Mathematics

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

AIMS Mathematics, 6(2): 1258–1275.
DOI:10.3934/math.2021078
Received: 02 September 2020
Accepted: 10 November 2020
Published: 12 November 2020

Research article

A relaxed generalized Newton iteration method for generalized absolute value equations

Yang Cao, Quan Shi* **and Sen-Lai Zhu**

School of Transportation and Civil Engineering, Nantong University, Nantong, 226019, P. R. China

* **Correspondence:** Email: shiquannt@sina.com; Tel: +8651385012789; Fax: +8651385012930.

Abstract: To avoid singular generalized Jacobian matrix and further accelerate the convergence of the generalized Newton (GN) iteration method for solving generalized absolute value equations $Ax - B|x| = b$, in this paper we propose a new relaxed generalized Newton (RGN) iteration method by introducing a relaxation iteration parameter. The new RGN iteration method involves the well-known GN iteration method and the Picard iteration method as special cases. Theoretical analyses show that the RGN iteration method is well defined and globally linearly convergent under suitable conditions. In addition, a specific sufficient condition is studied when the coefficient matrix A is symmetric positive definite. Finally, two numerical experiments arising from the linear complementarity problems are used to illustrate the effectiveness of the new RGN iteration method.

Keywords: generalized absolute value equations; Newton method; relaxation; globally convergence
Mathematics Subject Classification: 65F10

1. Introduction

Consider the following generalized absolute value equations (GAVE)

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

where $A, B \in \mathbb{R}^{n\times n}$ are two given matrices and $b \in \mathbb{R}^n$ is a given vector, and $|\cdot|$ denotes the componentwise absolute value. Especially, if $B = I$, where I stands for the identity matrix, then the GAVE (1.1) reduces to the standard absolute value equations (AVE)

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

The GAVE (1.1) and the AVE (1.1) arise in many scientific computing and engineering problems, including the linear programming problems, the linear complementarity problems (LCP), the bimatrix games, the quadratic programming and so on, see [1–4] for more details. Taking the well-known LCP as an example: for a given matrix $M \in \mathbb{R}^{n\times n}$ and a given vector $q \in \mathbb{R}^n$, find two vectors $z, w \in \mathbb{R}^n$ such that

$$z \geq 0, \quad w = Mz + q \geq 0 \quad \text{and} \quad z^T w = 0. \tag{1.3}$$

Here and thereafter, $(\cdot)^T$ denotes the transpose of either a vector or a matrix. By simply letting $z = |x| - x$ and $w = |x| + x$, then the LCP (1.3) can be equivalently transformed into the GAVE

$$(M + I)x - (M - I)|x| = q \quad \text{with} \quad x = \frac{1}{2}((M - I)z + q). \tag{1.4}$$

In fact, the GAVE (1.4) can also be transformed into the LCP (1.3) [5, 6]. For more details of the relation between the GAVE (1.1) and the LCP (1.3), please see [7–9].

In recent decades, much more attention has been paid to the GAVE (1.1) and the AVE (1.2). On one hand, some sufficient conditions have been studied to guarantee the existence and uniqueness of the solution of the GAVE (1.1) [10–14]. Rohn showed that the GAVE (1.1) is uniquely solvable for any $b \in \mathbb{R}^n$ if $\sigma_{\min}(A) > \sigma_{\max}(|B|)$ [10]. Wu and Li extended the results of [10], and exhibited that the GAVE (1.1) is uniquely solvable for any $b \in \mathbb{R}^n$ if $\sigma_{\min}(A) > \sigma_{\max}(B)$ [11, 12]. In [13], Rohn et al. showed that the GAVE (1.1) is uniquely solvable for any $b \in \mathbb{R}^n$ if $\rho(|A^{-1}B|) < 1$. Here and in the sequel, $\sigma_{\min}(\cdot)$, $\sigma_{\max}(\cdot)$ and $\rho(\cdot)$ denote the minimal singular value, the maximal singular value and the spectral radius of the corresponding matrix, respectively. On the other hand, many efficient iteration methods, including the concave minimization algorithm [15, 16], the sign accord algorithm [17], the optimization algorithm [18], the hybrid algorithm [19], the preconditioned AOR iterative method [20], the Picard-HSS iteration method [21], the Newton-type method [22–24] and so on, have been studied for solving the GAVE (1.1).

Due to the existence of the nonlinear term $B|x|$, the GAVE (1.1) can be regarded as a system of nonlinear equations

$$F(x) = 0 \quad \text{with} \quad F(x) = Ax - B|x| - b. \tag{1.5}$$

As a result, the well-known Newton iteration method

$$x^{k+1} = x^k - F'(x^k)^{-1} F(x^k), \ k = 0, 1, 2, \cdots, \tag{1.6}$$

can be used provided that the Jacobian matrix $F'(x)$ of $F(x)$ exists and is invertible. However, the Newton iteration method (1.6) can not be used directly to solve the GAVE (1.1) since $F(x) = Ax - B|x| - b$ is non-differentiable. For the special case, i.e., $B = I$, by considering $F(x) = Ax - |x| - b$ as a piece-wise linear vector function, Mangasarian in [22] used the generalized Jacobian $\partial|x|$ of $|x|$ based on a subgradient of its components and presented the following generalized Newton (GN) iteration method

$$x^{(k+1)} = (A - D(x^{(k)}))^{-1} b, \quad k = 0, 1, 2, \cdots \tag{1.7}$$

to get an approximate solution of the AVE (1.2), where $D(x^{(k)}) = \partial|x| = diag(sign(x^{(k)}))$ and $sign(x)$ stands for a vector with components equal to 1, 0, or -1 depending on whether the corresponding component of x is positive, zero or negative. Theoretical analysis showed that the GN iteration method (1.7) is globally linearly convergent under certain conditions [22]. Hu et al. extended the GN iteration scheme (1.7) to solve the GAVE (1.1) and proposed a weaker convergence condition [25]. For a general matrix B, the specific GN iteration scheme is

