Numerical Algorithms (2022) 90:669–685
https://doi.org/10.1007/s11075-021-01203-9

ORIGINAL PAPER

On the generalized AOR and CG iteration methods for a class of block two-by-two linear systems

Fariba Bakrani Balani[1] · Masoud Hajarian[1]

Received: 19 March 2021 / Accepted: 14 September 2021 / Published online: 14 October 2021

Abstract
In this work, we utilize the generalized AOR (GAOR) and CG (GCG) methods for constructing iteration methods to solve the block two-by-two linear systems which arise from the solution of the complex symmetric linear systems of equations. In order to compare the GAOR and GCG methods with some existing methods, we present some numerical examples to illustrate the performance of these methods.

Keywords Complex symmetric linear systems · Block two-by-two matrix · GAOR method · GCG method · Preconditioner · Iteration methods

Mathematics Subject Classification (2010) 65F10

1 Introduction

This paper is devoted to study of the numerical solution of linear systems of the form

$$\mathcal{A}u = b, \quad \text{with} \quad \mathcal{A} = W + iT, \tag{1.1}$$

where $\mathcal{A} \in \mathbb{C}^{n\times n}$ is complex symmetric matrix, $W \in \mathbb{R}^{n\times n}$ and $T \in \mathbb{R}^{n\times n}$ are large and sparse symmetric positive definite matrices which imply that the complex symmetric matrix $\mathcal{A}$ is nonsingular. The right-hand side vector $b \in \mathbb{C}^n$ is given and $i = \sqrt{-1}$ denotes the imaginary unit. The linear systems (1.1) are widely used in sciences and engineering including diffuse optical tomography [1], electrical power system modeling [2], quantum mechanics [3], and molecular scattering [4]; see, e.g., [5–8]

✉ Masoud Hajarian
m_hajarian@sbu.ac.ir

Fariba Bakrani Balani
farbakrani@gmail.com

1 Department of Applied Mathematics, Faculty of Mathematical Sciences, Shahid Beheshti University, Tehran, Iran

and references therein for more details concerning the applications of this kind of linear systems.

To numerically approximate the solution of the complex symmetric linear systems (1.1), several iteration methods have been considered by many authors over the years. A number of efficient techniques based on the extension of Krylov subspace methods have been developed in the literature for solving (1.1), such as the conjugate orthogonal conjugate gradient (COCG) method [9], the quasi-minimal residual (QMR) method [10], the conjugate A-orthogonal conjugate residual (COCR) method [11], and the symmetric complex bi-conjugate gradient (SCBiCG) method [12, 13].

On the basis of the Hermitian and skew-Hermitian splitting (HSS) of the coefficient matrix, Bai et al. [14] established the HSS iteration method for solving the non-Hermitian positive definite linear systems. Any non-Hermitian matrix A can be decomposed into its Hermitian and skew-Hermitian parts as $A = H + S$ where

$$H = \frac{1}{2}(A + A^*), \qquad S = \frac{1}{2}(A - A^*). \tag{1.2}$$

The symbol A^*(or A^T) denotes the complex conjugate transpose of $A \in \mathbb{C}^{n\times n}$(or $A \in \mathbb{R}^{n\times n}$). The construction of the HSS iteration method is based on matrix splitting (1.2) similar in spirit to the classical alternating direction implicit (ADI) method. It is immediate to see that W and iT are the Hermitian and skew-Hermitian parts of the complex symmetric matrix $\mathcal{A}$, respectively. Then, the HSS method produces the approximate solutions of (1.1) with the following iteration scheme:

The HSS iteration method: Let α be a positive constant and I be the identity matrix. Given an initial guess $u^{(0)}$. For $k = 0, 1, 2, \ldots,$ until $u^{(k)}$ converges, compute

$$\begin{cases} (\alpha I + W)u^{(k+\frac{1}{2})} = (\alpha I - iT)u^{(k)} + b, \\ (\alpha I + iT)u^{(k+1)} = (\alpha I - W)u^{(k+\frac{1}{2})} + b. \end{cases} \tag{1.3}$$

At each iteration step of the HSS method, we need to solve the shifted skew-Hermitian linear subsystem with coefficient matrix $\alpha I + iT$, which may be problematic behavior in the actual implementations of this method. To avoid solving the shifted skew-Hermitian subsystem of linear equations, Bai et al. [15] introduced a modified HSS (MHSS) method which is much more efficient than the HSS method. To accelerate the convergence rate of the MHSS iteration method, Bai et al. [16] proposed a preconditioned variant of the MHSS (PMHSS) method. The PMHSS iteration scheme is given as follows:

The PMHSS iteration method: Let α be a positive constant and $V \in \mathbb{R}^{n\times n}$ be a symmetric positive definite matrix. Given an initial guess $u^{(0)}$. For $k = 0, 1, 2, \ldots,$ until $u^{(k)}$ converges, compute

$$\begin{cases} (\alpha V + W)u^{(k+\frac{1}{2})} = (\alpha V - iT)u^{(k)} + b, \\ (\alpha V + T)u^{(k+1)} = (\alpha V + iW)u^{(k+\frac{1}{2})} - ib. \end{cases} \tag{1.4}$$

We comment here that if the matrix V is equal to the identity matrix, then the PMHSS method reduces to MHSS method. To improve the computational efficiency of the MHSS and PMHSS methods, more practical iteration schemes for the complex symmetric linear systems (1.1) have been derived recently. By introducing another parameter in the PMHSS scheme, the generalized PMHSS (GMPHSS) iteration method was developed in [17] which is a kind of useful variant of the AHSS iteration method initially proposed and discussed in [18] and later in [19]; see also [20]. In addition, by applying the lopsided technique studied in [21], Li et al. [22] proposed the lopsided PMHSS (LPMHSS) iteration method. There are also other variants of the MHSS-like methods; see, e.g., [23–25]. Subsequently, more and more other iteration methods have been presented in the literature such as the skew-normal splitting (SNS) method [26], the Hermitian normal splitting (HNS) method [27], the scale-splitting (SCSP) iteration method [28], the double-step scale splitting (DSS) iteration method [29], and the combination of real part and imaginary part (CRI) iteration method [30]. Here, we should realize that one class of important and effective iteration methods, called the quasi-HSS (QHSS) iteration methods, was designed and analyzed recently in [31] also based on the HSS-like iteration methods. However, if we use these iteration methods for finding the solution of linear systems (1.1), we face complex arithmetic. To avoid complex arithmetic, the problem (1.1) can be recast in real formulation. Let $u = x+iy$ and $b = p+iq$, where the vectors $x,\ y,\ p,\ q$ are all in $\mathbb{R}^n$. Then, the complex linear systems (1.1) can be expressed as two-by-two block structure

$$\mathscr{A}z = \begin{pmatrix} W & -T \\ T & W \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} p \\ q \end{pmatrix} = f, \tag{1.5}$$

which can be solved in real arithmetics by HSS method or some Krylov subspace methods such as GMRES method.

