Journal of Computational Mathematics
Vol.32, No.3, 2014, 312–331.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1401-CR6

THE GENERALIZED LOCAL HERMITIAN AND SKEW-HERMITIAN SPLITTING ITERATION METHODS FOR THE NON-HERMITIAN GENERALIZED SADDLE POINT PROBLEMS[*]

Hongtao Fan and Bing Zheng
School of Mathematics and Statistics, Lanzhou University, Lanzhou 730000, China
Email: bzheng@lzu.edu.cn fanht11@lzu.edu.cn

Abstract

For large and sparse saddle point problems, Zhu studied a class of generalized local Hermitian and skew-Hermitian splitting iteration methods for non-Hermitian saddle point problem [M.-Z. Zhu, Appl. Math. Comput. 218 (2012) 8816–8824]. In this paper, we further investigate the generalized local Hermitian and skew-Hermitian splitting (GLHSS) iteration methods for solving non-Hermitian generalized saddle point problems. With different choices of the parameter matrices, we derive conditions for guaranteeing the convergence of these iterative methods. Numerical experiments are presented to illustrate the effectiveness of our GLHSS iteration methods as well as the preconditioners.

Mathematics subject classification: 65F10, 65F50.
Key words: Generalized saddle point problems, Hermitian and skew-Hermitian matrix splitting, Iteration method, Convergence.

1. Introduction

Consider the following two-by-two block linear systems of the form

$$\begin{bmatrix} A & B^* \\ -B & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{1.1}$$

where $A \in \mathbb{C}^{n\times n}$, $B \in \mathbb{C}^{m\times n}$, $C \in \mathbb{C}^{m\times m}$, $x,\ f \in \mathbb{C}^n$, $y,\ g \in \mathbb{C}^m$ with $m \le n$. B^* denotes the conjugate transpose of the matrix B.

The two-by-two block linear system (1.1) is often called as a generalized saddle point problem with $C \neq O$ and a saddle point problem with $C = O$, which is important and arises in a large number of scientific and engineering applications, such as the field of computational fluid dynamics [17], mixed finite element approximations of elliptic partial differential equations [10], restrictively preconditioned conjugate gradient methods [26, 28], interior point methods in constrained optimization [7]. For more applications or a comprehensive survey one can refer to [9].

As is known, there are two kinds of methods to solve the linear systems: direct methods and iterative methods. Direct methods are widely employed when the size of the coefficient matrix is not too large, and are usually regarded as the robust methods. However, frequently, matrices A and B are large and sparse, so iterative methods, such as Uzawa type methods

[*] Received August 28, 2013 / Revised version received December 6, 2013 / Accepted January 15, 2014 / Published online May 22, 2014 /

[3, 6, 11, 12, 13, 15, 16, 19, 20, 22, 23, 31], HSS-type iteration methods [1, 2, 4, 5, 27], as well as preconditioned Krylov subspace iteration methods [14, 18], become more attractive than direct methods for solving the systems (1.1). In the case of A being a Hermitian positive definite, a large amount of efficient methods as well as their numerical properties have been studied in the literature. Bai et al. [2] established the Hermitian and skew-Hermitian splitting (HSS) iteration method and a class of preconditioned Hermitian and skew-Hermitian splitting (PHSS) iteration method [5] for general non-Hermitian linear system, which naturally result in the HSS-type iteration methods for solving the non-Hermitian(generalized) saddle point problems, i.e., the case of A being non-Hermitian. For example, Benzi and Golub [8] directly applied the HSS iteration technique to the non-Hermitian (generalized) saddle point problems. A preconditioned Hermitian and skew-Hermitian (PHSS) method for solving saddle point problem (1.1) was further presented by Bai et al. [5] and the accelerated Hermitian and skew-Hermitian (AHSS) splitting methods by Bai and Golub [1]. The more HSS-type iteration methods can be found in [21, 24, 25, 30]. Recently, Jiang et al. [21] proposed a local Hermitian and skew-Hermitian splitting (LHSS) iteration method and a modified LHSS iteration method (MLHSS) for the non-Hermitian saddle point problems with matrix $C = O$ by adding the proper parameter matrices to the Hermitian and skew-Hermitian parts of A and (2,2) zero block matrix in the split matrices, respectively. Furthermore, Zhu [32] investigated a generalized local Hermitian and skew-Hermitian splitting method (GLHSS) by adding one more parameter matrix to the (2,1) position in the first matrix of the splitting. These iteration methods can be applied to solve the non-Hermitian generalized saddle point problems with the restriction condition that C is Hermitian positive semi-definite by transforming the case of $C \neq O$ into its equivalent form of $C = O$. These mean that these methods are not appropriate for solving the non-Hermitian generalized saddle point problems with C being Hermitian positive definite. To bridge this gap, in this paper we use the same parameter matrices strategy as in [32] and propose a generalized local Hermitian and non-Hermitian splitting iteration method (still called GLHSS method) for the non-Hermitian generalized saddle point problems with the matrix C being Hermitian positive definite. The convergence properties are also discussed.

The rest of the paper is organized as follows. Section 2 gives the GLHSS iteration method for non-Hermitian generalized saddle point problems and the convergence properties are discussed under certain conditions for the case C being a special Hermitian positive definite. In Sections 3, we derive several algorithms by different choices of the parameter matrices. We simply describe the effectiveness of the GLHSS splitting presented in this paper as a preconditioner for preconditioned Krylov subspace methods such as GMRES method in Section 4. Section 5 provides some numerical experiments to illustrate our theory and some concluding remarks are given in Section 6.

2. The GLHSS Iteration Methods

Let $A = H + S$ be the Hermitian and skew-Hermitian splitting of A in which $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$ are the Hermitian and skew-Hermitian parts of A, respectively. A is called an non-Hermitian positive definite matrix if $H = \frac{1}{2}(A + A^*)$ is positive definite, or simply, A is positive definite. In particular, A is called being Hermitian dominant if $\|H\| > \|S\|$. From now on, unless otherwise stated, we always assume that the matrix A is positive and Hermitian dominant.

For the coefficient matrix in the two-by-two linear system (1.1), we make the following special splitting of :

$$\begin{bmatrix} A & B^* \\ -B & C \end{bmatrix} = \begin{bmatrix} Q_1 + H & 0 \\ -B + Q_3 & Q_2 \end{bmatrix} - \begin{bmatrix} Q_1 - S & -B^* \\ Q_3 & Q_2 - C \end{bmatrix}, \tag{2.1}$$

where the matrix $B \in \mathbb{C}^{m\times n}$ is of full row-rank, $Q_1 \in \mathbb{C}^{n\times n}$ is a Hermitian positive semi-definite matrix, $C \in \mathbb{C}^{m\times m}$ and $Q_2 \in \mathbb{C}^{m\times m}$ are Hermitian positive definite matrices, and $Q_3 \in \mathbb{C}^{m\times n}$ is an arbitrary matrix. Under these assumptions, the coefficient matrix of system (1.1) is nonsingular (see [8]) and the non-Hermitian generalized saddle point problem (1.1) has an unique solution. A generalized local Hermitian and skew-Hermitian (GLHSS) splitting iteration method can be derived by this special splitting as follows

$$\begin{bmatrix} Q_1 + H & 0 \\ -B + Q_3 & Q_2 \end{bmatrix} \begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} Q_1 - S & -B^* \\ Q_3 & Q_2 - C \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix} + \begin{bmatrix} f \\ g \end{bmatrix}. \tag{2.2}$$

The GLHSS iteration method (2.2) can be equivalently described as the following computational process:

Algorithm 2.1. *(GLHSS iteration method) Let $H = \frac{1}{2}(A + A^*)$ and $S = \frac{1}{2}(A - A^*)$ be respectively the Hermitian and skew-Hermitian parts of A. Assume that $Q_1 \in \mathbb{C}^{n\times n}$ is a Hermitian positive semi-definite matrix, $Q_2 \in \mathbb{C}^{m\times m}$ is a Hermitian positive definite matrix, and $Q_3 \in \mathbb{C}^{m\times n}$ is an arbitrary matrix. Given initial vector $\begin{pmatrix} x_0^T & y_0^T \end{pmatrix}^T \in C^{n+m}$. For $n = 0, 1, 2, \cdots$ until the iteration sequence $\{\begin{pmatrix} x_n^T & y_n^T \end{pmatrix}^T\}$ converges to the solution of the non-Hermitian generalized saddle point problem (1.1), compute*

$$\begin{cases} x_{n+1} = x_n + (Q_1 + H)^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + Q_2^{-1}((B - Q_3)x_{n+1} + Q_3 x_n - C y_n + g). \end{cases} \tag{2.3}$$

