THE PROXIMAL METHODS FOR SOLVING
ABSOLUTE VALUE EQUATION

SAMIRA SHAHSAVARI AND SAEED KETABCHI*

Department of Applied Mathematics
Faculty of Mathematical Sciences
University of Guilan, Rasht, Iran

(Communicated by Yongzhong Song)

Abstract. In this paper, by considering that the objective function of the least squares NP-hard absolute value equations (AVE) \(Ax - |x| = b\), is non-convex and non-smooth, two types of proximal algorithms are proposed to solve it. One of them is the proximal difference-of-convex algorithm with extrapolation and another is the proximal subgradient method. The convergence results of the proposed methods are proved under certain assumptions. Moreover, a numerical comparison is presented to demonstrate the effectiveness of the suggested methods.

1. Introduction. Recently, the problem of finding a solution for absolute value equations (AVE)
\[Ax - |x| = b,\] (1)
where \(A \in \mathbb{R}^{n \times n}\), \(b \in \mathbb{R}^n\), and \(|x|\) denotes the component-wise absolute value of vector \(x \in \mathbb{R}^n\), has led to the use of many algorithms. For example, Mangasarian proposed generalized Newton method [11], successive linear programming method [12], and primal-dual bilinear programming algorithm [13], for solving AVE (1). Zhang and Wei in [24] exhibited generalized Newton method with combining the semi-smooth and the smoothing Newton steps, and Li [9] used a modified generalized Newton method when all the singular values of \(A\) exceed 1. Ning et al. [16] considered improved adaptive differential evolution for solving absolute value equations; in this algorithm, they use global and local search. Cruz et al. [1] used the inexact non-smooth Newton method, and established global linear convergence of the method. Salkuyeh [22] considered the PicardHSS iteration method and provided sufficient conditions for the convergence of this method, and Edalatpour et al. [4] presented a generalization of the GaussSeidel iteration method for solving AVE (1). Finally, Moosaei et al. [15] introduced two methods for solving AVE (1), i.e., the Homotopy perturbation method, and the Newton method with the Armijo step. They indicated that solving AVE (1) is equivalent to solving the unconstrained optimization problem
\[
\min_{x \in \mathbb{R}^n} F(x) = \frac{1}{2} \|Ax - |x| - b\|^2.
\] (2)

2010 Mathematics Subject Classification. Primary: 90C26; Secondary: 90C25, 65D10.

Key words and phrases. Absolute value equation, The proximal difference-of-convex algorithm, The proximal subgradient method, Non-convex and non-smooth.

* Corresponding author: Saeed Ketabchi.
The objective function of the problem (2) is non-convex, non-smooth and level-bounded; however, the Newton method with the Armijo step was applied without using the non-convexity property of the problem (2). Therefore, in this paper, to consider the non-convexity property of the problem (2), we rewrite the objective function of this problem in two ways:

1- Sum of a smooth convex function which has Lipschitz-continuous gradient with modulus \( L > 0 \), a proper closed convex function, and a continuous convex function.

2- Sum of a convex function and a non-differentiable function.

Thus, in this paper, we obtain two new problems which are equivalent to the problem (2) in Sections 2 and 3, and solve them by the proximal algorithms. In Section 4, we present the numerical results and compare the performance of the algorithms with the Friedman test. In Section 5, we present our conclusion.

In this paper, \( \mathbb{R}^n \) is the \( n \)-dimension Euclidean space, \( \| \cdot \| \) and \( \| \cdot \|_\infty \) denote 2-norm and infinity-norm, respectively. The identity matrix of size \( n \) is denoted by \( I \). \( A^T \) represents transpose of matrix \( A \in \mathbb{R}^{n \times n} \). \( x^T x \) denotes inner product of vector \( x \in \mathbb{R}^n \), and \((x)_+ \) indicates a vector of elements \( \max(0, x_i) \) for \( i = 1, \ldots, n \). For \( x \in \mathbb{R}^n \), \( \text{sign}(x) \) denotes a vector with components equal to 1, 0, \(-1\) depending on whether the corresponding component of \( x \) is positive, zero or negative, respectively. For a function \( f : \mathbb{R}^n \rightarrow [\infty, \infty] \), the domain of this function is denoted by \( \text{dom}(f) \). The function \( f \) is said to be proper if it never equals \(-\infty \) and \( \text{dom}(f) \neq \emptyset \) and, this proper function is closed if it is lower semicontinuous. If \( f \) is differentiable at point \( x \in \mathbb{R}^n \), then the gradient of this function at \( x \) is denoted by \( \nabla f(x) \). Otherwise, we denote the generalized gradient of \( f \) at \( x \) by \( \partial f(x) \) and define it as below:

\[
\partial f(x) = \{ \xi \in \mathbb{R}^n : f(y) \geq f(x) + \xi^T (y - x), \forall y \in \text{dom}(f) \},
\]

and finally, for given \( \sigma > 0 \), \( \text{prox}_{\sigma f}(y) \) is called the proximal operator of \( \sigma f \) at \( y \in \mathbb{R}^n \) and define as follows:

\[
\text{prox}_{\sigma f}(y) = \text{arg} \min_{x \in \mathbb{R}^n} f(x) + \frac{\sigma}{2} \| x - y \|^2.
\]

2. The proximal difference-of-convex algorithm with extrapolation for AVE. The proximal difference-of-convex algorithm (PDCA) has been presented to solve a specific class of difference-of-convex (DC) optimization problems, whose the objective function is the sum of a smooth convex function with Lipschitz gradient, a proper closed convex function and a continuous concave function (see [5]). Then, this algorithm was developed by Wen et al. in [23]. They used the extrapolation techniques for accelerating PDCA, and therefore, introduced the proximal difference-of-convex algorithm with extrapolation (PDCAe).

