Comp. Appl. Math. (2017) 36:259–279
DOI 10.1007/s40314-015-0228-1

Supermemory gradient methods for monotone nonlinear equations with convex constraints

Yigui Ou[1] · **Yuanwen Liu**[1]

Received: 2 February 2014 / Revised: 6 March 2015 / Accepted: 2 April 2015 /
Published online: 11 April 2015

Abstract This paper presents two new supermemory gradient algorithms for solving convex-constrained nonlinear monotone equations, which combine the idea of supermemory gradient method with the projection method. The feature of these proposed methods is that at each iteration, they do not require the Jacobian information and solve any subproblem, even if they do not store any matrix. Thus, they are suitable for solving large-scale equations. Under mild conditions, the proposed methods are shown to be globally convergent. Preliminary numerical results show that the proposed methods are efficient and can be applied to solve large-scale nonsmooth equations.

Keywords Monotone equations · Supermemory gradient method · Projection method · Global convergence

Mathematics Subject Classification 90C30 · 65K05 · 65H10

1 Introduction

In this paper, we consider the following problem: finding a vector $x \in R^n$ such that

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

where $F : R^n \to R^n$ is continuous and monotone mapping, i.e.,

$$(F(x) - F(y))^T (x - y) \geq 0, \quad \forall x, y \in R^n, \tag{1.2}$$

Communicated by Ernesto G. Birgin.

Supported by NNSF of China (No. 11261015) and NSF of Hainan Province (No. 111001).

✉ Yigui Ou
ouyigui@126.com

[1] Department of Applied Mathematics, Hainan University, Haikou 570228, China

$X \subset R^n$ is a nonempty closed convex set, and sometimes an n-dimensional box, i.e., $X = \{x \in R^n : l \leq x \leq u\}$. Throughout the paper, $\|\cdot\|$ denotes the Euclidean norm on R^n, and F_k denotes $F(x_k)$.

Systems of monotone equations commonly arise in many applications, for instance, they are used as subproblems of the generalized proximal algorithms with Bregman distance (Iusem and Solodov 1997). Some monotone variational inequality problems can be converted into systems of nonlinear monotone equations (Zhao and Li 2001). Moreover, the equations with convex constraints come from the problems such as the power flow equations and the l_1-norm regularized optimization problems in compressive sensing, see Refs. Wood and Wollenberg (1996) and Figueiredo (2007) for instance.

In recent years, this special class of optimization problem (1.1) has been studied extensively and there are already many numerical methods for solving it, such as Newton-based methods, Levenberg–Marquardt method, trust region method and projection method, see Refs. Tong and Zhou (2005), Fan (2013), Jia and Zhu (2011) and Wang (2007) for instance. However, these methods need to solve a linear system of equations or a trust region subproblem at each iteration using the Jacobian matrix or an approximation of it, thus they are only suitable for small-scale problems. More recently, large-scale nonlinear constrained equations have been attracting more and more attention from researchers, and some efficient methods have been proposed, which mainly include the spectral gradient method and conjugate gradient method, see Refs. Yu et al. (2009) and Xiao and Zhu (2013) for instance. The main feature of these methods is that at each iteration, the search direction can be obtained without using the Jacobian information and storing any matrix. The drawback of these methods is that they use the previous one-step iterative information at best to generate the search direction at each iteration. It is worth mentioning that the derivative-free methods for problem (1.1) with $X = R^n$ also belong to the conjugate gradient scheme, see Refs. Yan et al. (2010), Li and Li (2011) and Ahookhosh et al. (2013) for instance.

Similar methods to the conjugate gradient methods for unconstrained optimization [see Ref. Hager and Zhang 2006 for details] are the memory gradient (MG) method and the supermemory gradient (SMG) method (Shi and Shen 2005). Not only is their basic idea simple but also they avoid the computation and the storage of some matrices so that they are also suitable to solve large-scale optimization problems. Compared with CG methods, however, the main difference is that MG and SMG methods can sufficiently use the previous multi-step iterative information to generate a new iterate at each iteration, and hence it is more helpful to design algorithms with fast convergence rate, see Refs. Shi and Shen (2005), Narushima and Yabe (2006), Ou and Wang (2012), Ou and Liu (2014) and Sun and Ba (2011) for instance.

Motivated by the supermemory gradient method (Ou and Liu 2014) and the projection strategy (Wang 2007), we propose two new supermemory gradient methods for solving the nonlinear monotone equations (1.1). The main properties of the proposed methods are that we establish the global convergence without using any merit function and making the differentiability assumption on F. Furthermore, these methods do not solve any subproblem and store any matrix. Thus, they can be applied to solve large-scale nonlinear equations. Preliminary numerical results are reported to show that the proposed methods are efficient and stable.

The rest of the paper is organized as follows. In Sect. 2, the detailed algorithm is outlined and then some of its properties are given. Section 3 is devoted to analyze the global convergence property of our algorithm under some mild conditions. In Sect. 4, further discussion on the choice of search direction is given. In Sect. 5, numerical results and comparison are

reported to show the efficiency of the proposed methods. Some conclusions are summarized in the final section.

2 Supermemory gradient algorithm

This section is devoted to constructing a new supermemory gradient method for solving problem (1.1), then giving some properties of the proposed method.

We first recall the iterative scheme of supermemory gradient method (Ou and Liu 2014) for solving the unconstrained optimization problem:

$$\min_{x\in R^n} f(x), \tag{2.1}$$

where $f : R^n \to R$ is a continuously differentiable function whose gradient at x_k is denoted by $g_k := \nabla f(x_k)$. Given any starting point $x_0 \in R^n$, the algorithm in Ou and Liu (2014) is to generate a sequence $\{x_k\}$ which satisfies the following recursive form

$$x_{k+1} = x_k + \alpha_k d_k, \quad k = 0, 1, 2, \ldots, \tag{2.2}$$

where α_k is a stepsize, and d_k is a descent direction of $f(x)$ at x_k defined by

$$d_k = \begin{cases} -g_k, & k \le m, \\ -g_k + \sum_{i=1}^{m} \beta_k^i d_{k-i}, & k \ge m+1, \end{cases} \tag{2.3}$$

where m denotes the number of the past iterations remembered, and

$$\beta_k^i = \frac{\rho \|g_k\|}{\|d_{k-i}\|} \tag{2.4}$$

with $\rho \in (0, \frac{1}{m})$. For details, see Ref. Ou and Liu (2014).

Based on the above scheme (2.3)–(2.4), here we define d_k as

$$d_k = \begin{cases} -F_k, & k \le m, \\ -F_k + \sum_{i=1}^{m} \beta_k^i d_{k-i}, & k \ge m+1, \end{cases} \tag{2.5}$$

where

$$\beta_k^i = \frac{\rho \|F_k\|}{\sum_{i=1}^{m} \|d_{k-i}\|} \tag{2.6}$$

with $\rho \in (0, \frac{1}{2})$.

To describe our algorithm, we introduce the definition of projection operator $P_\Omega[\cdot]$ which is defined as a mapping from R^n to a nonempty closed convex subset Ω:

$$P_\Omega[x] = \arg\min_{y\in\Omega}\{\|y - x\|\}, \quad \forall x \in R^n. \tag{2.7}$$

A well-known property of this operator is that it is nonexpansive [see Wang (2007)], namely,

$$\|P_\Omega[x] - P_\Omega[y]\| \le \|x - y\|, \quad \forall x, y \in R^n. \tag{2.8}$$

We now formally state the steps of the supermemory gradient algorithm for solving problem (1.1) as follows.

Algorithm 2.1

Step 0 Given an initial point $x_0 \in X$, $\epsilon \geq 0$, $\sigma \in (0,1)$, $\rho \in (0, \frac{1}{2})$, an integer $m > 0$, and $\beta \in (0,1)$. Set $k := 0$.