To deduce convergence properties of GLHSS iteration (2.3), we now consider the spectral radius, that is $\rho(\mathcal{T})$, of the iteration matrix $\mathcal{T}$ of the GLHSS method (2.3). The iteration matrix of the GLHSS iteration is given by

$$\mathcal{T} = \begin{bmatrix} Q_1 + H & 0 \\ -B + Q_3 & Q_2 \end{bmatrix}^{-1} \begin{bmatrix} Q_1 - S & -B^* \\ Q_3 & Q_2 - C \end{bmatrix}. \tag{2.4}$$

It is known that the GLHSS iteration method (2.3) converges if and only if $\rho(\mathcal{T}) < 1$.

Before investigating the conditions for $\rho(\mathcal{T}) < 1$ holds, we now describe some characters of eigenvalues and eigenvectors of the iteration matrix $\mathcal{T}$.

Lemma 2.1. *Let A be non-Hermitian positive definite and B be of full row rank. If λ is an eigenvalue of the iteration matrix $\mathcal{T}$ defined in* (2.4)*, then* $\lambda \neq 1$.

Proof. Let λ be an eigenvalue of $\mathcal{T}$ and $(u^*, v^*)^*$ be its corresponding eigenvector, where $u \in \mathbb{C}^n$ and $v \in \mathbb{C}^m$. Then we have:

$$\begin{bmatrix} Q_1 - S & -B^* \\ Q_3 & Q_2 - C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \lambda \begin{bmatrix} Q_1 + H & 0 \\ -B + Q_3 & Q_2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix},$$

or equivalently,

$$\begin{cases} [(\lambda-1)Q_1+\lambda H+S]u+B^*v=O, \\ [(\lambda-1)Q_3-\lambda B]u+[(\lambda-1)Q_2+C]v=O. \end{cases} \tag{2.5}$$

If $\lambda=1$, then (2.5) is reduced into

$$\begin{cases} Au+B^*v=O, \\ -Bu+Cv=O. \end{cases}$$

Since the coefficient matrix $\begin{bmatrix} A & B^* \\ -B & C \end{bmatrix}$ is nonsingular, the eigenvector $(u^*,v^*)^*$ is zero vector. This is a contradiction. □

Lemma 2.2. ([21]) *If S is a skew-Hermitian matrix, then $i\cdot S$ (i is the imaginary unit) is a Hermitian matrix and u^*Su is a purely imaginary number or zero for all $u\in\mathbb{C}^n$.*

Lemma 2.3. *Assume A is non-Hermitian positive definite and Hermitian dominant, and B is of full row rank. If $(u^*,v^*)^*$ is an eigenvector of the iteration matrix $\mathcal{T}$ corresponding to the eigenvalue λ, then $u\neq 0$. Furthermore, if A is Hermitian dominant and $v=0$, then $|\lambda|<1$.*

Proof. If $(u^*,v^*)^*$ is an eigenvector corresponding to the eigenvalue λ of the iteration matrix $\mathcal{T}$, then $u\neq 0$. In fact, if $u=0$, then $B^*v=0$. Since B^* is of full column rank, $B^*v=0$ implies $v=0$, which contradicts the assumption that $(u^*,v^*)^*$ is an eigenvector.

If $v=0$, from (2.5) we have

$$\begin{cases} (\lambda H+S+(\lambda-1)Q_1)u=0, \\ ((1-\lambda)Q_3+\lambda B)u=0. \end{cases} \tag{2.6}$$

Since $u\neq 0$, we can define

$$\lambda=a+ib,\ \alpha=\frac{u^*Hu}{u^*u}>0,\ \ -\beta=\frac{u^*iSu}{u^*u},\ \gamma=\frac{u^*Q_1u}{u^*u}\geq 0,$$

where i is imaginary unit. Then from the first equation in (2.6), we have

$$(a+bi)\alpha+i\beta+(a+bi-1)\gamma=0.$$

Note that $\alpha>0$ and $\gamma\geq 0$, it then follows that:

$$a=\frac{\gamma}{\alpha+\gamma}\geq 0,\quad b=\frac{-\beta}{\alpha+\gamma}.$$

Since the non-Hermitian matrix A is Hermitian dominant, we have $|\beta|<|\alpha|$ which immediately results in the following conclusion:

$$|\lambda|=|a+bi|=\sqrt{\frac{\beta^2+\gamma^2}{(\alpha+\gamma)^2}}<1.$$

This completes the proof of this lemma. □

Lemma 2.4. ([29]) *Both roots of the complex quadratic equation $\lambda^2+\phi\lambda+\varphi=0$ have modulus less than one if and only if $|\phi-\overline{\phi}\varphi|+|\varphi|^2<1$, where $\overline{\phi}$ denotes the conjugate complex of ϕ.*

The following theorem gives the conditions for guaranteeing the convergence of the GLHSS method. For completeness, this theorem includes the result for $C=0$ given in [32] as a special case.

Theorem 2.1. *Assume A be a non-Hermitian matrix with the positive definite Hermitian part $H=\frac{1}{2}(A+A^*)$ and the skew-Hermitian part $S=\frac{1}{2}(A-A^*)$. Let the matrix B have full row rank, Q_1 be Hermitian positive semi-definite, Q_2 be Hermitian positive definite, and $Q_3\in\mathbb{C}^{m\times n}$ be such that $B^*Q_2^{-1}Q_3$ is Hermitian. Assume $(u^*,v^*)^*$ is an eigenvector of the matrix $\mathcal{T}$ with $u\in\mathbb{C}^n$ and $v\in\mathbb{C}^m$ corresponding to the eigenvalue λ. Denote by*

$$\alpha=\frac{u^*Hu}{u^*u},\quad -\beta=\frac{u^*i\cdot Su}{u^*u},\quad \gamma=\frac{u^*Q_1u}{u^*u},$$
$$\eta=\frac{u^*B^*Q_2^{-1}Bu}{u^*u},\quad \tau=\frac{u^*B^*Q_2^{-1}Q_3u}{u^*u}.$$

Then we have the following conclusions:

1. *When $C=0$ [32], the GLHSS method is convergent if $\alpha,\beta,\gamma,\eta,\tau$ satisfy the following condition:*
$$0\le\eta\le\frac{2(\alpha-\tau)((\alpha-\tau)(\alpha+\tau+2\gamma)-\beta^2)}{(\alpha-\tau)^2+\beta^2}. \tag{2.7}$$

2. *When C is Hermitian positive definite, we can choose $Q_2=\frac{1}{\delta}C$ with $\delta\neq 0$ a real constant, i.e., $C=\delta Q_2$, and then the GLHSS method is convergent if α, β, γ, η and τ satisfy one of the following conditions:*

   (a) *If $Z=0$ and $Y<0$, then*
$$0\le\eta\le\frac{-Y}{X};$$

   (b) *If $Z>0$, $\triangle=Y^2-4XZ>0$ and $\frac{-Y}{2X}>0$, then*
$$\frac{-Y-\sqrt{\triangle}}{2X}\le\eta\le\frac{-Y+\sqrt{\triangle}}{2X};$$

   (c) *If $Z<0$ and $\triangle=Y^2-4XZ>0$, then*
$$0\le\eta\le\frac{-Y+\sqrt{\triangle}}{2X},$$

   where
$$X=E^2+(1-\delta)^2\beta^2; \tag{2.8a}$$
$$Y=2E[EF-(\delta-1)\beta^2]+2(1-\delta)\beta^2[G-F(\delta-1)]; \tag{2.8b}$$
$$Z=[FE-(\delta-1)\beta^2]^2+[G-F(\delta-1)]^2\beta^2-[GE-(\delta-1)^2\beta^2]^2; \tag{2.8c}$$
$$E=\alpha-\tau+\delta\gamma;\quad F=(\delta-1)\alpha+(\delta-2)\gamma-\tau; \tag{2.8d}$$
$$G=\alpha+\tau-(\delta-2)\gamma;\quad \textit{and}\quad 0<\delta<2. \tag{2.8e}$$

