Journal of Applied Analysis and Computation
Volume 13, Number 4, August 2023, 2006–2025
Website:http://www.jaac-online.com
DOI:10.11948/20220335

SIGNAL RECOVERY WITH CONSTRAINED MONOTONE NONLINEAR EQUATIONS THROUGH AN EFFECTIVE THREE-TERM CONJUGATE GRADIENT METHOD*

Peiting Gao[1,†], Wen Zheng[1,†], Tao Wang[1], Yifei Li[1] and Futong Li[1]

Abstract In this paper, we introduce a three-term conjugate gradient-type projection method for solving constrained monotone nonlinear equations. In this method, we firstly undertake the transformation of the relative matrices proposed by Yao and Ning. Secondly, we obtain the new relative matrices involving two parameters. Subsequently, we construct a relationship for the two parameters via the quasi-Newton equation and obtain the parameters by simplifying maximum eigenvalue of the new relative matrices. Finally, combining the three-term conjugate gradient method with projection technique, we establish an efficient three-term conjugate gradient-type projection algorithm. Meanwhile, we also give some theoretical analysis about the global convergence and R-linear convergence of the proposed algorithm under quite reasonable technical assumptions. Performance comparisons show that our proposed method is competitive and efficient for solving large-scale nonlinear monotone equations with convex constraints. Furthermore, the presented algorithm is also applied to recovery of a sparse signal in compressive sensing, and obtain practical, efficient and competitive performance in comparing with state-of-the-art algorithms.

Keywords Quasi-Newton equation, conjugate gradient method, global convergence, signal recovery.

MSC(2010) 65F10, 90C52, 65K05.

1. Introduction

In this paper, we aim at finding the numerical solutions of the following system of nonlinear equations:

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

†The corresponding author.
Email: gaopeiting3680@163.com(P. Gao), zhengwen@tyut.edu.cn(W. Zheng)
[1]College of Computer Science and Technology(College of Data Science), Taiyuan University of Technology, No. 79 West Street Yingze, Taiyuan 030024, China

*This research was partially supported by the Natural Science Foundation of Shanxi Province, China(No. 202203021212255).

where Ω is a non-empty closed convex set in $\mathcal{R}^n$, and $F : \mathcal{R}^n \to \mathcal{R}^n$ is continuous and monotone. Here, monotone means that

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

Most state-of-the-art solvers are built on conjugate gradient(CG)-based methods to solve (1.1)(Refs. [4–7, 13–16, 19, 20]). Due to low storage requirement and simplicity, some of these methods have been successfully applied to recovery of a sparse signal problem(Refs. [4, 5, 7, 13, 19]), and seem to have the potential to deal with image restoration [20].

Recently, Gao et.al. [4] introduced a three-term CG-type projection method involving signal parameter to deal with (1.1). Furthermore, the method is applied to decode a sparse signal in compressed sensing. Some numerical results reported that the proposed method attained better performance than its competitors.But, due to the search direction in [4] dissatisfied with quasi-Newton equation, it may lead to bad numerical results in practice.

In order to overcome the above drawbacks, the CG methods involving double parameters for solving unconstrained optimization problems are proposed(Refs. [1, 8, 18]). As far as we know, three-term CG method with double parameters are only used to solve unconstrained optimization problems. It is interesting to study three-term CG methods with double parameters for solving (1.1). To further improve the effectiveness [18], by mean of [1], we propose a double parameters projection method for constrained monotone nonlinear equations with application to compressive sensing.

The main contribution of this paper is to develop an efficient three-term CG-type projection method for solving (1.1). As is known to us, the quasi-Newton equation plays an important role in the CG method. The novelty of the proposed method is that the method satisfies the quasi-Newton equation and the parameters are obtained by simplifying the maximum eigenvalue of the relative matrices. The detailed description are showed as follows. Firstly, we undertake the transformation of the relative matrices obtained by Yao and Ning [18]. Secondly, based on [1], we obtain the new relative matrices that includes double parameters. Whereafter, the two parameters are connected by the quasi-Newton secant equation and obtained by simplifying this maximum eigenvalue of the relative matrices. Finally, combining the three-term CG method with projection technique, we establish a three-term CG-type projection algorithm to solve (1.1). Some numerical results are presented to illustrate that the proposed method is capable of providing considerable speed advantages over methods [4]. Moreover, we also do some numerical experiments to verify the proposed method is efficient and promising better than two algorithms it was tested with in compressive sensing.

This paper is organized as follows. In Section 2, we present an efficient three-term CG-type projection algorithm for solving (1.1). In Section 3, the global convergence of the proposed algorithm is proved under milder conditions. In Section 4, under appropriate assumptions, the R-linear convergence rate of this algorithm is proved. In Section 5, computational experiments are reported to show the efficiency of the proposed algorithm. In Section 6, we apply the proposed algorithm to deal with sparse signal reconstruction in compressive sensing.

2. The Proposed Algorithm

In this section, we propose an iterative update matrix(relative matrix) and obtain the CG parameters under the Wolfe line search, then present our three-term CG algorithm. By combining projection technique, we obtain an efficient projection method to solve (1.1).

Due to simplicity and low storage, three-term CG methods have been widely used for unconstrained optimization problem:

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

where $f : R^n \to R$ is continuously differentiable. The iterative formula of these methods are computed by

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

where $\alpha_k > 0$ is step-size computed by some line search and d_{k+1} is the search direction determined by

$$d_{k+1} = -Q_{k+1}g_{k+1}, \qquad k \geq 0 \tag{2.1}$$

where Q_{k+1} is the iterative update matrix of d_{k+1}, with $d_0 = -\nabla f(x_0)$.

The updating formula for the matrix Q_{k+1} proposed by Yao and Ning [18] can be expressed as the following form

$$Q_{k+1} = I + A_2^{k+1} + \gamma_k A_1^{k+1},$$

where $A_2^{k+1} = \frac{y_k s_k^T - s_k y_k^T}{s_k^T y_k}$, $A_1^{k+1} = \frac{s_k s_k^T}{s_k^T y_k}$, and γ_k is a positive parameter, with $s_k = x_{k+1} - x_k$, $y_k = g_{k+1} - g_k$.

Motivated by [18], we proposed the iterative update matrix as follows

$$Q_{k+1}^1 = Q_{k+1}^T = I + A_2^{k+1} + \gamma_k A_1^{k+1}, \tag{2.2}$$

where $A_2^{k+1} = \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k}$, $A_1^{k+1} = \frac{s_k s_k^T}{s_k^T y_k}$, and γ_k is a positive parameter.

Recently, Anderi [1] proposed the following matrix with double parameters

$$Q_{k+1} = \delta_k I + \delta_k A_2^{k+1} + \gamma_k A_1^{k+1}, \tag{2.3}$$

where $A_2^{k+1} = -\frac{s_k y_k^T + y_k s_k^T}{s_k^T y_k}$, $A_1^{k+1} = \frac{s_k s_k^T}{s_k^T y_k}$, δ_k and γ_k are positive parameters.

Combining (2.2) with (2.3), we proposed the following iterative update matrix

$$Q_{k+1} = \delta_k I + \delta_k A_2^{k+1} + \gamma_k A_1^{k+1}, \tag{2.4}$$

where $A_2^{k+1} = \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k}$, $A_1^{k+1} = \frac{s_k s_k^T}{s_k^T y_k}$, δ_k and γ_k are positive parameters.

