Hindawi Publishing Corporation
Abstract and Applied Analysis
Volume 2014, Article ID 305643, 12 pages
http://dx.doi.org/10.1155/2014/305643

Research Article

Sufficient Descent Conjugate Gradient Methods for Solving Convex Constrained Nonlinear Monotone Equations

San-Yang Liu, Yuan-Yuan Huang, and Hong-Wei Jiao

School of Mathematics and Statistics, Xidian University, Xi'an 710071, China

Correspondence should be addressed to San-Yang Liu; liusanyang@126.com and Yuan-Yuan Huang; yyuanhuang@126.com

Received 22 October 2013; Accepted 15 December 2013; Published 12 January 2014

Academic Editor: Zhongxiao Jia

Copyright © 2014 San-Yang Liu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Two unified frameworks of some sufficient descent conjugate gradient methods are considered. Combined with the hyperplane projection method of Solodov and Svaiter, they are extended to solve convex constrained nonlinear monotone equations. Their global convergence is proven under some mild conditions. Numerical results illustrate that these methods are efficient and can be applied to solve large-scale nonsmooth equations.

1. Introduction

Consider the constrained monotone equations

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

where $F : R^n \rightarrow R^n$ is continuous and satisfies the following monotonicity:

$$(F(x) - F(y))^T (x - y) \geq 0, \quad \forall x, y \in \Omega, \tag{2}$$

and $\Omega \subset R^n$ is a nonempty closed convex set. Under these conditions, the solution set X^* of problem (1) is convex [1]. This problem has many applications, such as the power flow equation [2, 3] and some variational inequality problems which can be converted into (1) by means of fixed point maps or normal maps if the underlying function satisfies some coercive conditions [4].

In recent years, the study of the iterative methods to solve problem (1) with $\Omega = R^n$ has received much attention. The pioneer work was introduced by Solodov and Svaiter in [5], where the proposed method was called inexact Newton method which combines elements of Newton method, proximal point method, and projection strategy and required that F is differentiable. Its convergence was proven without any regularity assumptions. And a further study about its convergence properties was given by Zhou and Toh [6]. Then utilizing the projection strategy in [5], Zhou and Li extended the BFGS methods [7] and the limited memory BFGS methods [8] to solve problem (1) with $\Omega = R^n$. A significant improvement is that these methods converge globally without requiring the differentiability of F.

Conjugate gradient methods are another class of numerical methods [9–15] after spectral gradient methods [16–18] extended to solve problem (1), and the study of this aspect is just catching up. As is well known, conjugate gradient methods are very efficient to solve large-scale unconstrained nonlinear optimization problem

$$\min f(x), \quad x \in R^n, \tag{3}$$

where f is smooth, due to their simple iterations and their low memory requirements. In [19], they were divided into three categories, that is, early conjugate gradient methods, descent conjugate gradient methods, and sufficient descent conjugate gradient methods. Early conjugate gradient methods rarely ensure a (sufficient) descent condition

$$g_k^T d_k \leq -c\|g_k\|^2, \quad \forall k \geq 0, \ c > 0, \tag{4}$$

where $g_k = g(x_k)$ is the gradient of f at x_k (the kth iteration) and d_k is a search direction, while the later two categories always satisfy the descent property. One well-known sufficient descent conjugate gradient method, namely, CG_DESCENT, was presented by Hager and Zhang [20, 21] and satisfied the sufficient descent condition (4) with $c = 7/8$.

Inspired by Hager and Zhang's work, a unified framework of some sufficient descent conjugate gradient methods was presented in [19, 22]. And by the use of Gram-Schmidt orthogonalization, the other unified framework of some sufficient descent conjugate gradient methods was presented in [23].

Although conjugate gradient methods have been investigated extensively for solving unconstrained optimization problems, the study of them to solve nonlinear monotone equations is relatively rare. For the unconstrained case of monotone equations, Cheng [10] first introduced a PRP type method which is a combination of the well-known PRP conjugate gradient method [24, 25] and the hyperplane projection method [5]. Then some derivative-free methods were presented [11–13] which also belong to the conjugate gradient scheme. More recently, Xiao and Zhu [9] presented a modified version of the CG_DESCENT method to solve the constrained nonlinear monotone equations. And under some mild conditions, they proved that their proposed method is globally convergent. We have mentioned that there are two unified frameworks of some sufficient descent conjugate gradient methods, and the CG_DESCENT method belongs to one unified framework. Since the CG_DESCENT method can be used to solve the constrained monotone equations, then, it is natural for us to think about the two unified frameworks. So, in this paper, we extend the conjugate gradient methods who belong to the two unified frameworks to solve the constrained monotone equations and do some numerical experiments to test their efficiency.

The rest of this paper is organized as follows. In Section 2, the motivation to investigate two unified frameworks of some sufficient descent conjugate gradient methods is given. Then these methods are developed to solve problem (1) and are described by a model algorithm. In Section 3, we prove the global convergence of the model algorithm under some mild conditions. In Section 4, we give several specific versions of the model algorithm, test them over some test problems, and compare their numerical performance with that of the conjugate gradient method proposed in [9]. Finally, some conclusions are given in Section 5.

2. Motivation and Algorithms

In this section, we simply describe the hyperplane projection method of Solodov and Svaiter and introduce two classes of sufficient descent conjugate gradient methods for solving large-scale unconstrained optimization problems. Combined with the hyperplane projection method, we extend sufficient descent conjugate gradient methods to solve large-scale constrained nonlinear equations (1).

For convenience, we first give the definition of projection operator $P_\Omega(\cdot)$ which is defined as a mapping from R^n to its a nonempty closed convex subset Ω:

$$P_\Omega(x) = \arg\min\{\|y-x\| \mid y \in \Omega\}, \quad \forall x \in R^n. \tag{5}$$

And its two fundamental properties are

$$\|P_\Omega(x) - P_\Omega(y)\| \le \|x-y\|, \quad \forall x, y \in R^n, \tag{6}$$

$$(x - P_\Omega(x))^T (y - P_\Omega(x)) \le 0, \quad \forall x \in R^n,\ y \in \Omega. \tag{7}$$

Now, we recall the hyperplane projection method in [5] for the unconstrained case of problem (1). Let x_k $(k \ge 0)$ be the current iteration and $z_k = x_k + \alpha_k d_k$, where α_k is a step length obtained by means of a one-dimensional line search and d_k is a search direction. If x_k is not a solution and satisfies

$$(x_k - z_k)^T F(z_k) > 0, \tag{8}$$

then the hyperplane

$$H_k = \{x \in R^n \mid (x - z_k)^T F(z_k) = 0\} \tag{9}$$

strictly separates the current iteration x_k from the solution set of problem (1). By the property (7) of the projection operator, it is not difficult to verify that

$$P_{H_k}(x_k) = x_k - \frac{(x_k - z_k)^T F(z_k)}{\|F(z_k)\|^2} F(z_k) \tag{10}$$

is closer to the solution set than the iteration x_k. Then the next iteration is generated by $x_{k+1} = P_{H_k}(x_k)$.