$$x^{(k+1)} = (A - BD(x^{(k)}))^{-1}b, \quad k = 0, 1, 2, \cdots . \tag{1.8}$$

Recently, convergence results of the GN iteration schemes (1.7) and (1.8) have been further discussed in [26–28]. From the GN iteration scheme (1.7) or (1.8), we can see that the coefficient matrix $A - D(x^k)$ or $A - BD(x^k)$ is changed at each iteration step. For large problems, it is very difficult especially if the coefficient matrix is ill-conditioned. In addition, if the generalized Jacobian matrix is singular, then the GN iteration method fails. To remedy these, a lot of effective improvements have been presented, such as a stable and quadratic locally convergent iteration scheme [28], the generalized Traub's method [29], the modified GN iteration method [24, 30], the inexact semi-smooth Newton iteration method [31], a new two-step iterative method [32] and so on. All these improvements greatly accelerate the convergence rate of the GN iteration method. However, when the singular generalized Jacobian matrix happens, these newly developed iteration methods fail, too.

In this paper, by introducing a relaxation iteration parameter, we propose a relaxed generalized Newton (RGN) iteration method to solve the GAVE (1.1). In fact, the RGN iteration method is a generalization of the GN iteration method [22] and the Picard iteration method [13] studied recently. The advantage of the new RGN iteration method is twofold. By introducing suitable iteration parameter, it not only can avoid singularity of the generalized Jacobian matrix, but also improves the convergence rate. Theoretically, we prove that the RGN iteration method is well defined and globally linearly convergent under certain conditions. Moreover, a specific sufficient convergence condition is presented when the coefficient matrix A is symmetric positive definite. With two numerical examples, we show that the new proposed RGN iteration method is much more efficient than some existing Newton-type iteration methods.

The rest of this paper is organized as follows. In Section 2, the RGN iteration method is introduced to solve the GAVE (1.1). Convergence analyses are studied in detail in Section 3. In Section 4, two numerical examples from the LCP (1.3) are presented to demonstrate the effectiveness of our new method. Finally, we end this paper with some conclusions and outlook in Section 5.

2. The relaxed generalized Newton iteration method

In this section, a new relaxed generalized Newton iteration method is introduced to solve the GAVE (1.1).

Let $\theta \geq 0$ be a nonnegative real parameter. Based on the Newton iteration scheme (1.6) and the ideas studied [22, 30], a new iteration scheme is introduced to solve the GAVE (1.1)

$$F(x^k) + (\partial F(x^k) + (1 - \theta)BD(x^k))(x^{k+1} - x^k) = 0. \tag{2.1}$$

Substituting $F(x^k) = Ax^k - B|x^k| - b$ (1.5) and the generalized Jacobian $\partial F(x^k) = A - BD(x^k)$ into (2.1), we obtain

$$Ax^k - B|x^k| - b + (A - \theta BD(x^k))(x^{k+1} - x^k) = 0. \tag{2.2}$$

Since $D(x) = diag(sign(x))$ is a diagonal matrix and satisfies $D(x^k)x^k = |x^k|$ [22]. Then the new iteration scheme (2.1) or (2.2) is simplified into the following final form

$$(A - \theta BD(x^k))x^{k+1} = b + (1 - \theta)B|x^k|, \quad k = 0, 1, 2, \cdots . \tag{2.3}$$

Here, the iteration parameter θ can be regarded as a role of relaxation, which can avoid the singularity problems and adjust the condition number of the coefficient matrix $A - \theta BD(x^k)$ so as to improve the convergence rate of the GN iteration method (1.8). So, we call the new iteration method (2.3) the relaxed generalized Newton (RGN) iteration method. In particular, if $\theta = 1$, then the RGN iteration method (2.3) reduces to the GN iteration method (1.8). If $\theta = 0$, then the RGN iteration method (2.3) becomes

$$Ax^{k+1} = b + B|x^k|, \quad k = 0, 1, 2, \cdots, \tag{2.4}$$

which is known as the Picard iteration method [7, 13].

The RGN iteration method (2.3) is well defined provided that the coefficient matrix $A - \theta BD(x^k)$ is nonsingular at each iteration step. The following theorem presents a sufficient condition. To this end, we first define a set of matrices

$$\mathcal{D} := \{\textit{an } n \times n \textit{ diagonal matrix with each diagonal element being } 1,\ -1 \textit{ or } 0\} \tag{2.5}$$

since the diagonal matrix $D(x) = diag(sign(x))$ may change at each iteration step.

Theorem 2.1. *Let $A, B \in \mathbb{R}^{n\times n}$, $\theta \geq 0$ be a nonnegative real parameter and $D \in \mathbb{R}^{n\times n}$ be any matrix in the set $\mathcal{D}$ (2.5). Let $\lambda_{min}(A^TA)$ and $\lambda_{max}(B^TB)$ be the smallest eigenvalue of A^TA and the largest eigenvalue of B^TB, respectively. If*

$$\frac{\lambda_{min}(A^TA)}{\lambda_{max}(B^TB)} > \theta^2, \tag{2.6}$$

then $A - \theta BD$ is nonsingular and the RGN iteration method (2.3) is well defined.

