Research Article
Modified SOR-Like Method for Absolute Value Equations

Cui-Xia Li and Shi-Liang Wu
School of Mathematics, Yunnan Normal University, Kunming 650500, Yunnan, China
Correspondence should be addressed to Cui-Xia Li; lixiatk@126.com
Received 27 July 2020; Revised 29 September 2020; Accepted 11 November 2020; Published 24 November 2020

1. Introduction

Consider the absolute value equations (AVE)

\[ Ax - |x| = b, \]  

where \( A \in \mathbb{R}^{n \times n} \) and \( b \in \mathbb{R}^n \) and \( |x| \) denotes all the components of the vector \( x \in \mathbb{R}^n \) by absolute value. Replacing “\( |x| \)” in (1) by “\( B|x| \)” with \( B \in \mathbb{R}^{n \times n} \) naturally generates the general AVE [1, 2]. At present, the AVE gradually attracts considerable attention because some optimization problems such as linear programming, convex quadratic programming, and linear complementarity problem [3–7] can be formulated as the AVE (1).

In recent years, to efficiently find the numerical solution of the AVE (1), a great deal of effort is developing iteration methods. For example, for solving the AVE (1), a generalized Newton method was presented in [8] and is simply described as follows:

\[
x^{k+1} = (A - D(x^k))^{-1} \left( |x^k| + b \right), \quad k = 0, 1, \ldots
\]  

where \( D(x^k) = \text{diag}(\text{sign}(x^k)) \); \( D(x) = \text{diag}(\text{sign}(x)) \) denotes a diagonal matrix corresponding to \( \text{sign}(x) \). There are other forms of the generalized Newton method; one can see [9–13] for more details. Clearly, at every iteration step of the generalized Newton method (2), the inverse of the matrix \( A - D(x^k) \) should be computed. Noting that the matrix \( A - D(x^k) \) is changed with the iteration index \( k \), the calculations of the generalized Newton method may be very costly. To overcome this changed iteration matrix, the Picard iteration method in [14] is easily considered as follows:

\[
x^{k+1} = A^{-1} \left( |x^k| + b \right), \quad k = 0, 1, \ldots
\]  

Clearly, the Picard iteration method (3) is needed to compute the inverse of the matrix \( A \). Similarly, by reformulating the AVE (1) as a nonlinear equation with two-by-two block form, combining with the classical SOR-like iteration method, an SOR-like iteration method in [15] was proposed to solve it and it is simply described as follows:

\[
\begin{align*}
x^{(k+1)} &= (1 - \omega)x^{(k)} + \omega A^{-1}(y^{(k)} + b), \\
y^{(k+1)} &= (1 - \omega)y^{(k)} + \omega|x^{(k+1)}|,
\end{align*}
\]

where \( \omega > 0 \). (4)

Some convergence conditions of the SOR-like iteration method were given when the involved parameter satisfied certain conditions. Further, from the aspect of the involved iteration matrix of the SOR-like iteration method, in [16], some new convergence conditions were presented.

It is noted that if the matrix \( A \) in (3) or (4) is ill-conditioned, then at every iteration step of the Picard and SOR-
like methods an ill-conditioned linear system needs to be solved. In this case, the cost of the calculation of the inverse of the matrix \( A \) may be high. To overcome the inverse of the matrix \( A \), Li [17] extended the classical AOR iteration method for solving the AVE and discussed the convergence properties for the AOR method. By using the Gauss–Seidel splitting, the generalized Gauss–Seidel (GGS) iteration method was presented in [18] to solve the AVE (1).

In this paper, we fasten on the SOR-like iteration method for solving the AVE (1). By expressing equivalently the implicit fixed-point equation of the AVE as a nonlinear equation with two-by-two block form, a modified SOR-like iteration method is presented by a concrete matrix splitting of the involved coefficient matrix. A considerable advantage of the modified SOR-like iteration method is the inverse of the matrix \( A \) can be avoided. From this point of view, the computing efficiency of the modified SOR-like iteration method may be better than the SOR-like iteration method when both are used to solve the AVE (1).

For our later analysis, here some terminologies are briefly explained. Let \( \mathbb{R}^n \) be the finite dimensional Euclidean space, whose norm is denoted by \( \| \cdot \| \). For \( x \in \mathbb{R}^n \), \( \text{sign}(x) \) denotes a vector with elements equal to 1, 0, -1 depending on whether the value of the corresponding element of \( x \) is larger than zero, equal to zero, or less than zero.