We consider the iterative scheme of conjugate gradient methods for solving the unconstrained optimization problem (3). For any given starting point $x_0 \in R^n$, a sequence $\{x_k\}$ is generated by the following recursive relation:

$$x_{k+1} = x_k + \alpha_k d_k, \tag{11}$$

where α_k is a steplength and d_k is a descent direction. One way to generate d_k is

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -g_k + \beta_k d_{k-1}, & \text{if } k \ge 1, \end{cases} \tag{12}$$

where $g_k = g(x_k)$ and β_k is a scalar. The formula of β_k in the CG_DESCENT method is defined as

$$\beta_k^{HZ} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}} - \frac{2\|y_{k-1}\|^2}{(d_{k-1}^T y_{k-1})^2} g_k^T d_{k-1}, \tag{13}$$

where $y_{k-1} = g_k - g_{k-1}$. Then the direction d_k from (12) satisfies the sufficient descent condition (4) with $c = 7/8$. For more efficient versions of the CG_DESCENT method, please refer to [26, 27].

In [19, 22], a generalization of (13) was given by

$$\beta_k^G = \frac{g_k^T b_k}{a_k} - \frac{C\|b_k\|^2}{a_k^2} g_k^T d_{k-1}, \tag{14}$$

where $a_k \in R$, $b_k \in R^n$, and $C > 1/4$. Obviously, β_k^{HZ} is a special case of (14) with $a_k = d_{k-1}^T y_{k-1}$, $b_k = y_{k-1}$, and $C = 2$. More recently, Xiao and Zhu [9] presented a modified version of the CG_DESCENT method to solve the constrained problem (1). This work inspires us to extend the general case (14) to solve problem (1). So, we define

$$d_k = \begin{cases} -F_k, & \text{if } k = 0, \\ -F_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{15}$$

where $F_k = F(x_k)$ and the scalar β_k is defined as

$$\beta_k = \frac{F_k^T b_k}{a_k} - \frac{\theta_k \|b_k\|^2}{a_k^2} F_k^T d_{k-1} \tag{16}$$

with $\epsilon\|d_{k-1}\| \leq a_k \in R$ ($\epsilon > 0$), $\theta_k > 1/2$, and $b_k \in R^n$. Moreover, the formula of β_k proposed by Xiao and Zhu [9] corresponds to (16) with $\theta_k = 2, a_k = d_{k-1}^T y_{k-1}^*$, and $b_k = y_{k-1}^*$, where $y_k^* = y_k + \lambda_k \alpha_k \|F_k\| d_k$, $y_k = F_{k+1} - F_k$, and $\lambda_k = 1 + \|F_k\|^{-1} \max\{0, -(\alpha_k d_k^T y_k^* / \|\alpha_k d_k\|^2)\}$.

The other general way of producing sufficient descent conjugate gradient methods for solving the unconstrained optimization (3) was provided in [23]. By using the Gram-Schmidt orthogonalization, the search direction d_k is generated by

$$d_k = \begin{cases} -g_k, & \text{if } k = 0, \\ -\left(1 + \beta_k \dfrac{g_k^T d_{k-1}}{\|g_k\|^2}\right) g_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{17}$$

where β_k is a scalar, and its definition could be the same as that in (12). Obviously, it always satisfies $g_k^T d_k = -\|g_k\|^2$. In this paper, we will prove that the class of sufficient descent conjugate gradient methods can also be extended to solve problem (1) with the corresponding search direction d_k defined as

$$d_k = \begin{cases} -F_k, & \text{if } k = 0, \\ -\left(1 + \beta_k \dfrac{F_k^T d_{k-1}}{\|F_k\|^2}\right) F_k + \beta_k d_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{18}$$

where the formula of β_k could be (16).

Now we introduce the two unified frameworks of some sufficient descent conjugate gradient methods to solve problem (1) by adopting the projection strategy in [5]. We state the steps of the model algorithm as follows.

Algorithm 1.

Step 0. Choose an initial point $x_0 \in \Omega$, $\rho > 0$, $\sigma \in (0,1)$, $t \in (0,1)$ and $\epsilon > 0$. Set $k := 0$.

Step 1. If $\|F(x_k)\| \leq \epsilon$, stop. Otherwise, generate d_k by certain iteration formula which satisfies sufficient descent condition.

Step 2. Let α_k be the largest $\alpha \in \{\rho t^j \mid j = 0, 1, 2, \ldots\}$ such that

$$-F(x_k + \alpha d_k)^T d_k \geq \sigma\alpha \|F(x_k + \alpha d_k)\| \|d_k\|^2, \tag{19}$$

and then compute $z_k = x_k + \alpha_k d_k$.

Step 3. Compute the new iterate x_{k+1} by

$$x_{k+1} = P_\Omega\left(x_k - \frac{(x_k - z_k)^T F(z_k)}{\|F(z_k)\|^2} F(z_k)\right). \tag{20}$$

Set $k := k + 1$. Go to Step 1.

(i) If the search direction d_k in Step 1 is generated by the formula (15), we name the algorithm as Algorithm 1(a). And if the search direction d_k in Step 1 is generated by the formula (18), we name the algorithm as Algorithm 1(b).

3. Convergence Analysis

In this section, we analyze the convergence properties of Algorithm 1. We first make the following assumptions.

Assumption 2. The mapping F is L-Lipschitz continuous on the nonempty closed convex set Ω; that is, there exists a constant $L > 0$ such that

$$\|F(x) - F(y)\| \leq L\|x - y\|, \quad \forall x, y \in \Omega. \tag{21}$$

Assumption 3. The solution set X^* of problem (1) is nonempty.

Assumption 4. The parameter β_k satisfies inequality

$$|\beta_k| \leq \frac{\gamma\|F_k\|}{\|d_{k-1}\|}, \tag{22}$$

where γ is a positive number.

Assumption 4 is not difficult to satisfy. Taking the parameter β_k in (15) as an example, if there exists a large number $M < \infty$ such that $\|b_k\| \leq M$ for all k, then it satisfies the inequality (22). In fact,

$$\begin{aligned} |\beta_k| &= \left|\frac{F_k^T b_k}{a_k} - \frac{\theta_k\|b_k\|^2}{a_k^2} F_k^T d_{k-1}\right| \\ &\leq \left|\frac{F_k^T b_k}{a_k}\right| + \left|\frac{\theta_k\|b_k\|^2}{a_k^2} F_k^T d_{k-1}\right| \\ &\leq \frac{\|F_k\|\|b_k\|}{|a_k|} + \frac{\theta_k\|b_k\|^2}{a_k^2}\|F_k\|\|d_{k-1}\| \\ &\leq \frac{\|F_k\| M}{\epsilon\|d_{k-1}\|} + \frac{\theta_k M^2}{\epsilon^2\|d_{k-1}\|^2}\|F_k\|\|d_{k-1}\| \\ &= \left(\frac{M}{\epsilon} + \frac{\theta_k M^2}{\epsilon^2}\right)\frac{\|F_k\|}{\|d_{k-1}\|}. \end{aligned} \tag{23}$$

The following two lemmas show that the search direction d_k, no matter from (15) or (18), satisfies the sufficient descent condition.

Lemma 5. *If $a_k \neq 0$, $\theta_k > 1/2$, and d_k is generated by (15), then, for every $k \geq 0$,*

$$F_k^T d_k \leq -\left(1 - \frac{1}{4\theta_k}\right)\|F_k\|^2. \tag{24}$$

Table 1: Numerical results for Problem 10.

