Applied Mathematics and Computation 242 (2014) 707–715

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A practical asymptotical optimal SOR method

Guo-Yan Meng *

Department of Mathematics, Xinzhou Teachers University, Xinzhou, Shanxi Province, PR China
State Key Laboratory of Scientific/Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, Beijing, PR China

ARTICLE INFO

Keywords:
Optimal relaxation factors
SOR
Linear systems

ABSTRACT

This paper presents a practical asymptotical optimal successive over-relaxation (SOR) method for solving the large sparse linear system. Based on two optimization models, asymptotically optimal relaxation factors are given, which are computed by solving the low-order polynomial equations in each iteration. The coefficients of the two polynomials are determined by the residual vector and the coefficient matrix A of the real linear system. The numerical examples show that the new methods are more feasible and effective than the classical SOR method.

1. Introduction

Consider the numerical solution of a large sparse system of linear equations

$$Ax = b \tag{1.1}$$

where $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is a known nonsingular matrix and $x,\ b \in \mathbb{R}^n$ are vectors. The splitting iterative method, especially, the successive over-relaxation (SOR) method [9], is one of the important tools for solving the linear system (1.1). We write

$$A = D - L - U$$

with $D = diag(A)$, and assume $det(D) \neq 0,\ L$ strictly lower- and U strictly upper-triangular matrices, respectively. Thus, the classical SOR iterative method can be expressed as

$$x_{k+1} = L_\omega x_k + g_\omega, \quad k = 0, 1, 2, \ldots. \tag{1.2}$$

where

$$L_\omega = (D - \omega L)^{-1}((1-\omega)D + \omega U), \quad g_\omega = \omega(D - \omega L)^{-1}b.$$

It is easy to verify that if we set

$$M_\omega := \frac{1}{\omega}(D - \omega L), \tag{1.3}$$

then the SOR iteration (1.2) in correction form is as follows:

* Address: Department of Mathematics, Xinzhou Teachers University, Xinzhou, Shanxi Province, PR China.
E-mail address: mgy1226@126.com

http://dx.doi.org/10.1016/j.amc.2014.06.034
0096-3003/

$$x_{k+1} = x_k + M_\omega^{-1} r_k, \quad r_k = b - Ax_k.$$

For $\omega = 1$, the SOR becomes the Gauss–Seidel method. And it is important for the SOR method that the necessary condition of the convergence does not depend on properties of A.

Theorem 1.1 [9]. *A necessary condition for the SOR method to converge is* $|\omega - 1| \leq 1$. *(For* $\omega \in \mathbb{R}$ *this condition becomes* $\omega \in (0, 2)$.*)*

Hadjidimos [6] carefully summarized the SOR method, including some general convergency results and more special ones in the case that A possesses some further properties, e.g. positive definiteness, L-, M- and P-cyclic consistently ordered property, etc. Especially for the latter, also see [9], analytic formula about the optimal relaxation factor ω_{opt} is given, namely, the optimal relaxation factor which maximizes the asymptotic rate of convergence is precisely specified as the unique positive real root of the equation

$$(\rho(J)\omega_{opt})^p = \left(p^p (p-1)^{1-p}\right)(\omega_{opt} - 1), \tag{1.4}$$

where $\rho(J)$ is the spectral radius of the associated Jacobi matrix. In particular, for $p = 2$, ω_{opt} of (1.4) can be expressed equivalently as

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho^2(J)}}. \tag{1.5}$$

In a word, the rate of convergence of the SOR method depends on the value of relaxation factor ω. Furthermore, Eiermann and Varga [4] showed that no polynomial acceleration of the SOR method (for any real ω) is asymptotically faster than the SOR scheme with $\omega = \omega_{opt}$ under the assumption

$$\sigma(J^2) \subset [0, \beta^2] \quad (\beta = \rho(J)),$$

where $\sigma(J^2)$ the spectrum of J^2. If an optimal, or a nearly optimal relaxation factor is easily obtained, the SOR method itself becomes one of the most efficient and practical iterative method for the system of linear Eq. (1.1). Based on the above facts, the SOR-like method [5] or the generalized SOR (GSOR) method [2] is discussed for solving the augmented linear system. Even Bai et al. [3] used an SOR scheme to accelerate the normal/skew-Hermitian splitting (NSS) iteration and discussed the convergence conditions for this SOR scheme. Unexceptionally, these methods all discussed the optimal relaxation factor. Though the optimal value can be determined for certain special problems, in general it can be determined only by using an eigenvalue analysis. This approach is not feasible in practice due to the complicated computation and the high computing cost. Therefore, we pay more attention to the practical methods for choosing the optimal parameters of the SOR method itself.

Wen, Meng and Wang [11] presented the optimization model to determine the optimal parameters in each iteration for quasi-Chebyshev accelerated (QCA) method, and Wang and Meng [10] made use of the standard quadratic programming technique to choose the optimal weighting matrices at each step of the iteration method. In [8] an adaptive SOR (ASOR) algorithm is given, which computes a nearly optimal relaxation factor ω. While Bai and Chi [1] obtained the optimal relaxation factor of the SOR method by solving a quintic polynomial equation. Similarly, Huang [7] chosen the optimal parameters using a cubic polynomial equation, which comes from the minimization of the F-norm. In this paper, we concentrate on the determination of the optimal relaxation factor and propose a simple strategy for approximating the optimal parameter in the SOR method, which is practical and less costly.

