Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 20, No. 1, pp. 23–44, 2024

AN INEXACT RELAXED GENERALIZED NEWTON ITERATIVE METHOD FOR SOLVING GENERALIZED ABSOLUTE VALUE EQUATIONS*

Dongmei Yu, Yiming Zhang† and Yifei Yuan

Abstract: In this paper, for solving the generalized absolute value equations (GAVE), an inexact relaxed generalized Newton (IRGN) iterative method is developed, which can adopt a relative error tolerance. Linear convergence of the IRGN iterative method is established under suitable conditions, and theoretical analysis of the inexact schemes support the efficient computational implementations of the exact schemes. It has been found that the IRGN iterative method involves the classical generalized Newton (GN) iterative method as a special case. Some numerical results are given to demonstrate the viability and robustness of the proposed methods.

Key words: *generalized absolute value equations, generalized Newton method, relaxed, inexact, convergence*

Mathematics Subject Classification: *65F10, 65H10, 90C30*

1 Introduction

In this paper, we consider the generalized absolute value equations (GAVE) of the form

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

where $A, B \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n$ are given, $x \in \mathbb{R}^n$ is an unknown vector that to be determined, and $|x| = (|x_1|, |x_2|, \ldots, |x_n|)^\top$ with x_l being the l-th entry of x. The GAVE (1.1) was first introduced in [32] and further investigated in a more general context [12, 21, 29, 38] and references therein. In the special case $B = -I$ or B is nonsingular, the GAVE (1.1) can be reduced to the absolute value equations (AVE)

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

The GAVE (1.1) and the AVE (1.2) arise in many areas of scientific computing and engineering applications, such as linear programming problem, mixed integer programming problem, bimatrix games, complementarity problem, quadratic programming problem, interval linear

*This work was supported partially by the National Natural Science Foundation of China (12201275), the Ministry of Education in China of Humanities and Social Science Project (21YJCZH204) and the Liaoning Provincial Department of Education (JYTZD2023072).

†Corresponding author.

DOI: https://doi.org/10.61208/pjo-2023-016

equations and many other applications, see, e.g., [23, 27–29, 31] and references therein. Furthermore, if $B = 0$, then the GAVE (1.1) reduces to a system of linear equations $Ax = b$ which has general applications in scientific computation.

In recent years, much more attention has been paid to the GAVE (1.1) and the AVE (1.2). On one hand, many researchers have concentrated on formulating conditions that ensure the existence, nonexistence and the uniqueness of solutions of GAVE (1.1) or of the type AVE (1.2), see [19, 21, 26, 36–38] and references therein. Particularly, one of the known sufficient conditions for the GAVE (1.1) is described in Lemma 1.1.

Lemma 1.1 ([37]). *If matrices A and B satisfy $\sigma_{max}(B) < \sigma_{min}(A)$, then the GAVE* (1.1) *has a unique solution for any b.*

It is worth mentioning that the well-known linear complementarity problem (LCP), the generalized LCP (GLCP) and the horizontal LCP (HLCP) can be transformed into the GAVE (1.1) or the AVE (1.2) which owns a very special and simple structure, and vice versa [19, 21]. On the other hand, to efficiently solve the GAVE (1.1) and the AVE (1.2), many numerical methods have been developed, such as the Newton-type methods [5, 17, 18, 22, 39, 40], the neural network approaches (also known as dynamic models in the literature) [2, 6, 24, 25], SOR-like iteration methods [8, 14, 16], the concave minimization methods [1, 20, 44], the Gauss–Seidel iteration method [9] and so forth. More versions of numerical iteration methods for solving the GAVE (1.1) and the AVE (1.2) include, but are not limited to, those in [4, 42, 43, 45].

The GAVE (1.1) can be viewed as the special case of the following system of nonlinear equations

$$F(x) = 0 \quad \text{with} \quad F(x) := Ax + B|x| - b. \tag{1.3}$$

With regard to the nonlinear equations (1.3), the efficient solver calling to mind is the well-known Newton method

$$F(x^k) + F'(x^k)(x^{k+1} - x^k) = 0, \quad k = 0, 1, 2, \ldots, \tag{1.4}$$

which can be used provided that the Jacobian matrix $F'(x)$ of $F(x)$ exists and is invertible. However, the Newton iterative method (1.4) can not be used directly to solve the GAVE (1.1) on account of the fact that the non-differentiable term $B|x|$ in (1.3). To address the problem, some smoothing Newton-type methods have been investigated for solving the GAVE (1.1). For example, Jiang and Zhang [15] proposed a smoothing–type algorithm for the GAVE (1.1). Saheya et al. [34] recast the GAVE (1.1) as a system of nonsmooth equations and proposed four new smoothing functions along with a smoothing–type algorithm to solve the GAVE (1.1). Tang and Zhou [35] proposed a quadratically convergent descent method for the GAVE (1.1). Mangasarian [22] utilized the generalized Jacobian $\partial|x|$ of $|x|$ based on a subgradient of its components and directly proposed the following generalized Newton (GN) iterative method to solve the AVE (1.2)

$$x^{k+1} = [A - D(x^k)]^{-1}b, \quad k = 0, 1, 2, \ldots, \tag{1.5}$$

where $D(x^k) \doteq \mathbf{diag}(\mathbf{sign}(x^k))$ with $\mathbf{sign}(x)$ denotes a vector with components equal to $-1, 0$ or 1, respectively, depending on whether the corresponding component of the vector x is negative, zero or positive. Whereafter, Hu et al. [13] extended the GN iteration scheme (1.5) to solve the GAVE (1.1), then the GN iterative scheme becomes

$$x^{k+1} = [A + BD(x^k)]^{-1}b, \quad k = 0, 1, 2, \ldots. \tag{1.6}$$

Under appropriate hypotheses, the convergence theory of the exact GN iterative method was established which confirmed that the computational efficiency overmatches the successive linearization method by some numerical experiments in [21]. Furthermore, some effective improvements on the GN iterative method have been presented, such as the modified GN iterative method [17, 18], the inexact semi-smooth Newton iterative method [5]. However, from the GN iterative scheme (1.5) or (1.6), 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 difficult especially if the coefficient matrix is ill-conditioned. In addition, if the generalized Jacobian matrix is singular, then the GN iterative method fails. To address these shortcomings and further accelerate the convergence of the GN iterative method, we present a new version of the GN iterative method in this paper, which is named as the relaxed generalized Newton (RGN) iterative method, by introducing a relaxation matrix for solving the GAVE (1.1). Also, the inexact version of the RGN (IRGN) iterative method is investigated for solving the GAVE (1.1). We discuss the convergence properties of the presented method in detail and give some convergence conditions under suitable assumptions. Numerical experiments are given to illustrate the performance and effectiveness of the proposed iterative methods.

This paper is structured as follows. Section 2 is devoted to the development of the RGN and the IRGN iterative methods for solving the GAVE (1.1). In Section 3, the linear convergence of the IRGN iterative method is explored. Section 4 reports the numerical results. Finally, some concluding remarks are given in Section 5.

At the end of this section, we present some notations which will be used throughout this paper. Let $\mathbb{R}^{n\times n}$ be the set of all $n\times n$ real matrices and $\mathbb{R}^n = \mathbb{R}^{n\times 1}$. I represents the identity matrix with suitable dimension. The transposition of a matrix or vector is denoted by $\cdot^\top$. For a vector $x = (x_1, x_2, \ldots, x_n)^\top \in \mathbb{R}^n$, x_i refers to its i-th entry, $|x|$ is in $\mathbb{R}^n$ with its i-th entry $|x_i|$, and $|\cdot|$ denotes the absolute value for real scalar. For $x \in \mathbb{R}^n$, $\|x\|$ denotes its 2-norm and $\mathbf{diag}(x)$ indicates a diagonal matrix with x_i as its diagonal entries for every $i = 1, 2, \ldots, n$. We use $\mathbf{tridiag}(a, b, c)$ to denote a matrix that has a, b, c as the subdiagonal, main diagonal and superdiagonal entries, respectively. $\mathbf{Tridiag}(A, B, C)$ denotes a block tridiagonal matrix that has A, B, C as the subdiagonal, main diagonal and superdiagonal block entries in the matrix, respectively. By default, $\|A\|$ denotes the spectral norm of A and is defined by the formula $\|A\| \doteq \max\{\|Ax\| : x \in \mathbb{R}^n, \|x\| = 1\}$. $\sigma_{min}(\cdot)$ and $\sigma_{max}(\cdot)$ denotes the minimum and the maximum singular value of the matrix, respectively. $\rho(\cdot)$ denotes the spectral radius of the matrix.

2 The RGN and IRGN Iterative Methods

In this section, we will describe our methods for solving the GAVE (1.1) and prove their convergence.

