Available online at www.isr-publications.com/jnsa
J. Nonlinear Sci. Appl., 10 (2017), 3643–3654
Research Article

ISSN : 2008-1898

Journal of Nonlinear Sciences and Applications

Journal Homepage: www.tjnsa.com - www.isr-publications.com/jnsa

Picard splitting method and Picard CG method for solving the absolute value equation

Chang-Qing Lv[a,b], Chang-Feng Ma[a,*]

[a]College of Mathematics and Informatics, Fujian Key Laborotary of Mathematical Analysis and Applications, Fujian Normal University, Fuzhou, 350117, P. R. China.

[b]School of Mathematics and Statistics, Zaozhuang University, Zaozhuang, 277160, P. R. China.

Communicated by Y.-Z. Chen

Abstract

In this paper, we combine matrix splitting iteration algorithms, such as, Jacobi, SSOR or SAOR algorithms with Picard method for solving absolute value equation. Then, we propose Picard CG for solving the absolute value equation. We discuss the convergence of those methods we proposed. At last, some examples are provided to illustrate the efficiency and validity of methods that we present.

Keywords: Absolute value equation, Picard algorithm, matrix splitting iteration method, conjugate gradient method.
2010 MSC: 65H10, 47H10.

1. Introduction

We consider the absolute value equation (AVE)

$$Ax - |x| = b, \tag{1.1}$$

where $A \in R^{n\times n}$, $b \in R^n$, and $|x|$ denotes each component of x with absolute value. In [2, 3, 16], many mathematical programming problems can be reduced to a linear complementary (LCP) and LCP is equivalent to an absolute value equation (1.1). And many methods are proposed for solving AVE (see [5–15, 17, 19–24, 26]). Mangasaruan and Meyer [16] show that the equation (1.1) has a unique solution for any right-hand side b if all singular values of the coefficient matrix A exceed one. Mangasaruan [12] considered a generalized Newton method for AVE (1.1) and investigated its convergence properties. Since the linear system of equations needs to be solved in each generalized Newton iteration step and its coefficient matrix are changed correspondingly, the cost of the generalized Newton method is expensive. Rohn et al. [24] propose another method to solve AVE (1.1), which is called Picard iteration method and its iteration sequence is as follows

$$x^{k+1} = A^{-1}(|x^k| + b).$$

*Corresponding author
Email address: macf@fjnu.edu.cn (Chang-Feng Ma)

doi:10.22436/jnsa.010.07.24

Received 2017-06-15

Bai and Yang [1] considered the weakly nonlinear system $Ax = \phi(x)$ and proposed Picard-HSS method for solving it. By comparing the weakly nonlinear system with the AVE (1.1), Salkuyeh [25] gave the Picard-HSS method for solving the absolute value equation (1.1). Inspired by the idea of Picard iteration and HSS splitting method, we propose Picard splitting iteration method and Picard CG method for solving absolute value equation based on the Picard iteration, and prove the convergence of those methods. At last, we give several numerical experiments.

The remainder of this paper is organized as follows. In Section 2, we give brief introduction of basic splitting iteration and basic conjugate gradient method for solving linear equation system. In Section 3, we propose a Picard splitting iteration and investigate its convergence. In Section 4, we show the Picard conjugate gradient method and its convergence. In Section 5, we present some numerical experiments. Finally, we give our conclusions in Section 6.

2. Matrix splitting method and conjugate gradient method

2.1. The splitting iteration method

A splitting of A is a decomposition $A = M - N$ with M nonsingular.

A splitting yields an iterative method as follows: $Ax = Mx - Nx = b$ implies $Mx = Nx + b$, that is, $x = M^{-1}Nx + M^{-1}b = Rx + c$, where $R = M^{-1}N, c = M^{-1}b$. So we can take

$$x^{k+1} = Rx^k + c. \tag{2.1}$$

Iteration sequence (2.1) in the matrix-vector form can be equivalently rewritten as

$$x^{k+1} = \sum_{i=0}^{k} R^i b.$$

Given x^0, these methods generate a sequence x^k converging to the solution $A^{-1}b$ of $Ax = b$ and x^{k+1} is cheap to compute from x^k. If x^k defined in Eq. (2.1) is convergent, the splitting $A = M - N$ is called convergent splitting.

Lemma 2.1 ([18]). *Suppose $\|\cdot\|$ is any operator norm. $\rho(R) < 1$ if and only if $x^{k+1} = Rx^k + c$ converges for any initial vertex x^0.*

Suppose $A \in C^{n\times n}$. Let $A = M_i - N_i$ (i = 1, 2) be two splittings of the matrix A and $x^0 \in C^n$ be a given initial vertex. x^k is a two-step iteration sequence defined as

$$\begin{cases} M_1 x^{k+\frac{1}{2}} = N_1 x^k + b, \\ M_2 x^{k+1} = N_2 X^{k+\frac{1}{2}} + b, \end{cases} \tag{2.2}$$

where $k = 1, 2, \cdots$.

Lemma 2.2 ([18]). *Let $A = M_i - N_i$ $(i = 1, 2)$ be two splittings of the matrix A and $x^0 \in C^n$ be a given initial vertex. If x^k is a two-step iteration sequence defined in Eq. (2.2), then*

$$x^{k+1} = \tilde{R}_1 x^k + \tilde{T}(k = 1, 2, \cdots),$$

where

$$\tilde{R}_1 = M_2^{-1}N_2M_1^{-1}N_1, \quad \tilde{T} = M_2^{-1}(M_1 + N_2)M_1^{-1}.$$

By Lemma 2.2, two-step iteration sequence (2.2) can be equivalently rewritten as

$$x^{k+1} = \tilde{R}_1^k x^0 + \sum_{i=0}^{k} \tilde{R}_1^i \tilde{T} b. \tag{2.3}$$

By Lemma 2.1, if $\rho(\tilde{R}_1) < 1$, then the two-step iteration sequence (2.3) converges to the exact solution of $Ax = b$ for any initial guess $x^0 \in c^{n\times n}$.