Proof. We argue it by contradiction. If $A - \theta BD$ is singular, then there exists a nonzero vector x such that

$$(A - \theta BD)x = 0.$$

In addition, since $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix with each diagonal element being 1, −1 or 0, D^TD is a diagonal matrix, too, and each diagonal element is either 1 or 0. Thus, it holds

$$x^Tx \geq x^TD^TDx.$$

Then we have the following contradiction

$$\theta^2\lambda_{max}(B^TB) < \lambda_{min}(A^TA) \leq \frac{x^TA^TAx}{x^Tx} = \theta^2\frac{x^TD^TB^TBDx}{x^Tx} \leq \theta^2\frac{x^TD^TB^TBDx}{x^TD^TDx} = \theta^2\frac{w^TB^TBw}{w^Tw} \leq \theta^2\lambda_{max}(B^TB).$$

Therefore, $A - \theta BD$ is nonsingular and the RGN iteration method (2.3) is well defined provided that the condition (2.6) holds. □

Remark 2.1. *It should be noted that the condition given in Theorem 2.1 is a theoretical generalization of some recent works. In particular, if $B = I$ and $\theta = 1$, then the condition (2.6) becomes $\lambda_{min}(A^TA) > 1$, which means that all singular values of A exceed 1 and is in good agreement with [22, Lemma 2.1]. If only $\theta = 1$, then the condition (2.6) is the one given in [25, Theorem 3.1]. In addition, if $\theta = 0$, then the condition (2.6) is equivalent to show that the matrix A is nonsingular, which clearly shows that the Picard iteration method (2.4) is well defined.*

To better implement the new RGN iteration method (2.3) in actual computations, we present an algorithmic version of the RGN iteration method (2.3) as follows. Here, $\|\cdot\|_2$ denotes the Euclidean norm of either a vector or a matrix.

Algorithm 2.1. *(The RGN iteration method)*

```
1). Choose an arbitrary initial vector $x^0$ and a nonnegative parameter $\theta$. Given $\varepsilon$ and set $k = 0$;
2). If $\|Ax^k - B|x^k| - b\|_2 \le \varepsilon\|b\|_2$, stop;
3). Compute $D(x^k) = diag(sign(x^k))$;
4). Solve the following linear system to obtain $x^{k+1}$
        $(A - \theta BD(x^k))x^{k+1} = b + (1-\theta)B|x^k|$;
5). Set $k = k + 1$. Go to Step 2.
```

3. Convergence analysis

In this section, we will establish the convergence theory of the RGN iteration method (2.3) for solving the GAVE (1.1). Specially, two general convergence conditions of the RGN iteration method (2.3) will be presented firstly. Then, a sufficient convergence condition is proposed when the system matrix A is symmetric positive definite. In addition, as the special cases of our new RGN iteration method (2.3), the convergence conditions of the GN iteration method (1.8) and the Picard iteration method (2.4) can be acquired immediately.

3.1. General sufficient convergence conditions

In this subsection, we first study some sufficient convergence conditions only when the RGN iteration method (2.3) is well defined.

Theorem 3.1. *Let $A, B \in \mathbb{R}^{n\times n}$, $\theta \ge 0$ be a nonnegative real parameter and satisfy the condition (2.6). Let $D \in \mathbb{R}^{n\times n}$ be any matrix in the set $\mathcal{D}$ (2.5). If*

$$\|(A - \theta BD)^{-1}\|_2\|B\|_2 < \frac{1}{1+\theta}, \tag{3.1}$$

then the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

Proof. Let x^* be a solution of the GAVE (1.1), then it satisfies

$$Ax^* - B|x^*| - b = 0. \tag{3.2}$$

Subtracting (3.2) from (2.3), we obtain

$$\begin{aligned} A(x^{k+1} - x^*) &= \theta BD(x^k)x^{k+1} + (1-\theta)B|x^k| - B|x^*| \\ &= \theta BD(x^k)(x^{k+1} - x^* + x^*) + (1-\theta)B|x^k| - B|x^*| \\ &= \theta BD(x^k)(x^{k+1} - x^*) + \theta BD(x^k)(x^* - x^k) + B(|x^k| - |x^*|). \end{aligned}$$

Hence,

$$(A - \theta BD)(x^{k+1} - x^*) = \theta BD(x^k)(x^* - x^k) + B(|x^k| - |x^*|).$$

By assumptions, we have

$$x^{k+1} - x^* = (A - \theta BD(x^k))^{-1}[\theta BD(x^k)(x^* - x^k) + B(|x^k| - |x^*|)]. \tag{3.3}$$

Taking the Euclidean norm on both sides of (3.3), we obtain

$$\begin{aligned}
\|x^{k+1} - x^*\|_2 &= \|(A - \theta BD(x^k))^{-1}[\theta BD(x^k)(x^* - x^k) + B(|x^k| - |x^*|)]\|_2 \\
&\leq \|(A - \theta BD(x^k))^{-1}\|_2 \cdot \|\theta BD(x^k)(x^* - x^k) + B(|x^k| - |x^*|)\|_2 \\
&\leq \|(A - \theta BD(x^k))^{-1}\|_2 \cdot (\|\theta BD(x^k)\|_2 + \|B\|_2)\|x^k - x^*\|_2 \\
&\leq \|(A - \theta BD(x^k))^{-1}\|_2 \cdot (\theta\|B\|_2 + \|B\|_2)\|x^k - x^*\|_2 \\
&= (1 + \theta)\|(A - \theta BD(x^k))^{-1}\|_2\|B\|_2\|x^k - x^*\|_2,
\end{aligned} \tag{3.4}$$

where the inequality $\| \, |x^k| - |x^*| \, \|_2 \leq \|x^k - x^*\|_2$ is used. From (3.4), we can see that the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1) provided that the condition (3.1) is satisfied. □

Theorem 3.2. *Under the conditions in Theorem 3.1. Further assume that A is nonsingular. If*

$$\|A^{-1}\|_2\|B\|_2 < \frac{1}{1 + 2\theta}, \tag{3.5}$$

then the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

Proof. According to Theorem 3.1, we only need to verify the condition (3.1). Under the condition (3.5), by the Banach perturbation lemma (see [33, Lemma 2.3.3]), we have

$$\begin{aligned}
\|(A - \theta BD)^{-1}\|_2\|B\|_2 &\leq \frac{\|A^{-1}\|_2\|B\|_2}{1 - \|A^{-1}\|_2\|\theta BD\|_2} \\
&\leq \frac{\|A^{-1}\|_2\|B\|_2}{1 - \theta\|A^{-1}\|_2\|B\|_2} \\
&< \frac{\frac{1}{1+2\theta}}{1 - \frac{\theta}{1+2\theta}} \\
&= \frac{1}{1 + \theta}.
\end{aligned}$$

Therefore, the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1) if the condition (3.5) is satisfied. □

As mentioned in Section 2, the well-known GN iteration method (1.8) and the Picard iteration method (2.4) are special cases of the new RGN iteration method (2.3) when the relaxation parameter θ is 1 and 0, respectively. By simply letting $\theta = 1$ and 0, we can obtain the following two corollaries, which describe the convergence conditions of the GN iteration method (1.8) and the Picard iteration method (2.4), respectively, for solving the GAVE (1.1).