The outline of the paper is as follows. In Section 2, we describe the strategy for obtaining the asymptotically optimal relaxation factor in detail. In Section 3, we use some numerical experiments to show the effectiveness of the scheme. Finally, in Section 4, we end the paper with some conclusions.

2. The practical asymptotical optimal SOR method

In this section, we first describe the modified asymptotical optimal SOR method. Without loss of generality, we assume that diagonal matrix of the matrix A is the identity matrix. Accordingly, the splitting matrix M_ω in the (1.3) can be expressed as

$$M_\omega = \frac{1}{\omega}(I - \omega L).$$

MAOSOR Method (Modified Asymptotical Optimal SOR Method) Let $x_0 \in \mathbb{R}^n$ be an arbitrary initial guess. For $k = 0, 1, 2, \ldots$ until the sequence of iterates $\{x_k\}_{k=0}^{\infty} \subset \mathbb{R}^n$ converges, compute the next iteration x_{k+1} according to the following procedure.

(1) Compute $r_k = b - Ax_k$.
(2) Compute ω_k, which is the solution of the following optimization problem:

- when A is a symmetric positive definite matrix, set $x = x_k + M_\omega^{-1} r_k$,

$$\min_\omega \frac{1}{2} x^T A x - x^T b; \tag{2.1}$$

- when A is a nonsymmetric matrix, let $r = b - Ax,\ x = x_k + M_\omega^{-1} r_k$,

$$\min_\omega r^T r. \tag{2.2}$$

(3) Compute $x_{k+1} = x_k + M_{\omega_k}^{-1} r_k$.

If ω_k is independent of the iteration index k, namely, ω_k is not calculated in Step (2), the MAOSOR method is the classical SOR method.

Next, we discuss the concrete solutions of the optimization problems (2.1) and (2.2).

Theorem 2.1. *Let x_k be generated by the MAOSOR iterative method, and let $T_\omega = I - AM_\omega^{-1}$. Then the solution ω_{opt} of the optimization problem (2.1) is precisely specified as the root of the equation*

$$\phi(\omega) = -\frac{1}{\omega^2} r_k^T T_\omega^T M_\omega^{-2} r_k = 0 \tag{2.3}$$

and the solution ω_{opt} of the optimization problem (2.2) satisfies the equation

$$\psi(\omega) = -\frac{2}{\omega^2} r_k^T T_\omega^T A M_\omega^{-2} r_k = 0;$$

Furthermore, $\phi(\omega)$ and $\psi(\omega)$ has the following explicitly computing form

$$\begin{aligned}\phi(\omega) = &-r_k^T r_k - \sum_{i=1}^{n-1} \omega^i \left((i+1) r_k^T L^i r_k - \sum_{j=0}^{i-1} (j+1) r_k^T S_{ij} r_k \right) \\ &- \sum_{i=n}^{2n-2} \omega^i \left((2n-1-i) r_k^T L^i r_k - \sum_{j=i-n}^{n-1} (j+1) r_k^T S_{ij} r_k - \sum_{j=n}^{i-1} (2n-1-j) r_k^T S_{ij} r_k \right) \\ &+ \sum_{i=2n-1}^{3n-2} \omega^i \left(\sum_{j=i-n}^{2n-2} (2n-1-j) r_k^T S_{ij} r_k \right)\end{aligned} \tag{2.4}$$

and

$$\begin{aligned}\psi(\omega) = &-2 \left(r_k^T A r_k + \sum_{i=1}^{n-1} \omega^i \left((i+1) r_k^T A L^i r_k - \sum_{j=0}^{i-1} (j+1) r_k^T P_{ij} r_k \right) \right. \\ &\left. + \sum_{i=n}^{2n-2} \omega^i \left((2n-1-i) r_k^T A L^i r_k - \sum_{j=i-n}^{n-1} (j+1) r_k^T P_{ij} r_k - \sum_{j=n}^{i-1} (2n-1-j) r_k^T P_{ij} r_k \right) - \sum_{i=2n-1}^{3n-2} \omega^i \left(\sum_{j=i-n}^{2n-2} (2n-1-j) r_k^T P_{ij} r_k \right) \right).\end{aligned} \tag{2.5}$$

where $S_{ij} = (L^T)^{i-j-1} A L^j$ and $P_{ij} = (L^T)^{i-j-1} A^T A L^j$.