Proof. The conclusion for the case $C = 0$, its proof can be found in [32]. So we just need to prove the case $C = \delta Q_2$ with $\delta \neq 0$. Let λ be an eigenvalue of $\mathcal{T}$ and $(u^*, v^*)^*$ be the corresponding eigenvector. From (2.5), we have

$$\begin{cases}(\lambda H + S + (\lambda - 1)Q_1)u + B^*v = 0,\\ ((1-\lambda)Q_3 + \lambda B)u = ((\lambda - 1)Q_2 + C)v,\end{cases}$$

which, instead of C by $C = \delta Q_2$, becomes that

$$\begin{cases}(\lambda H + S + (\lambda - 1)Q_1)u + B^*v = 0,\\ ((1-\lambda)Q_3 + \lambda B)u = (\lambda - 1 + \delta)Q_2 v.\end{cases} \tag{2.9}$$

If $\lambda = 1 - \delta$, then the Eq. (2.9) leads to

$$\begin{cases}(\delta(Q_1 + H) - A)u = B^*v,\\ (\delta Q_3 + (1-\delta)B)u = 0,\end{cases}$$

or equivalently,

$$\begin{cases}u \in \operatorname{null}(\delta Q_3 + (1-\delta)B),\\ v = (B(H+Q_1)^{-1}B^*)^{-1}(\delta B - B(Q_1 + H)^{-1}A))u.\end{cases}$$

Here, null$(\cdot)$ is used to represent the null space of the corresponding matrix. If $\lambda \neq 1 - \delta$, from the second equality in (2.9), we have

$$v = \frac{1}{\lambda + \delta - 1} Q_2^{-1}((1-\lambda)Q_3 + \lambda B)u. \tag{2.10}$$

Substituting the expression (2.10) of vector v into the first equation in (2.9), we have

$$\Big(\lambda H + S + (\lambda - 1)Q_1\Big)u + \frac{1-\lambda}{\lambda + \delta - 1} B^* Q_2^{-1} Q_3 u + \frac{\lambda}{\lambda + \delta - 1} B^* Q_2^{-1} B u = 0. \tag{2.11}$$

If $Bu = 0$, then the equality (2.11) reduces to the following form

$$\Big(\lambda H + S + (\lambda - 1)Q_1\Big)u + \frac{1-\lambda}{\lambda + \delta - 1} B^* Q_2^{-1} Q_3 u = 0.$$

And then multiplying both sides of above equality from left with $\frac{u^*}{u^*u}$, we obtain the following equation

$$\frac{u^*(\lambda H + S + (\lambda - 1)Q_1)u}{u^*u} = 0.$$

Now using the same techniques as the proof of Lemma 2.3, we can easily know that $|\lambda| < 1$.

If $Bu \neq 0$, which means that $\eta > 0$. Multiplying both sides of equality (2.11) from left with $\frac{u^*}{u^*u}$ and then rearranging it as the quadratical equation in λ, we have

$$\lambda^2 + \frac{\eta - \alpha - 2\gamma + \delta\alpha + \delta\gamma - \tau + \beta i}{\alpha + \gamma}\lambda + \frac{\tau + \gamma - \delta\gamma + (\delta - 1)\beta i}{\alpha + \gamma} = 0. \tag{2.12}$$

Now, according to Lemma 2.4, we known that both roots of the complex Eq. (2.12) satisfy $|\lambda| < 1$ if and only if

$$\left|\frac{\eta - \alpha - 2\gamma + \delta\alpha + \delta\gamma - \tau + \beta i}{\alpha + \gamma} - \frac{\eta - \alpha - 2\gamma + \delta\alpha + \delta\gamma - \tau - \beta i}{\alpha + \gamma} \cdot \frac{\tau + \gamma - \delta\gamma + (\delta - 1)\beta i}{\alpha + \gamma}\right| \\ + \left|\frac{\tau + \gamma - \delta\gamma + (\delta - 1)\beta i}{\alpha + \gamma}\right|^2 < 1. \tag{2.13}$$

Through a more complicated process of computations, the inequality (2.13) can be finally simplified as the follow equivalent form

$$X\eta^2 + Y\eta + Z < 0, \tag{2.14}$$

where X, Y and Z are taken as in (2.8). The conclusions in part 2 of Theorem 2.1 now easily follow from (2.14). This completes the proof. $\square$

3. Several Algorithms

Under the assumption of Theorem 2.1, the generalized iterative method (2.3) leads to different iterative methods with different choices of the matrices Q_1,Q_2 and Q_3. In this section, we always assume $C = \delta Q_2$ ($\delta \neq 0$) and present some special GLHSS iteration methods which are stated in three parts depending on the three cases $Q_3 = 0$, $Q_3 = tB$ and $Q_3 = -tQ_2B$. The parameters α, β, γ, η and τ are same as those in Theorem 2.1.

Case I: $Q_3 = 0$. In this case, the GLHSS iterative scheme (2.3) becomes

$$\begin{cases} x_{n+1} = x_n + (Q_1 + H)^{-1}(f - Ax_n - B^*y_n), \\ y_{n+1} = y_n + Q_2^{-1}(Bx_{n+1} - Cy_n + g). \end{cases} \tag{3.1}$$

Algorithm 3.1. *Under the assumption of Theorem* 2.1*, if* $Q_1 = 0$,$Q_2 = \mu I_m$*, then the GLHSS method* (3.1) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + H^{-1}(f - Ax_n - B^*y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g), \end{cases} \tag{3.2}$$

and the method (3.2) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$*, then*

$$0 \leq \eta \leq \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$*, then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \leq \eta \leq \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$*, then*

$$0 \leq \eta \leq \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= \alpha^2 + (1-\delta)^2\beta^2, \\ Y &= 2\alpha(\delta - 1)(\alpha^2 - \beta^2) + 4\alpha\beta^2\delta(1-\delta)^2, \\ Z &= (\delta - 1)^2(\alpha^2 - \beta^2) + 4\alpha^2\beta^2\delta^2(1-\delta)^2 - [\alpha^2 - (\delta - 1)^2\beta^2]^2. \end{aligned}$$

Algorithm 3.2. *Under the assumption of Theorem 2.1, if $Q_1 = \omega I_n$, $Q_2 = \mu I_m$, then the GLHSS method* (3.1) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + (\omega I_n + H)^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g), \end{cases} \tag{3.3}$$

and the method (3.3) *is convergent provided that α, β, η satisfy one of the following conditions*

(a). *If $Z = 0$ and $Y < 0$, then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ and $\frac{-Y}{2X} > 0$, then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If $Z < 0$ and $\triangle = Y^2 - 4XZ > 0$, then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= (\alpha + \delta\omega)^2 + (1 - \delta)^2\beta^2, \\ Y &= 2(\alpha + \delta\omega)[(\alpha + \delta\omega)(\delta\alpha - \alpha + \delta\omega - 2\omega) - (\delta - 1)\beta^2] \\ &\quad + 2(1 - \delta)\beta^2(2\alpha - \delta\alpha - \delta\omega + 2\omega)\delta, \\ Z &= [(\delta\alpha - \alpha + \delta\omega - 2\omega)(\alpha + \delta\omega) - (\delta - 1)\beta^2]^2 + (2\alpha - \delta\alpha - \delta\omega + 2\omega)^2\delta^2\beta^2 \\ &\quad - [(\alpha - \delta\omega + 2\omega)(\alpha + \delta\omega) - (\delta - 1)^2\beta^2]^2. \end{aligned}$$

Algorithm 3.3. *Under the assumption of Theorem* 2.1, *if $Q_1 = \omega H$, $Q_2 = \mu I_m$, then the GLHSS method* (3.1) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + \frac{1}{1+\omega} H^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g), \end{cases} \tag{3.4}$$

and the method (3.4) *is convergent provided that α, β, η satisfy one of the following conditions*