Corollary 3.1. *Let $A, B \in \mathbb{R}^{n\times n}$, and $D \in \mathbb{R}^{n\times n}$ be any matrix in the set $\mathcal{D}$ (2.5). Assume that $A - BD$ is nonsingular. If*

$$\|(A - BD)^{-1}\|_2\|B\|_2 < \frac{1}{2},$$

or if A is nonsingular and

$$\|A^{-1}\|_2\|B\|_2 < \frac{1}{3},$$

then the GN iteration method (1.8) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

Corollary 3.2. *Let $A, B \in \mathbb{R}^{n\times n}$. Assume that A is nonsingular. If*

$$\|A^{-1}\|_2\|B\|_2 < 1,$$

then the Picard iteration method (2.4) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

3.2. The case of symmetric positive definite

In this subsection, we turn to discuss the convergence conditions of the RGN iteration method (2.3) for solving the GAVE (1.1) when the system matrix A is symmetric positive definite.

Theorem 3.3. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix, $B \in \mathbb{R}^{n\times n}$, $D \in \mathbb{R}^{n\times n}$ be any matrix in the set $\mathcal{D}$ (2.5), θ be a positive constant and satisfy (2.6). Further assume that μ_{min} is the smallest eigenvalue of the matrix A and $\|B\|_2 = \tau$. If*

$$\mu_{min} > (1 + 2\theta)\tau, \tag{3.6}$$

then the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

Proof. Since the matrix A is symmetric positive definite, it is easy to check that

$$\|A^{-1}\|_2\|B\|_2 \leq \frac{\tau}{\mu_{min}}.$$

If μ_{min} and τ further satisfy the condition (3.6), we have

$$\|A^{-1}\|_2\|B\|_2 \leq \frac{\tau}{\mu_{min}} < \frac{1}{1 + 2\theta}.$$

Therefore, by Theorem 3.2, we obtain that the RGN iteration method (2.3) converges linearly from any starting point to a solution x^* of the GAVE (1.1). This completes the proof. □

In Theorem 3.3, by setting $\theta = 1$ and 0, we have the following two corollaries to guarantee the convergence of the GN iteration method (1.8) and the Picard iteration method (2.4) for solving the GAVE (1.1), respectively.

Corollary 3.3. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and μ_{min} be its smallest eigenvalue. Let $B \in \mathbb{R}^{n\times n}$ and $\|B\|_2 = \tau$. Let $D \in \mathbb{R}^{n\times n}$ be any matrix in the set $\mathcal{D}$ (2.5). If*

$$\mu_{min} > 3\tau,$$

then the GN iteration method (1.8) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

Corollary 3.4. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix and μ_{min} be its smallest eigenvalue. Let $B \in \mathbb{R}^{n\times n}$ and $\|B\|_2 = \tau$. If*

$$\mu_{min} > \tau,$$

then the Picard iteration method (2.4) converges linearly from any starting point to a solution x^* of the GAVE (1.1).

4. Numerical experiments

In this section, we present some numerical examples arising from two types of LCP (1.3) to show the effectiveness of the RGN iteration method (2.6), and demonstrate the advantages of the new RGN iteration method over the well-known Lemke's method, the Picard iteration method (2.4), the GN iteration method (1.8) and the modified GN iteration (MGN) method [30] from aspects of the iteration counts (denoted by "IT") and the elapsed CPU times (denoted by "CPU"). The first LCP comes from [4], which have been studied for standard test problem by many researchers. The second LCP arising from the practical traffic single bottleneck models [34], which are usually solved by the well-known Lemke's method. Note that the Lemke's method is one of the most efficient direct methods for solving the LCP (1.3).

In our experiments, the initial guess vector is the zero vector. All runs are terminated once

$$\mathrm{RES}(x^{(k)}) := \frac{\|Ax^{(k)} - B|x^{(k)}| - b\|_2}{\|b\|_2} \le 10^{-7}$$

or if the prescribed iteration number $k_{\max} = 5000$ is exceeded. At each iteration step of the RGN iteration method, the Picard iteration method, the GN iteration method and the MGN iteration method, we need to solve a system of linear equations with the coefficient matrix $A - \theta BD$, A, $A - BD$ and $A + I - BD$ respectively. Here, these systems of linear equations are solved by the sparse LU factorization when these coefficient matrices are nonsymmetric and by the sparse Cholesky factorization when these coefficient matrices are symmetric positive definite. To efficiently implement the RGN iteration method, we need to choose the relaxation iteration parameter θ in advance. The convergence rates of all parameter dependent iteration methods heavily depend on the particular choice of the iteration parameter. The analytic determination of the value θ which results in the fastest convergence of the RGN iteration method appears to be quite a difficult problem. Here, the relaxation iteration parameter θ used in the new RGN iteration method is chosen to be the experimentally optimal one $\theta_{\exp}$, which leads to the smallest iteration step. In the following tables, "-" means that the corresponding iteration method does not converge to the approximate solution within $k_{\max}$ iteration steps or even diverges. All computations are run in MATLAB (version R2014a) in double precision, Intel(R) Core(TM) (i5-3337U CPU, 8G RAM) Windows 8 system. Here, we use the Matlab codes presented in https://ww2.mathworks.cn/matlabcentral/fileexchange/41485 to test the Lemke's method.

