/0.149 | 159/23/0.196 | 127/26/0.176 | 124/26/0.158 |
| | 5000 | 76/18/0.284 | 210/29/0.446 | 184/32/0.405 | 176/28/0.370 |
| | 10,000 | 95/22/0.650 | 315/39/1.375 | 290/43/1.319 | 277/38/1.209 |
| P7 | 500 | 73/12/0.045 | 195/26/0.049 | 166/23/0.100 | 177/27/0.048 |
| | 1000 | 76/15/0.056 | 230/27/0.093 | 625/108/0.310 | 225/31/0.098 |
| | 3000 | 93/17/0.183 | 364/38/0.384 | 897/145/1.201 | 362/40/0.395 |
| | 5000 | 108/20/0.317 | 486/46/0.862 | 1059/164/2.600 | 469/48/0.844 |
| | 10,000 | 139/24/0.783 | 677/57/2.470 | 1255/176/6.502 | 664/60/2.525 |
| P8 | 500 | 2/1/0.001 | 2/1/0.032 | 2/1/0.002 | 2/1/0.001 |
| | 1000 | 2/1/0.003 | 2/1/0.002 | 2/1/0.003 | 2/1/0.003 |
| | 3000 | 2/1/0.006 | 2/1/0.005 | 2/1/0.006 | 2/1/0.006 |
| | 5000 | 2/1/0.017 | 2/1/0.008 | 2/1/0.007 | 2/1/0.008 |
| | 10,000 | 2/1/0.015 | 2/1/0.011 | 2/1/0.015 | 2/1/0.013 |

We employ the performance profiles [21], which are defined by the following fraction:

$$\rho_v(\tau) = \frac{1}{|P|}\left|\left\{p \in P : \log_2\left(\frac{t_{p,v}}{\min\{t_{p,v} : v \in V\}}\right) \leq \tau\right\}\right|.$$

to describe the performance of the algorithms. Here, $P$ is the test set, $|P|$ is the number of problems in the test set $P$, $V$ is the set of optimization solvers, and $t_{p,v}$ is the CPU time (or the number of function evaluations, or the number of iterations) for $p \in P$ and $v \in V$.

We analyze the values of parameters $\xi$, $\kappa$, and $r$ in SDCG, among which the value of parameter $r$ directly affects the search direction and its descent property. Firstly, we investigate the numerical results corresponding to parameter values of $r = 0.0001$, 0.05, 0.1, and 1 in terms of the number of function evaluations, the number of iterations, and CPU

time. We focus on parameter $r$ which affects the search direction and descent property, and test $r = 0.0001, 0.05, 0.1$, and 1. Figure 1 shows that $r = 0.0001, 0.05$, and 0.1 yield nearly the same function evaluation counts, with overlapping curves. In contrast, $r = 1$ requires more function evaluations, reducing efficiency. Figure 2 confirms this trend: $r = 0.0001, 0.05$, and 0.1 have similar iteration counts, while $r = 1$ leads to more iterations. Together, these figures show that $r$ in 0.0001–0.1 maintains algorithm efficiency, supporting our choice of $r = 0.0001$ for subsequent experiments. Furthermore, based on the CPU time presented in Figure 3, we can see that $r = 0.0001$ and $r = 1$ perform better than $r = 0.05$ and $r = 0.1$. And the algorithm achieves the most stable overall performance when $r = 0.0001$. Therefore, the value of $r = 0.0001$ is adopted in subsequent experiments.