The objective function of problem (2) can be reformulated as follows:

\[
F(x) = \frac{1}{2} \| Ax - b \|^2 + \frac{1}{2} \| x \|^2 + (b - Ax)_+^T | x | - (Ax - b)_+^T | x |.
\]

Letting \( S = Ax - b \), and the fact that \((t)_+ = \frac{1}{2}(|t| + t)\) for every \( t \in \mathbb{R} \), we get

\[
F(x) = \frac{1}{2} \| Ax - b \|^2 + \frac{1}{2} \| x \|^2 + \frac{1}{2} (| - S | - S)^T | x | - \frac{1}{2} (| S | + S)^T | x |.
\]
Since the function $F$ is non-smooth, we use the following two smooth approximate functions for $|t|$ (see [2, 6, 20]):

$$\phi_1(t, \mu) = \begin{cases} 
-t, & \text{if } t < -\frac{\mu}{2}, \\
\frac{t^2}{\mu} + \frac{\mu}{4}, & \text{if } -\frac{\mu}{2} \leq t \leq \frac{\mu}{2}, \\
t, & \text{if } t > \frac{\mu}{2}.
\end{cases}$$

$$\phi_2(t, \mu) = \begin{cases} 
-t - \frac{\mu}{2}, & \text{if } t < -\mu, \\
\frac{t^2}{2\mu}, & \text{if } -\mu \leq t \leq \mu, \\
t - \frac{\mu}{2}, & \text{if } t > \mu.
\end{cases}$$

where $\mu > 0$ is a smoothing parameter. For fixed $\mu > 0$, there are first-order derivatives of $\phi_1$ and $\phi_2$, and their gradients are Lipschitz continuous (see [20, 2]). Hence, the function $F$ is approximated by smooth functions using the smoothing functions $\phi_i$, $i = 1, 2$, as follows:

$$F(x) = \frac{1}{2} \|Ax - b\|^2 + \frac{1}{2} \|x\|^2 + \frac{1}{2}(\Phi_i(-S) - S)^t\Phi_i(x) - \frac{1}{2}(\Phi_i(S) + S)^t\Phi_i(x),$$

(4)

where $\Phi_i(x) = (\phi_i(x_1, \mu), \phi_i(x_2, \mu), \ldots, \phi_i(x_n, \mu))$, $i = 1, 2$. So, problem (2) is smoothed as following:

$$\min_{x \in \mathbb{R}^n} F(x)$$

(5)

In the following proposition, we prove that problem (5) is a special class of DC programming problem, and we also indicate that the sequence of points generated by PDCAe converges to a stationary point of the problem (2).

**Proposition 1.** (i) Problem (5) is a DC programming problem that its objective is the sum of a smooth convex function with Lipschitz gradient, a proper closed convex function and a continuous concave function.

(ii) Suppose $\{x_k\}$ is a sequence generated by PDCAe, then any accumulation point of $\{x_k\}$ is a stationary point of problem (5).

**Proof.** (i) The optimization problem (2) can be rewritten as follows:

$$\min_{x \in \mathbb{R}^n} F(x) = f(x) + p(x) - h(x),$$

(6)

where

$$\begin{align*}
f(x) &= \frac{1}{2}(\Phi_i(-S) - S)^t\Phi_i(x), \\
p(x) &= \frac{1}{2}\|Ax - b\|^2 + \frac{1}{2}\|x\|^2, \\
h(x) &= -\frac{1}{2}(\Phi_i(S) + S)^t\Phi_i(x).
\end{align*}$$

(7)

Clearly $f$ is a smooth convex function with Lipschitz continuous gradient with modulus $L = 2\mu(\|A\| + \|b\|)$, also, $p$ is a proper closed convex function, and $h$ is a continuous concave function.

(ii) It can be proved similar to [23].

□
To execute PDCA, we need to calculate the gradient of the smooth convex function \( h(x) \) at \( x_k \), which is calculated as follows:

\[
\zeta_k = \nabla h(x_k) = \frac{1}{2} A^t(\nabla S \Phi_i(S_k) + I_n)^t \Phi_i(x_k) + \frac{1}{2} \nabla \Phi_i(x_k)(\Phi_i(S_k) + S_k),
\]

and the following optimization problem is solved to obtain the new iteration

\[
x_{k+1} = \text{prox}_{L \rho}(y_k - \frac{1}{L}(\nabla f(y_k) - \zeta_k))
\]

\[
= \arg \min_{x \in \mathbb{R}^n} p(x) + (\nabla f(y_k) - \zeta_k)^t x + \frac{L}{2} \|x - y_k\|^2
\]

\[
= \arg \min_{x \in \mathbb{R}^n} \frac{1}{2}\|Ax - b\|^2 + \frac{1}{2}\|x\|^2 + (\nabla f(y_k) - \zeta_k)^t x + \frac{L}{2} \|x - y_k\|^2,
\]

where \( y_k = x_k + \beta_k(x_k - x_{k-1}) \), and \( \beta_k \) is the extrapolation parameter presented in the article [23].