Example 4.1. *([4]) Consider the LCP (1.3), in which $M \in \mathbb{R}^{n\times n}$ is given by $M = \widehat{M} + \mu I \in \mathbb{R}^{n\times n}$ and $q \in \mathbb{R}^n$ is given by $q = -Mz^{(*)}$, where*

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

is a block-tridiagonal matrix,

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

is a tridiagonal matrix, and

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

is the unique solution of the LCP (1.3). Here $n = m^2$. From the discussion presented in Section 1, we can see that the LCP (1.3) can be equivalently expressed as the GAVE (1.1), where $A = M + I$ and $B = M - I$. The exact solution of the GAVE (1.1) is

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

For the first example, we take two cases of the parameter μ, i.e., $\mu = 0$ and $\mu = -1$, in actual computations. For each parameter μ, four increasing sizes, i.e., $m = 30, 60, 90, 120$, are considered. The corresponding dimensions for each problem are $n = 900, 3600, 8100, 14400$, respectively. Note that for the case $\mu = 0$, both the matrices M and A are symmetric positive definite, while for the case $\mu = -1$, the matrix M is symmetric indefinite and the matrix A is symmetric positive definite. In Tables 1 and 2, we list the numerical results of different methods for $\mu = 0$ and $\mu = -1$, respectively.

Table 1. Numerical results for Example 4.1 with $\mu = 0$.

Method		m			
		30	60	90	120
Lemke's	IT	900	3600	-	-
	CPU	35.1067	2537.5000	-	-
Picard	IT	318	1134	2397	4080
	CPU	0.0589	0.9686	4.1924	12.8770
	RES	9.7418e-8	9.9368e-8	9.9926e-8	9.9875e-8
GN	IT	2	2	2	2
	CPU	0.0267	0.2281	1.0153	3.9991
	RES	1.2664e-15	1.8294e-15	2.2163e-15	2.6124e-15
MGN	IT	325	1140	2404	4086
	CPU	2.6393	108.7958	1004.0956	4909.9687
	RES	9.7488e-8	9.9906e-8	9.9553e-8	9.9875e-8
RGN	$\theta_{\exp}$	1	1	1	1
	IT	2	2	2	2
	CPU	0.0267	0.2281	1.0153	3.9991
	RES	1.2664e-15	1.8294e-15	2.2163e-15	2.6124e-15

Table 2. Numerical results for Example 4.1 with $\mu = -1$.

Method		m			
		30	60	90	120
Lemke's	IT	116	236	296	296
	CPU	1.5217	12.8138	41.6554	86.0441
Picard	IT	-	-	-	-
	CPU	-	-	-	-
	RES	-	-	-	-
GN	IT	17	17	17	16
	CPU	0.0943	1.2698	5.7498	17.7465
	RES	3.7830e-17	6.9307e-8	4.5695e-8	8.9227e-8
MGN	IT	17	16	16	16
	CPU	0.0921	1.2537	5.4685	17.1072
	RES	4.1276e-8	8.9075e-8	7.8098e-8	7.4129e-8
RGN	$\theta_{\exp}$	0.9970	0.9990	0.9993	0.9996
	IT	5	5	5	5
	CPU	0.0457	0.4612	2.1202	6.0750
	RES	4.7405e-8	1.1536e-9	5.3963e-10	8.9285e-11

From Tables 1 and 2, we can see that the GN iteration method and the new RGN iteration method perform much better than the other three computational methods in terms of both iteration steps and elapsed CPU times. For the case $\mu = 0$, the Lemke's method can not converge to a satisfactory solution for large problems. Although the Picard iteration method and the MGN iteration method converge, the numerical results show that the convergence rates of these two iteration methods are very slow. We have also noticed that the iteration steps of the Picard iteration step and the MGN iteration step are almost the same, but the elapsed CPU times show that the MGN iteration method costs much more expensive than the Picard iteration method. The reason is that the coefficient matrix of the MGN iteration method is changed at each iteration step. The best choice of the relaxation iteration parameter in the RGN iteration method is $\theta_{exp} = 1$, which means that the GN iteration method is the best one. For the case $\mu = -1$, the Lemke's method computes the exact solution for all test problems, but the iteration steps and the elapsed CPU times show that this method is not competitive in actual computations. The Picard iteration method diverges. This is because the matrix M is indefinite and the convergence conditions in Corollary 3.2 and Corollary 3.4 can not be satisfied. Among the GN iteration method, the MGN iteration method and the RGN iteration method, we can see from the numerical results that the new RGN iteration method is the best one.

Example 4.2. *([4]) Consider the LCP (1.3), in which $M = \widehat{M} + +\mu I \in \mathbb{R}^{n\times n}$ and $q = -Mz^{(*)}$. Different from Example 4.1, we assume that the matrix $\widehat{M}$ in the second example is nonsymmetric, i.e.,*

$$\widehat{M} = Tridiag(-I, S, -I) = \begin{bmatrix} S & -0.5I & 0 & \cdots & 0 & 0 \\ -1.5I & S & -0.5I & \cdots & 0 & 0 \\ 0 & -1.5I & S & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & S & -0.5I \\ 0 & 0 & 0 & \cdots & -1.5I & S \end{bmatrix} \in \mathbb{R}^{n\times n}$$

is a block-tridiagonal matrix,

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

is a tridiagonal matrix, and

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

is the unique solution of the LCP (1.3).

Similar to Example 4.1, the second example can also be equivalently expressed as the GAVE (1.1). Note that Example 4.2 has the same exact solution as that of Example 4.1. For the second example, we still take two parameters for μ, i.e., $\mu = 0$ and $\mu = -1$. For each parameter μ, we consider four increasing sizes, i.e., $m = 30, 60, 90, 120$. Thus, the total dimensions for each problem are $n = 900, 3600, 8100, 14400$, respectively. Different from Example 4.1, both the matrices M and A are nonsymmetric positive definite for the case $\mu = 0$. For the case $\mu = -1$, the matrix M is nonsymmetric indefinite and the matrix A is nonsymmetric positive definite.