For the block two-by-two linear systems (1.5), some efficient iteration methods have been investigated. For example, Bai et al. [32] developed the PMHSS iteration method for solving the block two-by-two linear systems (1.5) and applied it to solve distributed control problems. The block PMHSS iteration scheme is algorithmically described as follows:

The block PMHSS iteration method: Let $(x^{(0)}; y^{(0)}) \in \mathbb{R}^{2n}$ be an initial guess. Using the following iteration scheme, for $k = 0, 1, 2, \ldots,$ compute $\left\{(x^{(k+1)}; y^{(k+1)})\right\}$ until $\left\{(x^{(k)}; y^{(k)})\right\}$ converges:

$$\begin{cases} \begin{pmatrix} \alpha V + W & 0 \\ 0 & \alpha V + W \end{pmatrix} \begin{pmatrix} x^{(k+\frac{1}{2})} \\ y^{(k+\frac{1}{2})} \end{pmatrix} = \begin{pmatrix} \alpha V & T \\ -T & \alpha V \end{pmatrix} \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} + \begin{pmatrix} p \\ q \end{pmatrix}, \\ \begin{pmatrix} \alpha V + T & 0 \\ 0 & \alpha V + T \end{pmatrix} \begin{pmatrix} x^{(k+1)} \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} \alpha V & -W \\ W & \alpha V \end{pmatrix} \begin{pmatrix} x^{(k+\frac{1}{2})} \\ y^{(k+\frac{1}{2})} \end{pmatrix} + \begin{pmatrix} q \\ -p \end{pmatrix}, \end{cases} \tag{1.6}$$

where α is a given positive constant and $V \in \mathbb{R}^{n\times n}$ is a prescribed symmetric positive definite matrix.

In 2005, Bai et al. [20] first introduced the generalized successive overrelaxation (GSOR) method to find the solution of the augmented linear systems and discussed the optimal iteration parameters. Recently, Salkuyeh et al. [33] applied the GSOR method to the linear systems (1.5) and derived the GSOR iteration method for solving the complex linear systems which is defined in the following:

The GSOR iteration method: Let α be a positive constant and $(x^{(0)}; y^{(0)}) \in \mathbb{R}^{2n}$ be an initial guess. For $k = 0, 1, 2, \ldots,$ until the sequence of iterates $\left\{(x^{(k)}; y^{(k)})\right\}$ converges, compute

$$\begin{cases} Wx^{(k+1)} = (1-\alpha)Wx^{(k)} + \alpha T y^{(k)} + \alpha p, \\ Wy^{(k+1)} = -\alpha T x^{(k+1)} + (1-\alpha)Wy^{(k)} + \alpha q. \end{cases} \tag{1.7}$$

It is worth mentioning that the iteration scheme in (1.7) is a straightforward application of the GSOR iteration method for the saddle-point linear systems. Inspired by the ideas of the upper and lower triangular (ULT) iteration method and the parameterized ULT (PULT) iteration method in [34, 35], Li et al. [36] designed the symmetric block triangular splitting (SBTS) iteration method for solving (1.5). The SBTS iteration method can be described as follows:

The SBTS iteration method: Let α be a positive constant and $(x^{(0)}; y^{(0)}) \in \mathbb{R}^{2n}$ be an initial guess. For $k = 0, 1, 2, \ldots,$ until the sequence of iterates $\left\{(x^{(k)}; y^{(k)})\right\}$ converges, compute the next iteration according to the following procedures

$$\begin{cases} Wx^{(k+\frac{1}{2})} = Ty^{(k)} + p, \\ \alpha W y^{(k+\frac{1}{2})} = (\alpha-1)Wy^{(k)} - Tx^{(k+\frac{1}{2})} + q, \\ \alpha W y^{(k+1)} = (\alpha-1)Wy^{(k+\frac{1}{2})} - Tx^{(k+\frac{1}{2})} + q, \\ Wx^{(k+1)} = Ty^{(k+1)} + p. \end{cases} \tag{1.8}$$

By following the idea of the shift-splitting iterative and preconditioning methods [37], Zeng et al. have proposed the generalized shift-splitting iteration method for solving linear systems (1.5) in [38]. For more iteration methods and additional references, we refer to [39, 40].

A classical accelerated overrelaxation (AOR) method introduced by Hadjidimos (1978) [41] is a simple iteration method to solve the linear systems of equations, which is used in scientific computing and engineering applications. For instance, it has been used to solve augmented linear system or generalized saddle-point problems [42, 43]. In this paper, we employ the generalized AOR (GAOR) iteration method as a special case of PIU method in [44] and an extension of GSOR method in [20] to find the solution of the block two-by-two linear systems (1.5). We also consider the use of the generalized coujugate gradient (GCG) method of Concus and Golub [45] and Widlund [46] for problems of the form (1.5). Indeed, the ideas of the GAOR and the GCG methods are utilized to construct iteration schemes for solving the real equivalent linear systems in the next section. The remainder of this paper is organized as follows. In Section 3, some numerical experiments are tested to show the performance of these two methods. Section 4 is devoted to some brief conclusions.

2 The GAOR and GCG methods for block two-by-two linear systems

This section consists of two parts. In Section 2.1, the generalized AOR (GAOR) method for solving (1.5) is studied and its convergence properties are stated. The second subsection is concerned with the generalized coujugate gradient (GCG) method and it is applied to solve the linear systems (1.5).