By quasi-Newton equation, we obtain the relationship between two parameters as follows:

$$\begin{aligned} Q_{k+1}y_k &= \left(\delta_k I + \delta_k \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k} + \gamma_k \frac{s_k s_k^T}{s_k^T y_k}\right) y_k \\ &= \delta_k y_k + \delta_k \frac{s_k \|y_k\|^2 - y_k s_k^T y_k}{s_k^T y_k} + \gamma_k \frac{s_k^T y_k}{s_k^T y_k} s_k \end{aligned}$$

$$
\begin{aligned}
&= \delta_k y_k + \delta_k \frac{\|y_k\|^2}{s_k^T y_k} s_k - \delta_k y_k + \gamma_k s_k \\
&= \delta_k \frac{\|y_k\|^2}{s_k^T y_k} s_k + \gamma_k s_k \\
&= \left(\gamma_k + \delta_k \frac{\|y_k\|^2}{s_k^T y_k} \right) s_k.
\end{aligned}
$$

Then we obtain

$$
\gamma_k = 1 - \delta_k \frac{\|y_k\|^2}{s_k^T y_k}. \tag{2.5}
$$

By (2.4) and (2.5), we can get

$$
\begin{aligned}
Q_{k+1} &= \delta_k I + \delta_k \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k} + \left(1 - \delta_k \frac{\|y_k\|^2}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k} \\
&= \delta_k \left(I + \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k} + \left(\frac{1}{\delta_k} - \frac{\|y_k\|^2}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k} \right) \\
&= \delta_k B_{k+1},
\end{aligned} \tag{2.6}
$$

where $B_{k+1} = I + \frac{s_k y_k^T - y_k s_k^T}{s_k^T y_k} + \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k} \right) \frac{s_k s_k^T}{s_k^T y_k}$, with $\theta_k = \frac{1}{\delta_k}$.

Theorem 2.1. *The matrix B_{k+1} in (2.6) is nonsingular, and its eigenvalues consist of 1 ($(n-2)$ multiplicity),*

$$
\lambda_{k+1}^+ = 1 + \frac{a}{2} + \frac{1}{2} \sqrt{a^2 - 4 \frac{\|s_k\|^2 \|y_k\|^2}{(s_k^T y_k)^2} + 4},
$$

and

$$
\lambda_{k+1}^- = 1 + \frac{a}{2} - \frac{1}{2} \sqrt{a^2 - 4 \frac{\|s_k\|^2 \|y_k\|^2}{(s_k^T y_k)^2} + 4},
$$

with $a = \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k} \right) \frac{\|s_k\|^2}{s_k^T y_k}$.

Proof. By the Wolfe line search condition, we have that $s_k^T y_k > 0$, which implies that y_k, s_k are nonzero vectors. Let V be the vector space spanned by $\{s_k, y_k\}$ and $V^\perp$ is the orthogonal complement of the vector space V. Obviously, $dim(V) \leq 2$ and $dim(V^\perp) \geq n-2$. Thus, there exists a set of mutually orthogonal unit vectors $\{u_k^i\} \subset V^\perp$ such that

$$
s_k^T u_k^i = y_k^T u_k^i = 0, \ i = 1, 2, \ldots, n-2.
$$

Combining the definition of B_{k+1}, we know

$$
B_{k+1} u_k^i = u_k^i, \ i = 1, 2, \ldots, n-2.
$$

Therefore, B_{k+1} has $(n-2)$ eigenvalues, 1, corresponding to the eigenvectors $\{u_k^i\}_{i=1}^{n-2}$.

Now, we are interested to find the rest eigenvalues of B_{k+1}, denoted by λ_{k+1}^+ and λ_{k+1}^-, respectively.

By the formula of algebra (see [10]):

$$det(I + pq^T + uv^T) = (1 + q^T p)(1 + v^T u) - (p^T v)(q^T u),$$

where

$$p = \frac{s_k}{s_k^T y_k}, q = y_k, u = \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k}\right)\frac{s_k}{s_k^T y_k} - \frac{y_k}{s_k^T y_k}, v = s_k,$$

it is easily derived that

$$det(B_{k+1}) = \theta_k \frac{\|s_k\|^2}{s_k^T y_k}.$$

So, B_{k+1} is a nonsingular matrix and we have

$$\lambda_{k+1}^+ \lambda_{k+1}^- = \theta_k \frac{\|s_k\|^2}{s_k^T y_k}. \tag{2.7}$$

By a direct computation, we have

$$tr(B_{k+1}) = n + \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k}\right)\frac{\|s_k\|^2}{s_k^T y_k}.$$

Thus, we obtain

$$\lambda_{k+1}^+ + \lambda_{k+1}^- + n - 2 = n + \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k}\right)\frac{\|s_k\|^2}{s_k^T y_k},$$

and further

$$\lambda_{k+1}^+ + \lambda_{k+1}^- = 2 + \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k}\right)\frac{\|s_k\|^2}{s_k^T y_k}. \tag{2.8}$$

By (2.7) and (2.8), we construct the following quadratic equation:

$$\lambda^2 - \left(2 + \left(\theta_k - \frac{\|y_k\|^2}{s_k^T y_k}\right)\frac{\|s_k\|^2}{s_k^T y_k}\right)\lambda + \theta_k \frac{\|s_k\|^2}{s_k^T y_k} = 0.$$

By a straightforward computation, we get

$$\lambda_{k+1}^+ = 1 + \frac{a}{2} + \frac{1}{2}\sqrt{a^2 - 4\frac{\|s_k\|^2\|y_k\|^2}{(s_k^T y_k)^2} + 4}, \tag{2.9}$$

$$\lambda_{k+1}^- = 1 + \frac{a}{2} - \frac{1}{2}\sqrt{a^2 - 4\frac{\|s_k\|^2\|y_k\|^2}{(s_k^T y_k)^2} + 4}.$$

Last, it is easy to prove that $\lambda_{k+1}^- \geq 1$, and so $\lambda_{k+1}^+ \geq \lambda_{k+1}^- \geq 1$(by (2.12) below). The proof is complete.

In fact, by the formula of λ_{k+1}^+, θ_k must satisfy

$$\theta_k \geq \frac{\|y_k\|^2}{s_k^T y_k} + \frac{2}{\|s_k\|^2}\sqrt{\|s_k\|^2\|y_k\|^2 - (s_k^T y_k)^2} \tag{2.10}$$

or

$$0 < \theta_k \leq \frac{\|y_k\|^2}{s_k^T y_k} - \frac{2}{\|s_k\|^2}\sqrt{\|s_k\|^2\|y_k\|^2 - (s_k^T y_k)^2}. \tag{2.11}$$

To remove the radical of (2.9), we match this formula into a complete square formula. Then, by (2.5) and (2.10), it is easy to obtain

$$\theta_k = \frac{2\|y_k\|^2}{s_k^T y_k}, \delta_k = \frac{s_k^T y_k}{2\|y_k\|^2}, \gamma_k = \frac{1}{2}. \tag{2.12}$$

By (2.1), (2.4) and (2.12), we obtain the search direction as follows:

$$d_{k+1} = -\frac{s_k^T y_k}{2\|y_k\|^2} g_{k+1} - \frac{g_{k+1}^T y_k}{2\|y_k\|^2} s_k + \frac{g_{k+1}^T s_k}{2\|y_k\|^2} y_k - \frac{g_{k+1}^T s_k}{2 s_k^T y_k} s_k. \tag{2.13}$$

We now present our projection algorithm to deal with (1.1) by combining (2.13) with the projection technology proposed by Solodov and Svaiter [11], and the search direction d_k is computed by

$$d_{k+1} = -\frac{s_k^T \bar{y}_k}{2\|\bar{y}_k\|^2} F_{k+1} - \frac{F_{k+1}^T \bar{y}_k}{2\|\bar{y}_k\|^2} s_k + \frac{F_{k+1}^T s_k}{2\|\bar{y}_k\|^2} \bar{y}_k - \frac{F_{k+1}^T s_k}{2 s_k^T \bar{y}_k} s_k, \tag{2.14}$$

where $\bar{y}_k = F_{k+1} - F_k + r s_k$, $s_k = x_{k+1} - x_k$,with $r > 0$ a constant. □

The proposed algorithm is summarized as follows.

