Computational and Applied Mathematics (2020) 39:154
https://doi.org/10.1007/s40314-020-01180-0

Efficient HSS-based preconditioners for generalized saddle point problems

Ke Zhang[1] · **Lin-Na Wang**[1]

Received: 10 April 2019 / Revised: 3 April 2020 / Accepted: 29 April 2020 / Published online: 28 May 2020

Abstract
A fast iteration method based on HSS is proposed for solving the nonsymmetric generalized saddle point problem. It converges to the unique solution of the generalized saddle point problem unconditionally. We devise a new preconditioner induced by the new iteration method. We analyze the spectrum of the preconditioned coefficient matrix, and reveal the relation between the theoretically required number of iteration steps and the dimension of the preconditioned Krylov subspace. Furthermore, some practical inexact variants of the new preconditioner have been developed to reduce the computational overhead. Numerical experiments validate the effectiveness of the proposed preconditioners.

Keywords Generalized saddle point problem · Preconditioner · Hermitian and skew-Hermitian splitting · Krylov subspace method

Mathematics Subject Classification 65F10 · 65N22

1 Introduction

We consider the solution of the generalized saddle point linear system

$$\begin{bmatrix} A & B^T \\ -B & C \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ -g \end{bmatrix}, \quad \text{or } \mathcal{A}u = b, \tag{1}$$

where A is an $n \times n$ symmetric and positive definite matrix, B is an $m \times n$ full-rank matrix, C is an $m \times m$ symmetric and positive semidefinite matrix and $m \leq n$. With such assumptions, the existence and uniqueness of the solution of (1) can be assured. The block linear system (1) stems from many real-world applications, including the constrained optimal control (Betts 2001), the computational fluid dynamics (Saad 2003) and the mixed finite element of elliptic PDEs (Benzi et al. 2005), etc.

Communicated by Natasa Krejic.

✉ Ke Zhang
xznuzk123@126.com

[1] Department of Mathematics, Shanghai Maritime University, Shanghai 201306, People's Republic of China

Recent years have witnessed vigorous developments in the numerical algorithms for solving (1). Due to the large size and sparsity of the coefficient matrix $\mathcal{A}$, iterative methods are usually favored. In addition to those ad hoc ones, existing approaches include the stationary iteration schemes (Bai et al. 2005b; Zhang and Shang 2010; Yun 2013; Zhang et al. 2019), the matrix-splitting methods (Bai et al. 2008; Cao et al. 2016; Bai and Golub 2007; Li and Wu 2015; Liang and Zhang 2016; Shen 2014; Zeng and Ma 2016), the more involved Krylov subspace methods (Bai 2015; Saad 2003) and their hybrid variants (Cao and Yi 2016). Among all methods, the restarted GMRES method (Saad and Schultz 1986) has received much attention and been used extensively in view of their effectiveness. However, it is known that GMRES(k) often suffers from slow convergence or even stagnation. Therefore, a reasonable preconditioner must be chosen to accelerate GMRES(k). By far, some kinds of preconditioners pertaining to Krylov subspace methods have been proposed, including the HSS-type preconditioners (Cao et al. 2016; Bai and Golub 2007; Huang et al. 2009; Liao and Zhang 2019), the matrix splitting preconditioners (Bai et al. 2005a; Ling and Liu 2017; Murphy et al. 2000; Shen and Shi 2016; Simoncini 2004; Zhang et al. 2014, 2017), the constraint preconditioners (Keller et al. 2000; Shen et al. 2019; Zhang et al. 2011) and the dimensional splitting preconditioners (Ke and Ma 2017; Cao et al. 2013); see (Benzi et al. 2005) for developments up to the year 2005 and more recent surveys (Bai 2015; Rozložník 2018; Wathen 2015).

The essential idea behind developing a good preconditioner is that the preconditioner is expected to approximate the coefficient matrix in some sense such that the preconditioned matrix will have a clustered spectrum (away from the origin) or readily be computed. For this reason, we are interested in constructing efficient preconditioners that preserve the structure of the original coefficient matrix in (1) as much as possible. In the context of solving (generalized) saddle point system, some efforts have been made towards this goal in recent years. For example, Murphy et al. (2000) propose a Schur-complement-based block-diagonal preconditioner which yields a preconditioned matrix with exactly three or exactly two distinct eigenvalues. Following this reasoning, de Sturler and Liesen (2005) derive block-diagonal preconditioners from the splitting of the (1,1)-block matrix given in Murphy et al. (2000) from which the solution of the original problem can be restored by solving a smaller related linear system; see also (Beik et al. 2017). By incorporating the (1,2) and (2,1) matrix blocks, Keller et al. (2000) devise a constraint preconditioner that is different from the original system only in the (1,1)-block. Pan et al. (2006) propose a deteriorated positive-definite and skew-symmetric splitting (DPSS) preconditioner which stimulates the research of matrix-splitting-based preconditioners (Cao et al. 2015; Zhang et al. 2014).

