AIMS Mathematics, 8(5): 10558–10578.
DOI: 10.3934/math.2023536
Received: 06 November 2022
Revised: 09 February 2023
Accepted: 17 February 2022
Published: 02 March 2023

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

Research article

New matrix splitting iteration method for generalized absolute value equations

Wan-Chen Zhao and Xin-Hui Shao*

Department of Mathematics, College of Sciences, Northeastern University, Shenyang 110819, China

* **Correspondence:** Email: xinhui1002@126.com.

Abstract: In this paper, a relaxed Newton-type matrix splitting (RNMS) iteration method is proposed for solving the generalized absolute value equations, which includes the Picard method, the modified Newton-type (MN) iteration method, the shift splitting modified Newton-type (SSMN) iteration method and the Newton-based matrix splitting (NMS) iteration method. We analyze the sufficient convergence conditions of the RNMS method. Lastly, the efficiency of the RNMS method is analyzed by numerical examples involving symmetric and non-symmetric matrices.

Keywords: generalized absolute value equation; relaxation; Newton-type matrix splitting method; convergence
Mathematics Subject Classification: 65F10, 90C05, 90C30

1. Introduction

We focus on the following generalized absolute value equations (GAVE):

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

where $A, B \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Here, the notation "$|\cdot|$" denotes the absolute value. Particularly, if $B = I$ or B is invertible, GAVE (1.1) is simplified to the absolute value equations (AVE):

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

Especially, when $B = 0$, the GAVE (1.1) is simplified to the linear system $Ax = b$, which plays a significant role in scientific computing problems.

The main significance of the GAVE (1.1) and the AVE (1.2) is that many problems in different fields may be transformed into the form of absolute value equations. Such as the linear programming problems, the linear complementarity problems (LCP), the quadratic programming, the mixed integer programming, the bimatrix game, and so on, see e.g. [1–6] for more details. For example, give a matrix $Q \in \mathbb{R}^{n\times n}$ and a vector $q \in \mathbb{R}^n$, the linear complementarity problems (LCP) is to find $z \in \mathbb{R}^n$ so that

$$z \geq 0, w \coloneqq Qz + q \geq 0, and\ z^T(Qz + q) = 0. \tag{1.3}$$

Lately, to obtain the numerical solutions of the GAVE (1.1) and the AVE (1.2), some numerical methods have been proposed, such as the Newton-type method [7–12], the sign accord method [13], the SOR-like iteration method [14–16], the neural network method [17,18] and so on.

The GAVE (1.1) can be transformed into the form of nonlinear equations, so some Newton-type methods are formed to obtain the solutions of the GAVE (1.1) and the AVE (1.2). In [9], using the generalized Jacobian matrix and based on the famous Newton iteration method, the generalized Newton (GN) method is directly established for solving the AVE (1.2). In [19], the application of GN method is extended to the GAVE (1.1). The Jacobian matrix of GN method changes with the iteration. When solving the large-scale problems, it is infeasible and wasteful, especially the Jacobian matrix is ill-conditioned. To overcome this shortcoming, a modified Newton-type (MN) method is proposed in [8]. To balance the MN method, SSMN methods are established in [20,21]. Then, a more common Newton-based matrix splitting (NMS) method is established in [11]. The matrix A is expressed as $A = M - N$. At every iteration, we need to calculate the linear system with coefficient matrix $\Omega + M$, where Ω is a given positive semi-definite matrix. However, if $\Omega + M$ is ill-conditioned, solving this linear system may be costly or impossible in practice. This motivates us to introduce a nonnegative real parameter $\theta \geq 0$ in the MN iteration frame based on matrix splitting and propose a new relaxed Newton-based matrix splitting (RNMS) method to solve the GAVE (1.1). If different splits are selected, the new RNMS method can be simplified into the above methods, and a series of new methods can be generated.

The layout of the rest is organized as follows. In Section 2, we establish a new relaxed method to solve the GAVE (1.1). In Section 3, the associated convergence analysis is given. The numerical results are reported in In Section 4, and some conclusions are given in final section.

2. The relaxed Newton-based matrix splitting iteration method

The existence of the nonlinear term $B|x|$ makes solving the GAVE (1.1) can be transformed into solving the nonlinear function $F(x)$

$$F(x) \coloneqq Ax - B|x| - b = 0. \tag{2.1}$$

Let $F(x) = H(x) + G(x)$, where $H(x)$ is a differentiable function and $G(x)$ is a Lipschitz continuous function, and Ω is a positive semi-definite matrix. Setting

$$H(x) = \Omega x + Ax,$$

and

$$G(x) = -\Omega x - B|x| - b,$$

we can substitute them into the MN iteration frame proposed in [22]. If the Jacobian matrix

$$H'(x) = \Omega + A$$

is invertible, the modified Newton-type (MN) iteration method for solving the GAVE (1.1) is established as follows:

Algorithm 2.1. [8] The modified Newton-type (MN) iteration method.
Step 1. Choose an arbitrary initial guess $x^{(0)} \in \mathbb{R}^n$ and the norm of relative residual vector as "RES", and let $k := 0$;
Step 2. For $k = 0,1,2,\cdots,$ computing $x^{(k+1)} \in \mathbb{R}^n$ by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - (\Omega + A)^{-1}\left(Ax^{(k)} - B\left|x^{(k)}\right| - b\right) \\ &= (\Omega + A)^{-1}\left(\Omega x^{(k)} + B\left|x^{(k)}\right| + b\right), \end{aligned} \tag{2.2}$$

where $\Omega + A$ is nonsingular and Ω is a known positive semi-definite matrix;
Step 3. If $\left|x^{(k+1)} - x^{(k)}\right| <$ RES, break. Otherwise, let $k + 1$ replace k and go to Step 2.

Based on the MN method, Zhou [11] proposed the Newton-based matrix splitting (NMS) iteration method to solve the GAVE (1.1) combining Newton method with matrix splitting technique.

Let

$$A = M - N,$$

and set

$$\bar{H}(x) = \Omega x + Mx,$$

$$\bar{G}(x) = -\Omega x - Nx - B|x| - b.$$

Zhou substituted them into the MN iteration frame proposed in [22] and put forward the Newton-based matrix splitting iteration method as follows:

Algorithm 2.2. [11] The Newton-based matrix splitting (NMS) iteration method.
Step 1. Choose an arbitrary initial guess $x^{(0)} \in \mathbb{R}^n$ and the norm of relative residual vector as "RES", and let $k := 0$;
Step 2. For $k = 0,1,2,\cdots,$ computing $x^{(k+1)} \in \mathbb{R}^n$ by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - (\Omega + M)^{-1}\left(Ax^{(k)} - B\left|x^{(k)}\right| - b\right) \\ &= (\Omega + M)^{-1}\left((\Omega + N)x^{(k)} + B\left|x^{(k)}\right| + b\right), \end{aligned} \tag{2.4}$$

where $\Omega + M$ is nonsingular and Ω is a known positive semi-definite matrix;
Step 3. If $\left|x^{(k+1)} - x^{(k)}\right| <$ RES, break. Otherwise, let $k + 1$ replace k and go to Step 2.