2.1 The GAOR method

We consider the following splitting of the coefficient matrix $\mathscr{A}$ in (1.5)

$$\mathscr{A} = \begin{pmatrix} W & -T \\ T & W \end{pmatrix} = \mathscr{D} - \mathscr{L} - \mathscr{U}, \tag{2.1}$$

where

$$\mathscr{D} = \begin{pmatrix} W & 0 \\ 0 & W+Q \end{pmatrix}, \quad \mathscr{L} = \begin{pmatrix} 0 & 0 \\ -T & 0 \end{pmatrix}, \quad \mathscr{U} = \begin{pmatrix} 0 & T \\ 0 & Q \end{pmatrix}, \tag{2.2}$$

in which $Q \in \mathbb{R}^{n\times n}$ is given symmetric positive definite matrix. Let $z^{(k)} = (x^{(k)}; y^{(k)})$ be the kth approximation solution, then the following iteration scheme can be constructed

$$(\mathscr{D} - r\mathscr{L})z^{(k+1)} = [(1-\omega)\mathscr{D} + (\omega - r)\mathscr{L} + \omega\mathscr{U}]z^{(k)} + \omega f. \tag{2.3}$$

It is not difficult to see that (2.3) can be reformulated as follows

$$z^{(k+1)} = \mathcal{G}z^{(k)} + \omega(\mathscr{D} - r\mathscr{L})^{-1} f, \tag{2.4}$$

where

$$\mathcal{G} = \begin{pmatrix} (1-\omega)I & \omega W^{-1}T \\ \Phi & \Psi \end{pmatrix}, \tag{2.5}$$

in which $\Phi = \omega(r-1)(W+Q)^{-1}T$, $\Psi = -r\omega(W+Q)^{-1}TW^{-1}T + (W+Q)^{-1}[Q + (1-\omega)W]$ and ω, r are two positive parameters. Simple computations show that the iteration scheme (2.4) can be recast as follows:

The GAOR iteration method: Let ω and r be positive constants and $(x^{(0)}; y^{(0)}) \in \mathbb{R}^{2n}$ be an initial guess. For $k = 0, 1, 2, \ldots,$ until the iteration sequence $\left\{(x^{(k)}; y^{(k)})\right\}$ converges, compute

$$\begin{cases} x^{(k+1)} = (1-\omega)x^{(k)} + \omega W^{-1}(Ty^{(k)} + p), \\ y^{(k+1)} = y^{(k)} + (W+Q)^{-1}[T((r-\omega)x^{(k)} - rx^{(k)}) - \omega W y^{(k)} + \omega q]. \end{cases} \tag{2.6}$$

The GAOR method involves two parameters ω, r and one preconditioning matrix Q. The iteration method (2.6) reduces to that of (1.7) when $r = \omega$ and $Q = 0$. It is expected that by the proper choices of the parameters and preconditioning matrix, the GAOR method has faster convergence rate. As was said above, the GAOR method is a special case of PIU method [44] and a generalized inexact accelerated overrelaxation (GIAOR) method [20] (by replacing $P = W$ and $Q := W + Q$).

In the following discussion, we just state the convergence properties of the GAOR iteration method. The sketch of the proof of theorems can be found in [20]; we omit the details here.

Theorem 1 *Let W, Q and T be symmetric positive definite matrices. Assume that λ is an eigenvalue of the iteration matrix $\mathcal{G}$ and $z = (x; y)$ is the corresponding eigenvector. If $r = 1$, then $\lambda = 1 - \omega$ is an eigenvalue of $\mathcal{G}$ at least with multiplicity of n and the other eigenvalues satisfy*

$$\lambda = \frac{(\beta + \gamma) - \omega(\beta + r\alpha)}{\beta + \gamma}, \tag{2.7}$$

where

$$\alpha := \frac{y^T T W^{-1} T y}{y^T y}, \qquad \beta := \frac{y^T W y}{y^T y}, \qquad \gamma := \frac{y^T Q y}{y^T y}.$$

Corollary 1 *Let the conditions of Theorem 1 be satisfied. Then, the GAOR iteration method (2.4) is convergent if*

$$0 < \omega < \min\left\{2, \frac{2(\beta + \gamma)}{\beta + r\alpha}\right\}. \tag{2.8}$$

In the following theorem, the case $r \neq 1$ for the GAOR iteration method is considered.

Theorem 2 *Let W, Q and T be symmetric positive definite matrices. Assume that λ is an eigenvalue of the iteration matrix $\mathcal{G}$ and $z = (x; y)$ is the corresponding eigenvector. Then λ satisfies the following quadratic equation*

$$\lambda^2 - \frac{2(\gamma + \beta - \omega\beta) - \omega(\gamma + \alpha r)}{\beta + \gamma}\lambda + \frac{(1 - \omega)(\beta - \omega\beta + \gamma) + \omega\alpha(\omega - r)}{\beta + \gamma} = 0. \tag{2.9}$$

Theorem 3 *Let W, Q, and T be symmetric positive definite matrices. Then, the GAOR method is convergent if the following condition holds*

$$\frac{(\omega^2 - 2\omega)\beta + \omega^2\alpha - \omega\gamma}{\omega\alpha} < r < \frac{(\frac{\omega^2}{2} - 2\omega + 2)\beta + \frac{\omega^2}{2}\alpha + (2 - \omega)\gamma}{\omega\alpha}. \tag{2.10}$$

To implement all the methods mentioned in previous section efficiently, there is an important issue of how to choose the iteration parameters. The selection of the optimal value of iteration parameter that minimizes the spectral radius of the iteration matrix of the GSOR and SBTS methods depends on the eigenvalues μ_{max} and μ_{min} of $S = W^{-1}T$, which is not an easy task to be obtained when the size of $S = W^{-1}T$ is large enough. Hence, good estimates of the eigenvalues are required for the methods to be efficient. Besides, how to find the optimal parameters and the most efficient preconditioning matrix for the GAOR method are difficult and we leave this as a topic for further research. In the next subsection, we give a brief overview of generalized CG (GCG) iteration method and use this method to solve the linear systems of (1.5). The GCG method has the advantage that no priori information about the spectral radius of iteration matrix is needed for estimating parameters.

2.2 The GCG method

Concus and Golub [45] and Widlund [46] proposed the GCG iteration method for solving the systems of linear equations $Ax = b$, where A is an $n \times n$ real matrix and has positive definite symmetric part, i.e., $M = \frac{1}{2}(A + A^T)$ is positive definite. The matrix M is taken into account as a preconditioner for an associated conjugate gradient method. This method is quite interesting and valuable if the system $Mv = g$ can be solved with less computational effort than the original system $Ax = b$ and this limits its range of applicability. In this case, we can write A in the following form:

$$A = M - N, \tag{2.11}$$

where

$$M = \frac{1}{2}(A + A^T), \qquad N = \frac{1}{2}(A^T - A).$$

Concus and Golub [45] and Widlund [46] considered the following generalized CG procedure

$$x^{(k+1)} = x^{(k-1)} + \omega_{k+1}(v^{(k)} + x^{(k)} - x^{(k-1)}), \tag{2.12}$$

where $v^{(k)} = M^{-1}(b - Ax^{(k)})$ and

$$\omega_{k+1} = \begin{cases} 1 & if \;\; k = 0; \\ \left[1 + \frac{(Mv^{(k)}, v^{(k)})}{(Mv^{(k-1)}, v^{(k-1)})\omega_k}\right]^{-1} & if \;\; k \neq 0. \end{cases}$$

