Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 7(10): 18675–18689.
DOI: 10.3934/math.20221027
Received: 23 May 2022
Revised: 03 August 2022
Accepted: 08 August 2022
Published: 22 August 2022

Research article

Two effective inexact iteration methods for solving the generalized absolute value equations

Miao Guo and Qingbiao Wu*

Department of Mathematics, Zhejiang University, Hangzhou, 310018, Zhejiang, China

* **Correspondence:** Email: qbwu@zju.edu.cn.

Abstract: Modified Newton-type methods are efficient for addressing the generalized absolute value equations. In this paper, to further speed up the modified Newton-type methods, two new inexact modified Newton-type iteration methods are proposed. The sufficient conditions for the convergence of the two proposed inexact iteration methods are given. Moreover, to demonstrate the efficacy of the new method, several numerical examples are provided.

Keywords: generalized absolute value equation; linear complementarity problem; modified Newton-type method; inexact iteration method; convergence analysis
Mathematics Subject Classification: 47H10, 65H10

1. Introduction

The main research of this article is about the generalized absolute value equation (GAVE):

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

where $A, B \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ and $|x| := (|x_1|, |x_2|, \cdots, |x_n|)^T$. Actually, when B is a identity matrix, the GAVE (1.1) can be easily transformed into the absolute value equation (AVE):

$$Ax - |x| = b. \tag{1.2}$$

The GAVE (1.1) was first proposed by Rohn as a class of nonlinear nondifferentiable optimization problems [1]. After that, the AVE (1.2) and GAVE (1.1) has applications in many areas of optimization, including linear complementarity problem, bimatrix games, constrained least squares problems, and so on, see [1–6]. For instance, the famous linear complementarity problem (LCP) [5]: Determining z such that

$$Mz + q \geq 0,\ z \geq 0 \ and\ z^T(Mz + q) = 0,\ with\ M \in \mathbb{R}^{n\times n},\ q \in \mathbb{R}^n. \tag{1.3}$$

In [3], Mangasarian has proved that the GAVE (1.1) and LCP (1.3) can be transformed into each other under some conditions, and he also proposed the unique solvability of the AVE. Moreover, many scholars have further studied the solvability and unique solution theory of the AVE (1.2) [7–9]. In the early algorithm research, some efficient iteration methods have been proposed to address the GAVE (1.1) and AVE (1.2), which can be found in Mangasarian [2, 10, 11], Rohn [12], and so on. Then in [13], in order to overcome the difficulty of constructing the gradient of the absolute value term, Mangasarian presented the generalized Newton (GN) algorithm through using the generalized Jacobian matrix.This method may be summed up as follows:

$$x^{(k+1)} = (A - D(x^{(k)}))^{-1}b, \quad k = 0, 1, 2, ..., \tag{1.4}$$

where $D(x) := diag(sgn(x))$, $x \in \mathbb{R}^n$, $sgn(x)$ denotes a vector with components equal to 1, 0, -1 depending on whether the corresponding components of x is positive, zero or negative. Moreover, the GN algorithm is also able to apply on the GAVE (1.1) if we use $BD(x^{(k)})$ instead of $D(x^{(k)})$. Based on the GN method, Li [14] proposed a modified generalized Newton method. It should be noticed that the Jacobian matrix is changed with respect to the iteration index k, which will increase the amount of computation greatly. In order to stay away from the changed Jacobian matrix, many literature on the result of the AVE (1.2) and GAVE (1.1) are proposed. For instance, Rohn et al. presented the Picard method when A is invertible in [15]:

$$x^{(k+1)} = A^{-1}(|x^{(k)}| + b), \quad k = 0, 1, 2, ... \tag{1.5}$$

In [16], Wang et al. added a new term Ωx (Ω is positive semi-definite and it can be guaranteed that $A + \Omega$ is invertible) into the general problem and noticed that $Ax + \Omega x$ is differential but $\Omega x + B|x| + b$ non-differential, then studied a modified Newton-type(MN) iteration method:

$$x^{(k+1)} = (A + \Omega)^{-1}(\Omega x^{(k)} + B|x^{(k)}| + b), \quad k = 0, 1, 2, ..., \tag{1.6}$$

To further speed up the AOR method, Li et al. [17] presented the generalization version algorithm. Actually, the above algorithms all belong to Newton-based matrix splitting methods and their relaxed versions, which is proposed in [18]. Sometimes the coefficient matrix A has some special properties, for example, A is M-matrix, under which condition Ali et al. [19] proposed two new generalized Gauss-Seidel iteration methods. In [20], a new iteration method was presented by redesigning equivalently the AVE in (1.2) as a 2×2 block nonlinear equation. Also using matrix blocking, the block-diagonal and anti-block-diagonal splitting (BAS) method [21] and modified BAS method [22] are proposed.

Recently, in order to continue to improve the efficiency of solving the AVE (1.2), some inexact iteration algorithms have been studied. Every step in iteration of the Picard method can be viewed as solving a linear system with the coefficient matrix A. Therefore, Salkuyeh [23] presented the Picard-HSS method for AVE (1.2), which used the HSS method [24] to estimate the result $x^{(k+1)}$ at each Picard iteration. On this basis, Miao et al. [25] used single-step HSS (SHSS) method [26] to address the linear system in Picard iteration and proposed the Picard-SHSS method, a new Picard-type method. Inspired by these, we adopt the modified Newton-type iteration method with faster convergence speed as the outer iterative method, HSS and SHSS as the inner iterative method respectively, then we obtain two new inexact iterative algorithms, abbreviated as MN-HSS method and MN-SHSS method.