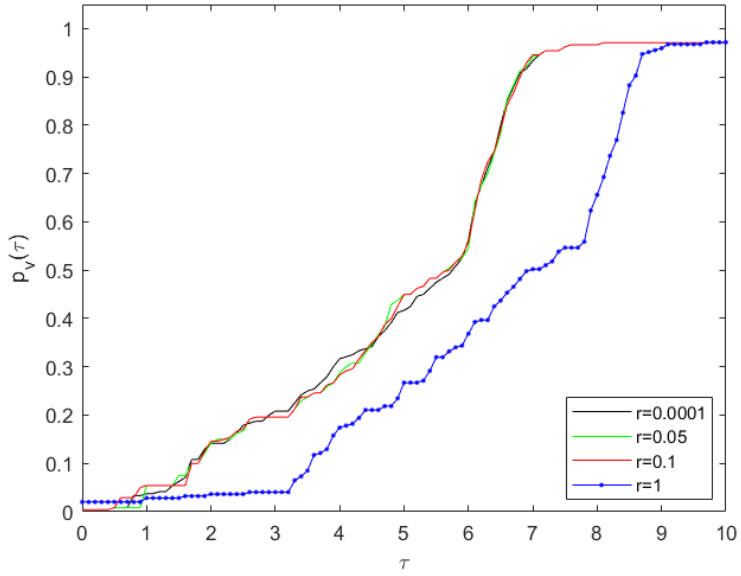


**Figure 1.** Performance profiles of SDCG with different $r$ based on the number of function evaluations.

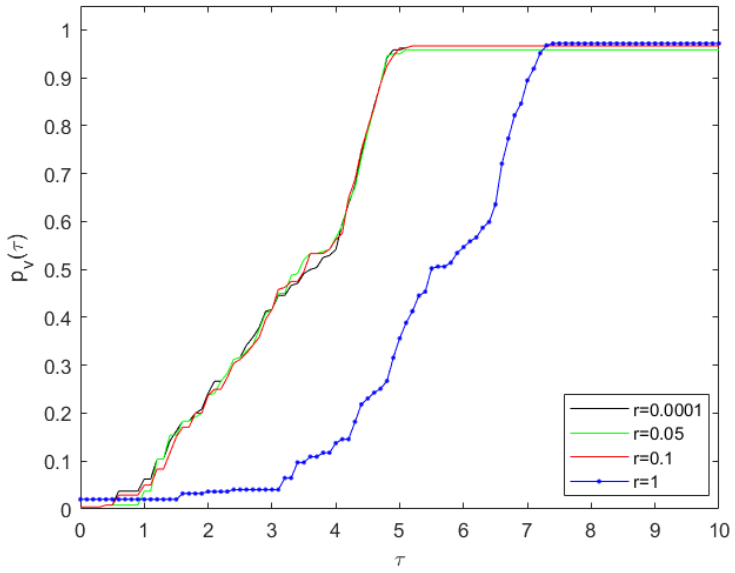


**Figure 2.** Performance profiles of SDCG with different $r$ based on the number of iterations.

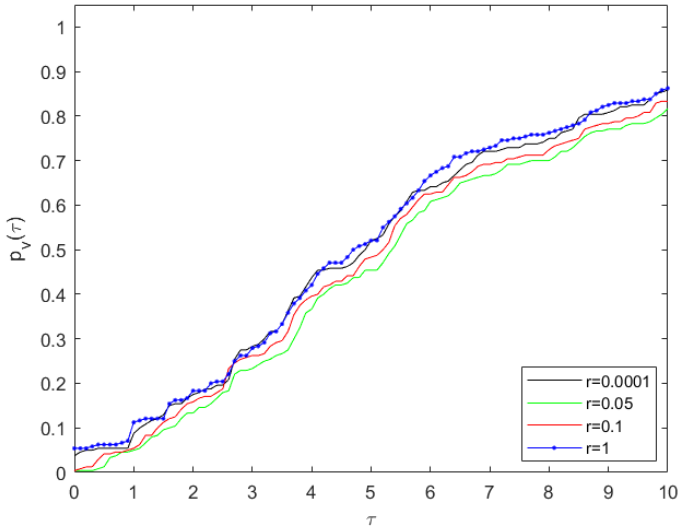


**Figure 3.** Performance profiles of SDCG with different $r$ based on the CPU time.

Next, we investigate the impacts of the initial step size $\xi$ and parameter $\kappa$ on the algorithm performance. In SDCG, we adopt $d_k$ as the search direction, and solve test problems by setting the initial step size to $\xi = 0.1$, 0.5, and 1 and parameter $\kappa$ to 1 and 1.9, respectively, with detailed results presented in Figures 4–6. When $\xi = 0.1$, $\kappa = 1.9$, and $r = 0.0001$, SDCG achieves superior performance in terms of the number of function evaluations and iterations, while its CPU time curve is slightly higher than those under other parameter configurations. This indicates that with such parameter settings, the algorithm exhibits favorable stability and requires a shorter average time to solve test problems.

Based on the above experimental results, we further evaluate the numerical performance of the search direction of SDCG under the parameter configuration of $\xi = 0.1$, $\kappa = 1.9$, and $r = 0.0001$, and conduct a comparative analysis with the DFCG, DFPB1 and MRMIL1 algorithms.

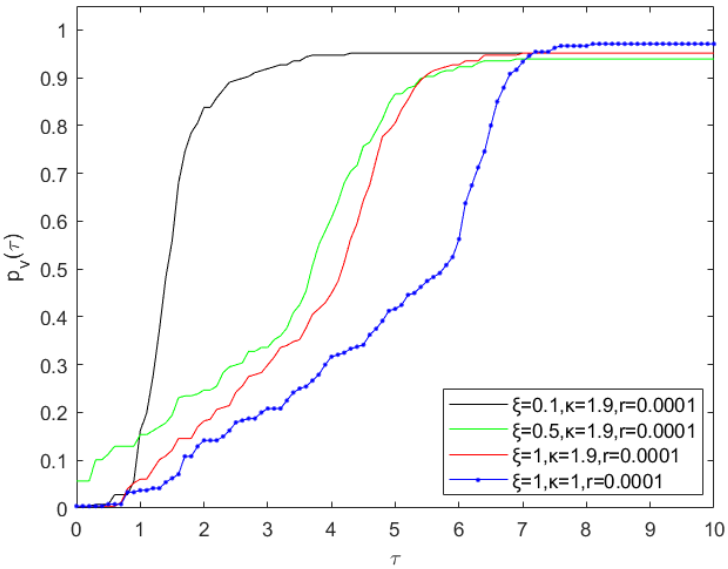


**Figure 4.** Performance profiles of SDCG with different $\xi$ and $\kappa$ based on the number of function evaluations.