If we define \( \Psi(x) = \frac{1}{2} \|Ax - b\|^2 + \frac{1}{2} \|x\|^2 + (\nabla f(y_k) - \zeta_k)^t x + \frac{L}{2} \|x - y_k\|^2 \), then, PDCA method for solving AVE can be written as follows:

- **Algorithm 1 : PDCA for AVE**
  1. Choose the parameters \( \mu_0, \theta_0, \theta_1, \epsilon > 0 \), and \( \{\beta_k\} \subseteq [0,1) \) with \( \sup_k \beta_k < 1 \), and the starting point \( x_0 \in \text{dom}(p) \). Set \( x_{-1} = x_0 \) and \( k = 0 \).
  2. **While** \( \|F(x_k)\|_{\infty} > \epsilon \) **do**
    - Calculate \( \zeta_k \) from
      \[
      \zeta_k = \frac{1}{2} A^t(\nabla S \Phi_i(S_k) + I_n)^t \Phi_i(x_k) + \frac{1}{2} \nabla \Phi_i(x_k)(\Phi_i(S_k) + S_k).
      \]
    - Set \( y_k = x_k + \beta_k(x_k - x_{k-1}) \) and calculate \( \nabla f(y_k) \).
    - Calculate \( x_{k+1} \) by solving the problem
      \[
      \min_{x \in \mathbb{R}^n} \Psi(x) = \frac{1}{2} \|Ax - b\|^2 + \frac{1}{2} \|x\|^2 + (\nabla f(y_k) - \zeta_k)^t x + \frac{L}{2} \|x - y_k\|^2.
      \]
    - Update the smoothing parameter \( \mu_k \) as:
      - If \( \|\Psi(x_{k+1})\|_{\infty} > \theta_1 \mu_k \), set \( \mu_{k+1} = \mu_k \), else set \( \mu_{k+1} = \theta_2 \mu_k \).
      - If \( x_{k+1} = x_k \) then **break**, and \( x_{k+1} \) is a solution, otherwise set \( k = k + 1 \).
  3. **end**

3. **Proximal subgradient method for solving AVE.** In this section, we use the proximal subgradient method (PSM) for solving the problem (2). This algorithm is proposed by Lions and Mercier [10] and Passty in [17]. For further reading see [14] and references therein.

We rewrite the objective function of the problem (2) as:

\[
F(x) = g(x) + f(x),
\]

where the functions \( g, f : \mathbb{R}^n \rightarrow \mathbb{R} \) are defined as follows:

\[
\begin{align*}
g(x) &= \frac{1}{2} \|Ax - |x||^2 - b^t Ax + (b)_+^t x + \frac{\gamma}{2} \|x||^2, \\
f(x) &= -(-b)_+^t x - \frac{\gamma}{2} \|x||^2,
\end{align*}
\]

in which the parameter \( \gamma \) is positive. Clearly the function \( f \) is a concave. In the following proposition we prove that the function \( g \) is convex for \( \gamma \geq 0 \).

**Proposition 2.** There exists a \( \gamma > 0 \) such that the function \( g \) is convex.
The Proximal Methods for Solving Absolute Value Equation

Proof. We rewrite the function \( g \) as follows:

\[
g(x) = k_1(x) + k_2(x),
\]

where

\[
\begin{align*}
k_1(x) &= -x^tAx|_{x} + \frac{\gamma}{2} \|x\|^2, \\
k_2(x) &= \frac{1}{2} \|Ax\|^2 + \frac{1}{2} \|x\|^2 - b^tAx + (b)^t_+ |x|,
\end{align*}
\]

Clearly, \( k_2(x) \) is a convex function. So, it is enough to prove that the function \( k_1(x) \) is convex for sufficiently large values of \( \gamma > 0 \). To do this, we define the matrix \( B_x \) as

\[
B_x = \frac{\gamma}{2} I - \frac{1}{2} (AD_x + D_x A').
\]  

Then, we have

\[
k_1(x) = \frac{\gamma}{2} \|x\|^2 - (\frac{1}{2} x^t(AD_x)x + \frac{1}{2} x^t(AD_x)^tx)
= x^t(\frac{\gamma}{2} I - \frac{1}{2} (AD_x + D_x A'))x
= x^tB_x x.
\]

Therefore,

\[
k_1(x) = x^tB_x x
= \sum_{i=1}^{n}((\frac{\gamma}{2} - A_{ii}\text{sign}(x_i))x_i^2) - \sum_{i=1}^{n} \sum_{j=1}^{n} (A_{ij}\text{sign}(x_i) + A_{ji}\text{sign}(x_j))x_i x_j,
\]

where \( A_{ij}, i, j = 1, \ldots, n \) are the entries of the matrix \( A \). Since, \( \text{sign}(x_i) = -1 \) or 1 or 0, \( i = 1, \ldots, n \), then, \( A_{ij}\text{sign}(x_i) \in \mathbb{R}, i, j = 1, \ldots, n \). So, there exist \( \alpha_i, \lambda_i \in \mathbb{R}, i = 1, \ldots, n \) such that \( k_1 \) is converted to

\[
k_1(x) = \sum_{i=1}^{n}((\frac{\gamma}{2} - \alpha_i)x_i^2 + \lambda_i x_i).
\]

Therefore, for sufficiently large values of \( \gamma > 0 \), the function \( k_1(x) \) is convex.

Now, Let \( x_0 \) be an initial point and \( x_0 \in \text{dom}(g) \). Since \( f \) is Lipschitz continuous at \( x_k \), its generalized gradient exists and is calculated as follows:

\[

\xi_k = -D_k(-b)_+ - \gamma x_k,
\]

where \( D_k = \text{diag}(\text{sign}(x_k)) \). Hence, we calculate the proximal of the function \( g \) at the point \( (x_k - \frac{1}{\sigma} \xi_k) \) as follows:

\[
x_{k+1} = \text{prox}_{\sigma g}(x_k - \frac{1}{\sigma} \xi_k) = \arg \min_{x \in \mathbb{R}^n} g(x) + \frac{\sigma}{2} \|x - (x_k - \frac{1}{\sigma} \xi_k)\|^2,
\]

and from (11), we have

\[
x_{k+1} = \arg \min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - |x|\|^2 - b^tAx + (b)^t_+ |x| + \frac{\gamma}{2} \|x\|^2 + \xi_k^t x + \frac{\sigma}{2} \|x - x_k\|^2.
\]  

The problem (14) is convex and can be solved efficiently using convex optimization methods.
Now, we state the proximal subgradient method for solving absolute value equations:

Algorithm 2: PSM for AVE
1- Choose $x_0 \in \text{dom}(g)$, $\epsilon > 0$, and set $k = 0$.
2- while $\|c_k\|_\infty \geq \epsilon$ do
   Solve the convex subproblem below and obtain the solution $y_k$,
   $$\min_{x \in \mathbb{R}^n} \frac{1}{2} \|Ax - |x|\|^2 - b^T Ax + (b)_+^T |x| + \frac{\gamma}{2} \|x\|^2 + \xi_k x + \frac{\sigma}{2} \|x - x_k\|^2.$$ 
   Set $c_k = y_k - x_k$.
   Put $x_{k+1} = y_k$ and $k = k + 1$.
3- end.

