Journal of Mathematical Modeling
Vol. 13, No. 2, 2025, pp. 251–262. Research Article

JMM

Smoothing the absolute value equations by the component-wise analysis

Mandana Moccari[†*]

[†]*Department of Mathematics, Hamedan Branch, Islamic Azad University, Hamedan, Iran*
Email(s): m_moccari@yahoo.com

Abstract. In this study, an efficient smooth function is introduced to determine the solutions of Absolute Value Equations (AVEs) using a two-step extension of Traub's method. Additionally, a novel approach is proposed for solving AVEs in a component-wise manner. Cubic convergence is achieved under mild assumptions. The results demonstrate that the proposed method is highly effective, as validated by numerical examples.

Keywords: Absolute value equation, Traub's method, high order of convergence, smooth functions.
AMS Subject Classification 2010: 65F10, 15A06, 90C05.

1 Introduction

In this paper, we introduce an efficient high-order method for solving the Absolute Value Equation (AVE) of the form

$$Ax - |x| = b, \tag{1}$$

where $A \in \mathbb{R}^{n\times n}$ and $b \in \mathbb{R}^n$ are given, and $|x|$ denotes the component-wise absolute value of the vector $x \in \mathbb{R}^n$. The general form of the AVE, initially introduced by Rohn [21], is

$$Ax + B|x| = b, \tag{2}$$

where $B \in \mathbb{R}^{n\times n}$. The problem of solving AVEs has garnered significant attention due to its complex nature, particularly when associated with second-order cones. For instance, Hu et al. [11] demonstrated global linear and local quadratic convergence for an extended Newton's method applied to AVE (2). Additionally, Mangasarian and Meyer [17] showed that AVE (1) can be transformed into an Ordinary Linear Complementarity Problem (LCP) [19].

*Corresponding author
Received: 1 September 2024/ Revised: 125 October 2024/ Accepted: 9 November 2024
DOI: 10.22124/jmm.2024.28343.2501

© 2025 University of Guilan

Recently, numerous theoretical and numerical studies have focused on addressing the Non-determinis -tic Polynomial-time (NP-hard) nature of AVEs [6, 22]. The increasing importance of approximating numerical solutions for AVE (1) is underscored by the contributions of Zainali et al. [26], Lotfi [14], Edalatpour et al. [3], and Farhadsefat [4]. Among these efforts, Feng and Liu [5] proposed an improved generalized Newton's method for AVE (1), while Zhang and Wei [27] investigated methods for solving AVE (1) when the interval matrix $[A-I, A+I]$ is regular, with I as the identity matrix. Haghani [8] further extended Traub's two-step method for AVE (1), achieving linear convergence.

Notably, the function $G(x) = Ax - |x| - b$ is nonlinear and not continuously differentiable [2]. To address this, Qi and Sun [20] employed the Clarke generalized Jacobian [2], demonstrating the quadratic convergence of Newton's method. Mangasarian [16] showed that when the singular values of A exceed 1, the semi-smooth Newton's method is well-defined and converges linearly. Tang and Zhou [22] utilized the smooth function $\phi_p(\varepsilon, x) = \sqrt[p]{\varepsilon^p + x^p}$, obtaining quadratic convergence for Newton's method. Furthermore, Yilmaz in [24] introduces additional smooth functions, while Hashemi and Ketabchi employ four smooth functions for solving AVEs [9]. Yilmaz and Sahiner in [25] effectively address non-Lipschitz AVEs using smooth functions.

These foundational works inspired us to leverage smooth functions to develop a new algorithm that not only requires less precision but also exhibits greater stability for solving AVE (1) with a high order of convergence. Specifically, we extend Traub's method [18] by incorporating smooth functions, achieving cubic convergence. The substantial advancements in iterative methods for matrix equations and the optimization of smooth functions in recent years, particularly those that enhance convergence and computational efficiencyhave significantly motivated our approach to solving absolute value equations [1, 13, 15, 28].

For clarity, we now recall our notation. Here, $x^t y$ represents the inner product of x and y, where x^t is the transpose of x. We denote a column vector in $\mathbb{R}^n$ by (a_j). The norm of a vector x is given by $||x|| = \sqrt{x^t x}$. An $n \times n$ matrix A is represented by (a_{ij}), where a_{ij} is the (i, j)-th component of A. The matrix $D(x)$ is a diagonal matrix with vector x on the diagonal, and $||A||$ denotes the spectral norm of matrix A.