Let $\Omega \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix, based on the Newton method (1.4), a relaxed generalized Newton iterative scheme is developed to solve the GAVE (1.1)

$$F(x^k) + (F'(x^k) + \Omega)(x^{k+1} - x^k) = 0, \quad k = 0, 1, 2, \ldots. \tag{2.1}$$

Substituting (1.3) and the generalized Jacobian $F'(x^k) = A + BD(x^k)$ into (2.1), the relaxed generalized Newton (RGN) iterative method is established and the detail is given in Algorithm 2.1.

Algorithm 2.1. (**The RGN iterative method**) Let $A, B \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Given a positive semi-definite matrix $\Omega \in \mathbb{R}^{n\times n}$ such that $A + \Omega + BD(x^k)$ is invertible. Assume

that $x^0 \in \mathbb{R}^n$ is an arbitrary initial guess, compute

$$x^{k+1} = [A + \Omega + BD(x^k)]^{-1}(\Omega x^k + b) \tag{2.2}$$

for $k = 0, 1, 2, \ldots$ until the iterative sequence $\{x^k\}$ is convergent.

From Algorithm 2.1, the introducing matrix Ω can be regarded as a role of relaxation, which can avoid the singularity and adjust the condition number of the coefficient matrix $A + \Omega + BD(x^k)$ so as to accelerate the convergence rate of the GN iterative method (1.6). In particular, if $\Omega = 0$, then the RGN iterative method (2.2) reduces to the GN iterative method (1.6). If $\Omega = I$ and $B = -I$, then the RGN iterative method (2.2) reduces to the modified GN (MGN) iterative method in [17]. The iteration sequence $\{x^k\}$ generated by Algorithm 2.1 has the following general convergence property.

Theorem 2.2. *Let $A, B \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$ and $\Omega \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix such that $A + \Omega + BD(x)$ is invertible for $x \in \mathbb{R}^n$. If A is invertible and*

$$\|A^{-1}\| < \frac{1}{2\|\Omega\| + 3\|B\|}, \tag{2.3}$$

then the iteration sequence $\{x^k\}$ generated by Algorithm 2.1 converges linearly from any starting point to a unique solution $x^* \in \mathbb{R}^n$ of the GAVE (1.1).

Proof. Let $x^* \in \mathbb{R}^n$ be a solution of the GAVE (1.1), then we have

$$[A + \Omega + BD(x^*)]x^* = \Omega x^* + b. \tag{2.4}$$

Subtracting (2.4) from (2.2), we obtain

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

Taking norms on both sides and using certain properties of vector norm, we obtain

$$\begin{aligned}\|x^{k+1} - x^*\| &\le \|[A + \Omega + BD(x^k)]^{-1}\|\|\Omega(x^k - x^*) + BD(x^k)(x^k - x^*) + B(|x^*| - |x^k|)\| \\ &\le \|[A + \Omega + BD(x^k)]^{-1}\|(\|\Omega\| + 2\|B\|)\|x^k - x^*\| \\ &< \|x^k - x^*\|,\end{aligned}$$

where the first inequality follows from $\||x| - |x^*|\| \le \|x - x^*\|$ and the last inequality holds from $\|[A + \Omega + BD(x^k)]^{-1}\|(\|\Omega\| + 2\|B\|) < 1$, that is

$$\|[A + \Omega + BD(x)]^{-1}\| < \frac{1}{\|\Omega\| + 2\|B\|}. \tag{2.5}$$

For each $x \in \mathbb{R}^n$, we get

$$A + \Omega + BD(x) = A[I + A^{-1}(\Omega + BD(x))],$$

from which we take norms on both sides of above equality and use certain properties of matrix norm, then

$$\begin{aligned}\|[A + \Omega + BD(x)]^{-1}\| &= \|[I + A^{-1}(\Omega + BD(x))]^{-1}A^{-1}\| \\ &\le \|[I + A^{-1}(\Omega + BD(x))]^{-1}\|\|A^{-1}\|.\end{aligned} \tag{2.6}$$

Based on the Banach perturbation [10, Lemma 2.3.3], since

$$\|A^{-1}(\Omega + BD(x))\| < \|A^{-1}\|\|\Omega + BD(x)\| < \|A^{-1}\|(\|\Omega\| + \|B\|) < 1,$$

which implies

$$\|A^{-1}\| < \frac{1}{\|\Omega\| + \|B\|},$$

it follows that

$$\begin{aligned}\|[I + A^{-1}(\Omega + BD(x))]^{-1}\| &\leq \frac{1}{1 - \|A^{-1}(\Omega + BD(x))\|} \\ &\leq \frac{1}{1 - \|A^{-1}\|(\|\Omega\| + \|B\|)}. \end{aligned} \tag{2.7}$$

According to (2.5), (2.6) and (2.7), a sufficient condition of (2.5) is that

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

which is satisfied if and only if (2.3) holds.

According to [41], since $\|A^{-1}\| = \sigma_{max}(A^{-1}) = \frac{1}{\sigma_{min}(A)}$ and $\|B\| = \sigma_{max}(B)$, then $\sigma_{min}(A) > \sigma_{max}(B)$ holds if (2.3) satisfies. It follows from Lemma 1.1 that the RGN iterative method converges linearly from any starting point to a unique solution x^* of the GAVE (1.1). □

Based on Algorithm 2.1, at each iteration step, the RGN iterative method requires the exact solution of the linear system with coefficient matrix $A + \Omega + BD(x)$ depends on x, which might be computationally expensive. To alleviate the burden of each step and further improve the computational efficiency, we propose an inexact version of Algorithm 2.1 for solving the GAVE (1.1).

Algorithm 2.3. (**The IRGN iterative method**) Let $A, B \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$. Given an initial guess $x^0 \in \mathbb{R}^n$, a residual relative error tolerance $\theta \in [0, 1)$ and a positive semi-definite matrix $\Omega \in \mathbb{R}^{n\times n}$ such that $A + \Omega + BD(x^k)$ is invertible, for $k = 0, 1, 2, \ldots$ until the iteration sequence $\{x^k\}$ is convergent, compute x^{k+1} in terms of (2.2), satisfying the relative residual error criteria

$$\|[A + \Omega + BD(x^k)]x^{k+1} - \Omega x^k - b\| \leq \theta\|F(x^k)\|. \tag{2.8}$$

It is noteworthy that, in absence of errors, i.e., $\theta = 0$, the IRGN iterative method (2.8) will retrieve the RGN iterative method (2.2). Consequently, Algorithm 2.3 also involves the inexact versions of the GN iterative method (1.6) and the MGN iterative method in [17] as special cases. In the next section, we will analyze the convergence properties of the iteration sequence $\{x^k\}$ generated by the IRGN iterative method.

3 Convergence Analysis

The aim of this section is to analyze the general convergence of the IRGN iterative method in the context of solving the GAVE (1.1).

Before establishing the convergence of the sequence $\{x^k\}$ generated by Algorithm 2.3, we define the concept of a family of maps according to (2.8) which is crucial in our study and explore their properties.

Definition 3.1. For $0 \leq \theta < 1$, $\mathcal{N}_\theta$ is the family of maps N_θ: $\mathbb{R}^n \to \mathbb{R}^n$ such that

$$\|[A + \Omega + BD(x)]N_\theta(x) - \Omega x - b\| \leq \theta\|F(x)\|, \forall x \in \mathbb{R}^n.$$

If $A + \Omega + BD(x)$ is invertible, then the family $\mathcal{N}_0$ has a single element for each $x \in \mathbb{R}^n$, that is to say, the RGN iteration map N_0: $\mathbb{R}^n \to \mathbb{R}^n$ defined by

$$N_0(x) = [A + \Omega + BD(x)]^{-1}(\Omega x + b).$$

From Definition 3.1, if $0 \leq \theta \leq \theta' < 1$ then $\mathcal{N}_0 \subseteq \mathcal{N}_\theta \subseteq \mathcal{N}_{\theta'}$, where θ' is the upper bound of θ. This means that $\theta'\|F(x)\|$ is the maximum relative error that we can tolerate. Hence $\mathcal{N}_\theta$ is non-empty for all $0 \leq \theta < 1$. For any $0 < \theta < 1$ and $N_\theta \in \mathcal{N}_\theta$, $N_\theta(x) = x$ if and only if $F(x) = 0$.

This indicates that the fixed points of the IRGN iteration map N_θ are the same as fixed points of the exact RGN iteration map.

From Algorithm 2.3 and Definition 3.1, the outcome of the IRGN iterative method is

$$x^{k+1} = N_\theta(x^k), \quad k = 0, 1, 2, \ldots \tag{3.1}$$

with $N_\theta \in \mathcal{N}_\theta$ and $\theta \in [0, 1)$. The following lemma lays the foundation of the convergence analysis hereinafter, which is inspired by [5].