The remainder of this article is laid out as follows. In Section 2, the MN-HSS method is described, and we investigate its convergence property. Section 3 is devoted to introducing the MN-SHSS method and analyzing its convergence. Section 4 contains several numerical experiments, showing the feasibility and efficiency of the two new inexact iteration methods. Section 5 draws some conclusions.

2. The MN-HSS method for solving GAVE

At the beginning of this section, some notations are given blow. $ones(n,1)$ represents a vector like $(1,1,\cdots,1)^T \in \mathbb{R}$. $\rho(A)$ indicates the spectral radius of A. The symbol $(\cdot)^T$ represents the transepose of a vector or matrix. And we have $\lim\limits_{k\to+\infty} A^k = 0$ if and only if $\rho(A) < 1$ (see [27,28]).

Given $A \in \mathbb{R}^{n\times n}$ be a non-Hermitian positive definite matrix. Then $H = \frac{1}{2}(A + A^T)$, $S = \frac{1}{2}(A - A^T)$, which are the Hermitian part and skew-Hermitian part of A, $A = H + S$. Then the HSS iteration method was presented by Bai et al. [24] for addressing positive definite system $Ax = b$. The iterative process of the HSS method can be expressed as follows:

$$\begin{cases} x^{(l+\frac{1}{2})} = (\alpha I + H)^{-1}((\alpha I - S)x^{(l)} + b), \\ x^{(l+1)} = (\alpha I + S)^{-1}((\alpha I - H)x^{(l+\frac{1}{2})} + b), \end{cases} \tag{2.1}$$

where α is a positive constant.

Section 1 has introduced the MN method, which is equivalent to the following form

$$(A + \Omega)x^{(k+1)} = \Omega x^{(k)} + B|x^{(k)}| + b, \quad k = 0, 1, 2, \dots . \tag{2.2}$$

Similar to the Picard-HSS method, we can choose a suitable Ω to make sure the positive definiteness of $A + \Omega$. The HSS method (2.1) can be applied as the inner iteration method of the MN method for solving the GAVE (1.1), called MN-HSS method.

Algorithm (The MN-HSS iteration method):

For $k = 0, 1, \dots$, until converge, do:

Set $x^{(k,0)} := x^{(k)}$

For $l = 0, 1, \dots$, until converge, do:

$$\begin{cases} x^{(k,l+\frac{1}{2})} = (\alpha I + H)^{-1}((\alpha I - S)x^{(k,l)} + \Omega x^{(k)} + B|x^{(k)}| + b), \\ x^{(k,l+1)} = (\alpha I + S)^{-1}((\alpha I - H)x^{(k,l+\frac{1}{2})} + \Omega x^{(k)} + B|x^{(k)}| + b), \end{cases} \tag{2.3}$$

where $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T)$, $S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$, α is a positive constant.

Endfor

Set $x^{(k+1)} := x^{(k,l_k)}$.

Endfor

The following theorem proves that the MN-HSS algorithm converges under necessary conditions.

Theorem 2.1. *Let the matrix $A, B \in \mathbb{R}^{n\times n}$ and choose suitable $\Omega \in \mathbb{R}^{n\times n}$ such that $A + \Omega$ is a positive definite matrix, then $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T)$, $S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$ are the Hermitian and skew-Hermitian parts of $A + \Omega$ respectively. Let also $\| A^{-1}B \|_2 < 1$ and $\| (A + \Omega)^{-1} \|_2 < \frac{1}{\|B\|_2 + \|\Omega\|_2}$. Then the GAVE (1.1) has a unique solution x^*, and for any starting vector $x^{(0)} \in \mathbb{R}^n$ and any sequence of positive integers l_k, $k = 0, 1, \dots$, the iteration sequence $\{x^{(k)}\}_{k=0}^{\infty}$ produced by the MN-HSS iteration method converges to x^* provided that $l = \liminf\limits_{k\to\infty} l_k \geq N$, where N is a natural number satisfying*

$$\| T_\alpha^s \|_2 < \frac{1 - \| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| \Omega \|_2)}{\| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| A \|_2)}, \ \forall s > N,$$

where $T_\alpha = M_\alpha^{-1}N_\alpha,\ M_\alpha = \frac{1}{2\alpha}(\alpha I + H)(\alpha I + S),\ N_\alpha = \frac{1}{2\alpha}(\alpha I - H)(\alpha I - S).$

Proof. Let $G_\alpha = M_\alpha^{-1}$, then according to the iteration formula (2.3), we can write the (k+1)th iterate $x^{(k+1)}$ of the MN-HSS iteration method as

$$x^{(k+1)} = T_\alpha^{l_k} x^{(k)} + \sum_{j=0}^{l_k-1} T_\alpha^j G_\alpha(\Omega x^{(k)} + B|x^{(k)}| + b),\ k = 0, 1, 2, ... \tag{2.4}$$

Since $\| A^{-1}B \|_2 < 1$, the GAVE (1.1) has a unique result x^* [3], which satisfying

$$x^* = T_\alpha^{l_k} x^* + \sum_{j=0}^{l_k-1} T_\alpha^j G_\alpha(\Omega x^* + B|x^*| + b),\ k = 0, 1, 2, ... \tag{2.5}$$

By subtracting (2.5) from (2.4), we have

$$x^{(k+1)} - x^* = T_\alpha^{l_k}(x^{(k)} - x^*) + \sum_{j=0}^{l_k-1} T_\alpha^j G_\alpha(\Omega(x^{(k)} - x^*) + B|x^{(k)}| - B|x^*|). \tag{2.6}$$