Therefore, the implementation of the GCG method for the splitting (2.11) is summarized as follows:

The GCG iteration method:

```
1. Let $x^{(0)}$ be given and set $x^{(-1)} = 0$.
2. For $k = 0, 1, \ldots,$ $until$ "$convergence$" $do$
    2.1. Solve $Mv^{(k)} = b - Ax^{(k)}$;
    2.2. Compute $\rho_k = (Mv^{(k)}, v^{(k)})$;
    2.3. $\omega_{k+1} = \begin{cases} 1 & if \;\; k = 0; \\ [1 + \frac{\rho_k}{\rho_{k-1}\omega_k}]^{-1} & if \;\; k \neq 0; \end{cases}$
    2.4. Compute $x^{(k+1)} = x^{(k-1)} + \omega_{k+1}(v^{(k)} + x^{(k)} - x^{(k-1)})$.
```

From the GCG algorithm, we can see that the cost per iteration is one matrix multiply (by A), one solve of system of the form $Mv = g$ and $2n$ multiplies. The GCG procedure converges to the true solution of (1.5) in a finite number of iterates in the absence of rounding errors. The reader is referred to the papers [46, 47] for further details about this method.

Now the GCG method is extended to a block version for solving linear systems (1.5). We consider the following splitting of the coefficient matrix $\mathscr{A}$ of (1.5)

$$\mathscr{A} = \mathscr{M} - \mathscr{N}, \tag{2.13}$$

where

$$\mathscr{M} = \frac{1}{2}(\mathscr{A} + \mathscr{A}^T) = \begin{pmatrix} W & 0 \\ 0 & W \end{pmatrix}, \qquad \mathscr{N} = \frac{1}{2}(\mathscr{A}^T - \mathscr{A}) = \begin{pmatrix} 0 & T \\ -T & 0 \end{pmatrix},$$

in which $\mathscr{M}$ and $\mathscr{N}$ are the symmetric and negetive skew-symmetric part of $\mathscr{A}$, respectively. It is clear that $\mathscr{M}$ is positive definite. Hence, the GCG method can be applied to the linear systems (1.5). Based on the splitting (2.13) and the idea of the GCG method, the following iteration scheme can be constructed to solve (1.5):

The GCG iteration method for real block linear systems:

```
1. Let $x^{(0)},\ y^{(0)}$ be given and set $x^{(-1)} = y^{(-1)} = 0$.
2. For $k = 0, 1, \ldots,$ until "convergence" do
    2.1. Solve $\mathscr{M}v^{(k)} = \begin{pmatrix} p \\ q \end{pmatrix} - \mathscr{A}\begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} \equiv r^{(k)}$;
    2.2. Compute $\rho_k = (\mathscr{M}v^{(k)}, v^{(k)})$;
    2.3. $\omega_{k+1} = \begin{cases} 1 & if\ \ k = 0; \\ [1 + \frac{\rho_k}{\rho_{k-1}\omega_k}]^{-1} & if\ \ k \neq 0; \end{cases}$
    2.4. Compute $\begin{pmatrix} x^{(k+1)} \\ y^{(k+1)} \end{pmatrix} = \begin{pmatrix} x^{(k-1)} \\ y^{(k-1)} \end{pmatrix} + \omega_{k+1}\left(v^{(k)} + \begin{pmatrix} x^{(k)} \\ y^{(k)} \end{pmatrix} - \begin{pmatrix} x^{(k-1)} \\ y^{(k-1)} \end{pmatrix}\right)$.
```

From the above algorithm, it is seen that the GCG method requires extra inner product per iteration step and needs to compute the initial residual vector at the starting of algorithm. However, the computational cost of the inner product ω_{k+1} may be increased. The GCG method is attractive by effectively solving an equation $\mathscr{M}v^{(k)} = r^{(k)}$ at each iteration and this subsystem of linear equations can be solved directly by some direct method, e.g., the Cholesky factorization of the coefficient matrix.

3 Numerical experiments

In this section, we discuss the numerical performance of the iteration methods described in the previous sections for solving systems of linear (1.5). Numerical results of the well-known GMRES method with $l = 30$ as the restart and the preconditioned GMRES(30) methods are also given. In the tables, the number of iteration steps, the elapsed CPU time in seconds and the relative residual error are denoted by IT, CPU and RES, respectively. For solving all linear subsystems involved in these iteration methods, we use the Cholesky factorization of the coefficient matrices. In all of the following experiments, the iteration is terminated when $||f - \mathscr{A}z^{(k)}||_2/||f||_2 < 10^{-6}$ or if the number of iteration steps exceed 3000. The starting vector is equal to $z^{(0)} = (x^{(0)}; y^{(0)}) = (0\,; 0)$. We comment that all the numerical experiments are implemented on a 64-bit 1.80 GHz core i7 processor and

12.00GB RAM using MATLAB version R2017a. The following examples are taken from [15, 22].

Example 1 Consider the complex symmetric linear systems of (1.1) as follows

$$\Big[\Big(K+\frac{3-\sqrt{3}}{\tau}I\Big)+i\Big(K+\frac{3+\sqrt{3}}{\tau}I\Big)\Big]u=b, \tag{3.1}$$

where τ is the time step-size and K is the five-point centered difference matrix approximating the negative Laplacian operator with homogeneous Dirichlet boundary conditions on a uniform mesh in the unit square $[0,1]\times[0,1]$ with the mesh-size $h=\frac{1}{m+1}$. The matrix $K\in\mathbb{R}^{n\times n}$ has the tensor product form $K=I_m\otimes V_m+V_m\otimes I_m$ with $V_m=h^{-2}tridiag(-1,2,-1)\in\mathbb{R}^{m\times m}$. Hence, K is a block tridiagonal matrix with $n=m^2$. In our tests, we take

$$W=K+\frac{3-\sqrt{3}}{\tau}I,\qquad T=K+\frac{3+\sqrt{3}}{\tau}I,$$

and the vector b is given with its jth entry:

$$b_j=\frac{(1-i)j}{\tau(j+1)^2},\quad j=1,2,\dots,n.$$

Furthermore, the linear system (3.1) is normalized by multiplying both sides with h^2 and we set $\tau=h$.

Example 2 Consider the following complex systems

$$\Big[(-\omega^2M+K)+i(\omega C_V+C_H)\Big]u=b, \tag{3.2}$$

where M and K are inertia and stiffness matrices, C_V and C_H are viscous and hysteretic damping matrices, respectively, and ω is a driving circular frequency constant. In this example, we take $M=I$, $C_V=10I$ and $C_H=\mu K$ with $\mu=0.02$ being a damping coefficient. The matrix K is defined analogously to Example 1. We also set $\omega=\pi$ and the right-hand side vector b is taken to be $b=(1+i)\mathcal{A}\mathbf{1}$ with $\mathbf{1}$ being the vector of all entries equal to one. In addition, the complex linear system (3.2) is normalized by multiplying both sides with h^2.