In this algorithm, the step length $\sigma$ is a constant positive number ($\sigma = \gamma$). In the following theorem, we prove the local convergence of Algorithm 2.

Theorem 3.1. Let $\{x_k\}$ be the sequence generated by Algorithm 2, then

(i) For all $k$, $F(x_{k+1}) < F(x_k)$.
(ii) The sequence $\{x_k\}$ is bounded.
(iii) Suppose $\pi$ is an accumulation point of the sequence $\{x_k\}$, then $\pi$ is a stationary point of $F$.

Proof. (i) Since $x_{k+1} = \text{prox}_{\sigma g}(x_k - \frac{1}{\sigma} \xi_k)$, we have

$$\frac{1}{2} \|Ax_{k+1} - |x_{k+1}|\|^2 - b^T Ax_{k+1} + (b)_+^T |x_{k+1}| + \frac{\gamma}{2} \|x_{k+1}\|^2$$

$$- (b)_+^T D_k x_{k+1} - \gamma x_k^T x_{k+1} + \frac{\sigma}{2} \|x_{k+1} - x_k\|^2$$

$$\leq \frac{1}{2} \|Ax_k - |x_k|\|^2 - b^T Ax_k + (b)_+^T |x_k| + \frac{\gamma}{2} \|x_k\|^2 - (b)_+^T D_k x_k - \gamma x_k^T x_k$$

$$= \frac{1}{2} \|Ax_k - |x_k|\|^2 - b^T Ax_k + b^T |x_k| - \frac{\gamma}{2} \|x_k\|^2$$

$$= F(x_k) - \frac{\gamma}{2} \|x_k\|^2 - \frac{1}{2} \|b\|^2. \quad (15)$$

On the other hand, we have

$$\frac{1}{2} \|Ax_{k+1} - |x_{k+1}|\|^2 - b^T Ax_{k+1} + (b)_+^T |x_{k+1}| + \frac{\gamma}{2} \|x_{k+1}\|^2 - (b)_+^T D_k x_{k+1}$$

$$- \gamma x_k^T x_{k+1} + \frac{\sigma}{2} \|x_{k+1} - x_k\|^2$$

$$= \frac{1}{2} \|Ax_{k+1} - |x_{k+1}|\|^2 - b^T Ax_{k+1} + (b)_+^T |x_{k+1}| - (b)_+^T |x_{k+1}|$$

$$+ (b)_+^T |x_{k+1}| + \frac{\gamma}{2} \|x_{k+1}\|^2 - (b)_+^T D_k x_{k+1} - \gamma x_k^T x_{k+1} + \frac{\gamma}{2} \|x_k\|^2$$

$$- \frac{\gamma}{2} \|x_k\|^2 - \frac{1}{2} \|b\|^2 + \frac{1}{2} \|b\|^2$$

$$= F(x_{k+1}) - (b)_+^T |x_{k+1}| + \frac{\gamma}{2} \|x_{k+1} - x_k\|^2 - (b)_+^T D_k x_{k+1}$$

$$- \frac{\gamma}{2} \|x_k\|^2 - \frac{1}{2} \|b\|^2$$

$$\geq F(x_{k+1}) - \frac{\gamma}{2} \|x_k\|^2 - \frac{1}{2} \|b\|^2. \quad (16)$$
We conclude from inequalities (15) and (16) that
\[ F(x_{k+1}) \leq F(x_k) + \frac{\gamma}{2} \|x_{k+1} - x_k\|^2 \leq F(x_k). \] (17)

(ii) From (17) we can conclude that
\[ 0 \leq F(x_{k+1}) \leq F(x_{k+1}) + \frac{\gamma}{2} \|x_{k+1} - x_k\|^2 \leq F(x_0). \]
Therefore, from part (i) and the above-mentioned inequality, we achieve the conclusion that the sequence \( \{x_k\} \) is bounded.

(iii) We note that, when \( x_k \to x \) then \( D_k \to D \), where \( D_k = \text{diag}(\text{sign}(x_k)) \) and \( D = \text{diag}(\text{sign}(x)) \). Since \( x_{k+1} = \text{prox}_{\sigma g}(x_k - \frac{1}{\sigma} \xi_k) \), so
\[ 0 \in \partial g(x_{k+1}) + \sigma(x_{k+1} - (x_k - \frac{1}{\sigma} \xi_k)), \]
or
\[ 0 = (A - D_{k+1})^t(A - D_k)x_{k+1} - A^t b + D_{k+1}(b)_+ + \gamma x_{k+1} + \sigma(x_{k+1} - (x_k - \frac{1}{\sigma} \xi_k)). \]
Now if \( k \to +\infty \), then
\[ 0 = (A - D)^t(A - D)x - A^t b + D(b)_+ - D(-b)_+ \]
\[ \implies 0 = (A - D)^t(A - D)x - A^t b + Db \]
\[ \implies 0 = (A - D)^t((A - D)x - b) \]
\[ \implies 0 \in \partial F(x). \]

4. Numerical experiments. In this section, we report the numerical results for testing our algorithms PSM and PDCA, and the Newton method with the Armijo step (NMA) introduced in [15] on a numerical example which is generated randomly with a personal computer (CORE i5, CPU 2.50 GHZ, 4 GB memory, MATLAB 2013), and then we use the Friedman test to compare the performance (accuracy and running time) of these algorithms. To compare algorithms, we select the tolerance parameter \( \epsilon \) equal to \( 10^{-6} \).