Step 1 Compute F_k. If $\|F_k\| \leq \epsilon$, then stop.

Step 2 Construct a memory search direction d_k using (2.5)–(2.6).

Step 3 Compute the trial point $z_k = x_k + \alpha_k d_k$, where $\alpha_k = \beta^{l_k}$ with l_k being the smallest nonnegative integer l such that

$$-F(x_k + \beta^l d_k)^T d_k \geq \sigma \beta^l \|F(x_k + \beta^l d_k)\| \|d_k\|^2. \tag{2.9}$$

Step 4 Compute the new iterate x_{k+1} by

$$x_{k+1} = P_X \left[x_k - \frac{(x_k - z_k)^T F(z_k)}{\|F(z_k)\|^2} F(z_k) \right]. \tag{2.10}$$

Step 5 Set $k := k + 1$, and go to Step 1.

Remark 2.1 The line search scheme (2.9) is similar to that in Refs. Yan et al. (2010), Li and Li (2011) and Ahookhosh et al. (2013). Because it does not include any matrix and gradient information, it can be used to devise efficient methods for solving large-scale nonlinear equations. From Lemma 3.1 below, it follows that the line search scheme (2.9) is well defined, namely, it terminates in a finite number of steps.

It should be pointed out that there are two distinct differences between the proposed algorithm and the algorithms in Refs. Yu et al. (2009) and Xiao and Zhu (2013), that is, the construction of search direction d_k and the line search scheme, see Refs. Yu et al. (2009) and Xiao and Zhu (2013) for details.

Remark 2.2 Step 4 has a nice geometric interpretation as follows: suppose x_k is not a solution to problem (1.1). Then, using the line search scheme (2.9), the trial point z_k is obtained to satisfy

$$F(z_k)^T (z_k - x_k) < 0. \tag{2.11}$$

On the other hand, using the monotonicity property of $F(x)$, for any $\bar{x}$ such that $F(\bar{x}) = 0$, we conclude that

$$F(z_k)^T (z_k - \bar{x}) \geq 0, \tag{2.12}$$

which, together with (2.11), implies that the hyperplane

$$H_k = \{x \in R^n | F(z_k)^T (x - z_k) = 0\} \tag{2.13}$$

strictly separates the current iterate x_k from the solution set of problem (1.1). Based on this fact, we can see that the next iterate x_{k+1} is computed by projecting x_k onto the intersection of the feasible set X with the halfspace $H_k^- = \{x \in R^n | F(z_k)^T (x - z_k) \leq 0\}$.

Obviously, the search direction d_k defined by (2.5) and (2.6) satisfies the following properties:

$$F_k^T d_k \leq -(1 - \rho)\|F_k\|^2, \quad \forall k, \tag{2.14}$$

and

$$\|d_k\| \leq (1 + \rho)\|F_k\|, \quad \forall k. \tag{2.15}$$

3 Convergence analysis

In this section, we analyze the global convergence property of Algorithm 2.1 when it is applied to problem (1.1). For this purpose, we first make the following assumptions.

A1 The solution set X^* of problem (1.1) is nonempty.
A2 The mapping F is Lipschitz continuous on the nonempty closed convex set X, namely, there exists a constant $L > 0$ such that

$$\|F(x) - F(y)\| \leq L\|x - y\|, \quad \forall x, y \in X. \tag{3.1}$$

In what follows we assume that $F_k \neq 0$ for all k, namely, Algorithm 2.1 generates an infinite sequence $\{x_k\}$. Otherwise, we obtain a solution of problem (1.1).

The following lemma shows that Algorithm 2.1 is well defined.

Lemma 3.1 *Algorithm 2.1 is well defined, namely, there exists a nonnegative number l_k satisfying the line search scheme (2.9) for all k.*

Proof By contradiction, suppose that there exists $k_0 \geq 0$ such that (2.9) fails to hold for any nonnegative number l, namely

$$-F(x_{k_0} + \beta^l d_{k_0})^T d_{k_0} < \sigma\beta^l \|F(x_{k_0} + \beta^l d_{k_0})\| \|d_{k_0}\|^2, \quad \forall l \geq 0. \tag{3.2}$$

Taking $l \to +\infty$ on both sides of (3.2) and using the continuity of F, we have

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

which contracts this fact that $-F_k^T d_k \geq (1 - \rho)\|F_k\|^2 > 0$ for all k.

Lemma 3.2 *Let $x^* \in X^*$ and the sequences $\{x_k\}$ and $\{z_k\}$ be generated by Algorithm 2.1. Then we have*

$$\|x_{k+1} - x^*\|^2 \leq \|x_k - x^*\|^2 - \sigma^2\|x_k - z_k\|^4, \tag{3.3}$$

and

$$\lim_{k \to +\infty} \|x_k - z_k\| = \lim_{k \to +\infty} \alpha_k \|d_k\| = 0. \tag{3.4}$$

Proof From the line search scheme (2.9), it follows that

$$F(z_k)^T (x_k - z_k) = -\alpha_k F(z_k)^T d_k \geq \sigma\alpha_k^2 \|F(z_k)\| \|d_k\|^2, \tag{3.5}$$

which implies that

$$\frac{F(z_k)^T (x_k - z_k)}{\|F(z_k)\|} \geq \sigma\alpha_k^2 \|d_k\|^2 = \sigma\|x_k - z_k\|^2. \tag{3.6}$$

Using the monotonicity of mapping F and $F(x^*) = 0$, we have

$$F(z_k)^T (z_k - x^*) \geq F(x^*)^T (z_k - x^*) = 0. \tag{3.7}$$

This together with (3.5) implies that

$$F(z_k)^T (x_k - x^*) \geq F(z_k)^T (x_k - z_k) \geq 0. \tag{3.8}$$

Combining this inequality with (2.8) and (3.6) gives

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &= \|P_X[x_k-\frac{(x_k-z_k)^TF(z_k)}{\|F(z_k)\|^2}F(z_k)]-P_X[x^*]\|^2\\ &\le \|x_k-\frac{(x_k-z_k)^TF(z_k)}{\|F(z_k)\|^2}F(z_k)-x^*\|^2\\ &\le \|x_k-x^*\|^2-\frac{[F(z_k)^T(x_k-z_k)]^2}{\|F(z_k)\|^2}\\ &\le \|x_k-x^*\|^2-\sigma^2\|x_k-z_k\|^4. \end{aligned} \tag{3.9}$$

Thus, this inequality (3.3) holds true. Furthermore, it follows directly from (3.3) that the sequence $\{\|x_k-x^*\|\}$ is decreasing monotonically and convergent. Taking $k\to+\infty$ on both sides of (3.3) yields (3.4). This completes the proof. ‖

Corollary 3.3 *Let the sequence $\{x_k\}$ be generated by Algorithm 2.1. If Assumptions* A1 *and* A2 *hold, then there exists a constant $M>0$ such that*

$$\|F_k\|\le M,\quad \forall k, \tag{3.10}$$

and

$$\|F(x_k+\alpha_k\beta^{-1}d_k)\|\le M,\quad \forall k. \tag{3.11}$$

Proof Let $x^*\in X^*$. It follows from A2 and (3.3) that

$$\|F_k\|=\|F_k-F(x^*)\|\le L\|x_k-x^*\|\le L\|x_{k-1}-x^*\|\le\cdots\le L\|x_0-x^*\|. \tag{3.12}$$