In the literature, a common way of implementing the HSS iteration (Bai et al. 2003) is to alternate between the Hermitian and skew-Hermitian parts of the coefficient matrix. By switching the role of the two splitting matrices in the Hermitian and skew-Hermitian splitting, Zhang (2018) proposes an efficient variant of HSS preconditioner (EVHSS) for solving (1). In that work, it is noted that the contraction factors differ though swapping the Hermitian part with the skew-Hermitian part gives the same asymptotic convergence rate as in the original HSS iteration. The theoretical result on convergence is proved in Zhang (2018) and is further refined by Chen (2018). Numerical performance of EVHSS is very promising when compared with some existing HSS-type preconditioners.

When implementing EVHSS, one often needs to strike a balance in choosing the relaxation parameter to make the EVHSS preconditioner maximally preserve the structure of $\mathcal{A}$, though a theoretical quasi-optimal parameter is given. Motivated by this observation, we propose a new iteration scheme for solving (1). The unconditional convergence of the new stationary method is proved. Induced by the new method, an HSS-type preconditioner is also devised to accelerate the convergence of GMRES. By construction, the new preconditioner

resembles more from the coefficient matrix $\mathcal{A}$ than EVHSS. Theoretical analysis shows that the preconditioned matrix with the new preconditioner is endowed with a well-clustered eigenvalue distribution which is a welcome for Krylov subspace methods. To make the new preconditioner practical, we also give a framework for developing inexact preconditioners tailored to large and sparse generalized saddle point problems.

The remainder of this paper is as follows. In Sect. 2, we first give a quick recap of the EVHSS iteration/preconditioner, and then introduce the new stationary iteration and analyze its unconditional convergence. In Sect. 3, we present the new preconditioner induced by the new iteration scheme in Sect. 2, investigate the spectrum information of the preconditioned matrix, and come up with a framework of practical inexact preconditioners. In Sect. 4, we employ some numerical experiments to verify the effectiveness of the new preconditioner and its inexact variants. Finally, some conclusions are given in Sect. 5.

2 A new iteration scheme and its convergence analysis

The new iteration method in this work is motivated by the EVHSS iteration (Zhang 2018). Therefore, we first give a sketch of the EVHSS iteration method/preconditioner in this section. Aware of the possible problem of choosing the relaxation parameter in EVHSS, we then present the new iteration scheme for solving (1) and finally establish the unconditional convergence result.

2.1 The EVHSS iteration/preconditioner

In the language of the Hermitian and skew-Hermitian splitting (Bai et al. 2003), the generalized saddle point matrix $\mathcal{A}$ in (1) admits the following splitting:

$$\mathcal{A} = \begin{bmatrix} A & 0 \\ 0 & C \end{bmatrix} + \begin{bmatrix} 0 & B^T \\ -B & 0 \end{bmatrix} \equiv \mathcal{H} + \mathcal{S},$$

where $\mathcal{H}$ and $\mathcal{S}$ are the Hermitian and skew-Hermitian parts of $\mathcal{A}$, respectively. The corresponding HSS preconditioner for (1) is then given by

$$\mathcal{P}_{HSS} = \frac{1}{2\alpha}(\alpha I + \mathcal{H})(\alpha I + \mathcal{S}), \tag{2}$$

where $\alpha > 0$. In Zhang (2018), it is stated that the contraction factor can be different if the Hermitian and skew-Hermitian parts are interchanged.

Motivated by this finding, Zhang proposes an efficient variant of $\mathcal{P}_{HSS}$ by swapping $\alpha I + \mathcal{H}$ and $\alpha I + \mathcal{S}$ in (2). The resulting preconditioner EVHSS reads as

$$\mathcal{P}_{\text{EVHSS}} = \frac{1}{\alpha}\begin{bmatrix} A & B^T \\ -B & \alpha I \end{bmatrix}\begin{bmatrix} \alpha I & 0 \\ 0 & \alpha I + C \end{bmatrix} = \begin{bmatrix} A & B^T(I + \frac{1}{\alpha}C) \\ -B & \alpha I + C \end{bmatrix}. \tag{3}$$

The difference between $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{A}$ is given by

$$\mathcal{R}_{\text{EVHSS}} = \mathcal{P}_{\text{EVHSS}} - \mathcal{A} = \begin{bmatrix} 0 & \frac{1}{\alpha}B^T C \\ 0 & \alpha I \end{bmatrix}. \tag{4}$$

Based on the matrix splitting (4), Zhang constructs the EVHSS iteration by

$$x^{(k+1)} = Gx^{(k)} + \tilde{c},$$

where $G = \mathcal{P}_{\text{EVHSS}}^{-1}\mathcal{R}_{\text{EVHSS}}$ and $\tilde{c} = \mathcal{P}_{\text{EVHSS}}^{-1}b$.