In this paper, we give a new method for solving the GAVE (1.1). Let $A = M - N$ and introduce a nonnegative real parameter $\theta \geq 0$ in the MN iteration frame proposed in [22], we can set

$$\tilde{H}(x) = \Omega x + \theta Mx,$$

and

$$\tilde{G}(x) = -\Omega x - (\theta - 1)M - Nx - B|x| - b.$$

Therefore, we can obtain the new RNMS method and describe below:

Algorithm 2.3. The relaxed Newton-type matrix splitting (RNMS) iteration method.
Step 1. Choose an arbitrary initial guess $x^{(0)} \in \mathbb{R}^n$ and the norm of relative residual vector as "RES", and let $k := 0$;
Step 2. For $k = 0,1,2,\cdots$, computing $x^{(k+1)} \in \mathbb{R}^n$ by

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - (\Omega + \theta M)^{-1}\left(Ax^{(k)} - B\left|x^{(k)}\right| - b\right) \\ &= (\Omega + \theta M)^{-1}\left((\Omega + (\theta - 1)M + N)x^{(k)} + B\left|x^{(k)}\right| + b\right), \end{aligned} \tag{2.4}$$

where $\Omega + \theta M$ is nonsingular, Ω is a known positive semi-definite matrix and $\theta \geq 0$ is a nonnegative relaxation parameter;
Step 3. If $\left|x^{(k+1)} - x^{(k)}\right| <$ RES, break. Otherwise, let $k + 1$ replace k and go to Step 2.

Remark 2.1. Obviously, if we put $\theta = 1$, then the RNMS method (2.4) is simplified to the NMS method [11]. Because the Picard method [22,23] and the MN method [8] are two special cases of the NMS method [11], they are also special cases of the new RNMS method (2.4).

Algorithm 2.3 gives a more common framework of the relaxed Newton-type matrix splitting iteration method for solving the GAVE (1.1). If the matrix A is split into $D - L - U$, where D is the diagonal part of A, L and U are the strictly lower and upper triangle parts of A respectively, the following series of relaxed Newton-type matrix splitting forms will be generated:

(a) Let $M = A$, $N = \Omega = 0$ and $\theta = 1$, Algorithm 2.3 is simplified to the Picard method [22,23]

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

(b) Let $M = A$, $N = 0$ and $\theta = 1$, Algorithm 2.3 is simplified to the MN method [8]

$$x^{(k+1)} = (\Omega + A)^{-1}\left(\Omega x^{(k)} + B\left|x^{(k)}\right| + b\right).$$

(c) Let $M = D$ and $N = L + U$, Algorithm 2.3 gives the relaxed Newton-based Jacobi (RNJ) method

$$x^{(k+1)} = (\Omega + \theta D)^{-1}\left((\Omega + (\theta - 1)D + L + U)x^{(k)} + B\left|x^{(k)}\right| + b\right).$$

(d) Let $M = D - L$ and $N = U$, Algorithm 2.3 gives the relaxed Newton-based Gauss-Seidel (RNGS) method

$$x^{(k+1)} = (\Omega + \theta D - \theta L)^{-1}\left((\Omega + (\theta - 1)(D - L) + U)x^{(k)} + B\left|x^{(k)}\right| + b\right).$$

(e) Let $M = \frac{1}{\alpha}D - L$ and $N_P = \left(\frac{1}{\alpha} - 1\right)D + U$, Algorithm 2.3 gives the relaxed Newton-type SOR (RNSOR) method

$$x^{(k+1)} = (\alpha\Omega + \theta D - \theta\alpha L)^{-1}\left((\alpha\Omega + (\theta - \alpha)D - \alpha(\theta - 1)L + \alpha U)x^{(k)} + \alpha B\left|x^{(k)}\right| + \alpha b\right).$$

(f) Let $M = \frac{1}{\alpha}(D - \beta L)$ and $N = \frac{1}{\alpha}((1 - \alpha)D + (\alpha - \beta)L + \alpha U)$, Algorithm 2.3 gives the relaxed Newton-type AOR (RNAOR) method

$$x^{(k+1)} = (\alpha\Omega + \theta D - \theta\beta L)^{-1}\left((\alpha\Omega + (\theta - \alpha)D - (\theta\beta - \alpha)L + \alpha U)x^{(k)} + \alpha B\left|x^{(k)}\right| + \alpha b\right).$$

(g) Let $M = H$ and $N = -S$, where $H = \frac{1}{2}(A + A^T)$ and $S = \frac{1}{2}(A - A^T)$, Algorithm 2.3 gives the relaxed Newton-type Hermitian and Skew-Hermitian (RNHSS) method

$$x^{(k+1)} = (\Omega + \theta H)^{-1}\left((\Omega + (\theta - 1)H - S)x^{(k)} + B\left|x^{(k)}\right| + b\right).$$

Remark 2.2. Let $M = \frac{1}{2}(A + \Omega)$, $N = -\frac{1}{2}(A - \Omega)$, we can obtain the relaxed SSMN (RSSMN) method

$$x^{(k+1)} = (\theta\Omega + \theta A)^{-1}\left((\theta\Omega + (\theta - 2)A)x^{(k)} + 2B\left|x^{(k)}\right| + 2b\right). \tag{2.5}$$

If $\theta = 1$, the RSSMN method is simplified to the SSMN method [20,21]

$$x^{(k+1)} = (\Omega + A)^{-1}\left((\Omega - A)x^{(k)} + 2B\left|x^{(k)}\right| + 2b\right). \tag{2.6}$$

3. Convergence property

We will discuss the convergence analysis of the RNMS method for solving the GAVE (1.1) in this section. Firstly, according to the matrix 2-norm, two convergence conditions in common cases are given. Secondly, if A is positive definite or H_+-matrix, then some special convergence theorems of the RNMS method are provided. The RNMS method can be simplified to the NMS method, so its convergence conditions can be obtained immediately.

Before that, briefly introduce the following symbols and definitions. Let $A = (a_{ij}) \in \mathbb{R}^{n\times n}$. If $a_{ij} \le 0$ for any $i \ne j$, then A is a Z-matrix. Let 0 represent a zero matrix. If $A^{-1} \ge 0$ and A is a Z-matrix, then A is a nonsingular M-matrix. If the comparison matrix $\langle A\rangle$ of A is an M-matrix, then A is an H-matrix, where the form of $\langle A\rangle = (\langle a\rangle_{ij})$ is as follows:

$$\langle a\rangle_{ij} = \begin{cases} |a_{ii}|, & \text{if } i = j, \\ -|a_{ii}|, & \text{if } i \ne j. \end{cases}$$

If A is an H-matrix with positive diagonal terms, then A is an H_+-matrix. If A is symmetric and satisfies $x^T Ax > 0$ for all nonzero vectors x, then A is symmetric positive definite. We define $|A| = (|a_{ij}|)$ and $\rho(A)$ represent the absolute value matrix and the spectral radius, respectively.

3.1 General sufficient convergence property

Theorems 3.1 and 3.2 present the general sufficient convergence of Algorithm 2.3 when the correlation matrix is invertible.

Theorem 3.1. Let $A = M - N$ be its splitting, $A, B \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta \ge 0$ be a nonnegative relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta M$ is invertible.