The rest of the paper is organized as follows. Section 2 presents the AVEs, discusses smooth functions, and introduces solvability conditions. In Section 3, we establish the local convergence of Traub's method with cubic convergence and compute a bound for the condition number of the proposed function. Finally, numerical examples are provided to validate the theoretical results.

2 Defining the smooth function

If $A = (a_{ij}) \in \mathbb{R}^{n\times n}$, $b = (b_j) \in \mathbb{R}^n$ and $x = (x_j) \in \mathbb{R}^n$, we can define

$$G(x) = Ax - |x| - b, \tag{3}$$

and

$$G_1(x) = Ax - x - b, \quad G_2(x) = Ax + x - b. \tag{4}$$

Now, for given vector x, the smooth function $F(x)$ is given by

$$F(x) = D(G_1(x))\, D(G_2(x))u, \tag{5}$$

where $u=(u_j)$ is a column vector with $u_j=1$ for all j. Note that

$$F(x)=\left(D(Ax-b)^2-D(x)^2\right)u. \tag{6}$$

Then vectors $G_1(x)$ and $G_2(x)$ are shown by components-wise form as follows

$$G_1(x)=\left(\sum_{j=1}^{n}a_{ij}x_j-b_i-x_i\right)\in\mathbb{R}^n,\quad G_2(x)=\left(\sum_{j=1}^{n}a_{ij}x_j-b_i+x_i\right)\in\mathbb{R}^n. \tag{7}$$

Therefore,

$$F(x)=\left((\sum_{j=1}^{n}a_{ij}x_j-b_i)^2-x_i^2\right)\in\mathbb{R}^n. \tag{8}$$

The Jacobian of $F(x)$ at x is

$$\partial F(x)=2(D(Ax-b)A-D(x)). \tag{9}$$

Lemma 1. *If (3) is solvable, that is, there is x^* such that $G(x^*)=0$, then $F(x)$ is solvable and $F(x^*)=0$.*

Proof. If $G(x)$ is solvable and has a solution x^*, then $Ax^*-|x^*|-b=0$, or $Ax^*-b=|x^*|$. It is deduced that $F(x^*)=0$. □

Therefore, given the condition proposed by Mangasarian [17], we conclude that if the Lemma 2 is satisfied, and $G(x)$ is a solvable function, then $F(x)$ is also solvable.

Lemma 2 ([17])**.** *If the smallest singular value of A is greater than 1, then $G(x)$ is solvable.*

Lemma 3. *Let the singular values of A be greater than 1, and the vector $Ax-b$ be positive. If all components of $F(x)$ are greater than or equal to zero, then $G(x)$ is solvable.*

Proof. Suppose $F(x)_i$ is the i-th component of $F(x)$. So,

$$F(x)_i=(\sum_{j=1}^{n}a_{ij}x_j-b_i)^2-x_i^2\geq 0,\quad \text{for every}\quad i. \tag{10}$$

Therefore,

$$\left\{x=(x_i);\quad \sum_{j=1}^{n}a_{ij}x_j-b_i-x_i\geq 0\quad\text{and}\quad \sum_{j=1}^{n}a_{ij}x_j-b_i+x_i\geq 0,\quad \text{for}\quad 1\leq i\leq n\right\}\neq\varnothing. \tag{11}$$

Hence, the bilinear program

$$0=\min_{x\in\mathscr{R}^n}\left\{\sum_{i=1}^{n}F(x)_i;\quad \sum_{j=1}^{n}a_{ij}x_j-b_i-x_i\geq 0\quad\text{and}\quad \sum_{j=1}^{n}a_{ij}x_j-b_i+x_i\geq 0,\quad \text{for}\quad 1\leq i\leq n\right\} \tag{12}$$

is feasible, where

$$\sum_{i=1}^{n}F(x)_i=\left(\sum_{j=1}^{n}a_{ij}x_j-b_i-x_i\right)^t\left(\sum_{j=1}^{n}a_{ij}x_j-b_i+x_i\right).$$

So, $F(x)$ has solution, thus, based on the [17, Proposition 3], the AVE $G(x)$ is also solvable. □

3 Convergence analysis of the proposed method

It is known that the two-step Traub's method is presented as follows

$$\begin{aligned} y^k &= x^k - [\partial F(x^k)]^{-1} F(x^k), \\ x^{k+1} &= y^k - [\partial F(x^k)]^{-1} F(y^k), \end{aligned} \tag{13}$$

for $k = 0, 1, 2, \ldots$. In Section 2, $\partial F(x)$ and $F(x)$ are defined. In the following theorem, existence and boundedness of the inverse Jacobian $F(x)$ are proved.

