Bangmod Int. **J.** **M**ath. **&** **C**omp. **S**ci.
Vol. 5, No. 1-2, 2019; Pages 1 - 26
http://bangmod-jmcs.kmutt.ac.th/
ISSN : 2408-154X

A modified self-adaptive conjugate gradient method for solving convex constrained monotone nonlinear equations with applications to signal recovery problems

Auwal Bala Abubakar[1,3], **Poom Kumam**[1,2,*] **and Aliyu Muhammed Awwal**[2,4]
[1] *KMUTTFixed Point Research Laboratory, Room SCL 802 Fixed Point Laboratory, Science Laboratory Building, Department of Mathematics, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 126 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand.*

[2] *KMUTT-Fixed Point Theory and Applications Research Group, Theoretical and Computational Science Center (TaCS), Science Laboratory Building, Faculty of Science, King Mongkut's University of Technology Thonburi (KMUTT), 26 Pracha-Uthit Road, Bang Mod, Thrung Khru, Bangkok 10140, Thailand.*

[3] *Department of Mathematical Sciences, Faculty of Physical Sciences, Bayero University, Kano. Kano, Nigeria.*

[4] *Department of Mathematics, Faculty of Science, Gombe State University, Gombe, Nigeria.*

E-mails: auwal201@gmail.com (ABA); poom.kum@kmutt.ac.th (PK) and aliyumagsu@gmail.com (AMA)

**Corresponding author.*

Abstract In this article, we propose a modified self-adaptive conjugate gradient algorithm for handling nonlinear monotone equations with the constraints being convex. Under some nice conditions, the global convergence of the method was established. Numerical examples reported show that the method is promising and efficient for solving monotone nonlinear equations. In addition, we applied the proposed algorithm to solve sparse signal reconstruction problems.

MSC: 65K05, 90C52, 90C56, 52A20.
Keywords: Non-linear equations, Conjugate gradient method, Projection method, Convex constraints, signal reconstruction problem.

Submission date: 10 May 2019 / Acceptance date: 5 December 2019 / Available online 31 December 2019

Published by Theoretical and Computational Science Center (TaCS),
King Mongkut's University of Technology Thonburi (KMUTT)

1. Introduction

Suppose Ω is a nonempty, closed and convex subset of R^n, F a continuous function from R^n to R^n. A constrained nonlinear monotone equation involves finding a point $x \in \Omega$, such that

$$F(x) = 0. \tag{1.1}$$

Many algorithms have been proposed in literature to solve nonlinear constrained equations, some of which are the trust region [3] and the Levenberg-Marquardt method [6]. However, the need for these methods to compute and store matrix in every iteration, make them unsuitable for solving large-scale nonlinear equations.

Conjugate gradient (CG) methods is an iterative method developed for handling unconstrained optimization problem [1, 9, 11, 15, 16, 20, 27, 28]. CG methods does not require matrix storage, which makes it one of the efficient methods for handling large-scale unconstrained optimization problems. Moreover, generating a descent direction does not always hold based on the secant conditions. In order to obtain a descent direction, Narushima et al. [15] and Zhang et al. [28] proposed three term CG methods, which always generate a descent direction, and established the convergence of the methods under some suitable conditions. Also in [16], Narushima proposed a smoothing CG algorithm, which combine the smoothing approach with the Polak–Ribière–Polyak CG methods in [27], to handle unconstrained non-smooth equations. The convergence of the method was established under some mild conditions.

Methods for solving unconstrained problems sometimes become less useful, as in many practical applications, such as equilibrium problems, the solution of the unconstrained problem may lie outside the constrained set Ω. This reason made researchers shift their attention to the constrained case (1.1). In the last few years, many kinds of algorithms for solving nonlinear monotone equations with convex constrained set Ω have been developed and one of the popular is the projection method. For example, in [23] Wang et al. proposed a projection method for solving systems of monotone nonlinear equations with convex constraints. The method was based on the inexact Newton backtracking technique and the direction was obtained by minimization of a linear system together with the constrained condition at each iteration. Also, in [22] Wang et al. presented a modification of the method in [18], and the global convergence as well as the super-linear rate of convergence were established under same conditions in [23]. However, the direction of the methods in [18, 22] were determined by minimization of linear equations at each step. In trying to avoid solving the linear equation to obtain the direction at each step, Xiao and Zhu [26] proposed a projected CG methods, which combines the CG-DESCENT method in [10] and the projection technique by Solodov and Svaiter [19]. In [14], a modification of the method in [26] was proposed by Liu and Li. The advantage of this modification was that it improves the numerical performance of the method in [26] and still retains its nice properties. Furthermore, Wang et al. [24] proposed a self-adaptive three-term CG methods for solving constrained nonlinear monotone equations. The method can be viewed as combination of the CG methods, the projection method and the self-adaptive method.

Motivated by the above methods, we propose a modification of the method in [24] for solving nonlinear monotone equations with convex constraints. The modification improves the numerical performance of the method in [24] and still inherits its nice properties. The difference between the two methods is that y_{k-1} in [24] is replaced by w_{k-1} (More details can be found in the next section). Under appropriate conditions, the global convergence of the proposed method is established. Numerical results presented show that the proposed method is efficient and promising compared to some similar existing algorithms.

The remaining part of this paper is organized as follows. In section 2, we state some preliminaries and then present the algorithm. The global convergence of the proposed method is proved in section 3. In section 4, we report some numerical experiments to show its performance in solving nonlinear monotone equations with convex constraints, and lastly apply it to solve some signal recovery problems.

2. Preliminaries and algorithm

This section gives some basic concepts and properties of the projection mapping as well as some assumptions. $\|\cdot\|$ denotes the Euclidean norm throughout the paper.

Definition 2.1. Let $\Omega \subset R^n$ be a nonempty closed convex set. Then for any $x \in R^n$, its orthogonal projection onto Ω, denoted by $P_\Omega(x)$, is defined by

$$P_\Omega(x) = \arg\min\{\|x-y\| \ : y \in \Omega\}.$$

The following lemma provides us with some well-known properties of the projection mapping.

Lemma 2.2. *[24] Let $\Omega \subset R^n$ be a nonempty, closed and convex set. Then the following statements are true:*

1. $(x - P_\Omega(x))^T(P_\Omega(x) - z) \geq 0, \quad \forall x \in R^n,\ z \in \Omega.$
2. $\|P_\Omega(x) - P_\Omega(y)\| \leq \|x - y\|, \quad \forall x, y \in R^n.$
3. $\|P_\Omega(x) - z\|^2 \leq \|x - z\|^2 - \|x - P_\Omega(x)\|^2, \quad \forall x \in R^n,\ z \in \Omega.$

All through this article, we assume the following

(A_1) The solution set of (1.1), denoted by Ω', is nonempty.

(A_2) The mapping F is monotone, that is,

$$(F(x) - F(y))^T(x - y) \geq 0, \quad \forall x, y \in R^n.$$

(A_3) The mapping $F(.)$ is Lipschitz continuous, that is there exists a positive constant L such that

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

Algorithm 2.3. ***Modified Self-adaptive CG method (MSCG)***

Step 0. *Given an arbitrary initial point $x_0 \in \Omega$, parameters $\beta > 0$, $r > 0$, $0 < \mu < 2$, $\sigma > 0$, $0 < \rho < 1$, $Tol > 0$, and set $k := 0$.*

Step 1. *If $\|F(x_k)\| \leq Tol$, stop, otherwise go to **Step 2.***

Step 2. *Compute*

$$d_k = \begin{cases} -F(x_k), & \text{if } k = 0, \\ -F(x_k) + \beta_k^{'} d_{k-1} - \theta_k^{'} w_{k-1}, & \text{if } k \geq 1, \end{cases} \tag{2.1}$$

where

$$\beta_k^{'} = \frac{F(x_k)^T w_{k-1}}{d_{k-1}^T w_{k-1}}, \quad \theta_k^{'} = \frac{F(x_k)^T d_{k-1}}{d_{k-1}^T w_{k-1}} \tag{2.2}$$

$$y_{k-1} = F(x_k) - F(x_{k-1}) + r s_{k-1}, \quad s_{k-1} = x_k - x_{k-1}, \tag{2.3}$$

$$w_{k-1} = y_{k-1} + t_{k-1} d_{k-1}, \quad t_{k-1} = 1 + max\left\{0, -\frac{d_{k-1}^T y_{k-1}}{d_{k-1}^T d_{k-1}}\right\}. \tag{2.4}$$

Step 3. *Compute the step length $\alpha_k = \beta\rho^{m_k}$ and m_k is the smallest non-negative integer m such that*

$$-\langle F(x_k + \beta\rho^m d_k), d_k\rangle \geq \sigma\beta\rho^m \|d_k\|^2. \tag{2.5}$$

Step 4. *Set $z_k = x_k + \alpha_k d_k$ and compute*

$$x_{k+1} = P_\Omega[x_k - \mu\zeta_k F(z_k)]$$

where

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

Step 5. *Let $k = k + 1$ and go to **Step 1**.*

It can be observed that the modification made is by replacing β_k^{HS}, θ_k in [24] with $\beta_k^{'}$, $\theta_k^{'}$ respectively in the proposed algorithm.