2.2. The conjugate gradient method

Algorithm 2.3 (Conjugate Gradient Method (CG)). (for Hermitian positive definite problems)
Given an initial guess x_0, compute $r_0 = b - Ax_0$ and set $p_0 = r_0$.
$k := 1$;
while $\|r_k\| > \varepsilon$
$\alpha_{k-1} = \frac{\langle r_{k-1}, r_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}$,
$x_k = x_{k-1} + \alpha_{k-1}p_{k-1}$,
$r_k = r_{k-1} - \alpha_{k-1}Ap_{k-1}$,
$\beta_{k-1} = \frac{\langle r_k, r_k\rangle}{\langle r_{k-1}, r_{k-1}\rangle}$,
$p_k = r_k + \beta_{k-1}p_{k-1}$,
k:=k+1;
end.

3. The Picard splitting iteration method

In this section, we will combine Picard method with matrix splitting method for solving the absolute value equation (1.1). The Picard iteration sequence for solving the AVE (1.1) is as follows

$$Ax^{k+1} = |x^k| + b.$$

Let $A = M(\alpha) - N(\alpha)$ and x^{k+1} can be approximately computed by splitting iteration such as Jacobi method, SSOR method, SAOR, and HSS method. Let

$$M(\alpha)x^{(k,l+1)} = N(\alpha)x^{(k,l)} + |x^k| + b \tag{3.1}$$

and

$$x^k = x^{(k-1,l_{k-1})},$$

where $R_1(\alpha) = M(\alpha)^{-1}N(\alpha)$ and $l = 0, 1, \cdots, l_k$, $k = 0, 1, \cdots$. Eq. (3.1) can be equivalently rewritten as

$$x^{k+1} = R_1^{l_k}(\alpha)x^k + \sum_{i=0}^{l_k-1} R_1^i(\alpha)M(\alpha)^{-1}(|x^k| + b). \tag{3.2}$$

From the above analysis, we can obtain Picard single-step splitting iteration algorithm as follows.

Algorithm 3.1 (Picard single-step splitting iteration algorithm). Choose any $x^0 \in R^n$ and $\varepsilon > 0$, $r^0 = b + |x^0| - Ax^0$, $k = 0$;
while $\|r^k\|/\|r^0\| > \varepsilon$
$l := 0$,
$x^{(k,l)} = x^k$,
$r^{(k,l)} = b + |x^k| - Ax^{(k,l)}$;
while $\|r^{(k,l)}\|/\|r^0\| > \epsilon$
$x^{(k,l)} = R_1(\alpha)x^{(k,l-1)} + M^{-1}(|x^k| + b)$,
$r^{(k,l)} = b + |x^{(k,l)}| - Ax^{(k,l)}$,
$l := l + 1$;
end.
$x^{k+1} = x^{(k,l_k)}$;
$r^{k+1} = b + |x^{k+1}| - Ax^{k+1}$
$k := k + 1$;
end.

For the uniqueness of solution of AVE (1.1), an interesting existence result is as follows.

Lemma 3.2 ([16]). *The AVE* (1.1) *is uniquely solvable for any* b *if* $\|A^{-1}\|_2 < 1$.

Theorem 3.3. *Suppose* $A = M(\alpha) - N(\alpha)$ *is convergent splitting. If* $\eta = \|A^{-1}\|_2 < 1$ *and* $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, *then the AVE* (1.1) *has a unique solution* x^* *and the iteration sequence* $\{x^k\}$ *produced by Algorithm* 3.1 *converges to* x^* *for any initial guess* $x^0 \in c^n$ *and any sequence of positive integers* l_k, $k = 1, 2, \cdots$.

Proof. According to Lemma 3.2 and $\eta < 1$, AVE (1.1) has a unique solution x^*. On the other hand, because Eq. (3.1) can be rewritten as Eq. (3.2), the solution x^* of AVE (1.1) satisfies the following equation

$$x^* = R_1^{l_k}(\alpha)x^* + \sum_{i=0}^{l_k-1} R_1^i(\alpha)M(\alpha)^{-1}(|x^*| + b). \tag{3.3}$$

From Eqs. (3.2) and (3.3), we have

$$x^{k+1} - x^* = R_1^{l_k}(\alpha)(x^k - x^*) + \sum_{i=0}^{l_k-1} R_1^i(\alpha)M(\alpha)^{-1}(|x^k| - |x^*|). \tag{3.4}$$

Since $\rho(R_1(\alpha)) < 1$, we have

$$\begin{aligned} \sum_{j=0}^{l_k-1} R_1(\alpha)M^{-1}(\alpha) &= (I - R_1(\alpha)^{l_k})(I - R_1(\alpha))^{-1}M(\alpha)^{-1} \\ &= (I - R_1(\alpha)^{l_k})(M(\alpha) - M(\alpha)R_1(\alpha))^{-1} \\ &= (I - R_1(\alpha)^{l_k})(M(\alpha) - N(\alpha))^{-1} = (I - R_1(\alpha)^{l_k})A^{-1}. \end{aligned} \tag{3.5}$$