It follows from [24] that $\rho(T_\alpha) < 1$, then

$$\begin{aligned}
\sum_{j=0}^{l_k-1} T_\alpha^j G_\alpha &= (I - T_\alpha^{l_k})(I - T_\alpha)^{-1} G_\alpha \\
&= (I - T_\alpha^{l_k})(I - M_\alpha^{-1}N_\alpha)^{-1} M_\alpha^{-1} \\
&= (I - T_\alpha^{l_k})(M_\alpha - N_\alpha)^{-1} \\
&= (I - T_\alpha^{l_k})(A + \Omega)^{-1}.
\end{aligned}$$

Then the Eq (2.6) turns into

$$\begin{aligned}
x^{(k+1)} - x^* &= T_\alpha^{l_k}(x^{(k)} - x^*) + (I - T_\alpha^{l_k})(A + \Omega)^{-1}(\Omega(x^{(k)} - x^*) + B|x^{(k)}| - B|x^*|) \\
&= T_\alpha^{l_k}((A + \Omega)^{-1}A(x^{(k)} - x^*) - (A + \Omega)^{-1}B(|x^{(k)}| - |x^*|)) + (A + \Omega)^{-1}(\Omega(x^{(k)} - x^*) + B|x^{(k)}| - B|x^*|).
\end{aligned}$$

It is obvious that for any $x, y \in \mathbb{R}^n,\ \| |x| - |y| \|_2 \leq \| x - y \|_2$. Therefore,

$$\begin{aligned}
\| x^{(k+1)} - x^* \|_2 \leq [\ &\| T_\alpha^{l_k} \|_2 (\| (A + \Omega)^{-1} \|_2 \| A \|_2 + \| (A + \Omega)^{-1} \|_2 \| B \|_2) + \| (A + \Omega)^{-1} \|_2 \| \Omega \|_2 \\
&+ \| (A + \Omega)^{-1} \|_2 \| B \|_2\ ]\ \| x^{(k)} - x^* \|_2\ .
\end{aligned}$$

On the other hand, since $\rho(T_\alpha) < 1$, $\lim_{s\to\infty} T_\alpha^s = 0$. Therefore, there is a natural number N such that

$$\| T_\alpha^s \|_2 < \frac{1 - \| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| \Omega \|_2)}{\| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| A \|_2)},\ \forall s \geq N.$$

Thus, assume that $l = \liminf_{k\to\infty} l_k \geq N$, then we can obtain the result. This completes the proof. ■

□

The MN-HSS method can also be expressed as the residual-updating form:
Algorithm (The residual-updating variant of MN-HSS iteration method):
For $k = 0, 1, ...,$ until converge, do:
Set $d^{(k,0)} = 0$ and $b^{(k)} = B|x^{(k)}| + b - Ax^{(k)}$
For $l = 0, 1, ...,$ until converge, do:

$$\begin{cases} d^{(k,l+\frac{1}{2})} = (\alpha I + H)^{-1}((\alpha I - S)d^{(k,l)} + b^{(k)}), \\ d^{(k,l+1)} = (\alpha I + S)^{-1}((\alpha I - H)d^{(k,l+\frac{1}{2})} + b^{(k)}), \end{cases}$$

where $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T),\ S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$, α is a positive constant.
Endfor
Set $x^{(k+1)} = x^{(k)} + d^{(k,l_k)}$.
Endfor

3. The MN-SHSS method for solving GAVE

From the iteration scheme (2.3) we can notice that the each step of HSS iteration method is equivalent to solving two linear subsystems, whose coefficient matrices are Hermitian and skew-Hermitian respectively. The first equation with Hermitian coefficient matrix can be solved quickly by conjugate gradient method or Cholesky factorization. However, the other subsystem needs much more computation. So as to accelerate the algorithm, we replace the inner iteration method with single-step HSS (SHSS) method, then we present a new inexact method in this section, named as MN-SHSS iteration method.

Compared with the HSS method, the SHSS method ignores the second subsystem with skew-Hermitian matrix, which can be expressed as

$$(\alpha I + H)x^{(l+1)} = (\alpha I - S)x^{(l)} + b, \tag{3.1}$$

by limiting the iteration parameter α, Li et al. [26] has proved that the SHSS converges to the unique solution for any initial guess $x^{(0)}$.
Algorithm (The MN-SHSS iteration method):
For $k = 0, 1, ...,$ until converge, do:
Set $x^{(k,0)} := x^{(k)}$
For $l = 0, 1, ...,$ until converge, do:

$$x^{(k,l+1)} = (\alpha I + H)^{-1}((\alpha I - S)x^{(k,l)} + \Omega x^{(k)} + B|x^{(k)}| + b), \tag{3.2}$$

where $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T),\ S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$, α is a positive constant.
Endfor
Set $x^{(k+1)} := x^{(k,l_k)}$.
Endfor

The next theorem shows that the MN-SHSS method is convergent under a restriction on the parameter α.

Theorem 3.1. *Let the matrix $A, B \in \mathbb{R}^{n\times n}$ and choose suitable $\Omega \in \mathbb{R}^{n\times n}$ such that $A + \Omega$ is a positive definite matrix, then $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T),\ S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$ are the Hermitian and skew-Hermitian parts of $A + \Omega$ respectively. Let also $\| A^{-1}B \|_2 < 1$, $\| (A + \Omega)^{-1} \|_2 < \frac{1}{\|B\|_2+\|\Omega\|_2}$, and α be a constant number such that $\alpha > max\{0, \frac{\delta_{max}^2-\lambda_{min}^2}{2\lambda_{min}}\}$, where δ_{max} is the largest singular value of S and λ_{min} is the smallest eigenvalue of H. Then the GAVE (1.1) has a unique solution x^*, and for any starting vector $x^{(0)} \in \mathbb{R}^n$ and any sequence of positive integers $l_k,\ k = 0, 1, ...,$ the iteration sequence $\{x^{(k)}\}_{k=0}^{\infty}$ produced by the MN-SHSS iteration method converges to x^* provided that $l = \liminf_{k\to\infty} l_k \geq N$, where N is a natural number satisfying*

$$\| K_\alpha^s \|_2 < \frac{1- \| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| \Omega \|_2)}{\| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| A \|_2)},\ \forall s > N,$$

where $K_\alpha = M_\alpha^{-1}N_\alpha,\ M_\alpha = \alpha I + H,\ N_\alpha = \alpha I - S$.