Remark 2.4.

$$\begin{aligned} & F(x_k)^T d_k \\ & = -F(x_k)^T F(x_k) + \frac{F(x_k)^T (F(x_k)^T w_{k-1}) d_{k-1} - F(x_k)^T (F(x_k)^T d_{k-1}) w_{k-1}}{d_{k-1}^T w_{k-1}} \\ & = -\|F(x_k)\|^2 + \frac{(F(x_k)^T d_{k-1})(F(x_k)^T w_{k-1}) - (F(x_k)^T w_{k-1})(F(x_k)^T d_{k-1})}{d_{k-1}^T w_{k-1}} \\ & = -\|F(x_k)\|^2. \end{aligned} \tag{2.6}$$

Using Cauchy-Schwartz inequality, we get

$$\|F(x_k)\| \leq \|d_k\|. \tag{2.7}$$

Remark 2.5. From the definition of w_{k-1}, t_{k-1} and (2.7), we have

$$d_{k-1}^T w_{k-1} \geq d_{k-1}^T y_{k-1} + \|d_{k-1}\|^2 - d_{k-1}^T y_{k-1} = \|d_{k-1}\|^2 \tag{2.8}$$

3. Convergence analysis

To prove the global convergence of Algorithm 2.3, the following lemmas are needed. The following lemma shows that Algorithm 2.3 is well-defined.

Lemma 3.1. *Suppose that assumptions (A_1)-(A_3) hold, then there exists a step-length α_k satisfying the line search (2.5) $\forall k \geq 0$.*

Proof. Suppose there exists $k_0 \geq 0$ such that (2.5) does not hold for any non-negative integer i, i.e.,

$$-\langle F(x_{k_0} + \beta\rho^i d_{k_0}), d_{k_0}\rangle < \sigma\beta\rho^i \|d_{k_0}\|^2.$$

Using assumption (A_3) and allowing $i \to \infty$, we get

$$-\langle F(x_{k_0}), d_{k_0}\rangle \leq 0. \tag{3.1}$$

Also from (2.6), we have

$$-\langle F(x_{k_0}), d_{k_0}\rangle = \|F(x_{k_0})\|^2 > 0,$$

which contradicts (3.1). The proof is complete. ∎

Lemma 3.2. *Suppose that (A_3) hold and the sequences $\{x_k\}$ and $\{z_k\}$ be generated by Algorithm 2.3. The we have*

$$\alpha_k \geq \rho \min\left\{\beta, \rho\frac{\|F(x_k)\|^2}{(L+\sigma)\|d_k\|^2}\right\}.$$

Proof. Suppose $\alpha_k \neq \beta$, then $\frac{\alpha_k}{\rho}$ does not satisfy equation (2.5), that is

$$-F\left(x_k + \frac{\alpha_k}{\rho}d_k\right)^T d_k < \sigma\frac{\alpha_k}{\rho}\|d_k\|^2.$$

This combined with (2.6) and the fact that F is Lipschitz continuous yields

$$\begin{aligned}\|F(x_k)\|^2 &= -F(x_k)^T d_k \\ &= \left(F(x_k + \frac{\alpha_k}{\rho}d_k) - F(x_k)\right)^T d_k - F\left(x_k + \frac{\alpha_k}{\rho}d_k\right)^T d_k \\ &\leq L\frac{\alpha_k}{\rho}\|d_k\|^2 + \sigma\frac{\alpha_k}{\rho}\|d_k\|^2 \\ &= \frac{L+\sigma}{\rho}\alpha_k\|d_k\|^2.\end{aligned} \tag{3.2}$$

The above equation implies

$$\alpha_k \geq \rho\frac{\|F(x_k)\|^2}{(L+\sigma)\|d_k\|^2},$$

which completes the proof. ∎

Lemma 3.3. *Suppose that assumptions (A_1)-(A_3) hold, then the sequences $\{x_k\}$ and $\{z_k\}$ generated by Algorithm 2.3 are bounded. Moreover, we have*

$$\lim_{k\to\infty} \|x_k - z_k\| = 0, \tag{3.3}$$

and

$$\lim_{k\to\infty} \|x_{k+1} - x_k\| = 0. \tag{3.4}$$

Proof. We will start by showing that the sequences $\{x_k\}$ and $\{z_k\}$ are bounded. Suppose $\bar{x} \in \Omega^{'}$, then by monotonicity of F, we get

$$\langle F(z_k), x_k - \bar{x}\rangle \geq \langle F(z_k), x_k - z_k\rangle. \tag{3.5}$$

Also by definition of z_k and the line search (2.5), we have

$$\langle F(z_k), x_k - z_k\rangle \geq \sigma\alpha_k^2\|d_k\|^2 \geq 0. \tag{3.6}$$

So, we have

$$\begin{aligned}
&\|x_{k+1} - \bar{x}\|^2\\
&= \|P_\Omega[x_k - \mu\zeta_k F(z_k)] - \bar{x}\|^2\\
&\leq \|x_k - \mu\zeta_k F(z_k) - \bar{x}\|^2\\
&= \|x_k - \bar{x}\|^2 - 2\mu\zeta_k\langle F(z_k), x_k - \bar{x}\rangle + \|\mu\zeta_k F(z_k)\|^2\\
&= \|x_k - \bar{x}\|^2 - 2\mu\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2}\langle F(z_k), x_k - \bar{x}\rangle + \mu^2\left(\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|}\right)^2\\
&\leq \|x_k - \bar{x}\|^2 - 2\mu\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|^2}\langle F(z_k), x_k - z_k\rangle + \mu^2\left(\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|}\right)^2\\
&\leq \|x_k - \bar{x}\|^2 - \mu(2-\mu)\left(\frac{\langle F(z_k), x_k - z_k\rangle}{\|F(z_k)\|}\right)^2\\
&= \|x_k - \bar{x}\|^2 - \mu(2-\mu)\frac{\sigma^2\|x_k - z_k\|^4}{\|F(z_k)\|^2}.
\end{aligned} \tag{3.7}$$

Thus the sequence $\{\|x_k - \bar{x}\|\}$ is non increasing and convergent, and hence $\{x_k\}$ is bounded. Furthermore, from equation (3.7), we have

$$\|x_{k+1} - \bar{x}\|^2 \leq \|x_k - \bar{x}\|^2, \tag{3.8}$$

and we can deduce recursively that

$$\|x_k - \bar{x}\|^2 \leq \|x_0 - \bar{x}\|^2, \quad \forall k \geq 0.$$

Then from Assumption (A_3), we obtain

$$\|F(x_k)\| = \|F(x_k) - F(\bar{x})\| \leq L\|x_k - \bar{x}\| \leq L\|x_0 - \bar{x}\|.$$

If we let $L\|x_0 - \bar{x}\| = \omega$, then the sequence $\{F(x_k)\}$ is bounded, that is,

$$\|F(x_k)\| \leq \omega, \quad \forall k \geq 0. \tag{3.9}$$

By the definition of z_k, equation (3.6), monotonicity of F and the Cauchy-Schwatz inequality, we get

$$\sigma\|x_k - z_k\| = \frac{\sigma\|\alpha_k d_k\|^2}{\|x_k - z_k\|} \leq \frac{\langle F(z_k), x_k - z_k\rangle}{\|x_k - z_k\|} \leq \frac{\langle F(x_k), x_k - z_k\rangle}{\|x_k - z_k\|} \leq \|F(x_k)\|. \tag{3.10}$$

The boundedness of the sequence $\{x_k\}$ together with equations (3.9)-(3.10), implies that the sequence $\{z_k\}$ is bounded.

Since $\{z_k\}$ is bounded, then for any $\bar{x} \in \Omega'$, the sequence $\{z_k - \bar{x}\}$ is also bounded, that is, there exists a positive constant $\nu > 0$ such that

$$\|z_k - \bar{x}\| \leq \nu, \ \forall k \geq 0.$$

This together with Assumption (A_3) yields

$$\|F(z_k)\| = \|F(z_k) - F(\bar{x})\| \leq L\|z_k - \bar{x}\| \leq L\nu.$$

Therefore, using equation (3.7), we have

$$\mu(2-\mu)\frac{\sigma^2}{(L\nu)^2}\|x_k - z_k\|^4 \leq \|x_k - \bar{x}\|^2 - \|x_{k+1} - \bar{x}\|^2,$$

which implies

$$\mu(2-\mu)\frac{\sigma^2}{(L\nu)^2}\sum_{k=0}^{\infty}\|x_k - z_k\|^4 \leq \sum_{k=0}^{\infty}(\|x_k - \bar{x}\|^2 - \|x_{k+1} - \bar{x}\|^2) \leq \|x_0 - \bar{x}\| < \infty. \tag{3.11}$$

Equation (3.11) implies

$$\lim_{k\to\infty}\|x_k - z_k\| = 0.$$

However, using statement 2 of lemma 2.2, the definition of ζ_k and the Cauchy-Schwatz inequality, we have

$$\begin{aligned}\|x_{k+1} - x_k\| &= \|P_\Omega[x_k - \mu\zeta_k F(z_k)] - x_k\| \\ &= \|x_k - \mu\zeta_k F(z_k) - x_k\| \\ &= \|\mu\zeta_k F(z_k)\| \\ &= \mu\|x_k - z_k\|, \ \forall k \geq 0,\end{aligned} \tag{3.12}$$

which yields

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

∎

Remark 3.4. By equation (3.3) and definition of z_k, we have

$$\lim_{k\to\infty} \alpha_k \|d_k\| = 0. \tag{3.13}$$

Theorem 3.5. *Suppose that assumptions (A_1)-(A_3) hold and let the sequence $\{x_k\}$ be generated by Algorithm 2.3, then*

$$\liminf_{k\to\infty} \|F(x_k)\| = 0. \tag{3.14}$$

Proof. Assume that equation (3.14) is not true, then there exists a constant $\epsilon > 0$ such that

$$\|F(x_k)\| \geq \epsilon, \quad \forall k \geq 0. \tag{3.15}$$

We will first show that the sequence $\{d_k\}$ is bounded. From the definition of t_{k-1}, we have

$$\begin{aligned} |t_{k-1}| &= \left|1 + max\left\{0, -\frac{d_{k-1}^T y_{k-1}}{\|d_{k-1}\|^2}\right\}\right| \\ &\leq 1 + \frac{|d_{k-1}^T y_{k-1}|}{\|d_{k-1}\|^2} \\ &\leq 1 + \frac{\|d_{k-1}\|\|y_{k-1}\|}{\|d_{k-1}\|^2} \\ &= 1 + \frac{\|y_{k-1}\|}{\|d_{k-1}\|}. \end{aligned} \tag{3.16}$$