Theorem 1. *Let x^* be a solution of $F(x) = 0$, which x^* has no zero components. Suppose the singular values of A exceed 1. Then there is an $N(x^*)$ neighborhood from x^* and a constant C, such that for each $x \in N(x^*)$, $\partial F(x)$ is nonsingular and*

$$\left\| \partial F(x)^{-1} \right\| \leq C. \tag{14}$$

Proof. By (9) and (3), the following relation is obtained

$$\partial F(x^*) = 2\left(D(Ax^* - b)A - D(x^*)\right) = 2\left(D(|x^*|)A - D(x^*)\right) = 2D(|x^*|)(A - sign(x^*)), \tag{15}$$

where $sign(x^*)$ is the matrix sign function. If $(A - sign(x^*))^{-1}$ does not exist, then for some $x \neq 0$, we have that $Ax = sign(x^*)x$, which gives the contradiction

$$x^t x < x^t A^t A x = x^t sign(x^*)^t A x = x^t sign(x^*)^t sign(x^*) x = x^t x,$$

where the first inequality is a result of [17, Lemma 1] and the end equality as a consequence of [10, Theorem 5.1]. As a results, $\partial F(x^*)$ is nonsingular.

Now, suppose that for each $N(x^*)$ neighborhood from x^* there is $y \in N(x^*)$, where $\partial F(y)$ is singular or $\|\partial F(y)^{-1}\| \to \infty$. So, there is a sequence $\{y_k\}$ such that $y_k \to x^*$, and $\partial F(y_k)$ is singular or $\|\partial F(y)^{-1}\| \to \infty$. In addition, by defining the function F, $F(x)$ and $\partial F(x)$ are Lipschitz functions. By passing the sequence, $\partial F(x^*)$ must be singular, which is a contradiction. □

Now, we can show that the proposed method (13) is convergent with cubic convergence.

Theorem 2. *Suppose x^* is a solution of $F(x) = 0$. Then, the iteration method (13) is well-defined and converges to x^* in a neighborhood of x^* with cubic convergence.*

Proof. Using Theorem 1, x^* has a neighborhood $N(x^*)$ in which for every $x \in N(x^*)$, $\partial F(x)$ is nonsingular and bounded. So, the proposed method (13) is well-defined. We consider the convergence of the proposed method (13) as component-wise. For the first part, we have

$$\begin{aligned} y^k - x^* &= x^k - x^* - \partial F(x^k)^{-1} F(x^k) = \partial F(x^k)^{-1} \left[\partial F(x^k)(x^k - x^*) - F(x^k) \right] \\ &= \partial F(x^k)^{-1} \left[2(D(Ax^k - b)A - D(x^k))(x^k - x^*) - D(Ax^k - b)D(Ax^k - b)u + D(x^k)D(x^k)u \right]. \end{aligned}$$

For simplification, we put

$$\mathscr{A} = 2(D(Ax^k - b)A - D(x^k))(x^k - x^*) - D(Ax^k - b)D(Ax^k - b)u + D(x^k)D(x^k)u. \tag{16}$$

Thus, we have

$$\begin{aligned}\mathscr{A}_i &= 2(\sum_{j=1}^{n} a_{ij}x_j)^2 - 2(\sum_{j=1}^{n} a_{ij}x_j^k)(\sum_{j=1}^{n} a_{ij}x_j^*) - 2x_i^k x_i^k + 2x_i^k x_i^* - (\sum_{j=1}^{n} a_{ij}x_j^k)^2 \\ &\quad -2b_i\sum_{j=1}^{n} a_{ij}x_j^k + 2b_i\sum_{j=1}^{n} a_{ij}x_j^* + 2b_i\sum_{j=1}^{n} a_{ij}x_j^k - b_ib_i + x_i^k x_i^k \\ &= (\sum_{j=1}^{n} a_{ij}x_j^k - \sum_{j=1}^{n} a_{ij}x_j^*)^2 - (x_i^k - x_i^*)^2 - (\sum_{j=1}^{n} a_{ij}x_j^* - b_i)^2 + x_i^{*2}.\end{aligned}$$