Proof. Let x^* be the unique solution of the system of linear Eq. (1.1), and

$$\varepsilon_k = x_k - x^*, \quad r_k = b - Ax_k, \quad k = 0, 1, 2, \ldots \tag{2.6}$$

be the error and the residual vectors, respectively. Then (2.6) yields

$$r_k = b - Ax_k = Ax^* - Ax_k = -A\varepsilon_k. \tag{2.7}$$

While the model (2.1) is equivalent to the following problem:

$$\min_\omega \Phi(\omega) = \frac{1}{2} x^T A x - x^T b + \frac{1}{2} (x^*)^T A x^*.$$

Thus, combining the MAOSOR method with (2.7), we obtain

$$\begin{aligned}\Phi(\omega) &= \frac{1}{2} (x - x^*)^T A (x - x^*) = \frac{1}{2} \left(\varepsilon_k + M_\omega^{-1} r_k \right)^T A \left(\varepsilon_k + M_\omega^{-1} r_k \right) = \frac{1}{2} \left(A^{-1} (-r_k + A M_\omega^{-1} r_k) \right)^T \left(-r_k + A M_\omega^{-1} r_k \right) \\ &= \frac{1}{2} r_k^T (I - A M_\omega^{-1})^T A^{-1} (I - A M_\omega^{-1}) r_k = \frac{1}{2} r_k^T T_\omega^T A^{-1} T_\omega r_k.\end{aligned}$$

Moreover, according to [1] we have

$$\phi(\omega) := \frac{d\Phi(\omega)}{d\omega} = -\frac{1}{\omega^2} r_k^T T_\omega^T M_\omega^{-2} r_k.$$

This also means that the solution of the optimization problem (2.1), namely, the stationary points of $\Phi(\omega)$, are the roots of the equation $\phi(\omega)=0$. When A is a nonsymmetric matrix, from (2.2) we have

$$\Psi(\omega)=r^Tr=\left(b-Ax_k-AM_\omega^{-1}r_k\right)^T\left(b-Ax_k-AM_\omega^{-1}r_k\right)=r_k^T(I-AM_\omega^{-1})^T(I-AM_\omega^{-1})r_k=r_k^TT_\omega^TT_\omega r_k.$$

Similarly, according to [1] we know

$$\psi(\omega):=\frac{d\Psi(\omega)}{d\omega}=-\frac{2}{\omega^2}r_k^TT_\omega^TAM_\omega^{-2}r_k$$

and the roots of the equation $\psi(\omega)=0$ are the stationary points of $\Psi(\omega)$.

To obtain explicit scheme (2.4) and (2.5), we need to approximate M_ω^{-1}. Fortunately, Bai and Chi [1] has observed that

$$M_\omega^{-1}=\omega(I-\omega L)^{-1}=\omega\sum_{i=0}^{n-1}(\omega L)^i, \tag{2.8}$$

since L is a strictly lower-triangle matrix, and $L^n=0$ naturally holds. We choose the complete $n-1$ polynomial (2.8) rather than a cubic polynomial [1]. Thus we have

$$M_\omega^{-2}=\omega^2\left(\sum_{i=0}^{n-1}(i+1)(\omega L)^i+\sum_{i=n}^{2n-2}(2n-1-i)(\omega L)^i\right)$$

and

$$T_\omega=I-AM_\omega^{-1}=I-\sum_{i=0}^{n-1}\omega^{i+1}AL^i.$$

Thereby, when A is a symmetric positive definite matrix, we immediately obtain

$$\begin{aligned}T_\omega^TM_\omega^{-2}=\omega^2\left(I-\sum_{i=0}^{n-1}\omega^{i+1}AL^i\right)^T\left(\sum_{i=0}^{n-1}(i+1)(\omega L)^i+\sum_{i=n}^{2n-2}(2n-1-i)(\omega L)^i\right)=\omega^2\left(I+\sum_{i=1}^{n-1}\omega^i\left((i+1)L^i-\sum_{j=0}^{i-1}(j+1)S_{ij}\right)\right.\\ \left.-\sum_{i=2n-1}^{3n-2}\omega^i\left(\sum_{j=i-n}^{2n-2}(2n-1-j)S_{ij}\right)+\sum_{i=n}^{2n-2}\omega^i\left((2n-1-i)L^i-\sum_{j=i-n}^{n-1}(j+1)S_{ij}-\sum_{j=n}^{i-1}(2n-1-j)S_{ij}\right)\right),\end{aligned}$$

which gives (2.4) by substituting it into (2.3). Similarly, if A is a nonsymmetric matrix,

$$\begin{aligned}T_\omega^TAM_\omega^{-2}=\omega^2\left(I-\sum_{i=0}^{n-1}\omega^{i+1}(L^i)^TA^T\right)A\left(\sum_{i=0}^{n-1}(i+1)(\omega L)^i+\sum_{i=n}^{2n-2}(2n-1-i)(\omega L)^i\right)=\omega^2\left(A+\sum_{i=1}^{n-1}\omega^i\left((i+1)AL^i-\sum_{j=0}^{i-1}(j+1)P_{ij}\right)\right.\\ \left.-\sum_{i=2n-1}^{3n-2}\omega^i\left(\sum_{j=i-n}^{2n-2}(2n-1-j)P_{ij}\right)+\sum_{i=n}^{2n-2}\omega^i\left((2n-1-i)AL^i-\sum_{j=i-n}^{n-1}(j+1)P_{ij}-\sum_{j=n}^{i-1}(2n-1-j)P_{ij}\right)\right).\end{aligned}$$