We consider numerical examples of AVE (1) which have been generated by the following MATLAB code:

```
n=input('Enter n: ');
d=input('Enter d in (0,1]:');
A = 100 * sprandsym(n,d);
A = (A - .5 * spones(A));
x = .01 * (rand(n,1) - rand(n,1));
b = A * x - abs(x);
```

These examples are generated 10 times for each dimension. Then, we present the average numerical results for some dimensions in Table 1. We present the numerical results for \( n = 100, 200, \ldots, 7500 \), \( d = 1 \) and for \( n = 8000, d = 0.1 \).

Table 1 provides the following information:
- \( n \): represents the size of matrix \( A \);
- \( k \): is the average number of iterations of algorithms;
- Time(s): is the average elapsed CPU times;
\[ f^* = \frac{1}{10} \sum_{i=1}^{10} (||Ax^* - |x^*| - b||_\infty): \text{ where } x^* \text{ is the solution obtained from the algorithm, and if } f^* > 10^{-6}, \text{ or } \text{Time} > 600 \text{ s, then instead of } f^* \text{ and Time(s)} \text{ and } k, \text{ we put "-".} \]

PDCA\(_e(\phi_1)\) and PDCA\(_e(\phi_2)\): represent PDCA\(_e\) that using the smoothing functions \(\phi_1\) and \(\phi_2\), respectively.

Table 1. Numerical results

| \(n\) | PSM | PDCA\(_e(\phi_1)\) | PDCA\(_e(\phi_2)\) | NMA |
|------|-----|------------------|------------------|-----|
| 100  | 3   | 1.343e-16 0.00  | 6 6.154e-09 0.01 | 6   | 6.944e-09 0.01 | 3   | 3.823e-14 0.00 |
| 200  | 4   | 4.843e-13 0.00 | 15 6.129e-09 0.14 | 15  | 6.286e-08 0.14 | 3   | 5.586e-13 0.02 |
| 300  | 4   | 5.796e-13 0.01 | 11 3.107e-08 0.33 | 11  | 4.963e-08 0.35 | 3   | 1.505e-13 0.07 |
| 400  | 4   | 2.154e-12 0.03 | 15 5.349e-08 1.06 | 15  | 4.930e-08 1.13 | 3   | 6.933e-14 0.15 |
| 500  | 3   | 1.485e-12 0.05 | 21 4.123e-08 2.99 | 22  | 4.932e-08 3.13 | 3   | 4.771e-14 0.31 |
| 600  | 4   | 2.503e-12 0.09 | 28 5.157e-09 6.97 | 34  | 2.770e-07 8.64 | 3   | 7.683e-14 0.65 |
| 700  | 4   | 4.905e-12 0.12 | 40 2.558e-08 17.97 | 49  | 5.079e-07 18.40 | 3   | 1.674e-13 0.38 |
| 800  | 3   | 1.871e-12 0.18 | 9  1.788e-08 5.43 | 10  | 4.993e-07 6.57 | 3   | 7.482e-13 1.22 |
| 900  | 3   | 2.344e-12 0.25 | 10 4.824e-08 8.49 | 12  | 5.009e-07 10.44 | 3   | 4.830e-13 1.80 |
| 1000 | 4   | 2.253e-11 0.31 | 187 1.234e-08 211.21 | 235 | 4.535e-07 263.12 | 3   | 7.547e-13 2.87 |
| 1200 | 3   | 8.500e-12 0.50 | 60 5.735e-09 113.34 | 64  | 5.072e-07 123.40 | 3   | 1.488e-12 4.46 |
| 1400 | 4   | 6.218e-12 0.75 | 22 1.730e-08 65.07 | 27  | 5.041e-07 82.38 | 3   | 4.787e-13 6.92 |
| 1600 | 4   | 1.247e-11 1.13 | 56 2.300e-08 244.24 | 71  | 5.219e-07 313.12 | 3   | 1.394e-12 9.93 |
| 1800 | 4   | 2.627e-11 1.64 | 77 1.894e-08 470.20 | 58  | 5.054e-07 538.70 | 3   | 3.876e-12 14.34 |
| 2000 | 4   | 7.298e-11 2.11 | 29 2.036e-06 250.21 | 28  | 6.790e-06 254.94 | 3   | 3.274e-11 20.24 |
| 2500 | 4   | 1.363e-10 3.76 | 21 3.548e-06 360.32 | 21  | 7.261e-06 363.47 | 3   | 6.407e-13 37.47 |
| 3000 | 4   | 1.984e-10 6.18 | - - - - - - | -   | 1.219e-12 64.59 | 3   | 4.807e-12 94.75 |
| 3500 | 4   | 1.148e-10 9.72 | - - - - - - | -   | 1.482e-12 114.05 | 3   | 4.812e-12 164.84 |
| 4000 | 4   | 4.863e-11 13.39 | - - - - - - | -   | 1.786e-08 165.84 | 3   | 1.002e-12 251.87 |
| 4500 | 4   | 1.867e-10 19.14 | - - - - - - | -   | 3.843e-09 223.21 | 3   | 3.879e-12 308.54 |
| 5000 | 4   | 5.360e-10 25.71 | - - - - - - | -   | 3.039e-09 296.17 | 3   | 3.879e-12 397.24 |
| 5500 | 4   | 5.942e-10 35.34 | - - - - - - | -   | 2.743e-08 440.63 | 3   | 3.879e-12 521.20 |
| 6000 | 4   | 8.873e-11 41.20 | - - - - - - | -   | 3.897e-11 651.06 | 3   | 3.879e-12 751.67 |
| 6500 | 3   | 8.380e-11 52.38 | - - - - - - | -   | - - - - - - | 3   | 3.879e-12 751.67 |
| 7000 | 4   | 1.026e-10 71.85 | - - - - - - | -   | - - - - - - | 3   | 3.879e-12 751.67 |
| 7500 | 4   | 3.892e-10 82.25 | - - - - - - | -   | - - - - - - | 3   | 3.879e-12 751.67 |
| 8000 | 4   | 5.812e-11 161.77 | - - - - - - | -   | - - - - - - | 3   | 3.879e-12 751.67 |