(a). *If $Z = 0$ and $Y < 0$, then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ and $\frac{-Y}{2X} > 0$, then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= (\delta\omega\alpha + \alpha)^2 + (1-\delta)^2\beta^2, \\ Y &= 2(\delta\omega\alpha + \alpha)[(\delta\omega\alpha + \alpha)(\delta\alpha - \alpha + (\delta - 2)\omega\alpha) - (\delta - 1)\beta^2] \\ &\quad + 2(1-\delta)\beta^2(2\alpha - \delta\alpha - \delta\omega\alpha + 2\omega\alpha)\delta - (((\delta - 1) + (\delta - 2)\omega)\alpha)(\delta - 1)], \\ Z &= [(\delta\alpha - \alpha + (\delta\omega\alpha - 2\omega\alpha)(\delta\omega\alpha + \alpha) - (\delta - 1)\beta^2]^2 \\ &\quad + (2\alpha - \delta\alpha - \delta\omega\alpha + 2\omega\alpha)^2\delta^2\beta^2 \\ &\quad - [(\alpha - \delta\omega\alpha - 2\omega\alpha)(\delta\omega\alpha + \alpha) - (\delta - 1)^2\beta^2]^2. \end{aligned}$$

Case II: $Q_3 = tB, t \neq 0$. In this case, the GLHSS iterative scheme (2.3) becomes

$$\begin{cases} x_{n+1} = x_n + (Q_1 + H)^{-1}(f - Ax_n - B^*y_n), \\ y_{n+1} = y_n + Q_2^{-1}((1-t)Bx_{n+1} + tBx_n - Cy_n + g). \end{cases} \tag{3.5}$$

Algorithm 3.4. *Under the assumption of Theorem* 2.1, *if* $Q_1 = 0$, $Q_2 = \mu I_m$, *then the GLHSS method* (3.5) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + H^{-1}(f - Ax_n - B^*y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}((1-t)Bx_{n+1} + tBx_n - Cy_n + g), \end{cases} \tag{3.6}$$

and the method (3.6) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= (\alpha - t\eta)^2 + (1-\delta)^2\beta^2, \\ Y &= 2(\alpha - t\eta)[(\alpha - t\eta)(\delta\alpha - \alpha - t\eta) - (\delta - 1)\beta^2] + 2(1-\delta)\beta^2(2\alpha - \delta\alpha + t\eta)\delta, \\ Z &= [(\delta\alpha - \alpha - t\eta)(\alpha - t\eta) - (\delta - 1)\beta^2]^2 + (2\alpha - \delta\alpha + t\eta)^2\delta^2\beta^2 \\ &\quad - [\alpha^2 - t^2\eta^2 - (\delta - 1)^2\beta^2]^2. \end{aligned}$$

Algorithm 3.5. *Under the assumption of Theorem* 2.1, *if* $Q_1 = \omega I_n$, $Q_2 = \mu I_m$, *then the GLHSS method* (3.5) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + (\omega I_n + H)^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}((1-t)Bx_{n+1} + tBx_n - Cy_n + g), \end{cases} \tag{3.7}$$

and the method (3.7) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= (\alpha - t\eta + \delta\omega)^2 + (1-\delta)^2\beta^2, \\ Y &= 2(\alpha - t\eta + \delta\omega)[(\alpha - t\eta + \delta\omega)(\delta\alpha - \alpha + \delta\omega - 2\omega - t\eta) - (\delta - 1)\beta^2] \\ &\quad + 2(1-\delta)\beta^2(2\alpha - \delta\alpha - \delta\omega + 2\omega + t\eta)\delta, \\ Z &= [((\delta-1)\alpha + (\delta-2)\omega - t\eta)(\alpha - t\eta + \delta\omega) - (\delta-1)\beta^2]^2 \\ &\quad + (2\alpha - \delta\alpha - \delta\omega + 2\omega + t\eta)^2\delta^2\beta^2 - [(\alpha + t\eta - \delta\omega + 2\omega)(\alpha - t\eta + \delta\omega) - (\delta-1)^2\beta^2]^2. \end{aligned}$$

Algorithm 3.6. *Under the assumption of Theorem* 2.1, *if* $Q_1 = \omega H$, $Q_2 = \mu I_m$, *then the GLHSS method* (3.5) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + \frac{1}{1+\omega}H^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}((1-t)Bx_{n+1} + tBx_n - Cy_n + g), \end{cases} \tag{3.8}$$

and the method (3.8) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned}
&X = ((\delta\omega + 1)\alpha - t\eta)^2 + (1-\delta)^2\beta^2,\\
&Y = 2(\delta\omega\alpha + \alpha - t\eta)[(\delta\omega\alpha + \alpha - t\eta)(\delta\alpha - \alpha + (\delta - 2)\omega\alpha - t\eta) - (\delta - 1)\beta^2]\\
&\quad + 2(1-\delta)\beta^2(2\alpha - \delta\alpha - \delta\omega\alpha + 2\omega\alpha + t\eta)\delta,\\
&Z = [(\delta\alpha - \alpha + (\delta-2)\omega\alpha - t\eta)(\delta\omega\alpha + \alpha - t\eta) - (\delta-1)\beta^2]^2 + (2\alpha - \delta\alpha - \delta\omega\alpha\\
&\qquad + 2\omega\alpha + t\eta)^2\delta^2\beta^2 - [(\alpha - (\delta-2)\omega\alpha + t\eta)(\delta\omega\alpha + \alpha - t\eta) - (\delta-1)^2\beta^2]^2.
\end{aligned}$$

Case III: $Q_3 = -tQ_2B,\ t \neq 0$. In this case, the GLHSS iterative scheme (2.3) becomes

$$\begin{cases} x_{n+1} = x_n + (Q_1 + H)^{-1}(f - Ax_n - B^*y_n),\\ y_{n+1} = y_n + Q_2^{-1}(Bx_{n+1} - Cy_n + g) + tB(x_{n+1} - x_n). \end{cases} \tag{3.9}$$

Algorithm 3.7. *Under the assumption of Theorem* 2.1, *if* $Q_1 = 0,\ Q_2 = \mu I_m$, *then the GLHSS method* (3.9) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + H^{-1}(f - Ax_n - B^*y_n),\\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g) + tB(x_{n+1} - x_n), \end{cases} \tag{3.10}$$

and the method (3.10) *is convergent provided that* $\alpha,\ \beta,\ \eta$ *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned}
&X = (\alpha + t\eta)^2 + (1-\delta)^2\beta^2,\\
&Y = 2(\alpha + t\eta)[(\alpha + t\eta)(\delta\alpha - \alpha + t\eta) - (\delta-1)\beta^2] + 2(1-\delta)\beta^2\delta(2\alpha - \delta\alpha - t\eta),\\
&Z = [(\delta\alpha - \alpha + t\eta)(\alpha + t\eta) - (\delta-1)\beta^2]^2 + (2\alpha - \delta\alpha - t\eta)^2\delta^2\beta^2\\
&\quad - [\alpha^2 - t^2\eta^2 - (\delta-1)^2\beta^2]^2.
\end{aligned}$$

Algorithm 3.8. *Under the assumption of Theorem* 2.1, *if* $Q_1 = \omega I_n$,$Q_2 = \mu I_m$, *then the GLHSS method* (3.9) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + (\omega I_n + H)^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g) + tB(x_{n+1} - x_n), \end{cases} \tag{3.11}$$

and the method (3.11) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned} X &= (\alpha + \delta\omega + t\eta)^2 + (1-\delta)^2\beta^2, \\ Y &= 2(\alpha + \delta\omega + t\eta)[(\alpha + \delta\omega + t\eta)(\delta\alpha - \alpha + \delta\omega - 2\omega + t\eta) - (\delta - 1)\beta^2] \\ &\quad + 2(1-\delta)\beta^2(2\alpha - \delta\alpha - \delta\omega + 2\omega - t\eta)\delta, \\ Z &= [(\delta\alpha - \alpha + \delta\omega - 2\omega + t\eta)(\alpha + \delta\omega + t\eta) - (\delta - 1)\beta^2]^2 \\ &\quad + (2\alpha - \delta\alpha - \delta\omega + 2\omega - t\eta)^2\delta^2\beta^2 - [(\alpha - \delta\omega + 2\omega - t\eta)(\alpha + \delta\omega + t\eta) - (\delta - 1)^2\beta^2]^2. \end{aligned}$$