Therefore, by taking the norm, it is obtained as follows

$$\|y^k - x^*\| \leq c\left(\|A\|^2\|x^k - x^*\|^2 + \|x^k - x^*\|^2\right) = c(\|A\|^2 + 1)\|x^k - x^*\|^2.$$

This means that, the quadratic convergence has been obtained for the first step, which can be considered as the quadratic convergence for Newton's Method.

Now, the second step of the proposed method (13) is considered. We have

$$x^{k+1} - x^* = y^k - x^* - \partial F(x^k)^{-1}F(y^k) = \partial F(x^k)^{-1}\left[\partial F(x^k)(y^k - x^*) - F(y^k)\right].$$

Again, for simplification, we put

$$\beta = 2(D(Ax^k - b)A - D(x^k))(y^k - x^*) - D(Ay^k - b)D(Ay^k - b)u + D(y^k)D(y^k)u.$$

So, we can write down

$$\begin{aligned}\beta_i = &\ 2(\sum_{j=1}^{n} a_{ij}x_j^k)(\sum_{j=1}^{n} a_{ij}y_j^k) - 2b_i\sum_{j=1}^{n} a_{ij}y_j^k - 2x_i^k y_i^k - 2\sum_{j=1}^{n} a_{ij}x_j^k\sum_{j=1}^{n} a_{ij}x_j^* \\ &+ 2b_i\sum_{j=1}^{n} a_{ij}x_j^* + 2x_i^k x_i^* - (\sum_{j=1}^{n} a_{ij}y_j^k)^2 + 2b_i\sum_{j=1}^{n} a_{ij}y_j^k - b_ib_i + y_i^k y_i^k \\ = &\ (y_i^k - x_i^*)(y_i^k + x_i^*) - 2x_i^k(y_i^k - x_i^*) - (\sum_{j=1}^{n} a_{ij}x_j^*)^2 + 2b_i\sum_{j=1}^{n} a_{ij}x_j^* \\ &- b_ib_i - x_i^*x_i^* + (\sum_{j=1}^{n} a_{ij}x_j^* - \sum_{j=1}^{n} a_{ij}y_j^k)(\sum_{j=1}^{n} a_{ij}x_j^* + \sum_{j=1}^{n} a_{ij}y_j^k) \\ &+ 2\sum_{j=1}^{n} a_{ij}x_j^k(\sum_{j=1}^{n} a_{ij}y_j^k - \sum_{j=1}^{n} a_{ij}x_j^*) \\ = &\ (y_i^k - x_i^*)(y_i^k + x_i^* - 2x_i^k) + \sum_{j=1}^{n} a_{ij}(y_j^k - x_j^*)(2\sum_{j=1}^{n} a_{ij}x_j^k - \sum_{j=1}^{n} a_{ij}x_j^* - \sum_{j=1}^{n} a_{ij}y_j^k) \\ = &\ (y_i^k - x_i^*)(y_i^k + x_i^* + x_i^* - x_i^* - x_i^k - x_i^k) \\ &+ \sum_{j=1}^{n} a_{ij}(y_j^k - x_j^*)\left(\sum_{j=1}^{n} a_{ij}((x_j^k - x_j^*) + (x_j^* - y_j^k) + (x_j^k - x_j^*))\right).\end{aligned}$$

So, by taking the norm, it is obtained as follows:

$$\begin{aligned}\|x^{k+1}-x^*\| \leq\ & c\|y^k-x^*\|\left(\|y^k-x^*\|+2\|x^k-x^*\|\right)+\|A\|\|y^k-x^*\|\left(\|A\|(2\|x^k-x^*\|+\|x^*-y^k\|)\right)\\ \leq\ & c(1+\|A\|^2)(2+\|x^k-x^*\|)\|x^k-x^*\|^3.\end{aligned}$$