Algorithm 2.1.
Step 0: Given an initial point $x_0 \in \Omega \subset \mathcal{R}^n$, and $0 < \rho < 1$, $0 < \sigma < 1$, $r > 0$, $\xi > 0$, and $\varepsilon > 0$. Set $k = 0$.
Step 1: Let $||F_k|| \leq \varepsilon$, stop. Otherwise go to step 2.
Step 2: Computing d_k: If $k = 0$, then $d_k = -F_k$. Otherwise, d_{k+1} is computed by (2.14).
Step 3: Set $z_k = x_k + \alpha_k d_k$, and compute $\alpha_k = \xi\rho^m$ in which m is the smallest nonnegative integer such that

$$-F(z_k)^T d_k \geq \sigma\alpha_k ||d_k||^2. \tag{2.15}$$

Step 4: If $z_k \in \Omega$ and $F(z_k) = 0$, then $x_{k+1} = z_k$, stop. Otherwise, the next iterative point x_{k+1}:

$$x_{k+1} = P_\Omega[x_k - \xi_k F(z_k)], \tag{2.16}$$

where

$$\xi_k = \frac{F(z_k)^T (x_k - z_k)}{||F(z_k)||^2}.$$

Step 5: Compute $\bar{y}_k = F_{k+1} - F_k + r s_k$, $s_k = x_{k+1} - x_k$, set $k := k + 1$, go to step 1.

Remark 2.2. It can be proved that there always exists a nonnegative integer m which satisfies the inequality (2.15). In fact, if the inequality (2.15) does not hold for any nonnegative integer i, namely,

$$-F(x_k + \xi\rho^i d_k)^T d_k < \sigma\xi\rho^i \|d_k\|^2.$$

By the continuity of F, $\rho \in (0, 1)$, and letting $i \to \infty$, we obtain

$$-F(x_k)^T d_k \leq 0. \tag{2.17}$$

However, by Lemma 3.2 below, we have

$$-F(x_k)^T d_k \geq c\|F(x_k)\|^2 > 0,$$

which contradicts (2.17).

3. Convergence property

In this section, we discuss the convergence of Algorithm 2.1. We need the two assumptions as follows:

(H_1) The solution set of (1.1), denoted by Ω^*, is nonempty and closed.

(H_2) The mapping F is Lipschitz continuous on $\mathcal{R}^n$; that is, there exists constant $L > 0$ such that

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

We give the following lemmas to prove the convergence of Algorithm 2.1.

Lemma 3.1. *Suppose that assumptions $(H_1) - (H_2)$ hold, and let $\{x_k\}$ and $\{z_k\}$ be the sequences generated by Algorithm 2.1. Then we have*

$$r\|s_k\|^2 \leq s_k^T \bar{y}_k \leq (L + r)\|s_k\|^2. \tag{3.1}$$

The detailed proof is similar to Lemma 3.1 which originates from [1]. Thus, we omit the proof here.

Remark 3.1. By Cauchy-Schwarz inequality and (3.1), we have

$$r\|s_k\|^2 \leq s_k^T \bar{y}_k \leq \|s_k\|\|\bar{y}_k\|,$$

then we have

$$r\|s_k\| \leq \|\bar{y}_k\|. \tag{3.2}$$

By the definition of $\bar{y}_k$, Cauchy-Schwarz inequality and (H2), we have

$$\|\bar{y}_k\| \leq \|F_{k+1} - F_k\| + r\|s_k\| \leq L\|x_{k+1} - x_k\| + r\|s_k\| \leq (L + r)\|s_k\|. \tag{3.3}$$

By (3.2) and (3.3), we have

$$r\|s_k\| \leq \|\bar{y}_k\| \leq (L + r)\|s_k\|. \tag{3.4}$$

Combining (3.1) with (3.4), we obtain

$$\frac{r}{(L + r)^2} \leq \frac{s_k^T \bar{y}_k}{\|\bar{y}_k\|^2} \leq \frac{L + r}{r^2}. \tag{3.5}$$

Lemma 3.2. *Suppose that assumptions $(H_1) - (H_2)$ hold, and let $\{x_k\}$ and $\{z_k\}$ be the sequences generated by Algorithm 2.1. Then we have*

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

with $c = \min\{1, \frac{r}{2(L+r)^2}\}$.

Proof. By (2.14) and (3.5), we have

$$\begin{aligned}
F_{k+1}^T d_{k+1} &= F_{k+1}^T \left(-\frac{s_k^T \bar{y}_k}{2\|\bar{y}_k\|^2} F_{k+1} - \frac{F_{k+1}^T \bar{y}_k}{2\|\bar{y}_k\|^2} s_k + \frac{F_{k+1}^T s_k}{2\|\bar{y}_k\|^2} \bar{y}_k - \frac{F_{k+1}^T s_k}{2 s_k^T \bar{y}_k} s_k \right) \\
&= -\frac{s_k^T \bar{y}_k}{2\|\bar{y}_k\|^2}\|F_{k+1}\|^2 - \frac{F_{k+1}^T \bar{y}_k F_{k+1}^T s_k}{2\|\bar{y}_k\|^2} + \frac{F_{k+1}^T \bar{y}_k F_{k+1}^T s_k}{2\|\bar{y}_k\|^2} - \frac{(F_{k+1}^T s_k)^2}{2 s_k^T \bar{y}_k} \\
&\leq -\frac{s_k^T \bar{y}_k}{2\|\bar{y}_k\|^2}\|F_{k+1}\|^2 \\
&\leq -\frac{r}{2(L + r)^2}\|F_{k+1}\|^2,
\end{aligned}$$

with $c = \min\{1, \frac{r}{2(L+r)^2}\}$. We complete the proof. □

Lemma 3.3. *Suppose that assumptions $(H_1) - (H_2)$ hold, and let $\{x_k\}$ and $\{z_k\}$ be the sequences generated by Algorithm 2.1. Then we have*

$$\alpha_k \geq \min \rho \left\{1, \frac{c||F_k||^2}{(L+\sigma)||d_k||^2}\right\}.$$

The detailed proof is similar to Lemma 3.1 which originates from [1]. Thus, we omit the proof here.

Lemma 3.4. *Suppose that assumptions $(H_1) - (H_2)$ hold. Let d_k be the search direction determined by Algorithm 2.1. Then we have*

$$\|d_{k+1}\| \leq A\|F_{k+1}\|,\ \alpha_k \geq B,$$

where

$$A = \max\left\{\frac{L+r}{2r^2} + \frac{1}{r} + \frac{1}{2r}, 1\right\},$$
$$B = \min \rho \left\{1, \frac{c}{(L+\sigma)A^2}\right\}.$$

Proof. By (2.14),(3.1) and (3.4), then we have

$$\begin{aligned}\|d_{k+1}\| &\leq \frac{s_k^T \bar{y}_k}{2\|\bar{y}_k\|^2}\|F_{k+1}\| + \frac{\|s_k\|}{2\|\bar{y}_k\|}\|F_{k+1}\| + \frac{\|s_k\|}{2\|\bar{y}_k\|}\|F_{k+1}\| + \frac{\|s_k\|^2}{2s_k^T\bar{y}_k}\|F_{k+1}\| \\ &\leq \frac{(L+r)\|s_k\|^2}{2\|\bar{y}_k\|^2}\|F_{k+1}\| + \frac{\|s_k\|}{\|\bar{y}_k\|}\|F_{k+1}\| + \frac{\|s_k\|^2}{2s_k^T\bar{y}_k}\|F_{k+1}\| \\ &\leq \frac{L+r}{2r^2}\|F_{k+1}\| + \frac{1}{r}\|F_{k+1}\| + \frac{1}{2r}\|F_{k+1}\|.\end{aligned}$$

Let $A = \max\left\{\frac{L+r}{2r^2} + \frac{1}{r} + \frac{1}{2r}, 1\right\}$. By this inequality and Lemma 3.3 we obtain the second inequality. The proof is complete. □

Remark 3.2. By the boundedness of $\{x_k\}$ (by Lemma 3.5 below) and (H_2), there exists a constant $\zeta > 0$ such that $\|F_k\| \leq \zeta,\ \forall k \geq 0$. By Lemma 3.4, we have $\|d_k\| \leq A\|F_k\| \leq A\zeta,\ \forall k \geq 0$. Thus $\{\|d_k\|\}$ is bounded. The following result shows the global convergence of Algorithm 2.1.

Lemma 3.5. *Suppose that assumptions $(H_1) - (H_2)$ hold, and let $\{x_k\}$ and $\{z_k\}$ be the sequences generated by Algorithm 2.1. $\forall x^* \in \Omega^*$. Then we have*

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

In particular, the sequence $\{x_k\}$ is bounded and

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0.$$

Furthermore,

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

The detailed proof is similar to Lemma 3.5 which originates from [1]. Thus, we omit the proof here.