Algorithm 3.9. *Under the assumption of Theorem* 2.1, *if* $Q_1 = \omega H$, $Q_2 = \mu I_m$, *then the GLHSS method* (3.9) *becomes the following iteration method:*

$$\begin{cases} x_{n+1} = x_n + \frac{1}{1+\omega}H^{-1}(f - Ax_n - B^* y_n), \\ y_{n+1} = y_n + \frac{1}{\mu}(Bx_{n+1} - Cy_n + g) + tB(x_{n+1} - x_n), \end{cases} \tag{3.12}$$

and the method (3.12) *is convergent provided that* α, β, η *satisfy one of the following conditions*

(a). *If* $Z = 0$ *and* $Y < 0$, *then*

$$0 \le \eta \le \frac{-Y}{X};$$

(b). *If* $Z > 0$, $\triangle = Y^2 - 4XZ > 0$ *and* $\frac{-Y}{2X} > 0$, *then*

$$\frac{-Y - \sqrt{\triangle}}{2X} \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X};$$

(c). *If* $Z < 0$ *and* $\triangle = Y^2 - 4XZ > 0$, *then*

$$0 \le \eta \le \frac{-Y + \sqrt{\triangle}}{2X},$$

where

$$\begin{aligned}
X &= (\alpha + t\eta + \delta\omega\alpha)^2 + (1-\delta)^2\beta^2,\\
Y &= 2(\alpha + t\eta + \delta\omega\alpha)[(\alpha + t\eta + \delta\omega\alpha)(\alpha\delta - \alpha + \delta\omega\alpha - 2\omega\alpha + t\eta) - (\delta - 1)\beta^2]\\
&\quad + 2(1-\delta)\beta^2(2\alpha - \alpha\delta - \delta\omega\alpha + 2\omega\alpha - t\eta)\delta,\\
Z &= [(\alpha\delta - \alpha + \delta\omega\alpha - 2\omega\alpha + t\eta)(\alpha + t\eta + \delta\omega\alpha) - (\delta - 1)\beta^2]^2\\
&\quad + (2\alpha - \alpha\delta - \delta\omega\alpha + 2\omega\alpha - t\eta)^2\delta^2\beta^2\\
&\quad - [(\alpha - \delta\omega\alpha + 2\omega\alpha - t\eta)(\alpha + t\eta + \delta\omega\alpha) - (\delta - 1)^2\beta^2]^2.
\end{aligned}$$

4. Preconditioner for Preconditioned Krylov Subspace Methods

The generalized local Hermitian and skew-Hermitian splitting (2.1) also provides a class of preconditioners for preconditioned Krylov subspace methods, especially for the preconditioned GMRES method. If we write the GLHSS splitting (2.1) as $\mathcal{A} = \mathcal{M} - \mathcal{N}$, then $\mathcal{M}$ can be used as a preconditioner for GMRES [24] or any other non-Hermitian Krylov method. The convergence rate of the preconditioned GMRES method depends on the particular choice of the parameters and parameter matrices. How to find the values of the parameters and the parameter matrices that optimize the rate of convergence appears to be a difficult problem in general. In fact, in practice the convergence rate depends, to a large extent, on the size, shape, and location of the entire spectrum of the preconditioned matrix $\mathcal{M}^{-1}\mathcal{A}$. So in this paper we also illustrate the spectrum distribution of the preconditioned matrix $\mathcal{M}^{-1}\mathcal{A}$ by numerical experiments and show the effectiveness of GLHSS splitting (2.1) as a preconditioner for preconditioned GMRES method (PGMRES). The restarted PGMRES(m) algorithm with restart parameter m is used. Comparing with restarted GMRES(m) method for original system, the computational results of PGMRES(m) are much better than those without preconditioning, see Tables 5.5-5.6 and Figure 5.1-5.3.

5. Numerical Results

In this section, we use the numerical example to illustrate the theoretical results and show the effectiveness of the GLHSS preconditioning matrix for solving non-Hermitian generalized saddle point problem in the sense of iteration step (denoted as "IT"), elapsed CPU time in seconds (denoted as "CPU"), and relative residual error (denoted as "RES") defined by

$$\text{RES} := \frac{\sqrt{\| f - Ax^{(k)} - B^T y^{(k)} \|_2^2 + \| g + Bx^{(k)} - Cy^{(k)} \|_2^2}}{\sqrt{\| f \|_2^2 + \| g \|_2^2}}.$$

We consider the non-Hermitian generalized saddle point-type matrix

$$\mathcal{A} = \begin{bmatrix} A & B^* \\ -B & C \end{bmatrix}, \tag{5.1}$$

where the sub-matrices $A = \nu A_1 + N$, N has only two diagonal lines of nonzero which start from the 2nd and the nth columns, i.e.,

$$N = \begin{pmatrix} 0 & -1 & 0 & \cdots & 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & -1 & \cdots & 0 & 0 & -1 & \ddots & 0 \\ 0 & 0 & 0 & -1 & 0 & \cdots & 0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \cdots & \ddots & -1 \\ 0 & \cdots & 0 & 0 & 0 & -1 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & -1 & 0 & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & -1 & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 & \ddots & 0 & \ddots & -1 \\ 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix},$$

$$A_1 = \begin{bmatrix} I \otimes T & 0 \\ 0 & I \otimes T + I \otimes T \end{bmatrix} \in \mathbb{R}^{2p^2 \times 2p^2},$$

$$B = \begin{bmatrix} I \otimes F \\ F \otimes I \end{bmatrix} \in \mathbb{R}^{p^2 \times 2p^2}, \quad C = I \in \mathbb{R}^{p^2 \times p^2},$$

where $T = \frac{1}{h^2}\text{tridiag}(-1, 2, -1) \in \mathbb{R}^{p \times p}$, $F = \frac{1}{h}\text{tridiag}(-1, 1, 0) \in \mathbb{R}^{p \times p}$ with $\otimes$ being the Kronecker product symbol, $h = \frac{1}{p+1}$ the discretization meshsize. We set $m = p^2$ and $n = 2p^2$, hence, the total number of variables is $m + n = 3p^2$.

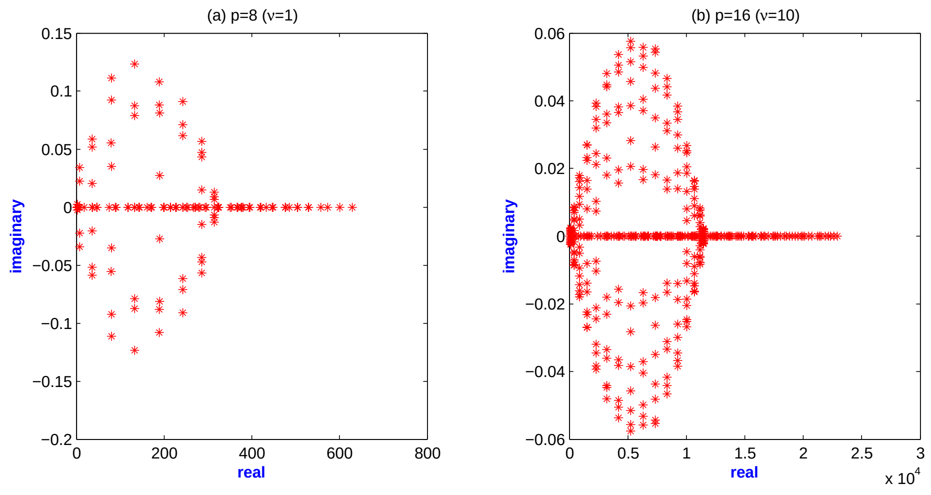

Fig. 5.1. The eigenvalue distribution of original matrix $\mathcal{A}$ for the Example.