If

$$\|(\Omega + \theta M)^{-1}\|_2 < \frac{1}{\|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2}, \tag{3.1}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Proof. Suppose that the GAVE (1.1) has a solution x^*, then x^* satisfies the following equation:

$$Ax^* - B|x^*| = b, \tag{3.2}$$

which is equal to

$$(\Omega + \theta M)x^* = \Omega x^* + (\theta - 1)Mx^* + Nx^* + B|x^*| + b. \tag{3.3}$$

Subtracting (3.3) from (2.4) gives the error expression as follows:

$$(\Omega + \theta M)(x^{(k+1)} - x^*) = (\Omega + (\theta - 1)M + N)(x^{(k)} - x^*) + B(\left|x^{(k)}\right| - |x^*|). \tag{3.4}$$

Noticing that $\Omega + \theta M$ is invertible, we have

$$x^{(k+1)} - x^* = (\Omega + \theta M)^{-1}((\Omega + (\theta - 1)M + N)(x^{(k)} - x^*) + B(\left|x^{(k)}\right| - |x^*|). \tag{3.5}$$

Using the 2-norm for (3.5), we get

$$\begin{aligned}\left\|x^{(k+1)} - x^*\right\|_2 &= \left\|(\Omega + \theta M)^{-1}((\Omega + (\theta - 1)M + N)(x^{(k)} - x^*) + B(\left|x^{(k)}\right| - |x^*|))\right\|_2 \\ &\leq \|(\Omega + \theta M)^{-1}\|_2 \cdot \left\|(\Omega + (\theta - 1)M + N)(x^{(k)} - x^*) + B(\left|x^{(k)}\right| - |x^*|)\right\|_2 \\ &\leq \|(\Omega + \theta M)^{-1}\|_2 \cdot (\|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2)\left\|x^{(k)} - x^*\right\|_2. \end{aligned} \tag{3.6}$$

According to the condition (3.1), the $\left\{x^{(k)}\right\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Theorem 3.2. Let $A = M - N$ be nonsingular, where M is also nonsingular, $A, B \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta > 0$ be a positive relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta M$ is nonsingular.

If

$$\|(\theta M)^{-1}\|_2 < \frac{1}{\|\Omega\|_2 + \|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2}, \tag{3.7}$$

then the iterative sequence $\left\{x^{(k)}\right\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Proof. From the Banach Lemma [25], we can have

$$\begin{aligned}\|(\Omega + \theta M)^{-1}\|_2 \leq \frac{\|(\theta M)^{-1}\|_2}{1 - \|(\theta M)^{-1}\|_2\|\Omega\|_2} &< \frac{\dfrac{1}{\|\Omega\|_2 + \|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2}}{1 - \dfrac{\|\Omega\|_2}{\|\Omega\|_2 + \|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2}} \\ &= \frac{1}{\|\Omega + (\theta - 1)M + N\|_2 + \|B\|_2}. \end{aligned} \tag{3.8}$$

Then the conclusion is drawn from Theorem 3.1.

Assuming $B = I$, the GAVE (1.1) can be simplified to the AVE (1.2). Therefore, the RNMS method is also suitable for solving the AVE (1.2).

Corollary 3.1. Let $A = M - N$ be its splitting, $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta \geq 0$ be a nonnegative relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta M$ is invertible.

If

$$\|(\Omega + \theta M)^{-1}\|_2 < \frac{1}{\|\Omega + (\theta - 1)M + N\|_2 + 1}, \tag{3.9}$$

then the iterative sequence $\left\{x^{(k)}\right\}_{k=1}^{+\infty}$ created by Algorithm 2.3 to solve the AVE (1.2) is convergent.

Corollary 3.2. Let $A = M - N$ be nonsingular, where M is also nonsingular, $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta > 0$ be a positive relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta M$ is nonsingular.

If

$$\|(\theta M)^{-1}\|_2 < \frac{1}{\|\Omega\|_2 + \|\Omega + (\theta-1)M + N\|_2 + 1}, \tag{3.10}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 to solve the AVE (1.2) is convergent.

Based on Remark 2.2, we can draw the following corollaries.

Corollary 3.3. Let $A, B \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta \geq 0$ be a nonnegative relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta A$ is invertible.

If

$$\|(\theta\Omega + \theta A)^{-1}\|_2 < \frac{1}{\|\theta\Omega + (\theta-2)A\|_2 + 2\|B\|_2}, \tag{3.11}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by RSSMN method is convergent.

Corollary 3.4. Let $A \in \mathbb{R}^{n\times n}$ be nonsingular, $B \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, $\theta > 0$ be a positive relaxation parameter and Ω be a positive semi-definite matrix which makes $\Omega + \theta A$ is nonsingular.

If

$$\|(\theta A)^{-1}\|_2 < \frac{1}{\|\theta\Omega\|_2 + \|\theta\Omega + (\theta-2)A\|_2 + 2\|B\|_2}, \tag{3.12}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by RSSMN method is convergent.

Remark 3.1. If we put $\theta = 1$, the RSSMN method is simplified to the SSMN method [21,22]. Therefore, the convergence conditions of SSMN method in [21,22] can be obtained from Corollaries 3.3 and 3.4.

3.2 Special sufficient convergence property

If A is positive definite or H_+-matrix, we can get Theorem 3.3–3.5, for $\Omega = \omega I$ with $\omega > 0$, respectively.

Theorem 3.3. Let $A = H + S$ be a positive definite matrix, where $H = \frac{1}{2}(A + A^T)$ and $S = \frac{1}{2}(A - A^T)$, $\theta > 0$ be a positive relaxation parameter and $\Omega = \omega I$ with $\omega > 0$.

Further denote that λ_{min} and λ_{max} are the minimum and the maximum eigenvalues of the matrix H respectively, σ_{max} is the maximum value of the absolute values of the eigenvalues of the matrix S and assume $\|B\|_2 = \tau$. If

$$\omega + \theta\lambda_{min} - \tau > \sqrt{\omega^2 + (\theta-1)^2{\lambda_{max}}^2 + {\sigma_{max}}^2}, \tag{3.13}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Proof. In fact, according to Theorem 3.1, we only need to get

$$\|(\Omega + \theta M)^{-1}\|_2(\|\Omega + (\theta-1)M + N\|_2 + \|B\|_2) < 1. \tag{3.14}$$

By assumptions, we have

$$\|(\Omega+\theta H)^{-1}\|_2(\|\Omega+(\theta-1)H-S\|_2+\|B\|_2)$$

$$=\max_{\lambda\in\mathrm{sp}(H),\sigma\in\mathrm{sp}(S)}\frac{|\omega+(\theta-1)\lambda-\sigma|+\tau}{\omega+\theta\lambda}\leq\frac{\sqrt{\omega^2+(\theta-1)^2{\lambda_{max}}^2+{\sigma_{max}}^2}+\tau}{\omega+\theta\lambda_{min}}. \tag{3.15}$$

It follows, if

$$\omega+\theta\lambda_{min}-\tau>\sqrt{\omega^2+(\theta-1)^2{\lambda_{max}}^2+{\sigma_{max}}^2}, \tag{3.16}$$

then the $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Assuming $B=I$, a similar corollary can be obtained.

Corollary 3.5. Let $A=H+S$ be a positive definite matrix, where $H=\frac{1}{2}(A+A^T)$ and $S=\frac{1}{2}(A-A^T)$, $\theta>0$ be a positive relaxation parameter and $\Omega=\omega I$ with $\omega>0$.

Further denote that λ_{min} and λ_{max} are the minimum and the maximum eigenvalues of the matrix H respectively, and σ_{max} is the maximum value of the absolute values of the eigenvalues of the matrix S. If

$$\omega+\theta\lambda_{min}-1>\sqrt{\omega^2+(\theta-1)^2{\lambda_{max}}^2+{\sigma_{max}}^2}, \tag{3.17}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 to solve the AVE (1.2) is convergent.

Moreover, let $A=M-N$, where M is symmetric positive definite, we can get Theorem 3.4.