2.2 A new iteration method

In Zhang (2018), the EVHSS preconditioner is shown to outperform some other existing HSS-based preconditioners with appropriate choices of α. In practice, however, we require to strike a balance between the value of α and $1/\alpha$, as observed from (4); sufficiently large value of α makes the (1,2)-block a better approximation to the zero matrix but a (2,2)-block matrix with large entries in (4), and vice versa. In this work, we sidestep this problem by simply abandoning the (2,2)-block αI in (4), which yields the new preconditioner

$$\mathcal{P}_{\text{new}} = \begin{bmatrix} A & B^T(\frac{1}{\alpha}C + I) \\ -B & C \end{bmatrix}, \tag{5}$$

where $\alpha > 0$. We note that another variant can be obtained by letting the (1,2)-block in (4) be zero. Nevertheless, we only consider the variant in (5) since it is efficient and easier to be analyzed theoretically. Consequently, the corresponding difference matrix between $\mathcal{P}_{\text{new}}$ and $\mathcal{A}$ presents to be

$$\mathcal{R}_{\text{new}} = \mathcal{P}_{\text{new}} - \mathcal{A} = \begin{bmatrix} 0 & \frac{1}{\alpha}B^T C \\ 0 & 0 \end{bmatrix}. \tag{6}$$

Hence, the following stationary iteration scheme induced from the splitting (6) is given by

$$\begin{bmatrix} A & B^T(\frac{1}{\alpha}C + I) \\ -B & C \end{bmatrix} \begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{\alpha}B^T C \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix} + b,$$

or in a compact form, i.e.,

$$x^{(k+1)} = \Gamma x^{(k)} + c, \tag{7}$$

where $\Gamma = \mathcal{P}_{\text{new}}^{-1}\mathcal{R}_{\text{new}}$, $c = \mathcal{P}_{\text{new}}^{-1}b$ and $k = 0, 1, \ldots$.

It is known that the iteration (7) converges if the spectral radius of the iteration matrix Γ is less than 1, which is guaranteed by the following theorem.

Theorem 1 *Suppose that the matrices A, B and C are defined in (1), and α is a positive constant. Let (θ_i, v_i) be the ith eigenpair of the matrix $\alpha^{-1}S^{-1}BA^{-1}B^T C$, where $S = C + BA^{-1}B^T(\alpha^{-1}C + I)$. Then $\rho(\Gamma) < 1$, that is, the iterative scheme (7) converges to the unique solution of (1) unconditionally.*

Proof The new preconditioner $\mathcal{P}_{\text{new}}$ in (5) can be factorized as

$$\begin{aligned} \mathcal{P}_{\text{new}} &= \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T(\frac{1}{\alpha}C + I) \\ -B & C \end{bmatrix} \\ &= \begin{bmatrix} A & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ -B & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & A^{-1}B^T(\frac{1}{\alpha}C + I) \\ 0 & I \end{bmatrix}, \end{aligned} \tag{8}$$

where $S = C + BA^{-1}B^T(\frac{1}{\alpha}C + I)$. As a result, we have

$$\mathcal{P}_{\text{new}}^{-1} = \begin{bmatrix} I & -A^{-1}B^T(\frac{1}{\alpha}C + I) \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ B & I \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & I \end{bmatrix}. \tag{9}$$

Using (7) and (9), we rewrite the iteration matrix Γ as

$$\Gamma = \mathcal{P}_{\text{new}}^{-1}\mathcal{R}_{\text{new}} = \begin{bmatrix} 0 & T_1 \\ 0 & T_2 \end{bmatrix}, \tag{10}$$

where $T_1 = \alpha^{-1}A^{-1}B^TC - \alpha^{-1}A^{-1}B^T(\alpha^{-1}C+I)S^{-1}BA^{-1}B^TC$ and $T_2 = \alpha^{-1}S^{-1}BA^{-1}B^TC$. From (10), it is clear that Γ has the eigenvalue 0 with multiplicity n while the remaining eigenvalues of Γ are given by those of T_2. Then it suffices to look into the eigenvalues of T_2. Let (θ_i, v_i) be the ith eigenpair of the matrix T_2, i.e.,

$$BA^{-1}B^TCv_i = \alpha\theta_i Sv_i. \tag{11}$$

Multiplying $v_i^*/v_i^*v_i$ on both sides of (11) yields

$$\frac{v_i^*BA^{-1}B^TCv_i}{v_i^*v_i} = \alpha\theta_i\frac{v_i^*Sv_i}{v_i^*v_i}. \tag{12}$$