Theorem 3.1. *Suppose that assumptions* $(H_1) - (H_2)$ *hold, and let* $\{x_k\}$ *be the sequence generated by Algorithm 2.1. Then we have*

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

The detailed proof is similar to Theorem 3.1 which originates from [1]. Thus, we omit the proof here.

4. R-linear convergence rate

In order to prove the R-linear convergence rate of Algorithm 2.1, an extra assumption should be added:

(H_3) For any $x^* \in \Omega^*$, there exist positive constants μ and δ and $\mu \in (0,1)$ such that

$$\mu dist(x, \Omega^*) \leq ||F(x)||, \quad \forall x \in N(x^*, \delta),$$

where $dist(x, \Omega^*)$ denotes the distance from x to the solution set Ω^*, and

$$N(x^*, \delta) = \{x \in \mathcal{R}^n | ||x - x^*|| \leq \delta\}. \tag{4.1}$$

The following result gives the R-linear convergence rate of Algorithm 2.1.

Theorem 4.1. *Suppose assumptions* $(H_1) - (H_3)$ *hold. Let* $\{x_k\}$ *be the sequence generated by Algorithm 2.1.Then, the sequence* $\{dist(x_k, \Omega^*)\}$ *is Q-linearly convergent to 0 and the sequence* $\{x_k\}$ *is R-linear convergent to* $\bar{x} \in \Omega^*$.

The detailed proof is similar to Theorem 4.1 which originates from [1]. Thus, we omit the proof here.

5. Numerical results

In this section, we present some numerical results to show the performance of our algorithm. Because of our method involving the relative matrices, the methods (A1(Algorithm 2.1a) [4] and A2(Algorithm 2.1b) [4]) proposed by [1] also involves it. Therefore, we compare them in terms of NI, NF, and CPU time, where "NI" and "NF" represent the iterative number, and the iterative number of function value, respectively.The parameters of compared algorithms are taken from [1]. We choose the parameters for the line search employed as $\sigma = 10^{-4}$, $r = 1$, $\xi = 1$, $\rho = 0.4$. The terminate conditions for three algorithms are obtained by [4]. The coding environment for the algorithms are Matlab R2017a and run on a PC with 2.30 GHZ CPU processor and RAM 8.00 GB.

The tested problems are as follows:

Problem 5.1. *The problem is taken from [6]. The mapping* F *is taken as* $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T$, *where*

$$f_1(x) = x_1 - \exp\cos((x_1 + x_2)/2),$$
$$f_i(x) = x_i - \exp\cos((x_{i-1} + x_i + x_{i+1})/i), \ i = 2, 3, \cdots, n-1,$$

$$f_n(x) = x_n - \exp\cos((x_{n-1} + x_n)/n),$$

and $\Omega = \mathcal{R}^n_+$.

Table 1. The experiment results of Problem 5.1 for three algorithms

		A1	A2	Algorithm 2.1
IP	Dim	NI/NF/CPU/FV	NI/NF/CPU/FV	NI/NF/CPU/FV
X1	20000	27/17/0.10/4.79e-6	27/17/0.09/4.79e-6	27/2/0.07/6.60e-6
	80000	27/16/0.38/7.30e-6	27/16/0.36/7.30e-6	28/2/0.34/5.66e-6
	100000	28/17/0.49/2.88e-6	28/17/0.50/2.88e-6	28/2/0.46/6.06e-6
X2	20000	29/19/0.10/8.94e-6	29/19/0.09/8.94e-6	27/2/0.09/6.14e-6
	80000	32/21/0.46/4.17e-6	32/21/0.43/4.17e-6	27/2/0.34/9.46e-6
	100000	33/22/0.59/3.23e-6	33/22/0.62/3.23e-6	28/2/0.43/5.62e-6
X3	20000	31/20/0.11/2.64e-6	31/20/0.10/2.64e-6	27/2/0.08/7.11e-6
	80000	32/20/0.48/1.30e-6	32/20/0.47/1.30e-6	28/2/0.36/6.07e-6
	100000	31/19/0.53/3.92e-6	31/19/0.58/3.92e-6	28/2/0.45/6.51e-6
X4	20000	28/18/0.11/6.36e-6	28/18/0.12/6.36e-6	27/2/0.08/7.78e-6
	80000	29/17/0.38/6.01e-6	29/17/0.36/6.01e-6	28/2/0.31/6.67e-6
	100000	30/18/0.61/3.12e-6	30/18/0.57/3.12e-6	28/2/0.44/7.15e-6
X5	20000	29/18/0.10/7.13e-6	29/18/0.13/7.13e-6	27/2/0.09/8.79e-6
	80000	31/19/0.46/4.92e-6	31/19/0.58/4.92e-6	28/2/0.35/7.54e-6
	100000	31/19/0.60/3.66e-6	31/19/0.66/3.66e-6	28/2/0.46/8.09e-6
X6	20000	32/21/0.13/4.31e-6	32/21/0.14/4.31e-6	27/2/0.11/6.93e-6
	80000	31/19/0.43/3.25e-6	31/19/0.54/3.25e-6	28/2/0.57/5.92e-6
	100000	31/19/0.65/4.42e-6	31/19/0.59/4.42e-6	28/2/0.43/6.35e-6
X7	20000	33/21/0.13/3.92e-6	33/21/0.13/3.92e-6	30/2/0.09/8.04e-6
	80000	33/20/0.50/1.85e-6	33/20/0.50/1.85e-6	31/2/0.38/6.91e-6
	100000	33/19/0.61/4.15e-6	33/19/0.59/4.15e-6	31/2/0.48/7.40e-6

Problem 5.2. *The problem is from [4]. The mapping* F *is taken as* $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T$, *where*

$$\begin{aligned} f_1(x) &= x_1(2x_1^2 + 2x_2^2) - 1, \\ f_i(x) &= x_i(x_{i-1}^2 + 2x_i^2 + x_{i+1}^2) - 1, \ \ i = 2, 3, \cdots, n-1, \\ f_n(x) &= x_n(2x_{n-1}^2 + 2x_n^2) - 1, \end{aligned}$$

and $\Omega = \mathcal{R}^n_+$.

Problem 5.3. *The problem is from [12]. Let* $\Omega = R^n_+$ *and the mapping* F *be* $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T$, *where where* $f_i(x) = expx_i - 1, i = 1, 2, \cdots, n$, *and* $\Omega = R^n_+$.

Table 2. The experiment results of Problem 5.2 for three algorithms

		A1	A2	Algorithm 2.1
Dim	IP	NI/NF/CPU/FV	NI/NF/CPU/FV	NI/NF/CPU/FV
5000	X1	14/26/0.02/2.59e-6	14/26/0.02/2.59e-6	12/4/0.02/1.96e-6
	X2	14/24/0.02/4.32e-6	14/24/0.02/4.32e-6	11/4/0.02/4.32e-6
	X3	15/31/0.01/3.68e-6	15/31/0.01/3.60e-6	18/8/0.02/2.92e-6
	X4	14/25/0.01/7.04e-6	14/25/0.01/7.04e-6	11/5/0.02/4.59e-6
	X5	32/57/0.02/6.02e-6	31/55/0.01/8.33e-6	20/4/0.01/4.62e-6
20000	X1	15/27/0.03/2.71e-6	15/27/0.04/2.71e-6	12/4/0.03/3.91e-6
	X2	14/24/0.03/8.65e-6	14/24/0.03/8.65e-6	11/4/0.02/8.63e-6
	X3	15/31/0.02/7.20e-6	15/31/0.02/7.20e-6	18/8/0.03/5.83e-6
	X4	13/21/0.03/3.30e-6	13/21/0.02/3.30e-6	11/5/0.02/9.17e-6
	X5	33/59/0.05/5.78e-6	32/57/0.04/7.92e-6	20/4/0.02/9.24e-6
30000	X1	15/27/0.04/3.32e-6	15/27/0.05/3.32e-6	12/4/0.03/4.79e-6
	X2	15/26/0.04/2.52e-6	15/26/0.04/2.52e-6	12/4/0.03/1.83e-6
	X3	15/31/0.05/8.82e-6	15/31/0.04/8.82e-6	18/8/0.04/7.15e-6
	X4	13/21/0.04/4.04e-6	13/21/0.04/4.04e-6	12/5/0.03/1.95e-6
	X5	39/69/0.09/7.51e-6	37/65/0.08/7.21e-6	21/4/0.04/4.64e-6
100000	X1	15/27/0.09/6.06e-6	15/27/0.11/6.06e-6	12/4/0.07/8.75e-6
	X2	15/24/0.09/3.67e-6	15/24/0.10/3.67e-6	12/4/0.07/3.35e-6
	X3	16/33/0.12/6.99e-6	16/33/0.13/6.99e-6	19/8/0.11/2.26e-6
	X4	13/21/0.09/7.37e-6	13/21/0.09/7.37e-6	12/5/0.07/3.56e-6
	X5	44/77/0.29/6.79e-6	40/69/0.25/7.37e-6	21/4/0.10/8.48e-6