Proof. According to the iteration formula (3.2), we can get the kth iteration $x^{(k+1)}$ of the MN-SHSS method:

$$x^{(k+1)} = K_\alpha^{l_k} x^{(k)} + \sum_{j=0}^{l_k-1} K_\alpha^j M_\alpha^{-1}(\Omega x^{(k)} + B|x^{(k)}| + b). \tag{3.3}$$

On the other hand, the unique result x^* of GAVE (1.1) satisfies

$$x^* = K_\alpha^{l_k} x^* + \sum_{j=0}^{l_k-1} K_\alpha^j M_\alpha^{-1}(\Omega x^* + B|x^*| + b). \tag{3.4}$$

Subtracting (3.4) from (3.3), then we have

$$x^{(k+1)} - x^* = K_\alpha^{l_k}(x^{(k)} - x^*) + \sum_{j=0}^{l_k-1} K_\alpha^j M_\alpha^{-1}(\Omega(x^{(k)} - x^*) + B|x^{(k)}| - B|x^*|), \tag{3.5}$$

In [26], Li et al. has proved that $\rho(K_\alpha) < 1$ when $\alpha > max\{0, \frac{\delta_{max}^2-\lambda_{min}^2}{2\lambda_{min}}\}$, hence $\sum_{j=0}^{l_k-1} K_\alpha^j M_\alpha^{-1} = (I - K_\alpha^{l_k})(A + \Omega)^{-1}$. Then (3.5) becomes

$$x^{(k+1)} - x^* = K_\alpha^{l_k}(x^{(k)} - x^*) + (I - K_\alpha^{l_k})(A + \Omega)^{-1}(\Omega(x^{(k)} - x^*) + B|x^{(k)}| - B|x^*|). \tag{3.6}$$

Therefore, we can calculate that

$$\begin{aligned} \| x^{(k+1)} - x^* \|_2 \leq [\| K_\alpha^{l_k} \|_2 (\| (A + \Omega)^{-1} \|_2 \| A \|_2 + \| (A + \Omega)^{-1} \|_2 \| B \|_2) + \| (A + \Omega)^{-1} \|_2 \| \Omega \|_2 \\ + \| (A + \Omega)^{-1} \|_2 \| B \|_2]\ \| x^{(k)} - x^* \|_2\ . \end{aligned} \tag{3.7}$$

Due to the condition of $\rho(K_\alpha) < 1$, we can get that $K_\alpha^s \to 0$ as $s \to \infty$. So there is a natural number N such that

$$\| K_\alpha^s \|_2 < \frac{1- \| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| \Omega \|_2)}{\| (A + \Omega)^{-1} \|_2 (\| B \|_2 + \| A \|_2)},\ \forall s \geq N.$$

Then if $l = \liminf_{k\to\infty} l_k \geq N$, it is distinct that $\| x^{(k+1)} - x^* \|_2 < \| x^{(k)} - x^* \|_2$. That is to say, the iteration sequence $\{x^{(k)}\}_{k=0}^{\infty}$converges to x^*, which is produced by the MN-SHSS iteration method. This completes the proof. ■

□

In the same way, the MN-SHSS method also has residual form:

Algorithm (residual-updating variant of the MN-SHSS iteration method):

For $k = 0, 1, ...,$ until converge, do:

Set $d^{(k,0)} = 0$ and $b^{(k)} = B|x^{(k)}| + b - Ax^{(k)}$

For $l = 0, 1, ...,$ until converge, do:

$$d^{(k,1+1)} = (\alpha I + H)^{-1}((\alpha I - S)d^{(k,l)} + b^{(k)}),$$

where $H = \frac{1}{2}((A + \Omega) + (A + \Omega)^T),\ S = \frac{1}{2}((A + \Omega) - (A + \Omega)^T)$, α is a positive constant.

Endfor

Set $x^{(k+1)} = x^{(k)} + d^{(k,l_k)}$.

Endfor

4. Numerical experiments

In this section, the effectiveness of MN-HSS method and MN-SHSS method is showed by comparing them with other existing inexact iteration method for solving GAVE (1.2): the Picard-HSS method and the Picard-SHSS method. In addition, we also compare our methods with a descent method [29]. For this purpose, we compare from the following aspects: the number of outer iteration steps (denoted by 'IT'), the elapsed CPU time (denoted by 'CPU') and the residual error(denoted by 'RES'), where the 'RES' is set to be

$$RES := \frac{\| Ax^{(k)} - B|x^{(k)}| - b \|_2}{\| b \|_2}.$$

In our computations, we all use the residual-updating version and optimal parameters α resulting in the least iteration numbers of the algorithms mentioned above and the zero vector as our starting iteration vector. We use the Cholesky factorization to solve the equations whose coefficient matrix is $\alpha I + H$ and the LU factorization for the other subsystem. All runs are aborted once the residual error satisfies $RES <= 10^{-7}$ or the maximum iteration number $k_{max} = 500$ is exceeded. As for the inner iterations, we set the stopping criterion of different methods as:

$$\frac{\| b^{(k)} - Ad^{(k,l)} \|_2}{\| b^k \|_2} \leq 0.01,$$

and a limit on the number of iterations $10(l_k = 10,\ k = 1, 2, ...)$ for inner iterations are used. The parameters used in the desent method are chosen as $\sigma = 0.2$, $\delta = 0.8$,$\gamma = 0.001$, $p = 3$ and $\epsilon_0 = 0.01$.