Table 3. Numerical results for Example 4.2 with $\mu = 0$.

Method		m			
		30	60	90	120
Lemke's	IT	900	3600	-	-
	CPU	37.4746	2680.6527	-	-
Picard	IT	47	47	66	84
	CPU	0.0801	0.5793	2.4141	8.6564
	RES	7.8841e-8	9.7605e-8	5.1331e-8	5.0082e-8
GN	IT	2	2	2	2
	CPU	0.0315	0.2400	1.1082	4.4723
	RES	1.2977e-15	1.9780e-15	2.3953e-15	2.8208e-15
MGN	IT	42	66	87	107
	CPU	1.5449	9.7049	89.1371	223.3420
	RES	9.5719e-8	6.7966e-8	7.7690e-8	8.4657e-8
RGN	θ_{exp}	1	1	1	1
	IT	2	2	2	2
	CPU	0.0315	0.2400	1.1082	4.4723
	RES	1.2977e-15	1.9780e-15	2.3953e-15	2.8208e-15

Table 4. Numerical results for Example 4.2 with $\mu = -1$.

Method		m			
		30	60	90	120
Lemke's	IT	75	135	195	255
	CPU	0.8728	9.3958	39.9455	81.1532
Picard	IT	-	-	-	-
	CPU	-	-	-	-
	RES	-	-	-	-
GN	IT	18	18	18	18
	CPU	0.1870	1.8350	5.8001	20.9968
	RES	1.7803e-16	1.8059e-16	1.8148e-16	1.8211e-16
MGN	IT	47	44	43	42
	CPU	0.8859	6.6168	32.7976	116.8199
	RES	8.8063e-8	9.7329e-8	8.4501e-8	8.2862e-8
RGN	θ_{exp}	0.9998	0.9997	0.9999	0.9999
	IT	9	9	8	8
	CPU	0.1302	1.2335	3.1282	12.2499
	RES	2.2628e-10	2.4845e-10	9.8938e-8	7.3902e-8

Tables 3 and 4 list the corresponding numerical results of different methods for $\mu = 0$ and $\mu = -1$, respectively. These numerical results further confirm the observations obtained from Tables 1 and 2, i.e., the GN iteration method and the new RGN iteration method are superior to other three computational methods in terms of computing efficiency. For the case $\mu = 0$, the Lemke's method converge very slow for small problems and do not converge within the given maximum iteration step for large problems. The other four computational methods are convergent. However, the Picard iteration method and the MGN iteration method converge very slow. The GN iteration method and the new RGN iteration method have the same computational results and converge very fast. That means the GN iteration method is the best one. Most important, the iteration steps of both the GN iteration method and the new proposed RGN iteration method are constant as the problem sizes grow. For the case $\mu = -1$, the Lemke's method performs much better than itself for the case $\mu = 0$. However, the computational results show that the Lemke's method is still not competitive in real applications. The Picard iteration method is still divergent. The reason is the same as that in Example 4.1. The GN iteration method, the MGN iteration method and the proposed RGN iteration method converge faster than the Lemke's method. From these numerical results, we see again that the RGN iteration method performs best among the three Newton-based iteration methods.

Example 4.3. *([34]) The second example comes from the single bottleneck model with both the homogeneous commuters and the heterogeneous commuters. The dynamic equilibrium conditions for the single bottleneck model can be transformed into the LCP (1.3), in which the system matrix M and the vector q have the following specific structure*

$$M = \begin{bmatrix} 0 & M_1 & M_2 & -M_3^T \\ -M_1^T & S & 0 & 0 \\ 0 & M_1 & I & 0 \\ M_3 & 0 & 0 & 0 \end{bmatrix} \quad \textit{and} \quad q = \begin{bmatrix} q_1 \\ s\mathbf{1} \\ q_2 \\ q_3 \end{bmatrix},$$

where the submatrices $M_1 \in \mathbb{R}^{(\Upsilon G)\times\Upsilon}$, $M_2 \in \mathbb{R}^{(\Upsilon G)\times(\Upsilon G)}$, $M_3 \in \mathbb{R}^{G\times(\Upsilon G)}$, $S \in \mathbb{R}^{\Upsilon\times\Upsilon}$ are

$$M_1 = \begin{bmatrix} I \\ \vdots \\ I \end{bmatrix}, M_2 = \begin{bmatrix} \frac{\beta_1+\gamma_1}{\alpha_1+\gamma_1} I & & 0 \\ & \ddots & \\ 0 & & \frac{\beta_g+\gamma_g}{\alpha_g+\gamma_g} I \end{bmatrix}, M_3 = \begin{bmatrix} \frac{1}{\alpha_1+\gamma_1}\mathbf{1}^T & 0 \\ & \vdots \\ 0 & \frac{1}{\alpha_g+\gamma_g}\mathbf{1}^T \end{bmatrix}, S = \begin{bmatrix} s & & & 0 \\ -s & \ddots & & \\ & \ddots & \ddots & \\ 0 & & -s & s \end{bmatrix},$$

the subvectors are

$$q_1 = \begin{bmatrix} -\frac{\alpha_1}{\alpha_1+\gamma_1}(\tau_1^* - \tau) & \cdots & -\frac{\alpha_g}{\alpha_g+\gamma_g}(\tau_g^* - \tau) \end{bmatrix} \in \mathbb{R}^{\Upsilon G},$$
$$q_2 = \begin{bmatrix} -(\tau_1^* - \tau) & \cdots & -(\tau_g^* - \tau) \end{bmatrix} \in \mathbb{R}^{\Upsilon G},$$

and

$$q_3 = \begin{bmatrix} -\frac{N_1}{\alpha_1+\gamma_1} & \cdots & -\frac{N_g}{\alpha_g+\gamma_g} \end{bmatrix} \in \mathbb{R}^{G}.$$