Problem 5.4. *The problem is from [21]. The mapping F is taken as $F(x) = (f_1(x), f_2(x), \cdots, f_n(x))^T$, where*

$$f_1(x) = 2x_1 - x_2 + \exp x_1 - 1,$$
$$f_i(x) = -x_{i-1} + 2x_i - x_{i+1} + \exp x_i - 1, \ i = 2, 3, \cdots, n-1,$$
$$f_n(x) = -x_{n-1} + 2x_n + \exp x_n - 1,$$

and $\Omega = \mathcal{R}^n_+$.

For testing Problem 5.1, the initial point are showed as follows:

$$X_1 = (1, 1, \cdots, 1)^T, \quad X_2 = (\frac{1}{n}, \frac{2}{n}, \cdots, 1)^T, \quad X_3 = (\frac{1}{2}, \frac{1}{2^2}, \cdots, \frac{1}{2^n})^T,$$
$$X_4 = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \cdots, 0)^T, \quad X_5 = (1, \frac{1}{2}, \cdots, \frac{1}{n})^T, \quad X_6 = (0, 0, \cdots, 0)^T,$$
$$X_7 = (10, 10, \cdots, 10)^T.$$

The detailed results are listed in Table 1.

For testing Problem 5.2, the initial point are showed as follows:

$$X_1 = (1, 1, \cdots, 1)^T, \quad X_2 = (0, 0, \cdots, 0)^T, \quad X_3 = (10, 10, \cdots, 10)^T,$$

Table 3. The experiment results of Problem 5.3 for three algorithms

		A1	A2	Algorithm 2.1
IP	Dim	NI/NF/CPU/FV	NI/NF/CPU/FV	NI/NF/CPU/FV
X1	5000	21/22/0.03/8.78e-6	21/22/0.02/8.78e-6	17/2/0.01/5.51e-6
	30000	21/22/0.05/8.78e-6	21/22/0.05/8.78e-6	17/2/0.04/5.51e-6
	80000	21/22/0.10/8.78e-6	21/22/0.10/8.78e-6	17/2/0.07/5.51e-6
	100000	21/22/0.12/8.78e-6	21/22/0.13/8.78e-6	17/2/0.08/5.51e-6
X2	5000	22/23/0.01/8.69e-6	22/23/0.02/8.69e-6	19/2/0.02/5.16e-6
	30000	22/23/0.06/8.69e-6	22/23/0.05/8.69e-6	19/2/0.04/5.16e-6
	80000	22/23/0.11/8.69e-6	22/23/0.12/8.69e-6	19/2/0.09/5.16e-6
	100000	22/23/0.15/8.69e-6	22/23/0.14/8.69e-6	19/2/0.11/5.16e-6
X3	5000	28/25/0.03/8.07e-6	28/25/0.02/8.07e-6	23/2/0.01/8.11e-6
	30000	28/24/0.07/9.49e-6	28/24/0.07/9.49e-6	25/2/0.05/5.05e-6
	80000	29/25/0.16/9.30e-6	29/25/0.17/9.30e-6	25/2/0.13/8.25e-6
	100000	28/23/0.19/6.08e-6	28/23/0.20/6.08e-6	25/2/0.16/9.22e-6
X4	5000	26/25/0.02/8.39e-6	26/25/0.02/8.39e-6	22/2/0.02/5.28e-6
	30000	26/24/0.06/9.51e-6	26/24/0.05/9.51e-6	23/2/0.05/6.46e-6
	80000	27/25/0.14/9.31e-6	27/25/0.16/9.31e-6	24/2/0.11/5.28e-6
	100000	25/22/0.18/9.56e-6	25/22/0.17/9.56e-6	24/2/0.14/5.90e-6
X5	5000	34/42/0.02/6.51e-6	34/42/0.03/6.51e-6	31/10/0.02/6.94e-6
	30000	36/41/0.10/9.23e-6	36/41/0.09/9.23e-6	32/10/0.08/8.50e-6
	80000	23/25/0.16/7.02e-6	23/25/0.16/7.02e-6	33/10/0.18/6.94e-6
	100000	23/25/0.18/7.84e-6	23/25/0.20/7.84e-6	33/10/0.21/7.76e-6
X6	5000	30/29/0.02/6.95e-6	30/29/0.02/6.95e-6	22/2/0.01/6.46e-6
	30000	31/27/0.06/8.82e-6	31/27/0.06/8.82e-6	23/2/0.06/7.92e-6
	80000	32/27/0.18/6.85e-6	32/27/0.18/6.85e-5	24/2/0.11/6.46e-6
	100000	32/27/0.23/7.65e-6	32/27/0.21/7.65e-6	24/2/0.16/7.23e-6

$$X_4 = (\frac{1}{3}, \frac{1}{3}, \cdots, \frac{1}{3})^T, \quad X_5 = (-\frac{1}{4}, \frac{1}{4}, \cdots, (-1)^i \frac{1}{4})^T.$$

The detailed results are listed in Table 2.

For testing Problem 5.3, the initial point are showed as follows:

$$X_1 = (\frac{1}{2}, \frac{1}{2^2}, \cdots, \frac{1}{2^n})^T, \quad X_2 = (1, \frac{1}{2}, \cdots, \frac{1}{n})^T, \quad X_3 = (\frac{1}{n}, \frac{2}{n}, \cdots, 1)^T,$$
$$X_4 = (-\frac{1}{4}, \frac{1}{4}, \cdots, (-1)^i \frac{1}{4})^T, \quad X_5 = (10, 10, \cdots, 10)^T, \quad X_6 = (\frac{1}{3}, \frac{1}{3}, \cdots, \frac{1}{3})^T.$$

The detailed results are showed in Table 3.

For testing Problem 5.4, the dimension of X is 20000, and the initial points are showed as follows:

$$X_1 = (\frac{1}{3}, \frac{1}{3}, \cdots, \frac{1}{3})^T, \quad X_2 = (\frac{1}{n}, \frac{2}{n}, \cdots, 1)^T, \quad X_3 = (\frac{1}{2}, \frac{1}{2^2}, \cdots, \frac{1}{2^n})^T,$$

$$X_4 = (-\frac{1}{4}, \frac{1}{4}, \cdots, (-1)^i \frac{1}{4})^T, \quad X_5 = (1, \frac{1}{2}, \cdots, \frac{1}{n})^T, \quad X_6 = (1, 1, \cdots, 1)^T,$$
$$X_7 = (1 - \frac{1}{n}, 1 - \frac{2}{n}, \cdots, 0)^T, \quad X_8 = (\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n})^T, \quad X_9 = (\frac{1}{5}, \frac{1}{5}, \cdots, \frac{1}{5})^T,$$
$$X_{10} = (10, 10, \cdots, 10)^T, \quad X_{11} = (\frac{1}{3}, \frac{1}{3^2}, \cdots, \frac{1}{3^n})^T, \quad X_{12} = (1, \frac{1}{2^2}, \cdots, \frac{1}{n^2})^T,$$
$$X_{13} = (0.1, 0.1, \cdots, 0.1)^T, \quad X_{14} = (\frac{1}{n} - 1, \frac{2}{n} - 1, \cdots, 0)^T, \quad X_{15} = (5, 5, \cdots, 5)^T.$$

The detailed results are showed in Table 4.