From Eq. (3.4) and Eq. (3.5), we have

$$\begin{aligned} \|x^{k+1} - x^*\|_2 &= \|R_1^{l_k}(\alpha)(x^k - x^*) + \sum_{i=0}^{l_k-1} R_1^i(\alpha)M(\alpha)^{-1}(|x^k| - |x^*|)\|_2 \\ &= \|R_1^{l_k}(\alpha)(x^k - x^*) - (I - R_1(\alpha)^{l_k})A^{-1}(|x^k| - |x^*|)\|_2 \\ &= \|R_1(\alpha)^{l_k}\left((x^k - x^*) + A^{-1}(|x^k| - |x^*|)\right) - A^{-1}(|x^k| - |x^*|)\|_2 \\ &\leqslant \left(\|R_1(\alpha)\|_2^{l_k}(1+\eta) + \eta\right)\|x^k - x^*\|_2 \\ &\leqslant \left(\|R_1(\alpha)\|_2^{l_k}(1+\eta) + \eta\right)\left(\|R_1(\alpha)\|_2^{l_{k-1}}(1-\eta) + \eta\right)\cdots\left(\|R_1(\alpha)\|_2^{l_1}(1+\eta) + \eta\right)\|x^1 - x^*\|_2 \\ &\leqslant \left(\|R_1(\alpha)\|_2(1+\eta) + \eta\right)\left(\|R_1(\alpha)\|_2(1-\eta) + \eta\right)\cdots\left(\|R_1(\alpha)\|_2(1+\eta) + \eta\right)\|x^1 - x^*\|_2 \\ &= \left(\|R_1(\alpha)\|_2(1+\eta) + \eta\right)^k\|x^1 - x^*\|_2. \end{aligned}$$

Since $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, then the desired result can be obtained. □

Remark 3.4. Suppose $A \in C^{n\times n}$. Let D be the diagonal , $-L$ be the strictly lower triangular parts, and $-U$ be the strictly upper triangular parts of A, respectively. If $M = D$, $N = L + U$, Algorithm 3.1 is called Picard-Jacobi method. If $M = D - L$, $N = U$, Algorithm 3.1 is called Picard-GS method.

Now we consider two-step iteration method for solving AVE (1.1). Suppose $A \in C^{n\times n}$. Let $A = M_i(\alpha) - N_i(\alpha)$ $(i = 1, 2)$ be splitting of the matrix A and $x^0 \in C^n$ be a given initial vertex. If $\{x^k\}$ is a two-step iteration sequence defined as

$$\begin{cases} M_1(\alpha)x^{(k,l+\frac{1}{2})} = N_1(\alpha)x^{(k,l)} + |x^k| + b, \\ M_2(\alpha)x^{(k,l+1)} = N_2(\alpha)X^{(k,l+\frac{1}{2})} + |x^k| + b, \end{cases} \quad (l = 0, 1, \cdots, l_k, k = 0, 1, \cdots), \tag{3.6}$$

Eq. (3.6) can be equivalently rewritten as

$$x^{k+1} = \tilde{R}_1^{l_k}(\alpha)x^k + \sum_{i=0}^{l_k-1} \tilde{R}_1^i(\alpha)\tilde{T}(\alpha)(|x^k| + b)(k = 1, 2, \cdots), \tag{3.7}$$

where $x^{k+1} = x^{(k,l_k)}$ and

$$\tilde{R}_1(\alpha) = M_2^{-1}(\alpha)N_2(\alpha)M_1^{-1}(\alpha)N_1(\alpha), \quad \tilde{T}(\alpha) = M_2^{-1}(\alpha)(M_1(\alpha) + N_2(\alpha))M_1^{-1}(\alpha). \tag{3.8}$$

Algorithm 3.5 (Picard two step splitting iteration algorithm). Choose any $x^0 \in R^n$ and $\varepsilon > 0$, $r^0 = b + |x_0| - Ax_0$, $k = 0$;
while $\|r^k\|/\|r^0\| > \varepsilon$,
$l := 0$,
$x^{(k,l)} = x^k$,
$r^{(k,l)} = b + |x^k| - Ax^{(k,l)}$,
$p^{(k,l+1)} = r^{(k,l)}$,
$\rho^{(k,l)} = r^{(k,l)}Ar^{(k,l)}$,
while $\|r^{(k,l)}\|/\|r^0\| > \epsilon$
$M_1(\alpha)x^{(k,l+\frac{1}{2})} = N_1(\alpha)x^{(k,l)} + |x^k| + b$,
$M_2(\alpha)x^{(k,l+1)} = N_2(\alpha)x^{(k,l+\frac{1}{2})} + |x^k| + b$,
$r^{(k,l+1)} = b + |x^{(k,l+1)}| - Ax^{(k,l+1)}$,
$l := l + 1$;
end.
$x^{k+1} = x^{(k,l_k)}$,
$r^{k+1} = b + |x^{k+1}| - Ax^{k+1}$,
$k := k + 1$;
end.

Theorem 3.6. *Suppose* $A = M_i - N_i, (i = 1, 2)$ *is convergent splitting and* $K = (I - \tilde{R}_1(\alpha))^{-1}\tilde{T}(\alpha)$. *Let* $\eta = \|K\|_2 < 1$ *and* $\|\tilde{R}_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$. *If the AVE* (1.1) *has a solution* x^*, *then the iteration sequence* $\{x^k\}$ *produced by Algorithm* 3.5 *converges to* x^* *for any initial guess* $x^0 \in c^n$ *and any sequence of positive integer* $l_k, k = 1, 2, \cdots$.