Also from definition of y_{k-1} and assumption (A_3), we have

$$\begin{aligned} \|y_{k-1}\| &\leq \|F(x_k) - F(x_{k-1})\| + r\|s_{k-1}\| \\ &\leq (L + r)\|s_{k-1}\| \\ &\leq (L + r)\alpha_{k-1}\|d_{k-1}\|. \end{aligned} \tag{3.17}$$

Furthermore by definition of w_{k-1}, (3.16) and (3.17), we obtain

$$\begin{aligned} \|w_{k-1}\| &= \|y_{k-1} + t_{k-1}d_{k-1}\| \\ &\le \|y_{k-1}\| + |t_{k-1}|\|d_{k-1}\| \\ &\le (L+r)\alpha_{k-1}\|d_{k-1}\| + \left(1 + \frac{\|y_{k-1}\|}{\|d_{k-1}\|}\right)\|d_{k-1}\| \\ &= (L+r)\alpha_{k-1}\|d_{k-1}\| + \|d_{k-1}\| + \|y_{k-1}\| \\ &\le (2(L+r)\alpha_{k-1} + 1)\|d_{k-1}\|. \end{aligned} \tag{3.18}$$

Therefore, by (2.1), (2.8), (3.9), (3.18) and Cauchy-Schwatz inequality, we have

$$\begin{aligned} \|d_k\| &\le \|F(x_k)\| + \frac{\|F(x_k)\|\|w_{k-1}\|\|d_{k-1}\|}{|d_{k-1}^T w_{k-1}|} + \frac{\|F(x_k)\|\|d_{k-1}\|\|w_{k-1}\|}{|d_{k-1}^T w_{k-1}|} \\ &\le \|F(x_k)\| + (4(L+r)\alpha_{k-1} + 2)\|F(x_k)\| \\ &= (1 + 4(L+r)\alpha_{k-1} + 2)\|F(x_k)\| \\ &\le (1 + 4(L+r)\beta + 2)\omega. \end{aligned} \tag{3.19}$$

Letting $C = (1 + 4(L+r)\beta + 2)\omega$, then $\|d_k\| \le C, \quad \forall k \ge 0$.
Combining (2.7) and (3.15), we have

$$\|d_k\| \ge \|F(x_k)\| \ge \epsilon, \quad \forall k \ge 0.$$

As $z_k = x_k + \alpha_k d_k$ and $\lim_{k\to\infty} \|x_k - z_k\| = 0$, we get $\lim_{k\to\infty} \alpha_k\|d_k\| = 0$ and

$$\lim_{k\to\infty} \alpha_k = 0. \tag{3.20}$$

On the other side, lemma 3.2 and (3.19) imply $\alpha_k\|d_k\| \ge \min\left\{\beta\epsilon \frac{\epsilon^2}{(L+\sigma)C^2}\right\}$, which contradicts with (3.20). Therefore, (3.14) must hold. ∎

4. Some Applications and Numerical examples

This section reports some numerical results to show the efficiency of Algorithm 2.3. For convenience sake, we denote Algorithm 2.3 by **MSCG** method. We also divide this section into two. First we compare **MSCG** method with **PCG** method [14] by solving some monotone nonlinear equations with convex constraints using different initial points and several dimensions. Secondly, the **MSCG** method is applied to solve signal recovery problems. All codes were written in MATLAB R2017a and run on a PC with intel COREi5 processor with 4GB of RAM and CPU 2.3GHZ.

4.1. Numerical examples on some convex constrained nonlinear monotone equations

Same line search implementation was used for both **MSCG** and **PCG** and the specific parameters used for each method are as follows:

MSCG method: $\beta = 1$, $\mu = 1.8$, $\rho = 0.6$, $r = 0.1$, $\sigma = 0.0001$.

PCG method: All parameters are choosen as in [14].

All runs were stopped whenever

$\|F(x_k)\| < 10^{-6}$.

We test problems 1 to 9 with dimensions of $n = 1000,\ 5000,\ 10,000,\ 50,000,\ 100,000$ and different initial points: $x_1 = (1,1,...,1)^T$, $x_2 = (2,2,...,2)^T$, $x_3 = (3,3,...,3)^T$, $x_4 = (5,5,...,5)^T$, $x_5 = (8,8,...,8)^T$, $x_6 = (0.5,0.5,...0.5)^T$, $x_7 = (0.1,0.1,...,0.1)^T$, $x_8 = (10,10,...,10)^T$. The numerical results in Tables 1-9 report the number of iterations (ITER), number of function evaluations (FVAL), CPU time in seconds (TIME) and the norm at the approximate solution (NORM). The symbol '$-$' is used to indicate that the number of iterations exceeds 1000 and/or the number of function evaluations exceeds 2000.

The problem functions $F(x) = (f_1(x), f_2(x), ..., f_n(x))^T$, where $x = (x_1, x_2, ..., x_n)^T$, and feasible sets $\Omega \subset R^n$ tested are listed as follows:

Problem 1 Modified exponential function

$$\begin{aligned} F_1(x) &= e^{x_1} - 1 \\ F_i(x) &= e^{x_i} + x_{i-1} - 1 \text{ for } i = 2,3,...,n \\ \text{and} \quad \Omega &= \mathbb{R}^n_+. \end{aligned}$$

Problem 2 Logarithmic Function

$$F_i(x) = \ln(|x_i| + 1) - \frac{x_i}{n}, \text{ for } i = 2,3,...,n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 3 [29]

$$F_i(x) = 2x_i - \sin|x_i|,\ i = 1,2,3,...,n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 4 [13]

$$F_i(x) = \min\left(\min(|x_i|, x_i^2), \max(|x_i|, x_i^3)\right) \text{ for } i = 2,3,...,n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 5 Strictly convex function [23]

$$F_i(x) = e^{x_i} - 1, \text{ for } i = 2,3,...,n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

Problem 6 Linear monotone problem

$$F_1(x) = 2.5x_1 + x_2 - 1$$
$$F_i(x) = x_{i-1} + 2.5x_i + x_{i+1} - 1 \text{ for } i = 2, 3, ..., n-1$$
$$F_n(x) = x_{n-1} + 2.5x_n - 1$$
and $\Omega = \mathbb{R}^n_+$.

Problem 7 Tridiagonal Exponential Problem [4]

$$F_1(x) = x_1 - e^{\cos(h(x_1+x_2))}$$
$$F_i(x) = x_i - e^{\cos(h(x_{i-1}+x_i+x_{i+1}))} \text{ for } i = 2, 3, ..., n-1$$
$$F_n(x) = x_n - e^{\cos(h(x_{n-1}+x_n))},$$
where $h = \dfrac{1}{n+1}$
and $\Omega = \mathbb{R}^n_+$.

Problem 8

$$F_1(x) = 3x_1^3 + 2x_2 - 5 + \sin(x_1 - x_2)\sin(x_1 + x_2)$$
$$F_i(x) = 3x_i^3 + 2x_{i+1} - 5 + \sin(x_i - x_{i+1})\sin(x_i + x_{i+1}) + 4x_i - x_{i-1}e^{x_{i-1}-x_i} - 3$$
for $i = 2, 3, ..., n-1$
$$F_n(x) = x_{n-1}e^{x_{n-1}-x_n} - 4x_n - 3,$$
where $h = \dfrac{1}{n+1}$
and $\Omega = \mathbb{R}^n_+$.

Problem 9

$$F_i(x) = x_i - \sin|x_i - 1|, \; i = 1, 2, 3, ..., n \quad \text{and} \quad \Omega = \mathbb{R}^n_+.$$

The numerical results indicate that the **MSCG** method is more effective than the **PCG** method for the given problems as it solves and win 7 out of 9 of the problems tested both in terms of number of iterations, number of function evaluations and CPU time (see Tables 1-7). In particular, the **PCG** method fails to solve problems 4 completely while **MSCG** was able to solve all the problems except for the initial points x_6 and x_7 (see Table 4). Therefore, we can conclude that **MSCG** method is a very effecient tool for solving nonlinear monotone equations with convex constraints, especially for large-scale dimensions.

4.2. Experiments on solving some signal recovery problems in compressive sensing

There are many problems in signal processing and statistical inference involving finding sparse solutions to ill-conditioned linear systems of equations. Among popular approach is minimizing an objective function which contains quadratic (ℓ_2) error term and a sparse ℓ_1-regularization term, i.e.,