Example 3 Consider the linear systems of (1.1) with

$$T=I\otimes V+V\otimes I,\qquad W=10(I\otimes V_c+V_c\otimes I)+9(e_1e_m^T+e_me_1^T)\otimes I, \tag{3.3}$$

where $V=tridiag(-1,2,-1)\in\mathbb{R}^{m\times m}$, $V_c=V-e_1e_m^T-e_me_1^T\in\mathbb{R}^{m\times m}$, e_1 and e_m are the first and the last unit vectors in $\mathbb{R}^m$, respectively. We take the right-hand side vector b to be $b=(1+i)\mathcal{A}\mathbf{1}$ with $\mathbf{1}$ being the vector of all entries equal to one. Here, T and W correspond to the five-point centered difference matrices approximating the negative Laplacian operator with homogeneous Dirichlet boundary conditions and periodic boundary conditions, respectively, on a uniform mesh in the unit square $[0,1]\times[0,1]$ with the mesh size $h=\frac{1}{m+1}$.

Example 4 We consider the following complex Helmholtz equation

$$-\Delta u + \sigma_1 u + i\sigma_2 u = g, \tag{3.4}$$

where σ_1 and σ_2 are real coefficient functions and u satisfies Dirichlet boundary conditions in $D = [0, 1] \times [0, 1]$. By discretizing the problem with finite differences on a $m \times m$ grid with mesh size $h = \frac{1}{m+1}$, we obtain a system of linear equations

$$[(K + \sigma_1 I) + i\sigma_2 I]u = b, \tag{3.5}$$

where K is the five-point centered difference matrix approximating the negative Laplacian operator $-\Delta$. For this problem, K is defined the same as in Example 1. We also take the right-hand side vector $b = (1 + i)\mathcal{A}\mathbf{1}$, with $\mathbf{1}$ being the vector of all entries equal to one and normalized the system by multiplying both sides with h^2. In actual computations, we set $\sigma_1 = \sigma_2 = 100$.

According to [33, 36], the optimal parameters for the GSOR and SBTS methods are selected based on the following formulas

$$\alpha_{GSOR} = \frac{2}{1 + \sqrt{1 + \mu_{max}^2}}, \qquad \alpha_{SBTS} = \frac{\gamma \pm \sqrt{\gamma^2 - 2\gamma}}{2}, \tag{3.6}$$

where $\gamma = 2 + \mu_{max}^2 + \mu_{min}^2$, and μ_{max}, μ_{min} are the largest and the smallest eignvalues of $S = W^{-1}T$. In our implementations, the eigenvalues μ_{max} and μ_{min} are obtained by means of the power and inverse power methods, respectively. The parameter of PMHSS iteration method is chosen experimentally which minimizes the

Table 1 Choice of parameters α for tested methods

Example	Method	Grid					
		16×16	32×32	64×64	128×128	256×256	512×512
	PMHSS	1.4	1.4	1.4	1.4	1.4	1.4
No. 1	GSOR	0.552	0.497	0.459	0.437	0.424	0.418
	SBTS	0.531	0.524	0.520	0.518	0.517	0.516
	PMHSS	0.9	0.9	0.9	0.9	0.9	0.9
No. 2	GSOR	0.455	0.457	0.457	0.457	0.457	0.457
	SBTS	0.5217	0.5219	0.5219	0.5219	0.5219	0.5219
	PMHSS	0.5	0.5	0.5	0.5	0.5	0.5
No. 3	GSOR	0.908	0.776	0.566	0.354	0.199	0.105
	SBTS	0.700	0.610	0.539	0.511	0.503	0.503
	PMHSS	0.7	0.9	1	1	1	1
No. 4	GSOR	0.868	0.869	0.869	0.869	0.869	0.869
	SBTS	0.662	0.662	0.662	0.663	0.663	0.663

number of iteration steps. The optimal parameters of the PMHSS, GSOR, and SBTS iteration methods are listed in Table 1 for different values of m. The selection of the optimal parameters in the GAOR method that minimize the rate of convergence is a difficult task. In the GAOR iteration method, the parameters ω and r can be determined by trial and error on a small example and then used with good results on larger problems. In our computations, we take $\omega = 0.6,\ r = 0.9$ in Examples 1 and 2, $\omega = 0.3,\ r = 0.4$ in Example 3 and $\omega = 1,\ r = 0.95$ in Example 4. Moreover, in the block PMHSS and GAOR methods, the preconditioning matrices V and Q are chosen as W and $tW + T$, respectively. Here, t is a given real positive constant. It is clear that matrix $tW + T$ is symmetric positive definite and in actual computations, we set $t = 0.01$.

In Tables 2, 3, 4 and 5, the numerical results are presented for Examples 1–4. According to the exprerimental results for all examples, the GAOR method outperforms the PMHSS, GSOR, and SBTS methods in terms of IT, CPU time, and RES, especially when the size of problem increases (except for Example 3 that the iteration number for grids $m = 16,\ 32,\ 64$ with the estimation parameters are more than those of the other iteration methods). From the reported results in Tables 2–5, we can conclude that the iteration steps of the GAOR iteration method almost remain stable. Therefore, the GAOR iteration method is almost independent of the problem size. From the above results, we can observe that if suitable preconditioning matrix Q and numbers ω, r are chosen, the GAOR iteration method is comparable to the mentioned methods. A dash (–) in Tabel 4 means that the method not only has many iterations but also takes a long time, so no result is obtained.