Proof. Since AVE (1.1) has a solution x^* and Eq. (3.7) can be rewritten as Eq. (3.8), the solution x^* satisfies the following equation

$$x^* = \tilde{R}^{l_k}(\alpha)x^* + \sum_{i=0}^{l_k-1} \tilde{R}^i(\alpha)\tilde{T}(\alpha)(|x^*| + b) \quad (k = 1, 2, \cdots). \tag{3.9}$$

By Eqs. (3.7) and (3.9)

$$x^{k+1} - x^* = \tilde{R}_1^{l_k}(\alpha)(x^k - x^*) + \sum_{i=0}^{l_k-1} R_1^i(\alpha)\tilde{T}(\alpha)(|x^k| - |x^*|).$$

Since $\rho(\tilde{R}_1(\alpha)) < 1$, we have

$$\sum_{j=0}^{l_k-1} \tilde{R}_1(\alpha)M^{-1}(\alpha) = (I - \tilde{R}_1(\alpha)^{l_k})(I - \tilde{R}_1(\alpha))^{-1}\tilde{T}(\alpha).$$

According to $\eta = \|K\|_2 = \|(I - \tilde{R}_1(\alpha))^{-1}\tilde{T}(\alpha)\|_2$, we have

$$\|x^{k+1} - x^*\|_2 = \|\tilde{R}_1^{l_k}(\alpha)(x^k - x^*) - (I - \tilde{R}_1(\alpha)^{l_k})K(|x^k| - |x^*|)\|_2$$

$$
\begin{aligned}
&= \|\tilde{R}_1(\alpha)^{l_k}\left[(x^k - x^*) + K(|x^k| - |x^*|)\right] - K(|x^k| - |x^*|)\|_2 \\
&\leqslant \left(\|\tilde{R}_1(\alpha)\|_2^{l_k}(1+\eta) + \eta\right)\|x^k - x^*\|_2 \\
&\leqslant \left(\|\tilde{R}_1(\alpha)\|_2^{l_k}(1+\eta) + \eta\right)\left(\|\tilde{R}_1(\alpha)\|_2^{l_{k-1}}(1+\eta) + \eta\right)\cdots\left(\|\tilde{R}_1(\alpha)\|_2^{l_1}(1+\eta) + \eta\right)\|x^1 - x^*\|_2 \\
&\leqslant \left(\|\tilde{R}_1(\alpha)\|_2(1+\eta) + \eta\right)\left(\|\tilde{R}_1(\alpha)\|_2(1+\eta) + \eta\right)\cdots\left(\|\tilde{R}_1(\alpha)\|_2(1+\eta) + \eta\right)\|x^1 - x^*\|_2 \\
&= \left(\|\tilde{R}_1(\alpha)\|_2(1+\eta) + \eta\right)^k\|x^1 - x^*\|_2.
\end{aligned}
$$

Since $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, then $\|\tilde{R}_1(\alpha)\|_2(1+\eta)+\eta) < 1$, when $k \to \infty$, the right hand side of above equation tends to 0, thus the iteration sequence x^k converges to x^*. The desired result can be obtained. □

Remark 3.7. If $M_1 = \frac{1}{\alpha}D - L$, $N_1 = \frac{1-\alpha}{\alpha}D + U$, $M_2 = \frac{1}{\alpha}D - U$, $N_2 = \frac{1-\alpha}{\alpha}D + L$, Algorithm 3.5 is called Picard-SSOR algorithm. If $M_1 = \frac{D-rL}{\alpha}$, $N_1 = \frac{1-\alpha}{\alpha}D + \frac{\alpha-r}{\alpha}L + U$, $M_2 = \frac{D-rU}{\alpha}$, $N_2 = \frac{1-\alpha}{\alpha}D + \frac{\alpha-r}{\alpha}U + L$, Algorithm 3.5 is called Picard-SAOR algorithm. If $H = \frac{A+A^H}{2}$, $H = \frac{A-A^H}{2}$ and $M_1 = \alpha I + H$, $N_1 = \alpha I - S$, $M_2 = \alpha I + s$, $N_2 = \alpha I - H$, Algorithm 3.5 is called Picard-HSS algorithm.

Theorem 3.8. *Suppose iteration sequence $\{x^k\}$ generated by the Picard-SSOR algorithm. If $\eta = \|A^{-1}\|_2 < 1$ and $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, then the AVE (1.1) has a unique solution x^* and the iteration sequence $\{x^k\}$ converges to x^* for any initial guess $x^0 \in c^n$ and any sequence of positive integers l_k, $k = 1, 2, \cdots$.*

Proof. Since $\eta < 1$, according to Lemma 3.2, AVE (1.1) has a unique solution x^*. By Theorem 3.6, we just prove $A^{-1} = K$, where $K = (I - \tilde{R}_1(\alpha))^{-1}\tilde{T}(\alpha)$. According to Remark 3.7, we know that $M_1 = \frac{1}{\alpha}D - L$, $N_1 = \frac{1-\alpha}{\alpha}D + U$, $M_2 = \frac{1}{\alpha}D - U$, $N_2 = \frac{1-\alpha}{\alpha}D + L$. From Eq. (3.8) and Lemma 2.2, we can obtain

$$
\begin{aligned}
\tilde{R}_1 &= M_2^{-1}N_2M_1^{-1}N_1 = (D - \alpha U)^{-1}\left((1-\alpha)D + \alpha L\right)(D - \alpha L)^{-1}\left((1-\alpha)D + \alpha U\right), \\
\tilde{T} &= M_2^{-1}(M_1 + N_2)M_1^{-1} = \alpha(2-\alpha)(D - \alpha U)^{-1}D(D - \alpha L)^{-1}.
\end{aligned}
$$

Let

$$
M = \frac{1}{\alpha(2-\alpha)}(D - \alpha L)D^{-1}(D - \alpha U)
$$

and

$$
N = \frac{1}{\alpha(2-\alpha)}\left((1-\alpha)D + \alpha L\right)D^{-1}\left((1-\alpha)D + \alpha U\right).
$$