Table 4. The experiment results of Problem 5.4 for three algorithms

	A1	A2	Algorithm 2.1
IP	NI/NF/CPU/FV	NI/NF/CPU/FV	NI/NF/CPU/FV
X1	137/204/0.33/9.33e-6	137/204/0.33/9.33e-6	46/2/0.07/7.64e-6
X2	131/214/0.32/8.42e-6	132/215/0.31/8.89e-6	43/3/0.08/4.82e-6
X3	74/120/0.12/8.23e-6	74/120/0.13/8.23e-6	22/3/0.05/5.69e-6
X4	67/118/0.20/9.57e-6	67/118/0.19/9.57e-6	33/3/0.08/8.60e-6
X5	78/131/0.18/8.81e-6	78/131/0.15/8.81e-6	28/3/0.04/6.92e-6
X6	107/202/0.28/9.35e-6	132/228/0.30/8.92e-6	49/3/0.09/6.22e-6
X7	131/214/0.29/8.43e-6	132/215/0.29/8.94e-6	43/3/0.06/5.16e-6
X8	51/76/0.12/9.78e-6	5151/76/0.11/9.78e-6	18/2/0.04/7.09e-6
X9	134/197/0.36/9.89e-6	134/197/0.27/9.89e-6	40/2/0.06/7.12e-6
X10	149/256/0.35/8.55e-6	149/256/0.36/8.55e-6	53/10/0.08/6.71e-6
X11	54/95/0.09/8.69e-6	54/95/0.09/8.69e-6	22/3/0.04/6.20e-6
X12	56/98/0.10/9.92e-6	56/98/0.09/9.92e-6	25/3/0.03/6.90e-6
X13	128/190/0.29/8.60e-6	128/190/0.28/8.60e-6	38/2/0.05/3.74e-6
X14	48/88/0.12/9.01e-6	48/88/0.10/9.01e-6	32/2/0.06/5.86e-6
X15	133/215/0.31/8.26e-6	132/212/0.27/8.20e-6	44/6/0.08/3.08e-6

In Table 5, the initial points are generated by the command rand (0,1) in Matlab for Problem 5.1-5.4.

In Tables 1-5, "Dim" represents the problem dimension, and "IP" represents the initial points. "P" represents the testing problem,"FV" represents the final value of $||F_k||$. From Tables 1-5, we can see that our algorithm outperforms the compared algorithms above mentioned in term of "NI", "NF", "CPU" for testing problems. For Problem 5.1 and Problem 5.3, the reported results in Table 1 and 3 showed that our algorithm is insensitive to the initial points and dimensions. For Problem 5.2, the reported results in Table 2 illustrated that our algorithm is sensitive to the initial points and insensitive to the dimensions. For Problem 5.4, the reported results in Table 4 that our algorithm is sensitive to the initial points. From Tables 1-5, we noticed that our algorithm is far better than compared algorithms in terms of NF, and so we didn't draw the performance curve proposed by Dolan and Morè [3] of NF. Then, we obtain Figures 1 and 2. We can see from Figures 1 and 2 that our algorithm is more competitive with compared algorithms in term of CPU and NI.

Table 5. The experiment results of Problem 5.1-5.4 for three algorithms

		A1	A2	Algorithm 2.1
P	Dim	NI/NF/CPU/FV	NI/NF/CPU/FV	NI/NF/CPU/FV
5.1	20000	27/17/0.09/4.50e-6	27/17/0.10/5.39e-6	27/2/0.07/8.24e-6
		28/18/0.11/9.73e-6	28/18/0.10/4.14e-6	27/2/0.08/8.22e-6
	80000	32/21/0.52/3.94e-6	32/20/0.40/7.10e-6	28/2/0.29/6.51e-6
		30/19/0.39/4.69e-6	31/20/0.49/6.87e-6	28/2/0.30/7.05e-6
	100000	33/21/0.54/1.83e-6	32/20/0.60/5.49e-6	28/2/0.46/6.88e-6
		32/20/0.51/2.20e-6	31/19/0.47/5.84e-6	28/2/0.38/7.68e-6
5.2	20000	26/48/0.04/8.08e-6	25/45/0.05/6.26e-6	28/4/0.03/7.25e-6
		27/51/0.04/8.93e-6	24/42/0.04/3.73e-6	28/4/0.02/8.79e-6
	80000	35/55/0.15/6.77e-6	36/57/0.18/7.90e-6	29/4/0.09/8.65e-6
		36/57/0.18/5.05e-6	35/55/0.15/8.84e-6	29/4/0.09/9.24e-6
	100000	36/57/0.23/7.83e-6	36/57/0.20/4.60e-6	29/4/0.14/9.53e-6
		37/59/0.21/4.66e-6	35/55/0.19/9.82e-6	30/4/0.11/5.58e-6
5.3	20000	28/24/0.04/7.77e-6	28/24/0.04/7.73e-6	24/2/0.04/8.19e-6
		28/24/0.04/7.81e-6	28/24/0.04/7.71e-6	24/2/0.04/8.16e-6
	80000	29/25/0.16/9.30e-6	29/25/0.17/9.32e-6	25/2/0.13/8.24e-6
		29/25/0.17/9.28e-6	29/25/0.18/9.33e-6	25/2/0.14/8.23e-6
	100000	28/23/0.20/6.11e-6	28/23/0.20/6.05e-6	25/2/0.16/9.20e-6
		28/23/0.19/6.10e-6	28/23/0.19/6.09e-6	25/2/0.15/9.22e-6
5.4	20000	128/198/0.28/9.89e-6	125/196/0.29/8.24e-6	47/3/0.08/5.18e-6
		132/203/0.30/8.14e-6	136/207/0.31/8.28e-6	42/3/0.06/6.38e-6
	80000	134/199/1.05/9.84e-6	132/199/1.00/8.28e-6	43/4/0.28/9.22e-6
		137/205/1.04/8.50e-6	136/204/1.22/8.54e-6	46/3/0.24/7.39e-6
	100000	137/204/1.25/9.14e-6	135/202/1.30/9.23e-6	46/3/0.34/7.11e-6
		133/197/1.22/8.34e-6	139/206/1.30/9.39e-6	44/3/0.28/8.70e-6

6. Applications in Compressive Sensing

In this section, compared with A1 [4] and A2 [4], we apply our algorithm for dealing with a typical compressive sensing scenario in terms of MSE, NI and CPU time.

6.1. Compressive Sensing

Compressive sensing is a signal recovery technique for efficiently reconstructing a sparse signal. It has been proved that this technique can effectively recover a sparse signal from some sampling measurements whose number can be dramatically less than the original signal, by solving the underdetermined linear systems.

Following [2], we consider the problem of recovering an unknown vector $x_0 \in \mathcal{R}^n$ from incomplete and contaminated observations:

$$b = Ax_0 + e, \tag{6.1}$$

where $b \in \mathcal{R}^k$ is the observed data, $A \in \mathcal{R}^{k\times n}(k << n)$, and $e \in \mathcal{R}^k$ is an error term.

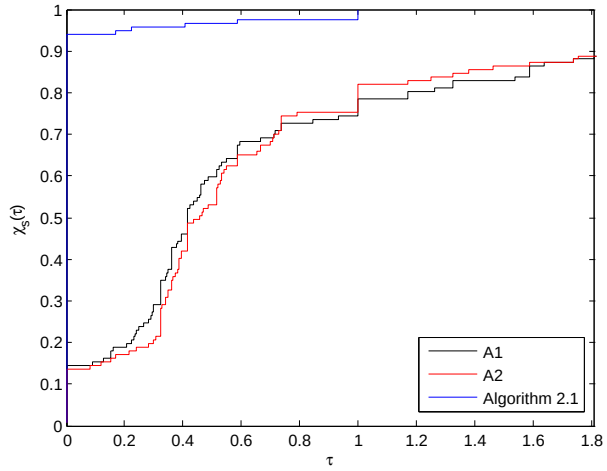

Figure 1. Performance profiles of three algorithms with respect to the CPU time.