The two numerical experiments below are performed in MATLAB R2020b.

Example 1. [30] The first example comes from the LCP (1.3). In [3, 30], it has been known that the LCP (1.3) can be transformed into the following form:

$$(M + I)x - (M - I)|x| = q \quad with\ x = \frac{1}{2}((M - I)z + q), \tag{4.1}$$

which belongs to the GAVE (1.1) obviously. Then we let $A = M + I$ and $B = M - I$, where $M = \hat{M} + \mu I$

and the right-hand vector q is determined by $q = -Mz^*$, in which

$$\hat{M} = Tridiag(-I, S, -I) = \begin{bmatrix} S & -I & 0 & \cdots & 0 & 0 \\ -I & S & -I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -I & S \end{bmatrix} \in \mathbb{R}^{n \times n}$$

is a block tridiagonal matrix,

$$S = Tridiag(-1, 4, -1) = \begin{bmatrix} 4 & -1 & 0 & \cdots & 0 & 0 \\ -1 & 4 & -1 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -1 & 4 \end{bmatrix} \in \mathbb{R}^{m \times m}$$

is a tridiagonal matrix, $n = m^2$. we can know that $z^* = 1.2 * ones(n, 1)$ is the unique result of the LCP(1.3) and the exact solution of the GAVE(4.1) is $x^* = 0.6 * ones(n, 1)$.

In this experiment, we pick three scenarios for the parameter $\mu = 4, -1, -4$. The matrices A and M are both symmetric positive definite when $\mu = 4$, and symmetric indefinite when $\mu = -4$. The matrix A is symmetric positive definite if $\mu = -1$, but M symmetric indefinite. We choose four different sizes of n ($n = 3600, 4900, 6400, 8100$) for each parameter μ. Considering the simplicity and efficiency of our experiments, we use the same matrix $\Omega = 5I$ in the MN-HSS and MN-SHSS methods. In the methods we test, the value α is chosen to result in the fewest iteration steps, see Table 1. In Tables 2–4, we list the numerical outcomes of the studied four inexact iteration methods for $\mu = 4, -1, -4$. In these tables, "-" indicates that the relevant algorithm is not convergent to the answer under the limitation of the maximum number k_{max} of iterative steps or even diverge.

Table 1. The values of α for the testing methods in Example 1.

	Method	n=3600	n=4900	n=6400	n=8100
μ=4	Picard-HSS	7.6	7.5	7.6	7.5
	Picard-SHSS	0.8	0.1	0.1	0.1
	MN-HSS	11.4	11.6	11.2	11.4
	MN-SHSS	0.1	0.1	0.1	0.1
μ=−1	Picard-HSS	-	-	-	-
	Picard-SHSS	-	-	-	-
	MN-HSS	10.4	10.4	10.4	10.4
	MN-SHSS	0.4	0.1	0.3	0.1
μ=−4	Picard-HSS	-	-	-	-
	Picard-SHSS	-	-	-	-
	MN-HSS	8.7	7.9	7.8	7
	MN-SHSS	0.1	0.1	0.1	0.5

Table 2. Numerical results of the testing methods for Example 1 with $\mu = 4$.

Methods		n=3600	n=4900	n=6400	n=8100
	IT	63	62	63	62
Picard-HSS	CPU	7.2267	13.2534	23.4046	37.3651
	RES	7.8433e-8	8.7166e-8	7.8872e-8	8.7574e-8
	IT	72	73	73	72
Picard-SHSS	CPU	4.8826	7.5158	13.4642	20.8528
	RES	9.6906e-8	8.8579e-8	8.4765e-8	9.938e-8
	IT	3	3	3	3
The descent method	CPU	3.1777	7.5666	16.1594	31.5073
	RES	1.1654e-9	1.1643e-9	1.1635e-9	1.1628e-9
	IT	12	12	12	12
MN-HSS	CPU	1.8227	3.5410	6.6261	11.3629
	RES	3.7154e-8	3.9922e-8	4.8592e-8	3.8995e-8
	IT	12	12	12	12
MN-SHSS	CPU	0.7606	1.4825	2.6502	4.4347
	RES	4.9155e-8	4.9827e-8	5.0337e-8	5.0737e-8

Table 3. Numerical results of the testing methods for Example 1 with $\mu = -1$.

Methods		n=3600	n=4900	n=6400	n=8100
	IT	-	-	-	-
Picard-HSS	CPU	-	-	-	-
	RES	-	-	-	-
	IT	-	-	-	-
Picard-SHSS	CPU	-	-	-	-
	RES	-	-	-	-
	IT	6	6	6	6
The descent method	CPU	6.1136	14.769	31.1904	64.9632
	RES	2.4038e-9	2.0506e-9	1.7879e-9	1.5849e-9
	IT	69	68	68	68
MN-HSS	CPU	7.9967	14.7294	25.6464	43.1252
	RES	8.7811e-8	9.9034e-8	9.3516e-8	8.8168e-8
	IT	68	68	68	68
MN-SHSS	CPU	4.5587	7.8019	14.1982	22.0090
	RES	9.9988e-8	9.5126e-8	8.7701e-8	8.5202e-8

From Table 2, we can see that when $\mu = 4$, for all matrix scales the five evaluated methods can successfully create an estimated solution to the GAVE (4.1). And for each tested method, the CPU time also increases with the increase of the scale n of the coefficient matrix. Through the study of the numerical results, we can know that the two inexact iteration methods proposed by us are better than the Picard-HSS and Picard-SHSS methods in the number of iteration steps and CPU time. Although the descent method need less iteration steps, our methods spend less CPU time. When $\mu = -1, -4$, in which instance the convergence of Picard-HSS and Piard-SHSS iteration methods cannot be guaranteed, hence Tables 3 and 4 show that the two Picard-type methods converge slowly or do not converge, while the MN-HSS and MN-SHSS method can still converge fast. The selectivity of matrix Ω provides higher stability for the algorithm we propose. When $\mu = -1$, our methods still have better performance in elapsed CPU time than the descent method generally, and the descent method can not converge under constraints of maximum iteration number.