Then $A = M - N$,

$$
\begin{aligned}
M^{-1}N &= (D - \alpha U)^{-1}D(D - \alpha L)^{-1}\left((1-\alpha)D + \alpha L\right)D^{-1}\left((1-\alpha)D + \alpha U\right) \\
&= (D - \alpha U)^{-1}(I - \alpha LD^{-1})^{-1}\left((1-\alpha)I + \alpha LD^{-1}\right)\left((1-\alpha)D + \alpha U\right) \\
&= (D - \alpha U)^{-1}\left((1-\alpha)I + \alpha LD^{-1}\right)(I - \alpha LD^{-1})^{-1}\left((1-\alpha)D + \alpha U\right) \\
&= (D - \alpha U)^{-1}\left((1-\alpha)D + \alpha L\right)D^{-1}D(D - \alpha L)^{-1}\left((1-\alpha)D + \alpha U\right) = \tilde{R}_1.
\end{aligned}
\tag{3.10}
$$

Since $M^{-1} = \tilde{T}(\alpha)$ and Eq. (3.10), we have

$$
K = (I - \tilde{R}_1)^{-1}\tilde{T} = (I - M^{-1}N)^{-1}M^{-1} = A^{-1}.
$$

Since $\eta = \|K\|_2 = \|A^{-1}\|_2 < 1$, and by Theorem 3.6, the desired result can be obtained. □

Similar to Theorem 3.8, we have the convergence of Picard-SAOR Algorithm.

Theorem 3.9. *Suppose iteration sequence $\{x^k\}$ generated by the Picard-SAOR algorithm . If $\eta = \|A^{-1}\|_2 < 1$ and $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, then the AVE (1.1) has a unique solution x^* and the iteration sequence $\{x^k\}$ converges to x^* for any initial guess $x^0 \in c^n$ and any sequence of positive integers l_k, $k = 1, 2, \cdots$.*

Proof. Since $\eta < 1$, according to Lemma 3.2, AVE (1.1) has a unique solution x^*. By Theorem 3.6, we just prove $A^{-1} = K$, where $K = (I - \tilde{R}_1(\alpha))^{-1}\tilde{T}(\alpha)$. According to Remark 3.7, $M_1 = \frac{D-rL}{\alpha}$, $N_1 = \frac{1-\alpha}{\alpha}D + \frac{\alpha-r}{\alpha}L + U$, $M_2 = \frac{D-rU}{\alpha}$, $N_2 = \frac{1-\alpha}{\alpha}D + \frac{\alpha-r}{\alpha}U + L$.

From Eq. (3.8) and Lemma 2.2, we can obtain

$$\begin{aligned}
\tilde{R}_1 &= M_2^{-1}N_2M_1^{-1}N_1 \\
&= (D - rU)^{-1}\left((1-\alpha)D + (\alpha - r)U + \alpha L\right)(D - rL)^{-1}\left((1-\alpha)D + (\alpha - r)L + \alpha U\right) \\
&= (D - rU)^{-1}(D - rU - \alpha A)(D - rL)^{-1}(D - rL - \alpha A), \\
\tilde{T} &= M_2^{-1}(M_1 + N_2)M_1^{-1} \\
&= \alpha(D - rU)^{-1}\left((2-\alpha)D + (\alpha - r)L + (\alpha - r)U\right)(D - rL)^{-1} \\
&= \alpha(D - rU)^{-1}((2-r)D + (r-\alpha)A)(D - rL)^{-1}.
\end{aligned}$$

Let

$$M = \frac{1}{\alpha}(D - \alpha L)((2-r)D + (r-\alpha)A)^{-1}(D - \alpha U)$$

and

$$N = \frac{1}{\alpha}(D - rU - \alpha A)\left((2-r)D + (r-\alpha)A\right)^{-1}(D - rL - \alpha A).$$

Then $A = M - N$,

$$\begin{aligned}
M^{-1}N &= (D - \alpha U)^{-1}((2-r)D + (r-\alpha)A)(D - \alpha L)^{-1} \\
&\quad \times (D - rU - \alpha A)\left((2-r)D + (r-\alpha)A\right)^{-1}(D - rL - \alpha A) \\
&= (D - \alpha U)^{-1}(D - rU - \alpha A)(D - \alpha L)^{-1} \\
&\quad \times ((2-r)D + (r-\alpha)A)((2-r)D + (r-\alpha)A)^{-1}(D - rL - \alpha A) \\
&= (D - rU)^{-1}(D - rU - \alpha A)(D - rL)^{-1}(D - rL - \alpha A) = \tilde{R}_1.
\end{aligned} \tag{3.11}$$

Since $M^{-1} = \tilde{T}(\alpha)$ and Eq. (3.11),

$$K = (I - \tilde{R}_1)^{-1}\tilde{T} = (I - M^{-1}N)^{-1}M^{-1} = A^{-1}.$$

Since $\eta = \|K\|_2 = \|A^{-1}\|_2 < 1$, and by Theorem 3.6, the desired result can be obtained. □

The next theorem provides sufficient conditions for the convergence of the Picard-HSS method to solve AVE (1.1).