Lemma 3.2. *Suppose that $A+\Omega+BD(x)$ is invertible for each $x \in \mathbb{R}^n$. Let $0 \leq \theta < 1$ and $N_\theta \in \mathcal{N}_\theta$. If $F(x^*) = 0$ then for any $x \in \mathbb{R}^n$ satisfying*

$$\begin{aligned}\|N_\theta(x) - x^*\| \leq &\|[A + \Omega + BD(x)]^{-1}\|[\theta(\|A + \Omega + BD(x)\| \\ &+ 2\|B\| + \|\Omega\|) + 2\|B\| + \|\Omega\|]\|x - x^*\|.\end{aligned} \tag{3.2}$$

Proof. Let $x \in \mathbb{R}^n$. In the light of $F(x^*) = 0$ and the fact that $[A + \Omega + BD(x)]x = F(x) + (\Omega x + b)$, we can derive that

$$\begin{aligned}N_\theta(x) - x^* = &[A + \Omega + BD(x)]^{-1}([A + \Omega + BD(x)]N_\theta(x) - (\Omega x + b) \\ &+ [F(x^*) - F(x) + [A + \Omega + BD(x)](x - x^*)]).\end{aligned}$$

Taking norms on both sides and utilizing the triangle inequality, one can obtain

$$\begin{aligned}\|N_\theta(x) - x^*\| \leq &\|[A + \Omega + BD(x)]^{-1}\|(\|[A + \Omega + BD(x)]N_\theta(x) - (\Omega x + b)\| \\ &+ \|F(x^*) - F(x) + [A + \Omega + BD(x)](x - x^*)\|).\end{aligned} \tag{3.3}$$

In view of Definition 3.1, then it follows from (3.3) that

$$\|N_\theta(x) - x^*\| \leq \|[A+\Omega+BD(x)]^{-1}\|(\theta\|F(x)\| + \|F(x^*) - F(x) + [A+\Omega+BD(x)](x-x^*)\|). \tag{3.4}$$

On the other hand, note that $F(x^*) = 0$ implies that

$$F(x) = (A + \Omega + BD(x))(x - x^*) - [F(x^*) - F(x) - [A + \Omega + BD(x)](x^* - x)],$$

from which we obtain

$$\|F(x)\| \leq \|A + \Omega + BD(x)\|\|x - x^*\| + \|F(x^*) - F(x) - [A + \Omega + BD(x)](x^* - x)\|. \tag{3.5}$$

Furthermore, by some calculations, it holds that

$$F(x^*) - F(x) - [A + \Omega + BD(x)](x^* - x) = B(|x^*| - |x|) - BD(x)(x^* - x) - \Omega(x^* - x),$$

then it can be concluded that

$$\|F(x^*) - F(x) - [A + \Omega + BD(x)](x^* - x)\| \leq (2\|B\| + \|\Omega\|)\|x^* - x\|, \tag{3.6}$$

where $\||x| - |x^*|\| \leq \|x - x^*\|$ is utilized.

Combining (3.5) and (3.6), we get

$$\|F(x)\| \leq (\|A + \Omega + BD(x)\| + 2\|B\| + \|\Omega\|)\|x - x^*\|. \tag{3.7}$$

Substituting the inequalities (3.6) and (3.7) into (3.4), one obtains (3.2), thus achieving the desired assertion immediately. □

Now, we are in position to prove the main results of this section.

Theorem 3.3. *Let $A, B \in \mathbb{R}^{n\times n}$, $\Omega \in \mathbb{R}^{n\times n}$ be a positive semi-definite matrix, $b \in \mathbb{R}^n$. Assume that $A+\Omega+BD(x)$ is invertible for all $x \in \mathbb{R}^n$. Then, every sequence $\{x^k\}$ generated by Algorithm 2.3 starting at $x_0 \in \mathbb{R}^n$ with the residual relative error tolerance $0 \leq \theta < 1$. Additionally, for any $x \in \mathbb{R}^n$, if A is invertible and*

$$\|A^{-1}\| < \frac{1}{\theta(\|A\| + 3\|B\| + 2\|\Omega\|) + 3\|B\| + 2\|\Omega\|}, \tag{3.8}$$

then the sequence $\{x^k\}$ converges linearly to $x^* \in \mathbb{R}^n$, the unique solution of the GAVE (1.1).

Proof. For any starting point $x^0 \in \mathbb{R}^n$, according to Definition 3.1 and Algorithm 2.3, the well-defineness of $\{x^k\}$ follows from invertibility of $A+\Omega+BD(x)$. In the light of (3.1) and (3.2), since x^* is the solution of the GAVE (1.1), together with $F(x^*) = 0$, we conclude that for $k = 1, 2, \ldots$, the sequence $\{x^k\}$ satisfies

$$\begin{aligned}\|x^{k+1} - x^*\| \leq &\|[A + \Omega + BD(x^k)]^{-1}\|(\theta(\|A + \Omega + BD(x^k)\| \\ &+ 2\|B\| + \|\Omega\|) + 2\|B\| + \|\Omega\|)\|x^k - x^*\|.\end{aligned} \tag{3.9}$$

Taking (3.9) into account, we obtain

$$\|[A + \Omega + BD(x^k)]^{-1}\|(\theta(\|A + \Omega + BD(x^k)\| + 2\|B\| + \|\Omega\|) + 2\|B\| + \|\Omega\|) < 1, \tag{3.10}$$

which is a sufficient condition that the sequence $\{x^k\}$ converges linearly to x^*.

Similarly to the proof of the Theorem 2.2, combining (2.6), (2.7), and (3.2), we can deduce that

$$\begin{aligned}\|N_\theta(x^k) - x^*\| \leq &\|[A + \Omega + BD(x^k)]^{-1}\|[\theta(\|A + \Omega + BD(x^k)\| + 2\|B\| + \|\Omega\|) \\ &+2\|B\| + \|\Omega\|]\|x^k - x^*\| \\ \leq &\|[I + A^{-1}(\Omega + BD(x^k))]^{-1}\|\|A^{-1}\|[\theta(\|A\| + \|\Omega\| + \|B\| + 2\|B\| + \|\Omega\|) \\ &+2\|B\| + \|\Omega\|]\|x^k - x^*\| \\ \leq &\frac{\|A^{-1}\|}{1 - \|A^{-1}\|(\|\Omega\| + \|B\|)}[\theta(\|A\| + 3\|B\| + 2\|\Omega\|) + 2\|B\| + \|\Omega\|]\|x^k - x^*\|.\end{aligned}$$

Therefore, a sufficient condition of (3.10) is

$$\frac{\|A^{-1}\|}{1 - \|A^{-1}\|(\|\Omega\| + \|B\|)}[\theta(\|A\| + 3\|B\| + 2\|\Omega\|) + 2\|B\| + \|\Omega\|] < 1,$$

which is equivalent to (3.8), and (3.8) is also a sufficient condition of $\sigma_{min}(A) > \sigma_{max}(B)$. It follows from Lemma 1.1 that the IRGN iterative method converges linearly from any starting point to a unique solution x^* of the GAVE (1.1). □

Remark 3.4. If $\theta = 0$, (3.8) reduces to (2.3), a sufficient convergence condition for the RGN iterative method. Moreover, according to (3.8), we get

$$0 \leq \theta < \frac{1 - \|A^{-1}\|(3\|B\| + 2\|\Omega\|)}{\|A^{-1}\|(\|A\| + 3\|B\| + 2\|\Omega\|)}. \tag{3.11}$$

When $\Omega = 0$ and $B = -I$, (3.11) reduces to $0 \leq \theta < \frac{1-3\|A^{-1}\|}{\|A^{-1}\|(\|A\|+3)}$, when $\Omega = 0$, $B = -I$ and $\theta = 0$, (3.8) reduces to $\|A^{-1}\| < \frac{1}{3}$, then Theorem 3.3 reduces to [5, Theorem 2] for the AVE (1.2). When $\Omega = I$ and $B = -I$, (2.5) reduces to $\|(A + I - D)^{-1}\| < \frac{1}{3}$, which is the result of [17, Theorem 3.1] for the AVE (1.2).

Remark 3.5. For the relaxation matrix Ω involved in the RGN and IRGN methods, we give some parameter choices for the GAVE (1.1) being uniquely solvable. When $\theta \neq 0$, according to Theorem 3.3, we know (3.8) herein has an implicit condition of $\|\Omega\| > \frac{1}{(2\theta+2)\|A^{-1}\|} - \frac{3}{2}\|B\| - \frac{\theta}{2\theta+2}\|A\|$. Similarly, when $\theta = 0$, i.e. the RGN method, we have $\|\Omega\| > \frac{1}{2\|A^{-1}\|} - \frac{3}{2}\|B\|$.