Finally, direct computations give (2.5). □

The computation of the root of $\phi(\omega)=0$ or $\psi(\omega)=0$ in Theorem 2.1 is the most expensive part of the MAOSOR approach and we would like to avoid it. So we are more inclined to choose the lower-order truncations of polynomials $\phi(\omega)$ and $\psi(\omega)$. That means, if A is a symmetric positive definite matrix, the accurate cubic polynomial is very suitable for approximating $\phi(\omega)$ in (2.4),

$$\begin{aligned}\phi(\omega)&\approx-r_k^Tr_k-\sum_{i=1}^{3}\omega^i\left((i+1)r_k^TL^ir_k-\sum_{j=0}^{i-1}(j+1)r_k^TS_{ij}r_k\right)\\&=-\left(r_k^Tr_k+\omega r_k^T(2L-A)r_k+\omega^2r_k^T(3L^2-2AL-L^TA)r_k+\omega^3r_k^T(4L^3-3AL^2-2L^TAL-(L^T)^2A)r_k\right)\\&=-\left(\hat\alpha_0+\hat\alpha_1\omega+\hat\alpha_2\omega^2+\hat\alpha_3\omega^3\right),\end{aligned} \tag{2.9}$$

where

$$\begin{aligned}&\hat\alpha_0=r_k^Tr_k,\quad \hat\alpha_1=2r_k^TLr_k-r_k^TAr_k,\quad \hat\alpha_2=3r_k^TL^2r_k-3r_k^TALr_k,\\&\hat\alpha_3=4r_k^TL^3r_k-4r_k^TAL^2r_k-2r_k^TL^TALr_k.\end{aligned}$$

While A is a nonsymmetric matrix, the quartic polynomial is a good choice to approximate $\psi(\omega)$ in (2.5), namely,

$$\begin{aligned}\psi(\omega) &\approx -2\left(r_k^T A r_k + \sum_{i=1}^{4}\omega^i\left((i+1)r_k^T A L^i r_k - \sum_{j=0}^{i-1}(j+1)r_k^T P_{ij} r_k\right)\right)\\ &= -2\left(r_k^T A r_k + \omega r_k^T(2AL - A^TA)r_k + \omega^2 r_k^T\left(3AL^2 - 2A^TAL - L^TA^TA\right)r_k\right.\\ &\quad \left.+\omega^3 r_k^T\left(4AL^3 - 3A^TAL^2 - 2L^TA^TAL - (L^T)^2A^TA\right)r_k + \omega^4 r_k^T\left(5AL^4 - 4A^TAL^3 - 3L^TA^TAL^2 - 2(L^T)^2A^TAL - (L^T)^3A^TA\right)r_k\right)\\ &= -2\left(\hat{\beta}_0 + \hat{\beta}_1\omega + \hat{\beta}_2\omega^2 + \hat{\beta}_3\omega^3 + \hat{\beta}_4\omega^4\right), \end{aligned} \tag{2.10}$$

where

$$\begin{aligned}&\hat{\beta}_0 = r_k^T A r_k, \quad \hat{\beta}_1 = 2r_k^T AL r_k - r_k^T A^T A r_k, \quad \hat{\beta}_2 = 3\left(r_k^T AL^2 r_k - r_k^T A^T AL r_k\right),\\ &\hat{\beta}_3 = 4r_k^T AL^3 r_k - 4r_k^T A^T AL^2 r_k - 2r_k^T L^T A^T AL r_k,\\ &\hat{\beta}_4 = 5\left(r_k^T AL^4 r_k - r_k^T A^T AL^3 r_k - r_k^T L^T A^T AL^2 r_k\right).\end{aligned}$$

Therefore, in Step (2) of the MAOSOR method, ω_k is actually determined simply by (2.9) or (2.10), which also means that we can rewrite the MAOSOR method in a more practical form.

PAOSOR Method (Practical Asymptotical Optimal SOR Method) Let $x_0 \in \mathbb{R}^n$ be an arbitrary initial guess. For $k = 0, 1, 2, \ldots$ until the sequence of iterates $\{x_k\}_{k=0}^{\infty} \subset \mathbb{R}^n$ converges, compute the next iteration x_{k+1} according to the following procedure.

(1) Compute $r_k = b - Ax_k$.
(2) By the Newton method compute ω_k, which is the root of the following equation:
- when A is a symmetric positive definite matrix,

$$1 + \frac{\hat{\alpha}_1}{\hat{\alpha}_0}\omega + \frac{\hat{\alpha}_2}{\hat{\alpha}_0}\omega^2 + \frac{\hat{\alpha}_3}{\hat{\alpha}_0}\omega^3 = 0; \tag{2.11}$$

- when A is a nonsymmetric matrix,

$$1 + \frac{\hat{\beta}_1}{\hat{\beta}_0}\omega + \frac{\hat{\beta}_2}{\hat{\beta}_0}\omega^2 + \frac{\hat{\beta}_3}{\hat{\beta}_0}\omega^3 + \frac{\hat{\beta}_4}{\hat{\beta}_0}\omega^4 = 0. \tag{2.12}$$

(3) Compute $x_{k+1} = x_k + \omega_k(I - \omega_k L)^{-1} r_k$.