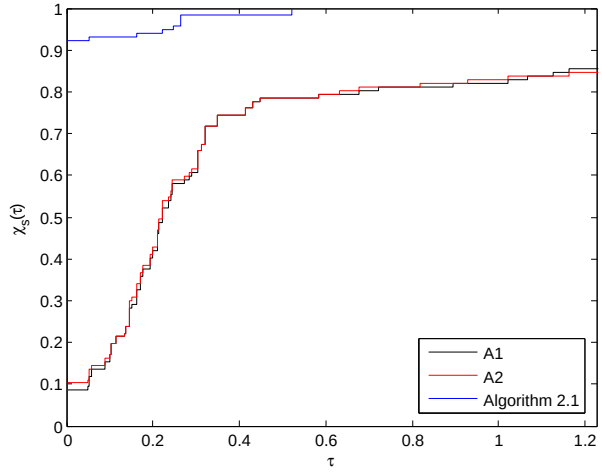

Figure 2. Performance profiles of three algorithms with respect to the number of iteration.

($k << n$) means that the number of sampling measurements can be dramatically less than the original signal.

A regularization technique is used for overcoming the ill-conditioned nature of matrix A in (6.1) when trying to recover the original signal x_0 from noiseless observations $b = Ax$ or noisy observations $b = Ax + e$. Therefore, to find the sparse solutions x_0 of (6.1) is to solve the following convex unconstrained optimization problem:

$$\min_{x\in\mathcal{R}^n} \tau\|x\|_1 + \frac{1}{2}\|b - Ax\|_2^2, \tag{6.2}$$

where $\tau > 0$ is a parameter, $\|\nu\|_1$ and $\|\nu\|_2$ denote the l_1-norm and l_2-norm of $\nu \in \mathcal{R}^n$, respectively.

Problem (6.2) can be transformed as a convex quadratic program problem, which has been done in [4,5,7,13,19,20]. Splitting $x \in \mathcal{R}^n$ into positive and negative parts,

we have

$$x = u - v,\ u \geq 0,\ v \geq 0, \tag{6.3}$$

where $u_i = \max\{0, x_i\}$ and $v_i = \max\{-x_i, 0\}$ with $i \in \{1, 2, \ldots, n\}$. By l_1-norm, the $\|x\|_1$ can be rewritten as

$$\|x\|_1 = e_n^T u + e_n^T v, \tag{6.4}$$

where $e_n = (1, 1, \ldots, 1)^T \in \mathcal{R}^n$.

Combining (6.3) and (6.4), Problem (6.2) can be rewritten as the bound-constrained quadratic program as follows:

$$\min_{u,v} \frac{1}{2}\|b - A(u - v)\|_2^2 + \tau e_n^T u + \tau e_n^T v, \tag{6.5}$$

with $u \geq 0$, $v \geq 0$.

With the help of l_2-norm defined by scalar product, Problem (6.5) can be rewritten in following form:

$$\min_{z \geq 0} \frac{1}{2} z^T H z + c^T z, \tag{6.6}$$

where

$$z = \begin{bmatrix} u \\ v \end{bmatrix}, \quad c = \begin{bmatrix} \tau e_n - A^T b \\ \tau e_n + A^T b \end{bmatrix}, \quad H = \begin{bmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{bmatrix}.$$

It is obvious that matrix H is positive semi-definite. So, Problem (6.6) is a convex quadratic program problem. It was proved in [17] that z is a solution of Problem (6.6) if and only if z is a solution of the equations:

$$F(z) = \min\{z, Hz + c\} = 0,\ z \geq 0, \tag{6.7}$$

where the function F is vector valued, and the "min" is defined as componentwise minimum.

From [9, Lemma 3] and [17, Lemma 2.2], we know that $F : \mathcal{R}^{2n} \to \mathcal{R}^{2n}$ is Lipschitz continuous and monotone. Hence, Equation (6.7) can be effectively solved by the proposed algorithms.

6.2. Numerical results

In this section, the parameters of our algorithm are chosen as: $\sigma = 1 \times 10^{-4}$, $\rho = 0.6$, $r = 1$, $\xi = 10$. The parameters of A1 and A2 are taken from [4].

In our experiments, the main goal is to recover a sparse signal of size n from k sampling measurements with Gaussian noise, in which the number of samples is dramatically smaller than the size of the original signal. The quality of restoration is measured by the mean of squared error (MSE):

$$MSE = \frac{1}{n}\|x_0 - x^*\|^2,$$

where x_0 is the original signal and x^* is the restored signal.

We select $n = 2^{12}$ and $k = 2^{10}$ for the present experiments. The original signal x_0 contains 2^8 randomly nonzero elements. The random matrix A is the Gaussian

matrix generated by the commend $rand(n, k)$ in Matlab. The measurements b is obtained by

$$b = Ax_0 + e,$$

where e is Gaussian noise with $N(0, 10^{-4})$. In order to compare three algorithms in relatively fair condition, we run the three codes from the same initial points and use the same continuation technique for the selection of the parameter τ in (6.8). The iteration starts with $z_0 = (u_0, v_0)^T$ with $u_0 = v_0 = A^T b$, and the termination condition is defined as

$$\frac{\|f(x_k) - f(x_{k-1})\|}{\|f(x_{k-1})\|} < 10^{-5},$$

where

$$f(x) = \tau\|x\|_1 + \frac{1}{2}\|b - Ax\|_2^2 \tag{6.8}$$

is the objective function.

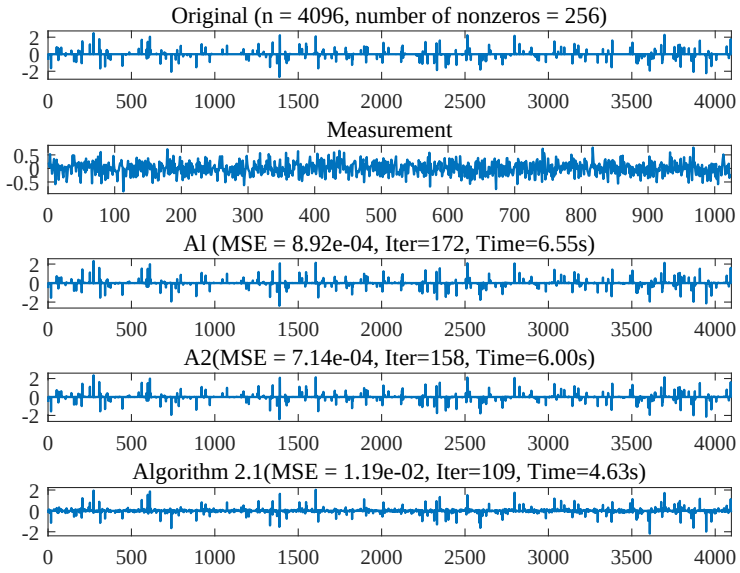

Figure 3. From top to bottom: the original signal, the measurements, and the reconstructed signals by three algorithms.

Fig. 3 shows the original sparse signal x_0, the measurements b, and the reconstructed signal x^* by three algorithms. Furthermore, to illustrate the performance of three algorithms, we draw four figures to show their change trends of MSE(Fig. 4) and objective function values(Fig. 5) in terms of iteration number and CPU time (in second), respectively. From Figures 3-5, we can observe that the original signal was reconstructed almost exactly from the measurements by three algorithms, while our algorithm require less iteration numbers and CPU time than the other algorithms. But, we could notice the MSE value of our algorithm is a little bigger than the others.

For a fair comparison, we also did extra ten experiments where ten original signals x_0 were randomly generated and the results are shown in Table 6. Table 6 shows that our algorithm requires less iteration numbers and CPU time(s) than

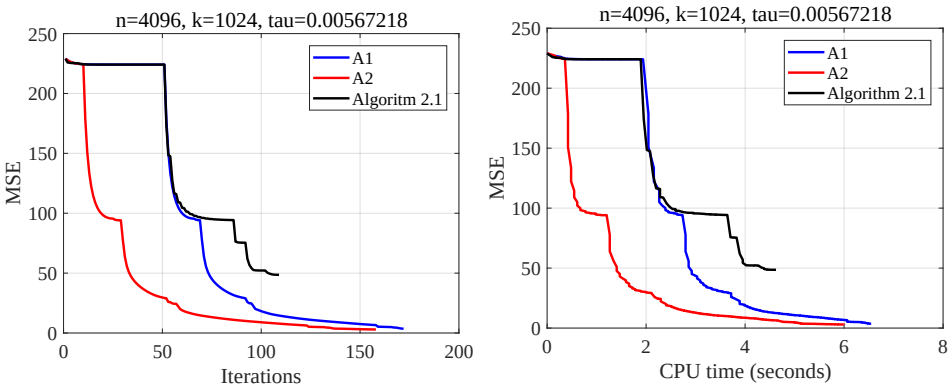

Figure 4. Plots of MSE versus iteration and time.