Remark 3.6. Theoretically, according to (3.11), $\frac{1-\|A^{-1}\|(3\|B\|+2\|\Omega\|)}{\|A^{-1}\|(\|A\|+3\|B\|+2\|\Omega\|)}$ which is the upper bound of θ, is generally expensive to compute or hard to estimate. During the practice, based on this theoretical guidance,

$$\theta_k = min\{0.9, \frac{1}{max\{1, k - l_{max}\}}\} \tag{3.12}$$

with $l_{max} = 10$ is utilized in the next section. Here, k counts the number of outer iteration steps.

Finally, we will go deeper in the choice of the relaxation matrix Ω. If the setting of Ω is too complicated (for instance, if Ω is dense or ill-conditioned), it will further increase the cost of each iterative step. One way to choose Ω is let it be a diagonal matrix, especially be a scalar matrix, that is, let $\Omega = \omega I(\omega \geq 0)$. In this case, we consider the convergence conditions of the IRGN iterative method for solving the GAVE (1.1) and the AVE (1.2), which also provides the existence of the relaxation matrix Ω under some mild conditions.

Theorem 3.7. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix, $B \in \mathbb{R}^{n\times n}$, $\Omega = \omega I(\omega \geq 0)$ such that $A + \Omega + BD(x)$ is invertible for $x \in \mathbb{R}^n$. Let $\lambda_{min}(A)$ and $\lambda_{max}(A)$ be the minimum and the maximum eigenvalues of the matrix A, respectively, and $\varrho = \|B\|$. If*

$$\theta\lambda_{max}(A) + 3(\theta + 1)\varrho + 2(\theta + 1)\omega < \lambda_{min}(A), \tag{3.13}$$

then the IRGN iterative method converges linearly from any starting point to a unique solution $x^* \in \mathbb{R}^n$ of the GAVE (1.1).

Proof. According to [41], we get

$$\varrho = \sup_{x\neq 0} \frac{\|Bx\|}{\|x\|} = \max_{\|x\|=1} \|Bx\| = \sigma_{max}(B).$$

Since A is a symmetric positive definite matrix, we get $\lambda_{min}(A) = \sigma_{min}(A)$, then (3.13) is obtained from (3.8), which guarantees the uniqueness of solution.

□

Corollary 3.8. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix, $\Omega = \omega I(\omega \geq 0)$ such that $A+\Omega - D(x)$ is invertible for $x \in \mathbb{R}^n$. Let $\lambda_{min}(A)$ and $\lambda_{max}(A)$ be the minimum and the maximum eigenvalues of the matrix A, respectively. If*

$$\theta\lambda_{max}(A) + 3\theta + 3 + 2(\theta+1)\omega < \lambda_{min}(A),$$

then the IRGN iterative method converges linearly from any starting point to a unique solution $x^* \in \mathbb{R}^n$ of the AVE (1.2).

4 Numerical Results

In this section, we will report some concrete examples for solving the GAVE (1.1) to illustrate the performance of the suggested methods. We compare our methods with some existing algorithms which indicate that the proposed methods are very promising. Concretely, the following six algorithms will be tested.

1. GN: Mangasarian's generalized Newton method for the AVE (1.2) [22]. Given an initial guess $x^0 \in \mathbb{R}^n$, for $k = 0, 1, 2 \ldots$ until the iteration sequence $\{x^k\}$ is convergent, compute

$$\left[A - D(x^k)\right] x^{k+1} = b. \tag{4.1}$$

   For the GAVE (1.1), the iterative scheme (4.1) becomes [39]

$$\left[A + BD(x^k)\right] x^{k+1} = b.$$

   It should be mentioned that a generalized Newton method is proposed for solving the GAVE associated with second–order cone in [13]. In addition, the sufficient condition that the GAVE (1.1) has a unique solution generalizes to a sufficient condition that $\|A^{-1}\| < \frac{1}{3\|B\|}$ and $\|B\| > 1$.

2. MGN: the modified generalized Newton method for the AVE (1.2) [17]. Given an initial guess $x^0 \in \mathbb{R}^n$, find some step x^{k+1} satisfying

$$\left[A + I - D(x^k)\right] x^{k+1} = x^k + b \tag{4.2}$$

   for $k = 0, 1, 2, \ldots$ until convergence. For the GAVE (1.1), the iteration scheme (4.2) changes to [39]

$$\left[A + I + BD(x^k)\right] x^{k+1} = x^k + b.$$

3. RGN: the exact relaxed generalized Newton iterative method, i.e., Algorithm 2.1.

4. Picard: the Picard iteration method for the GAVE (1.1) [33]. Given an initial guess $x^0 \in \mathbb{R}^n$, for $k = 0, 1, 2, \ldots$ until convergence, compute

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

5. IGN: the inexact semi-smooth Newton method for the AVE (1.2) [5]. Given an initial guess $x^0 \in \mathbb{R}^n$, if $F(x^k) \neq 0$, the iteration sequence $\{x^k\}$ is generated by

$$\left[A - D(x^k)\right] x^{k+1} = b + r_k \quad \text{with} \quad \|r_k\| \leq \theta_k \left\|Ax^k - |x^k| - b\right\| \tag{4.3}$$

   for $k = 0, 1, 2, \ldots$ until convergence. Here, $\theta_k \geq 0$ is the residual relative error tolerance. For the GAVE (1.1), (4.3) is slightly modified as

$$\left[A - BD(x^k)\right] x^{k+1} = b + r_k \quad \text{with} \quad \|r_k\| \leq \theta_k \left\|F(x^k)\right\|.$$

Here, $\theta_k \geq 0$ is the residual relative error tolerance, which derives the generalized upper bound calculated from [5], and is defined as

$$\theta_k = 0.999 \cdot \frac{1 - 3\|A^{-1}\|\|B\|}{\|A^{-1}\|(\|A\| + 3\|B\|)}.$$

6. IRGN: the inexact relaxed generalized Newton iterative method, i.e., Algorithm 2.3, where θ_k is calculated according to (3.12).

The numerical experiments are explained in several aspects in the following. In this paper, considering that the matrix $\Omega = \omega I$ is a scalar matrix, the choice of parameters ω are particularly important, which greatly affects the CPU time of numerical experiments. In order to facilitate the comparison of algorithms, the parameter ω in this paper is the optimal parameter ω_{exp} obtained directly through experiments, which leads to the smallest number of iteration steps of the IRGN iterative method, herein $\omega = [0.01 : 0.01 : 2]$ in Example 4.1 and Example 4.2, $\omega = [0.05 : 0.05 : 1]$ in Example 4.3.

On the other hand, compared with the iteration schemes of the GN, MGN, RGN and Picard methods, each step of these methods requires the exact solution of a system of linear equations with $A + BD(x^k)$, $A + I + BD(x^k)$, $A + \omega I + BD(x^k)$ and A being the coefficient matrix, respectively. It is proposed in [22] that the algorithm can be further optimized by the sparse LU factorization in combination with column AMD reordering if all coefficient matrices are not guaranteed to be positive. In addition, for the inexact iterative methods, the approximate solution of the system of linear equation is also obtained with $A + BD(x^k)$ and $A + \omega I + BD(x^k)$ being the coefficient matrix of the IGN and IRGN methods, respectively. The CG [11] or LSQR [30] algorithm is utilized to solve the equations approximately in [22].

All test problems are conducted under MATLAB R2016a on a personal computer with 1.19 GHz central processing unit (Intel(R) Core(TM) i5-1035U), 8.00 GB memory and Windows 10 operating system. "IT" denotes the number of iterations, "CPU" denotes the elapsed CPU time in seconds and "RES" denotes the residual relative error. In actual computations, we use $(1, 0, 1, 0, \ldots)^\top \in \mathbb{R}^n$ as the initial vector x^0 in Example 4.1 and Example 4.2, and $(0, 0, \ldots, 0, 0,)^\top \in \mathbb{R}^n$ as the initial vector x^0 in Example 4.3. The stopping criterion is set to be

$$RES(x^k) \doteq \frac{\|Ax^k + B|x^k| - b\|_2}{\|b\|_2} < 10^{-7}$$

or up to the preset maximum number of outer iteration steps $k_{max} = 500$ (the symbol "–" is utilized in the following tables to illustrate this case). In order to compare the superiority of algorithms for solving the GAVE, and to show that the algorithms solving the GAVE can also be applied to LCP, we choose the equivalent problem (i.e. LCP) of the GAVE for comparison in Example 4.1 and Example 4.2.