Theorem 3.10 ([25]). *Let $A \in C^{n\times n}$ be a positive definite matrix, suppose iteration sequence $\{x^k\}$ generated by the Picard-HSS algorithm. If $\eta = \|A^{-1}\|_2 < 1$ and $\|R_1(\alpha)\|_2 < \frac{1-\eta}{1+\eta}$, then the AVE (1.1) has a unique solution x^* and the iteration sequence $\{x^k\}$ converges to x^* for any initial guess $x^0 \in c^n$ and any sequence of positive integers $l_k, k = 1, 2, \cdots$.*

4. The Picard CG method

In this section, we combine the picard method with CG iteration and propose the Pciard-CG algorithm for solving the AVE (1.1). Suppose A is a symmetric positive definite matrix such that $\|A^{-1}\|_2 < 1$. Let $A \in R^{n\times n}$ be a nonsingular matrix and $x \in R^n$; A-norm of vector x is $\|A^{\frac{1}{2}}x\|_2$, denoted by $\|x\|_A$.

Algorithm 4.1 (Picard conjugate gradient algorithm). Choose any $x_0 \in R^n$ and $\varepsilon > 0$,

Let $r_0 = b + |x_0| - Ax_0$,

$k := 0$;

while $\|r_k\|/\|r_0\| > \varepsilon$

$l := 0$,
$x_{(k,l)} = x_k$, $r_{(k,l)} = b + |x_k| - Ax_{(k,l)}$,
$p_{(k,l+1)} = r_{(k,l)}$, $\rho_{(k,l)} = r_{(k,l)}^T r_{(k,l)}$;
while $\|r_{(k,l)}\|/\|r_0\| > \epsilon$
$z_{(k,l)} = Ap_{(k,l)}$,
$\alpha_{(k,l)} = \rho_{(k,l-1)}/z_{(k,l)}^T p_{(k,l)}$,
$x_{(k,l)} = x_{(k,l-1)} + \alpha p_{(k,l)}$,
$r_{(k,l)} = r_{(k,l-1)} - \alpha_{(k,l-1)} z_{(k,l-1)}$,
$\rho_{(k,l)} = r_{(k,l)}^T r_{(k,l)}$,
$\beta_{(k,l)} = \rho_{(k,l)}/\rho_{(k,l-1)}$,
$p_{(k,l+1)} = r_{(k,l)} + \beta p_{(k,l)}$,
$l := l + 1$;
end.
$x_{k+1} = x_{(k,l_k)}$,
$r_{k+1} = b + |x_{k+1}| - Ax_{k+1}$,
$k := k + 1$;
end.

Lemma 4.2 ([4]). *If the iteration solution $\{x_k\}$ is generated by Algorithm 4.1, then x_k satisfies the following inequality*

$$\frac{\|x_k - x^*\|_A}{\|x_0 - x^*\|_A} \leqslant \frac{1}{C_k(1 + \frac{2}{\kappa - 1})}$$

and

$$\frac{1}{C_k(1 + \frac{2}{\kappa-1})} = \frac{1}{C_k(\frac{b+a}{b-a})} = \frac{2\sigma^k}{1 + \sigma^{2k}}, \tag{4.1}$$

where C_k is k order Chebyshev polynomial, $\sigma = (\sqrt{b} - \sqrt{a})/(\sqrt{b} + \sqrt{a})$ and $\kappa = \|A\|_2\|A^{-1}\|_2 = \frac{\sqrt{b}}{\sqrt{a}}$.

Theorem 4.3. *If A is a symmetric positive definite matrix such that $\eta = \|A^{-1}\|_2 < 1$, then the AVE (1.1) has a unique solution x^* and the iteration sequence $\{x_k\}$ converges to x^*.*

Proof. Since $\eta < 1$, according to Lemma 3.2, AVE (1.1) has a unique solution x^*.

We fix outer iteration step k, suppose x_k^* is the exactly solution of the equation $Ax = b + |x_k|$, we can obtain

$$Ax_k^* = b + |x_k|.$$

Since x^* is exactly solution of the AVE (1.1) satisfying the equation $Ax^* = b + |x^*|$, we can obtain that

$$x_k^* - x^* = A^{-1}(|x_k| - |x^*|). \tag{4.2}$$

By Lemma 4.2, Eq. (4.2), and

$$[C_{l_k}(1 + \frac{2}{\kappa - 1})]^{-1} = \frac{2\sigma^{l_k}}{1 + \sigma^{2l_k}}, \quad \eta = \|A^{-1}\|_2,$$

we have

$$\begin{aligned}
\|x_{k+1} - x^*\|_A &= \|x_{k+1} - x_k^* + x_k^* - x^*\|_A \\
&\leqslant [C_{l_k}(1 + \frac{2}{\kappa - 1})]^{-1}\|x_k - x_k^*\|_A + \|x_k^* - x^*\|_A \\
&\leqslant [C_{l_k}(1 + \frac{2}{\kappa - 1})]^{-1}\|x_k - x^* + x^* - x_k^*\|_A + \|x_k^* - x^*\|_A
\end{aligned}$$

$$
\begin{aligned}
&\leqslant [C_{l_k}(1+\frac{2}{\kappa-1})]^{-1}\left(\|x_k-x^*\|_A+\|x^*-x_k^*\|_A\right)+\|x_k^*-x^*\|_A\\
&\leqslant [C_{l_k}(1+\frac{2}{\kappa-1})]^{-1}\left(\|x_k-x^*\|_A+\|A^{-1}\|_2\|x_k-x^*\|_A\right)+\|A^{-1}\|_2\|x_k-x^*\|_A\\
&\leqslant \left\{\frac{2\sigma^{l_k}}{1+\sigma^{2l_k}}(1+\eta)+\eta\right\}\|x_k-x^*\|_A\\
&\leqslant \left\{\frac{2\sigma^{l_k}}{1+\sigma^{2l_k}}(1+\eta)+\eta\right\}\left\{\frac{2\sigma^{l_{k-1}}}{1+\sigma^{2l_{k-1}}}(1+\eta)+\eta\right\}\cdots\left\{\frac{2\sigma^{l_1}}{1+\sigma^{2l_1}}(1+\eta)+\eta\right\}\|x_1-x^*\|_A\\
&\leqslant \left\{\frac{2\sigma}{1+\sigma^2}(1+\eta)+\eta\right\}^k\|x_1-x^*\|_A.
\end{aligned}
$$