Here, $\tau \in \mathbb{T} = [0, 1, 2, \cdots, \Upsilon]$ and $g \in \mathbb{G} = [1, 2, \cdots, G]$ are the indexes for the time interval and the user group, respectively. When $G = 1$ and $G > 1$, then the LCP (1.3) can be used to study the homogeneous case and the heterogeneous case, respectively. s denotes the bottleneck capacity with units given by number of vehicles per time interval and N_g denotes the number of individuals in group g. α_g, β_g and γ_g are the unit costs (or value) of the travel time, arriving early to work and arriving late to work in group g, respectively. τ_g^* is the preferred arrival time in group g. **1** *stands for a vector of all ones.*

For the second example, the total dimension is $n = 2\Upsilon G + \Upsilon + G$. It is proved in [34] that the system matrix M is a copositive matrix and the LCP (1.3) has a unique solution. In [34], the authors used the Lemke's method to solve such problems and simulate the single bottleneck model for both the homogeneous case ($G = 1$) and the heterogenous case ($G = 3$). In addition, they took the total demand $N = 25$, the bottleneck capacity $s = 3$ for time unit and the preferred arrival time $\tau^* = 7$. The time duration is 10 time units. For the homogeneous case, i.e., $G = 1$, the unit costs are taken as $\alpha = 2$, $\beta = 1$ and $\gamma = 4$ for per time unit. For the heterogeneous case, i.e., $G = 3$, the unit costs are taken as $\alpha : \beta : \gamma$ rations $= 2 : 1 : 4$ and $\tau_g^* = 6, 7, 8$ for groups $1 - 3$, respectively. For more detailed selection of these parameters, please see [34]. Here, the corresponding LCP is further equivalently transformed into the GAVE (1.1), where $A = M + I$ and $B = M - I$. Then the Picard iteration method, the GN iteration method, the MGN iteration method and the new RGN iteration method are applied to solve the GAVE.

Numerical results of different computational methods are listed in Tables 5 and 6 for $G = 1$ and $G = 3$, respectively. From these two tables, we can see that the Lemke's method is successfully applied to solve the single bottleneck model, but the elapsed CPU times indicate that the Lemke's method costs very expensive. The GN iteration method fails to solve the test problems. This is because the singularity of the coefficient matrix $A - BD(x^k)$ occurs in implementing the GN iteration method. The Picard iteration method and the MGN iteration method can only be applied to some small problems. For large problems, these two iteration methods fail to converge within the prescribed largest iteration number. Our new RGN iteration method can be successfully applied to solve all the test problems and costs very cheap. Therefore, the new RGN iteration method is a powerful computational method to solve the GAVE (1.1).

Table 5. Numerical results for Example 4.3 with homogeneous case ($G = 1$).

Method		Υ			
		10	60	360	720
Lemke's	IT	41	239	1313	2559
	CPU	0.0064	0.1036	35.5910	337.6484
Picard	IT	396	-	-	-
	CPU	0.0174	-	-	-
	RES	9.0472e-8	-	-	-
GN	IT	-	-	-	-
	CPU	-	-	-	-
	RES	-	-	-	-
MGN	IT	168	311	-	-
	CPU	0.0452	0.2032	-	-
	RES	9.9551e-8	9.2794e-8	-	-
RGN	θ_{exp}	0.9998	0.9910	0.9991	0.9997
	IT	10	28	28	30
	CPU	0.0050	0.0332	0.3459	1.4890
	RES	5.5293e-7	4.1419e-7	4.5825e-7	9.7686e-8

Table 6. Numerical results for Example 4.3 with heterogeneous case ($G = 3$).

Method		Υ			
		10	60	360	720
Lemke's	IT	65	386	2084	3220
	CPU	0.0138	0.8541	277.3680	1191.3000
Picard	IT	-	-	-	-
	CPU	-	-	-	-
	RES	-	-	-	-
GN	IT	-	-	-	-
	CPU	-	-	-	-
	RES	-	-	-	-
MGN	IT	170	140	2943	-
	CPU	0.0432	0.3254	109.9264	-
	RES	9.5758e-8	9.0419e-8	9.0338e-8	-
RGN	θ_{exp}	0.9330	0.9900	0.9980	0.9992
	IT	23	35	60	79
	CPU	0.1969	0.4008	2.8071	12.6720
	RES	6.6725e-7	1.1437e-7	3.8152e-7	7.3033e-9

5. Conclusion

In this paper, by introducing a relaxation iteration parameter, a new relaxed generalized Newton (RGN) iteration method is proposed to solve the generalized absolute value equations. We have proved that the RGN iteration method is well defined and converges globally under certain conditions. Two numerical examples, both arising from the well-known LCP problem, are used to illustrate the efficiency of the new computational method. Numerical results show that the RGN iteration method converges and has much better computing efficiency than some existing methods provided that suitable relaxation iteration parameters are chosen.

Just like most of the parameter-based iteration methods, the choice of the iteration parameter is an open and a challenging problem. Here, the RGN iteration method is proved to be only linearly convergent. In some recent works, the GN iteration method has been modified to be a globally and quadratically iteration method under very strong conditions. Therefore, how to improve the RGN iteration method needs further-in-depth studies. In addition, the generalized absolute value equations with general nonlinear term, which arise in nonlinear complementarity problems [35], implicit complementarity problems [36, 37], quasi complementarity problems [38, 39], are of great interesting. Future work should focus on estimating the quasi-optimal value of the relaxation iteration parameter, finding globally and quadratically convergent RGN iteration method, extending to solve more applications and so on.