By (3.4), we can deduce that there exists a constant $M_0>0$ such that

$$\alpha_k\|d_k\|\le M_0,\quad \forall k. \tag{3.13}$$

Combining (3.13) with (3.3) gives

$$\begin{aligned}\|F(x_k+\alpha_k\beta^{-1}d_k)\| &= \|F(x_k+\alpha_k\beta^{-1}d_k)-F(x^*)\|\\ &\le L\|x_k+\alpha_k\beta^{-1}d_k-x^*\|\\ &\le L\|x_k-x^*\|+L\beta^{-1}\alpha_k\|d_k\|\\ &\le L\|x_0-x^*\|+L\beta^{-1}M_0,\ \forall k. \end{aligned} \tag{3.14}$$

Let $M=L\|x_0-x^*\|+L\beta^{-1}M_0$. Then the conclusions (3.10) and (3.11) follow directly from the above inequalities (3.12) and (3.14). This completes the proof. ‖

Lemma 3.4 *Suppose that Assumptions* A1 *and* A2 *hold. Then there exists a constant $\gamma>0$ such that the stepsize α_k involved in Step 3 of Algorithm 2.1 satisfies*

$$\alpha_k\ge\min\left\{1,\ \gamma\frac{\|F_k\|^2}{\|d_k\|^2}\right\}. \tag{3.15}$$

Proof If $\alpha_k\ne 1$, then it follows from the acceptance rule of stepsize α_k in Step 3 of Algorithm 2.1 that $\alpha_k'=\frac{\alpha_k}{\beta}$ does not satisfy (2.9), namely

$$-F(x_k+\alpha_k'd_k)^Td_k<\sigma\alpha_k'\|F(x_k+\alpha_k'd_k)\|\|d_k\|^2. \tag{3.16}$$

This together with (2.14) and (3.11) implies that

$$\begin{aligned}(1-\rho)\|F_k\|^2 &\le -F_k^T d_k \\ &= [F(x_k+\alpha_k' d_k)-F_k]^T d_k - F(x_k+\alpha_k' d_k)^T d_k \\ &\le \alpha_k' L\|d_k\|^2 + \sigma\alpha_k'\|F(x_k+\alpha_k' d_k)\|\|d_k\|^2 \\ &\le \alpha_k'(L+\sigma M)\|d_k\|^2. \end{aligned} \tag{3.17}$$

Obviously, if $\|d_k\| \neq 0$, then (3.15) follows directly from (3.17), where $\gamma = \frac{\beta(1-\rho)}{L+\sigma M}$.

In what follows, we verify the validity of the assertion $\|d_k\| \neq 0$ for all k. In fact, it follows from (2.14) that

$$\begin{aligned}\|d_k\|^2 &= \|d_k+F_k\|^2 + \|F_k\|^2 - 2F_k^T(F_k+d_k) \\ &\ge \|F_k\|^2 - 2F_k^T(F_k+d_k) \\ &\ge 2(1-\rho)\|F_k\|^2 - \|F_k\|^2 \\ &= (1-2\rho)\|F_k\|^2, \quad \forall k, \end{aligned} \tag{3.18}$$

which implies that $\|d_k\| \neq 0$ for all k, due to $\rho \in (0, \frac{1}{2})$ and $|F_k\| \neq 0$ for all k. This completes the proof. ‖

Using these lemmas mentioned above, we obtain the global convergence of Algorithm 2.1 as follows.

Theorem 3.5 *Suppose that Assumptions* A1 *and* A2 *hold. Then we have*

$$\lim_{k\to+\infty} \inf \|F_k\| = 0. \tag{3.19}$$

Furthermore, the sequence $\{x_k\}$ *generated by Algorithm* 2.1 *converges to a solution of* (1.1).

Proof Suppose on the contrary that there exists $\delta > 0$ such that

$$\|F_k\| \ge \delta, \quad \forall k. \tag{3.20}$$

From Corollary 3.3 and (2.15), it follows that there exists a constant $M > 0$ such that

$$\|d_k\| \le (1+\rho)\|F_k\| \le (1+\rho)M. \tag{3.21}$$

On the other hand, using (3.18) and (3.20), we have

$$\|d_k\| \ge \sqrt{1-2\rho}\,\|F_k\| \ge \sqrt{1-2\rho}\,\delta. \tag{3.22}$$

Combining (3.15) with (3.20)–(3.22) gives

$$\alpha_k\|d_k\| \ge \min\left\{1,\ \gamma\frac{\|F_k\|^2}{\|d_k\|^2}\right\}\|d_k\| \ge \min\left\{\sqrt{1-2\rho}\,\delta,\ \frac{\gamma\delta^2}{(1+\rho)M}\right\}.$$

This yields a contradiction with (3.4). Thus, the assertion (3.19) holds true.

Now, we will prove the second assertion. By the continuity of F and the boundedness of $\{x_k\}$ due to (3.3), it is clear that the sequence $\{x_k\}$ exists an accumulation point $\bar{x}$ such that $F(\bar{x}) = 0$ due to (3.19). Taking $x^* = \bar{x}$ in (3.3), we further deduce that the sequence $\{\|x_k - \bar{x}\|\}$ is convergent, and thus the sequence $\{x_k\}$ converges globally to $\bar{x}$. The proof is completed. ‖

At the end of this section, we must point out that the requirement (1.2) of monotonicity on F seems to be strong for the purpose of ensuring the global convergence property. Actually, if the mapping F has the following property:

$$F(x)^T(x - x^*) \geq 0, \quad \forall x \in R^n, \quad x^* \in X^*, \tag{3.23}$$

then it can be easily verified that all the preceding conclusions still hold true, provided that a little modification is made in the proof process of Lemma 3.2. Note that the property (3.23) is satisfied if F is monotone or pseudomonotone, but not vice versa, see Ref. Solodov and Svaiter (1999) for instance. Therefore, our convergence results can be restated under the assumption (3.23), which is considerably weaker than the assumption (1.2) of monotonicity on F. We omit them here.

4 Further discussion

From Step 2 of Algorithm 2.1, we can see that the search direction d_k plays a key role in designing a method for solving problem (1.1). In fact, there are some other possible choices of d_k. For example, we may choose

$$d_k = \begin{cases} -F_k, & \text{if } k \leq m, \\ -\lambda_k^{(k)} F_k - \sum_{i=1}^m \lambda_{k-i}^{(k)} d_{k-i}, & \text{if } k \geq m+1 \end{cases} \tag{4.1}$$

with

$$\lambda_k^{(k)} = 1 - \sum_{i=1}^m |\lambda_{k-i}^{(k)}|, \ \lambda_{k-i}^{(k)} = \begin{cases} \frac{\rho}{m} \cdot \frac{\|F_k\|^2}{\|F_k\|^2 + |F_k^T d_{k-i}|}, & \text{if } F_k^T d_{k-i} \geq 0, \\ -\frac{\rho}{m} \cdot \frac{\|F_k\|^2}{\|F_k\|^2 + |F_k^T d_{k-i}|}, & \text{if } F_k^T d_{k-i} < 0, \end{cases}$$

$(i = 1, 2, \ldots, m)$ and $0 < \rho < 1$.

Remark 4.1 Based on the direction d_k defined as (4.1), we can construct a new algorithm denoted by *Algorithm 4.1*, which is similar to Algorithm 2.1 except that the direction d_k in Step 2 is replaced by (4.1), we omit it here.

In what follows, we discuss the global convergence of Algorithm 4.1. To this end, we give two properties as follows.