In our computations, all runs are implemented in MATLAB (version 7.11.0.584 (R2010b)) with a machine precision 10^{-16} on a personal computer with AMD Athlon(tm) II P360 Dual-core Processor 2.30 GHZ, 2.00GB memory and are started from the initial vector $(x_0^T, y_0^T)^T = 0$, and terminated if the current iteration satisfies either RES$< 10^{-5}$ or the number of the prescribed iteration $\kappa_{\max} = 1000$ are exceed, and choose the right-hand side vector of $(f^*, g^*)^*$ such that the exact solution of the saddle point problem is $(x^*, y^*)^* = (1, 1, \cdots, 1)^*$.

Firstly, we use GLHSS method (Case I, II, III) with different choices of ω, μ and t to solve the example. Numerical results are listed in the following Tables 5.1–5.4 for $\nu = 1$ and $\nu = 10$,

Table 5.1: Numerical results for GLHSS iteration methods for Cases I, II and III for example for $p = 4$ and $p = 8$ ($\nu = 1$).

		$p = 4$				$p = 8$			
Case		(ω , μ t)	IT	RES	CPU	(ω , μ t)	IT	RES	CPU
I	3.1	(-,1.58,-)	8	5.2217×10^{-6}	0.0045	(-,1.52,-)	8	6.6737×10^{-6}	0.0122
	3.2	(0.01,1.93,-)	10	6.8236×10^{-6}	0.0034	(0.01,1.92,-)	11	6.7401×10^{-6}	0.0303
	3.3	(0.01,1.91,-)	10	5.4244×10^{-6}	0.0039	(0.01,1.90,-)	11	5.8429×10^{-6}	0.0340
II	3.4	(-,1.78,0.1)	9	8.4398×10^{-6}	0.0038	(-,1.72,0.1)	10	3.4360×10^{-6}	0.0141
	3.5	(0.1,1.76,0.1)	9	9.6860×10^{-6}	0.0039	(0.01,1.71,0.1)	10	3.4173×10^{-6}	0.0195
	3.6	(0.1,1.84,0.1)	11	2.7887×10^{-6}	0.0039	(0.01,1.88,0.1)	11	4.9835×10^{-6}	0.0269
III	3.7	(-,1.98,0.01)	11	3.8979×10^{-6}	0.0037	(-,1.97,0.01)	11	9.1160×10^{-6}	0.0194
	3.8	(0.01,1.99,0.01)	11	3.8334×10^{-6}	0.0031	(0.01,1.96,0.01)	11	9.0274×10^{-6}	0.0271
	3.9	(0.01,2.00,0.01)	12	4.8221×10^{-6}	0.0032	(0.01,2.01,0.01)	13	6.9002×10^{-6}	0.0290

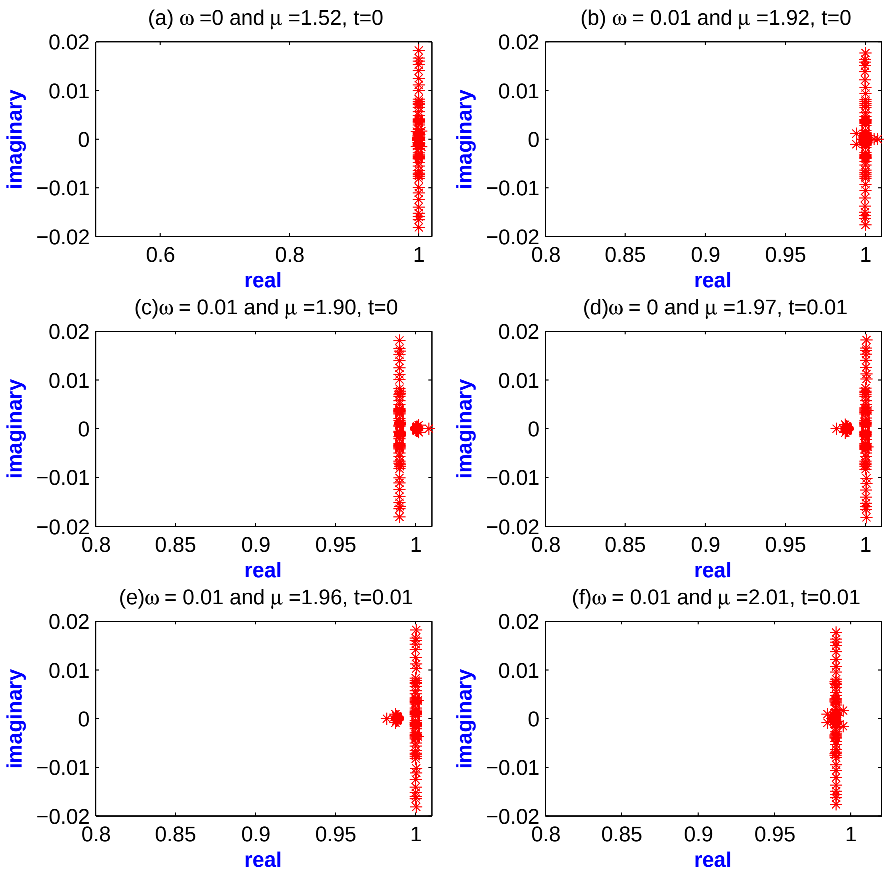

Fig. 5.2. The eigenvalue distributions of the preconditioned matrix with different values of ω,μ,and different values t for Example when $p = 8(\nu = 1)$.

respectively. We use IT, CPU and RES to represent the number of iteration steps, the elapsed CPU time in seconds and the norm of absolute residual vectors, respectively. In tables, " – " means that the parameter does not exist or equal to zero in the corresponding algorithm.

Table 5.2: Numerical results for GLHSS iteration methods for Cases I, II and III for example for $p = 16$ and $p = 24$ ($\nu = 1$).

		$p = 16$				$p = 24$			
Case		$(\omega, \mu\ t)$	IT	RES	CPU	$(\omega, \mu\ t)$	IT	RES	CPU
I	3.1	(-,1.48,-)	8	6.5461×10^{-6}	0.1777	(-,1.46,-)	8	5.6505×10^{-6}	1.2210
	3.2	(0.01,1.89,-)	11	8.1172×10^{-6}	0.2498	(0.01,1.88,-)	11	7.1306×10^{-6}	1.7314
	3.3	(0.01,1.87,-)	11	7.3241×10^{-6}	0.2574	(0.01,1.86,-)	11	6.5565×10^{-6}	1.7288
II	3.4	(-,1.67,0.1)	10	5.0526×10^{-6}	0.1000	(0.01,1.67,0.1)	10	5.3432×10^{-6}	1.6021
	3.5	(0.01,1.67,0.1)	10	5.0045×10^{-6}	0.2232	(0.01,1.67,0.1)	10	5.3046×10^{-6}	1.5108
	3.6	(0.01,1.85,0.1)	11	6.5935×10^{-6}	0.2436	(0.01,1.84,0.01)	11	6.0437×10^{-6}	1.7362
III	3.7	(-,1.95,0.01)	12	4.8680×10^{-6}	0.2738	(-,1.93,0.01)	11	9.1951×10^{-6}	1.7278
	3.8	(0.01,1.95,0.01)	12	4.8504×10^{-6}	0.2771	(0.01,1.93,0.01)	11	9.1713×10^{-6}	1.7413
	3.9	(0.01,2.02,0.01)	11	4.7504×10^{-6}	0.3343	(0.01,2.02,0.01)	14	4.7670×10^{-6}	2.7965

Table 5.3: Numerical results for GLHSS iteration methods for Cases I, II and III for example for $p = 4$ and $p = 8$ ($\nu = 10$).