The rest of the layout of this paper is divided into three sections. In the second section, the modified SOR-like iteration method is designed and its convergence conditions are presented. In the third section, some numerical experiments are reported. In the fourth section, some concluding remarks are given to end this paper.

### 2. Modified SOR-Like Iteration Method

In this section, the modified SOR-like iteration method is presented. For this purpose, by using \( y = |x| \) for the AVE (1), we have

\[
\begin{bmatrix}
Ax - y = b, \\
-x + y = 0,
\end{bmatrix}
\]

i.e.,

\[
\begin{pmatrix}
A & -I \\
-D & I
\end{pmatrix}
\begin{pmatrix}
x \\
y
\end{pmatrix}
= \begin{pmatrix}
b \\
0
\end{pmatrix}
= \bar{b},
\]

where \( \bar{b} = D(x) = \text{diag}(\text{sign}(x)) \). \( x \in \mathbb{R}^n \).

Let

\[
A = D - L - U,
\]

where \( D = \text{diag}(A) \), and \( L \) and \( U \) are strictly lower and upper triangular matrices obtained from \( A \), respectively. If we take

\[
\bar{A} = \bar{D} - \bar{L} - \bar{U},
\]

where

\[
\bar{D} = \begin{pmatrix}
D & 0 \\
0 & I
\end{pmatrix},
\]

\[
\bar{L} = \begin{pmatrix}
L & 0 \\
\bar{D} & 0
\end{pmatrix},
\]

\[
\bar{U} = \begin{pmatrix}
U & 1 \\
0 & 0
\end{pmatrix},
\]

then we have

\[
\begin{pmatrix}
x \\
y
\end{pmatrix} = M_\omega \begin{pmatrix}
x \\
y
\end{pmatrix} + \omega (\bar{D} - \omega \bar{L})^{-1} \begin{pmatrix}
b \\
0
\end{pmatrix},
\]

where \( M_\omega = (\bar{D} - \omega \bar{L})^{-1} [(1 - \omega)\bar{D} + \omega \bar{U}] \) and \( \omega > 0 \). Based on equation (10), the modified SOR-like iteration method is naturally obtained and described below.

The modified SOR-like iteration method: let the initial vectors \( x^{(0)} \in \mathbb{R}^n \) and \( y^{(0)} \in \mathbb{R}^n \) be given and \( \omega > 0 \). For \( k = 0, 1, 2, \ldots \) until the iteration sequence \( \{x^{(k)}, y^{(k)}\} \) is convergent, calculate

\[
\begin{align*}
x^{(k+1)} &= (D - \omega L)^{-1} \left( (1 - \omega)D + \omega U \right) x^{(k)} + \omega \left( y^{(k)} + b \right), \\
y^{(k+1)} &= (1 - \omega) x^{(k)} + \omega \left[ x^{(k+1)} \right].
\end{align*}
\]

(11)

Lemma 1 is quoted for the latter use.

**Lemma 1** (see [19]). Let \( \lambda \) be any root of \( x^2 - bx + d = 0 \) with \( b, d \in \mathbb{R} \). Then, \( |\lambda| < 1 \) if and only if \( |d| < 1 \) and \( |b| < 1 + d \).

Let the iteration errors be

\[
\begin{align*}
e^x_k &= x^* - x^{(k)}, \\
e^y_k &= y^* - y^{(k)},
\end{align*}
\]

where \((x^*, y^*)\) is the solution pair of equation (6) and \((x^{(k)}, y^{(k)})\) is generated by the iteration method (11). Then, the following convergence conditions with respect to the modified SOR-like iteration method (11) can be given (see Theorem 1).

**Theorem 1.** Let \( A \in \mathbb{R}^{n \times n} \) be nonsingular and \( b \in \mathbb{R}^n \). Denote

\[
\begin{align*}
s &= \| (D - \omega L)^{-1} [(1 - \omega)D + \omega U] \|, \\
v &= \| (D - \omega L)^{-1} \|, \\
\alpha &= |1 - \omega|, \\
\beta &= \omega^2 v.
\end{align*}
\]

If

\[
\begin{align*}
\alpha s &< 1, \\
\beta &< (1 - \alpha) (1 - s),
\end{align*}
\]

then
\[
\| (e^x_{k+1}, e^y_{k+1}) \| < \| (e^x_k, e^y_k) \|, \quad k = 0, 1, \ldots, \tag{15}
\]

where
\[
\| (e^x_k, e^y_k) \| = \sqrt{|e^x_k|^2 + |e^y_k|^2}. \tag{16}
\]