Init	Method	Dim = 5000 Iter/Nf/CPU	Dim = 10000 Iter/Nf/CPU	Dim = 20000 Iter/Nf/CPU	Dim = 30000 Iter/Nf/CPU
x_1	CGD_XZ	10/26/0.036	10/26/0.052	10/26/0.052	10/26/0.071
	Method 1	13/38/0.026	13/38/0.038	13/38/0.063	13/38/0.089
	Method 2	15/32/0.024	19/38/0.047	19/38/0.073	31/74/0.179
	Method 3	10/22/0.017	10/22/0.023	10/22/0.039	10/22/0.052
	Method 4	13/25/0.022	13/25/0.028	13/25/0.048	13/25/0.061
	Method 5	13/25/0.019	13/25/0.028	13/25/0.058	13/25/0.063
	Method 6	19/33/0.030	19/34/0.040	19/34/0.062	29/67/0.162
x_2	CGD_XZ	5/10/0.007	5/10/0.012	5/10/0.020	5/10/0.027
	Method 1	7/19/0.011	7/19/0.017	7/19/0.029	7/19/0.039
	Method 2	5/10/0.006	5/10/0.010	7/14/0.022	7/14/0.035
	Method 3	4/5/0.004	4/5/0.006	4/5/0.011	4/5/0.016
	Method 4	5/6/0.006	5/6/0.008	5/6/0.014	5/6/0.020
	Method 5	5/6 /0.005	5/6/0.008	5/6/0.013	5/6/0.020
	Method 6	5/6/0.005	5/6/0.009	6/8/0.017	6/8/0.023
x_3	CGD_XZ	4/8/0.006	4/8/0.010	4/8/0.016	4/8/0.021
	Method 1	7/18/0.010	7/18/0.015	7/18/0.029	7/18/0.037
	Method 2	4/7/0.005	4/7/0.008	4/7/0.013	4/7/0.018
	Method 3	3/4/0.003	3/4/0.005	3/4/0.009	3/4/0.012
	Method 4	4/5/0.005	4/5/0.007	4/5/0.011	4/5/0.015
	Method 5	4/5/0.005	4/5/0.006	4/5/0.012	4/5/0.015
	Method 6	4/5/0.004	4/5/0.007	4/5/0.011	4/5/0.014
x_4	CGD_XZ	15/47/0.027	19/60/0.055	17/51/0.080	21/76/0.141
	Method 1	14/47/0.025	17/59/0.048	17/58/0.077	21/75/0.137
	Method 2	14/28/0.016	14/28/0.026	14/28/0.047	14/28/0.064
	Method 3	18/28/0.022	18/28/0.031	18/28/0.053	18/28/0.076
	Method 4	14/21/0.017	14/21/0.023	14/21/0.043	14/21/0.056
	Method 5	10/15/0.012	10/15/0.017	10/15/0.028	10/15/0.040
	Method 6	21/33/0.025	21/33/0.038	21/33/0.060	21/33/0.085
x_5	CGD_XZ	16/41/0.027	24/65/0.066	19/49/0.084	19/51/0.118
	Method 1	17/49/0.029	17/55/0.044	19/60/0.084	18/58/0.111
	Method 2	13/26/0.017	13/26/0.026	13/26/0.045	13/26/0.060
	Method 3	23/35/0.028	23/35/0.040	23/35/0.067	23/35/0.096
	Method 4	22/34/0.025	22/34/0.040	22/34/0.068	22/34/0.093
	Method 5	21/30/0.023	21/30/0.036	21/30/0.060	21/30/0.082
	Method 6	25/41/0.033	24/40/0.046	24/40/0.070	24/40/0.095
x_6	CGD_XZ	16/41/0.027	23/56/0.060	18/47/0.073	19/51/0.112
	Method 1	16/51/0.027	15/47/0.038	17/54/0.075	17/54/0.101
	Method 2	13/26/0.015	13/26/0.026	13/26/0.043	13/26/0.058
	Method 3	23/35/0.026	23/35/0.040	23/35/0.067	23/35/0.092
	Method 4	22/34/0.025	22/34/0.042	22/34/0.067	22/34/0.088
	Method 5	21/30/0.024	21/30/0.035	21/30/0.060	21/30/0.085
	Method 6	25/41/0.029	24/40/0.042	24/40/0.072	24/40/0.097

Table 2: Numerical results for Problem 11.

Init	Method	Dim = 5000 Iter/Nf/CPU	Dim = 10000 Iter/Nf/CPU	Dim = 20000 Iter/Nf/CPU	Dim = 30000 Iter/Nf/CPU
x_1	CGD_XZ	5/12/0.019	5/12/0.025	5/12/0.033	5/12/0.039
	Method 1	7/21/0.023	7/21/0.030	7/21/0.039	7/21/0.054
	Method 2	5/13/0.015	5/14/0.017	5/14/0.028	5/15/0.039
	Method 3	12/25/0.022	12/25/0.036	12/25/0.047	12/25/0.065
	Method 4	5/8/0.009	5/8/0.012	5/8/0.017	5/8/0.025
	Method 5	5/8/0.009	5/8/0.014	5/8/0.017	5/8/0.025
	Method 6	5/10/0.008	5/11/0.022	5/11/0.021	5/12/0.030
x_2	CGD_XZ	3/6/0.004	3/6/0.009	3/6/0.013	3/6/0.018
	Method 1	6/17/0.011	6/17/0.018	6/17/0.027	6/17/0.039
	Method 2	3/6/0.004	3/6/0.007	3/6/0.012	3/6/0.015
	Method 3	9/18/0.014	9/18/0.022	9/18/0.034	9/18/0.046
	Method 4	3/4/0.004	3/4/0.006	3/4/0.009	3/4/0.013
	Method 5	3/4/0.004	3/4/0.006	3/4/0.010	3/4/0.013
	Method 6	3/4/0.004	3/4/0.006	3/4/0.009	3/4/0.013
x_3	CGD_XZ	14/41/0.028	14/41/0.043	14/41/0.073	14/41/0.101
	Method 1	5/13/0.008	5/13/0.013	5/13/0.021	5/13/0.031
	Method 2	13/37/0.022	13/37/0.036	13/37/0.061	13/37/0.086
	Method 3	10/20/0.015	10/20/0.024	10/20/0.040	10/20/0.056
	Method 4	14/28/0.021	14/28/0.033	14/28/0.054	14/28/0.075
	Method 5	14/28/0.021	14/28/0.032	14/28/0.055	14/28/0.080
	Method 6	14/28/0.020	14/28/0.032	14/28/0.052	14/28/0.076
x_4	CGD_XZ	17/50/0.034	17/50/0.051	17/50/0.086	17/50/0.124
	Method 1	20/68/0.038	17/50/0.044	19/54/0.085	18/51/0.118
	Method 2	18/44/0.025	16/38/0.040	17/42/0.072	17/42/0.096
	Method 3	20/40/0.028	21/42/0.043	20/40/0.075	18/36/0.094
	Method 4	15/30/0.020	15/30/0.034	15/30/0.057	15/30/0.080
	Method 5	16/32/0.026	16/32/0.039	16/32/0.063	16/32/0.083
	Method 6	22/41/0.031	19/38/0.042	19/38/0.066	19/38/0.097
x_5	CGD_XZ	21/59/0.044	21/68/0.068	17/52/0.089	16/42/0.106
	Method 1	19/61/0.033	18/56/0.051	17/50/0.078	16/48/0.107
	Method 2	12/23/0.016	12/23/0.023	12/23/0.043	12/23/0.063
	Method 3	17/34/0.024	17/34/0.036	17/34/0.063	17/34/0.088
	Method 4	32/52/0.040	32/52/0.070	31/50/0.101	31/50/0.148
	Method 5	21/35/0.028	21/35/0.043	21/35/0.073	21/35/0.101
	Method 6	29/39/0.034	27/36/0.048	27/36/0.079	27/36/0.112
x_6	CGD_XZ	21/59/0.041	22/68/0.071	20/62/0.097	16/42/0.104
	Method 1	17/59/0.035	19/67/0.058	18/63/0.092	18/52/0.118
	Method 2	14/27/0.020	14/27/0.031	14/27/0.050	14/27/0.074
	Method 3	17/34/0.023	17/34/0.040	17/34/0.060	17/34/0.090
	Method 4	30/48/0.042	31/50/0.060	31/50/0.103	31/50/0.143
	Method 5	21/35/0.028	21/35/0.040	21/35/0.073	21/35/0.101
	Method 6	29/39/0.032	27/36/0.046	27/36/0.083	27/36/0.112