Example 4.1 ([3])**.** This example comes from the following LCP. Given a matrix $M \in \mathbb{R}^{n\times n}$ and a vector $q \in \mathbb{R}^n$, finding a pair of real vectors $z \in \mathbb{R}^n$ and $w \in \mathbb{R}^n$ such that

$$z \geq 0,\ w = Mz + q \geq 0 \text{ and } z^\top w = 0, \tag{4.4}$$

where $M = \hat{M} + \mu I \in \mathbb{R}^{n\times n}$ and $q = -Mz^*$. Here

$$\hat{M} = \mathbf{Tridiag}(-I_m, S_m, -I_m) \in \mathbb{R}^{n\times n},\ S_m = \mathbf{tridiag}(-1, 4, -1) \in \mathbb{R}^{m\times m},$$

$z^* = (1.2, 0, 1.2, 0, \ldots, 1.2, 0, \ldots)^\top \in \mathbb{R}^n$ with $n = m^2$. From [19, 21], LCP (4.4) can be reduced to

$$(M + I)x + (I - M)|x| = q \text{ with } x = \frac{1}{2}[(M - I)z + q],$$

which belongs to the GAVE (1.1). In particular, $A = M + I$, $B = I - M$ and $b = q$. In this paper, $\mu = 4, -4$ are considered. The parameter μ is chosen to ensure that the iteration coefficient matrix is positive definite and then the inverse decomposition is changed by LU factorization.

When $\mu = 4$, A and M are symmetric positive definite matrices. Therefore, the inexact methods all use CG method to solve the internal linear system. The numerical results are shown in Table 1, from which we find that the number of iteration steps of the GN method is the smallest. Although the number of iteration steps of the IGN method is larger than that of the GN method, the CPU time is reduced to some extent. Compared with the MGN method, the RGN method increases the optimal selection of parameters, so to a certain extent, it not only reduces the number of iterations, but also reduces the CPU time. It shows that the RGN method is superior to the MGN method. Although the IRGN method has more iterations than the RGN method, but the CPU time is greatly reduced, and the IRGN method is better than other tested methods in terms of CPU time. In conclusion, under our setting, the IRGN method is a competitive method for solving the GAVE (1.1).

Table 1: Numerical results for Example 4.1 with $\mu = 4$

Method		m				
		60	70	80	90	100
	ω_{exp}	0.21	0.22	0.28	0.28	0.26
GN	IT	2	2	2	2	2
	CPU	0.0182	0.0282	0.0474	0.0950	0.0971
	RES	1.6100×10^{-16}	1.6728×10^{-16}	1.7062×10^{-16}	1.6045×10^{-16}	1.5968×10^{-16}
MGN	IT	9	9	9	9	9
	CPU	0.1114	0.1819	0.2537	0.5530	0.4864
	RES	1.9008×10^{-8}	1.9127×10^{-8}	1.9217×10^{-8}	1.9287×10^{-8}	1.9343×10^{-8}
RGN	IT	6	6	6	4	4
	CPU	0.0663	0.1100	0.1490	0.1643	0.1506
	RES	2.3984×10^{-8}	2.9859×10^{-8}	8.9727×10^{-8}	5.5920×10^{-9}	5.5981×10^{-9}
Picard	IT	57	56	56	56	56
	CPU	0.0339	0.0597	0.0845	0.1556	0.1516
	RES	8.4279×10^{-8}	9.7218×10^{-8}	9.0805×10^{-8}	8.5514×10^{-8}	8.1051×10^{-8}
IGN	IT	20	18	18	17	16
	CPU	0.0094	0.0114	0.0174	0.0341	0.0283
	RES	3.9137×10^{-9}	4.1742×10^{-8}	6.9371×10^{-8}	6.6205×10^{-8}	3.8782×10^{-8}
IRGN	IT	15	13	14	15	15
	CPU	**0.0084**	**0.0110**	**0.0157**	**0.0332**	**0.0259**
	RES	5.5863×10^{-8}	6.0704×10^{-8}	7.3012×10^{-8}	2.5509×10^{-8}	3.7447×10^{-8}

When $\mu = -4$, A and M are symmetric indefinite matrices. Therefore, the LSQR method is used to solve the internal linear system for the inexact methods. The other programs are designed the same as $\mu = 4$. The numerical results are shown in Table 2, from which we can find that the GN, IGN and Picard methods are failed. Compared with the MGN method, the RGN method increases the optimal selection of parameters, which reduces not only the number of iteration steps but also the CPU time to a certain extent. It shows that the RGN

method is superior to the MGN method, and the IRGN method is better than other tested methods in terms of CPU time.

Table 2: Numerical results for Example 4.1 with $\mu = -4$

Method		m				
		60	70	80	90	100
	ω_{exp}	1.12	1.01	1.13	1.02	1.02
GN	IT	–	–	–	–	–
	CPU	–	–	–	–	–
	RES	–	–	–	–	–
MGN	IT	22	22	22	23	23
	CPU	0.0361	0.0342	0.0455	0.0606	0.0940
	RES	5.5470×10^{-8}	7.5103×10^{-8}	9.7704×10^{-8}	4.1090×10^{-8}	5.0601×10^{-8}
RGN	IT	20	21	20	20	20
	CPU	0.0347	0.0324	0.0418	0.0535	0.0810
	RES	4.3743×10^{-8}	4.2353×10^{-8}	4.5749×10^{-8}	8.1717×10^{-8}	8.2914×10^{-8}
Picard	IT	–	–	–	–	–
	CPU	–	–	–	–	–
	RES	–	–	–	–	–
IGN	IT	–	–	–	–	–
	CPU	–	–	–	–	–
	RES	–	–	–	–	–
IRGN	IT	17	18	17	18	18
	CPU	**0.0161**	**0.0208**	**0.0162**	**0.0278**	**0.0384**
	RES	9.8291×10^{-8}	8.5102×10^{-8}	9.9811×10^{-8}	7.0584×10^{-8}	7.5483×10^{-8}

Example 4.2 ([3])**.** Considering the LCP (4.4), where the matrix $\hat{M}$ is slightly different. Here

$$\hat{M} = \mathbf{Tridiag}(-1.5I_m, S_m, -0.5I_m) \in \mathbb{R}^{n\times n},\ S_m = \mathbf{tridiag}(-1.5, 4, -0.5) \in \mathbb{R}^{m\times m}$$

and $z^* = (1.2, 0, 1.2, 0, \ldots, 1.2, 0, \ldots)^\top \in \mathbb{R}^n$ with $n = m^2$. In this example, when $\mu = 4$, A and M are asymmetric positive definite matrices. Therefore, LU factorization can be used to assist the inverse computation and the inexact methods all use CG method to solve the internal linear system. The numerical results are shown in Table 3.

From Table 3, we see that the IRGN method is slightly superior to the IGN method in terms of CPU time and iteration steps. Compared with the MGN method, the RGN method increases the optimal selection of parameters, which reduces not only the number of iteration steps but also the CPU time to a certain extent. Although the IRGN method has more iterations than the RGN method, but the CPU time is greatly reduced, and the IRGN method is better than other tested methods in terms of CPU time. This shows that the selection of parameter in the iterative scheme has a good performance in the face of specific problems.

Example 4.1 and Example 4.2 show that, for the cases that A and M are special matrices (symmetric positive definite matrices, symmetric indefinite matrices and asymmetric positive definite matrices), the IRGN method has superior performance. But most of actual data matrices have not special forms. In the following, we generate randomly the data matrices by using MATLAB software **rand** function to confirm the advantages of the proposed methods under certain conditions. We will illustrate it by Example 4.3.

Table 3: Numerical results for Example 4.2 with $\mu = 4$

Method		m				
		60	70	80	90	100
	ω_{exp}	0.01	0.04	0.05	0.05	0.05
GN	IT	2	2	2	2	2
	CPU	0.0165	0.0252	0.0380	0.0564	0.1265
	RES	3.2158×10^{-16}	3.2012×10^{-16}	3.1775×10^{-16}	3.1653×10^{-16}	3.2214×10^{-16}
MGN	IT	9	9	9	9	9
	CPU	0.1062	0.1799	0.2481	0.3600	0.6084
	RES	1.9000×10^{-8}	1.9120×10^{-8}	1.9211×10^{-8}	1.9281×10^{-8}	1.9338×10^{-8}
RGN	IT	4	5	5	5	5
	CPU	0.0348	0.0825	0.1107	0.1603	0.2717
	RES	5.5755×10^{-9}	1.6767×10^{-9}	4.0332×10^{-9}	4.0414×10^{-9}	4.0480×10^{-9}
Picard	IT	59	58	58	58	57
	CPU	0.0361	0.0572	0.0860	0.1139	0.1980
	RES	8.4267×10^{-8}	9.6624×10^{-8}	9.0238×10^{-8}	8.4971×10^{-8}	9.9995×10^{-8}
IGN	IT	19	19	18	21	21
	CPU	0.0088	0.0112	0.0129	0.0212	0.0382
	RES	9.7574×10^{-8}	6.4348×10^{-8}	9.0976×10^{-8}	5.9690×10^{-8}	4.8267×10^{-8}
IRGN	IT	16	15	15	16	16
	CPU	**0.0087**	**0.0108**	**0.0125**	**0.0184**	**0.0352**
	RES	1.8022×10^{-8}	7.2566×10^{-8}	7.1436×10^{-8}	7.7978×10^{-8}	8.4821×10^{-8}

Example 4.3. This example belongs to the GAVE (1.1) with certain dimension. For each n we generate 100 GAVEs with $\|B\| > 1$ and $\|A^{-1}\| < \frac{1}{3\|B\|}$, herein guarantees the GN and IGN methods to converge to the unique solution of the GAVE (1.1). In this example the development generating the GAVE (1.1) parallels the recent work by [5] on the IGN method for solving the AVE (1.2). In programming, let $singval(\cdot)$ be the singular value vector of matrix, $singval(B) = \frac{singval}{rand \cdot min(singval)}$ and $singval(A) = \frac{3 \cdot singval \cdot \|B\|}{rand \cdot min(singval)}$ such that $\|B\| > 1$ and $\|A^{-1}\| < \frac{1}{3\|B\|}$. The influence of the condition number of A and B on the tests will be discussed during the numerical implements.