Remark 1. In the above method, if $\hat{\alpha}_0 = 0$ in (2.11), then $r_k = 0$ is valid and x_k is the solution of the linear system (1.1). Similarly, if $\hat{\beta}_0 = 0$ and $r_k \neq 0$, from left to right, we can find the first $\hat{\beta}_i \neq 0$ $(i = 1, \ldots, 4)$. Thus (2.12) may be revised as

$$1 + \frac{\hat{\beta}_{i+1}}{\hat{\beta}_i}\omega + \ldots + \frac{\hat{\beta}_4}{\hat{\beta}_i}\omega^{4-i} = 0.$$

Remark 2. If $\frac{\|r_{k-1}\|}{\|r_k\|} \leqslant \epsilon$ and ϵ is nearly to 1, we apply a simple strategy to the above method, not to update ω_k in the $(k+1)$th iteration. In other words, we directly compute x_{k+1} with ω_{k-1} instead of ω_k in Step (3).

3. Numerical experiments

In this section, we test the PAOSOR iteration by numerical experiments. All iterations are started from a zero initial guess and terminated if the current iteration satisfies

$$RES = \frac{\|b - Ax_k\|_2}{\|b\|_2} \leqslant \varepsilon.$$

All the runs were performed in MATLAB on PC with 2.30 GHz processor and 4 GB memory. In our experiments, the right-hand side vector b is chosen such that the exact solution of the system of linear equations is $(1, 1, \ldots, 1)^T \in \mathbb{R}^n$.

In actual computations, we adopt the same strategy to calculate the asymptotical optimal relaxation factor ω in both AOSOR [1] and PAOSOR methods, and set $\beta = \gamma = 1$ for the AOSOR iteration. In addition, the Newton method for calculating the relaxation factor ω stops as long as the absolute value of the function in (2.11) or (2.12) is less than 0.01, respectively. Moreover, we only perform a new updating in $\omega \in (0, 2)$.

Problem. We consider the classical two-dimensional partial differential equation with Dirichlet boundary conditions:

$$\begin{cases} -\frac{\partial^2 u}{\partial t_1^2} - \frac{\partial^2 u}{\partial t_2^2} + \xi\frac{\partial u}{\partial t_1} + \zeta\frac{\partial u}{\partial t_1} + 4\sigma u = f(t_1, t_2) & (t_1, t_2) \in \Omega \\ u(t_1, t_2) = 0 & (t_1, t_2) \in \partial\Omega, \end{cases} \tag{3.1}$$

where ξ, ζ and σ are constants, Ω is the unit square $(0,1)\times(0,1)$ in $\mathbb{R}^2$, $\partial\Omega$ is the boundary of the domain Ω, and $f(t_1,t_2):\Omega\to\mathbb{R}^1$ is a given function.

Eq. (3.1) is discretized using the five-point difference with the equidistant step-size $h=\frac{1}{N+1}$ and $n=N^2$. The resulting matrix A is of the form

$$A=\begin{pmatrix} T & \mu_2 I & & & \\ \eta_2 I & T & \mu_2 I & & \\ & \ddots & \ddots & \ddots & \\ & & \eta_2 I & T & \mu_2 I \\ & & & \eta_2 I & T \end{pmatrix}\in\mathbb{R}^{n\times n},$$

here T is an N-by-N tridiagonal matrix

$$T=\text{tridiag}\,(\eta_1,\mu_0,\mu_1),$$

Table 1
Numerical results for $\xi=\zeta=\sigma=0$ ($\varepsilon=h^2/5$).

h^{-1}		32	64	128	256	512	1024
SOR	IT	64	129	258	530	1196	2811
	CPU	0.01	0.04	0.33	3.36	39.00	358.21
	RES	6.28e−5	2.25e−5	1.21−5	3.04e−6	7.62e−7	1.90e−7
GS	IT	561	2391	10,145	†	†	†
	CPU	0.06	0.73	13.52			
	RES	1.95e−4	4.88e−5	1.22e−5			
ASOR	IT	73	129	383	1876	7271	†
	CPU	0.01	0.03	0.28	5.65	114.27	
	RES	9.76e−4	1.76e−4	6.08e−5	1.53e−5	3.78e−6	
AOSOR	IT	56	117	257	775	3749	12,501
($\omega_0=1.48$)	CPU	0.02	0.09	0.66	8.20	206.87	2746.77
	RES	1.69e−4	4.58e−5	1.17e−5	3.02e−6	7.62e−7	1.91e−7
AOSOR	IT	50	86	130	195	297	407
($\omega_0<0.98$)	CPU	0.02	0.07	0.32	2.00	16.21	87.48
	RES	1.74e−4	4.78e−5	1.21e−5	3.05e−6	7.59e−7	1.90e−7
PAOSOR	IT	51	92	152	172	413	904
	CPU	0.02	0.08	0.41	2.10	26.13	210.60
	RES	1.82e−4	4.78e−5	1.22e−5	3.00e−6	7.63e−7	1.90e−7