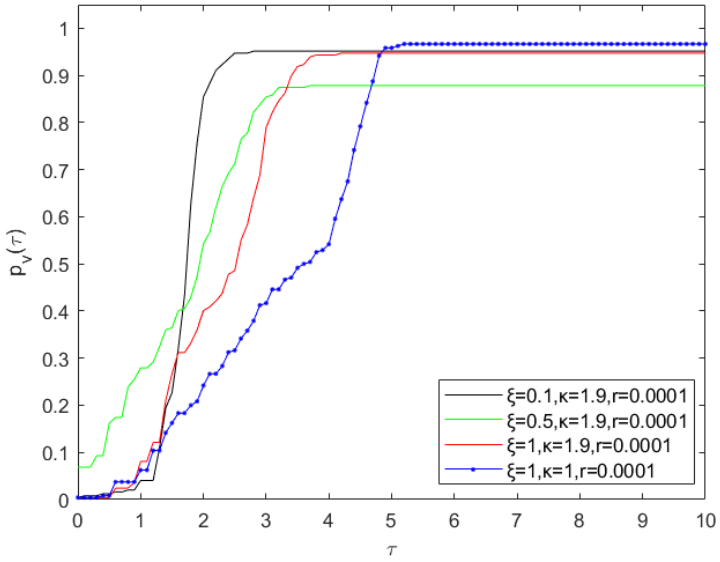


**Figure 5.** Performance profiles of SDCG with different $\xi$ and $\kappa$ based on the iterations.

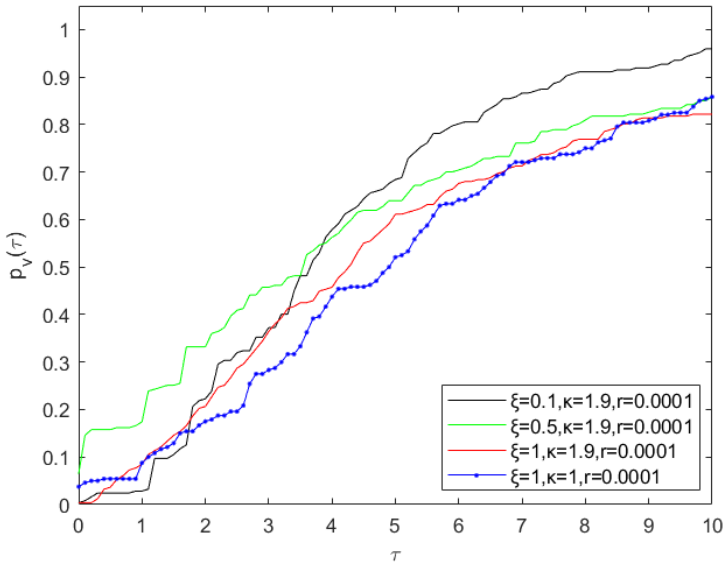


**Figure 6.** Performance profiles of SDCG with different $\xi$ and $\kappa$ based on the CPU time.

Figures 7–9 plot the performance curves of our proposed SDCG against those of the DFCG, MRMIL1, and DFPB1 algorithms, with a focus on three key metrics: the number of function evaluations, the number of iterations, and CPU time. Specifically, Figure 7 verifies that SDCG outperforms DFCG, DFPB1, and MRMIL1 in terms of the number of function evaluations. In Figure 8, it is clear that DFCG, MRMIL1, and DFPB1 yield better performance in iteration counts when $\tau < 1$; however, once $\tau \geq 1$, our proposed algorithm regains and maintains its superior performance. Figure 9 further underscores the advantages of SDCG: it can solve large-scale nonlinear monotone systems more rapidly and consistently stands out among the four tested algorithms.

Based on the above findings, we can conclude that the SDCG is significantly superior to the other three methods. This result demonstrates that our newly proposed subspace derivative-free conjugate gradient method exhibits strong stability when tackling large-scale nonlinear monotone systems: it reduces the required number of iterations and function evaluations, shortens CPU running time, and thus improves the overall efficiency.