In order to assess the computational behaviour of the proposed methods, we plot the performance profile graphics [7], and the performance measurement is the running CPU time, namely, we use the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s} : s \in \mathcal{S}\}},$$

where $t_{p,s}$ defines as the mean of the CPU time by using method s to solve problem p five times, $\mathcal{S}$ and $\mathcal{P}$ are the set of corresponding solvers and problems, respectively. For the overall evaluation of the solvers, $\rho_s(\tau)$ is defined as the probability of the solver $s \in \mathcal{S}$, which is the best possible radio that a performance radio $r_{p,s}$ is within a factor $\tau \in \mathbb{R}$. For large-scale sparse matrices, $\rho_s(\tau) = \frac{1}{n_p} size\{p \in \mathcal{P} : r_{p,s} \leq \tau\}$. However, in order to accurately judge the difference between solvers for general minitype matrices, we use logarithmic function to increase the gap between solvers, and $\rho_s(\tau)$ is defined as $\rho_s(\tau) = \frac{1}{n_p} size\{p \in \mathcal{P} : log_2(r_{p,s}) \leq \tau\}$. We set $r_{p,s} = r_M = 20$ if and only if solver s does not successfully solve problem p within 50 iterations.

For general minitype data matrices, using the density $(3n-2)/n^2$ of the tridiagonal matrix to determine the density of A and B. For example, when $n = 1000$, the density of

A and B both approximately equal to 0.003 (the actual density is 0.0029). When both the condition numbers of A and B are greater than or equal to 10, 10^2 or 10^4, respectively, the performance ratio images and results are used to analyze the effectiveness and robustness of the methods. Let $x^* = -100 + 200 \times rand(n, 1)$ and $b = Ax^* + B|x^*|$ is generated. The average condition numbers, maximum and minimum condition numbers of 100 GAVE problems under specific conditions obtained by numerical experiments are shown in Table 4.

Table 4: Basic information of numerical results for Example 4.3 with $n = 1000$

	$cond(A) \approx 10$			$cond(A) \approx 10^2$			$cond(A) \approx 10^4$		
	$cond(B)$								
	10	10^2	10^4	10	10^2	10^4	10	10^2	10^4
mean(cond(A))	28.4594	26.3420	26.0373	239.5526	238.7155	243.2022	19279	18813	19766
min(cond(A))	10.0215	10.0210	10.0881	100.2478	100.1344	100.6710	10007	10016	10003
max(cond(A))	98.7427	99.5035	93.2042	969.4184	985.0541	995.7401	99492	92251	91296
mean(cond(B))	26.3489	236.3534	18341	27.7889	237.2793	18984	27.1965	233.1595	18396
min(cond(B))	10.0093	100	10019	10.3195	100.1026	10012	10.0445	100.2196	10006
max(cond(B))	99.9168	959.6709	83750	94.9021	905.7980	84652	99.6455	938.8491	85791

Numerical results for Example 4.3 are reported in Table 5, Figure 1 and Figure 2. For the case that the average condition number of A is relatively small with any B, that the IRGN method performs best, and the probability of the IRGN method winning on a given problem is approximately equal to 0.99, which is slightly more efficient than the IGN method. All methods have high robustness of 100%. In addition, when the average condition number of B is relatively small, robustness rates of the IRGN method are 99%, 98% and 39% for $cond(A) \approx 10, 10^2, 10^4$, respectively. Moreover, it can be seen from Figure 1 and 2 that the IRGN method is also better than the exact methods. The numerical results also show that the inexact methods are more sensitive to the increase of the condition numbers of A than the exact methods, and the same phenomenon is reported in [5]. Unfortunately, its theoretical basis remains to be further studied.

Moreover, we find that the condition number of B has little influence on the effectiveness and robustness of the method, which is mainly determined by the state of A. Therefore, when the matrix A is ill-conditioned, the GN method can have high effectiveness. For the case that the matrix A is in good state, the IRGN method has good effectiveness and achieves the target robust value (100%) firstly.

In order to study the advantage of the inexact methods (IGN and IRGN), when the minimum value of condition number of A and B is set to 10, the influence of matrix density on the method is discussed by changing the matrix density of A and B. We specify that the density of A and B matrices is approximatively equal (from the point of view of the complementarity problem, the density of A and B is roughly equal). The basic results of numerical experiments are presented in Table 6.

Table 7 and Figure 3 show the influence of the density of matrix A and matrix B on the methods. On the whole, the inexact methods (IGN and IRGN) outperform accurate methods (GN, MGN and RGN). The smaller the density of matrix A and matrix B, the better the effectiveness and robustness of the inexact method (the effectiveness is more than 98% and the robustness is 100%). In a word, the IRGN method has better problem-solving ability when A and B matrices are large sparse matrices, and reaches the stable state first.

Consider the large-scale sparse matrices of A and B with certain condition number. Here, $n = 6000$ is used and $density(A) = density(B) \approx 2/n$ and $r_M = 20$. The average condition

Table 5: Numerical results for Example 4.3 with $n = 1000$

Method		$cond(A) \approx 10$			$cond(A) \approx 10^2$			$cond(A) \approx 10^4$		
		$cond(B)$								
		10	10^2	10^4	10	10^2	10^4	10	10^2	10^4
GN	Efficiency(%)	0	1	0	64	67	74	99	99	99
	Robustness(%)	100	100	100	100	100	100	100	100	100
MGN	Efficiency(%)	0	0	0	2	0	1	7	4	2
	Robustness(%)	100	100	100	100	100	100	100	100	100
RGN	Efficiency(%)	0	1	0	10	11	7	21	23	48
	Robustness(%)	100	100	100	100	100	100	100	100	100
IGN	Efficiency(%)	98	98	99	98	97	99	40	38	1
	Robustness(%)	100	100	100	100	100	100	100	100	100
IRGN	Efficiency(%)	99	99	99	98	97	100	39	37	1
	Robustness(%)	100	100	100	100	100	100	100	100	100

Table 6: Numerical results for Example 4.3

	$n = 2000$			$n = 1000$		
	density					
	0.0015	0.003	0.01	0.01	0.1	0.5
real density	0.0014	0.0029	0.0095	0.0095	0.0952	0.4754
mean(cond(A))	26.7903	25.2722	27.2399	25.7902	24.8097	27.6787
min(cond(A))	10.2309	10.0205	10.0996	10.0483	10.1043	10.0030
max(cond(A))	96.5553	98.0649	99.2338	96.9188	86.0889	94.0312
mean(cond(B))	27.2792	27.0169	27.2266	26.1242	26.7462	23.7670
min(cond(B))	10.0101	10.0226	10.0347	10.2438	10.0172	10.1214
max(cond(B))	95.2796	92.7942	96.5369	97.7455	98.7810	97.5212

Table 7: Numerical results for Example 4.3

Method		$n = 2000$			$n = 1000$		
		density					
		0.0015	0.003	0.01	0.01	0.1	0.5
GN	Efficiency(%)	0	0	0	0	0	14
	Robustness(%)	48	8	100	62	100	100
MGN	Efficiency(%)	0	0	0	0	0	13
	Robustness(%)	8	0	0	100	100	100
RGN	Efficiency(%)	0	0	0	0	0	11
	Robustness(%)	14	1	0	44	98	100
IGN	Efficiency(%)	98	99	99	99	100	100
	Robustness(%)	100	100	100	100	100	100
IRGN	Efficiency(%)	99	99	100	100	100	99
	Robustness(%)	100	100	100	100	100	100

numbers of A and B roughly equals 261.5653 and 278.3950 (the smallest and the largest values of A and B are [103.3240, 948.9985] and [100.2780, 955.4369], respectively) for the first set and 19878.52 and 16493.29 (the smallest and the largest values of A and B are [10072.43, 84520.42] and [10057.90, 76134.61], respectively) for the second set. Numerical results for this example are reported in Table 8 and Figure 4. When the condition number of A and B are order of 10^2, the IRGN method performs best and can solve 83% of the problems. In addition, the IRGN method firstly reach a stable state. For the large-scale

sparse GAVE with ill-conditioned matrices, the inexact methods are more sensitive to the increase of the condition number than the exact ones, which is consistent with the results of experiments in [5].