Lemma 4.1 *For any $k \geq 0$, we have*

$$F_k^T d_k \leq -(1 - \rho)\|F_k\|^2. \tag{4.2}$$

Proof If $k \leq m$, then (4.2) is obvious. If $k \geq m + 1$, then we have

$$|\lambda_{k-i}^{(k)}| = \frac{\rho}{m} \cdot \frac{\|F_k\|^2}{\|F_k\|^2 + |F_k^T d_{k-i}|},$$
$$|\lambda_{k-i}^{(k)} F_k^T d_{k-i}| = \lambda_{k-i}^{(k)} F_k^T d_{k-i}, \quad i = 1, 2, \ldots, m,$$

which, together with (4.1), implies that

$$\begin{aligned}
-F_k^T d_k &= \lambda_k^{(k)} \|F_k\|^2 + \sum_{i=1}^{m} \lambda_{k-i}^{(k)} F_k^T d_{k-i} \\
&= \left(1 - \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}|\right) \|F_k\|^2 + \sum_{i=1}^{m} \lambda_{k-i}^{(k)} F_k^T d_{k-i} \\
&= \|F_k\|^2 - \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}| (\|F_k\|^2 - |F_k^T d_{k-i}|) \\
&\geq \|F_k\|^2 - \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}| (\|F_k\|^2 + |F_k^T d_{k-i}|) \\
&= (1-\rho)\|F_k\|^2.
\end{aligned}$$

This completes this proof. ‖

Lemma 4.2 1. *For any $k \geq m$, we have*

$$\|d_k\| \leq \max_{1 \leq i \leq m} \{\|F_k\|, \ \|d_{k-i}\|\}. \tag{4.3}$$

2. *For any $k \geq 0$, we have*

$$\|d_k\| \leq \max_{0 \leq i \leq k} \|F_i\|. \tag{4.4}$$

Proof If $k \leq m$, then (4.3) is obvious. We now show that (4.3) holds for $k \geq m+1$. Clearly, $\lambda_k^{(k)} \leq 1$ and

$$\begin{aligned}
\lambda_k^{(k)} &= 1 - \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}| \\
&= 1 - \frac{\rho}{m} \sum_{i=1}^{m} \frac{\|F_k\|^2}{\|F_k\|^2 + |F_k^T d_{k-i}|} \\
&\geq 1 - \frac{\rho}{m} \cdot m \\
&= 1 - \rho > 0
\end{aligned}$$

This implies that

$$\sum_{i=0}^{m} |\lambda_{k-i}^{(k)}| = 1. \tag{4.5}$$

Hence, it follows from (4.1) and (4.5) that

$$\begin{aligned}
\|d_k\| &\leq \left(1 - \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}|\right) \|F_k\| + \sum_{i=1}^{m} |\lambda_{k-i}^{(k)}| \cdot \|d_{k-i}\| \\
&\leq \max_{1 \leq i \leq m} \{\|F_k\|, \ \|d_{k-i}\|\},
\end{aligned}$$

which implies that this inequality (4.3) holds true.

Using induction process, we can deduce directly from (4.3) and (4.1) that this inequality (4.4) also holds true. This completes this proof. ‖

Using Lemmas 4.1 and 4.2, we can easily establish the global convergence of Algorithm 4.1, whose proof is the same as that of Theorem 3.5 and thus is omitted.

Theorem 4.3 *Suppose that Assumptions* A1 *and* A2 *hold. Then we have*

$$\lim_{k\to+\infty} \inf \|F_k\| = 0.$$

Furthermore, the sequence $\{x_k\}$ *generated by Algorithm 4.1 converges to a solution of* (1.1).

5 Numerical tests

In this section, we present some numerical experiments to evaluate the performance of the proposed methods on two sets of test problems. At the same time, we give some comparisons with the related algorithms, including the performance profiles of Dolan and More (2002).

We first present some numerical experiments for Algorithms 2.1 and 4.1 on three constrained monotone testing problems, which are chosen from Refs. Yu et al. (2009) and Yan et al. (2010).

Problem 1 The elements of function F are given by

$$F_i(x) = \exp(x_i) - 1, \quad i = 1, 2, \ldots, n, \quad \text{and} \quad X = \{x \in R^n | x_i \geq 0, \quad i = 1, 2, \ldots, n\}.$$

Problem 2 The elements of function F are given by

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

Obviously, this problem is nonsmooth at point $(1, 1, \ldots, 1)^T \in R^n$.

Problem 3 The elements of function F are given by

$$F_1(x) = x_1 - \exp\left(\cos\left(\frac{x_1 + x_2}{n+1}\right)\right),$$
$$F_i(x) = x_i - \exp\left(\cos\left(\frac{x_{i-1} + x_i + x_{i+1}}{n+1}\right)\right), \quad i = 2, 3, \ldots, n-1,$$
$$F_n(x) = x_n - \exp\left(\cos\left(\frac{x_{n-1} + x_n}{n+1}\right)\right),$$

and $X = \{x \in R^n | x_i \geq 0, \ i = 1, 2, \ldots, n\}$.

To validate Algorithms 2.1 and 4.1 from a computational point of view, we compare them with the algorithm in Ref. Yu et al. (2009) (abbreviated *Algorithm Yu*) and the algorithm in Ref. Xiao and Zhu (2013) (abbreviated *Algorithm Xiao*) for the same problems mentioned above, where the latter two algorithms were also devised especially for the monotone case.

We implemented all the algorithms with the codes written in Matlab 7.12. The testing is performed on a PC computer with HPdx2810SE Pentium(R) Dual-Core CPU E5300 @ 2.60 GHZ 2.00 GB. Throughout the computational experiments, the parameters used in Algorithms 2.1 and 4.1 are chosen as follows: $\rho = 0.0001$, $\sigma = 0.0001$, $\beta = 0.9$, $m = 5$; the parameters used in Algorithm Yu are chosen as follows: $\beta = 0.5$, $\sigma = 0.01$, $r = 0.01$,

while the parameters used in Algorithm Xiao are chosen as follows: $\xi = 1,\ \rho = 0.5,\ \sigma = 0.0001$.

All testing examples start at six initial points listed as follows:

$$X1 = (10, 10, \ldots, 10)^T, \quad X2 = (1, 1, \ldots, 1)^T, \quad X3 = \left(1, \frac{1}{2}, \ldots, \frac{1}{n}\right),$$

$$X4 = (0.1, 0.1, \ldots, 0.1)^T, \quad X5 = \left(\frac{1}{n}, \frac{2}{n}, \ldots, 1\right)^T, \quad X6 = \left(1 - \frac{1}{n}, 1 - \frac{2}{n}, \ldots, 0\right)^T,$$

and the stopping condition for each algorithm is

$$\|F_k\| \leq 10^{-5}. \tag{5.1}$$

The numerical results are reported in Tables 1, 2 and 3, where the number n refers to the dimension of variables. The numerical results are given in the form of $It/NF/T/FN$, where It, NF, T and FN denote the number of iterations, the number of function evaluations, the CPU time in seconds and the final norm of F, respectively. If a method does not find a solution, but terminates by exceeding 300 s, or presets iteration limit $k = 5000$, we denote it by the word **F**.

We use the performance profile proposed by Dolan and More (2002) to display the performance of each implementation on the set of test problems. That is, for each method, we plot the fraction P of problems for which the method is within a factor τ of the smallest number of iterations/function evaluations/CPU times. Clearly, the left side of the figure gives the percentage of the test problems for which a method is the best one according to the number of iterations/function evaluations/CPU times, while the right side of the figure gives the percentage of the test problems that is successfully solved by these algorithms. For more details about the performance profile, please see Ref. Dolan and More (2002). Based on the testing results in Tables 1, 2 and 3, we give the performance profiles in Figs. 1, 2 and 3.

From Figs. 1, 2 and 3, we observe the following facts:

- In terms of the number of iterations, Algorithm 4.1 performs best, and Algorithm 2.1 performs roughly the same as Algorithm Yu, while Algorithm Xiao performs worst.
- In terms of the number of function evaluations, Algorithm 4.1 performs roughly the same as Algorithm Yu and Algorithm 2.1, but does better than Algorithm Xiao.
- In terms of the CPU time, Algorithm 4.1 performs roughly the same as Algorithm Yu, but both of them do better than Algorithm 2.1, while Algorithm Xiao performs worst.
- For the test problems, Algorithms 2.1 and 4.1 are competitive with the other two methods, in terms of the accuracy of approximate solutions obtained.