Table 1 shows that PDCA\(_e(\phi_1)\) and PDCA\(_e(\phi_2)\) have almost the same performance for solving AVE (1), but PSM outperforms PDCA\(_e(\phi_1)\) and PDCA\(_e(\phi_2)\), and we see that the accuracy and the iterations obtained by PSM are almost close to NMA but it has a higher speed. However, we compare the performance of the algorithms in terms of accuracy and time separately with the Friedman test. To perform this test, we should take the following steps:

1- Rank the data on the algorithms in each dimension separately such a way that the best performing algorithm gets the rank 1, the second best is ranked 2 and a similar procedure is followed for the rest, and calculate

\[ R_j = \frac{1}{m} \sum_i r_i^j, \]

where \(r_i^j\) is the rank of the \(j\)th algorithm on the \(i\)th dataset in the \(n\)th dimension, and \(l\) and \(m\) are the number of algorithms and the number of data, respectively.

2- Calculate

\[ \chi^2_F = \frac{12m}{l(l+1)} \left[ \sum_j R_j^2 - \frac{l(l+1)^2}{4} \right], \]

where distribution \(\chi^2_F\) has \(l - 1\) degrees of freedom, and calculate

\[ F_F = \frac{(m-1)\chi^2_F}{m(l-1) - \chi^2_F}, \]
which is distributed according to the F-distribution with \( l - 1 \) and \((l - 1)(m - 1)\) degrees of freedom.

3- If \( F_F > F(l - 1, (l - 1)(m - 1)) \), then the algorithms is statistically different.

Now, we will use the Friedman test to compare times of \( PSM \) and \( PDCA_1(\phi_1) \) and \( PDCA_1(\phi_2) \) and \( NMA \) are presented in Table 1. The following table shows the rank for each data in a separate column:

| \( n(d) \) | \( PSM \) | \( PDCA_1(\phi_1) \) | \( PDCA_1(\phi_2) \) | \( NMA \) |
|-------------|-------------|-------------|-------------|-------------|
| Time(s)     | Rank        | Time(s)     | Rank        | Time(s)     | Rank        |
| 100         | 0.00        | 1.5         | 0.01        | 3.5         | 0.01        | 3.5         | 0.60        | 1.5         |
| 200         | 0.00        | 1           | 0.14        | 3.5         | 0.14        | 3.5         | 0.02        | 2           |
| 300         | 0.01        | 1           | 0.33        | 3           | 0.35        | 4           | 0.07        | 2           |
| 400         | 0.03        | 1           | 1.06        | 3           | 1.13        | 4           | 0.15        | 2           |
| 500         | 0.05        | 1           | 2.99        | 3           | 3.13        | 4           | 0.31        | 2           |
| 600         | 0.09        | 1           | 6.97        | 3           | 8.64        | 4           | 0.65        | 2           |
| 700         | 0.12        | 1           | 17.97       | 3           | 18.40       | 4           | 0.88        | 2           |
| 800         | 0.18        | 1           | 5.43        | 3           | 6.57        | 4           | 1.22        | 2           |
| 900         | 0.25        | 1           | 8.49        | 3           | 10.44       | 4           | 1.80        | 2           |
| 1000        | 0.35        | 1           | 211.21      | 3           | 265.12      | 4           | 2.87        | 2           |
| 1200        | 0.50        | 1           | 113.34      | 3           | 123.40      | 4           | 4.46        | 2           |
| 1400        | 0.75        | 1           | 65.07       | 3           | 82.38       | 4           | 6.92        | 2           |
| 1600        | 1.15        | 1           | 244.24      | 3           | 313.12      | 4           | 9.93        | 2           |
| 1800        | 1.64        | 1           | 470.20      | 3           | 598.70      | 4           | 14.34       | 2           |
| 2000        | 2.11        | 1           | 250.21      | 3           | 254.94      | 4           | 20.24       | 2           |
| 2500        | 3.70        | 1           | 360.32      | 3           | 363.47      | 4           | 37.47       | 2           |
| 3000        | 6.18        | 1           | -3.5        | -3.5        | -3.5        | 64.59       | 2           |
| 3500        | 9.72        | 1           | -3.5        | -3.5        | -3.5        | 114.05      | 2           |
| 4000        | 13.99       | 1           | -3.5        | -3.5        | -3.5        | 165.84      | 2           |
| 4500        | 19.14       | 1           | -3.5        | -3.5        | -3.5        | 223.21      | 2           |
| 5000        | 25.71       | 1           | -3.5        | -3.5        | -3.5        | 296.17      | 2           |
| 5500        | 35.54       | 1           | -3.5        | -3.5        | -3.5        | 416.63      | 2           |
| 6000        | 41.20       | 1           | -3.5        | -3.5        | -3.5        | 531.20      | 2           |
| 6500        | 52.38       | 1           | -3.5        | -3.5        | -3.5        | 646.32      | 2           |
| 7000        | 71.85       | 1           | -3.5        | -3.5        | -3.5        | 822.22      | 2           |
| 7500        | 82.25       | 1           | -3.5        | -3.5        | -3.5        | 934.20      | 2           |
| 8000        | 163.77      | 1           | -3.5        | -3.5        | -3.5        | 1145.95     | 2           |