		$p = 4$				$p = 8$			
Case		$(\omega, \mu\ t)$	IT	RES(10^{-6})	CPU	$(\omega, \mu\ t)$	IT	RES(10^{-6})	CPU
I	3.1	(-,1.047,-)	4	5.5532	0.0013	(-,1.038,-)	4	3.9182	0.0056
	3.2	(0.01,1.085,-)	5	4.7400	0.0012	(0.01,1.059,-)	5	4.4439	0.0089
	3.3	(0.01,1.061,-)	5	7.4353	0.0013	(0.01,1.057,-)	5	8.4388	0.0175
II	3.4	(-,0.989,0.1)	5	8.3550	0.0020	(-,0.988,0.1)	5	8.9873	0.0058
	3.5	(0.1,0.988,0.1)	5	8.4672	0.0030	(0.01,0.988,0.1)	5	8.9870	0.0184
	3.6	(0.1,0.924,0.1)	6	3.0743	0.0016	(0.01,1.050,0.1)	5	1.3628	0.0129
III	3.7	(-,1.083,0.01)	5	1.1023	0.0012	(-,1.083,0.01)	5	1.2616	0.0193
	3.8	(0.01,1.083,0.01)	5	1.0972	0.0019	(0.01,1.083,0.01)	5	1.2588	0.0059
	3.9	(0.01,1.082,0.01)	5	6.8068	0.0019	(0.01,1.086,0.01)	5	7.4575	0.0077

Table 5.4: Numerical results for GLHSS iteration methods for Cases I, II and III for example for $p = 16$ and $p = 24$ ($\nu = 10$).

		$p = 16$				$p = 24$			
Case		$(\omega, \mu\ t)$	IT	RES(10^{-6})	CPU	$(\omega, \mu\ t)$	IT	RES(10^{-6})	CPU
I	3.1	(-,1.030,-)	4	2.3429	0.0750	(-,1.027,-)	4	1.6334	0.5173
	3.2	(0.01,1.035,-)	4	6.5195	0.2498	(0.01,1.033,-)	4	4.7400	0.4942
	3.3	(0.01,1.057,-)	5	7.2760	0.1021	(0.01,1.058,-)	5	6.1734	0.7008
II	3.4	(-,0.987,0.1)	5	8.0385	0.1046	(0.01,0.987,0.1)	5	7.1068	0.6839
	3.5	(0.01,0.098,0.1)	5	8.0370	0.1017	(0.01,0.987,0.1)	5	7.1059	0.6941
	3.6	(0.01,1.049,0.1)	5	1.2454	0.1063	(0.01,1.048,0.01)	5	1.1058	0.6856
III	3.7	(-,1.085,0.01)	5	1.1497	0.0971	(-,1.089,0.01)	5	1.0263	0.6835
	3.8	(0.01,1.086,0.01)	5	1.1479	0.0977	(0.01,1.088,0.01)	5	1.0254	0.6718
	3.9	(0.01,1.091,0.01)	5	5.8563	0.0983	(0.01,1.094,0.01)	5	4.6635	0.7142

In Tables 5.5 and 5.6 , we list the implemented results on IT, RES and CPU of GMRES(20) and PGMRES(20) when they are applied to solve the above-mentioned numerical problem. The numbers outside of brackets denote outer iteration numbers and inside the inner iteration numbers of GMRES(m) or PGMRES(m), respectively. From these numerical results, we can see that PGMRES(20) with GLHSS preconditioner possesses much less iteration steps and CPU time than GMRES(20) without any preconditioner. This means that GLHSS splitting can act as an efficient preconditioner for solving the non-Hermitian saddle point problems by GMRES

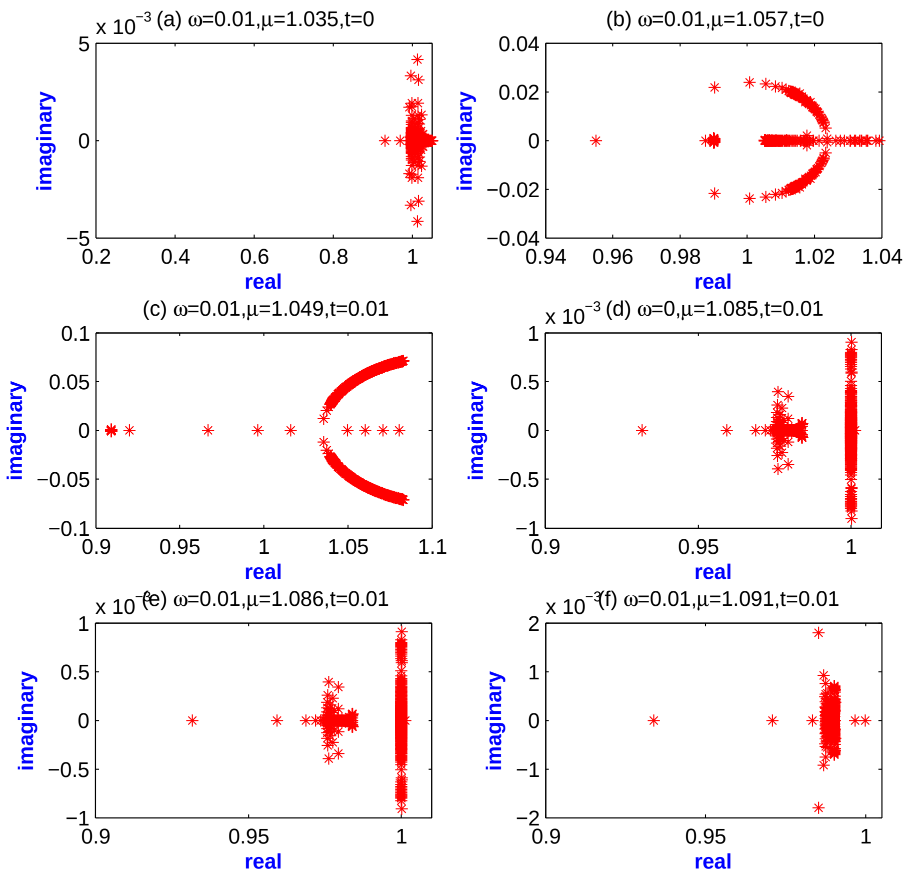

Fig. 5.3. The eigenvalue distributions of the preconditioned matrix with different values of ω,μ,and different values t for Example when $p = 16(\nu = 10)$.

Table 5.5: Numerical results for GMRES(20) and PGMRES(20) ($\nu = 1$).

	$p = 4$		$p = 8$		$p = 16$		$p = 24$	
	IT	CPU	IT	CPU	IT	CPU	IT	CPU
GMRES(20)	2(17)	0.0665	6(18)	0.2054	16(2)	0.9428	22(1)	2.3828
PGMRES(20)-3.1	1(7)	0.0241	1(7)	0.0638	1(8)	0.1594	1(8)	0.1850
PGMRES(20)-3.2	1(7)	0.0291	1(7)	0.0455	1(7)	0.1502	1(7)	0.2697
PGMRES(20)-3.3	1(7)	0.0138	1(7)	0.0495	1(7)	0.1414	1(7)	0.2951
PGMRES(20)-3.4	1(7)	0.0252	1(8)	0.0562	1(8)	0.0806	1(8)	0.1945
PGMRES(20)-3.5	1(7)	0.0262	1(8)	0.0542	1(8)	0.0759	1(8)	0.1905
PGMRES(20)-3.6	1(8)	0.0297	1(8)	0.0258	1(7)	0.1446	1(7)	0.1721
PGMRES(20)-3.7	1(7)	0.0280	1(7)	0.0438	1(7)	0.1633	1(8)	0.1965
PGMRES(20)-3.8	1(7)	0.0294	1(7)	0.0429	1(7)	0.1478	1(8)	0.1928
PGMRES(20)-3.9	1(7)	0.0434	1(7)	0.0480	1(7)	0.0767	1(7)	0.1711

method without preconditioners.

Fig. 5.1 shows the eigenvalue distributions of the original coefficient matrices with $\nu = 1$

Table 5.6: Numerical results for GMRES(20) and PGMRES(20) ($\nu = 10$).