Since $\eta=\|A^{-1}\|_2<1$, then $\frac{2\sigma}{1+\sigma^2}(1+\eta)+\eta<1$, when $k\to\infty$, the right hand side of above equation tends to 0, thus the iteration sequence x_k converges to x^*. This completes the proof. □

5. Numerical experiments

In this section, we give some numerical experiments to illustrate Picard splitting iteration Algorithm 3.1, Algorithm 3.5 and Picard CG Algorithm 4.1. The iterations have been carried out by MATLAB R2012b (8.0.0). In view of the influence of round-off errors, we regard a residual r as the zero vector if $\|r\|_2<10^{-11}$, where $\|.\|_2$ denotes the 2-norm of the vector.

Example 5.1. We consider Eq. (1.1) with the following matrix:

$$
A1=\begin{pmatrix}
2 & 1 & 2 & -3 & 4\\
3 & 4 & 2 & 2 & 1\\
0 & 4 & 7 & 2 & 4\\
-1 & -1 & -1 & 2 & 4\\
4 & 4 & 3 & 2 & 1
\end{pmatrix}.
$$

Table 1: Numerical result of Example 5.1.

	P-HSS	P-SAOR	P-SSOR	P-JACOBI	P-CG
linter number	38	43	39	39	42
cpu time(second)	0.0028	0.0045	0.0012	0.0018	0.0022
residual (e-10)	0.9079	0.9804	0.7878	0.6946	0.8247

The numerical experiments are shown in Table 1. From Table 1, it shows that P-HSS, P-SAOR, P-SSOR, and P-JACOBI algorithms are effective and their iterative number, cpu time, and residual are almost the same in size.

Example 5.2. We randomly choose a matrix A according to the following structure: $A=\text{round}(100*(\text{eye}(n;n)-0.02*(2*\text{rand}(n;n)-1))$, and choose $b(i)=(-i)^i, i=1,2,\cdots,n$. The random A has the structure for $n=6$ as follows

$$
A=\begin{pmatrix}
101 & 1 & 1 & 2 & -1 & 0\\
-1 & 99 & -2 & 0 & -1 & -2\\
1 & -1 & 101 & 2 & 2 & 1\\
0 & -1 & -1 & 98 & 0 & 1\\
1 & 0 & 0 & 2 & 101 & 1\\
0 & 2 & -2 & -1 & -1 & 101
\end{pmatrix}.
$$

Table 2: Numerical result of Example 5.2.

n		P-HSS	P-SAOR	P-SSOR	P-JACOBI	P-CG
500	linter number	11	7	6	6	18
	cpu time(second)	0.1215	0.0791	0.0205	0.0397	0.0040
	residual (e-10)	0.9977	0.9798	0.1023	0.3868	0.8139
1000	linter number	11	8	6	6	25
	cpu time(second)	0.5007	0.3772	0.0954	0.1953	0.0573
	residual (e-10)	0.9938	0.9234	0.2294	0.6016	0.4424
1500	linter number	11	9	6	6	31
	cpu time(second)	1.1322	0.9344	0.2427	0.4852	0.1613
	residual (e-10)	0.9657	0.9286	0.5023	0.5266	0.6099
2000	linter number	11	11	6	6	38
	cpu time(second)	2.0305	1.8382	0.4888	0.9111	0.3541
	residual (e-10)	0.9335	0.9824	0.9706	0.8667	0.8619

Numerical results for different values of n ($n = 500, 1000, 1500, 2000$) are reported in Table 2. From Table 2, the iterative number of P-HSS, P-SAOR, P-SSOR, and P-JACOBI method keep unchanged as order n increased. P-SSOR and P-JACOBI have the same iterative number, which is less than the other algorithm's. It is clear that the cpu time of the P-CG is less than the other algorithm's.

Example 5.3. Choose matrix $A = \text{spdiags}([-e, 4 * e, -e], [-1, 0, 1], n, n)$, and $b(i) = (-i)^i, i = 1, 2, \cdots, n$. The matrix A has a structure for $n = 6$ as follows

$$A = \begin{pmatrix} 4 & -1 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & 0 & 0 \\ 0 & -1 & 4 & -1 & 0 & 0 \\ 0 & 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -1 & 4 \end{pmatrix}.$$

Table 3: Numerical result of Example 5.3.

n		P-HSS	P-SAOR	P-SSOR	P-JACOBI	P-CG
500	linter number	19	22	19	20	22
	cpu time(second)	0.1599	0.3251	0.0756	0.2252	0.0039
	residual (e-10)	0.3968	0.9763	0.2865	0.5890	0.8593
1000	linter number	19	22	19	20	22
	cpu time(second)	0.7091	1.0285	0.3309	0.7877	0.0524
	residual (e-10)	0.3980	0.9803	0.2872	0.5910	0.7093
1500	linter number	19	22	19	20	22
	cpu time(second)	1.3815	2.2340	0.7224	1.6873	0.1157
	residual (e-10)	0.3984	0.9816	0.2874	0.5917	0.6345
2000	linter number	19	22	19	20	22
	cpu time(second)	2.3650	3.9210	1.2623	2.9596	0.2060
	residual (e-10)	0.3986	0.9823	0.2875	0.5921	0.5855

Numerical results for different values of n($n = 500, 1000, 1500, 2000$) are reported in Table 3. From Table 3, we can see all algorithms keep unchanged as the order n is increasing. Those algorithms we propose have almost the same iteration number. The cpu time of the P-CG is much less than the other algorithms.