Table 4. Numerical results of the testing methods for Example 1 with $\mu = -4$.

Methods		n=3600	n=4900	n=6400	n=8100
	IT	-	-	-	-
Picard-HSS	CPU	-	-	-	-
	RES	-	-	-	-
	IT	-	-	-	-
Picard-SHSS	CPU	-	-	-	-
	RES	-	-	-	-
	IT	-	-	-	-
The descent method	CPU	-	-	-	-
	RES	-	-	-	-
	IT	46	46	46	46
MN-HSS	CPU	6.9628	13.1563	21.5587	36.8080
	RES	9.8167e-8	9.3627e-8	8.8073e-8	8.5249e-8
	IT	46	46	46	46
MN-SHSS	CPU	3.1611	5.7271	9.9884	16.9793
	RES	9.7483e-8	9.0358e-8	8.4596e-8	8.2290

Example 2. The coefficient matrix A of the GAVE (1.1) is from the finite difference approximation the two-dimensional convection-diffusion equation

$$\begin{cases} -(u_{xx} + u_{yy}) + q(u_x + u_y) + pu = f(x, y), & (x, y) \in S, \\ u(x, y) = 0, & (x, y) \in \partial S, \end{cases}$$

where $S = (0, 1) \times (0, 1)$, ∂S is its boundary, q is a constant and p is a real number. For discretization, the five-point finite difference scheme are used on the diffusive terms and the central difference scheme on the convective terms respectively, then we can obtain the coefficient matrix A.

$$A = T_x \otimes I_m + I_m \otimes T_x + pI_n,$$

where I_m and I_n are the identity matrices of order m and n with $n = m^2$, $\otimes$ means the Kronecker product, $T_x = tridiag(a_2, a_1, a_3)_{m\times m}$ and $T_y = tridiag(a_2, 0, a_3)_{m\times m}$ with $a_1 = 4$, $a_2 = -1 - Re$, $a_3 = -1 + Re$. Here $Re = \frac{qh}{2}$ and $h = \frac{1}{m+1}$ are the mesh Reynolds number and the equidistant step size, respectively. q is a positive constant. Let the coefficient matrix $B = I$ and the exact solution $x^* = (-1, 2, -3, \cdots, (-1)^i i, \cdots, (-1)^n n)^T$, then the right-hand side vector can also be determined. In this instance the GAVE becomes a AVE (1.2) actually.

In our experiments, we test different values of p ($p = 0, -1$), q ($q = 0, 1$) and n ($n = 100, 400, 1600, 6400$). The parameters α of the four tested methods resulting in the fewest iteration steps are chosen, while the matrix Ω in our methods are chosen as $\Omega = \omega I$, see Tables 5 and 6. We refer to [25] for the parameters of Picard-HSS and Picard-SHSS methods.

Table 5. The values of α and ω for the testing methods in Example 2 ($p = 0$).

	Methods	n=100	n=400	n=1600	n=6400
q=0	Picard-HSS	11.69	12.6	13.4	13
	Picard-SHSS	5.745	6.5	6.4	6.6
	MN-HSS	2.2 $\Omega = 0.5I$	2.5 $\Omega = 0.7I$	2.2 $\Omega = 0.7I$	2.2 $\Omega = 0.7I$
	MN-SHSS	0.1 $\Omega = 0.5I$	0.1 $\Omega = 0.7I$	0.1 $\Omega = 0.7I$	0.1 $\Omega = 0.7I$
q=1	Picard-HSS	12.01	13.6	14	13
	Picard-SHSS	5.926	6.6	6.75	6.6
	MN-HSS	2.2 $\Omega = 0.5I$	2.4 $\Omega = 0.7I$	2.2 $\Omega = 0.7I$	2 $\Omega = 0.7I$
	MN-SHSS	0.3 $\Omega = 0.5I$	0.1 $\Omega = 0.7I$	0.1 $\Omega = 0.7I$	0.1 $\Omega = 0.7I$

Table 6. The values of α and ω for the testing methods in Example 2 ($p = -1$).

	Methods	n=100	n=400	n=1600	n=6400
q=0	Picard-HSS	13.8	11.2	10.36	10.1
	Picard-SHSS	6.96	5.7	5.21	5.1
	MN-HSS	1.1 $\Omega = I$	0.6 $\Omega = I$	0.5 $\Omega = I$	0.3 $\Omega = I$
	MN-SHSS	0.1 $\Omega = I$	0.1 $\Omega = I$	0.01 $\Omega = I$	0.01 $\Omega = I$
q=1	Picard-HSS	14	11.29	10.4	11
	Picard-SHSS	7.05	5.72	5.2	5.1
	MN-HSS	1 $\Omega = I$	0.6 $\Omega = I$	0.3 $\Omega = I$	0.1 $\Omega = I$
	MN-SHSS	0.1 $\Omega = I$	0.1 $\Omega = I$	0.1 $\Omega = I$	0.1 $\Omega = I$

From Tables 7 and 8, the iteration steps reveal that the Picard-type methods and the descent method converges faster than the MN-HSS method, but in terms of the elapsed CPU times, the MN-SHSS method we new proposed has better computing efficiency generally. In Tables 9 and 10, that is when $p = -1$, the iteration counts and CPU time of the Picard-type methods increase dramatically with the increase of matrix scale, which situation is more obvious on the descent method. In contrast, our methods still maintain a low iteration counts and high computational efficiency. Therefore, our new proposed method is very effective.