Proof. Since $d_0 = -F_0$, then $F_0^T d_0 = -\|F_0\|^2$ which satisfies (24). For every $k \geq 1$, multiplying (15) by F_k, we have

$$\begin{aligned} F_k^T d_k &= -\|F_k\|^2 + \beta_k F_k^T d_{k-1} \\ &= -\|F_k\|^2 + \frac{F_k^T b_k}{a_k} F_k^T d_{k-1} - \frac{\theta_k \|b_k\|^2}{a_k^2} \left(F_k^T d_{k-1}\right)^2 \\ &= \frac{-\|F_k\|^2 a_k^2 + a_k \left(F_k^T b_k\right)\left(F_k^T d_{k-1}\right) - \theta_k \|b_k\|^2 \left(F_k^T d_{k-1}\right)^2}{a_k^2}. \end{aligned} \tag{25}$$

TABLE 3: Numerical results for Problem 12.

Init	Method	Dim = 5000 Iter/Nf/CPU	Dim = 10000 Iter/Nf/CPU	Dim = 20000 Iter/Nf/CPU	Dim = 30000 Iter/Nf/CPU
x_1	CGD_XZ	24/66/0.095	24/66/0.130	21/60/0.205	21/60/0.308
	Method 1	14/34/0.037	15/38/0.070	15/46/0.146	17/56/0.250
	Method 2	26/66/0.063	28/75/0.126	32/102/0.317	41/136/0.612
	Method 3	30/49/0.060	30/49/0.099	30/49/0.191	30/49/0.278
	Method 4	23/43/0.050	23/43/0.085	22/42/0.159	22/42/0.224
	Method 5	24/44/0.051	23/43/0.083	22/42/0.160	22/42/0.222
	Method 6	25/61/0.060	28/76/0.127	33/103/0.320	38/120/0.538
x_2	CGD_XZ	19/55/0.056	19/55/0.112	18/53/0.177	18/53/0.260
	Method 1	16/40/0.041	18/54/0.090	13/33/0.112	14/41/0.189
	Method 2	18/36/0.038	18/36/0.068	18/36/0.127	18/36/0.186
	Method 3	19/37/0.040	19/37/0.070	18/36/0.130	18/36/0.188
	Method 4	19/37/0.040	19/37/0.073	18/36/0.128	18/36/0.188
	Method 5	19/37/0.042	19/37/0.072	18/36/0.127	18/36/0.193
	Method 6	18/36/0.039	18/36/0.069	18/36/0.128	18/36/0.188
x_3	CGD_XZ	20/57/0.059	19/54/0.095	19/54/0.183	19/54/0.266
	Method 1	15/43/0.041	12/28/0.050	13/34/0.111	18/55/0.251
	Method 2	19/37/0.040	19/37/0.068	20/39/0.141	20/39/0.202
	Method 3	19/36/0.039	19/36/0.068	19/36/0.130	19/36/0.195
	Method 4	21/40/0.046	21/40/0.081	21/40/0.147	21/40/0.213
	Method 5	21/40/0.046	21/40/0.081	21/40/0.148	21/40/0.214
	Method 6	19/37/0.041	19/37/0.073	18/36/0.128	18/36/0.186
x_4	CGD_XZ	19/54/0.054	19/54/0.098	19/54/0.181	19/54/0.263
	Method 1	12/28/0.029	12/29/0.054	16/51/0.163	14/36/0.175
	Method 2	19/37/0.041	19/37/0.072	20/39/0.142	20/39/0.205
	Method 3	19/36/0.041	19/36/0.077	19/36/0.134	19/36/0.190
	Method 4	20/38/0.043	20/38/0.076	20/38/0.141	20/38/0.207
	Method 5	20/38/0.043	20/38/0.075	20/38/0.141	20/38/0.204
	Method 6	19/37/0.040	19/37/0.068	18/36/0.128	18/36/0.188
x_5	CGD_XZ	20/57/0.058	20/57/0.101	20/57/0.190	20/57/0.273
	Method 1	17/40/0.040	16/49/0.080	12/28/0.097	12/28/0.141
	Method 2	19/37/0.040	19/37/0.072	19/37/0.138	20/39/0.211
	Method 3	20/38/0.042	20/38/0.074	20/38/0.143	20/38/0.204
	Method 4	21/40/0.046	21/40/0.075	21/40/0.149	21/40/0.212
	Method 5	21/40/0.049	21/40/0.077	21/40/0.149	21/40/0.212
	Method 6	20/39/0.043	20/39/0.076	20/39/0.142	19/38/0.196
x_6	CGD_XZ	19/54/0.052	19/54/0.094	19/54/0.182	19/54/0.264
	Method 1	15/40/0.038	12/28/0.052	12/30/0.097	17/51/0.233
	Method 2	19/37/0.038	19/37/0.068	19/37/0.136	20/39/0.203
	Method 3	19/36/0.040	19/36/0.075	19/36/0.133	19/36/0.197
	Method 4	20/38/0.043	20/38/0.076	20/38/0.140	20/38/0.202
	Method 5	20/38/0.042	20/38/0.075	20/38/0.142	20/38/0.206
	Method 6	19/37/0.040	19/37/0.072	19/37/0.133	18/36/0.185

Table 4: Numerical results for Problem 13.

Method	Init = x_1 Iter/Nf/CPU	Init = x_2 Iter/Nf/CPU	Init = x_3 Iter/Nf/CPU
CGD_XZ	8032/100720/3.926	6960/88626/3.395	6608/83404/3.206
Method 1	14474/81644/4.073	14776/83135/4.152	14105/79821/3.985
Method 2	14547/80367/3.784	14535/79834/3.782	14233/78523/3.724
Method 3	15974/67287/3.949	15950/67116/3.929	15522/64072/3.830
Method 4	14289/77460/4.229	14583/78517/4.262	14034/76578/4.094
Method 5	14633/79306/4.089	14585/78940/4.017	14428/78917/4.018
Method 6	12394/61365/3.314	13192/64579/3.495	12500/60647/3.319
Method	Init = x_4 Iter/Nf/CPU	Init = x_5 Iter/Nf/CPU	Init = x_6 Iter/Nf/CPU
CGD_XZ	8018/100772/3.874	118/1232/0.050	7583/92362/3.612
Method 1	14482/81707/4.059	125/445/0.029	14523/81523/4.078
Method 2	14490/80119/3.795	65/209/0.014	14829/81427/3.871
Method 3	15571/65689/3.826	125/586/0.032	15802/66762/3.889
Method 4	14618/78866/4.250	132/450/0.032	14716/79737/4.267
Method 5	14605/79474/4.059	127/440/0.030	14530/79232/4.032
Method 6	12928/62759/3.474	126/536/0.031	12651/63054/3.378

Table 5: Number of times when each method was the fastest.