Theorem 3.4. Let $A=M-N$ be its splitting, where M is symmetric positive definite, $A,B\in\mathbb{R}^{n\times n}$, $b\in\mathbb{R}^n$, $\theta>0$ be a positive relaxation parameter and $\Omega=\omega I$ with $\omega>0$.

Further denote that λ_{min} and λ_{max} are the minimum and the maximum eigenvalues of the matrix M, respectively, and assume$\|M^{-1}N\|_2=\sigma$, $\|B\|_2=\tau$. If

$$\lambda_{max}|\theta-1+\sigma|<\theta\lambda_{min}-\tau, \tag{3.18}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Proof. According to (3.6), we only need to get

$$\|(\Omega+\theta M)^{-1}(\Omega+(\theta-1)M+N+B)\|_2<1. \tag{3.19}$$

It can be transformed into proving the following equation:

$$\|(\Omega+\theta M)^{-1}\Omega\|_2+\|(\Omega+\theta M)^{-1}((\theta-1)M+N)\|_2+\|(\Omega+\theta M)^{-1}B\|_2<1. \tag{3.20}$$

Since

$$\|(\Omega+\theta M)^{-1}\Omega\|_2+\|(\Omega+\theta M)^{-1}B\|_2\leq\frac{\omega+\tau}{\omega+\theta\lambda_{min}}, \tag{3.21}$$

and

$$\begin{aligned}\|(\Omega+\theta M)^{-1}((\theta-1)M+N)\|_2 &\le \|(\Omega+\theta M)^{-1}M\|_2\|M^{-1}((\theta-1)M+N)\|_2\\ &= \|(\Omega+\theta M)^{-1}M\|_2\|\theta-1+M^{-1}N\|_2\\ &= \max_{\lambda\in \mathrm{sp}(M)}\frac{\lambda|\theta-1+\sigma|}{\omega+\theta\lambda} \le \frac{\lambda_{max}|\theta-1+\sigma|}{\omega+\theta\lambda_{min}},\end{aligned} \tag{3.22}$$

we just need

$$\frac{\lambda_{max}|\theta-1+\sigma|}{\omega+\theta\lambda_{min}}+\frac{\omega+\tau}{\omega+\theta\lambda_{min}}<1. \tag{3.23}$$

This implies under the condition (3.18), the $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 is convergent.

Theorem 3.5. Let $A\in\mathbb{R}^{n\times n}$ be an H_+-matrix, $A=M-N$ be H −splitting, $\theta>0$ be a positive relaxation parameter and $\Omega=\omega I$ with $\omega>0$. If

$$\|(\Omega+\langle\theta M\rangle)^{-1}\|_2<\frac{1}{\|\Omega+|\theta-1||M|+|N|+|B|\|_2}, \tag{3.24}$$

then the iterative sequence $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 converges.

Proof. By the assumptions and [26], we have

$$|(\Omega+\theta M)^{-1}|\le(\Omega+\langle\theta M\rangle)^{-1}. \tag{3.25}$$

Using the absolute value for (3.5), we get

$$\begin{aligned}\left|x^{(k+1)}-x^*\right| &= \left|(\Omega+\theta M)^{-1}\left((\Omega+(\theta-1)M+N)\left(x^{(k)}-x^*\right)+B\left(\left|x^{(k)}\right|-|x^*|\right)\right)\right|\\ &\le |(\Omega+\theta M)^{-1}|(|\Omega+(\theta-1)M+N|\left|x^{(k)}-x^*\right|+|B|\left|\left|x^{(k)}\right|-|x^*|\right|)\\ &\le (\Omega+\langle\theta M\rangle)^{-1}(\Omega+|\theta-1||M|+|N|+|B|)\left|x^{(k)}-x^*\right|.\end{aligned} \tag{3.26}$$

Since

$$\begin{aligned}&\rho((\Omega+\langle\theta M\rangle)^{-1}(\Omega+|\theta-1||M|+|N|+|B|)\\ &\quad\le \|(\Omega+\langle\theta M\rangle)^{-1}(\Omega+|\theta-1||M|+|N|+|B|)\|_2\\ &\quad\le \|(\Omega+\langle\theta M\rangle)^{-1}\|_2\|\Omega+|\theta-1||M|+|N|+|B|\|_2.\end{aligned} \tag{3.27}$$

It follows that, if the condition (3.24) is satisfied, then the $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 converges.

Corollary 3.6. Let $A\in\mathbb{R}^{n\times n}$ be an H_+-matrix, $A=M-N$ be H −splitting, $\theta>0$ be a positive relaxation parameter and $\Omega=\omega I$ with $\omega>0$. If

$$\|(\Omega+\langle\theta M\rangle)^{-1}\|_2<\frac{1}{\|\Omega+|\theta-1||M|+|N|+1\|_2}, \tag{3.28}$$

then the $\{x^{(k)}\}_{k=1}^{+\infty}$ created by Algorithm 2.3 to solve the AVE (1.2) converges.

4. Numerical experiments

This section provides two numerical examples to compare the Picard method [23,24], the MN

method [8], the NMS method [11] and the RNMS method in terms of the iteration step number (indicated as "IT"), the amount of CPU time (indicated as "CPU") and the norm of relative residual vector (indicated as "RES"). Here, "RES" was set to be

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

Here, we use MATLAB R2020B for all the experiments. All numerical computations are started from the initial vector

$$x^{(0)} = (1,0,1,0,\dots,1,0,\dots)^T \in \mathbb{R}^n.$$

The iteration is terminated once RES$< 10^{-6}$ or the largest number of iteration step k_{max} exceeds 500. In the following table, "−" denotes k_{max} is larger than 500 or the CPU times are larger than 500 s.

The article [3,9] show that if the eigenvalue of Q is not 1, the LCP (1.3) can lead to

$$(I+Q)x + (I-Q)|x| = q \; with \; x = \frac{1}{2}[(M-I)z + q]. \tag{4.1}$$

If we let $A = I + Q,\ B = Q - I,\ b = q$, (4.1) converts to the form of the GAVE (1.1). According to this, we give the following examples.

Example 4.1. [2] Let $n = m^2$. We consider the LCP (1.3), where $Q = \hat{Q} + \mu I \in \mathbb{R}^{n\times n}$ and $q = -Qz^* \in \mathbb{R}^n$ with

$$\hat{Q} = tridiag(-I, S, -I) \in \mathbb{R}^{n\times n}, S = tridiag(-1,4,-1) \in \mathbb{R}^{m\times m},$$

and

$$z^* = (1.2,1.2,1.2,\dots,1.2,\dots)^T \in \mathbb{R}^n$$

is the unique solution of the LCP. In this situation, the unique solution of the GAVE (1.1) is

$$x^* = (-0.6,-0.6,-0.6,\dots,-0.6,\dots)^T \in \mathbb{R}^n.$$

In [11], the NJ, NGS and NSOR methods are the representatives of the proposed NMS method. Compared with the Picard method and the MN method, it is concluded that the calculation efficiency of NMS method is higher in some cases. In our actual experiments, we can take $\Omega = \hat{Q}, 0.5\hat{Q}$ and $\mu = 4,-1$. Different α will affect the performance of the NSOR method, so the best experimental parameter is recorded as α_{exp} which minimizes the iteration step number of the NSOR method.

To demonstrate the superiority of the new RNMS method, we can take $\Omega = \theta\hat{Q}, 0.5\theta\hat{Q}$ and $\mu = 4,-1$ in our actual experiments. Obviously, the NMS method is a special case of $\theta = 1$ in the new RNMS method. By computation, matrices Q and A, when $\mu = 4$ are symmetric positive definite. Therefore, Cholesky decomposition can be used to assist in inversion. When $\mu = -1$, the former is symmetric indefinite, and the latter is symmetric positive definite. Therefore, LU decomposition can be used to assist the work of inversion.