Table 8: Numerical results for Example 4.3 with $n = 6000$

Method		$cond \approx 10^2$	$cond \approx 10^4$
GN	Efficiency(%)	0	50
	Robustness(%)	95	100
MGN	Efficiency(%)	0	0
	Robustness(%)	0	46
RGN	Efficiency(%)	0	0
	Robustness(%)	0	45
IGN	Efficiency(%)	17	1
	Robustness(%)	100	50
IRGN	Efficiency(%)	83	49
	Robustness(%)	100	50

5 Concluding Remarks

In this paper, an inexact relaxed generalized Newton (IRGN) iterative method is developed to solve the generalized absolute value equations. The involved system of linear equations are inexactly solved by means of adopting a relative error tolerance. Convergence properties of the new iteration schemes are analyzed in detail. Numerical experiments are reported to demonstrate the efficiency of the IRGN iteration method.

Acknowledgements

The authors are grateful to the referees for the comments and constructive suggestions, which are valuable in improving the quality of the original paper.

References

[1] L. Abdallah, M. Haddou and T. Migot, Solving absolute value equation using complementarity and smoothing functions, *J. Comput. Appl. Math.* 327 (2018) 196–207.

[2] J.H. Alcantara and J.S. Chen, A novel generalization of the natural residual function and a neural network approach for the NCP, *Neurocomputing* 413 (2020) 368–382.

[3] Z.-Z. Bai, Modulus–based matrix splitting iteration methods for linear complementarity problems, *Numer. Linear Algebra Appl.* 17 (2010) 917–933.

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

[5] J.Y.B. Cruz, O.P. Ferreira and 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.

[6] C.-R. Chen, Y.-N. Yang, D.-M. Yu and D.-R. Han, An inverse-free dynamical system for solving the absolute value equations, *Appl. Numer. Math.* 168 (2021) 170–181.

[7] E.D. Dolan and J.J.Moré, Benchmarking optimization software with performance profiles, *Math. Program.* 91 (2022) 201–213.

[8] X. Dong, X.-H. Shao and H.-L. Shen, A new SOR-like method for solving absolute value equations, *Appl. Numer. Math.* 156 (2020) 410–421.

[9] V. Edalatpour, D.Hezari and D.K. Salkuyeh, A generalization of the Gauss–Seidel iteration method for solving absolute value equations, *Appl. Math. Comput.* 293 (2017) 156–167.

[10] G.H. Golub and C.F. Van Loan, *Matrix Computations, 3rd edn*, The Johns Hopkins University Press, Maryland, 2009.

[11] M.R. Hestenes and E.L. Stiefel, Methods of conjugate gradients for solving linear systems, *J. Res. N.B.S.* 49 (1952) 409–436.

[12] M. Hladík, Bounds for the solutions of absolute value equations, *Comput. Optim. Appl.* 69 (2018) 243–266.

[13] S.-L. Hu, Z.-H. Huang and Q. Zhang, A generalized Newton method for absolute value equations associated with second order cones, *J. Comput. Appl. Math.* 235 (2012) 1490–1501.

[14] B.-H. Huang and W. Li, A modified SOR-like method for absolute value equations associated with second order cones, *J. Comput. Appl. Math.* 400 (2022) 113745.

[15] X. Jiang and Y. Zhang, A smoothing-type algorithm for absolute value equations, *J. Ind. Manag. Optim.* 9 (2013) 789–798.

[16] Y.-F. Ke and C.-F. Ma, SOR-like iteration method for solving absolute value equations, *Appl. Math. Comput.* 311 (2017) 195–202.

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

[18] X. Li and X.-X. Yin, A new modified Newton-type iteration methods for solving generalized absolute value equations, *arXiv preprint* (2021) arXiv: 2103.09452.

[19] O.L. Mangasarian and R.R. Meyer, Absolute value equations, *Linear Algebra Appl.* 419 (2006) 359–367.

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

[21] O.L. Mangasarian, Absolute value programming, *Comput. Optim. Appl.* 36 (2007) 43–53.

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

[23] O.L. Mangasarian, Knapsack feasibility as an absolute value equation solvable by successive linear programming, *Optim. Lett.* 3 (2009) 161–170.

[24] A. Mansoori and M. Erfanian, A dynamic model to solve the absolute value equations, *J. Comput. Appl. Math.* 333 (2018) 28–35.

[25] A. Mansoori, M. Eshaghnezhad and S. Effati, An efficient neural network model for solving the absolute value equations, *IEEE T. Circuits-II* 65 (2017) 391–395.

[26] F. Mezzadri, On the solution of general absolute value equations, *Appl. Math. Lett.* 107 (2020) 106462.

[27] H. Moosaei, S. Ketabchi and M. Hladík, Optimal correction of the absolute value equations, *J. Global Optim.* 79 (2021) 645–667.

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

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

[30] C.C. Paige and M.A. Saunders, LSQR: An algorithm for sparse linear equations and sparse least squares, *ACM Trans. Mathe. Softw. (TOMS)* 8 (1982) 43–71.

[31] J. Rohn, Systems of linear interval equations, *Linear Algebra Appl.* 126 (1989) 39–78.

[32] J. Rohn, A theorem of the alternatives for the equation $Ax+B|x| = b$, *Linear Multilinear Algebra* 52 (2004) 421–426.

[33] J. Rohn, V. Hooshyarbakhsh and R. Farhadsefat, An iterative method for solving absolute value equations and sufficient conditions for unique solvability, *Optim. Lett.* 8 (2014) 35–44.

[34] B. Saheya, C.-H. Yu and J.-S. Chen, Numerical comparisons based on four smoothing functions for absolute value equation, *J. Appl. Math. Comput.* 56 (2018) 1–19.

[35] J.-Y. Tang and J.-C. Zhou, A quadratically convergent descent method for the absolute value equation $Ax + B|x| = b$, *Oper. Res. Lett.* 47 (2019) 229–234.

[36] S.-L. Wu and C.-X. Li, The unique solvability of the absolute value equation, *Appl. Math. Lett.* 76 (2018) 195–200.

[37] S.-L. Wu and C.-X. Li, A note on unique solvability of the absolute value equation, *Optim. Lett.* 14 (2020) 1957–1960.

[38] S.-L. Wu and S.-Q. Shen, On the unique solution of the generalized absolute value equation, *Optim. Lett.* 15 (2021) 2017–2024.

[39] A. Wang, Y. Cao and J.-X. Chen, Modified Newton-type iteration methods for generalized absolute value equations, *J. Optim. Theory Appl.* 181 (2019) 216–230.

[40] H.-Y. Zhou, S.-L. Wu and C.-X. Li, Newton-based matrix splitting method for generalized absolute value equation, *J. Comput. Appl. Math.* 394 (2021) 113578.

[41] X.-Y. Yin and D.-D. Yang, Properties of singular values of matrices, *Studies In College Mathematics* 24 (2021) 56–58.

[42] Z.-S. Yu, L. Li and Y. Yuan, A modified multivariate spectral gradient algorithm for solving absolute value equations, *Appl. Math. Lett.* 121 (2021) 107461.

[43] D.-M. Yu, C.-R. Chen and D.-R. Han, A modified fixed point iteration method for solving the system of absolute value equations, *Optimization* 71 (2022) 449–461.

[44] M. Zamani and M. Hladík, A new concave minimization algorithm for the absolute value equation solution, *Optim. Lett.* 15 (2021) 2241–2254.