The proof is complete now. □

3.1 Conditioning

Zainali and Lotfi [26] computed the condition number of $G(x)$. In this paper, the condition number of $F(x)$ is calculated similar to the study by Wozniakowski [23]. The matrix A is considered as input data d for the $F(x)=F(X;d)=0$. So, the condition number of $F(x;d)$ is computed by the following formula

$$\text{cond}(F;d)=||F_x'(x^*;d)^{-1}F_d'(x^*;d)||\frac{||d||}{||x^*||}, \tag{17}$$

where, F_x' and F_d' stand for the derivatives of F with respect to x and d. x^* is also a solution of $F(x)=0$. Then the matrix form of $F(x)$ is $F(x;A)=D(Ax-b)D(Ax-b)-D(x)D(x)$, and

$$F_x'(x^*;A)=\partial F(x^*),\quad F_A'(x^*;A)=2x^*D(Ax^*-b)=2x^*D(|x^*|). \tag{18}$$

Hence, using the Theorem 1

$$\text{cond}(F;A)=||F_x'(x^*;A)^{-1}F_A'(x^*;A)||\frac{||A||}{||x^*||}\leq 2C||x^*||||A||. \tag{19}$$

We can prove it.

Lemma 4. *Let A be the data for the function (6). Then a bound for the condition number of $F(x;A)$ is*

$$\text{cond}(F;A)\leq 2C||x^*||||A||, \tag{20}$$

where, C is a bound for $\partial F(x^*)^{-1}$, and x^* is a solution of $F(x)$.

4 Numerical Examples

In this section, the theoretical results from the previous sections are validated through numerical experiments. The computational results of the method in (13) demonstrate that it effectively solves the AVE (1), with the computational error for function (6) being approximately equal to, or even lower than, that for the AVE (1). Several examples are provided to verify the conditions of Lemma 3, with various models used for input data. The algorithm, referred to as SA, is detailed below and implemented using Wolfram Mathematica 13.2.

Algorithm 1 An Smoothing Algorithm for Solving Absolute Value Equations(SA).

Choose the matrix $A \in \mathbb{R}^{n\times n}$, vector $b \in \mathbb{R}^n$, and initial vector x_0 with components strictly greater than 0.
Set tolerance and maximum iterations maxIter.
Define the function: $F(x) = \left(D(Ax-b)^2 - D(x)^2\right)u$
Define the Jacobian matrix: $\partial F(x) = \partial F(x) = 2(D(Ax-b)A - D(x))$
Set $j = 0$.
while $\|F(x_j)\|_2 >$ tolerance **and** $j <$ maxIter **do**
  if $min(|x_j|) = 0$ **then**
    Adjust x_j to ensure no component is zero:
    $x_j = 0.1 + \varepsilon$, where ε is a small positive amount.
  end if
  Solve the linear system $\partial F(x_j)H_j = -F(x_j)$ to find the vector H_j.
  Set $y_j = H_j + x_j$.
  Solve the linear system $\partial F(x_j)M_j = -F(y_j)$ to find the vector M_j.
  Set $x_{j+1} = M_j + y_j$.
  Increment $j = j + 1$.
end while
Return $\min(|x_j|)$) **and** x_j as the solution.

In Tables 1, 2, and 3, n and j denote the problem size and the number of iterations, respectively. The columns $||F(x^j)||$ and $||G(x^j)||$ represent the norms of the functions (6) and (3), respectively, calculated at the final iteration. The tolerance level has been set to 10^{-6}. Additionally, we compute the Approximate Computational Order of Convergence (ACOC) using the following formula [7]

$$\text{ACOC} = \frac{\ln\left(\frac{||z^{k+1}-z^k||}{||z^k-z^{k-1}||}\right)}{\ln\left(\frac{||z^k-z^{k-1}||}{||z^{k-1}-z^{k-2}||}\right)}. \tag{21}$$

To ensure that none of the components of the obtained root are zero, which is necessary to satisfy the conditions of Theorem 1, the absolute value of all components was calculated, and the minimum value among them was determined. Additionally, at each step of the proposed algorithm SA, a check is performed, and if any component is zero, a small value is added to the vector to avoid zero components.