Method	Iteration metric	Function evaluation metric	Time metric
CGD_XZ	17	0	3
Method 1	28	16	19
Method 2	26	22	24
Method 3	12	19	13
Method 4	12	15	11
Method 5	12	15	10
Method 6	10	16	11

Denote $u_k = a_k F_k/\sqrt{2\theta_k}$ and $v_k = \sqrt{2\theta_k}(F_k^T d_{k-1})b_k$. By applying the inequality $u_k^T v_k \leq 1/2(\|u_k\|^2 + \|v_k\|^2)$ to the second term in (25), we obtain (24). □

The lemma above is similar to Theorem 1.1 in [20]. And from this lemma, we can see that the descent property of d_k from (15) is independent of any line search and choices of the parameters a_k and b_k. While different choices of the parameters a_k, b_k, and θ_k may yield very different numerical behaviors.

Lemma 6. *Let $\{d_k\}$ be the sequence generated by* (18)*, and then, for all $k \geq 0$, it holds that*

$$F_k^T d_k = -\|F_k\|^2 \leq -\left(1 - \frac{1}{4\theta_k}\right)\|F_k\|^2, \tag{26}$$

where $\theta_k > 1/2$.

Proof. The desired result is very easy to obtain. In fact, if $k = 0$, it is clear that $F_0^T d_0 = -\|F_0\|^2 \leq -(1 - (1/4\theta_0))\|F_0\|^2$. If $k \geq 1$, we have

$$\begin{aligned} F_k^T d_k &= F_k^T\left(-\left(1 + \beta_k \frac{F_k^T d_{k-1}}{\|F_k\|^2}\right)F_k + \beta_k d_{k-1}\right) \\ &= -\|F_k\|^2 \leq -\left(1 - \frac{1}{4\theta_k}\right)\|F_k\|^2. \end{aligned} \tag{27}$$

□

The lemma above indicates that the descent property of d_k from (18) is independent of the choices of β_k.

Lemma 7. *Suppose Assumptions 2 and 3 hold. Let α_k be the steplength involved in Algorithm 1, and let sequences $\{x_k\}$ and $\{z_k\}$ be generated by Algorithm 1. Then steplength α_k is well defined and satisfies the following inequality:*

$$\alpha_k \geq \min\left\{\rho, \frac{t(1-(1/4\theta_k))}{L + \sigma\left\|F\left(x_k + \alpha_k t^{-1} d_k\right)\right\|}\frac{\|F_k\|^2}{\|d_k\|^2}\right\}. \tag{28}$$

Proof. Suppose that, at kth iteration, x_k is not a solution, that is, $F_k \neq 0$, and, for all $j = 0, 1, \ldots$, inequality (19) fails to hold, and then

$$-F\left(x_k + \rho t^j d_k\right)^T d_k < \sigma\rho t^j \left\|F\left(x_k + \rho t^j d_k\right)\right\| \|d_k\|^2. \tag{29}$$

Since F is continuous, taking the limits with respect to j on the both sides of (29) yields

$$-F(x_k)^T d_k \leq 0, \tag{30}$$

which contradicts Lemmas 5 and 6. So, the step length α_k is well defined and can be determined within a finite number of trials.

Now, we prove inequality (28). If $\alpha_k \neq \rho$, then by using the selection of α_k, we have

$$-F\left(x_k + \alpha_k t^{-1} d_k\right)^T d_k < \sigma \alpha_k t^{-1} \left\| F\left(x_k + \alpha_k t^{-1} d_k\right) \right\| \left\| d_k \right\|^2. \tag{31}$$

Combining it with (24), (26), and the Lipschitz continuity of F yields

$$\begin{aligned}
&\left(1 - \frac{1}{4\theta_k}\right) \left\| F_k \right\|^2 \\
&\quad \leq -F_k^T d_k \\
&\quad = \left(F\left(x_k + \alpha_k t^{-1} d_k\right) - F_k\right)^T d_k - F\left(x_k + \alpha_k t^{-1} d_k\right)^T d_k \\
&\quad \leq L \alpha_k t^{-1} \left\| d_k \right\|^2 + \sigma \alpha_k t^{-1} \left\| F\left(x_k + \alpha_k t^{-1} d_k\right) \right\| \left\| d_k \right\|^2.
\end{aligned} \tag{32}$$

From Lemmas 5 and 6, we have that

$$\begin{aligned}
\left\| d_k \right\|^2 &= \left\| d_k + F_k - F_k \right\|^2 \\
&= \left\| d_k + F_k \right\|^2 - 2F_k^T d_k - \left\| F_k \right\|^2 \\
&\geq \left(1 - \frac{1}{2\theta_k}\right) \left\| F_k \right\|^2.
\end{aligned} \tag{33}$$

Since $F_k \neq 0$, then (33) indicates $d_k \neq 0$. So, it follows from inequality (32) that

$$\alpha_k \geq \frac{t\left(1 - \left(1/4\theta_k\right)\right)}{L + \sigma \left\| F\left(x_k + \alpha_k t^{-1} d_k\right) \right\|} \frac{\left\| F_k \right\|^2}{\left\| d_k \right\|^2}. \tag{34}$$

Then inequality (28) is obtained. □

Lemma 8. *Let $x^* \in X^*$ and let sequences $\{x_k\}$ and $\{F_k\}$ be generated by Algorithm 1. Then one has*

$$\left\| x_{k+1} - x^* \right\|^2 \leq \left\| x_k - x^* \right\|^2 - \sigma^2 \left\| x_k - z_k \right\|^4. \tag{35}$$

Furthermore,

$$\lim_{k \to \infty} \left\| x_k - z_k \right\| = \lim_{k \to \infty} \alpha_k \left\| d_k \right\| = 0. \tag{36}$$

And there exists a positive number M such that $\|F_k\| \leq M$ and $\|F(x_k + \alpha_k t^{-1} d_k)\| \leq M$ for all $k \geq 0$.

Proof. Since $x^* \in X^*$, then $F(x^*) = 0$ and $P_\Omega(x^*) = x^*$. Since the mapping F is monotone, then $F(z_k)^T (z_k - x^*) \geq 0$; further,

$$F(z_k)^T \left(x_k - x^*\right) \geq F(z_k)^T \left(x_k - z_k\right). \tag{37}$$

By using (19) and $z_k = x_k + \alpha_k d_k$, we have

$$F(z_k)^T \left(x_k - z_k\right) = -\alpha_k F(z_k)^T d_k \geq \sigma \alpha_k^2 \left\| d_k \right\|^2 \left\| F\left(z_k\right) \right\|, \tag{38}$$

which implies that

$$\frac{F(z_k)^T \left(x_k - z_k\right)}{\left\| F\left(z_k\right) \right\|} \geq \sigma \alpha_k^2 \left\| d_k \right\|^2 = \sigma \left\| x_k - z_k \right\|^2. \tag{39}$$

Obviously, $F(z_k)^T (x_k - z_k) \geq 0$. By using the property (7) of the projection operator $P_\Omega(\cdot)$ and (37), we have

$$\begin{aligned}
&\left\| x_{k+1} - x^* \right\|^2 \\
&\quad = \left\| P_\Omega\left(x_k - \frac{F(z_k)^T \left(x_k - z_k\right)}{\left\| F\left(z_k\right) \right\|^2} F\left(z_k\right)\right) - P_\Omega\left(x^*\right) \right\|^2 \\
&\quad \leq \left\| x_k - \frac{F(z_k)^T \left(x_k - z_k\right)}{\left\| F\left(z_k\right) \right\|^2} F\left(z_k\right) - x^* \right\|^2 \\
&\quad \leq \left\| x_k - x^* \right\|^2 - \frac{\left(F(z_k)^T \left(x_k - z_k\right)\right)^2}{\left\| F\left(z_k\right) \right\|^2}.
\end{aligned} \tag{40}$$

Substituting the second term in (40) by (39), inequality (35) follows.