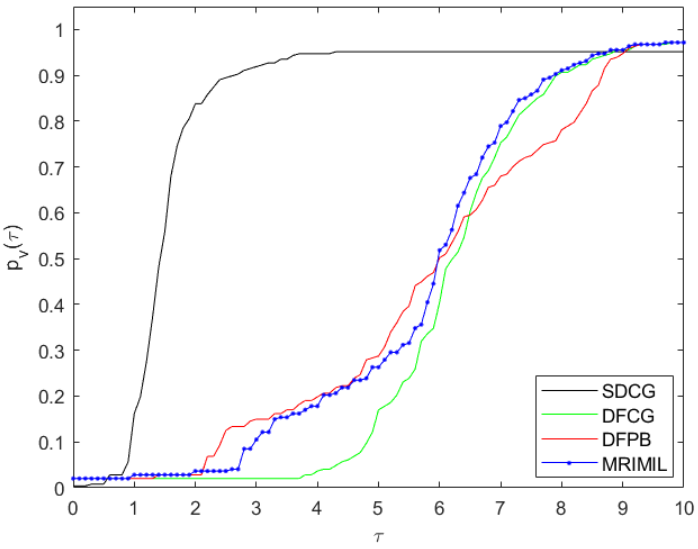


**Figure 7.** Performance profiles of different algorithms based on the number of function evaluations.

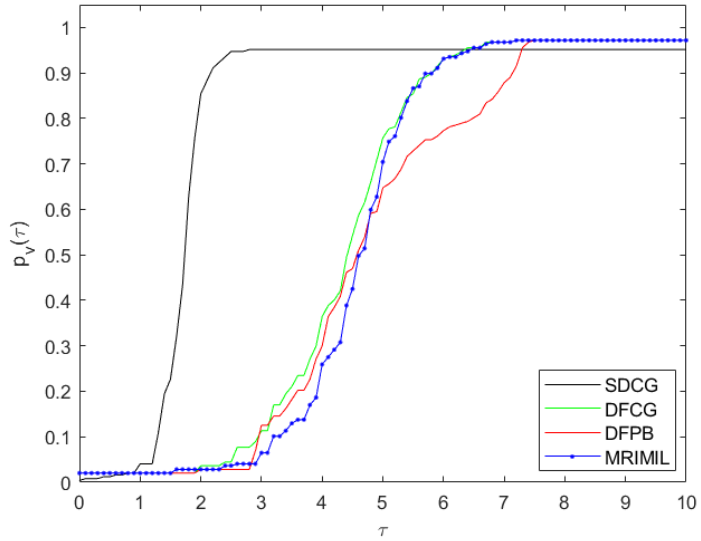


**Figure 8.** Performance profiles of different algorithms based on the number of iterations.

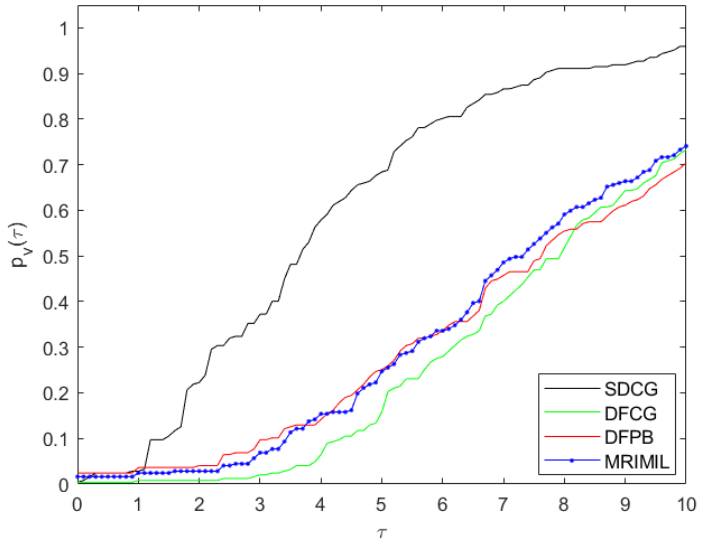


**Figure 9.** Performance profiles of different algorithms based on the CPU time.

## 5. Conclusions

In this research, we have successfully developed a novel and highly efficient algorithm. We initiate by meticulously analyzing the sufficient descent property of the search direction. This step was crucial as it provided a solid theoretical foundation for the subsequent convergence analysis. Through rigorous mathematical derivations and logical reasoning, we were able to establish both global convergence and convergence and the R-linear convergence rate. The numerical results obtained were highly encouraging. They clearly demonstrated that the proposed method outperforms many existing algorithms in terms of computational efficiency, accuracy, and robustness when dealing with nonlinear monotone equations with convex constraints.

However, we also recognize that there are still significant areas for further exploration. These models can potentially capture the complex characteristics of the equations more accurately, leading to more efficient and effective algorithms. This will be the primary focus of our future research, where we aim to push the boundaries of current knowledge and develop even more advanced techniques for solving these challenging problems.