Here, the RNJ, RNGS, RNSOR methods are used as representatives of the new RNMS method. The efficiency of RNSOR method is affected by different factors α and θ, so the best experimental parameters are recorded as α_{exp} and θ_{exp}, which minimizes the number of iterative steps of RNSOR method. Similarly, different Ω will affect the performance of the MN method, and the best experimental parameter is recorded as θ_{exp}. When the iteration step numbers are the same, take the minimum value of RES. The Tables 1–4 list the results of numerical tests (including IT, CPU and

RES) for the eight test methods, i.e., the Picard method, the MN method, the NJ method, the RNJ method, the NGS method, the RNGS method, the NSOR method and the RNSOR method.

Table 1. Experimental results for Example 4.1 for $\mu = 4$ with $\Omega = \theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	64	63	63	63	63	62
	CPU	0.0193	0.0252	0.0355	0.0495	0.0665	0.0753
	RES	8.9163e^{-07}	9.9983e^{-07}	9.4233e^{-07}	8.9429e^{-07}	8.5340e^{-07}	9.8482e^{-07}
	θ_{exp}	4.96	4.97	4.98	4.99	5.00	5.00
MN	IT	23	23	23	23	23	23
	CPU	0.0087	0.0121	0.0165	0.0219	0.0322	0.0384
	RES	7.4274e^{-07}	8.2275e^{-07}	8.8325e^{-07}	9.3056e^{-07}	9.6857e^{-07}	9.9977e^{-07}
NJ	IT	12	12	12	12	12	12
	CPU	0.0068	0.0092	0.0137	0.0171	0.0233	0.0254
	RES	8.5417e^{-07}	7.9553e^{-07}	7.4759e^{-07}	7.0745e^{-07}	6.7322e^{-07}	6.4359e^{-07}
	θ_{exp}	1.07	1.07	1.07	1.07	1.07	1.07
RNJ	IT	10	10	10	10	10	10
	CPU	0.0046	0.0078	0.0101	0.0129	0.0182	0.0208
	RES	2.6400e^{-07}	2.5922e^{-07}	2.5550e^{-07}	2.5252e^{-07}	2.5009e^{-07}	2.4806e^{-07}
NGS	IT	11	11	11	11	11	11
	CPU	0.0134	0.0175	0.0202	0.0275	0.0429	0.0469
	RES	3.5695e^{-07}	3.4869e^{-07}	3.4224e^{-07}	3.3705e^{-07}	3.3279e^{-07}	3.2923e^{-07}
	θ_{exp}	1.19	1.19	1.19	1.19	1.19	1.19
RNGS	IT	7	7	7	7	7	7
	CPU	0.0089	0.0119	0.0155	0.0200	0.0303	0.0338
	RES	2.4270e^{-07}	2.1969e^{-07}	2.0193e^{-07}	1.8772e^{-07}	1.7607e^{-07}	1.6629e^{-07}
	α_{exp}	0.83	0.83	0.83	0.83	0.84	0.84
NSOR	IT	8	8	8	8	8	8
	CPU	0.0127	0.0175	0.0193	0.0271	0.0382	0.0442
	RES	2.3870e^{-07}	2.3222e^{-07}	2.2716e^{-07}	2.2310e^{-07}	2.1932e^{-07}	2.1397e^{-07}
	α_{exp}	1.09	1.09	1.09	1.09	1.09	1.09
	θ_{exp}	1.28	1.28	1.28	1.28	1.28	1.28
RNSOR	IT	7	7	7	7	7	7
	CPU	0.0088	0.0122	0.0156	0.0204	0.0299	0.0335
	RES	2.2178e^{-07}	1.9959e^{-07}	1.8254e^{-07}	1.6896e^{-07}	1.5785e^{-07}	1.4857e^{-07}

Table 2. Experimental results for Example 4.1 for $\mu = 4$ with $\Omega = 0.5\theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	64	63	63	63	63	62
	CPU	0.0193	0.0252	0.0355	0.0495	0.0665	0.0753
	RES	8.9163e^{-07}	9.9983e^{-07}	9.4233e^{-07}	8.9429e^{-07}	8.5340e^{-07}	9.8482e^{-07}
	θ_{exp}	9.92	9.95	9.97	9.98	9.99	10.00
MN	IT	23	23	23	23	23	23
	CPU	0.0089	0.0143	0.0193	0.0224	0.0330	0.0446
	RES	7.4274e^{-07}	8.2275e^{-07}	8.8325e^{-07}	9.3056e^{-07}	9.6857e^{-07}	9.9977e^{-07}
NJ	IT	56	55	55	55	55	54
	CPU	0.0133	0.0269	0.0308	0.0388	0.0566	0.0657
	RES	8.7889e^{-07}	9.8614e^{-07}	9.3009e^{-07}	8.8251e^{-07}	8.4147e^{-07}	9.6551e^{-07}
	θ_{exp}	1.27	1.27	1.26	1.26	1.26	1.26
RNJ	IT	13	13	13	13	13	13
	CPU	0.0051	0.0087	0.0114	0.0149	0.0206	0.0236
	RES	8.8398e^{-07}	8.8020e^{-07}	8.7347e^{-07}	8.5566e^{-07}	8.4093e^{-07}	8.2853e^{-07}
NGS	IT	24	24	24	24	23	23
	CPU	0.0207	0.0252	0.0291	0.0386	0.0531	0.0613
	RES	7.2595e^{-07}	6.7811e^{-07}	6.3854e^{-07}	6.0513e^{-07}	9.7001e^{-07}	9.2790e^{-07}
	θ_{exp}	1.32	1.32	1.32	1.31	1.31	1.31
RNGS	IT	9	9	9	9	9	9
	CPU	0.0102	0.0134	0.0169	0.0222	0.0337	0.0356
	RES	2.7806e^{-07}	2.7047e^{-07}	2.6453e^{-07}	2.5912e^{-07}	2.5126e^{-07}	2.4456e^{-07}
	α_{exp}	0.75	0.75	0.75	0.75	0.75	0.75
NSOR	IT	11	11	11	11	11	11
	CPU	0.0129	0.0161	0.0212	0.0306	0.0417	0.0529
	RES	2.8494e^{-07}	2.7336e^{-07}	2.6410e^{-07}	2.5653e^{-07}	2.5020e^{-07}	2.4484e^{-07}
	α_{exp}	1.24	1.25	1.25	1.25	1.26	1.26
	θ_{exp}	1.64	1.65	1.65	1.65	1.66	1.66
RNSOR	IT	8	8	8	8	8	8
	CPU	0.0105	0.0132	0.0179	0.0219	0.0347	0.0367
	RES	2.4635e^{-07}	2.1966e^{-07}	1.9916e^{-07}	1.8300e^{-07}	1.6988e^{-07}	1.5887e^{-07}