Table 2
Numerical results for $\xi=\zeta=0$, $\sigma=2.5$ ($\varepsilon=h^2$).

h^{-1}		32	64	128	256	512	1024
SOR	IT	51	122	256	512	1024	2048
	CPU	0.01	0.04	0.31	3.02	32.02	253.79
	RES	9.70e−4	2.23e−4	1.12e−5	8.19e−5	5.65e−6	3.74e−7
GS	IT	290	1257	5412	18,936	†	†
	CPU	0.03	0.37	6.75	116.48		
	RES	9.76e−4	2.44e−4	6.10e−5	4.00e−5		
ASOR	IT	72	129	329	1376	5086	†
	CPU	0.01	0.03	0.25	4.16	80.50	
	RES	9.25e−4	1.88e−4	5.94e−5	1.51e−5	3.77e−6	
AOSOR	IT	44	94	198	414	1304	7586
($\omega_0=1.6$)	CPU	0.01	0.07	0.60	4.70	73.04	1654.40
	RES	8.79e−4	2.29e−4	5.88e−5	1.53e−5	3.80e−6	9.54e−7
AOSOR	IT	35	62	109	186	469	1362
($\omega_0<0.98$)	CPU	0.01	0.05	0.28	2.01	26.06	299.30
	RES	9.39e−4	2.43e−4	6.03e−5	1.53e−5	3.81e−6	9.53e−7
PAOSOR	IT	37	68	106	228	311	686
	CPU	0.01	0.06	0.28	2.76	19.67	166.39
	RES	9.62e−4	2.38e−4	6.07e−5	1.52e−5	3.81e−6	9.54e−7

where

$$\mu_0 = 4\left(1+\sigma h^2\right), \quad \mu_1 = -\left(1-\frac{1}{2}\xi h\right), \quad \mu_2 = -\left(1-\frac{1}{2}\zeta h\right),$$
$$\eta_1 = -\left(1+\frac{1}{2}\xi h\right), \quad \eta_2 = -\left(1+\frac{1}{2}\zeta h\right).$$

Different ξ, ζ and σ result in different matrix A (see [1] for more details).

The numerical results are reported in Tables 1–3 for different ξ, ζ and σ. In the tables, "IT" represents iteration step, "CPU" denotes elapsed CPU time in seconds, and the symbol "†" indicates that the iteration does not converge after 20,000 steps.

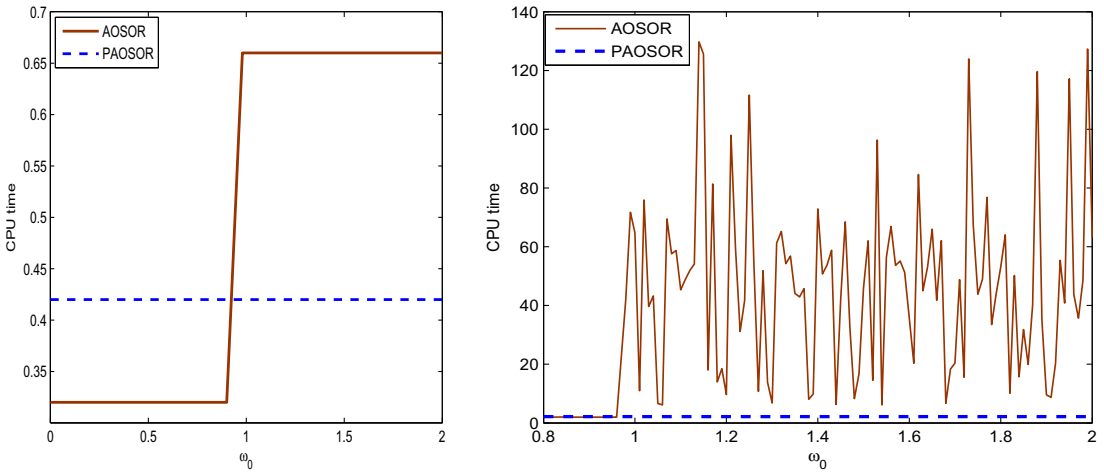
In Table 1 we show the computing results with $\xi = \zeta = \sigma = 0$. In this case, the matrix A is symmetric positive definite. The SOR method uses the optimal relaxation factor

$$\omega_{opt}^{(1)} = \frac{2}{1+sin(\pi h)}$$

by (1.5). From Table 1, we see that the iteration step for the PAOSOR method is less than both the classical SOR and ASOR [8] methods do to achieve the same computing accuracy. As for the AOSOR method, the rate of convergence depends on the choices of the initial guess ω_0 used in the Newton iteration. That is to say, if $\omega_0 \in (0, 0.98)$, the performance of AOSOR

Table 3
Numerical results for $\xi = 30,\ \zeta = 0,\ \sigma = 10$ ($\varepsilon = h^2$).