We also give the numerical results for the GMRES(30), the PMHSS, GSOR, and GAOR preconditioned GMRES(30) (PGMRES(30)) methods and the GCG itera-

Table 2 Numerical results for Example 1

Method	m	16	32	64	128	256	512
	IT	21	21	21	21	21	21
PMHSS	CPU	0.0024	0.0143	0.0861	0.9483	6.3743	48.4967
	RES	7.8411e-07	8.1006e-07	8.2302e-07	8.314e-07	8.3772e-07	8.4202e-07
	IT	23	26	26	29	26	27
GSOR	CPU	0.00096	0.0052	0.0341	0.4485	2.6122	21.3205
	RES	5.5868e-07	5.7229e-07	7.2335e-07	6.7005e-07	8.4357e-07	8.1531e-07
	IT	22	31	39	45	48	52
SBTS	CPU	0.0015	0.01	0.0953	1.3802	9.5569	82.0648
	RES	8.1751e-07	8.8435e-07	9.6064e-07	8.3404e-07	9.2557e-07	8.0831e-07
	IT	18	18	18	18	19	19
GAOR	CPU	0.00079	0.0034	0.0286	0.3171	1.9561	14.9379
	RES	3.2286e-07	5.1142e-07	7.0632e-07	8.5241e-07	3.4763e-07	5.0741e-07

Table 3 Numerical results for Example 2

Method	m	16	32	64	128	256	512
	IT	34	37	38	38	38	38
PMHSS	CPU	0.0037	0.0276	0.1493	1.6453	11.392	88.0628
	RES	7.9625e-07	7.025e-07	7.2213e-07	8.2914e-07	8.6555e-07	8.7629e-07
	IT	26	28	25	23	23	23
GSOR	CPU	0.0009	0.005	0.033	0.3782	2.413	18.2146
	RES	7.525e-07	7.1347e-07	5.9843e-07	9.9998e-07	8.8207e-07	8.6732e-07
	IT	78	77	77	77	77	77
SBTS	CPU	0.005	0.0248	0.1798	2.3079	15.5466	120.7689
	RES	8.6203e-07	9.5208e-07	9.5976e-07	9.6362e-07	9.6455e-07	9.6476e-07
	IT	19	17	17	16	16	16
GAOR	CPU	0.0007	0.0034	0.0272	0.2699	1.63	12.6371
	RES	4.778e-07	9.5444e-07	4.7654e-07	8.2619e-07	7.7446e-07	7.6478e-07

tion method in Tables 6, 7, 8 and 9. As seen, the GMRES(30) method converges slowly or even fails to converge without preconditioning. A good preconditioner can be improved the convergence rate of this method. The PMHSS and GAOR preconditioners are defined by

$$\mathscr{P}_{PMHSS}=\frac{\alpha+1}{2\alpha}\begin{pmatrix} I & -I \\ I & I \end{pmatrix}\begin{pmatrix} \alpha W+T & 0 \\ 0 & \alpha W+T \end{pmatrix}, \qquad \mathscr{P}_{GAOR}=\begin{pmatrix} W & 0 \\ rT & W+Q \end{pmatrix}. \tag{3.7}$$

In all cases, we see that the use of these three preconditioners leads to better performance and reduced the number of iteration of the GMRES(30). In Table 8, dagger symbol (†) means that no convergence has been obtained. From Tables 6–9, we

Table 4 Numerical results for Example 3

Method	m	16	32	64	128	256	512
	IT	30	30	30	31	34	36
PMHSS	CPU	0.0045	0.0297	0.2307	2.2294	18.2848	162.4137
	RES	7.9808e-07	7.7843e-07	7.6963e-07	7.143e-07	7.8033e-07	9.5409e-07
	IT	7	11	20	44	71	131
GSOR	CPU	0.0005	0.0044	0.0501	1.1889	12.7732	183.3644
	RES	4.7156e-07	4.4784e-07	4.5276e-07	9.5091e-07	8.0566e-07	8.6007e-07
	IT	8	16	43	152	555	–
SBTS	CPU	0.0011	0.0095	0.2058	7.3294	197.8019	–
	RES	5.8114e-07	9.0157e-07	8.6624e-07	9.207e-07	9.8231e-07	–
	IT	44	46	45	43	43	43
GAOR	CPU	0.0023	0.0142	0.1268	1.1384	7.8505	61.0846
	RES	9.8519e-07	7.7518e-07	9.9824e-07	9.287e-07	9.5037e-07	7.1059e-07

Table 5 Numerical results for Example 4

Method	m	16	32	64	128	256	512
	IT	30	36	39	40	40	40
PMHSS	CPU	0.0026	0.0190	0.1531	1.798	12.1975	93.8366
	RES	7.1117e-07	8.4304e-07	8.0381e-07	8.0746e-07	9.0815e-07	9.4086e-07
	IT	9	9	8	8	7	7
GSOR	CPU	0.0003	0.0017	0.0104	0.1146	0.7274	5.6961
	RES	2.7578e-07	2.1774e-07	5.7026e-07	2.2682e-07	8.262e-07	6.8742e-07
	IT	10	10	10	10	10	10
SBTS	CPU	0.0006	0.0030	0.0247	0.2973	1.9744	15.805
	RES	7.3355e-07	8.9664e-07	9.9912e-07	9.3415e-07	9.3674e-07	9.371e-07
	IT	8	8	7	7	6	5
GAOR	CPU	0.0002	0.0016	0.0107	0.1020	0.6085	4.0305
	RES	1.9245e-07	9.9945e-08	4.1871e-07	1.5403e-07	4.6042e-07	6.2429e-07

Table 6 Numerical results of GMRES(30), PGMRES(30), and GCG methods for Example 1

Method	m	16	32	64	128	256	512
GMRES(30)	IT	8(24)	18(30)	42(19)	85(12)	139(11)	214(29)
	CPU	0.0252	0.0519	0.6687	3.5144	37.4451	299.143
PMHSS-GMRES(30)	IT	1(8)	1(10)	1(11)	1(11)	1(11)	1(11)
	CPU	0.0053	0.014	0.0778	0.4233	2.2343	11.4347
GSOR-GMRES(30)	IT	1(12)	1(14)	1(16)	1(18)	1(20)	1(21)
	CPU	0.008	0.0283	0.1206	0.7909	4.2298	20.9036
GAOR-GMRES(30)	IT	1(9)	1(10)	1(11)	1(12)	1(12)	1(13)
	CPU	0.0059	0.0151	0.0780	0.4680	2.4283	13.6663
GCG	IT	16	20	24	28	30	32
	CPU	0.0024	0.0061	0.0407	0.399	3.2813	25.8069

Table 7 Numerical results of GMRES(30), PGMRES(30), and GCG methods for Example 2

Method	m	16	32	64	128	256	512
GMRES(30)	IT	3(18)	9(5)	49(17)	130(19)	285(6)	894(20)
	CPU	0.0081	0.024	0.7570	5.4378	80.122	1288.1007
PMHSS-GMRES(30)	IT	1(13)	1(14)	1(14)	1(14)	1(14)	1(14)
	CPU	0.0057	0.022	0.1009	0.5389	2.7444	14.8162
GSOR-GMRES(30)	IT	1(8)	1(8)	1(8)	1(8)	1(8)	1(8)
	CPU	0.0035	0.0119	0.0552	0.3021	1.5162	7.8608
GAOR-GMRES(30)	IT	1(6)	1(6)	1(6)	1(6)	1(6)	1(6)
	CPU	0.0027	0.0094	0.0425	0.2357	1.1942	5.9924
GCG	IT	10	9	9	9	8	8
	CPU	0.0006	0.0024	0.0137	0.1383	0.8307	6.4152