Table 3. Experimental results for Example 4.1 for $\mu = -1$ with $\Omega = \theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
	θ_{exp}	-	-	-	-	-	-
MN	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
NJ	IT	51	51	50	50	50	50
	CPU	0.0116	0.0220	0.0275	0.0375	0.0553	0.0801
	RES	8.7996e^{-07}	8.2701e^{-07}	9.7990e^{-07}	9.3796e^{-07}	9.0284e^{-07}	8.7295e^{-07}
	θ_{exp}	0.58	0.58	0.58	0.58	0.58	0.58
RNJ	IT	28	27	27	27	27	27
	CPU	0.0070	0.0131	0.0170	0.0217	0.0317	0.0370
	RES	6.9967e^{-07}	9.4400e^{-07}	8.7261e^{-07}	8.1572e^{-07}	7.6912e^{-07}	7.3012e^{-07}
NGS	IT	59	58	58	57	57	57
	CPU	0.0330	0.0402	0.0535	0.0721	0.1150	0.1159
	RES	8.5017e^{-07}	9.3343e^{-07}	8.7524e^{-07}	9.7771e^{-07}	9.2895e^{-07}	8.8685e^{-07}
	θ_{exp}	0.59	0.59	0.59	0.59	0.60	0.60
RNGS	IT	33	33	33	33	32	32
	CPU	0.0192	0.0258	0.0355	0.0458	0.0649	0.0768
	RES	7.7770e^{-07}	7.5546e^{-07}	7.5440e^{-07}	7.6188e^{-07}	9.9670e^{-07}	9.5728e^{-07}
	α_{exp}	1.34	1.33	1.32	1.31	1.30	1.29
NSOR	IT	54	53	53	53	53	52
	CPU	0.0297	0.0379	0.0529	0.0668	0.1018	0.1104
	RES	8.4350e^{-07}	9.6545e^{-07}	9.2671e^{-07}	8.9233e^{-07}	8.5697e^{-07}	9.9468e^{-07}
	α_{exp}	0.55	0.55	0.55	0.55	0.53	0.53
	θ_{exp}	0.40	0.40	0.40	0.40	0.39	0.39
RNSOR	IT	27	27	27	27	27	26
	CPU	0.0171	0.0225	0.0312	0.0388	0.0546	0.0646
	RES	9.3375e^{-07}	8.5594e^{-07}	7.9495e^{-07}	7.4552e^{-07}	6.9876e^{-07}	9.6657e^{-07}

Table 4. Experimental results for Example 4.1 for $\mu = -1$ with $\Omega = 0.5\theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
	θ_{exp}	-	-	-	-	-	-
MN	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
NJ	IT	35	35	35	35	35	34
	CPU	0.0089	0.0211	0.0232	0.0324	0.0449	0.0534
	RES	7.4497e^{-07}	7.2345e^{-07}	7.0672e^{-07}	6.9335e^{-07}	6.8240e^{-07}	9.8833e^{-07}
	θ_{exp}	0.83	0.83	0.83	0.83	0.83	0.82
RNJ	IT	28	28	28	28	28	27
	CPU	0.0071	0.0134	0.0175	0.0223	0.0326	0.0364
	RES	8.3291e^{-07}	7.7166e^{-07}	7.2727e^{-07}	6.9379e^{-07}	6.6774e^{-07}	9.9478e^{-07}
NGS	IT	41	41	41	41	40	40
	CPU	0.0241	0.0331	0.0449	0.0681	0.0835	0.0994
	RES	9.6413e^{-07}	8.9512e^{-07}	8.3906e^{-07}	7.9236e^{-07}	9.5607e^{-07}	9.1264e^{-07}
	θ_{exp}	0.74	0.74	0.74	0.73	0.73	0.73
RNGS	IT	29	29	29	29	29	28
	CPU	0.0182	0.0248	0.0322	0.0432	0.0637	0.0734
	RES	9.7972e^{-07}	9.0531e^{-07}	8.4562e^{-07}	7.8848e^{-07}	7.3750e^{-07}	9.6299e^{-07}
	α_{exp}	1.31	1.29	1.28	1.27	1.26	1.26
NSOR	IT	37	37	37	36	36	36
	CPU	0.0242	0.0306	0.0464	0.0563	0.0725	0.0849
	RES	8.0711e^{-07}	7.9160e^{-07}	7.7221e^{-07}	9.8918e^{-07}	9.5431e^{-07}	9.1519e^{-07}
	α_{exp}	0.71	0.71	0.71	0.71	0.71	0.71
	θ_{exp}	0.59	0.59	0.59	0.59	0.59	0.59
RNSOR	IT	28	27	27	27	27	27
	CPU	0.0183	0.0229	0.0296	0.0393	0.0559	0.0646
	RES	6.9923e^{-07}	9.2059e^{-07}	8.5325e^{-07}	7.9887e^{-07}	7.5381e^{-07}	7.1571e^{-07}

From the experimental data in Tables 1–4, it is easy to see that when the grid size n increases, the iteration step number and CPU time of the eight methods also increase. We can find that the RNSOR method has the least iteration step number and CPU time, and the Picard method has the most. The RNJ method, the RNGS method and the RNSOR method cost less than the NJ method, the NGS method and the NSOR method, respectively.

Example 4.2. [2,27] Let $n = m^2$. We consider the LCP (1.3), where $Q = \hat{Q} + \mu I \in \mathbb{R}^{n\times n}$ and $q = -Qz^* \in \mathbb{R}^n$ with

$$\hat{Q} = tridiag(-1.5I, S, -0.5I) \in \mathbb{R}^{n\times n}, S = tridiag(-1.5,4,-0.5) \in \mathbb{R}^{m\times m},$$

and

$$z^* = (1.2,1.2,1.2, \dots ,1.2, \dots)^T \in \mathbb{R}^n$$

is the unique solution of the LCP. In this situation, the unique solution of the GAVE (1.1) is

$$x^* = (-0.6, -0.6, -0.6, \dots , -0.6, \dots)^T \in \mathbb{R}^n$$

For comparison, we can take

$$\Omega = \theta\hat{Q}, 0.5\theta\hat{Q}$$

and

$$\mu = 4, -1$$

in our actual experiments. Therefore, Ω in the NMS method is equal to $\hat{Q}, 0.5\hat{Q}$.

For Example 4.2, we still compare the above eight methods: the Picard method, the MN method, the NJ method, the RNJ method, the NGS method, the RNGS method, the NSOR method and the RNSOR method.

By computation, matrix A, when $\mu = 4$ is a strictly diagonally dominant H_+-matrix, and when $\mu = -1$ is an irreducible and weakly diagonally dominant H_+-matrix. In the implementation operation, to assist the inversion, sparse Cholesky decomposition or sparse LU decomposition can be used.

From Tables 5 and 6, we can get the same conclusion as Tables 1 and 2. The RNSOR method has the least iteration step number and CPU time than other test methods. The numerical results in Tables 7 and 8 show that the Picard method and the MN method are not convergent, the NJ, NGS, NSOR methods are convergent, but there are many iterative steps.

The number of iteration steps of our new method is much less than that of the previous method. From these experimental data, we can again conclude that the new relaxed Newton-type matrix splitting (RNMS) method is superior to the Picard method, the MN method, and the NMS method for solving the GAVE (1.1).