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

Manuscript received 1 July 2022
revised 28 November 2022
accepted for publication 9 May 2023

Dongmei Yu
School of Business Administration, College of Science
Institute for Optimization and Decision Analytics,
Liaoning Technical University, Fuxin, 123000, China
E-mail address: yudongmei1113@163.com

Yiming Zhang
School of Business Administration, Liaoning Technical University
Huludao 125105, China;

Institute for Optimization and Decision Analytics
Liaoning Technical University , Fuxin 123000, China
E-mail address: yimingkeyan@163.com

Yifei Yuan
College of Science, Institute for Optimization and Decision Analytics
Liaoning Technical University
Fuxin, 123000, China
E-mail address: 472121474@stu.lntu.edu.cn

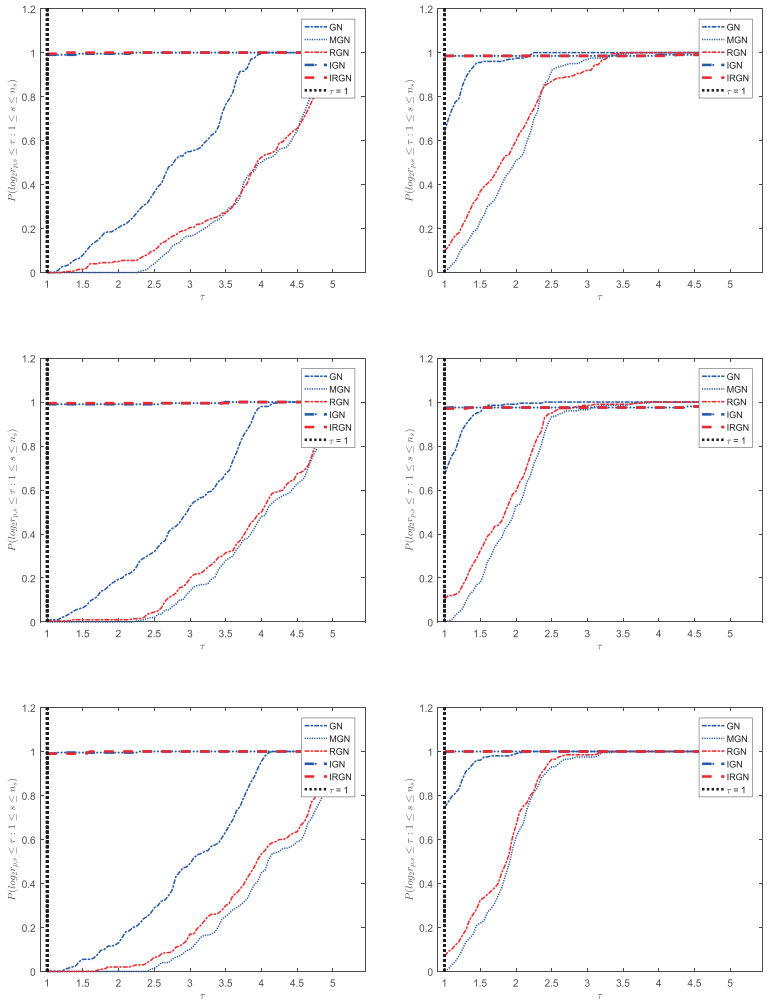

Figure 1: Performance profile graphics for Example 4.3 with $n = 1000$ and $density \approx 0.003$ (top-left: $cond(A) \approx 10$ and $cond(B) \approx 10$; middle-left: $cond(A) \approx 10$ and $cond(B) \approx 10^2$; bottom-left: $cond(A) \approx 10$ and $cond(B) \approx 10^4$; top-right: $cond(A) \approx 10^2$ and $cond(B) \approx 10$; middle-right: $cond(A) \approx 10^2$ and $cond(B) \approx 10^2$; bottom-right: $cond(A) \approx 10^2$ and $cond(B) \approx 10^4$.

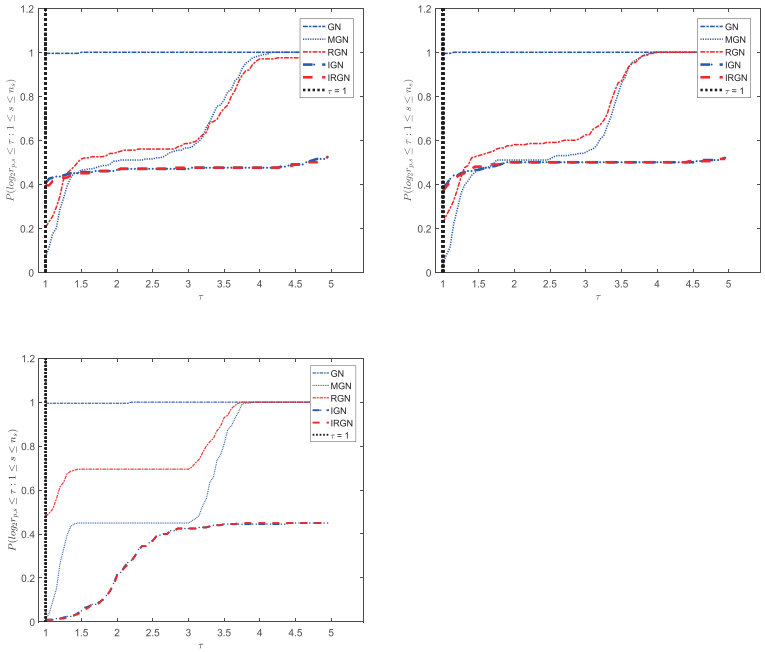

Figure 2: Performance profile graphics for Example 4.3 with $n = 1000$ and $density \approx 0.003$ (top-left: $cond(A) \approx 10^4$ and $cond(B) \approx 10$; top-right: $cond(A) \approx 10^4$ and $cond(B) \approx 10^2$; bottom: $cond(A) \approx 10^4$ and $cond(B) \approx 10^4$.

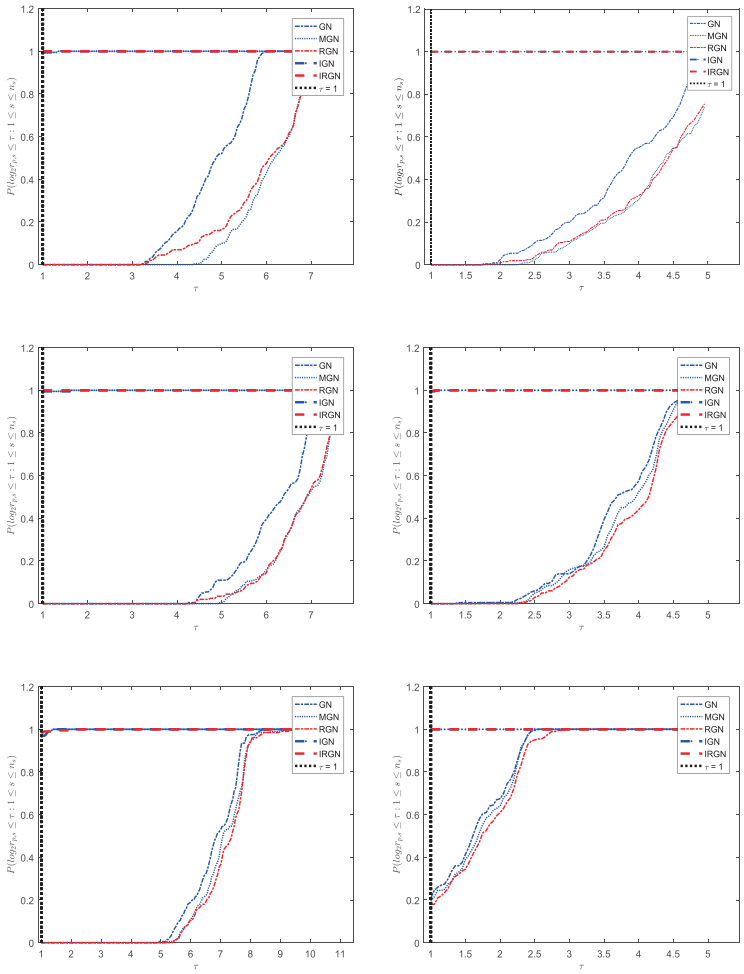

Figure 3: Performance profile graphics for Example 4.3 with $n = 2000$ (left three plots: top-left, *density* ≈ 0.0015; middle-left, *density* ≈ 0.003; bottom-left, *density* ≈ 0.01) and $n = 1000$ (right three plots: top-right, *density* ≈ 0.01; middle-right, *density* ≈ 0.1; bottom-right, *density* ≈ 0.5) that $cond(A)$ and $cond(B)$ are order of 10.

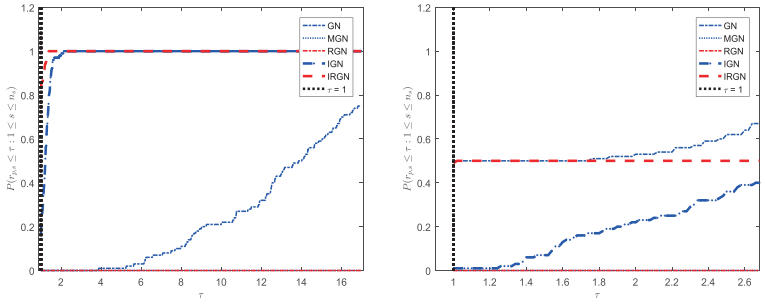

Figure 4: Performance profile graphics for Example 4.3 with $n = 6000$ (left plot: $cond(A)$ and $cond(B)$ are order of 10^2; right plot: $cond(A)$ and $cond(B)$ are order of 10^4).