Acknowledgments

The author are very much indebted to An Wang for writing the Matlab codes. This work is supported by the National Natural Science Foundation of China (Nos. 11771225, 61771265, 71771127), the Humanities and Social Science Foundation of the Ministry of Education of China (No. 18YJCZH274), the Science and Technology Project of Nantong City (No. JC2018142) and the '226' Talent Scientific Research Project of Nantong City.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. J. Rohn, A theorem of the alternatives for the equation $Ax + B|x| = b$, *Linear Multilinear Algebra*, **52** (2004), 421–426.
2. O. L. Mangasarian, Absolute value programming, *Comput. Optim. Appl.*, **36** (2007), 43–53.
3. 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.
4. Z. Z. Bai, Modulus-based matrix splitting iteration methods for linear complementarity problems, *Numer. Linear Algebra Appl.*, **17** (2010), 917–933.
5. R. W. Cottle, J. S. Pang, R. E. Stone, *The linear complementarity problem*, SIAM: Philadelphia, 2009.
6. O. L. Mangasarian, R. R. Meyer, Absolute value equations, *Linear Algebra Appl.*, **419** (2006), 359–367.
7. U. Sch*ä*fur, On the modulus algorithm for the linear complementarity problem, *Oper. Res. Lett.*, **32** (2004), 350–354.
8. Z. Z. Bai, L. L. Zhang, Modulus-based synchronous multisplitting iteration methods for linear complementarity problems, *Numer. Linear Algebra Appl.*, **20** (2013), 425–439.
9. N. Zheng, J. F. Yin, Accelerated modulus-based matrix splitting iteration methods for linear complementarity problem, *Numer. Algor.*, **64** (2013), 245–262.
10. J. Rohn, On unique solvability of the absolute value equation, *Optim. Lett.*, **3** (2009), 603–606.
11. S. L. Wu, C. X. Li, The unique solution of the absolute value equations, *Appl. Math. Lett.*, **76** (2018), 195–200.
12. S. L. Wu, C. X. Li, A note on unique solvability of the absolute value equation, *Optim. Lett.*, **14** (2020), 1957–1960.
13. 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.
14. S. L. Wu, P. Guo, On the unique solvability of the absolute value equation, *J. Optim. Theory Appl.*, **169** (2016), 705–712.

15. O. L. Mangasarian, Absolute value equation solution via concave minimization, *Optim. Lett.*, **1** (2007), 3–8.

16. O. L. Mangasarian, Linear complementarity as absolute value equation solution, *Optim. Lett.*, **8** (2014), 1529–1534.

17. J. Rohn, An algorithm for solving the absolute value equation, *Electron. J. Linear Algebra*, **18** (2009), 589–599.

18. O. A. Prokopyev, On equivalent reformulations for absolute value equations, *Comput. Optim. Appl.*, **44** (2009), 363–372.

19. O. L. Mangasarian, A hybrid algorithm for solving the absolute value equation, *Optim. Lett.*, **9** (2015), 1469–1474.

20. C. X. Li, A preconditioned AOR iterative method for the absolute value equations, *Int. J. Comput. Meth.*, **14** (2017), 1750016.

21. D. K. Salkuyeh, The Picard-HSS iteration method for absolute value equations, *Optim. Lett.*, **8** (2016), 2191–2202.

22. O. L. Mangasarian, A generalized Newton method for absolute value equations, *Optim. Lett.*, **3** (2009), 101–108.

23. 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.

24. 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.

25. 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** (2011), 1490–1501.

26. L. Caccetta, B. Qu, G. L. Zhou, A globally and quadratically convergent method for absolute value equations, *Comput. Optim. Appl.*, **48** (2011), 45–58.

27. Y. Y. Lian, C. X. Li, S. L. Wu, Weaker convergent results of the generalized Newton method for the generalized absolute value equations, *J. Comput. Appl. Math.*, **338** (2018), 221–226.

28. N. Zainali, T. Lotfi, On developing a stable and quadratic convergent method for solving absolute value equation, *J. Comput. Appl. Math.*, **330** (2018), 742–747.

29. F. K. Haghani, On generalized Traub's method for absolute value equations, *J. Optim. Theory Appl.*, **166** (2015), 619–625.

30. C. X. Li, A modified generalized Newton method for absolute value equations, *J. Optim. Theory Appl.*, **170** (2016), 1055–1059.

31. J. Y. Bello Cruz, O. P. Ferreira, L. F. Prudente, On the global convergence of the inexact semi-smooth Newton method for absolute value equation, *Comput. Optim. Appl.*, **65** (2016), 93–108.

32. J. M. Feng, S. Y. Liu, A new two-step iterative method for solving absolute value equations, *J. Inequal. Appl.*, **2019** (2019), 39.

33. G. H. Golub, C. F. Van Loan, *Matrix computations*, 3 Eds., Maryland: The Johns Hopkins University Press, 2009.

34. G. Ramadurai, S. V. Ukkusuri, J. Y. Zhao, J. S. Pang, Linear complementarity formulation for single bottleneck model with heterogeneous commuters, *Transport. Res. Part B*, **44** (2010), 193–214.

35. R. Li, J. F. Yin, On the convergence of modulus-based matrix splitting iteration methods for a class of nonlinear complementarity problems with H_+-matrices, *J. Comput. Appl. Math.*, **342** (2018), 202–209.

36. A. Wang, Y. Cao, Q. Shi, Convergence analysis of modulus-based matrix splitting iterative methods for implicit complementarity problems, *J. Inequal. Appl.*, **2018** (2018), 2.

37. Y. Cao, A. Wang, Two-step modulus-based matrix splitting iteration methods for implicit complementarity problems, *Numer. Algor.*, **82** (2019), 1377–1394.

38. S. L. Wu, P. Guo, Modulus-based matrix splitting algorithms for quasi-complementarity problems, *Appl. Numer. Math.*, **132** (2018), 127–137.

39. Q. Shi, Q. Q. Shen, T. P. Tang, A class of two-step modulus-basedmatrix splitting iteration methods for quasi-complementarity problems, *Comput. Appl. Math.*, **39** (2020), 11.