Table 8 Numerical results of GMRES(30), PGMRES(30), and GCG methods for Example 3

Method	m	16	32	64	128	256	512
GMRES(30)	IT	159(12)	370(4)	1217(6)	†	†	†
	CPU	0.2352	1.0953	18.8767	†	†	†
PMHSS-GMRES(30)	IT	1(12)	1(14)	1(15)	1(15)	1(15)	1(16)
	CPU	0.0055	0.0271	0.1474	0.6990	3.1238	17.6173
GSOR-GMRES(30)	IT	1(12)	1(14)	1(16)	1(20)	1(23)	1(26)
	CPU	0.0066	0.0267	0.1541	0.8102	4.9566	28.576
GAOR-GMRES(30)	IT	1(12)	1(14)	1(18)	1(22)	1(26)	1(27)
	CPU	0.0051	0.0268	0.1818	0.9636	5.4925	30.0014
GCG	IT	8	10	12	15	22	34
	CPU	0.0005	0.004	0.0426	0.3666	3.8141	47.2333

observe that the iteration steps of the PMHSS preconditined GMRES(30) are stable, while the IT of the GSOR and GAOR preconditioned GMRES(30) are growing with the increase of m for Examples 1 and 3. We can also see that the GCG method converges to the true solution of (1.5) in acceptable iterations and the results for this method show that the IT and CPU timings are clearly much better than GMRES(30), especially for larger problems. In comparison with PGMRES(30) methods, the GCG method requires few iterations and computational costs for some of these examples. In general, with problem size increases, the IT of the GCG method remains almost constant or grows slowly. This shows the advantage of the GCG method in the solution of linear systems (1.5). Hence, the GCG method can be competitive and effective for solving some of these kinds of problems. We stress that it is possible that the GAOR and GCG methods are not competitive for all test problems considered here and they may turn out to be useful on some other problems.

Table 9 Numerical results of GMRES(30), PGMRES(30), and GCG methods for Example 4

Method	m	16	32	64	128	256	512
GMRES(30)	IT	2(3)	3(23)	6(23)	15(11)	30(19)	90(22)
	CPU	0.0027	0.0112	0.0877	0.5330	7.2855	111.8883
PMHSS-GMRES(30)	IT	1(15)	1(15)	1(16)	1(16)	1(16)	1(16)
	CPU	0.0065	0.0221	0.1243	0.6573	3.1068	16.2519
GSOR-GMRES(30)	IT	1(7)	1(7)	1(7)	1(7)	1(7)	1(7)
	CPU	0.0029	0.0104	0.0682	0.3165	1.3604	6.8777
GAOR-GMRES(30)	IT	1(6)	1(6)	1(6)	1(6)	1(6)	1(6)
	CPU	0.0025	0.0092	0.0427	0.2411	1.17	5.9263
GCG	IT	11	11	10	10	9	9
	CPU	0.0013	0.0027	0.0147	0.1480	0.9195	7.5259

4 Conclusions

In this paper, we have revisited the use of the GAOR and GCG methods for solving the real equivalent formulation of complex linear systems (1.1). We reported some numerical experiments to compare the performance of the GAOR and GCG with recently proposed iteration methods in the literature. The numerical experiments show that the GAOR method may be better than PMHSS, GSOR and SBTS methods if we choose the suitable value of parametrs and preconditioning matrix. Furthermore, the GAOR iteration method was applied as a preconditioner to accelerate the convergence rate of GMRES(30). We also employed the generalization of the CG (GCG) method for solving the linear systems of (1.5). As seen, the GCG method works fine when the systems of the form $Mv = g$ are esay to solve. Our experiments indicate that the GCG method can outperform in practice. It appears that the methods described here, possibly with some modification, may be applicable in at least some of these kind of problems and exploring these would be an interesting area for future research.

Acknowledgements The authors are very much indebted to the editor and reviewers for their constructive and valuable comments and suggestions which greatly improved the original manuscript of this paper.

References

1. Arridge, S.R.: Optical tomography in medical imaging. Inverse Probl. **15**, 41–93 (1999)
2. Howle, V.E., Vavasis, S.A.: An iterative method for solving complex-symmetric systems arising in electrical power modeling. SIAM J. Matrix Anal. Appl. **26**, 1150–1178 (2005)
3. van Dijk, W., Toyama, F.M.: Accurate numerical solutions of the time-dependent schrödinger equation. Phys. Rev. E. **75**, 1–10 (2007)
4. Poirier, B.: Efficient preconditioning scheme for block partitioned matrices with structured sparsity. Numer. Linear Algebra Appl. **7**, 715–726 (2000)
5. Axelsson, O., Kucherov, A.: Real valued iterative methods for solving complex symmetric linear systems. Numer. Linear Algebra Appl. **7**, 197–218 (2000)
6. Axelsson, O., Neytcheva, M., Ahmad, B.: A comparison of iterative methods to solve complex valued linear algebraic systems. Numer. Algorithm **66**, 811–841 (2014)
7. Benzi, M., Bertaccini, D.: Block preconditioning of real-valued iterative algorithms for complex linear systems. IMA J. Numer. Anal. **28**, 598–618 (2007)
8. Bai, Z.-Z.: Block preconditioners for elliptic PDE-constrained optimization problems. Computing **91**, 379–395 (2011)
9. van der Vorst, H.A., Melissen, J.B.M.: A Petrov-Galerkin type method for solving Ax = b, where A is symmetric complex. IEEE Trans. Magn. **26**, 706–708 (1990)
10. Freund, R.W.: Conjugate gradient-type methods for linear systems with complex symmetric coefficient matrices. SIAM J. Matrix Anal. Appl. **13**, 425–448 (1992)
11. Sogabe, T., Zhang, S.-L.: A COCR method for solving complex symmetric linear systems. J. Comput. Appl. Math. **199**, 297–303 (2007)
12. Bunse-Gerstner, A., Stöver, R.: On a conjugate gradient-type method for solving complex symmetric linear systems. Linear Algbra Appl. **287**, 105–123 (1999)
13. Gu, X.-M., Clemens, M., Huang, T.-Z.: The SCBiCG class of algorithms for complex symmetric linear systems with applications in several electromagnetic model problems. Comput. Phys. Commun. **191**, 52–64 (2015)
14. Bai, Z.-Z., Golub, G.H., Ng, M.K.: Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. SIAM J. Matrix Anal. Appl. **24**, 603–626 (2003)