Table 7. Numerical results of the testing methods for Example 2 with $p = 0$ and $q = 0$.

Methods		n=100	n=400	n=1600	n=6400
	IT	36	32	30	28
Picard-HSS	CPU	0.0149	0.2109	2.8908	41.9898
	RES	9.8815e-8	9.3643e-8	9.9900e-8	9.9043e-8
	IT	37	32	30	29
Picard-SHSS	CPU	0.0086	0.0747	1.5120	23.6302
	RES	9.4432e-8	9.6857e-8	9.7569e-8	9.9256e-8
	IT	5	5	5	5
The descent method	CPU	0.0160	0.0334	0.7701	26.7046
	RES	1.5532e-12	8.5090e-11	3.6631e-16	2.0878e-16
	IT	52	44	41	38
MN-HSS	CPU	0.0120	0.1491	2.4287	32.2431
	RES	9.3740e-8	8.0719e-8	7.6775e-8	9.9322e-8
	IT	51	44	41	39
MN-SHSS	CPU	0.0036	0.0294	0.5693	9.2022
	RES	8.1308e-8	8.0474e-8	7.9437e-8	7.6008e-8

Table 8. Numerical results of the testing methods for Example 2 with $p = 0$ and $q = 1$.

Methods		n=100	n=400	n=1600	n=6400
	IT	35	32	31	28
Picard-HSS	CPU	0.0131	0.2037	3.0079	42.4078
	RES	9.1372e-8	9.8614e-8	9.3177e-8	9.7012e-8
	IT	36	32	31	29
Picard-SHSS	CPU	0.0072	0.0782	1.5742	23.1650
	RES	9.8685e-8	9.4248e-8	9.4655e-8	9.6233e-8
	IT	6	5	5	5
The descent method	CPU	0.0048	0.0278	0.7420	26.1576
	RES	1.5836e-11	3.7231e-14	1.8960e-14	2.1382e-16
	IT	41	44	40	38
MN-HSS	CPU	0.0087	0.1642	2.1150	33.1043
	RES	8.5432e-8	9.9138e-8	9.6349e-8	9.4551e-8
	IT	41	44	40	38
MN-SHSS	CPU	0.0028	0.0299	0.5588	9.0679
	RES	8.4236e-8	9.8677e-8	9.9416e-8	9.8603e-8

Table 9. Numerical results of the testing methods for Example 2 with $p = -1$ and $q = 0$.

Methods		n=100	n=400	n=1600	n=6400
	IT	24	62	206	-
Picard-HSS	CPU	0.0096	0.3811	19.5610	-
	RES	9.6164e-8	9.5608e-8	9.8436e-8	-
	IT	21	52	166	-
Picard-SHSS	CPU	0.0047	0.1255	8.2082	-
	RES	9.6537e-8	9.7425e-8	9.9905e-8	-
	IT	19	-	-	-
The descent method	CPU	0.0116	-	-	-
	RES	3.1376e-12	-	-	-
	IT	5	10	32	73
MN-HSS	CPU	0.0031	0.0700	3.0579	103.9036
	RES	9.5409e-8	7.5888e-8	6.886e-8	9.7832e-8
	IT	4	5	4	6
MN-SHSS	CPU	0.0007	0.0160	0.1692	5.6192
	RES	2.6017e-8	3.1319e-8	3.9399e-9	3.1310e-8

Table 10. Numerical results of the testing methods for Example 2 with $p = -1$ and $q = 1$.

Methods		n=100	n=400	n=1600	n=6400
	IT	24	61	203	-
Picard-HSS	CPU	0.0095	0.3904	22.6934	-
	RES	9.7842e-8	9.9013e-8	9.3292e-8	-
	IT	21	51	166	-
Picard-SHSS	CPU	0.0043	0.1228	8.1885	-
	RES	9.5087e-8	9.5556e-8	9.5308e-8	-
	IT	-	-	-	-
The descent method	CPU	-	-	-	-
	RES	-	-	-	-
	IT	6	10	20	42
MN-HSS	CPU	0.0026	0.0735	1.9553	69.7912
	RES	1.0635e-8	6.2950e-8	5.5025e-8	9.4565e-8
	IT	3	5	13	48
MN-SHSS	CPU	0.0008	0.0138	0.6755	37.6225
	RES	8.1615e-8	6.7032e-8	7.7787e-8	9.9529e-8

5. Conclusions

In this article, to accelerate the MN iteration method, we propose the MN-HSS and MN-SHSS iteration algorithms for addressing the generalized absolute value equation. Moreover, we give the sufficient conditions to show that our methods are convergent for addressing the GAVE. In the end, we provide some experiments to comfirm the feasibility and effectiveness of our novel presented methods.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Grant no. 1227010254, Research on some efficient and fast algorithms for complex nonlinear problems).

Conflict of interest

The authors declare that they have no conflicts of interest.