h^{-1}		32	64	128	256	512
SOR	IT	52	105	217	454	952
	CPU	0.01	0.04	0.27	2.77	30.10
	RES	8.67e−4	2.07e−4	5.93e−5	1.48e−5	3.75e−6
GS	IT	77	351	1517	6387	†
	CPU	0.01	0.11	1.82	38.30	
	RES	9.47e−4	2.43e−4	6.07e−5	1.52e−5	
ASOR	IT	191	206	191	248	5308
	CPU	0.02	0.04	0.14	0.80	84.01
	RES	8.50e−4	1.45e−4	5.10e−5	1.50e−5	3.74e−6
AOSOR	IT	21	71	378	671	2952
	CPU	0.02	0.05	0.50	4.11	92.62
	RES	7.21e−4	2.19e−4	6.04e−5	1.51e−5	3.80e−6
PAOSOR	IT	76	231	278	356	1196
	CPU	0.02	0.08	0.39	2.17	37.38
	RES	9.35e−4	2.37e−4	5.97e−5	1.49e−5	3.81e−6



(a) $h^{-1} = 128$ (b) $h^{-1} = 256$

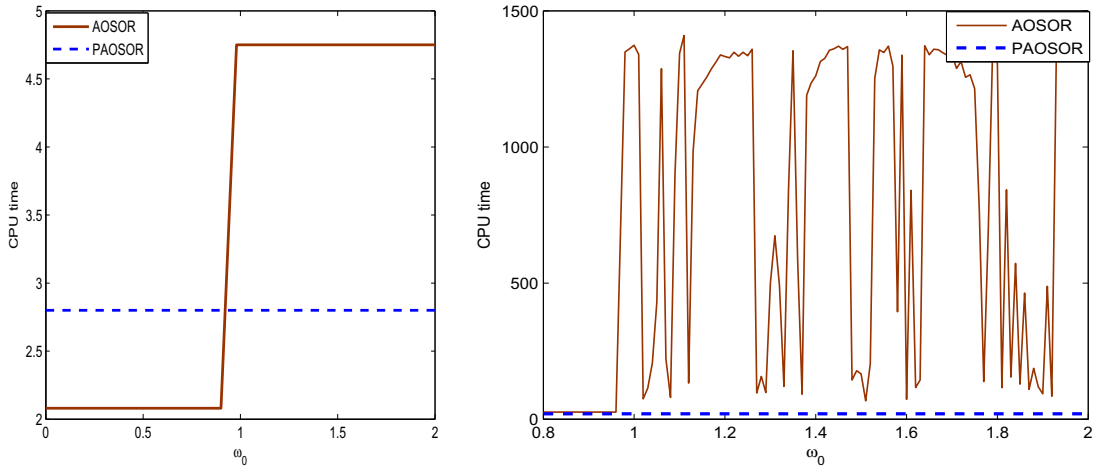
Fig. 1. Curves of the CPU time versus ω_0 for $\xi = \zeta = \sigma = 0$.

method is relatively stable, even better than the PAOSOR method in terms of iteration step and computing time. Fig. 1 clearly depicts this variation with respect to ω_0. At the same time we also notice that the graphs of the PAOSOR method are completely flat, which show that the convergence of PAOSOR method is independent of the choice of ω_0. This sensitivity is similar to the following Fig. 2. Besides, when h^{-1} increases, the number of iteration steps of the AOSOR method increases rapidly with respect to $\omega_0 \geqslant 0.98$. A possible illustration for this numerical phenomenon is that for the smaller step-size h, the variable relaxation factor ω becomes a constant after some iteration steps. Unless this constant ω is very close to ω_{opt}, the AOSOR method converges very slowly. In short, PAOSOR outperforms SOR and behaves much better than ASOR and AOSOR ($\omega_0 >= 0.98$) for large problem sizes.

In Table 2 we report results for SOR, GS, ASOR, AOSOR and PAOSOR methods with $\xi = \zeta = 0, \sigma = 2.5$. In this case, the matrix A is still symmetric positive definite. For the SOR method, we take the optimal relaxation factor

$$\omega_{opt}^{(2)} = \frac{2}{1+\sqrt{1-\frac{\cos^2 \pi h}{(1+\sigma h^2)^2}}}.$$

according to (1.5). Clearly, in iteration steps, the PAOSOR method is the best, and the GS method is the worst. In terms of computing times, PAOSOR costs less CPU time than SOR except for $h^{-1} = 64$, and more CPU time than AOSOR ($\omega_0 < 0.98$) only for $h^{-1} = 64$ and 256. Of course, it is also noted that ASOR costs least CPU time than other methods except for $h^{-1} > 128$, since it is updated after 5 iterative steps, whereas the PAOSOR/AOSOR method need to compute the asymptotical



(a) $h^{-1} = 256$ (b) $h^{-1} = 512$

Fig. 2. Curves of the CPU time versus ω_0 for $\xi = \zeta = 0,\ \sigma = 2.5$.