**Proof.** Let us subtract equation (11) from
\[
\begin{align*}
  x^* &= (D - \omega L)^{-1} \left( ((1 - \omega)D + \omega U)x^* + \omega (y^* + b) \right), \\
  y^* &= (1 - \omega)y^* + \omega|x^*|.
\end{align*}
\tag{17}
\]
with \((x^*, y^*)\) being the solution pair of equation (6). Then
\[
\begin{align*}
  e^x_{k+1} &= (D - \omega L)^{-1} \left( (1 - \omega)D + \omega U \right) e^x_k + \omega (D - \omega L)^{-1} e^y_k, \\
  e^y_{k+1} &= (1 - \omega)e^y_k + \omega \left( |x^*| - |x^{(k+1)}| \right).
\end{align*}
\tag{18}
\]

From (18), we can get
\[
\begin{align*}
  \|e^x_{k+1}\| &\leq \| (D - \omega L)^{-1} \| \cdot \| (1 - \omega)D + \omega U \| \cdot \|e^x_k\| \\
  &\leq \|e^y_{k+1}\| + \omega \|x^*| - |x^{(k+1)}|\|, \\
  \|e^y_{k+1}\| &\leq \|1 - \omega\| \|e^y_k\| + \omega \|x^*| - |x^{(k+1)}|\|
\end{align*}
\tag{19}
\]

It holds that
\[
\begin{pmatrix} -\omega & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \|e^x_{k+1}\| \\ \|e^y_{k+1}\| \end{pmatrix} \leq \begin{pmatrix} 0 & |1 - \omega| \\ s & \omega \nu \end{pmatrix} \begin{pmatrix} \|e^x_k\| \\ \|e^y_k\| \end{pmatrix}. \tag{20}
\]

By left-multiplying (20) by the nonnegative matrix
\[
\begin{pmatrix} 0 & 1 \\ 1 & \omega \end{pmatrix} > 0,
\tag{21}
\]
we have
\[
\begin{align*}
  \|e^x_{k+1}\| &\leq \begin{pmatrix} s & \omega \nu \\ s & \omega \nu \end{pmatrix} \begin{pmatrix} \|e^x_k\| \\ \|e^y_k\| \end{pmatrix} \\
  &\leq \begin{pmatrix} s & \omega \nu \\ s & \omega \nu \end{pmatrix} \begin{pmatrix} \|e^x_{k-1}\| \\ \|e^y_{k-1}\| \end{pmatrix} \\
  &\leq \begin{pmatrix} s & \omega \nu \\ s & \omega \nu \end{pmatrix} \begin{pmatrix} \|e^x_{k-2}\| \\ \|e^y_{k-2}\| \end{pmatrix} \\
  \vdots
\end{align*}
\tag{22}
\]

Let
\[
T = \begin{pmatrix} s & \omega \nu \\ \omega s & |1 - \omega| + \omega^2 \nu \end{pmatrix}.
\tag{23}
\]

Clearly, if \(\rho (T) < 1\), then \(\lim_{k \to \infty} T^k = 0\). This implies
\[
\begin{align*}
\lim_{k \to \infty} \|e^x_k\| &= 0, \\
\lim_{k \to \infty} \|e^y_k\| &= 0. \tag{24}
\end{align*}
\]

In this way, the iteration sequence \(\{x^{(k)}, y^{(k)}\}\) produced by the modified SOR-like method (11) can converge to the solution of equation (6).

Next, we just need to get sufficient conditions such that \(\rho (T) < 1\). Assumed that \(\hat{\lambda}\) denotes an eigenvalue of the matrix \(T\). Then
\[
(\lambda - s)(\lambda - |1 - \omega| - \omega^2 \nu) - \omega^2 \nu \hat{\lambda} = 0,
\tag{25}
\]
which is equal to
\[
\lambda^2 - (|1 - \omega| + \omega^2 \nu + s)\lambda + s|1 - \omega| = 0. \tag{26}
\]

Using Lemma 1 for equation (26), \(|\lambda| < 1\) is equal to
\[
\begin{align*}
|1 - \omega| + \omega^2 \nu + s < 1 + s|1 - \omega|. \\
|1 - \omega| + \omega^2 \nu + s < 1 + s|1 - \omega|.
\end{align*}
\tag{27}
\]

Therefore, if condition (14) holds, then \(\rho (T) < 1\).