Table 5. Experimental results for Example 4.2 for $\mu = 4$ with $\Omega = \theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	65	65	65	64	64	64
	CPU	0.0372	0.0465	0.0665	0.0902	0.1361	0.1554
	RES	9.8655e-07	9.2007e-07	8.6567e-07	9.7913e-07	9.3294e-07	8.9296e-07
	θ_{exp}	4.47	4.71	4.86	4.96	5.01	5.05
MN	IT	20	20	21	22	22	22
	CPU	0.0164	0.0243	0.0367	0.0470	0.0588	0.0749
	RES	4.5871e-07	9.1674e-07	7.3234e-07	5.3617e-07	6.8625e-07	8.1379e-07
NJ	IT	13	13	13	13	13	12
	CPU	0.0131	0.0188	0.0236	0.0290	0.0460	0.0487
	RES	5.1537e-07	4.7971e-07	4.5055e-07	4.2614e-07	4.0530e-07	9.7420e-07
	θ_{exp}	1.08	1.08	1.08	1.08	1.08	1.08
RNJ	IT	10	10	10	10	10	10
	CPU	0.0110	0.0137	0.0174	0.0220	0.0338	0.0383
	RES	3.4130e-07	3.3734e-07	3.3430e-07	3.3190e-07	3.2994e-07	3.2832e-07
NGS	IT	13	13	13	13	13	13
	CPU	0.0157	0.0213	0.0251	0.0350	0.0444	0.0552
	RES	3.8144e-07	3.8683e-07	3.9089e-07	3.9404e-07	3.9657e-07	3.9863e-07
	θ_{exp}	1.26	1.26	1.26	1.26	1.26	1.26
RNGS	IT	7	7	7	7	7	7
	CPU	0.0096	0.0121	0.0166	0.0240	0.0317	0.0348
	RES	1.9114e-07	1.7589e-07	1.6405e-07	1.5454e-07	1.4670e-07	1.4012e-07
	α_{exp}	0.79	0.79	0.79	0.79	0.79	0.79
NSOR	IT	7	7	7	7	7	7
	CPU	0.0131	0.0172	0.0200	0.0287	0.0406	0.0464
	RES	6.2609e-08	6.1037e-08	5.9806e-08	5.8817e-08	5.8004e-08	5.7324e-08
	α_{exp}	0.85	0.86	0.86	0.86	0.86	0.86
	θ_{exp}	1.08	1.09	1.09	1.09	1.09	1.09
RNSOR	IT	6	6	6	6	6	6
	CPU	0.0093	0.0115	0.0154	0.0201	0.0295	0.0325
	RES	2.4662e-07	2.2598e-07	2.0945e-07	1.9606e-07	1.8493e-07	1.7550e-07

Table 6. Experimental results for Example 4.2 for $\mu = 4$ with $\Omega = 0.5\theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	65	65	65	64	64	64
	CPU	0.0372	0.0465	0.0665	0.0902	0.1361	0.1554
	RES	9.8655e-07	9.2007e-07	8.6567e-07	9.7913e-07	9.3294e-07	8.9296e-07
	θ_{exp}	8.93	9.42	9.71	9.92	10.02	10.09
MN	IT	20	20	21	22	22	22
	CPU	0.0170	0.0228	0.0309	0.0418	0.0580	0.0684
	RES	4.5869e-07	9.1674e-07	7.3231e-07	5.3617e-07	6.8625e-07	8.1378e-07
NJ	IT	61	61	60	60	60	60
	CPU	0.0294	0.0390	0.0536	0.0708	0.0959	0.1183
	RES	9.4159e-07	8.8464e-07	9.9455e-07	9.4542e-07	9.0277e-07	8.6533e-07
	θ_{exp}	1.28	1.28	1.28	1.28	1.28	1.28
RNJ	IT	14	14	14	14	14	14
	CPU	0.0119	0.0161	0.0209	0.0261	0.0399	0.0455
	RES	3.8755e-07	3.7974e-07	3.7362e-07	3.6871e-07	3.6468e-07	3.6131e-07
NGS	IT	20	20	20	20	20	20
	CPU	0.0185	0.0234	0.0280	0.0373	0.0534	0.0631
	RES	4.8694e-07	4.7934e-07	4.7340e-07	4.6864e-07	4.6473e-07	9.9810e-07
	θ_{exp}	1.40	1.40	1.40	1.40	1.40	1.40
RNGS	IT	7	7	7	7	7	7
	CPU	0.0093	0.0124	0.0153	0.0197	0.0324	0.0355
	RES	8.7903e-08	8.3482e-08	8.0057e-08	7.7322e-08	7.5082e-08	7.3214e-08
	α_{exp}	0.74	0.74	0.74	0.74	0.74	0.74
NSOR	IT	9	9	9	9	9	9
	CPU	0.0122	0.0170	0.0245	0.0287	0.0392	0.0499
	RES	2.8429e-07	2.7806e-07	2.7317e-07	2.6924e-07	2.6599e-07	2.6328e-07
	α_{exp}	1.04	1.05	1.05	1.05	1.06	1.06
	θ_{exp}	1.46	1.48	1.48	1.48	1.49	1.49
RNSOR	IT	6	6	6	6	6	6
	CPU	0.0095	0.0129	0.0158	0.0211	0.0307	0.0345
	RES	8.8478e-07	8.1365e-07	7.5632e-07	7.0989e-07	6.7031e-07	6.3650e-07

Table 7. Experimental results for Example 4.2 for $\mu = -1$ with $\Omega = \theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
	θ_{exp}	-	-	-	-	-	-
MN	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
NJ	IT	144	142	140	138	136	135
	CPU	0.0622	0.0820	0.1083	0.1395	0.1922	0.2259
	RES	9.7797e^{-07}	9.5447e^{-07}	9.5175e^{-07}	9.6464e^{-07}	9.9032e^{-07}	9.6104e^{-07}
	θ_{exp}	0.58	0.58	0.58	0.58	0.58	0.58
RNJ	IT	80	80	78	78	76	76
	CPU	0.0362	0.0506	0.0686	0.0927	0.1248	0.1507
	RES	9.6243e^{-07}	8.2212e^{-07}	9.0684e^{-07}	8.0445e^{-07}	9.1373e^{-07}	8.2956e^{-07}
NGS	IT	108	106	105	103	102	101
	CPU	0.0506	0.0709	0.0895	0.1204	0.1769	0.1923
	RES	9.3164e^{-07}	9.6558e^{-07}	9.2823e^{-07}	9.9901e^{-07}	9.8872e^{-07}	9.8869e^{-07}
	θ_{exp}	0.70	0.70	0.74	0.76	0.79	0.79
RNGS	IT	73	73	76	77	80	81
	CPU	0.0339	0.0451	0.0641	0.0854	0.1262	0.1459
	RES	9.2604e^{-07}	9.2804e^{-07}	9.2604e^{-07}	9.7084e^{-07}	9.6950e^{-07}	9.7140e^{-07}
NSOR	α_{exp}	1.10	1.08	1.08	1.08	1.08	1.06
	IT	97	98	96	95	95	95
	CPU	0.0500	0.0731	0.1010	0.1295	0.1776	0.2097
	RES	9.6696e^{-07}	9.1422e^{-07}	9.8778e^{-07}	9.8099e^{-07}	9.0054e^{-07}	9.7730e^{-07}
RNSOR	α_{exp}	0.81	0.77	0.77	0.78	0.76	0.76
	θ_{exp}	0.48	0.46	0.46	0.47	0.46	0.46
	IT	61	60	60	59	59	58
	CPU	0.0336	0.0406	0.0541	0.0800	0.0997	0.1136
	RES	9.1008e^{-07}	9.7441e^{-07}	8.7713e^{-07}	9.8654e^{-07}	9.1768e^{-07}	9.7513e^{-07}

Table 8. Experimental results for Example 4.2 for $\mu = -1$ with $\Omega = 0.5\theta\hat{Q}$.