The inequality (35) shows that the sequence $\{\|x_k - x^*\|\}$ is convergent, and then taking the limits with respect to k on the both sides of (35) yields (36).

Since F is Lipschitz continuous, then from (35), we have

$$\left\| F_k \right\| = \left\| F\left(x_k\right) - F\left(x^*\right) \right\| \leq L \left\| x_k - x^* \right\| \leq L \left\| x_0 - x^* \right\|. \tag{41}$$

And from (36), we know that there exists a positive number M' such that $\alpha_k \|d_k\| \leq M'$, and then

$$\begin{aligned}
\left\| F\left(x_k + \alpha_k t^{-1} d_k\right) \right\| &= \left\| F\left(x_k + \alpha_k t^{-1} d_k\right) - F\left(x^*\right) \right\| \\
&\leq L \left\| x_k + \alpha_k t^{-1} d_k - x^* \right\| \\
&\leq L \left\| x_0 - x^* \right\| + L M' t^{-1}.
\end{aligned} \tag{42}$$

Denote $M = L\|x_0 - x^*\| + LM' t^{-1}$; we have that $\|F_k\| \leq M$ and $\|F(x_k + \alpha_k t^{-1} d_k)\| \leq M$. □

Theorem 9. *Let Assumption 4 hold and let $\{d_k\}$ be a sequence generated by Algorithm 1. Then*

$$\left\| d_k \right\| \leq \left(1 + 2\gamma\right) \left\| F_k \right\|. \tag{43}$$

And one has

$$\liminf_{k \to \infty} \left\| F_k \right\| = 0. \tag{44}$$

Proof. If d_k is generated by (15), we have

$$\left\| d_k \right\| \leq \left\| F_k \right\| + \left| \beta_k \right| \left\| d_{k-1} \right\| \leq \left(1 + \gamma\right) \left\| F_k \right\|, \tag{45}$$

which satisfies (43). If d_k is generated by (18), then

$$\left\| d_k \right\| \leq \left\| F_k \right\| + 2 \left| \beta_k \right| \left\| d_{k-1} \right\|. \tag{46}$$

The inequality (43) is obtained easily. From Lemma 8, we know that there exists $0 < M < \infty$ such that $\|F_k\| \leq M$, and then

$$\|d_k\| \leq (1+2\gamma) M. \tag{47}$$

Suppose (44) does not hold, then there exists $\epsilon > 0$ such that

$$\|F_k\| \geq \epsilon, \quad \forall k \geq 0. \tag{48}$$

From (33), we have that $\|d_k\| \geq \sqrt{(1 - 1/(2\theta_k))}\|F_k\|$, which implies

$$\|d_k\| \geq \sqrt{\left(1 - \frac{1}{(2\theta_k)}\right)}\epsilon, \quad \forall k \geq 0. \tag{49}$$

By inequalities (28), (47), (48), and (49), we have

$$\begin{aligned}\alpha_k \|d_k\| &\geq \min\left\{\rho, \frac{t(1-(1/4\theta_k))}{L+\sigma\|F(x_k+\alpha_k t^{-1}d_k)\|}\frac{\|F_k\|^2}{\|d_k\|^2}\right\}\|d_k\| \\ &\geq \min\left\{\rho, \frac{t(1-(1/4\theta_k))}{L+\sigma M}\frac{\epsilon^2}{(1+2\gamma)^2 M^2}\right\} \\ &\quad \times \sqrt{\left(1-\frac{1}{(2\theta_k)}\right)}\epsilon \\ &> 0.\end{aligned} \tag{50}$$

This contradicts (36). So the conclusion (44) holds. □

4. Numerical Experiments

In this section, we give some specific versions of Algorithm 1 and investigate their numerical behaviors. Let us review the HS conjugate gradient method [28]. It generates search direction d_k by (12) and parameter β_k by

$$\beta_k^{HS} = \frac{g_k^T y_{k-1}}{d_{k-1}^T y_{k-1}}. \tag{51}$$

Among early conjugate gradient methods, the HS method is a relatively efficient one. And many conjugate gradient methods are its improved versions, such as the well-known CG_DESCENT method. Now based on the HS method and Assumption 4, we give several specific versions of Algorithm 1(a) as follows.

Method 1. Consider Algorithm 1(a) with

$$\begin{aligned}\beta_k^1 &= \frac{F_k^T y_{k-1}}{\max\left\{0.5 d_{k-1}^T y_{k-1} + 0.5\|F_{k-1}\|^2, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}\|^2}{\left(\max\left\{0.5 d_{k-1}^T y_{k-1} + 0.5\|F_{k-1}\|^2, \epsilon\|d_{k-1}\|\right\}\right)^2} \\ &\quad \times F_k^T d_{k-1}.\end{aligned} \tag{52}$$

Method 2. Consider Algorithm 1(a) with

$$\begin{aligned}\beta_k^2 &= \frac{F_k^T y_{k-1}}{\max\left\{\max\left\{d_{k-1}^T y_{k-1}, \|F_{k-1}\|^2\right\}, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}\|^2}{\left(\max\left\{\max\left\{d_{k-1}^T y_{k-1}, \|F_{k-1}\|^2\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2} \\ &\quad \times F_k^T d_{k-1}.\end{aligned} \tag{53}$$

Method 3. Consider Algorithm 1(a) with

$$\begin{aligned}\beta_k^3 &= \frac{F_k^T y_{k-1}^*}{\max\left\{d_{k-1}^T y_{k-1}^*, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}^*\|^2}{\left(\max\left\{d_{k-1}^T y_{k-1}^*, \epsilon\|d_{k-1}\|\right\}\right)^2} F_k^T d_{k-1},\end{aligned} \tag{54}$$

where $y_{k-1}^* = y_{k-1} + \alpha_{k-1} d_{k-1}$.

Since the definition of parameter β_k in Algorithm 1(b) could be the same as that in Algorithm 1(a), and the descent property of d_k in Algorithm 1(b) is independent of the choices of parameter β_k, we can give several specific versions of Algorithm 1(b) as follows.

Method 4. Consider Algorithm 1(b) with (52).

Method 5. Consider Algorithm 1(b) with

$$\begin{aligned}\beta_k^5 &= \frac{F_k^T y_{k-1}}{\max\left\{\max\left\{d_{k-1}^T y_{k-1}, -F_{k-1}^T d_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}} \\ &\quad - \frac{2\|y_{k-1}\|^2}{\left(\max\left\{\max\left\{d_{k-1}^T y_{k-1}, -F_{k-1}^T d_{k-1}\right\}, \epsilon\|d_{k-1}\|\right\}\right)^2} \\ &\quad \times F_k^T d_{k-1}.\end{aligned} \tag{55}$$

Method 6. Consider Algorithm 1(b) with

$$\beta_k^6 = \frac{F_k^T y_{k-1}}{\max\left\{d_{k-1}^T y_{k-1}, \epsilon\|d_{k-1}\|\right\}}. \tag{56}$$

From Lemma 8, we know that a sequence $\{F_k\}$ generated by Algorithm 1 is norm bounded. Then it is easy to verify that the parameters β_k in Methods 1–6 satisfy Assumption 4. So, from the convergence analysis in Section 3, we know that Methods 1–6 are convergent in the sense that $\liminf_{k\to\infty}\|F_k\| = 0$.