$$\min_x \frac{1}{2}\|y - Ax\|_2^2 + \tau\|x\|_1, \tag{4.1}$$

TABLE 1. Numerical Results for **MSCG** and **PCG** for Problem 1 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	2	9	0.007489	0	52	189	0.110321	8.55E-07
	x_2	2	10	0.005551	0	56	204	0.035141	8.99E-07
	x_3	2	11	0.014899	0	56	205	0.033533	8.39E-07
	x_4	2	14	0.006243	0	65	237	0.041638	8.27E-07
	x_5	2	19	0.006022	0	78	310	0.05362	9.48E-07
	x_6	2	9	0.002681	0	54	196	0.035021	6.83E-07
	x_7	2	9	0.004044	0	49	178	0.042635	7.85E-07
	x_8	2	23	0.005336	0	69	266	0.043962	8.44E-07
5000	x_1	2	9	0.022857	0	50	183	0.09149	9.98E-07
	x_2	2	10	0.006854	0	56	205	0.103092	6.81E-07
	x_3	2	11	0.011049	0	54	199	0.104065	9.9E-07
	x_4	2	14	0.011355	0	64	234	0.115013	9.48E-07
	x_5	2	19	0.013144	0	70	278	0.131778	8.96E-07
	x_6	2	9	0.012189	0	53	193	0.09886	7.86E-07
	x_7	2	9	0.009516	0	47	172	0.095304	9.24E-07
	x_8	2	23	0.014401	0	70	269	0.128741	6.8E-07
10000	x_1	2	9	0.017626	0	51	187	0.173696	6.85E-07
	x_2	2	10	0.013878	0	55	202	0.242626	7.46E-07
	x_3	2	11	0.016471	0	55	203	0.181091	6.79E-07
	x_4	2	14	0.016887	0	64	234	0.207939	9.63E-07
	x_5	2	19	0.019661	0	87	467	0.370102	9.32E-07
	x_6	2	9	0.011591	0	52	190	0.166351	8.5E-07
	x_7	2	9	0.011988	0	46	169	0.15083	1E-06
	x_8	2	23	0.022006	0	68	262	0.215882	9.85E-07
50000	x_1	2	9	0.040326	0	49	181	0.707163	8.25E-07
	x_2	2	10	0.039481	0	54	199	0.901824	8.74E-07
	x_3	2	11	0.044192	0	53	197	0.779749	8.21E-07
	x_4	2	14	0.066019	0	63	232	1.047125	7.61E-07
	x_5	2	19	0.062454	0	73	293	1.479499	7.54E-07
	x_6	2	9	0.038738	0	51	187	0.769376	9.74E-07
	x_7	2	9	0.048211	0	46	170	0.650199	7.66E-07
	x_8	2	23	0.075239	0	68	262	1.034824	9.43E-07
100000	x_1	2	9	0.071954	0	49	181	1.456718	8.7E-07
	x_2	2	10	0.070337	0	53	196	1.564101	9.51E-07
	x_3	2	11	0.129537	0	53	197	1.632904	8.69E-07
	x_4	2	14	0.103765	0	62	229	1.858922	8.28E-07
	x_5	2	19	0.132033	0	91	394	2.909579	9.4E-07
	x_6	2	9	0.066613	0	51	188	1.480664	7.13E-07
	x_7	2	9	0.070007	0	46	170	1.334943	8.04E-07
	x_8	2	23	0.158984	0	68	262	1.981132	9.39E-07

where $x \in R^n$, $y \in R^k$ is an observation, $A \in R^{k\times n}$ $(k << n)$ is a linear operator, τ is a nonnegative parameter, $\|x\|_2$ denotes the Euclidean norm of x and $\|x\|_1 = \sum_{i=1}^n |x_i|$ is the ℓ_1-norm of x. It is easy to see that problem (4.1) is a convex unconstrained minimization problem. Due to the fact that if the original signal is sparse or approximately sparse in some orthogonal basis, problem (4.1) frequently appears in compressive sensing, and hence an exact restoration can be produced by solving (4.1).

TABLE 2. Numerical Results for **MSCG** and **PCG** for Problem 2 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	2	7	0.002287	0	4	9	0.005149	0
	x_2	3	10	0.002837	0	5	11	0.005127	0
	x_3	3	10	0.0042	0	6	13	0.004934	0
	x_4	3	10	0.004002	0	7	15	0.006266	0
	x_5	4	13	0.004927	0	9	19	0.009583	0
	x_6	2	7	0.002839	0	3	7	0.003695	0
	x_7	2	7	0.003192	0	2	5	0.003077	0
	x_8	5	16	0.005398	0	10	21	0.008973	0
5000	x_1	2	7	0.005581	0	4	9	0.009838	0
	x_2	3	10	0.012182	0	5	11	0.012215	0
	x_3	3	10	0.014385	0	6	13	0.016737	0
	x_4	3	10	0.009594	0	7	15	0.017589	0
	x_5	4	13	0.011344	0	9	19	0.020153	0
	x_6	2	7	0.007545	0	3	7	0.010968	0
	x_7	2	7	0.006425	0	2	5	0.008253	0
	x_8	5	16	0.018064	0	10	21	0.024443	0
10000	x_1	2	7	0.01418	0	4	9	0.018819	0
	x_2	3	10	0.014155	0	5	11	0.021572	0
	x_3	3	10	0.014925	0	6	13	0.024532	0
	x_4	3	10	0.014886	0	7	15	0.028731	0
	x_5	4	13	0.017717	0	9	19	0.033857	0
	x_6	2	7	0.010365	0	3	7	0.018178	0
	x_7	2	7	0.010509	0	2	5	0.012152	0
	x_8	5	16	0.022966	0	10	21	0.037341	0
50000	x_1	2	7	0.032816	0	4	9	0.059874	0
	x_2	3	10	0.057053	0	5	11	0.080753	0
	x_3	3	10	0.047544	0	6	13	0.088801	0
	x_4	3	10	0.045036	0	7	15	0.102315	0
	x_5	4	13	0.063642	0	9	19	0.129467	0
	x_6	2	7	0.034852	0	3	7	0.054439	0
	x_7	2	7	0.035797	0	2	5	0.035327	0
	x_8	5	16	0.087191	0	10	21	0.143865	0
100000	x_1	2	7	0.057021	0	4	9	0.119186	0
	x_2	3	10	0.091601	0	5	11	0.144375	0
	x_3	3	10	0.0876	0	6	13	0.176278	0
	x_4	3	10	0.11867	0	7	15	0.202121	0
	x_5	4	13	0.122207	0	9	19	0.254424	0
	x_6	2	7	0.061273	0	3	7	0.092884	0
	x_7	2	7	0.086732	0	2	5	0.064906	0
	x_8	5	16	0.137848	0	10	21	0.279527	0

Iterative methods for solving (4.1) have been presented in many literatures, (see [2, 5, 7, 8, 12, 21]). The most popular method among these methods is the gradient based method and the earliest gradient projection method for sparse reconstruction (GPRS) was proposed by Figueiredo et al. [8]. The first step of the GPRS method is to express (4.1) as a quadratic problem using the following process.
Let $x \in R^n$ and splitting it into its positive and negative parts. Then x can be formulated

Table 3. Numerical Results for **MSCG** and **PCG** for Problem 3 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	2	8	0.002131	0	11	29	0.007647	5.91E-07
	x_2	2	8	0.002618	0	11	29	0.006415	9.95E-07
	x_3	2	9	0.003474	0	11	30	0.007009	7.4E-07
	x_4	2	9	0.003267	0	13	36	0.010125	5.86E-07
	x_5	2	9	0.002792	0	12	34	0.006802	1.45E-07
	x_6	2	8	0.003134	0	11	29	0.008319	6.19E-07
	x_7	2	8	0.00293	0	10	26	0.008698	3.51E-07
	x_8	2	9	0.003217	0	12	34	0.011399	1.02E-07
5000	x_1	2	8	0.004982	0	12	31	0.023695	1.23E-07
	x_2	2	8	0.007135	0	12	32	0.024907	9.71E-07
	x_3	2	9	0.009629	0	12	33	0.024784	7.22E-07
	x_4	2	9	0.009662	0	14	38	0.028132	1.22E-07
	x_5	2	9	0.007819	0	12	34	0.025162	3.25E-07
	x_6	2	8	0.006236	0	12	31	0.022688	1.28E-07
	x_7	2	8	0.00624	0	10	26	0.019349	7.85E-07
	x_8	2	9	0.006238	0	12	34	0.026113	2.27E-07
10000	x_1	2	8	0.008726	0	12	31	0.042552	1.73E-07
	x_2	2	8	0.012242	0	13	34	0.04396	1.27E-07
	x_3	2	9	0.012729	0	13	35	0.041531	9.47E-08
	x_4	2	9	0.012833	0	14	38	0.043616	1.72E-07
	x_5	2	9	0.008266	0	12	34	0.040575	4.59E-07
	x_6	2	8	0.018599	0	12	31	0.037381	1.81E-07
	x_7	2	8	0.009012	0	11	29	0.034501	4.84E-07
	x_8	2	9	0.009328	0	12	34	0.042223	3.21E-07
50000	x_1	2	8	0.032446	0	12	31	0.139558	3.88E-07
	x_2	2	8	0.04508	0	13	34	0.153158	2.85E-07
	x_3	2	9	0.035499	0	13	35	0.154392	2.12E-07
	x_4	2	9	0.043984	0	14	38	0.165244	3.85E-07
	x_5	2	9	0.037903	0	13	37	0.162555	4.48E-07
	x_6	2	8	0.03081	0	12	31	0.135904	4.06E-07
	x_7	2	8	0.031101	0	12	31	0.133471	1.01E-07
	x_8	2	9	0.034806	0	12	34	0.148058	7.18E-07
100000	x_1	2	8	0.063409	0	12	31	0.272996	5.48E-07
	x_2	2	8	0.064054	0	13	34	0.298372	4.03E-07
	x_3	2	9	0.085457	0	13	35	0.303707	2.99E-07
	x_4	2	9	0.065247	0	14	38	0.329149	5.44E-07
	x_5	2	9	0.070299	0	13	37	0.324363	6.34E-07
	x_6	2	8	0.058418	0	12	31	0.272816	5.74E-07
	x_7	2	8	0.056572	0	12	31	0.274954	1.42E-07
	x_8	2	9	0.092306	0	13	37	0.357351	4.43E-07

as

$$x = u - v, \qquad u \geq 0, \quad v \geq 0,$$

where $u_i = (x_i)_+$, $v_i = (-x_i)_+$ for all $i = 1, 2, ..., n$, and $(.)_+ = \max\{0, .\}$. By definition of ℓ_1-norm, we have $\|x\|_1 = e_n^T u + e_n^T v$, where $e_n = (1, 1, ..., 1)^T \in R^n$. Now (4.1) can be

Table 4. Numerical Results for **MSCG** and **PCG** for Problem 4 with given initial points and dimensions