	n	3600	4900	6400	8100	10000	12100
Picard	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
	θ_{exp}	-	-	-	-	-	-
MN	IT	-	-	-	-	-	-
	CPU	-	-	-	-	-	-
	RES	-	-	-	-	-	-
NJ	IT	127	125	123	121	120	119
	CPU	0.0538	0.0741	0.1179	0.1356	0.1761	0.2109
	RES	9.6186e^{-07}	9.5122e^{-07}	9.6111e^{-07}	9.8708e^{-07}	9.5432e^{-07}	9.3224e^{-07}
	θ_{exp}	0.83	0.83	0.83	0.83	0.83	0.83
RNJ	IT	104	102	100	100	98	98
	CPU	0.0448	0.0609	0.0845	0.1096	0.1434	0.1769
	RES	9.3992e^{-07}	9.5833e^{-07}	9.9833e^{-07}	8.8561e^{-07}	9.4988e^{-07}	8.6237e^{-07}
NGS	IT	90	88	87	86	85	84
	CPU	0.0452	0.0609	0.0762	0.1052	0.1411	0.1721
	RES	9.1040e^{-07}	9.7080e^{-07}	9.4662e^{-07}	9.3824e^{-07}	9.4192e^{-07}	9.5544e^{-07}
	θ_{exp}	0.73	0.76	0.80	0.83	0.84	0.88
RNGS	IT	64	66	68	71	72	73
	CPU	0.0349	0.0527	0.0699	0.0904	0.1256	0.1347
	RES	8.8145e^{-07}	9.7844e^{-07}	9.6339e^{-07}	8.7367e^{-07}	8.9332e^{-07}	9.7252e^{-07}
	α_{exp}	1.08	1.08	1.07	1.06	1.06	1.04
NSOR	IT	81	80	80	80	80	80
	CPU	0.0374	0.0505	0.0705	0.0915	0.1330	0.1617
	RES	9.2347e^{-07}	9.0060e^{-07}	8.8527e^{-07}	8.9654e^{-07}	9.0857e^{-07}	9.3023e^{-07}
	α_{exp}	0.94	0.91	0.92	0.88	0.88	0.86
	θ_{exp}	0.68	0.66	0.67	0.64	0.64	0.63
RNSOR	IT	64	63	63	62	62	62
	CPU	0.0294	0.0457	0.0612	0.0880	0.1180	0.1456
	RES	8.7371e^{-07}	9.6096e^{-07}	8.5315e^{-07}	9.4529e^{-07}	8.7549e^{-07}	8.5632e^{-07}

5. Conclusions

In this paper, a class of relaxed Newton-type matrix splitting iteration methods have been established for solving the generalized absolute value equations by introducing a parameter and using the matrix splitting technique. To ensure the convergence of the RNMS method, some sufficient theorems are given. We use two numerical experiments from the LCP to show that compared with the existing Picard method [23,24], the existing MN method [8], the existing NMS method [11], the new RNMS method is feasible under certain conditions.

Acknowledgments

The authors would like to thank the three anonymous referees for providing helpful suggestions, which greatly improved the paper. This research was supported by the Fundamental Research Funds for the Central Universities (N2224005-1).

Conflict of interest

The authors declare that they have no competing interests.

References

1. J. Rohn, A theorem of the alternatives for the equation $Ax + B|x| = b$, *Linear Multilinear Algebra*, **52** (2004), 421–426. https://doi.org/10.1080/0308108042000220686
2. 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
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. O. L. Mangasarian, Absolute value programming, *Comput. Optim. Appl.*, **36** (2007), 43–53. https://doi.org/10.1007/s10589-006-0395-5
5. J. Rohn, Systems of linear interval equations, *Linear Algebra Appl.*, **126** (1989), 39–78. https://doi.org/10.1016/0024-3795(89)90004-9
6. L. Abdallah, M. Haddou, T. Migot, Solving absolute value equation using complementarity and smoothing functions, *J. Comput. Appl. Math.*, **327** (2018), 196–207. https://doi.org/10.1016/j.cam.2017.06.019
7. L. Caccetta, B. Qu, G. L. Zhou, A globally and quadratically convergent method for absolute value equations, *Comput. Optim. Appl.*, **48** (2011), 45–58. https://doi.org/10.1007/s10589-009-9242-9
8. 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
9. 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
10. Y. Cao, Q. Shi, S. L. Zhu, A relaxed generalized Newton iteration method for generalized absolute value equations, *AIMS Math.*, **6** (2021), 1258–1275. https://doi.org/10.3934/math.2021078

11. 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
12. C. Zhang, Q. J. Wei, Global and finite convergence of a generalized Newton method for absolute value equations, *J. Optim. Theory Appl.*, **143** (2009), 391–403. https://doi.org/10.1007/s10957-009-9557-9
13. J. Rohn, An algorithm for solving the absolute value equations, *Electron. J. Linear Algebra*, ***18*** (2009), 589–599. https://doi.org/10.13001/1081-3810.1332
14. P. Guo, S. L. Wu, C. X. Li, On the SOR-like iteration method for solving absolute value equations, *Appl. Math. Lett.*, **97** (2019), 107–113. https://doi.org/10.1016/j.aml.2019.03.033
15. Y. F. Ke, C. F. Ma, SOR-like iteration method for solving absolute value equations, *Appl. Math. Comput.*, **311** (2017), 195–202. https://doi.org/10.1016/j.amc.2017.05.035
16. X. Dong, X. H. Shao, H. L. Shen, A new SOR-like method for solving absolute value equations, *Appl. Numer. Math.*, **156** (2020), 410–421. https://doi.org/10.1016/j.apnum.2020.05.013
17. A. Mansoori, M. Erfanian, A dynamic model to solve the absolute value equations, *J. Comput. Appl. Math.*, **333** (2018), 28–35. https://doi.org/10.1016/j.cam.2017.09.032
18. C. R. Chen, Y. N. Yang, D. M. Yu, D. R. Han, An inverse-free dynamical system for solving the absolute value equations, *Appl. Numer. Math.*, **168** (2021), 170–181. https://doi.org/10.1016/j.apnum.2021.06.002
19. S. L. Hu, Z. H. Huang, Q. Zhang, A generalized Newton method for absolute value equations associated with second order cones, *J. Comput. Appl. Math.*, **235** (2012), 1490–1501. https://doi.org/10.1016/j.cam.2010.08.036
20. X. Li, X. X. Yin, A new modified Newton-type iteration methods for solving generalized absolute value equations, *ArXiv*, 2103. https://doi.org/10.48550/arXiv.2103.09452
21. C. X. Li, S. L. Wu, A shift splitting iteration method for generalized absolute value equations, *Comput. Meth. Appl. Math.*, **21** (2021), 863–872.
22. D. F. Han, The majorant method and convergence for solving nondifferentiable equations in Banach space, *Appl. Math. Comput.*, **118** (2001), 73–82. https://doi.org/10.1016/S0096-3003(99)00183-6
23. J. L. Dong, M. Q. Jiang, A modified modulus method for symmetric positive-definite linear complementarity problems, *Numer. Linear Algebra Appl.*, **16** (2009), 129–143. https://doi.org/10.1002/nla.609
24. 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
25. G. H. Golub, C. F. Van Loan, *Matrix computations*, Johns Hopkins University Press, 2013.
26. A. Frommer, G. Mayer, Convergence of relaxed parallel multisplitting methods, *Linear Algebra Appl.*, **119** (1989), 141–152. https://doi.org/10.1016/0024-3795(89)90074-8
27. S. L. Wu, C. X. Li, Two-sweep modulus-based matrix splitting iteration methods for linear complementarity problems, *J. Comput. Appl. Math.*, **302** (2016), 327–339. https://doi.org/10.1016/j.cam.2016.02.011

AIMS Press