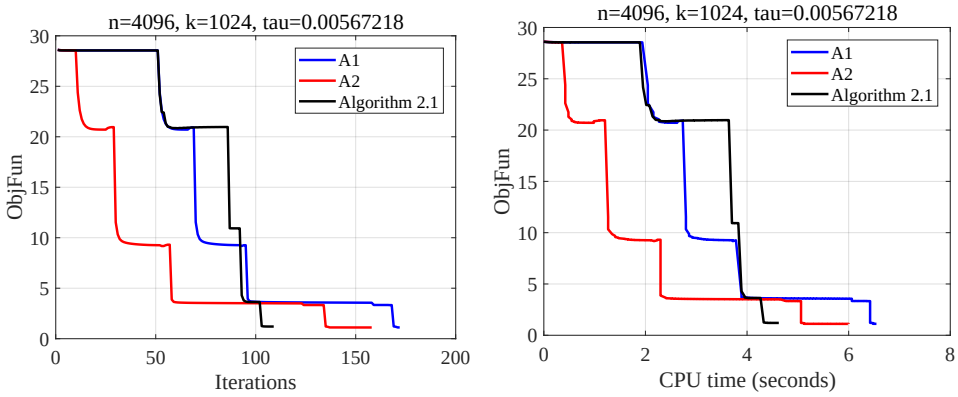

Figure 5. Plots of objective function value versus iteration and time.

others for any of the ten original signals, but the MSE value of our algorithm is a little bigger than the others.

7. Concluding remarks

In this paper, we developed a three-term CG-type projection method for dealing with monotone nonlinear equations with convex constraints. The novelty of the proposed method is that the method satisfies the quasi-Newton equation, and the parameters are obtained by simplifying the maximum eigenvalue of the relative matrices. We establish the global convergence and R-linear convergence are obtained under mild assumptions. Furthermore, our algorithm can be also used for dealing with compressive sensing efficiently. However there is an important issue worth studying in the future research, the issue is how to improve the parameters to obtain better the numerical performance of MSE value.

Table 6. Ten experiment results for three algorithms

	A1	A2	Algorithm 2.1
	MSE/NI/CPU	MSE/NI/CPU	MSE/NI/CPU
	1.19e-3/156/5.02	1.19e-3/116/3.66	9.87e-3/108/3.53
	8.03e-4/186/6.03	8.10e-4/148/4.39	9.27e-3/118/3.72
	7.31e-4/175/5.28	7.31e-4/136/4.16	6.00e-3/129/4.02
	4.32e-4/182/5.63	4.32e-4/142/4.22	7.46e-3/127/3.89
	6.60e-4/179/5.70	6.60e-4/139/4.80	9.64e-3/122/3.91
	1.19e-3/184/5.84	1.19e-3/144/4.66	9.96e-3/127/4.22
	1.48e-4/195/6.11	1.58e-4/151/4.75	7.88e-3/121/4.14
	3.36e-4/197/5.98	3.91e-4/139/4.14	5.59e-3/129/4.08
	5.68e-4/188/5.45	5.68e-4/147/4.44	9.22e-3/114/3.56
	1.86e-4/213/6.48	2.13e-4/171/5.27	8.95e-3/124/4.23
	5.70e-4/166/5.67	6.31e-4/119/3.50	7.56e-3/110/3.20
	1.49e-3/159/4.78	1.25e-3/148/4.81	9.13e-3/119/3.80
Average	6.92e-4/181.7/5.66	6.85e-4/141.7/4.40	8.38e-3/120.7/3.86

References

[1] N. Andrei, *A double parameter scaled BFGS method for unconstrained optimization,* J. Comput. Appl. Math., 2018, 332, 26–44.

[2] E. J. Candes, J. K. Romberg and T. Tao, *Stable signal recovery from incomplete and inaccurate measurements,* Pure Appl. Math., 2006, 59, 1207–1223.

[3] E. D. Dolan and J. J. Morè, *Benchmarking Optimization Sofeware with performance profiles,* Math. Program, 2002, 91, 201–213.

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

[5] P. Gao, T. Wang, X. Liu and Y. Wu, *An effificient three-term conjugate gradient-based algorithm involving spectral quotient for solving convex constrained monotone nonlinear equations with applications,* Comput. Appl. Math., 2022. Doi: 10.1007/s40314-022-01796-4.

[6] P. Gao and C. He, *An efficient three-term conjugate gradient method for nonlinear monotone equations with convex constraints,* Calcolo, 2018. Doi:10.1007/s10092-018-0291-2.

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

[8] J. Liu, Z. Lu, J. Xu, S. Wu and Z. Tu, *An efficient projection-based algorithm without Lipschitz continuity for large-scale nonlinear pseudo-monotone equations,* J. Comput. Appl. Math., 2022, 403, 113822.

[9] J. Pang, *inext newton methods for the nonlinear complementary problem,* Math. Program, 1986, 36, 54–71.

[10] W. Sun and X. Yuan, *Optimization Theory and methods,* Nonlinear Program, Springer Science+business Media, New York, 2005.

[11] M. V. Solodov and B. F. Svaiter, *Reformulation: nonsmooth, piecewise smooth, semismooth and smoothing methods,* In: Fukushima, M., Qi, L.(eds.) A globally convergent inexact Newton method for systems of monotone equations, Dordrecht: Kluwer Academic Publishers, 1998, 355–369.

[12] C. Wang, Y. Wang and C. Xu, *A projection method for a system of nonlinear monotone equations with convex constraints,* Math. Methods Oper. Res., 2007, 66, 33–46.

[13] M. Y. Waziri, K. Ahmed, A. S. Halilu and J. Sabi'u, *Two new Hager-Zhang iterative schemes with improved parameter choices for monotone nonlinear systems and their applications in compressed sensing,* Rairo-Oper. Res., 2021. Doi: 10.1051/ro/2021190.

[14] M. Y. Waziri, K. Ahmed and J. Sabi'u, *A family of Hager-Zhang conjugate gradient methods for system of monotone nonlinear equations,* Appl. Math. Comput., 2019, 361, 645–660.

[15] M. Y. Waziri, K. Ahmed and J. Sabi'u, *A Dai-Liao conjugate gradient method via modified secant equation for system of nonlinear equations,* Arab. J. Math., 2020, 9, 443–457.

[16] M. Y. Waziri, K. Ahmed and J. Sabi'u, *Descent Perry conjugate gradient methods for systems of monotone nonlinear equations,* Numer. Algorithems, 2020, 85, 763–785.

[17] Y. Xiao, Q. Wang and Q. Hu, *Non-smooth equations based method for l_1-norm problems with applications to compressed sensing,* Nonlinear Anal., 2011, 74, 3570–3577.

[18] S. Yao and L. Ning, *An adaptive three-term conjugate gradient method based on self-scaling memoryless BFGS matrix,* J. Comput. Appl. Math., 2018, 332, 72–85.

[19] J. Yin, J. Jian and X. Jiang, *A hybrid three-term conjugate gradient projection method for constrained nonlinear monotone equations with applications,* Numer. Algorithms, 2021, 88, 389–418.

[20] J. Yin, J. Jian and X. Jiang, *A generalized hybrid CGPM-based algorithm for solving large-scale convex constrained equations with applications to image restoration,* J. Comput. Appl. Math., 2021, 391, 113423.

[21] W. Zhou and D. Li, *A globally convergent BFGS method for nonlinear monotone equations without any merit functions,* Math. Comput., 2008, 77, 2231–2240.