Next, we test the performance of Methods 1–6 via the following four constrained monotone problems and compare them with the method (abbreviated as CGD_XZ) in [9].

Problem 10 (see [29]). The mapping F is taken as $F(x) = (F_1(x), \ldots, F_n(x))^T$, where $F_i(x) = e^{x_i} - 1$, $i = 1, \ldots, n$, and $\Omega = R_+^n$.

Problem 11 (see [17]). The mapping F is taken as $F(x) = (F_1(x), \ldots, F_n(x))^T$, where

$$F_i(x) = x_i - \sin(|x_i - 1|), \quad i = 1, 2, \ldots, n, \tag{57}$$

and $\Omega = \{x \in R^n \mid \sum_{i=1}^n x_i \le n, x_i \ge 0, i = 1, 2, \ldots, n\}$.

Problem 12 (see [30]). The mapping F is taken as $F(x) = (F_1(x), \ldots, F_n(x))^T$, where

$$\begin{aligned} F_1(x) &= x_1 - e^{\cos((x_1+x_2)/(n+1))}, \\ F_i(x) &= x_i - e^{\cos((x_{i-1}+x_i+x_{i+1})/(n+1))}, \quad i = 2, 3, \ldots, n-1, \\ F_n(x) &= x_n - e^{\cos((x_{n-1}+x_n)/(n+1))}, \end{aligned} \tag{58}$$

and $\Omega = R_+^n$.

Problem 13 (see [31]). The mapping F is given by

$$F(x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} x_1^3 \\ x_2^3 \\ 2x_3^3 \\ 2x_4^3 \end{pmatrix} + \begin{pmatrix} -10 \\ 1 \\ -3 \\ 0 \end{pmatrix} \tag{59}$$

and $\Omega = \{x \in R^4 \mid \sum_{i=1}^4 x_i \le 4, x_i \ge 0, i = 1, 2, 3, 4\}$.

For Methods 1–6 and CGD_XZ method, we set $\sigma = 10^{-4}$, $t = 0.5$, and $\rho = s_k^T s_k/(s_k^T y_k)$, where $s_k = x_{k+1} - x_k$ and $\rho \ge 1/L$ which is obtained by the monotonicity and the L-Lipschitz continuity of F. The stopping criterion is $\|F_k\|_\infty \le 10^{-5}$.

Our computations were carried out using MATLAB R2011b on a desktop computer with an Intel(R) Xeon(R) 2.40 GHZ CPU, 6.00 GB of RAM, and Windows operating system. The numerical results were reported in Tables 1, 2, 3, and 4, where the initial points $x_1 = (10, 10, \ldots, 10)^T$, $x_2 = (1, 1, \ldots, 1)^T$, $x_3 = (1, 1/2, \ldots, 1/n)^T$, $x_4 = (0.1, 0.1, \ldots, 0.1)^T$, $x_5 = (1/n, 2/n, \ldots, 1)^T$, and $x_6 = (1 - 1/n, 1 - 2/n, \ldots, 0)^T$ and Dim, Iter, Nf, and CPU stand for the dimension of the problem, the number of iterations, the number of function evaluations, and the CPU time elapsed in seconds, respectively. Table 5 showed the number that each method solved the test problems with the least iterations, the least function evaluations, and the best time, respectively.

The performance of the seven methods was evaluated using the profiles of Dolan and Morè [32]. That is, we plotted the fraction P of the test problems for which each of the methods was within a factor τ of the best time. Figures 1–3 showed the performance profiles referring to the number of iterations, the number of function evaluations, and CPU time, respectively. Figure 1 indicated that relative to the number of iterations, Methods 1 and 2 performed best for τ near 1. When $\tau \ge 1.3$, CGD_XZ was comparable with Methods 1 and 2 and had a higher number of wins than Methods 3–6. Figure 2 revealed that relative to the number

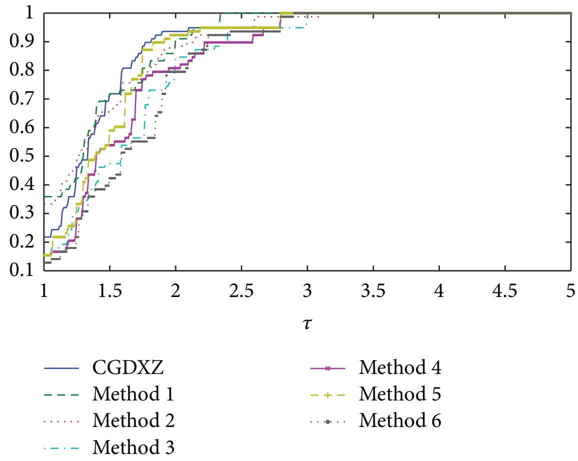

Figure 1: Performance profile based on the number of iterations.

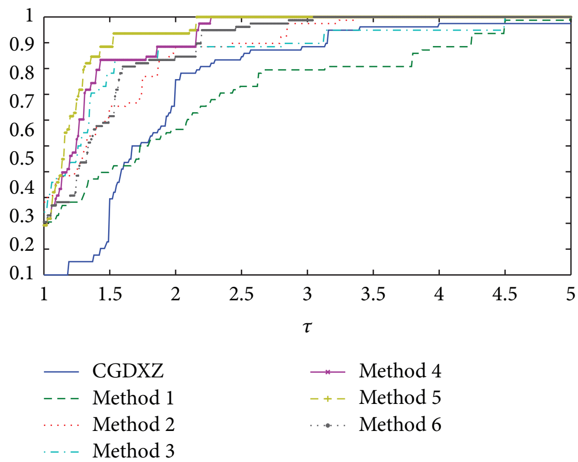

Figure 2: Performance profile based on the number of function evaluations.

of function evaluations, Method 2 performed best for τ near 1. When $\tau \ge 1.6$, Method 5 performed more robust and then Methods 3 and 4. Figure 3 revealed that relative to the CPU time metric, Method 2 performed best for τ near 1. Method 5 performed more robust when $\tau \ge 1.3$, and Methods 2–4 were competitive. While CGD_XZ performed worst, it had a lower number of wins than the rest of the methods. So, from the analysis above, we can conclude that all these methods were efficient to solve these test problems. If we consider the number of wins, Method 2 performed best which is also revealed by Table 5, while from the view of robustness, Method 5 performed best.

5. Conclusions

In this paper, we discussed two unified frameworks of some sufficient descent conjugate gradient methods and combined them with the hyperplane projection method of Solodov and Svaiter to solve convex constrained nonlinear monotone equations. The two unified frameworks inherit the advantages

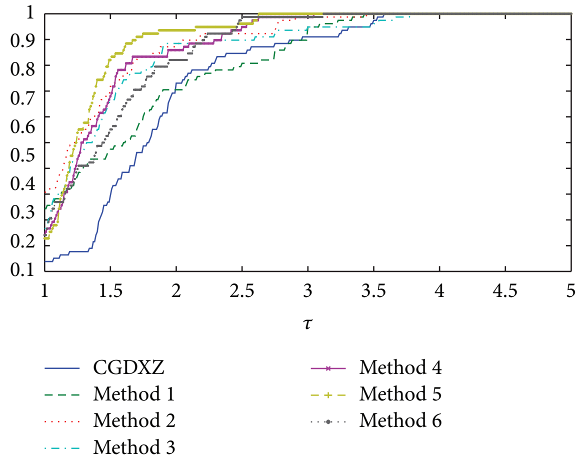

Figure 3: Performance profile based on the CPU time.

of some usual conjugate gradient methods for solving large-scale unconstrained minimization problems. That is, they satisfy the sufficient descent condition $F_k^T d_k \leq -c\|F_k\|^2$ ($c > 0$) independently of any line search, and they do not require F's Jacobian, then they are suitable to solve large-scale nonsmooth monotone equations. In Section 4, we gave several specific versions of the two unified frameworks and investigated their numerical behaviors over some test problems. From the numerical results, we concluded that these specific versions are efficient.