If the idea of this proof for the modified SOR-like method (11) is extended to the SOR-like method (4), then the corresponding matrix \(T\) is as follows:
\[
T = \begin{pmatrix} |1 - \omega| & \omega \nu \\ \omega |1 - \omega| + \omega^2 \nu & \lambda + s|1 - \omega| \end{pmatrix},
\tag{28}
\]
where \(\nu = \|A^{-1}\|\) (see [15]). By simple computations, we can get that if
\[
0 < \omega < 2, \quad \alpha + \sqrt{\omega^2 \nu} < 1,
\tag{29}
\]
then the SOR-like method (4) is convergent. Therefore, we obtain a new convergence condition for the SOR-like method (4) and see the following result.

**Theorem 2.** Let the conditions of Theorem 1 be satisfied. Denote
\[
\alpha = |1 - \omega|, \quad \beta = \omega^2 \nu.
\tag{30}
\]

If
\[
0 < \omega < 2, \quad \alpha + \sqrt{\beta} < 1, \tag{31}
\]
then
\[
\| (e^x_{k+1}, e^y_{k+1}) \| < \| (e^x_k, e^y_k) \|, \quad k = 0, 1, \ldots, \tag{32}
\]
Comparing Theorem 2 with Theorem 3.1 in [15], it is easy to see that the region of the parameter \( \omega \) in Theorem 2 is the same as that in Theorem 3.1 in [15]. Both demand \( 0 < \omega < 2 \) in Theorem 2 and Theorem 3.1 in [15]. The difference between Theorem 2 and Theorem 3.1 in [15] is on \( \alpha \) and \( \beta \). The former is \( \alpha + \sqrt{\beta} < 1 \) and the latter is \( \alpha^2 - 3\alpha^2 - 2\alpha\beta - 2\beta^2 + 1 > 0 \).

See Theorem 3.1 in [15]. In form, the latter is more complicated than the former.

From Theorem 1, Corollary 1 is obtained.

**Corollary 1.** Let the conditions of Theorem 1 be satisfied. Denote

\[
s = \| (D - \omega L)^{-1} ((1 - \omega)D + \omega U) \|
\]
\[
v = \| (D - \omega L)^{-1} \|
\]
\[
\alpha = |1 - \omega|,
\]
\[
\beta = \omega^2 v.
\]

If

\[
\max \left\{ 0, \frac{s - 1}{s} \right\} < \omega < \frac{1 + s}{s},
\]

\[
\beta < (1 - \alpha)(1 - s),
\]

then

\[
\left\| (e_k^x, e_{k+1}^x) \right\| < \left\| (e_{k-1}^x, e_k^x) \right\|, \quad k = 0, 1, \ldots,
\]

where

\[
\left\| (e_k^x, e_k^x) \right\| = \sqrt{\left\| e_k^x \right\|^2 + \left\| e_k^x \right\|^2}.
\]

### 3. Numerical Examples

In this section, two numerical examples are provided to show the effectiveness of the modified SOR-like method from two aspects: the iteration step (denoted by "IT") and the computing time (denoted by "CPU"). We compared the modified SOR-like method with the SOR-like method [15]. Here, all initial vector for these two testing methods are set to be zero vector, and both are terminated if the relative residual error (RES) satisfies

\[
\text{RES} = \frac{\| A x^{(k)} - x^{(k)} - b \|_2}{\| b \|_2} \leq 10^{-6},
\]

or if the iteration step is larger than 500. All the tests are performed in MATLAB 7.0.

In the following tables, "MSOR" and "SOR" denote the modified SOR-like method and the SOR-like method [15], respectively. "−" denotes the iteration steps larger than 500 or the CPU times (second) larger than 500 seconds.

To get fast convergence rate of the modified SOR-like method and the SOR-like method [15], the experimentally optimal parameter \( \omega_{\text{exp}} \) is adopted, which results in the smallest iteration step.

**Example 1** (see [6, 7, 17]). Let the AVE in (1) be composed with

\[
A = \tilde{A} + 4I \in \mathbb{R}^{\text{non}},
\]
\[
\tilde{A} = \text{tridiag}(-1.5I, S, -0.5I) \in \mathbb{R}^{\text{non}},
\]

with

\[
S = \text{tridiag}(-1.5, 4, -0.5) \in \mathbb{R}^{\text{non}},
\]

\[
b = Ax^* - |x^*|,
\]

where

\[
x^* = (-1, 1, -1, 1, \ldots, -1, 1)^T \in \mathbb{R}^n.
\]

In Table 1, we list some numerical results of the modified SOR-like method and the SOR-like method for Example 1.

In brief, the numerical results in Table 1 show that under certain conditions, the computational efficiency of the modified SOR-like method overmatches the SOR-like method.