Therefore, we could say that Algorithms 2.1 and 4.1 are competitive to Algorithms Yu and Xiao, in terms of the computational effort and accuracy.

To examine Algorithm 2.1's sensitivity to the choices of the parameters m and ρ, we also do some preliminary numerical experiments with varying values of them, and then give the performance profiles in Figs. 4 and 5. From Figs. 4 and 5, we can observe the following facts:

- The choice of ρ has little impact on Algorithm 2.1 if $\gamma \in (0, 0.001]$ and the other parameters keep the same as the initial values, i.e., $\sigma = 0.0001$, $\beta = 0.9$, $m = 5$.
- The choice of m has little impact on Algorithm 2.1 if $3 \leq m \leq 6$ and the other parameters keep the same as the initial values, i.e., $\rho = 0.0001$, $\sigma = 0.0001$, $\beta = 0.9$.

Similarly, it has been observed that Algorithm 4.1 is also not very sensitive to the choices of the parameters m and ρ.

Table 1 Numerical results for Problem 1

Initial	n	Algorithm 2.1	Algorithm 4.1	Algorithm Yu	Algorithm Xiao
$X1$	1000	13/64/0.050738/2.9713e−6	10/54/0.050551/6.0304e−9	14/42/0.056237/2.2807e−6	15/101/0.078603/1.7715e−6
$X2$	1000	8/27/0.026880/1.5844e−6	7/23/0.024155/2.7934e−9	6/15/0.029737/6.5113e−7	9/51/0.043292/2.0964e−6
$X3$	1000	8/34/0.046568/1.9600e−6	8/33/0.037803/2.9244e−6	7/18/0.047629/1.6562e−7	19/94/0.091869/3.9727e−6
$X4$	1000	7/19/0.021421/1.6382e−6	7/18/0.016507/3.2484e−10	5/13/0.024389/3.1762e−7	10/52/0.046119/1.9078e−6
$X5$	1000	21/66/0.055272/8.4352e−6	24/62/0.063997/9.0752e−6	8/19/0.022347/1.0630e−6	21/106/0.091342/9.4783e−6
$X6$	1000	21/61/0.046837/8.4244e−6	24/62/0.057929/8.8455e−6	8/19/0.026587/9.4700e−7	20/95/0.087884/1.709e−6
$X1$	5000	13/64/0.166468/6.6441e−6	10/54/0.166082/1.3484e−8	14/42/0.170014/5.0997e−6	16/106/0.302948/1.0300e−6
$X2$	5000	8/27/0.107768/3.5428e−6	7/23/0.105594/6.2463e−9	6/15/0.081084/1.4560e−6	10/56/0.188734/1.0492e−6
$X3$	5000	8/34/0.094113/1.9545e−6	8/33/0.097150/2.9066e−6	7/18/0.081660/1.6563e−7	20/100/0.306520/1.5098e−6
$X4$	5000	7/19/0.065142/3.6630e−6	7/18/0.065275/7.2636e−10	5/13/0.060568 /7.1022e−7	11/56/0.183371/1.5357e−6
$X5$	5000	22/64/0.206864/9.4213e−6	26/66/0.229623/7.8181e−6	8/19/0.093473/4.9114e−6	21/103/0.314461/6.3115e−6
$X6$	5000	22/64/0.224293/9.4189e−6	22/66/0.230660/5.7650e−10	8/19/0.103901/4.7717e−6	19/92/0.295472/5.3895e−6
$X1$	10,000	13/64/0.299053/9.3962e−6	10/54/0.282474/1.9070e−8	14/42/0.318381/7.2121e−6	16/106/0.567395/1.4566e−6
$X2$	10,000	8/27/0.139289/5.0103e−6	7/23/0.135916/8.8336e−9	6/15/0.139737/2.0591e−6	10/56/0.322363/1.4837e−6
$X3$	10,000	8/34/0.181916/1.9538e−6	8/33/0.178321/2.9044e−6	7/18/0.140618/1.6564e−7	19/94/0.557350/3.5643e−6
$X4$	10,000	7/19/0.130173/5.1803e−6	7/18/0.120448/1.0272e−9	5/13/0.121609/1.0044e−6	11/56/0.337332/2.1718e−6
$X5$	10,000	23/67/0.419247/6.6585e−6	23/69/0.445720/7.4902e−6	8/19/0.156011/6.9458e−6	26/126/0.726833/1.1940e−6
$X6$	10,000	23/67/0.410975/6.6573e−6	23/69/0.420079/7.2241e−10	8/19/0.160018/7.4815e−006	19/91/0.535375/1.0540e−6

Table 2 Numerical results for Problem 2

Initial	n	Algorithm 2.1	Algorithm 4.1	Algorithm Yu	Algorithm Xiao
$X1$	1000	6/33/0.735725/7.8181e−6	6/33/0.463759/7.8181e−6	8/21/0.439710/5.2840e−8	14/81/1.112153/4.4756e−6
$X2$	1000	4/26/0.301994/8.0832e−6	4/26/0.314738/8.0832e−6	5/12/0.259285/5.9522e−9	8/56/0.738088/2.4146e−6
$X3$	1000	10/74/0.838552/3.5966e−6	9/66/0.799450/2.0147e−6	23/70/1.282128/9.7818e−7	18/102/1.423449/1.8274e−6
$X4$	1000	5/33/0.378291/3.2406e−7	5/33/0.386343/3.2406e−7	23/70/1.250039/6.9259e−7	14/82/1.120463/1.3432e−6
$X5$	1000	12/92/1.031523/1.4215e−6	11/84/0.939628/6.2907e−6	7/17/0.359232/2.3953e−7	27/159/2.160560/1.1087e−6
$X6$	1000	11/84/0.966821/1.4286e−6	11/84/0.948650/1.4484e−7	7/17/0.353798/2.3617e−7	27/156/2.124915/3.3942e−6
$X1$	5000	7/41/2.849917/7.8704e−8	7/41/2.851434/8.7749e−8	8/21/2.427398/1.1815e−7	15/87/7.109234/1.7655e−6
$X2$	5000	5/34/2.296969/9.0364e−8	5/34/ 2.270427/9.0364e−8	5/12/1.571940/1.3310e−8	9/62/4.837005/1.0809e−6
$X3$	5000	12/90/5.973961/6.1683e−7	11/82/5.477864/1.3189e−6	25/76/8.040548/5.4767e−7	21/125/10.356632/2.1089e−6
$X4$	5000	5/33/2.220747/7.2463e−7	5/33/2.216270/7.2463e−7	24/72/7.918396/8.2277e−9	15/87/7.117451/1.0091e−6
$X5$	5000	12/92/6.211069/3.0472e−6	12/92/6.121102/3.9973e−7	7/17/2.048574/5.3294e−7	25/147/11.861205/1.6681e−6
$X6$	5000	11/84/5.567125/3.2077e−6	11/84/5.571157/9.2490e−6	7/17/2.052891/5.3143e−7	26/153/12.353363/4.7382e−6
$X1$	10,000	7/41/4.905680/1.1130e−7	7/41/4.931413/1.2410e−7	8/21/4.143385/1.6709e−7	15/87/12.236579/2.4969e−6
$X2$	10,000	5/34/3.861955/1.2779e−7	5/34/3.896158/1.2779e−7	5/12/2.508527/1.8823e−8	9/62/8.397330/1.5286e−6
$X3$	10,000	19/142/17.00798/6.0537e−7	19/138/16.18364/1.1360e−6	25/76/14.340159/7.7389e−7	19/113/16.112636/4.4575e−6
$X4$	10,000	5/33/3.864097/1.0248e−6	5/33/3.817078/1.0248e−6	24/72/13.386262/1.1636e−8	15/87/12.730290/1.4270e−6
$X5$	10,000	11/84/9.720422/4.3794e−6	12/92/11.055953/6.8174e−7	7/17/3.604348/7.5317e−7	22/123/17.881569/2.0766e−6
$X6$	10,000	11/84/9.820316/4.6048e−6	12/92/10.630229/5.1384e−7	7/17/3.580710/7.5211e−7	22/123/17.781163/2.0900e−6