References

1. J. Rohn, A theorem of the alternatives for the equation $Ax + B|x| = b$, *Linear Multilinear A.*, **52** (2004), 421–426. https://doi.org/10.1080/0308108042000220686
2. O. L. Mangasarian, Absolute value programming, *Comput. Optim. Appl.*, **36** (2007), 43–53. https://doi.org/10.1007/s10589-006-0395-5
3. O. L. Mangasarian, R. R. Meyer, Absolute value equations, *Linear Algebra Appl.*, **419** (2006), 359–367. https://doi.org/10.1016/j.laa.2006.05.004
4. S. L. Wu, P. Guo, Modulus-based matrix splitting algorithms for the quasi-complementarity problems, *Appl. Numer. Math.*, **132** (2018), 127–137. https://doi.org/10.1016/j.apnum.2018.05.017
5. R. W. Cottle, J. S. Pang, R. E. Stone, *The linear complementarity problem*, 1992.
6. N. Zheng, K. Hayami, J. F. Yin, Modulus-type inner outer iteration methods for nonnegative constrained least squares problems, *SIAM J. Matrix Anal. Appl.*, **37** (2016), 1250–1278. https://doi.org/10.1137/141002220
7. J. Rohn, On unique solvability of the absolute value equation, *Optim. Lett.*, **3** (2009), 603–606. https://doi.org/10.1007/s11590-009-0129-6
8. S. L. Wu, C. X. Li, The unique solution of the absolute value equations, *Appl. Math. Lett.*, **76** (2018), 195–200. https://doi.org/10.1016/j.aml.2017.08.012
9. S. L. Wu, C. X. Li, A note on unique solvability of the absolute value equation, *Optim. Lett.*, **14** (2020), 1957–1960. https://doi.org/10.1007/s11590-019-01478-x
10. O. L. Mangasarian, Absolute value equation solution via concave minimization, *Optim. Lett.*, **1** (2007), 3–8. https://doi.org/10.1007/s11590-006-0005-6
11. O. L. Mangasarian, Linear complementarity as absolute value equation solution, *Optim. Lett.*, **8** (2014), 1529–1534. https://doi.org/10.1007/s11590-013-0656-z
12. J. Rohn, An algorithm for solving the absolute value equations, *ELA*, **18** (2009), 589–599. https://doi.org/10.13001/1081-3810.1332
13. O. L. Mangasarian, A generalized Newton method for absolute value equations, *Optim. Lett.*, **3** (2009), 101–108. https://doi.org/10.1007/s11590-008-0094-5
14. C. X. Li, A modified generalized Newton method for absolute value equations, *J. Optim. Theory Appl.*, **170** (2016), 1055–1059. https://doi.org/10.1007/s10957-016-0956-4
15. 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. https://doi.org/10.1007/s11590-012-0560-y

16. A. Wang, Y. Cao, J. X. Chen, Modified Newton-type iteration methods for generalized absolute value equations, *J. Optim. Theory Appl.*, **181** (2019), 216–230. https://doi.org/10.1007/s10957-018-1439-6
17. C. X. Li, A generalization of the AOR iteration method for solving absolute value equations, *Electron. Res. Arch.*, **30** (2022), 1062–1074. https://doi.org/10.3934/era.2022056
18. H. Y. Zhou, S. L. Wu, C. X. Li, Newton-based matrix splitting method for generalized absolute value equation, *J. Comput. Appl. Math.*, **394** (2021), 113578. https://doi.org/10.1016/j.cam.2021.113578
19. R. Ali, I. Khan, A. Ali, A. Mohamed, Two new generalized iteration methods for solving absolute value equations using M-matrix, *AIMS Mathematics*, **7** (2022), 8176–8187. https://doi.org/10.3934/math.2022455
20. Y. F. Ke, The new iteration algorithm for absolute value equation, *Appl. Math. Lett.*, **99** (2020), 105990. https://doi.org/10.1016/j.aml.2019.07.021
21. C. X. Li, S. L. Wu, Block-diagonal and anti-block-diagonal splitting iteration method for absolute value equation, In: *Simulation tools and techniques*, Springer, Cham, **369** (2021), 572–581. https://doi.org/10.1007/978-3-030-72792-5_45
22. C. X. Li, L. Q. Yong, Modified BAS iteration method for absolute value equation, *AIMS Mathematics*, **7** (2021), 606–616. https://doi.org/10.3934/math.2022038
23. D. K. Salkuyeh, The Picard-HSS iteration method for absolute value equations, *Optim. Lett.*, **8** (2014), 2191–2202. https://doi.org/10.1007/s11590-014-0727-9
24. Z. Z. Bai, G. H. Golub, M. K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, *SIAM J. Matrix Anal. Appl.*, **24** (2003): 603–626. https://doi.org/10.1137/S0895479801395458
25. S. X. Miao, X. T. Xiong, J. Wen, On Picard-SHSS iteration method for absolute value equation, *AIMS Mathematics*, **6** (2021), 1743–1753. https://doi.org/10.3934/math.2021104
26. C. X. Li, S. L. Wu, A single-step HSS method for non-Hermitian positive definite linear systems, *Appl. Math. Lett.*, **44** (2015), 26–29. https://doi.org/10.1016/j.aml.2014.12.013
27. O. Axelsson, *Iterative solution methods*, Cambridge: Cambridge University Press, 1994. https://doi.org/10.1017/CBO9780511624100
28. Y. Saad, *Iterative methods for sparse linear systems*, New York: PWS Press, 2003. https://doi.org/10.1137/1.9780898718003
29. J. Y. Tang, J. C. Zhou, A quadratically convergent descent method for the absolute value equation $Ax + B|x| = b$, *Oper. Res. Lette.*, **47** (2019), 229–234. https://doi.org/10.1016/j.orl.2019.03.014
30. Z. Z. Bai, Modulus-based matrix splitting iteration methods for linear complementarity problems, *Numer. Linear Algebra Appl.*, **17** (2010), 917-933. https://doi.org/10.1002/nla.680

AIMS Press