Let us review problem (1) and introduce a monotone inclusion problem

$$0 \in T(x), \tag{60}$$

where the set-value mapping $T : R^n \rightarrow 2^{R^n}$ is maximal monotone. Obviously, the latter is more general than the former; then, our further investigation is to extend these sufficient descent conjugate gradient methods to solve problem (60).

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

This work was supported by the National Science Foundation of China, no. 61373174.

References

[1] J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, NY, USA, 1970.

[2] M. E. El-Hawary, *Optimal Power Flow: Solution Techniques, Requirement and Challenges*, IEEE Service Center, Piscataway, NJ, USA, 1996.

[3] A. J. Wood and B. F. Wollenberg, *Power Generations, Operations, and Control*, John Wiley & Sons, New York, NY, USA, 1996.

[4] Y.-B. Zhao and D. Li, "Monotonicity of fixed point and normal mappings associated with variational inequality and its application," *SIAM Journal on Optimization*, vol. 11, no. 4, pp. 962–973, 2001.

[5] M. V. Solodov and B. F. Svaiter, "A globally convergent inexact Newton method for systems of monotone equations," in *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, M. Fukushima and L. Qi, Eds., vol. 22, pp. 355–369, Kluwer Academic Publishers, Dordrecht, The Netherlands, 1999.

[6] G. Zhou and K. C. Toh, "Superlinear convergence of a Newton-type algorithm for monotone equations," *Journal of Optimization Theory and Applications*, vol. 125, no. 1, pp. 205–221, 2005.

[7] W.-J. Zhou and D.-H. Li, "A globally convergent BFGS method for nonlinear monotone equations without any merit functions," *Mathematics of Computation*, vol. 77, no. 264, pp. 2231–2240, 2008.

[8] W. Zhou and D. Li, "Limited memory BFGS method for nonlinear monotone equations," *Journal of Computational Mathematics*, vol. 25, no. 1, pp. 89–96, 2007.

[9] Y. Xiao and H. Zhu, "A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing," *Journal of Mathematical Analysis and Applications*, vol. 405, no. 1, pp. 310–319, 2013.

[10] W. Cheng, "A PRP type method for systems of monotone equations," *Mathematical and Computer Modelling*, vol. 50, no. 1-2, pp. 15–20, 2009.

[11] Q.-R. Yan, X.-Z. Peng, and D.-H. Li, "A globally convergent derivative-free method for solving large-scale nonlinear monotone equations," *Journal of Computational and Applied Mathematics*, vol. 234, no. 3, pp. 649–657, 2010.

[12] Q. Li and D.-H. Li, "A class of derivative-free methods for large-scale nonlinear monotone equations," *IMA Journal of Numerical Analysis*, vol. 31, no. 4, pp. 1625–1635, 2011.

[13] M. Ahookhosh, K. Amini, and S. Bahrami, "Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations," *Numerical Algorithms*, vol. 64, no. 1, pp. 21–42, 2013.

[14] M. Li, "A LS type method for solving large-scale nonlinear monotone equations," *Numerical Functional Analysis and Optimization*, 2013.

[15] W. Cheng, Y. Xiao, and Q.-J. Hu, "A family of derivative-free conjugate gradient methods for large-scale nonlinear systems of equations," *Journal of Computational and Applied Mathematics*, vol. 224, no. 1, pp. 11–19, 2009.

[16] W. La Cruz and M. Raydan, "Nonmonotone spectral methods for large-scale nonlinear systems," *Optimization Methods & Software*, vol. 18, no. 5, pp. 583–599, 2003.

[17] L. Zhang and W. Zhou, "Spectral gradient projection method for solving nonlinear monotone equations," *Journal of Computational and Applied Mathematics*, vol. 196, no. 2, pp. 478–484, 2006.

[18] Z. Yu, J. Lin, J. Sun, Y. Xiao, L. Liu, and Z. Li, "Spectral gradient projection method for monotone nonlinear equations with convex constraints," *Applied Numerical Mathematics*, vol. 59, no. 10, pp. 2416–2423, 2009.

[19] Y. H. Dai, "Nonlinear conjugate gradient methods," in *Wiley Encyclopedia of Operations Research and Management Science*, J. J. Cochran, L. A. Cox Jr, P. Keskinocak, J. P. Kharoufeh, and J. C. Smith, Eds., John Wiley & Sons, 2011.

[20] W. W. Hager and H. Zhang, "A new conjugate gradient method with guaranteed descent and an efficient line search," *SIAM Journal on Optimization*, vol. 16, no. 1, pp. 170–192, 2005.

[21] W. W. Hager and H. Zhang, "Algorithm 851: *CGDESCENT*, a conjugate gradient method with guaranteed descent," *ACM Transactions on Mathematical Software*, vol. 32, no. 1, pp. 113–137, 2006.

[22] G. Yu, L. Guan, and W. Chen, "Spectral conjugate gradient methods with sufficient descent property for large-scale unconstrained optimization," *Optimization Methods & Software*, vol. 23, no. 2, pp. 275–293, 2008.

[23] W. Cheng and Q. Liu, "Sufficient descent nonlinear conjugate gradient methods with conjugacy condition," *Numerical Algorithms*, vol. 53, no. 1, pp. 113–131, 2010.

[24] E. Polak and G. Ribière, "Note sur la convergence de méthodes de directions conjuguées," *Revue Francaise Dinformatique et Derecherche Opérationnelle, Série Rouge*, vol. 3, no. 16, pp. 35–43, 1969.

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

[26] Y.-H. Dai and C.-X. Kou, "A nonlinear conjugate gradient algorithm with an optimal property and an improved Wolfe line search," *SIAM Journal on Optimization*, vol. 23, no. 1, pp. 296–320, 2013.

[27] W. W. Hager and H. Zhang, "The limited memory conjugate gradient method," *SIAM Journal on Optimization*, vol. 23, no. 4, pp. 2150–2168, 2013.

[28] M. R. Hestenes and E. Stiefel, "Methods of conjugate gradients for solving linear systems," *Journal of Research of the National Bureau of Standards*, vol. 49, no. 6, pp. 409–436, 1952.

[29] C. Wang, Y. Wang, and C. Xu, "A projection method for a system of nonlinear monotone equations with convex constraints," *Mathematical Methods of Operations Research*, vol. 66, no. 1, pp. 33–46, 2007.

[30] G. Yu, S. Niu, and J. Ma, "Multivariate spectral gradient projection method for nonlinear monotone equations with convex constraints," *Journal of Industrial and Management Optimization*, vol. 9, no. 1, pp. 117–129, 2013.

[31] N. Yamashita and M. Fukushima, "Modified Newton methods for solving a semismooth reformulation of monotone complementarity problems," *Mathematical Programming*, vol. 76, no. 3, pp. 469–491, 1997.

[32] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Mathematical Programming*, vol. 91, no. 2, pp. 201–213, 2002.