		MSCG				PCG			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	2	8	0.002519	0	-	-	-	-
	x_2	2	8	0.004286	0	-	-	-	-
	x_3	2	8	0.003714	0	-	-	-	-
	x_4	2	8	0.004556	0	-	-	-	-
	x_5	2	8	0.005003	0	-	-	-	-
	x_6	-	-	-	-	-	-	-	-
	x_7	-	-	-	-	-	-	-	-
	x_8	2	8	0.002868	0	-	-	-	-
5000	x_1	2	8	0.012432	0	-	-	-	-
	x_2	2	8	0.011534	0	-	-	-	-
	x_3	2	8	0.009014	0	-	-	-	-
	x_4	2	8	0.008944	0	-	-	-	-
	x_5	2	8	0.009143	0	-	-	-	-
	x_6	-	-	-	-	-	-	-	-
	x_7	-	-	-	-	-	-	-	-
	x_8	2	8	0.010586	0	-	-	-	-
10000	x_1	2	8	0.013601	0	-	-	-	-
	x_2	2	8	0.017482	0	-	-	-	-
	x_3	2	8	0.020485	0	-	-	-	-
	x_4	2	8	0.016717	0	-	-	-	-
	x_5	2	8	0.016828	0	-	-	-	-
	x_6	-	-	-	-	-	-	-	-
	x_7	-	-	-	-	-	-	-	-
	x_8	2	8	0.014088	0	-	-	-	-
50000	x_1	2	8	0.059616	0	-	-	-	-
	x_2	2	8	0.068817	0	-	-	-	-
	x_3	2	8	0.064686	0	-	-	-	-
	x_4	2	8	0.061062	0	-	-	-	-
	x_5	2	8	0.067814	0	-	-	-	-
	x_6	-	-	-	-	-	-	-	-
	x_7	-	-	-	-	-	-	-	-
	x_8	2	8	0.064979	0	-	-	-	-
100000	x_1	2	8	0.115326	0	-	-	-	-
	x_2	2	8	0.120746	0	-	-	-	-
	x_3	2	8	0.135422	0	-	-	-	-
	x_4	2	8	0.138436	0	-	-	-	-
	x_5	2	8	0.129448	0	-	-	-	-
	x_6	-	-	-	-	-	-	-	-
	x_7	-	-	-	-	-	-	-	-
	x_8	2	8	0.120702	0	-	-	-	-

written as

$$\min_{u,v} \frac{1}{2}\|y - A(u-v)\|_2^2 + \tau e_n^T u + \tau e_n^T v, \qquad u \geq 0, \quad v \geq 0, \tag{4.2}$$

TABLE 5. Numerical Results for **MSCG** and **PCG** for Problem 5 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	2	9	0.00237	0	11	29	0.006732	1.54E-07
	x_2	2	10	0.002475	0	11	30	0.007149	1.79E-07
	x_3	2	11	0.002987	0	13	39	0.009395	2.64E-07
	x_4	2	14	0.003295	0	12	37	0.008608	2.18E-07
	x_5	2	19	0.00434	0	12	41	0.009546	2.64E-07
	x_6	2	8	0.00273	0	11	29	0.00746	2.65E-07
	x_7	2	8	0.003503	0	10	26	0.009129	2.28E-07
	x_8	2	23	0.004658	0	1	14	0.003566	0
5000	x_1	2	9	0.009611	0	11	29	0.01907	3.45E-07
	x_2	2	10	0.006664	0	11	30	0.019865	4E-07
	x_3	2	11	0.006446	0	13	39	0.02496	5.9E-07
	x_4	2	14	0.008092	0	12	37	0.024694	4.88E-07
	x_5	2	19	0.012408	0	12	41	0.02616	5.9E-07
	x_6	2	8	0.009391	0	11	29	0.020571	5.91E-07
	x_7	2	8	0.005317	0	10	26	0.019593	5.1E-07
	x_8	2	23	0.02448	0	1	14	0.010907	0
10000	x_1	2	9	0.007472	0	11	29	0.029425	4.87E-07
	x_2	2	10	0.012876	0	11	30	0.030602	5.65E-07
	x_3	2	11	0.011031	0	13	39	0.039	8.34E-07
	x_4	2	14	0.011727	0	12	37	0.03542	6.9E-07
	x_5	2	19	0.025132	0	12	41	0.035758	8.34E-07
	x_6	2	8	0.006515	0	11	29	0.028215	8.36E-07
	x_7	2	8	0.008092	0	10	26	0.03742	7.21E-07
	x_8	2	23	0.019344	0	1	14	0.012921	0
50000	x_1	2	9	0.028692	0	12	32	0.114236	4.75E-07
	x_2	2	10	0.030939	0	12	33	0.128343	5.51E-07
	x_3	2	11	0.03574	0	14	42	0.145184	8.13E-07
	x_4	2	14	0.042175	0	13	40	0.134906	6.73E-07
	x_5	2	19	0.051753	0	13	44	0.151662	8.14E-07
	x_6	2	8	0.030385	0	12	32	0.112251	8.16E-07
	x_7	2	8	0.025703	0	11	29	0.236718	7.04E-07
	x_8	2	23	0.074049	0	1	14	0.072093	0
100000	x_1	2	9	0.051896	0	12	32	0.379062	6.72E-07
	x_2	2	10	0.057322	0	12	33	0.296027	7.8E-07
	x_3	2	11	0.074678	0	15	44	0.420986	1.07E-07
	x_4	2	14	0.073831	0	13	40	0.37295	9.52E-07
	x_5	2	19	0.112341	0	14	46	0.371747	1.07E-07
	x_6	2	8	0.044526	0	13	34	0.3338	1.07E-07
	x_7	2	8	0.050034	0	11	29	0.274607	9.95E-07
	x_8	2	23	0.105816	0	1	14	0.079056	0

which is a bound-constrained quadratic program. However, from [8], equation (4.2) can be written in standard form as

$$\min_z \frac{1}{2} z^T D z + c^T z, \qquad \text{such that} \quad z \geq 0, \tag{4.3}$$

Table 6. Numerical Results for **MSCG** and **PCG** for Problem 6 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	38	243	0.026874	4.9E-07	80	366	0.04524	8.69E-07
	x_2	58	367	0.045364	5.9E-07	64	295	0.04116	9.27E-07
	x_3	58	365	0.040542	6.6E-07	81	371	0.04937	9.38E-07
	x_4	45	288	0.035103	9.3E-07	92	420	0.0599	9.21E-07
	x_5	43	276	0.036887	4.9E-07	98	447	0.06313	9.72E-07
	x_6	55	348	0.046344	8.2E-07	88	401	0.05872	8.89E-07
	x_7	48	304	0.037347	6.5E-07	93	423	0.05981	8.67E-07
	x_8	39	251	0.032886	7.3E-07	102	465	0.0686	8.53E-07
5000	x_1	39	250	0.095043	9E-07	77	353	0.16437	9.81E-07
	x_2	55	348	0.154642	6.3E-07	63	291	0.14918	8.23E-07
	x_3	43	275	0.1028	4.9E-07	80	367	0.17072	8.51E-07
	x_4	34	223	0.080845	4.9E-07	91	416	0.20171	8.36E-07
	x_5	51	322	0.122567	9.5E-07	97	443	0.24548	8.84E-07
	x_6	39	252	0.121755	8.1E-07	85	388	0.18099	9.98E-07
	x_7	35	226	0.119903	7.1E-07	90	410	0.19088	9.71E-07
	x_8	53	335	0.116079	4.8E-07	99	452	0.20959	9.56E-07
10000	x_1	42	270	0.247204	9.4E-07	76	349	0.33826	9.6E-07
	x_2	42	271	0.195421	7.3E-07	61	282	0.26813	9.8E-07
	x_3	39	252	0.227771	1E-06	80	367	0.3535	8.11E-07
	x_4	55	346	0.253661	7.8E-07	88	403	0.40217	9.94E-07
	x_5	39	252	0.215484	4.2E-07	96	439	0.42976	8.6E-07
	x_6	40	257	0.193518	6.3E-07	84	384	0.36918	9.64E-07
	x_7	40	259	0.1937	8.9E-07	90	410	0.39508	9.38E-07
	x_8	44	284	0.234578	4.8E-07	98	448	0.44134	9.32E-07
50000	x_1	66	417	1.357417	8.7E-07	76	349	1.42871	8.74E-07
	x_2	57	362	1.174249	5.3E-07	60	278	1.11554	8.73E-07
	x_3	53	338	1.175447	8.1E-07	77	354	1.47994	9.12E-07
	x_4	54	343	1.148698	5.3E-07	87	399	1.64205	9.08E-07
	x_5	64	403	1.332901	5.5E-07	93	426	1.76483	9.69E-07
	x_6	57	361	1.156373	9.7E-07	83	380	1.58523	8.85E-07
	x_7	56	356	1.183557	6.7E-07	89	406	1.65191	8.53E-07
	x_8	69	434	1.42208	6.9E-07	98	448	1.84774	8.45E-07
100000	x_1	57	363	2.595996	7.4E-07	75	345	3.36307	8.42E-07
	x_2	48	309	2.27289	9.2E-07	60	278	2.64065	8.38E-07
	x_3	56	359	2.542895	5.3E-07	76	350	3.48779	8.89E-07
	x_4	65	412	2.914784	9.9E-07	87	399	3.83986	8.66E-07
	x_5	59	376	2.674455	7.6E-07	93	426	4.06354	9.21E-07
	x_6	61	386	2.776046	7.1E-07	83	380	3.60289	8.44E-07
	x_7	52	333	2.472388	4.6E-07	88	402	3.86873	8.22E-07
	x_8	51	327	2.367317	8.5E-07	95	435	4.3091	9.98E-07

where $z = \begin{pmatrix} u \\ v \end{pmatrix}, \quad c = \tau e_{2n} + \begin{pmatrix} -b \\ b \end{pmatrix}, \quad b = A^T y, \quad D = \begin{pmatrix} A^T A & -A^T A \\ -A^T A & A^T A \end{pmatrix}.$
Clearly, D is a positive semi-definite matrix, which implies that equation (4.3) is a convex quadratic problem.