The table below reports the average ranks of the time algorithms with the distribution \( \chi^2_F \) and the test statistic \( F_F \) (in this paper, \( l = 4 \) and \( m = 27 \))

\[
\begin{array}{cccccc}
\chi^2_F & 1.02 & 3.17 & 3.69 & 2.13 & 67.78 & 133.32 \\
\end{array}
\]

\( F_F \) is distributed according to the F distribution with 3 and 76 degrees of freedom. The critical value for \( F(3, 76) \) and \( \alpha = 0.05 \) is 2.73. Since \( F_F > F(3, 76) \), so, the timing of the algorithms is statistically different. Therefore, we can use a post-hoc, Bonferroni-Dunn test, to discover which algorithms practically differ. In this test, there are significant differences in the performance of the two algorithms, if the average ranks difference is less than the critical difference

\[
CD = q_{\alpha} \sqrt{\frac{l(l+1)}{6m}}
\]
where $q_\alpha$ is the critical value (see more details in [21]). Here, $q_\alpha = 2.394$ for 4 algorithms and confidence level $\alpha = 0.05$, so, $CD = .084$. So, based on this test, the PSM speed is better than the other three algorithms, in other words

$$\text{Time(PSM)} > \text{Time(NMA)} > \text{Time(PDCA}_1(\phi_1)) > \text{Time(PDCA}_2(\phi_2)),$$

where “$>$” indicates better performance.

Now, we use the Friedman test and the Bonferroni-Dunn test to compare the accuracy of the algorithms. Therefore, in the following table, we indicate the ranking and accuracy of the algorithms.

### Table 3. Rank and accuracy of four algorithms

| n   | $PSM$       | $PDCA_1(\phi_1)$ | $PDCA_2(\phi_2)$ | $NMA$       |
|-----|-------------|------------------|------------------|-------------|
|     | $f^*$ Rank  | $f^*$ Rank       | $f^*$ Rank       | $f^*$ Rank  |
| 100 | 1.3014e-13  | 2                | 3.1549e-09       | 3.5         | 3.3234e-14 | 1     |
| 200 | 4.8438e-13  | 1.5              | 6.1291e-09       | 3           | 4.2898e-08 | 4     |
| 300 | 5.7916e-13  | 1.5              | 3.1076e-09       | 3           | 4.951e-08  | 4     |
| 400 | 2.1847e-12  | 2                | 5.3480e-09       | 3           | 4.9031e-08 | 4     |
| 500 | 1.4051e-12  | 2                | 4.1335e-09       | 3           | 4.921e-08  | 4     |
| 600 | 2.5018e-12  | 2                | 5.1572e-09       | 3           | 4.1501e-08 | 4     |
| 700 | 4.9005e-12  | 2                | 2.5569e-08       | 3           | 7.6831e-14 | 1     |
| 800 | 1.8771e-12  | 2                | 1.7880e-08       | 3           | 4.8301e-08 | 4     |
| 900 | 2.3440e-12  | 2                | 4.8424e-09       | 3           | 5.0099e-07 | 4     |
| 1000| 2.2558e-11  | 2                | 1.2343e-08       | 3           | 6.7439e-07 | 4     |
| 1200| 8.5001e-12  | 1.5              | 5.7357e-09       | 3           | 5.0522e-07 | 4     |
| 1400| 6.2138e-12  | 2                | 1.7305e-08       | 3           | 4.7878e-13 | 1     |
| 1600| 1.2471e-11  | 2                | 2.3000e-08       | 3           | 4.194e-07  | 4     |
| 1800| 2.6274e-11  | 2                | 1.8945e-08       | 3           | 5.0848e-07 | 4     |
| 2000| 7.3268e-11  | 1.5              | 2.0366e-06       | 3.5         | 6.7804e-06 | 3.5   |
| 2500| 1.5833e-11  | 2                | 3.5403e-06       | 3.5         | 6.4072e-13 | 1     |
| 3000| 1.9845e-11  | 2                | -                | 3.5         | 1.2198e-12 | 1     |
| 3500| 1.1487e-10  | 2                | -                | 3.5         | 1.4824e-12 | 1     |
| 4000| 4.8631e-11  | 1                | -                | 3.5         | 1.7890e-08 | 2     |
| 4500| 1.8675e-10  | 1                | -                | 3.5         | 3.4833e-09 | 2     |
| 5000| 7.5360e-11  | 2                | -                | 3.5         | 3.0396e-12 | 1     |
| 5500| 5.9422e-11  | 1                | -                | 3.5         | 2.7493e-8  | 2     |
| 6000| 8.8736e-11  | 2                | -                | 3.5         | 3.8974e-12 | 1     |
| 6500| 8.3807e-11  | 1                | -                | 3           | -          | 3     |
| 7000| 1.0264e-10  | 1                | -                | 3           | -          | 3     |
| 7500| 3.8926e-10  | 1                | -                | 3           | -          | 3     |
| 8000| 5.8126e-11  | 1                | -                | 3           | -          | 3     |

The following table indicates the average ranking of the accuracy algorithms and $\chi^2_F$ and $F_F$.

| $RP_{PSM}$ | $RP_{PDCA_1(\phi_1)}$ | $RP_{PDCA_2(\phi_2)}$ | $RNMA$ | $\chi^2_F$ | $F_F$ |
|-------------|------------------------|------------------------|--------|------------|------|
| 1.67        | 3.19                   | 3.67                   | 1.48   | 57.72      | 64.47|