**Example 2.** For the AVE in (1), we chose a random \( A \) by the following structure:

\[
A = 100 * \text{eye}(n) - 0.02 * (2 * \text{rand}(n,n) - 1),
\]

and its all singular values exceed 1. The right-hand side \( b \) is set to be \( b = Ax^* - |x^*| \), where

\[
x^* = (-1, 1, -1, 1, \ldots, -1, 1)^T \in \mathbb{R}^n.
\]
Although it is a very difficult task.

The computational efficiency of the modified SOR-like method overmatches the SOR-like method in certain conditions. He data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by National Natural Science Foundation of China (no. 11961082).

References

[1] J. Rohn, "A theorem of the alternatives for the equation $Ax + B|x| = b$," Linear and Multilinear Algebra, vol. 52, no. 6, pp. 421–426, 2004.

[2] O. L. Mangasarian, "Absolute value programming," Computational Optimization and Applications, vol. 36, no. 1, pp. 43–53, 2007.

[3] O. L. Mangasarian, "Absolute value equations via concave minimization," Optimization Letters, vol. 1, pp. 1–8, 2007.

[4] R. W. Cottle and G. B. Dantzig, "Complementary pivot theory of mathematical programming," Linear Algebra and Its Applications, vol. 1, no. 1, pp. 103–125, 1968.

[5] R. W. Cottle, J.-S. Pang, and R. E. Stone, The Linear Complementarity Problem, Academic, San Diego, CA, USA, 1992.

[6] S.-L. Wu and C.-X. Li, “Two-sweep modulus-based matrix splitting iteration methods for linear complementarity problems,” Journal of Computational and Applied Mathematics, vol. 302, pp. 327–339, 2016.

[7] Z.-Z. Bai, “Modulus-based matrix splitting iteration methods for linear complementarity problems,” Numerical Linear Algebra with Applications, vol. 17, no. 6, pp. 917–933, 2010.

[8] O. L. Mangasarian, “A generalized Newton method for absolute value equations,” Optimization Letters, vol. 3, no. 1, pp. 101–108, 2009.

[9] C.-X. Li, “A modified generalized Newton method for absolute value equations,” Journal of Optimization Theory and Applications, vol. 170, no. 3, pp. 1055–1059, 2016.

[10] L. Caccetta, B. Qu, and G. Zhou, “A globally and quadratically convergent method for absolute value equations,” Computational Optimization and Applications, vol. 48, no. 1, pp. 45–58, 2011.

[11] S.-L. Hu, Z.-H. Huang, and Q. Zhang, “A generalized Newton method for absolute value equations associated with second order cones,” Journal of Computational and Applied Mathematics, vol. 235, no. 5, pp. 1490–1501, 2011.

[12] S. Ketabchi and H. Moosaei, “An efficient method for optimal correcting of absolute value equations by minimal changes in the right hand side,” Computers & Mathematics with Applications, vol. 64, no. 6, pp. 1882–1885, 2012.

[13] C. Zhang and Q. J. Wei, “Global and finite convergence of a generalized Newton method for absolute value equations,” Journal of Optimization Theory and Applications, vol. 143, no. 2, pp. 391–403, 2009.

[14] J. Rohn, V. Hooshyarbakhsh, and R. Farhadsefat, “An iterative method for solving absolute value equations and sufficient conditions for unique solvability,” Optimization Letters, vol. 8, no. 1, pp. 35–44, 2014.

[15] Y.-F. Ke and C.-F. Ma, “SOR-like iteration method for solving absolute value equations,” Applied Mathematics and Computation, vol. 311, pp. 195–202, 2017.

[16] P. Guo, S.-L. Wu, and C.-X. Li, “On the SOR-like iteration method for solving absolute value equations,” Applied Mathematics Letters, vol. 97, pp. 107–113, 2019.
[17] C.-X. Li, “A preconditioned AOR iterative method for the absolute value equations,” *International Journal of Computational Methods*, vol. 14, no. 2, Article ID 1750016, 2017.

[18] V. Edalatpour, D. Hezari, and D. Khojasteh Salkuyeh, “A generalization of the Gauss-Seidel iteration method for solving absolute value equations,” *Applied Mathematics and Computation*, vol. 293, pp. 156–167, 2017.

[19] S.-L. Wu, T.-Z. Huang, and X.-L. Zhao, “A modified SSOR iterative method for augmented systems,” *Journal of Computational and Applied Mathematics*, vol. 228, no. 1, pp. 424–433, 2009.