Xiao et al. [26] translated (4.3) into a linear variable inequality problem which is equivalent to a linear complementarity problem. Furthermore, they pointed out that z

Table 7. Numerical Results for **MSCG** and **PCG** for Problem 7 with given initial points and dimensions

		MSCG				PCG			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	9	37	0.012528	9.18E-08	12	31	0.010964	2.45E-07
	x_2	8	33	0.008634	4.79E-07	12	31	0.014242	1.02E-07
	x_3	8	33	0.012001	1.88E-07	10	26	0.007035	9.94E-07
	x_4	9	37	0.012042	1.22E-07	12	31	0.012585	3.25E-07
	x_5	9	37	0.011777	2.82E-07	12	31	0.016446	7.53E-07
	x_6	9	37	0.007588	1.18E-07	12	31	0.013752	3.17E-07
	x_7	9	37	0.011852	1.4E-07	12	31	0.014276	3.74E-07
	x_8	9	37	0.012123	3.9E-07	13	34	0.017765	4.53E-07
5000	x_1	9	37	0.036716	2.04E-07	12	31	0.048108	5.51E-07
	x_2	9	37	0.03521	8.52E-08	12	31	0.044947	2.3E-07
	x_3	8	33	0.036131	4.18E-07	11	29	0.03674	9.73E-07
	x_4	9	37	0.034892	2.71E-07	12	31	0.057287	7.31E-07
	x_5	9	37	0.032141	6.27E-07	13	34	0.03866	7.39E-07
	x_6	9	37	0.033881	2.63E-07	12	31	0.041563	7.11E-07
	x_7	9	37	0.033572	3.11E-07	12	31	0.045607	8.39E-07
	x_8	9	37	0.036109	8.64E-07	14	36	0.049345	9.45E-08
10000	x_1	9	37	0.057811	2.88E-07	12	31	0.070204	7.79E-07
	x_2	9	37	0.057191	1.21E-07	12	31	0.070029	3.26E-07
	x_3	8	33	0.07548	5.91E-07	12	31	0.065584	1.28E-07
	x_4	9	37	0.052442	3.83E-07	13	34	0.072424	4.51E-07
	x_5	9	37	0.076822	8.86E-07	14	36	0.074676	9.69E-08
	x_6	9	37	0.063751	3.72E-07	13	34	0.085115	4.39E-07
	x_7	9	37	0.055204	4.39E-07	13	34	0.073913	5.18E-07
	x_7	10	41	0.083553	9.77E-08	14	36	0.087317	1.34E-07
50000	x_1	9	37	0.208485	6.45E-07	13	34	0.302189	7.6E-07
	x_2	9	37	0.231584	2.69E-07	12	31	0.272796	7.28E-07
	x_3	9	37	0.215684	1.06E-07	12	31	0.338126	2.86E-07
	x_4	9	37	0.233193	8.56E-07	14	36	0.425249	9.36E-08
	x_5	10	41	0.238034	1.59E-07	14	36	0.435261	2.17E-07
	x_6	9	37	0.284085	8.32E-07	13	34	0.591563	9.81E-07
	x_7	9	37	0.223528	9.82E-07	14	36	0.401554	1.07E-07
	x_8	10	41	0.244065	2.19E-07	14	36	0.474472	2.99E-07
100000	x_1	9	37	0.45449	9.12E-07	14	36	0.753856	9.97E-08
	x_2	9	37	0.504209	3.81E-07	13	34	0.692957	4.49E-07
	x_3	9	37	0.598617	1.49E-07	12	31	0.606275	4.04E-07
	x_4	10	41	0.494593	9.68E-08	14	36	0.767606	1.32E-07
	x_5	10	41	0.577411	2.24E-07	14	36	0.744496	3.06E-07
	x_6	10	41	0.561866	9.42E-08	14	36	0.75663	1.29E-07
	x_7	10	41	0.55687	1.11E-07	14	36	0.728991	1.52E-07
	x_8	10	41	0.562773	3.09E-07	14	36	0.703906	4.23E-07

is a solution of the linear complementarity problem if and only if it is a solution of the nonlinear equation:

$$F(z) = \min\{z, Dz + c\} = 0. \tag{4.4}$$

It was proved in [17, 25] that $F(z)$ is continuous and monotone. Therefore problem (4.1) can be translated into problem (1.1) and thus **MSCG** method can be applied to solve (4.1).

Table 8. Numerical Results for **MSCG** and **PCG** for Problem 8 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	0	1	0.001779	0	0	1	0.001328	0
	x_2	41	356	0.180046	9.5E-07	18	128	0.072022	5.55E-07
	x_3	40	352	0.166179	9.02E-07	19	137	0.082537	8.24E-07
	x_4	53	473	0.213444	7.73E-07	19	138	0.081841	4.96E-07
	x_5	31	284	0.142613	5.81E-07	45	313	0.167576	9.2E-07
	x_6	40	360	0.156393	8.3E-07	46	314	0.171545	7.72E-07
	x_7	31	279	0.123888	7.98E-07	32	219	0.125863	7.83E-07
	x_8	35	283	0.127523	9.52E-07	-	-	-	-
5000	x_1	0	1	0.001711	0	0	1	0.001672	0
	x_2	56	489	0.865066	8.21E-07	19	135	0.278625	4.25E-07
	x_3	44	383	0.655966	8E-07	20	144	0.304117	6.27E-07
	x_4	42	381	0.687588	7.45E-07	20	145	0.31177	3.73E-07
	x_5	28	257	0.438179	8.43E-07	48	331	0.665478	9.68E-07
	x_6	37	333	0.578789	7.75E-07	20	141	0.276562	4.24E-07
	x_7	30	270	0.4597	7.02E-07	29	199	0.384523	8.46E-07
	x_8	53	453	0.770932	8.01E-07	48	295	0.585834	7.16E-07
10000	x_1	0	1	0.003212	0	0	1	0.004846	0
	x_2	30	252	0.827242	7.1E-07	19	135	0.502917	6.37E-07
	x_3	52	445	1.444882	9.42E-07	20	144	0.544719	9.58E-07
	x_4	39	354	1.164713	7.61E-07	20	145	0.542192	5.52E-07
	x_5	27	248	0.825987	9.85E-07	48	331	1.314858	7.51E-07
	x_6	36	324	1.102443	7.47E-07	20	141	0.524563	3.81E-07
	x_7	30	270	0.88172	7.45E-07	27	187	0.696947	6.15E-07
	x_8	59	504	1.628672	7.67E-07	32	220	0.830439	6.31E-07
50000	x_1	0	1	0.01418	0	0	1	0.009638	0
	x_2	29	255	3.750516	8.09E-07	20	142	2.473295	5.46E-07
	x_3	46	398	5.801909	9.64E-07	21	151	2.648615	8.04E-07
	x_4	38	345	4.969609	9.49E-07	21	152	2.619938	4.6E-07
	x_5	26	239	3.471028	8.45E-07	42	293	5.07355	8.25E-07
	x_6	34	306	4.479681	7.04E-07	20	141	2.425991	7.18E-07
	x_7	28	252	3.612033	8.54E-07	24	167	2.9087	8.62E-07
	x_8	51	443	6.361079	9.55E-07	21	154	2.627603	4.8E-07
100000	x_1	0	1	0.027051	0	0	1	0.024137	0
	x_2	29	255	7.270722	9.08E-07	20	142	5.192622	8.61E-07
	x_3	44	368	10.87012	7.71E-07	22	158	5.5671	3.91E-07
	x_4	37	336	9.77921	9.4E-07	21	152	5.24037	7.3E-07
	x_5	26	239	7.044853	9.27E-07	43	299	10.94874	6.69E-07
	x_6	33	297	8.681605	8.44E-07	22	156	5.462871	5.41E-07
	x_7	27	243	7.180561	8.73E-07	23	161	5.708924	6.57E-07
	x_8	47	416	12.17001	7.77E-07	21	154	5.476608	6.48E-07

In this experiment, we consider a simple compressive sensing possible situation, where our goal is to reconstruct a sparse signal of length n from k observations. The quality of restoration is assessed by mean of squared error (MSE) to the original signal $\tilde{x}$,

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

where x_* is the recovered or restored signal. The signal size is chosen as $n = 2^{12}$, $k = 2^{10}$

Table 9. Numerical Results for **MSCG** and **PCG** for Problem 9 with given initial points and dimensions