Therefore, $F_F > F(3, 76)$, and the accuracy of the algorithms is statistically different. According to the Bonferroni-Dunn test to confidence level $\alpha = 0.05$, NMA has better accuracy than other algorithms for solving AVE, but the average rank difference between PSM and NMA is negligible (.19), so, PSM has good accuracy and

$$\text{Accuracy}(NMA) > \text{Accuracy}(PSM) > \text{Accuracy}(PDCA_1(\phi_1)) > \text{Accuracy}(PDCA_2(\phi_2)).$$
5. Conclusion. In this paper, we have presented two new approaches for solving the absolute value equations, the proximal difference-of-convex algorithm with extrapolation and the proximal subgradient method, and we proved their convergence to a solution for AVE (1). The numerical results and the Friedman test indicate that these are effective methods for solving absolute value equations, and the proximal subgradient method performs better than the proximal difference-of-convex algorithm with extrapolation in terms of speed and accuracy, and the accuracy of this algorithm is almost close to the Newton method with the Armijo step, but it has a higher speed to solve absolute value equations.

REFERENCES

[1] J. Y. Bello Cruz, O. P. Ferreira and L. F. Prudente, On the global convergence of the inexact semi-smooth Newton method for absolute value equation, Computational Optimization and Applications, 65 (2016), 93–108.
[2] X. Chen, Smoothing methods for nonsmooth, nonconvex minimization, Mathematical Programming, 134 (2012), 71–99.
[3] H. G. Dales, Banach Algebras and Automatic Continuity, London Mathematical Society Monographs, Clarendon Press, Oxford, 2000.
[4] V. Edalatpour, D. Hezari and D. K. Salkuyeh, A generalization of the Gauss Seidel iteration method for solving absolute value equations, Applied Mathematics and Computation, 293 (2017), 156–167.
[5] J. Gotoh, A. Takeda and K. Tono, DC formulations and algorithms for sparse optimization problems, Mathematical Programming, 169 (2018), 141–176.
[6] F. Hashemi and S. Ketabchi, Numerical comparisons of smoothing functions for optimal correction of an infeasible system of absolute value equations, Numerical Algebra, Control and Optimization, 10 (2020), 13–21.
[7] S. Ketabchi and H. Moosaei, An efficient method for optimal correcting of absolute value equations by minimal changes in the right hand side, Computers and Mathematics with Applications, 64 (2012), 1882–1885.
[8] S. Ketabchi and H. Moosaei, Minimum norm solution to the absolute value equation in the convex case, Journal of Optimization Theory and Applications, 154 (2012), 1080–1087.
[9] C.-X. Li, A modified generalized newton method for absolute value equations, Journal of Optimization Theory and Applications, 170 (2016), 1055–1059.
[10] P. L. Lions and B. Mercier, Splitting algorithms for the sum of two nonlinear operators, SIAM Journal on Numerical Analysis, 16 (1979), 964–979.
[11] O. L. Mangasarian, A generalized Newton method for absolute value equations, Optimization Letters, 3 (2009), 101–108.
[12] O. L. Mangasarian, Absolute value equation solution via concave minimization, Optimization Letters, 1 (2007), 3–8.
[13] O. L. Mangasarian, Primal-dual bilinear programming solution of the absolute value equation, Optimization Letters, 6 (2012), 1527–1533.
[14] R. D. Millan and M. P. Machado, Inexact proximal ε-subgradient methods for composite convex optimization problems, Journal of Global Optimization, 75 (2019), 1029–1060.
[15] H. Moosaei, S. Ketabchi, M. A. Noor, J. Iqbal and V. Hooshyarbakhshi, Some techniques for solving absolute value equations, Applied Mathematics and Computation, 268 (2015), 696–705.
[16] G. Ning and Y. Zhou, An improved differential evolution algorithm for solving absolute value equations, High Performance Computing and Applications, (2016), 38–47.
[17] G. B. Passty, Ergodic convergence to a zero of the sum of monotone operators in Hilbert space, Journal of Mathematical Analysis and Applications, 72 (1979), 383–390.
[18] D. T. Pham and H. A. Le Thi, Convex analysis approach to D.C. programming: theory, algorithms and applications, Acta Mathematica Vietnamica, 22 (1997), 289–355.
[19] T. J. Ransford, A short proof of Johnson’s uniqueness-of-norm theorem, Bulletin of the London Mathematical Society, 21 (1989), 487–488.
[20] B. Saheya, C. H. Yu and J. S. Chen, Numerical comparisons based on four smoothing functions for absolute value equation, Journal of Applied Mathematics and Computing, 56 (2018), 131–149.
[21] J. S. Sartakhti, H. Afrabandpey and M. Saraee, Simulated annealing least squares twin support vector machine (SA-LSTSV) for pattern classification, *Soft Computing*, **21** (2017), 4361–4373.

[22] D. K. Salkuyeh, The Picard-HSS iteration method for absolute value equations, *Optimization Letters*, **8** (2014), 2191–2202.

[23] B. Wen, X. Chen and T. K. Pong, A proximal difference-of-convex algorithm with extrapolation, *Computational Optimization and Applications*, **69** (2018), 297–324.

[24] C. Zhang and Q. J. Wei, Global and finite convergence of a generalized newton method for absolute value equations, *Journal of Optimization Theory and Applications*, **143** (2009), 391–403.

Received May 2020; 1st revision June 2020; Final revision July 2020.

*E-mail address:* samira.tmoalem@gmail.com, s_shahsavari@phd.guilan.ac.ir

*E-mail address:* sketabchi@guilan.ac.ir