**Author Contributions:** Data curation, Z.L., Z.F., R.M. and S.L.; Software, Z.L. and S.L.; Methodology, Z.L. and M.C.; Writing—original draft, Z.L., Z.F. and R.M.; Writing—review & editing, Z.F., Y.Y. and R.M.; Investigation, Z.F. and R.M.; Funding acquisition, M.C.; Supervision, M.C. and Y.Y.; Validation, Y.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work is supported by the Youth Science and Technology Talent Cultivation Project of Jilin Province (20250602005RC), General Program of Science and Technology Department of Jilin Province (20260102232JC), Graduate Innovation Project of Beihua University (2025039), and the Research Project on Higher Education Teaching Reform in Jilin Province (JLJY202529968143).

**Data Availability Statement:** The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

**Acknowledgments:** The authors are grateful to the editor and the anonymous reviewers for their valuable comments and suggestions, which have substantially improved this paper.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Sabi'u, J.; Al-Kawaz, R. An efficient modified conjugate gradient parameter for solving the system of symmetric nonlinear equations with application in motion control of coplanar robot. *Comput. Appl. Math.* **2025**, *44*, 49. [CrossRef]
2. Zhang, Z.; Liu, J. A fourth-order nonlinear equation studied by using a multivariate bilinear neural network method. *Nonlinear Dyn.* **2024**, *112*, 10229–10237. [CrossRef]
3. Hu, Y.; Wang, Y. An efficient projected gradient method for convex constrained monotone equations with applications in compressive sensing. *J. Appl. Math. Phys.* **2020**, *8*, 983–998. [CrossRef]
4. Muangchoo, K.; Phiangsungnoen, S. Hybrid CG-Like algorithm for nonlinear equations and image restoration. *Carpathian J. Math.* **2025**, *41*, 171–191. [CrossRef]
5. Sherman, A.H. On Newton-iterative methods for the solution of systems of nonlinear equations. *SIAM J. Numer. Anal.* **1978**, *15*, 755–771. [CrossRef]
6. Wang, X.; Tao, Y. A new Newton method with memory for solving nonlinear equations. *Mathematics* **2020**, *8*, 108. [CrossRef]
7. Fang, X.; Ni, Q.; Zeng, M. A modified quasi-newton method for nonlinear equations. *J. Appl. Math. Phys.* **2018**, *328*, 44–58. [CrossRef]
8. Berahas, A.S.; Byrd, R.H.; Nocedal, J. Derivative-free optimization of noisy functions via quasi-newton methods. *SIAM J. Optimiz.* **2019**, *29*, 965–993. [CrossRef]
9. Ma, C.; Jiang, L. Some research on Levenberg–Marquardt method for the nonlinear equations. *Appl. Math. Comput.* **2007**, *184*, 1032–1040. [CrossRef]
10. Chen, L.; Ma, Y. Shamanskii-like Levenberg-Marquardt method with a new line search for systems of nonlinear equations. *J. Syst. Sci. Complex.* **2020**, *33*, 1694–1707. [CrossRef]