	$p = 4$		$p = 8$		$p = 16$		$p = 24$	
	IT	CPU	IT	CPU	IT	CPU	IT	CPU
GMRES(20)	2(9)	0.0541	3(9)	0.1068	59(16)	4.7053	118(14)	26.1735
PGMRES(20)-3.1	1(3)	0.0130	1(3)	0.0267	1(4)	0.0869	1(4)	0.1404
PGMRES(20)-3.2	1(4)	0.0881	1(4)	0.0412	1(4)	0.0404	1(4)	0.1102
PGMRES(20)-3.3	1(4)	0.0175	1(5)	0.0271	1(4)	0.0806	1(4)	0.0969
PGMRES(20)-3.4	1(5)	0.0816	1(5)	0.0146	1(5)	0.0498	1(5)	0.1135
PGMRES(20)-3.5	1(5)	0.0096	1(5)	0.0339	1(7)	0.0726	1(5)	0.2283
PGMRES(20)-3.6	1(5)	0.0204	1(4)	0.0273	1(4)	0.0785	1(5)	0.1243
PGMRES(20)-3.7	1(4)	0.0178	1(4)	0.0314	1(4)	0.0414	1(4)	0.0959
PGMRES(20)-3.8	1(4)	0.0178	1(4)	0.0273	1(4)	0.0842	1(4)	0.1067
PGMRES(20)-3.9	1(3)	0.0144	1(4)	0.0261	1(4)	0.0401	1(4)	0.1050

and $\nu = 10$ respectively. Fig. 5.2 and Fig. 5.3 show the eigenvalue distributions of the preconditioned matrices with GLHSS preconditioner and different parameters for the above example. From these figures we can see that the eigenvalues of all preconditioned matrices become more clustered.

6. Conclusion

In this work, we extend the generalized local Hermitian and skew-Hermitian splitting iteration methods for the non-Hermitian saddle point problems to the non-Hermitian generalized saddle point problems. The convergence of new GLHSS method is discussed and the convergence conditions are given out. Numerical examples show the effectiveness of our method. In particular, our GLHSS splitting can act as an efficient preconditioner for Krylov subspace methods such as GMRES method. The parameters in our experiments are chosen randomly. So how to derive or choose the optimal parameters in the GLHSS method for the non-Hermitian generalized saddle point problems still need further study.

Acknowledgments. We would like to express our sincere gratitude to the anonymous referees whose constructive comments have the presentation of this paper greatly improved. The work was supported by the National Natural Science Foundation (No.11171371 and No.11101195).

References

[1] Z.-Z. Bai and G.H. Golub, Accelerated Hermitian and skew-Hermitian splitting iteration methods for saddle-point problems, *IMA J. Numer. Anal.,* **27** (2007), 1–23.

[2] Z.-Z. Bai, G.H. Golub and M.K. Ng, Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems, *SIAM J. Matrix Anal. Appl.,* **24** (2003), 603–626.

[3] Z.-Z. Bai and Z.-Q. Wang, On parameterized inexact Uzawa methods for generalized saddle point problems, *Linear Algebra Appl.,* **428** (2008), 2900–2932.

[4] Z.-Z. Bai, G.H. Golub and C.-K. Li, Optimal parameter in Hermitian and skew-Hermitian splitting method for certain two-by-two block matrices, *SIAM J. Sci. Comput.,* **28** (2006), 583–603.

[5] Z.-Z. Bai, G.H. Golub and J.-Y. Pan, Preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite linear systems, *Numer. Math.,* **98** (2004), 1–32.

[6] Z.-Z. Bai, B.N. Parlett and Z.-Q. Wang, On generalized successive overrelaxation methods for augmented linear systems, *Numer. Math.,* **102** (2005), 1–38.

[7] L. Bergamaschi, J. Gondzio and G. Zilli, Preconditioning indefinite systems in interior point methods for optimization, *Comput. Optim. Appl.,* **28** (2004), 149–171.

[8] M. Benzi and G.H. Golub, A precoditioner for generalized saddle point problems, *SIAM J. Matrix Anal. Appl.,* **26** (2004), 20–41.

[9] M. Benzi, G.H. Golub and J. Liesen, Numerical solution of saddle point problems, *Acta Numer.,* **14** (2005), 1–137.

[10] F. Brezzi and M. Fortin, Mixed and Hybrid Finite Element Methods, Springer-Verlag, New York and London, 1991.

[11] X.-J. Chen, On preconditioned Uzawa methods and SOR methods for saddle-point problems, *J. Comput. Appl. Math.,* **100** (1998), 207–224.

[12] F. Chen and Y.-L. Jiang, A generalization of the inexact parameterized Uzawa methods for saddle point problems, *Appl. Math. Comput.,* **206** (2008), 765–771.

[13] M.-R. Cui, Analysis of iterative algorithms of Uzawa type for saddle point problems, *Appl. Numer. Math.,* **50** (2004), 133–146.

[14] J.H. Bramble and J.E. Pasciak, A preconditioning technique for indefinite systems resulting from mixed approximations of elliptic problems, *Math. Comput.,* **50** (1988), 1–17.

[15] Z.-H. Cao, Fast Uzawa algorithms for solving non-symmetric stabilized saddle point problems, *Numer. Linear Algebra Appl.,* **11** (2004), 1–24.

[16] B. Zheng, Z.-Z. Bai and X. Yang, On semi-convergence of parameterized Uzawa methods for singular saddle point problems, *Linear Algebra Appl.,* **431** (2009), 808–817.

[17] H.C. Elman, D.J. Silvester and A.J. Wathen, Finite Elements and Fast Iterative Solvers, Numerical Mathematics and Scientific Computation, Oxford University Press, Oxford, 2005.

[18] T. Rusten and R. Winther, A preconditioned iterative method for saddle point problems, *SIAM J. Matrix Anal. Appl.,* **13** (1992), 887–904.

[19] J.H. Bramble, J.E. Pasciak and A.T. Vassilev, Uzawa type algorithms for nonsymmetric saddle point problems, *Math. Comput.,* **69** (1999), 667–689.

[20] G.H. Golub, X. Wu and J.-Y. Yuan, SOR-like methods for augmented systems, *BIT Numer. Math.,* **41** (2001), 71–85.

[21] M.-Q. Jiang and Y. Cao, On local Hermitian and skew-Hermitian splitting iteration methods for generalized saddle point problems, *J. Comput. Appl. Math.,* **231** (2009), 973–982.

[22] X.-F. Ling and X.-Z. Hu, On the iterative algorithm for large sparse saddle point problems, *Appl. Math. Comput.,* **178** (2006), 372–379.

[23] Y.-Q. Lin and Y.-M. Wei, Corrected Uzawa methods for solving large nonsymmetric saddle point problems, *Appl. Math. Comput.,* **183** (2006), 1108–1120.

[24] Z.-Z. Bai, Optimal parameters in the HSS-like methods for saddle point problems, *Numer. Linear Agebra Appl.,* **16** (2009), 447–479.

[25] Z.-H. Huang and T.-Z. Huang, Sepectral properties of the preconditioned AHSS iteration method for generalized saddle point problems, *Comput. Appl. Math.,* **29** (2010), 269–295.

[26] Z.-Z. Bai and G.-Q. Li, Restrictively preconditioned conjugate grdient methods for systems of linear equations, *IMA J. Numer. Anal.,* **23** (2003), 561–580.

[27] Z.-Z. Bai, G.H. Golub and C.-K. Li, Convergence properties of preconditioned Hermitian and skew-Hermitian splitting methods for non-Hermitian positive semidefinite matrices, *Math. Comput.,* **76** (2007), 287–298.

[28] J.-F. Yin and Z.-Z. Bai, The restrictively preconditioned conjugate gradient methods on normal residual for block two-by-two linear systems, *J. Comput. Math.,* **26** (2008), 240–249.

[29] D.M. Yong, Iterative Solution for Large Linear Sysytems, Academic Press, New York, 1971.

[30] J.-L. Li, D. Luo and Z.-J. Zhang, A generalization of local symmetric and skew-symmetric iteration methods for generalized saddle point problems, *J. Appl. Math. Inform.,* **29** (2011), 1167–1178.

[31] Y.-Y. Zhou and G.-F. Zhang, A generalization of parameterized inexact Uzawa method for generalized saddle point problems, *Appl. Math. Comput.,* **215** (2009), 599–607.

[32] M.-Z. Zhu, A generalization of the local Hermitian and skew-Hermitian splitting iteration methods for the non-Hermitian saddle point problem, *Appl. Math. Comput.,* **218** (2012), 8816–8824.