		MSCG				**PCG**			
DIMENSION	INITIAL POINT	ITER	FVAL	TIME	NORM	ITER	FVAL	TIME	NORM
1000	x_1	13	66	0.007835	3.04E-07	10	36	0.008142	3.03E-07
	x_2	13	64	0.012285	6.68E-07	10	34	0.011219	6.61E-07
	x_3	13	64	0.012086	6.68E-07	10	34	0.011537	7.04E-08
	x_4	13	65	0.019151	6.68E-07	11	37	0.009902	3.94E-07
	x_5	13	64	0.009038	6.68E-07	11	38	0.012195	1.5E-07
	x_6	10	51	0.011598	5.96E-07	8	29	0.010377	4.31E-07
	x_7	12	61	0.012959	6.38E-07	9	32	0.008083	5.32E-07
	x_8	13	65	0.016145	6.68E-07	12	40	0.011124	9.64E-07
5000	x_1	13	66	0.039759	6.79E-07	10	36	0.029266	6.77E-07
	x_2	14	69	0.039939	3.18E-07	11	38	0.031945	6.23E-07
	x_3	14	69	0.036569	3.18E-07	10	34	0.028428	1.57E-07
	x_4	14	70	0.044959	3.18E-07	11	37	0.02743	8.81E-07
	x_5	14	69	0.04045	3.18E-07	11	38	0.029214	3.36E-07
	x_6	11	56	0.029751	2.84E-07	8	29	0.026799	9.65E-07
	x_7	13	66	0.036743	3.04E-07	10	36	0.025915	5.02E-07
	x_8	14	70	0.03723	3.18E-07	13	44	0.035078	9.09E-07
10000	x_1	13	66	0.093164	9.61E-07	10	36	0.042408	9.57E-07
	x_2	14	69	0.080892	4.5E-07	11	38	0.066548	8.82E-07
	x_3	14	69	0.055136	4.5E-07	10	34	0.049121	2.23E-07
	x_4	14	70	0.080006	4.5E-07	12	41	0.051811	5.26E-07
	x_5	14	69	0.084106	4.5E-07	11	38	0.052024	4.75E-07
	x_6	11	56	0.04438	4.02E-07	9	33	0.042029	5.75E-07
	x_7	13	66	0.074654	4.3E-07	10	36	0.048007	7.1E-07
	x_8	14	70	0.072395	4.5E-07	14	47	0.066632	8.67E-08
50000	x_1	14	71	0.222734	4.58E-07	11	40	0.165454	9.02E-07
	x_2	15	74	0.259981	2.15E-07	12	41	0.187299	1.33E-07
	x_3	15	74	0.218328	2.15E-07	10	34	0.148232	4.98E-07
	x_4	15	75	0.222071	2.15E-07	13	44	0.203704	7.93E-08
	x_5	15	74	0.230332	2.15E-07	12	42	0.1771	4.48E-07
	x_6	11	56	0.168722	8.99E-07	10	36	0.158327	8.68E-08
	x_7	13	66	0.190625	9.62E-07	11	39	0.175721	1.07E-07
	x_8	15	75	0.253142	2.15E-07	14	47	0.22066	1.94E-07
100000	x_1	14	71	0.462082	6.48E-07	12	43	0.381488	8.61E-08
	x_2	15	74	0.552861	3.04E-07	12	41	0.365063	1.88E-07
	x_3	15	74	0.538822	3.04E-07	10	34	0.295517	7.04E-07
	x_4	15	75	0.595913	3.04E-07	13	44	0.379492	1.12E-07
	x_5	15	74	0.51213	3.04E-07	12	42	0.359235	6.34E-07
	x_6	12	61	0.461306	2.71E-07	10	36	0.305215	1.23E-07
	x_7	14	71	0.507379	2.9E-07	11	39	0.373767	1.51E-07
	x_8	15	75	0.525749	3.04E-07	14	47	0.42366	2.74E-07

and the original signal contains 2^7 randomly nonzero elements. A is the Gaussian matrix generated by the command $rand(m,n)$ in MATLAB. In addition, the measurement y is distributed with noise, that is, $y = A\tilde{x} + \eta$, where η is the Gaussian noise distributed normally with mean 0 and variance 10^{-4} ($N(0, 10^{-4})$).

To show the performance of the **MSCG** method in compressive sensing, we compare it with the **PCG** method. The parameters in both **MSCG** and **PCG** methods are chosen as $\beta = 1$, $\sigma = 10^{-4}$, $\rho = 0.8$, and $r = 0.1$ and the merit function used is

$f(x) = \frac{1}{2}\|y - Ax\|_2^2 + \tau\|x\|_1$. To achieve fairness in comparison, each code was run from same initial point, same continuation technique on the parameter τ, and observed only the behaviour of the convergence of each method to have a similar accurate solution. The experiment is initialized by $x_0 = A^T y$ and terminates when

$\frac{\|f_k - f_{k-1}\|}{\|f_{k-1}\|} < 10^{-5}$,

where f_k is the function evaluation at x_k.

In Fig. 1, **MSCG** and **PCG** methods recovered the disturbed signal almost exactly. In order to show visually the performance of both methods, four figures were plotted to demonstrate their convergence behaviour based on MSE, objective function values, number of iterations and CPU time (see Fig. 2-5). Furthermore, the experiment was repeated for 25 different noise samples (see Table 10). From the Table, it can be observed that the **MSCG** is more efficient in terms of iterations and CPU time than **PCG** method in most cases.

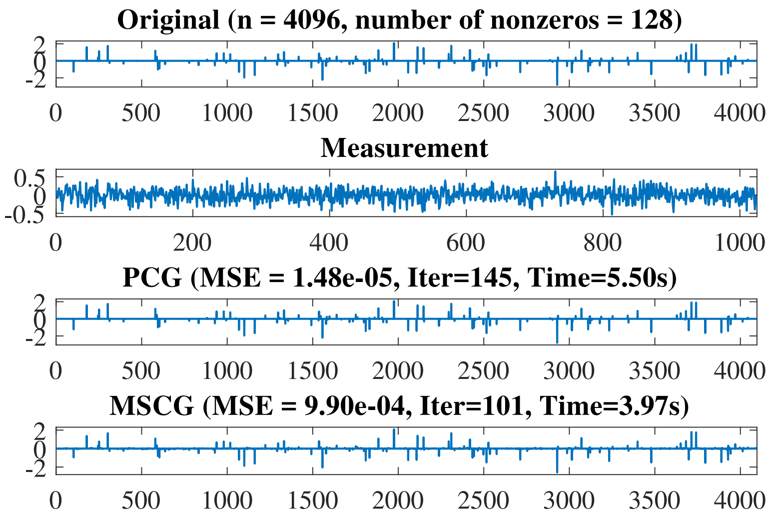

FIGURE 1. From top to bottom: the original image, the measurement, and the recovered signals by **PCG** and **MSCG** methods.

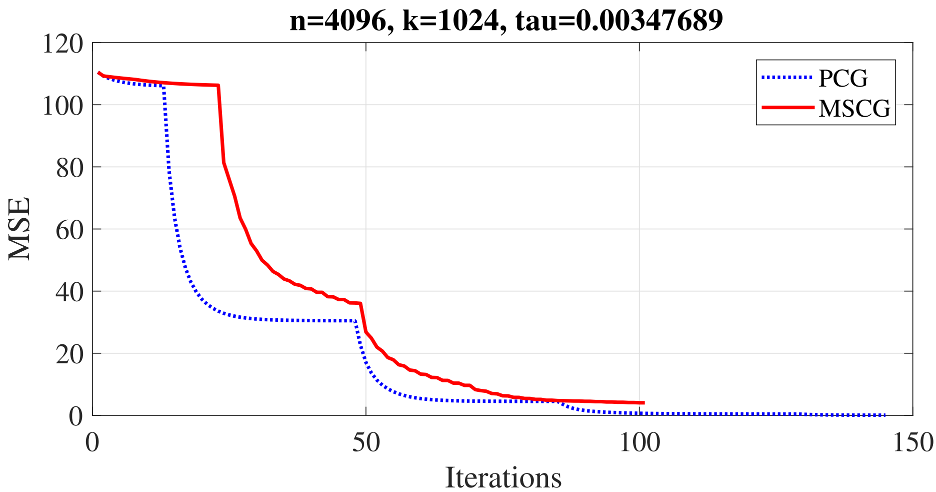

FIGURE 2. Iterations

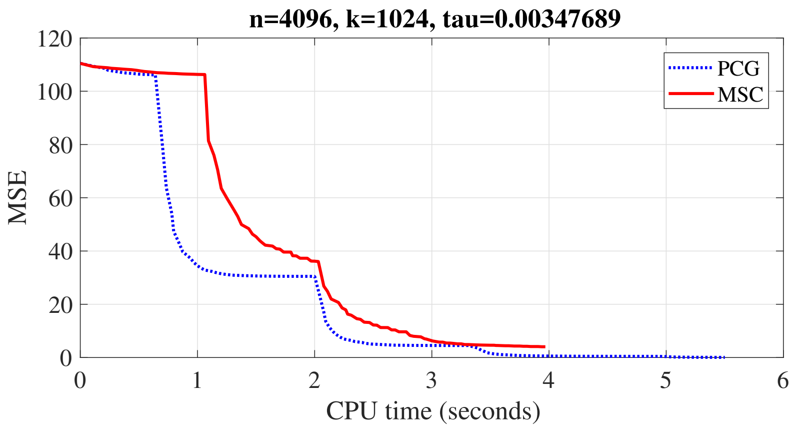

FIGURE 3. CPU time (seconds)

5. Conclusions

In this paper, a modified three-term conjugate gradient method for solving monotone nonlinear equations with convex constraints was presented. The proposed algorithm is suitable for solving non-smooth equations because it requires no Jacobian information of the nonlinear equations. Under some assumptions, global convergence properties of the proposed method was proved. Numerical experiments presented clearly shows how effective the **MSCG** algorithm is compared to the **PCG** algorithm of [14] for the given constrained problems. In addition, the **MSCG** algorithm was also shown to be effective in signal recovery problems.