15. Bai, Z.-Z., Benzi, M., Chen, F.: Modified HSS Iteration methods for a class of complex symmetric linear systems. Computing **87**, 93–111 (2010)
16. Bai, Z.-Z., Benzi, M., Chen, F.: On preconditioned MHSS iteration methods for complex symmetric linear systems. Numer. Algorithm **56**, 297–317 (2011)
17. Dehghan, M., Dehghani-Madiseh, M., Hajarian, M.: A generalized preconditioned MHSS method for a class of complex symmetric linear systems. Math. Model. Anal. **18**, 561–576 (2013)
18. Bai, Z.-Z., Golub, G.H.: Accelerated Hermitian and skew-Hermitian splitting iteration methods for saddle-point problems. IMA J. Numer. Anal. **27**, 1–23 (2007)
19. Bai, Z.-Z.: Optimal parameters in the HSS-like methods for saddle-point problems. Numer. Linear Algebra Appl. **16**, 447–479 (2009)
20. Bai, Z.-Z., Parlett, B.N., Wang, Z.-Q.: On generalized successive overrelaxation methods for augmented linear systems. Numer. Math. **102**, 1–38 (2005)
21. Li, L., Huang, T.-Z., Liu, X.-P.: Modified Hermitian and skew-Hermitian splitting methods for non-Hermitian positive-definite linear systems. Numer. Linear Algebra Appl. **14**, 217–235 (2007)
22. Li, X., Yang, A.-L., Wu, Y.-J.: Lopsided PMHSS Iteration method for a class of complex symmetric linear systems. Numer. Algorithm **66**, 555–568 (2014)
23. Zheng, Q.-Q., Ma, C.-F., Accelerated, P.MHSS.: Iteration methods for complex symmetric linear systems. Numer. Algorithm **73**, 501–516 (2016)
24. Yang, A.-L., Wu, Y.-J., Xu, Z.-J.: The semi-convergence properties of MHSS method for a class of complex nonsymmetric singular linear systems. Numer. Algorithm **66**, 705–719 (2014)
25. Cao, Y., Ren, Z.-R.: Two variants of the PMHSS iteration method for a class of complex symmetric indefinite linear systems. Appl. Math. Comput. **264**, 61–71 (2015)
26. Bai, Z.-Z.: Several splittings for non-Hermitian linear systems. Sci. Chin. Ser. A: Math. **51**, 1339–1348 (2008)
27. Wu, S.-L.: Several variants of the Hermitian and skew-Hermitian splitting method for a class of complex symmetric linear systems. Numer. Linear Algebra Appl. **22**, 338–356 (2015)
28. Hezari, D., Salkuyeh, D.K., Edalatpour, V.: A new iterative method for solving a class of complex symmetric system of linear equations. Numer. Algorithm **73**, 927–955 (2016)
29. Zheng, Z., Huang, F.-L., Peng, Y.-C.: Double-step scale splitting iteration method for a class of complex symmetric linear systems. Appl. Math. Lett. **73**, 91–97 (2017)
30. Wang, T., Zheng, Q.-Q., Lu, L.-Z.: A new iteration method for a class of complex symmetric linear systems. J. Comput. Appl. Math. **325**, 188–197 (2017)
31. Bai, Z.-Z.: Quasi-HSS iteration methods for non-Hermitian positive definite linear systems of strong skew-Hermitian parts. Numer. Linear Algebra Appl. **25**, e2116:1–19 (2018)
32. Bai, Z.-Z., Benzi, M., Chen, F., Wang, Z.-Q.: Preconditioned MHSS iteration methods for a class of block two-by-two linear systems with applications to distributed control problems. IMA J. Numer. Anal. **33**, 343–369 (2013)
33. Salkuyeh, D.K., Hezari, D., Edalatpour, V.: Generalized successive overrelaxation iterative method for a class of complex symmetric linear system of equations. Int. J. Comput. Math. **92**, 802–815 (2015)
34. Zheng, Q.-Q., Ma, C.-F.: A class of triangular splitting methods for saddle point problems. J. Comput. Appl. Math. **298**, 13–23 (2016)
35. Li, J.-T., Ma, C.-F.: The parameterized upper and lower triangular splitting methods for saddle point problems. Numer. Algorithm **76**, 1–13 (2017)
36. Li, X.-A., Zhang, W.-H., Wu, Y.-J.: On symmetric block triangular splitting iteration method for a class of complex symmetric system of linear equations. Appl. Math. Lett. **79**, 131–137 (2018)
37. Bai, Z.-Z., Yin, J.-F., Su, Y.-F.: A shift-splitting preconditioner for non-Hermitian positive definite matrices. J. Comput. Math. **24**, 539–552 (2006)
38. Zeng, M.-L., Zhang, G.-F.: Generalized shift-splitting iteration method for a class of two-by-two linear systems. J. Appl. Math. Comput. **53**, 1–13 (2015)
39. Liang, Z.-Z., Zhang, G.-F.: On SSOR iteration method for a class of block two-by-two linear systems. Numer. Algorithm **71**, 1–17 (2015)
40. Chen, C.-R., Ma, C.-F.: AOR-Uzawa iterative method for a class of complex symmetric linear system of equations. Comput. Math. Appl. **72**, 2462–2472 (2016)
41. Hadjidimos, A.: Accelerated overrelaxation method. Math. Comput. **32**, 149–157 (1978)
42. Li, C., Li, Z., Nie, Y.Y., Evans, D.J.: Generalized AOR method for the augmented system. Int. J. Comput. Math. **81**, 495–504 (2004)

43. Zhang, C.-H., Wang, X., Tang, X.-B.: Generalized AOR method for solving a class of generalized saddle point problems. J. Comput. Appl. Math. **350**, 69–79 (2019)
44. Bai, Z.-Z., Wang, Z.-Q.: On parameterized inexact Uzawa methods for generalized saddle point problems. Linear Algebra Appl. **428**, 2900–2932 (2008)
45. Concus, P., Golub, G.H.: A generalized conjugate gradient method for nonsymmetric systems of linear equations. In: Glowinski, R., Lions, J.L. (eds.) Computing Methods in Applied Sciences and Engineering, Lecture Notes in Economics and Mathematical Systems, vol. 134, pp. 56–65. Springer, Berlin (1976)
46. Widlund, O.: A Lanczos method for a class of unsymmetric systems of linear equations. SIAM J. Numer. Anal. **19**, 485–506 (1982)
47. Eisenstat, S.C.: A note on the generalized conjugate gradient method. SIAM J. Numer. Anal. **20**, 358–361 (1983)