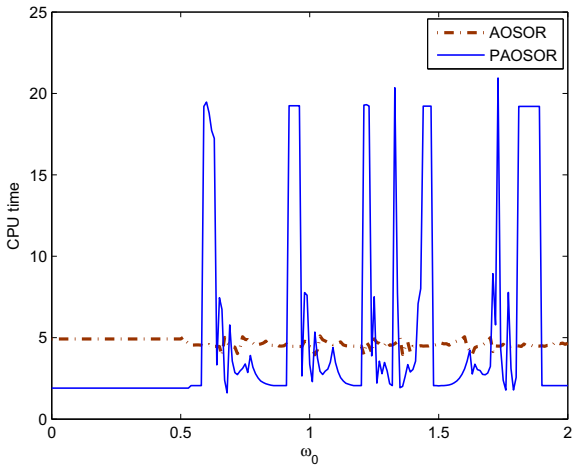


Fig. 3. Curves of the CPU time versus ω_0 for $\xi = 30,\ \zeta = 0,\ \sigma = 10$ with $h^{-1} = 256$.

optimal relaxation factor in every iterative step. Hence, with h^{-1} increasing, as iterative solvers PAOSOR is more outstanding than all other methods in terms of the iteration steps and the CPU time.

Finally, in Table 3 we show the numerical results for the nonsymmetric matrix A with $\xi = 30,\ \zeta = 0,\ \sigma = 10$. In this case, we do not have an analytical formula to compute the optimal relaxation factor, just like $\omega_{opt}^{(1)}$ or $\omega_{opt}^{(2)}$. So $\omega_{opt}^{(2)}$ is used as a good approximation to the exact optimal relaxation factor similar to [1]. We note that for more difficult problems, the ASOR approach becomes no longer robust, especially in terms of the iteration step. However, with h^{-1} increasing, the PAOSOR method outperforms AOSOR method, both in terms of CPU and in terms of iteration number. Similarly, except for $h^{-1} = 256$, the PAOSOR approach is nearly the same as the optimally tuned SOR method. Nevertheless, it should be pointed out that both AOSOR and PAOSOR methods are sensitive to the parameter ω_0. See Fig. 3. Although the curve of the AOSOR method changes relatively stable, CPU time of this method is longer than that of SOR method. And in the larger intervals of the parameter ω_0, we can observe that CPU time of the PAOSOR method is less than that of AOSOR, even less than that of SOR method.

4. Conclusion

In this paper, we have presented a new PAOSOR method for linear systems. Especially, the asymptotically optimal relaxation factor is discussed. The numerical experiments have shown that the PAOSOR method is especially efficient for the symmetric definite positive linear system. And it is important that the PAOSOR method does not rely on the initial guess of relaxation factor ω. Even though the coefficient matrix is nonsymmetric, the PAOSOR method can attain at almost the same effect as the optimal SOR method.

Acknowledgments

This work was finished when the author was visiting State Key Laboratory of Scientific and Engineering Computing, Institute of Computational Mathematics and Scientific/Engineering Computing, Academy of Mathematics and Systems Science, Chinese Academy of Sciences, PR China. Hence, the author is very much indebted to the warm hospitality and financial supports of the above institutions. Supported in part by The National Natural Science Foundation (No. 11371275), China.

References

[1] Z.-Z. Bai, X.-B. Chi, Asymptotically optimal successive overrelaxation methods for systems of linear equations, J. Comput. Math. 21 (2003) 603–612.
[2] Z.-Z. Bai, B.N. Parlett, Z.-Q. Wang, On generalized successive overrelaxation methods for augmented linear systems, Numer. Math. 102 (2005) 1–38.
[3] Z.-Z. Bai, G.H. Golub, M.K. Ng, On successive-overrelaxation acceleration of the Hermitian and skew-Hermitian splitting iterations, Numer. Linear Algebra Appl. 14 (2007) 319–335.
[4] M. Eiermann, R.S. Varga, Is the optimal best for the SOR iteration method?, Linear Algebra Appl 182 (1993) 257–277.
[5] G.H. Golub, X. Wu, J.-Y. Yuan, SOR-like methods for augmented systems, BIT Numer. Math. 41 (2001) 71–85.
[6] A. Hadjidimos, Successive overrelaxation (SOR) and related methods, J. Comput. Appl. Math. 123 (2000) 177–199.
[7] Y.-M. Huang, A practical formula for computing optimal parameters in the HSS iteration methods, J. Comput. Appl. Math. 255 (2014) 142–149.
[8] W.-W. Lin, Lecture Notes of Matrix Computations, National Tsinghua University, Hsinchu, Taiwan, 2008.
[9] R.S. Varga, Matrix Iterative Analysis, second ed., Springer, Berlin Heidelberg, Germany, 2000.
[10] C.-L. Wang, G.-Y. Meng, Parallel multisplitting two-stage iterative methods with general weighting matrices for non-symmetric positive definite systems, Appl. Math. Lett. 26 (2013) 1065–1069.
[11] R.-P. Wen, G.-Y. Meng, C.-L. Wang, Quasi-Chebyshev accelerated iteration methods based on optimization for linear systems, Comput. Math. Appl. 66 (2013) 934–942.