Table 3 Numerical results for Problem 3

Initial	n	Algorithm 2.1	Algorithm 4.1	Algorithm Yu	Algorithm Xiao
$X1$	1000	9/25/0.547673/2.2591e−6	9/25/0.395914/2.2911e−6	6/14/0.324187/1.1099e−7	17/86/1.312292/3.1089e−6
$X2$	1000	8/22/0.350130/5.3636e−6	8/22/0.337711/5.4142e−6	21/63/1.118632/6.1868e−6	20/116/1.567220/1.3824e−6
$X3$	1000	8/22/0.356579/8.4638e−6	8/22/0.341820/8.5437e−6	24/72/1.325037/1.2567e−6	18/96/1.401757/1.5743e−6
$X4$	1000	8/22/0.347774/8.1740e−6	8/22/0.347536/8.2511e−6	22/66/1.198547/4.7603e−6	23/129/1.818146/2.8211e−6
$X5$	1000	8/22/0.336119/6.9818e−6	8/22/0.345838/7.0476e−6	26/78/1.438305/2.6435e−7	22/108/ 1.556767/1.9616e−6
$X6$	1000	8/22/0.345559/6.9849e−6	8/22/ 0.345362/7.0507e−6	26/78/1.414044/2.6447e−7	21/113/1.619568/1.2572e−6
$X1$	5000	9/25/5.539640/5.0786e−6	7/18/1.833878/8.8443e−6	6/14/1.917296/2.4508e−7	23/125/11.262967/1.6531e−6
$X2$	5000	9/25/2.296174/1.1985e−6	7/18/1.699606/2.0817e−6	22/65/7.342433/1.3697e−7	14/71/6.367238/4.6264e−6
$X3$	5000	9/25/2.289349/1.8947e−6	7/18/1.663373/2.6722e−6	23/68/7.635335/1.0935e−7	21/109/9.652215/1.5062e−6
$X4$	5000	9/25/2.356509/1.8262e−6	7/18/1.693403/3.1712e−6	20/59/6.794649/8.1857e−7	19/97/8.626004/1.6590e−6
$X5$	5000	9/25/2.281473/1.5602e−6	7/18/1.673535/2.7544e−6	22/66/7.348086/9.0939e−6	8/92/8.226266/1.2723e−6
$X6$	5000	9/25/2.280177/1.5603e−6	7/18/1.660634/2.7546e−6	22/66/7.360942/9.0947e−6	18/92/8.175414/1.3553e−6
$X1$	10,000	9/25/3.805793/7.1835e−6	7/18/3.114778/1.9903e−6	7/17/3.765820/1.7495e−7	19/99/15.157439/1.9584e−6
$X2$	10,000	9/25/3.815148/1.6951e−6	7/18/2.783268/4.6836e−7	20/59/10.959488/7.5971e−7	18/98/14.058232/1.1839e−6
$X3$	10,000	9/25/3.803131/2.6807e−6	7/18/2.779474/6.0193e−7	22/65/12.2720213.0634e−7	19/104/15.183526/4.4127e−6
$X4$	10,000	9/25/3.785884/2.5830e−6	7/18/2.805785/7.1341e−7	21/62/11.917129/5.8455e−7	19/98/14.738366/4.2065e−6
$X5$	10,000	9/25/3.943582/2.2068e−6	7/18/2.847436/6.1965e−7	26/78/14.409721/8.3613e−7	22/117/16.935706/2.4063e−6
$X6$	10,000	9/25/3.829722/2.2069e−6	7/18/2.806587/6.1966e−7	26/78/14.501016/8.3617e−7	21/112/16.447163/4.9568e−6

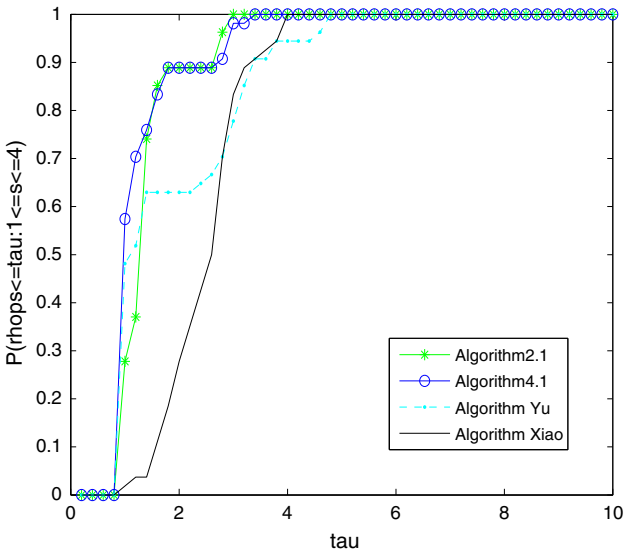

Fig. 1 Performance profile for the number of iterations

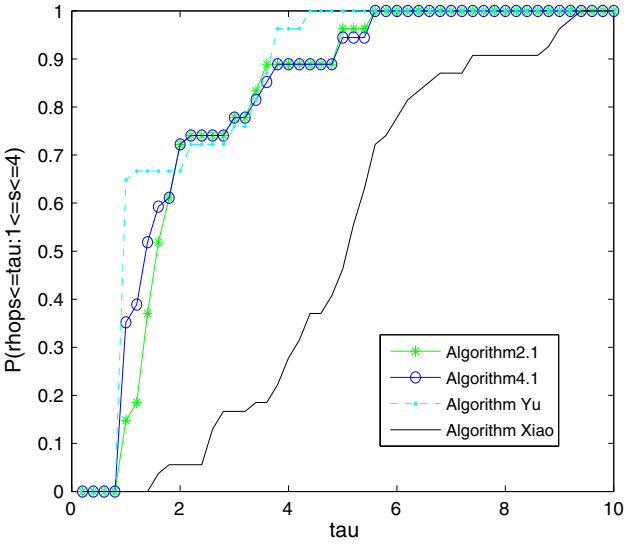

Fig. 2 Performance profile for the number of function evaluations

According to an anonymous referee's suggestion, in the second set of numerical experiments, we compare the performance of Algorithms 2.1 and 4.1 with two typical gradient-based methods for solving large-scale unconstrained optimization problems: L-BFGS method with the Wolfe line search (see Ref. Nocedal and Wright 1999 for details) and CG-HZ method with the Wolfe line search (Hager and Zhang 2005). The test problems are listed as follows (Li and Li 2011):