In Tables 1, 2, and 3, a comparison between the proposed Algorithm SA and Algorithm 1 from [12] is provided. For this purpose, the required parameters are defined as follows: $\delta = 0.5$, $\sigma = 10^{-5}$, $\mu_0 = 1$, $p = 2$, and $\beta = (\min(1, ||H_p(\mu_0, x_0, A, B, b)||))^2 + 0.1$. Additionally, the variables τ_k, α_k, and Δz_k are computed iteratively within the algorithm. The smoothing function $\phi_p(x,\mu) = \sqrt[p]{x^p + \mu^p}$ is defined, transforming the absolute value equation $G(x) = Ax - B|x| - b$ into

$$\begin{pmatrix} \mu \\ Ax - B\Phi_p(x,\mu) - b \end{pmatrix} = 0, \quad \Phi_p(x,\mu) = (\phi_p(x_1,\mu), \phi_p(x_2,\mu), \ldots, \phi_p(x_n,\mu))^T,$$

where it suffices for both μ and $H_p(\mu, x, A, B, b) = Ax - B\Phi_p(x,\mu) - b$ to vanish. Although the efficient Algorithm 1 in [12] incorporates multiple parameters to accelerate convergence, the algorithm SA presented in this paper achieves higher speed and accuracy. This is demonstrated through tests on three matrices, $A1$, $A2$, and $A3$, in the following examples.

Table 1: Comparison of the algorithm SA with Algorithm 1 in paper [12] for the matrix $A1$.

n	Algorithm SA						Algorithm 1 in [12]			
	j	$\|\|F(x^j)\|\|$	$\|\|G(x^j)\|\|$	time	ACOC	$min(\|x^*\|)$	j	$\|\|G(x^j)\|\|$	μ	time
10	7	5.35103e-11	2.47994e-11	0.	3.19738	1.05599	8	3.73066e-10	5.25627e-10	0.
50	8	3.21239e-11	8.33566e-12	0.	3.25347	1.08951	8	2.33259e-8	2.31964e-8	0.025625
100	9	2.34269e-7	1.13033e-7	0.	3.83023	1.00546	8	3.68088e-7	2.78098e-7	0.025625
300	9	4.07181e-10	9.76172e-11	0.046875	3.06315	1.00605	9	1.33156e-9	2.02348e-7	0.209375
500	9	1.04635e-9	2.75719e-10	0.109375	3.19207	1.0018	9	6.09483e-8	6.37309e-8	0.41875
1000	11	3.59595e-9	8.94203e-10	0.828125	2.96811	1.00002	10	1.17743e-8	1.16879e-7	2.125
3000	11	2.72738e-8	6.78912e-9	24.3906	2.70581	1.00039	10	1.26661e-7	7.30427e-9	36.8125

Table 2: Comparison of the proposed algorithm SA with Algorithm 1 in the paper [12] for the matrix $A2$.

n	Proposed Algorithm SA						Algorithm 1 in [12]			
	j	$\|\|F(x^j)\|\|$	$\|\|G(x^j)\|\|$	time	ACOC	$min(\|x^*\|)$	j	$\|\|G(x^j)\|\|$	μ	time
10	8	4.15038e-12	9.37486e-13	0.	2.90619	1.32536	9	1.7809e-10	2.83331e-10	0.015625
50	9	5.06895e-11	9.96302e-12	0.	2.81905	0.993656	9	9.37263e-9	9.97254e-9	0.015625
100	10	5.92947e-9	2.77732e-9	0.015625	3.90934	1.0351	9	1.75483e-7	1.45069e-7	0.03125
300	10	4.16966e-7	2.07083e-7	0.078125	3.18499	0.99055	10	1.48579e-10	1.21976e-10	0.140625
500	11	1.29293e-9	2.76103e-10	0.25	3.7573	0.846637	10	1.30956e-8	1.74386e-8	0.34375
1000	12	4.71025e-9	9.59544e-10	1.1875	2.56405	0.890179	11	9.87817e-10	1.22836e-10	2.29688
3000	14	3.81488e-8	7.4269e-9	31.5938	2.43326	0.949937	13	1.19409e-8	1.37132e-7	32.4688

Example 1. We consider a random matrix $A1$ as follows:

```
A1=10(5+8RandomReal[{0,1},{n,n}]);
B=-IdentityMatrix[n];
x=1+2RandomReal[{0,1},n];
b=A.x-Abs[x];
x0=x+.1
```

In this example, we choose positive elements for $A1$. Also, we choose a random vector x with the components in the interval $[1,3]$ with a uniform distribution.