11. Zheng, L.; Chen, L.; Ma, Y. A variant of the Levenberg-Marquardt method with adaptive parameters for systems of nonlinear equations. *AIMS Math.* **2022**, *7*, 1241–1256. [CrossRef]
12. Esmaeili, H.; Kimiaei, M. A new adaptive trust-region method for system of nonlinear equations. *Appl. Math. Model.* **2014**, *38*, 3003–3015. [CrossRef]
13. Porcelli, M.; Toint, P.L. Exploiting problem structure in derivative free optimization. *ACM Trans. Math. Softw.* **2022**, *48*, 1–25. [CrossRef]
14. Hare, W.; Roberts, L.; Royer, C. Expected decrease for derivative-free algorithms using random subspaces. *Math. Comp.* **2025**, *94*, 277–304. [CrossRef]
15. Chen, Y.; Hare, W.; Wiebe, A. Q-fully quadratic modeling and its application in a random subspace derivative-free method. *Comput. Optim. Appl.* **2024**, *89*, 317–360. [CrossRef]
16. Kimiaei, M.; Hassan Ibrahim, A.; Ghaderi, S. A subspace inertial method for derivative-free nonlinear monotone equations. *Optimization* **2025**, *74*, 269–296. [CrossRef]
17. Xie, P.; Yuan, Y. A new two-dimensional model-based subspace method for large-scale uncon strained derivative-free optimization: 2d-mosub. *Optim. Method. Softw.* **2025**, *41*, 118–150. [CrossRef]
18. Giovannelli, T.; Liuzzi, G.; Lucidi, S.; Rinaldi, F. Derivative-free methods for mixed-integer nonsmooth constrained optimization. *Comput. Optim. Appl.* **2022**, *82*, 293–327. [CrossRef]
19. Birgin, E.G.; Martínez, J.M. Accelerated derivative-free nonlinear least-squares applied to the estimation of manning coefficients. *Comput. Optim. Appl.* **2022**, *81*, 689–715. [CrossRef]
20. Kimiaei, M.; Neumaier, A. Effective matrix adaptation strategy for noisy derivative-free optimization. *Math. Prog. Comp.* **2024**, *16*, 459–501. [CrossRef]
21. Mohd Sham, N.D.S.; Mohamed, N.S. Derivative-free hybrid conjugate gradient method for constrained nonlinear equations with projection in line searches. *Matematika* **2025**, *41*, 123–134. [CrossRef]
22. Fang, X. A derivative-free method with the symmetric rank-one update for constrained nonlinear systems of monotone equations. *Symmetry* **2025**, *17*, 1956. [CrossRef]
23. Yuan, Y.; Stoer, J. A subspace study on conjugate gradient algorithms. *J. Appl. Math. Mech.* **1995**, *75*, 69–77. [CrossRef]
24. Tang, B.; Liu, J.; Peng, Y.; Huang, W.; Tao, Y. A subspace minimization conjugate gradient method for nonlinear monotone equations with convex constraints. *RAIRO Oper. Res.* **2025**, *59*, 2437–2449. [CrossRef]
25. Arfat, Y.; Khan, M.A.A.; Kumam, P. A hybrid steepest-descent approximants scheme for convex minimization over split equilibrium and fixed point problems. *Fixed Point Theory* **2025**, *26*, 377.
26. Arfat, Y.; Kumam, P.; Khan, M.A.A.; Cho, Y.J. A hybrid steepest-descent algorithm for convex minimization over the fixed point set of multivalued mappings. *Carpathian J. Math.* **2023**, *39*, 303–314. [CrossRef]
27. Yang, Y.; Chen, Y.; Lu, Y. A subspace conjugate gradient algorithm for large-scale uncon strained optimization. *Numer. Algor.* **2017**, *76*, 813–828. [CrossRef]
28. Zhang, L.; Zhou, W. Spectral gradient projection method for solving nonlinear monotone equations. *J. Comput. Appl. Math.* **2006**, *196*, 478–484. [CrossRef]
29. Solodov, M.V.; Svaiter, B.F. A globally convergent inexact Newton method for systems of monotone equations. In *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*; Fukushima, M., Qi, L., Eds.; Springer: Boston, MA, USA, 1998; pp. 355–369.
30. Gao, J.; Li, Y.; Cao, M.; Yang, Y.; Bai, X. A derivative-free conjugate gradient method for large-scale nonlinear systems of monotone equations. *Commun. Math. Sci.* **2023**, *21*, 543–557. [CrossRef]
31. Ahookhosh, M.; Amini, K.; Bahrami, S. Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations. *Numer. Algor.* **2013**, *64*, 21–42. [CrossRef]
32. Fang, X. A class of new derivative-free gradient type methods for large-scale nonlinear systems of monotone equations. *J. Inequal. Appl.* **2020**, *2020*, 93. [CrossRef]