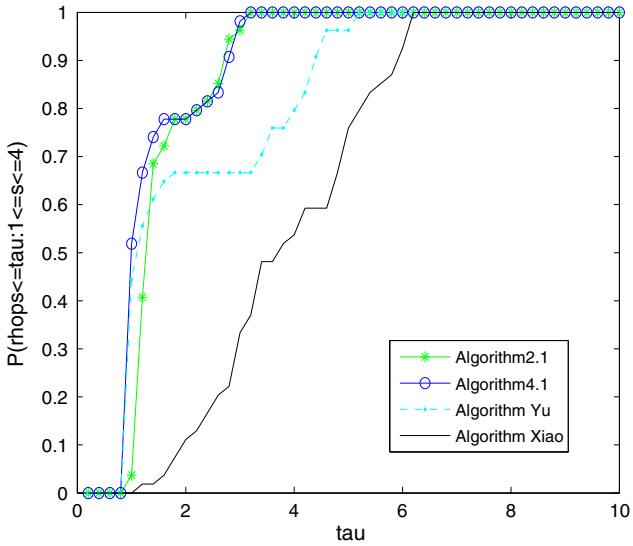

Fig. 3 Performance profile for the CPU time

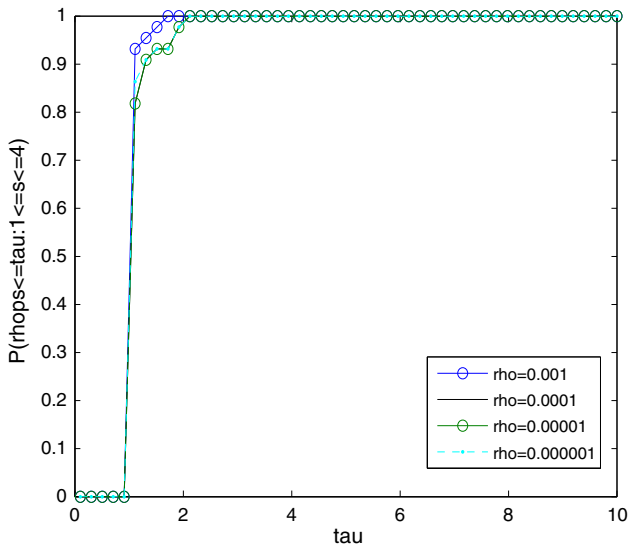

Fig. 4 Performance profile for CPU with varying values of ρ

Problem 4 The elements of function F are given by

$$F_i(x) = 2x_i - \sin(x_i), \quad i = 1, 2, \ldots, n, \quad \text{and} \quad X = R^n.$$

Problem 5 The elements of function F are given by

$$F_i(x) = \ln(x_i + 1) - \frac{x_i}{n}, \quad i = 1, 2, \ldots, n, \quad \text{and} \quad X = R^n.$$

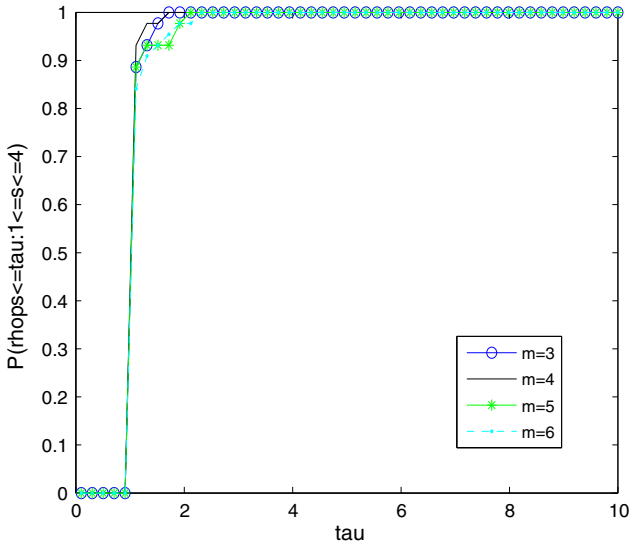

Fig. 5 Performance profile for CPU with varying values of m

These testing problems start at six initial points listed as follows:

$$X7 = (10, 10, \ldots, 10)^T, \quad X8 = (1, 1, \ldots, 1)^T, \quad X9 = \left(1, \frac{1}{2}, \ldots, \frac{1}{n}\right),$$
$$X10 = (0.1, 0.1, \ldots, 0.1)^T, \quad X11 = -X10, \quad X12 = \left(\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right)^T.$$

Throughout the computational experiments, the L-BFGS method and the CG-HZ method implement the Wolfe line search conditions (Nocedal and Wright 1999) with $c_1 = 0.01$ and $c_2 = 0.9$. In addition, the merit function $f(x) = \sum_{i=1}^{n} F_i^2(x)$ is used in implementing the L-BFGS method and the CG-HZ method. Each method is stopped if the condition (5.1) is satisfied.

Tables 4 and 5 show the test results, where M is the number of limited memory corrections stored, and the numerical results are given in the form of T/FN, while other meanings are the same as those in Table 1. Based on the test results in Tables 4 and 5, we also give the performance profiles in Fig. 6.

From Fig. 6, we can see that for the test problems, the proposed algorithms are competitive to the L-BFGS and CG-HZ methods for large-scale problems, in terms of the CPU times and robustness, while the latter two methods perform slightly better than the former in terms of the accuracy of approximate solutions obtained.

While it would be unwise to draw any firm conclusions from the limited numerical results and comparisons, they indicate some promise for the new approaches proposed in this paper, compared with the related methods mentioned above. Further improvement is expected from more suitable implementation.

Table 4 Numerical results for Problem 4

Initial	n	Algorithm 2.1	Algorithm 4.1	L-BFGS ($M = 5$)	CG-HZ
$X7$	1000	0.035525/5.1997e−6	0.032351/1.0494e−8	0.040530/1.6347e−9	0.077080/5.0105e−9
$X8$	1000	0.028782/1.9191e−6	0.029749/3.8730e−9	0.031511/0	0.065775/1.7359e−12
$X9$	1000	0.027349/1.8014e−6	0.011660/7.6701e−6	0.025786/1.8902e−8	0.057296/6.3475e−9
$X10$	1000	0.029011/3.1004e−6	0.018732/6.2288e−10	0.009580/2.5037e−	0.023951/1.8653e−6
$X11$	1000	0.016582/3.1004e−6	0.015378/6.2288e−10	0.005595/2.5097e−6	0.004238/1.4792e−6
$X12$	1000	0.011523/3.1623e−6	0.011287/3.1623e−6	0.006585/7.5897e−7	0.001201/1.1758e−16
$X7$	5000	0.107509/1.1575e−6	0.091493/2.3464e−8	0.188467/3.6553e−9	1.817904/1.1204e−8
$X8$	5000	0.109331/4.2913e−6	0.106956/8.6603e−9	0.117536/0	0.117536/3.8818e−12
$X9$	5000	0.107125/1.8158e−6	0.114574/7.7015e−6	0.032687/1.8436e−8	1.181637/6.4607e−9
$X10$	5000	0.061324/6.9327e−6	0.060783/1.3928e−9	0.121511/5.6119e−6	0.002434/4.1709e−6
$X11$	5000	0.063104/6.9327e−6	0.060920/1.3928e−9	0.023815/5.6119e−6	0.002268/4.1709e−6
$X12$	5000	0.043017/1.4142e−6	0.043217/1.4142e−6	0.007784/1.3576e−8	0.023142/5.7115e−16
$X7$	10,000	0.216766/1.6369e−6	0.141606/3.3184e−8	0.173004/5.1692e−9	6.717467/8.6271e−8
$X8$	10,000	0.197210/6.0688e−6	0.157126/1.2248e−8	0.017539/0	2.875060 /5.6811e−12
$X9$	10,000	0.208963/1.8176e−6	0.165072/7.7053e−6	0.092291/1.8433e−8	1.971839/2.8117e−9
$X10$	10,000	0.145317/9.8043e−6	0.153223/1.9697e−9	0.215370/7.9364e−6	0.007026/5.1147e−6
$X11$	10,000	0.120093/2.2068e−6	0.116030/6.1965e−9	0.031878/8.3613e−6	1.209409/2.4063e−19
$X12$	10,000	0.063977/1.0000e−5	0.063069/1.0000e−5	0.013800/2.4000e−9	0.04093/7.3997e−16