Example 2. In this example, we choose the matrix $A2$ with negative components. The vectors x and b are generated by

```
A2 = 10 (-3 - 7 RandomReal[{0, 1}, {n, n}]);
B=-IdentityMatrix[n];
x = (-1 - 3 RandomReal[{0, 1}, n]);
b = -(A.x - Abs[x]);
x0=x-0.1
```

Similar to the Example 1, the results are reported in Table 2.

Example 3. Here, we consider another matrix $A3$, with arbitrary components. The vectors x and b are also as follows

```
A3 = 10 (-20 +10 RandomReal[{0, 1}, {n, n}]);
B = -IdentityMatrix[n];
x = (-1 - 3 RandomReal[{0, 1}, n]);
b = A.x - Abs[x];
x0=x-0.1
```

Again, similar to the Example 1, the results are shown in Table 3.

Table 3: Comparison of the proposed algorithm SA with Algorithm 1 in paper [12] for the matrix $A3$.

n	Proposed Algorithm SA						Algorithm 1 in [12]			
	j	$\|\|F(x^j)\|\|$	$\|\|G(x^j)\|\|$	time	ACOC	$min(\|x^*\|)$	j	$\|\|G(x^j)\|\|$	μ	time
10	7	5.15581e-7	2.29392e-7	0.	2.95997	1.05959	8	2.23759e-10	3.40483e-10	0.015625
50	9	2.84346e-9	1.33862e-9	0.	3.73513	1.04304	8	1.22135e-8	1.27559e-8	0.015625
100	9	2.23854e-10	4.24258e-11	0.	2.32618	1.00115	8	1.60054e-7	1.33523e-7	0.03125
300	11	9.88384e-10	2.01637e-10	0.046875	2.44464	1.01019	9	2.98226e-10	2.51496e-10	0.109375
500	10	2.71921e-9	5.46811e-10	0.234375	2.63987	1.0029	9	1.09284e-8	1.45707e-8	0.28125
1000	12	3.37636e-8	1.5848e-8	0.859375	2.8777	1.00786	11	1.95668e-9	5.33908e-11	1.03125
3000	13	7.98398e-7	3.87551e-7	24.4688	2.56357	1.00019	12	1.83168e-8	1.65008e-7	18.2344

As shown in Tables 1-3, the proposed Algorithm SA works practically by the ACOC near to 3, and it can obtain a solution for the AVE (1), and yield desirable results for different scales of problems. The computation time is also displayed in a column titled "time," indicating the low computational cost.

Example 4. In this example, the proposed Algorithm SA is compared with the efficient Traub method in [8] using the Hilbert matrix, known for its ill-conditioning and numerical challenges. A scaling factor was applied to the Hilbert matrix, which was tested in various sizes. The results of this comparison, summarized in Table 4, highlight the performance and robustness of both methods. The Traub method, while often converging quickly, was highly sensitive to numerical precision, requiring a minimum setting of 200 to avoid divergence. Notably, in cases of convergence, the norm $||G(x^j)||$ was found to be exactly zero, raising concerns about the validity of the results. In contrast, the proposed method exhibited greater stability, converging consistently without such high precision requirements. Consequently, there were more instances of divergence in Traub method compared to the proposed method. The results, including computation time (time) and the number of iterations (j), are detailed in Table 4. The maximum number of iterations (maxIterations) was set to 50, with a tolerance of 10^{-10}

```
A =k HilbertMatrix[n];
b = Table[1,{i,1,n}];
x0=Table[0.1,{i,1,n}];
tolerance = 10^(-10);
maxIterations = 50
```

5 Conclusions

In this paper, a smooth function was introduced to determine the roots of the AVE (1), and Traub's method was extended using this proposed smooth function, resulting in cubic convergence. It can be concluded

Table 4: Comparison of the proposed method with Traub method in Paper [8].