Example 5.4. We consider the nonsymmetric matrix

$$A = \begin{pmatrix} 3 & 2 & 2 & \cdots & 2 \\ 0 & 3 & 2 & \cdots & 2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 3 & 2 \\ 2 & 2 & 2 & 2 & 3 \end{pmatrix}.$$

In the following, we give the result by running P-SSOR, P-SAOR and P-HSS algorithms.

Table 4: Numerical result of Example 5.4.

n		P-HSS	P-SAOR	P-SSOR
500	linter number	50	21	13
	cpu time(second)	0.0028	0.0116	0.0050
	residual (e-10)	Fail	9.6847	9.5762

In Table 4, as seen, both of P-SSOR and P-SAOR methods provide quite suitable results. However, we see that the Picard-HSS method fail to converge in maximum iterations (in tables it is denoted by Fail).

6. Conclusion

In this paper, we propose Picard splitting iteration method and Picard CG method for solving absolute value equation based on the Picard iteration and prove convergence of those algorithms. At last, we give several numerical experiments to compare the efficiency among the Picard splitting iteration methods and Picard CG method.

Acknowledgment

The authors thank the anonymous referee for helping to improve the original manuscript by valuable suggestions. The research was supported by Fujian Natural Science Foundation (Grant No. 2016J01005) and Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB18010202).

References

[1] Z.-Z. Bai, X. Yang, *On HSS-based iteration methods for weakly nonlinear systems*, Appl. Numer. Math., **59** (2009), 2923–2936. 1
[2] R. W. Cottle, G. B. Dantzig, *Complementary pivot theory of mathematical programming*, Linear Algebra and Appl., **1** (1968), 103–125. 1
[3] R. W. Cottle, J.-S. Pang, R. E. Stone, *The linear complementarity problem*, Computer Science and Scientific Computing, Academic Press, Inc., Boston, MA, (1992). 1
[4] T.-X. Gu, X.-W. Xu, X.-P. Liu, H.-B. An, X.-D. Hang, *Iterative methods and preconditioning techniques*, (Chinese) Science Press, Beijing, (2015). 4.2
[5] S.-L. Hu, Z.-H. Huang, *A note on absolute value equations*, Optim. Lett., **4** (2010), 417–424. 1
[6] S. Ketabchi, H. Moosaei, *An efficient method for optimal correcting of absolute value equations by minimal changes in the right hand side*, Comput. Math. Appl., **64** (2012), 1882–1885.
[7] S. Ketabchi, H. Moosaei, *Minimum norm solution to the absolute value equation in the convex case*, J. Optim. Theory Appl., **154** (2012), 1080–1087.
[8] S. Ketabchi, H. Moosaei, S. Fallahi, *Optimal error correction of the absolute value equation using a genetic algorithm*, Math. Comput. Model., **57** (2013), 2339–2342.
[9] O. L. Mangasarian, *Linear complementarity problems solvable by a single linear program*, Math. Programming, **10** (1976), 263–270.
[10] O. L. Mangasarian, *Absolute value equation solution via concave minimization*, Optim. Lett., **1** (2007), 3–8.
[11] O. L. Mangasarian, *Absolute value programming*, Comput. Optim. Appl., **36** (2007), 43–53.
[12] O. L. Mangasarian, *A generalized Newton method for absolute value equations*, Optim. Lett., **3** (2009), 101–108. 1

[13] O. L. Mangasarian, *Primal-dual bilinear programming solution of the absolute value equation*, Optim. Lett., **6** (2012), 1527–1533.
[14] O. L. Mangasarian, *Absolute value equation solution via dual complementarity*, Optim. Lett., **7** (2013), 625–630.
[15] O. L. Mangasarian, *Absolute value equation solution via linear programming*, J. Optim. Theory Appl., **161** (2014), 870–876. 1
[16] O. L. Mangasarian, R. R. Meyer, *Absolute value equations*, Linear Algebra Appl., **419** (2006), 359–367. 1, 3.2
[17] M. A. Noor, J. Iqbal, K. I. Noor, E. Al-Said, *On an iterative method for solving absolute value equations*, Optim. Lett., **6** (2012), 1027–1033. 1
[18] J. M. Ortega, W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York-London, (1970). 2.1, 2.2
[19] O. Prokopyev, *On equivalent reformulations for absolute value equations*, Comput. Optim. Appl., **44** (2009), 363–372. 1
[20] J. Rohn, *A theorem of the alternatives for the equation* $Ax + B|x| = b$, Linear Multilinear Algebra, **52** (2004), 421–426.
[21] J. Rohn, *An algorithm for solving the absolute value equation*, Electron. J. Linear Algebra, **18** (2009), 589–599.
[22] J. Rohn, *On unique solvability of the absolute value equation*, Optim. Lett., **3** (2009), 603–606.
[23] J. Rohn, *An algorithm for computing all solutions of an absolute value equation*, Optim. Lett., **6** (2012), 851–856.
[24] J. Rohn, V. Hooshyarbakhsh, R. Farhadsefat, *An iterative method for solving absolute value equations and sufficient conditions for unique solvability*, Optim. Lett., **8** (2014), 35–44. 1
[25] D. K. Salkuyeh, *The Picard-HSS iteration method for absolute value equations*, Optim. Lett., **8** (2014), 2191–2202. 1, 3.10
[26] A.-X. Wang, H.-J. Wang, Y.-K. Deng, *Interval algorithm for absolute value equations*, Cent. Eur. J. Math., **9** (2011), 1171–1184. 1