Table 5 Numerical results for Problem 5

Initial	n	Algorithm 2.1	Algorithm 4.1	L-BFGS ($M = 5$)	CG-HZ
$X7$	1000	0.021633/1.3025e−9	0.035549/2.7384e−8	F/· · · · ·	0.160925/9.1095e−10
$X8$	1000	0.011651/3.5989e−8	0.012168/3.5989e−8	0.107067/0	0.060023/2.2875e−8
$X9$	1000	0.027768/5.1018e−6	0.026161/3.3940e−7	0.119357/2.1607e−8	0.082597/4.5824e−7
$X10$	1000	0.007867/5.1671e−7	0.013580/5.1671e−7	0.148488/8.3416e−7	0.047715/5.0421e−11
$X11$	1000	0.012690/1.2134e−8	0.012318/1.2134e−8	0.121127/6.5377e−9	0.108993/1.6830e−6
$X12$	1000	0.005824/4.7412e−8	0.005943/4.7412e−8	0.007882/2.1184e−6	0.016463/6.1988e−6
$X7$	5000	0.121376/3.7014e−6	0.103545/4.2530e−9	F/· · · · ·	2.324181/2.3894e−10
$X8$	5000	0.078646/6.2639e−9	0.060322/6.2639e−9	0.092057/0	0.839026/4.9169e−8
$X9$	5000	0.092062/3.0112e−6	0.132575/5.8975e−7	0.088574/2.2055e−8	0.991495/4.7487e−6
$X10$	5000	0.046364/1.8262e−7	0.039732/3.1712e−7	0.079612/8.1857e−6	0.656544/1.6590e−10
$X11$	5000	0.075229/6.2691e−7	0.070844/6.2691e−7	0.059276/1.4474e−8	0.359509/3.8401e−6
$X12$	5000	0.047618/4.2416e−6	0.033661/4.2416e−6	0.059732/3.7389e−8	0.018750/1.4105e−6
$X7$	10,000	0.229804/5.1192e−6	0.177057/2.5400e−9	F/· · · · ·	9.299786/1.9640e−10
$X8$	10,000	0.108464/3.6181e−9	0.098387/3.6181e−9	0.047662/0	3.347611/6.9185e−8
$X9$	10,000	0.158438/3.0306e−6	0.180751/5.7651e−6	0.140105/2.1703e−8	3.985263/4.7115e−6
$X10$	10,000	0.070213/1.2137e−7	0.071586/1.2137e−7	0.078596/2.8285e−6	2.657101/1.7141e−10
$X11$	10,000	0.099172/3.7798e−7	0.105279/3.7798e−7	0.097139/2.1798e−8	2.007984/7.5488e−13
$X12$	10,000	0.036717/1.4998e−6	0.042842/1.4998e−6	0.057065/6.6047e−9	0.010320/4.9935e−7

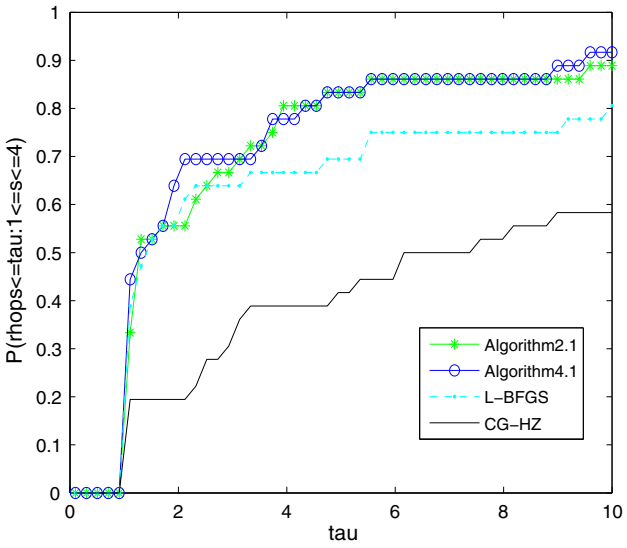

Fig. 6 Performance profile for the CPU time

6 Conclusion

In this paper, we propose two new supermemory gradient methods for convex-constrained nonlinear monotone equations. The main properties of the proposed methods are that we establish the global convergence without using any merit function and making the differentiability assumption. Furthermore, these methods do not solve any subproblem and store any matrix. Thus, they can be applied to solve large-scale nonlinear equations. Preliminary numerical results are reported to show that the proposed methods are efficient.

Since the most computational cost of each algorithm for problem (1.1) is to determine the search direction d_k and find the stepsize α_k in line search, we will study some more effective methods for constructing a descent direction d_k and an inexpensive line search in our future research. Furthermore, we should proceed to make some numerical comparisons of the performance between our proposed methods and some popular algorithms for nonsmooth problems in the future research work.

Acknowledgements The authors are very grateful to the anonymous referees and the associate editor for their valuable comments and suggestions that greatly improved this paper.

References

Ahookhosh M, Amini K, Bahrami S (2013) Two derivative-free projection approaches for systems of large-scale nonlinear monotone equations. Numer Algorithms 64:21–42

Dolan ED, More JJ (2002) Benchmarking optimization software with performance profiles. Math Program Ser A 91:201–213

Fan JY (2013) On the Levenberg–Marquardt method for convex constrained nonlinear equations. J Ind Manag Optim 9:227–241

Figueiredo M, Nowak R, Wright SJ (2007) Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems. IEEE J Sel Topics Signal Process 1:586–597

Hager WW, Zhang HC (2005) A new conjugate gradient method with guaranteed descent and an efficient line search. SIAM J Optim 16:170–192
Hager WW, Zhang HC (2006) A survey of nonlinear conjugate gradient methods. Pac J Optim 2:35–58
Iusem AN, Solodov MV (1997) Newton-type methods with generalized distance for constrained optimization. Optimization 41:257–278
Jia CX, Zhu DT (2011) Projected gradient trust-region method for solving nonlinear systems with convex constraints. Appl Math J Chin Univ (Ser B) 26:57–69
Li QN, Li DH (2011) A class of derivative-free methods for large-scale nonlinear monotone equations. IMA J Numer Anal 31:1625–1635
Narushima Y, Yabe H (2006) Global convergence of a memory gradient method for unconstrained optimization. Comput Optim Appl 35:325–346
Nocedal J, Wright SJ (1999) Numerical optimization. Springer, New York
Ou YG, Wang GS (2012) A new supermemory gradient method for unconstrained optimization problems. Optim Lett 6:975–992
Ou YG, Liu Y (2014) A nonmonotone supermemory gradient algorithm for unconstrained optimization. J Appl Math Comput 46:215–235
Shi ZJ, Shen J (2005) A new supermemory gradient method with curve search rule. Appl Math Comput 170:1–16
Solodov MV, Svaiter BF (1999) A new projection method for variational inequality problems. SIAM J Control Optim 37:765–776
Sun M, Bai QG (2011) A new descent memory gradient method and its global convergence. J Syst Sci Complexity 24:784–794
Tong XJ, Zhou SZ (2005) A smoothing projected Newton-type method for semismooth equations with bound constraints. J Ind Manag Optim 1:235–250
Wang CW, Wang YJ, Xu CL (2007) A projection method for a system of nonlinear monotone equations with convex constraints. Math Methods Oper Res 66:33–46
Wood AJ, Wollenberg BF (1996) Power generations, operations, and control. Wiley, New York
Xiao YH, Zhu H (2013) A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. J Math Anal Appl 405:310–319
Yan QR, Peng XZ, Li DH (2010) A globally convergent derivative-free method for solving large scale nonlinear monotone equations. J Comput Appl Math 234:649–657
Yu ZS, Lin J, Sun J et al (2009) Spectral gradient projection method for monotone nonlinear equations with convex constraints. Appl Numer Math 59:2416–2423
Zhao YB, Li D (2001) Monotonicity of fixed point and normal mapping associated with variational inequality and is applications. SIAM J Optim 11:962–973