		Proposed method				Traub method		
k	n	j	$\|\|F(x^j)\|\|$	$\|\|G(x^j)\|\|$	time	j	$\|\|G(x^j)\|\|$	time
1	219	37	2.36511e-13	1.2339e-14	1.25	50	33832	257.938
50	9	9	2.80679e-16	8.00593e-16	0.	50	0.12569	0.0625
50	58	10	1.29697e-15	3.04452e-15	0.015625	50	3.37652	5.45313
100	128	29	1.53322e-15	5.34754e-15	0.296875	50	0.211609	24.3906
300	182	36	4.595e-15	1.97757e-14	0.75	50	0.331168	49.0156
500	44	45	2.05302e-15	1.50696e-14	0.09375	50	1.65797	4.23438
500	94	19	7.29298e-14	8.94958e-14	0.140625	50	0.00508866	15.9219
700	36	14	9.61267e-15	1.83493e-14	0.015625	9	0.	0.421875
1000	65	14	2.91708e-15	1.63757e-14	0.046875	50	0.00759145	7.09375

that the proposed smoothing method (13) provides reliable solutions for problems of any scale, given an appropriate choice of initial data. Furthermore, the computational error is comparable to or lower than that of the original AVE (1). A practical bound for the condition number of the proposed equation (6) was also derived. Compared to the methods discussed, the proposed approach demonstrates superior stability, speed, and accuracy. Finally, numerical examples were presented to validate the theoretical findings.

Acknowledgements

The author sincerely thanks the reviewers for their valuable feedback and insightful comments, which have significantly improved this manuscript. Your constructive suggestions are greatly appreciated.

References

[1] J.H. Alcantara, J.S. Chen, M.K. Tam, *Method of alternating projections for the general absolute value equation*, J. Fixed Point Theory Appl. **25** (2023) 39.

[2] F.H. Clarke, *Optimization and Nonsmooth Analysis*, SIAM, 1990.

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

[4] R. Farhadsefat, T. Lotfi, J. Rohn, *A note on regularity and positive definiteness of interval matrices*, Open Math. **10** (2012) 322–328.

[5] J. Feng, S. Liu, *An improved generalized Newton method for absolute value equations*, SpringerPlus **5** (2016) 1042.

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

[7] M. Grau-Sanchez, M. Noguera, J. Gutierrez, *On some computational orders of convergence*, Appl. Math. Lett. **23** (2010) 472–478.

[8] F.K. Haghani, *On generalized Traubs method for absolute value equations*, J. Optim. Theory Appl. **166** (2015) 619–625.

[9] F. Hashemi, S. Ketabchi, *Numerical comparisons of smoothing functions for optimal correction of an infeasible system of absolute value equations*, Numer. Algebra Control Optim. **10** (2019) 13–21.

[10] N.J. Higham, *Functions of Matrices: Theory and Computation*, SIAM, 2008.

[11] 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.

[12] X. Jiang, Y. Zhan, *A smoothing-type algorithm for absolute value equations.*, J. Ind. Manag. Optim. **9** (2013) 4.

[13] E. Khosravi Dehdezi, S. Karimi, *A fast and efficient newton-shultz-type iterative method for computing inverse and moore-penrose inverse of tensors*, J. Math. Model. **9** (2021) 645–664.

[14] T. Lotfi, H. Vieseh, *A note on unique solvability of the absolute value equation*, J. Linear. Topological Algebra. **2** (2013) 77–81.

[15] W.H. Luo, J. Guo, L. Yin, *A dimension expanded newton-type method for absolute value equations*, J. Appl. Math. Comput. **70** (2024) 3219–3233.

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

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

[18] M. Moccari, T. Lotfi, V. Torkashvand, *On the stability of a two-step method for a fourth-degree family by computer designs along with applications*, Int. J. Nonlinear Anal. Appl. **14** (2023) 261–282.

[19] P.M. Pardalos, *The linear complementarity problem in: Advances in Optimization and Numerical Analysis*, Springer, 1994.

[20] L. Qi, J. Sun, *A nonsmooth version of Newton's method*, Math. Program. **58** (1993) 353–367.

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

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

[23] H. Wozniakowski, *Numerical stability for solving nonlinear equations*, Numer. Math. **27** (1976) 373–390.

[24] N. Yilmaz, *Introducing three new smoothing functions: Analysis on smoothing-newton algorithms*, J. Math. Model. **463-479** (2024) 463–479.

[25] N. Yilmaz, A. Sahiner, *Smoothing techniques in solving non-lipschitz absolute value equations*, Int. J. Comput. Math. **100** (2023) 867–879.

[26] 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.

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

[28] R. Ziadi, A. Bencherif-Madani, *A mixed algorithm for smooth global optimization*, J. Math. Model. **11** (2023) 207–228.