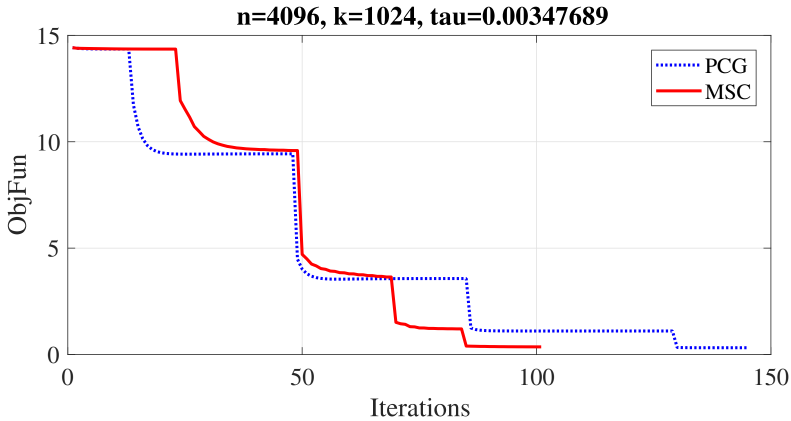

FIGURE 4. Iterations

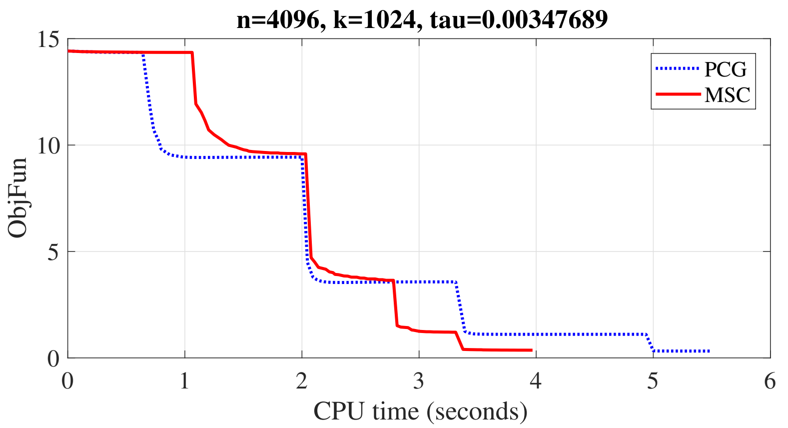

FIGURE 5. CPU time (seconds)

ACKNOWLEDGEMENTS

- The authors acknowledge the financial support provided by King Mongkut's University of Technology Thonburi through the "KMUTT 55th Anniversary Commemorative Fund".
- The first author was supported by the Petchra Pra Jom Klao Doctoral Scholarship Academic for Ph.D. Program at KMUTT. This project is supported by the **T**heoretical and **C**omputational **S**cience (TaCS) **C**enter under **C**omputational and **A**pplied **S**cience for **S**mart research **I**nnovation (CLASSIC), Faculty of Science, KMUTT.
- Moreover, this project was partially supported by the Thailand Research Fund (TRF) and the King Mongkut's University of Technology Thonburi (KMUTT) under the TRF Research Scholar Award (Grant No. RSA6080047).

Table 10. Twenty five experiment results of ℓ_1-norm regularization problem for **MSCG** and **PCG** methods

	MSCG			PCG		
	MSE	ITER	CPU(s)	MSE	ITER	CPU(s)
	9.90E-04	101	3.97	1.48E-05	145	5.5
	1.58E-03	103	3.31	1.63E-05	140	4.5
	7.68E-04	113	3.17	1.30E-05	133	3.47
	1.07E-03	128	3.45	1.70E-05	145	3.95
	1.23E-03	122	3.2	1.38E-05	143	3.77
	1.62E-03	88	2.34	1.48E-05	139	3.72
	1.66E-03	114	3.19	1.84E-05	132	3.59
	2.63E-03	95	2.75	1.83E-05	123	3.41
	1.16E-03	99	2.67	1.22E-05	113	2.92
	1.91E-03	107	2.84	1.79E-05	114	2.92
	2.18E-03	106	2.69	2.09E-05	110	2.81
	8.60E-04	107	2.77	1.63E-05	131	3.38
r=0.1	1.33E-03	102	2.78	1.27E-05	143	3.78
	1.03E-03	119	4.53	1.06E-05	140	5.34
	1.15E-03	110	3.03	1.48E-05	135	3.61
	1.77E-03	110	4.27	1.69E-05	148	5.75
	1.36E-03	103	3.83	1.47E-05	114	4.34
	1.67E-03	112	3.42	1.78E-05	120	3.88
	1.21E-03	107	4.38	1.47E-05	114	4.91
	9.99E-04	101	3.86	1.47E-05	145	5.55
	1.58E-03	103	2.78	1.63E-05	140	3.7
	7.68E-04	113	3.16	1.30E-05	133	3.92
	1.07E-03	128	4.77	1.70E-05	145	5.59
	1.23E-03	122	3.23	1.38E-05	143	3.91
	1.62E-03	88	2.41	1.48E-05	139	3.81

References

[1] Mehiddin Al-Baali, Yasushi Narushima, and Hiroshi Yabe. A family of three-term conjugate gradient methods with sufficient descent property for unconstrained optimization. *Computational Optimization and Applications*, 60(1):89–110, Jan 2015.

[2] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[3] Stefania Bellavia, Maria Macconi, and Benedetta Morini. Strscne: A scaled trust-region solver for constrained nonlinear equations. *Computational Optimization and Applications*, 28(1):31–50, Apr 2004.

[4] Yang Bing and Gao Lin. An efficient implementation of merrill's method for sparse or partially separable systems of nonlinear equations. *SIAM Journal on Optimization*, 1(2):206–221, 1991.

[5] Ernesto G Birgin, José Mario Martínez, and Marcos Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization*, 10(4):1196–1211, 2000.

[6] Kanzow C., Yamashita N., and Fukushima M. Levenberg–marquardt methods with strong local convergence properties for solving nonlinear equations with convex constraints. *Journal of Computational and Applied Mathematics*, 172(2):375–397, 2004.

[7] Mário AT Figueiredo and Robert D Nowak. An EM algorithm for wavelet-based image restoration. *IEEE Transactions on Image Processing*, 12(8):906–916, 2003.

[8] Mário AT Figueiredo, Robert D Nowak, and Stephen J Wright. Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. *IEEE Journal of selected topics in signal processing*, 1(4):586–597, 2007.

[9] W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16(1):170–192, 2005.

[10] W. Hager and H. Zhang. A new conjugate gradient method with guaranteed descent and an efficient line search. *SIAM Journal on Optimization*, 16(1):170–192, 2005.

[11] William W Hager and Hongchao Zhang. A survey of nonlinear conjugate gradient methods. *Pacific journal of Optimization*, 2(1):35–58, 2006.

[12] Elaine T Hale, Wotao Yin, and Yin Zhang. A fixed-point continuation method for ℓ_1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*, 43:44, 2007.

[13] William La Cruz. A spectral algorithm for large-scale systems of nonlinear monotone equations. *Numerical Algorithms*, 76(4):1109–1130, 2017.

[14] JK Liu and SJ Li. A projection method for convex constrained monotone nonlinear equations with applications. *Computers & Mathematics with Applications*, 70(10):2442–2453, 2015.

[15] Y. Narushima, H. Yabe, and J. Ford. A three-term conjugate gradient method with sufficient descent property for unconstrained optimization. *SIAM Journal on Optimization*, 21(1):212–230, 2011.

[16] Yasushi Narushima. A smoothing conjugate gradient method for solving systems of nonsmooth equations. *Applied Mathematics and Computation*, 219(16):8646–8655, 2013.

[17] Jong-Shi Pang. Inexact Newton methods for the nonlinear complementarity problem. *Mathematical Programming*, 36(1):54–71, 1986.

[18] M. Solodov and B. Svaiter. A new projection method for variational inequality problems. *SIAM Journal on Control and Optimization*, 37(3):765–776, 1999.

[19] Michael V Solodov and Benav F Svaiter. A globally convergent inexact newton method for systems of monotone equations. In *Reformulation: Nonsmooth, Piecewise Smooth, Semismooth and Smoothing Methods*, pages 355–369. Springer, 1998.

[20] Kaori Sugiki, Yasushi Narushima, and Hiroshi Yabe. Globally convergent three-term conjugate gradient methods that use secant conditions and generate descent search directions for unconstrained optimization. *Journal of Optimization Theory and Applications*, 153(3):733–757, Jun 2012.

[21] Ewout Van Den Berg and Michael P Friedlander. Probing the pareto frontier for basis pursuit solutions. *SIAM Journal on Scientific Computing*, 31(2):890–912, 2008.

[22] Chuanwei Wang and Yiju Wang. A superlinearly convergent projection method for constrained systems of nonlinear equations. *Journal of Global Optimization*, 44(2):283–296, Jun 2009.

[23] Chuanwei Wang, Yiju Wang, and Chuanliang Xu. A projection method for a system of nonlinear monotone equations with convex constraints. *Mathematical Methods of Operations Research*, 66(1):33–46, Aug 2007.

[24] X. Y. Wang, S. J. Li, and X. P. Kou. A self-adaptive three-term conjugate gradient method for monotone nonlinear equations with convex constraints. *Calcolo*, 53(2):133–145, Jun 2016.

[25] Yunhai Xiao, Qiuyu Wang, and Qingjie Hu. Non-smooth equations based method for ℓ_1-norm problems with applications to compressed sensing. *Nonlinear Analysis: Theory, Methods & Applications*, 74(11):3570–3577, 2011.

[26] Yunhai Xiao and Hong Zhu. A conjugate gradient method to solve convex constrained monotone equations with applications in compressive sensing. *Journal of Mathematical Analysis and Applications*, 405(1):310–319, 2013.

[27] Li Zhang, Weijun Zhou, and Dong-Hui Li. A descent modified polak–ribière–polyak conjugate gradient method and its global convergence. *IMA Journal of Numerical Analysis*, 26(4):629–640, 2006.

[28] Li Zhang, Weijun Zhou, and Donghui Li. Some descent three-term conjugate gradient methods and their global convergence. *Optimization Methods and Software*, 22(4):697–711, 2007.

[29] Weijun Zhou and Donghui Li. Limited memory BFGS method for nonlinear monotone equations. *Journal of Computational Mathematics*, 25(1):89–96, 2007.

Bangmod International
Journal of Mathematical Computational Science
ISSN: 2408-154X
Bangmod-JMCS Online @ http://bangmod-jmcs.kmutt.ac.th/

Journal office:
Theoretical and Computational Science Center (TaCS)
Science Laboratory Building, Faculty of Science
King Mongkuts University of Technology Thonburi (KMUTT)
126 Pracha Uthit Road, Bang Mod, Thung Khru, Bangkok, Thailand 10140
Website: http://tacs.kmutt.ac.th/
Email: tacs@kmutt.ac.th