By the definition of S, we obtain from (12) that

$$\frac{v_i^*BA^{-1}B^TCv_i}{v_i^*v_i} = \theta_i\left(\alpha\frac{v_i^*Cv_i}{v_i^*v_i} + \frac{v_i^*BA^{-1}B^TCv_i}{v_i^*v_i} + \alpha\frac{v_i^*BA^{-1}B^Tv_i}{v_i^*v_i}\right). \tag{13}$$

Denote by

$$\mu_i = \frac{v_i^*Cv_i}{v_i^*v_i}, \varepsilon_i + \eta_i\iota = \frac{v_i^*BA^{-1}B^TCv_i}{v_i^*v_i}, \gamma_i = \frac{v_i^*BA^{-1}B^Tv_i}{v_i^*v_i}, \tag{14}$$

where ι is the imaginary unit. It follows from (13) and (14) that

$$\theta_i = \frac{\varepsilon_i + \eta_i\iota}{\varepsilon_i + \eta_i\iota + \alpha(\mu_i + \gamma_i)}.$$

Therefore, the spectral radius of the iteration matrix Γ is given by

$$\begin{aligned}\rho(\Gamma) = \max_i|\theta_i| &= \max_i\left|\frac{\varepsilon_i + \eta_i\iota}{\varepsilon_i + \eta_i\iota + \alpha(\mu_i + \gamma_i)}\right| \\ &= \max_i\sqrt{\frac{\varepsilon_i^2 + \eta_i^2}{(\varepsilon_i + \alpha\mu_i + \alpha\gamma_i)^2 + \eta_i^2}}.\end{aligned}$$

Since A is symmetric positive definite, B is of full rank, and C is symmetric positive semidefinite, then we obtain from (14) that $\gamma_i > 0$ and $\mu_i \geq 0$, which, coupled with $\alpha > 0$, shows that $\rho(\Gamma) < 1$. Therefore, the new iteration converges to the unique solution of (1) unconditionally. □

3 A new preconditioner and its properties

The convergence rate of the stationary methods (including the classical Jacobi, Gauss–Seidel and SOR) can be slow when compared with the more sophisticated Krylov subspace methods. For this reason, we will not elaborate on the implementation details of the new iteration scheme (7). Alternatively, we are interested in exploiting the matrix $\mathcal{P}_{\text{new}}$ as a preconditioner to accelerate the Krylov subspace methods, which is the main task of this section.

3.1 Spectrum of the preconditioned matrix

It is known that the convergence behavior of preconditioned iterative methods relies heavily on eigen-information of the preconditioned matrix. Therefore, we investigate the eigenvalue distribution of the preconditioned matrix $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ in this subsection.

We begin the discussion by considering the eigenvalue problem

$$\begin{bmatrix} A & B^T \\ -B & C \end{bmatrix} \begin{bmatrix} A & B^T(\frac{1}{\alpha}C + I) \\ -B & C \end{bmatrix}^{-1} \begin{bmatrix} p \\ q \end{bmatrix} = \lambda \begin{bmatrix} p \\ q \end{bmatrix}. \tag{15}$$

The following result describes the eigenvalue distribution of the preconditioned matrix.

Theorem 2 *The eigenvalues of $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ are either 1 (with multiplicity at least n) or of the form*

$$\lambda_i = \frac{\alpha(\mu_i + \gamma_i)}{\alpha(\mu_i + \gamma_i) + \varepsilon_i + \eta_i \iota}, \tag{16}$$

where μ_i, γ_i, ε_i and η_i are defined in (14).

Proof Since the right-preconditioned matrix $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ is similar to its left-preconditioned counterpart $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$, then it reduces to examining the eigenvalues of $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$. By using (10), we have

$$\mathcal{P}_{\text{new}}^{-1}\mathcal{A} = \mathcal{P}_{\text{new}}^{-1}(\mathcal{P}_{\text{new}} - \mathcal{R}_{\text{new}}) = I - \Gamma = \begin{bmatrix} I & -T_1 \\ 0 & I - T_2 \end{bmatrix}, \tag{17}$$

where T_1, T_2 are defined in (10). It follows from (11) and (14) that the eigenvalues of $I - T_2$ are of the form

$$\lambda_i = \frac{\alpha(\mu_i + \gamma_i)}{\alpha(\mu_i + \gamma_i) + \varepsilon_i + \eta_i \iota},$$

which completes the proof. □

Remark 1 As shown in (16), the nonunit eigenvalues λ_i approach 1 if α is sufficiently large. Under this condition, all nonunit eigenvalues of $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ huddle around the point (1, 0), which is appealing for the convergence of Krylov subspace methods; see the numerical examples in Sect. 4.

3.2 Right-preconditioned GMRES

In this subsection, we employ the new preconditioner to speed up the convergence of GMRES (Saad and Schultz 1986). The implementation details are given in Algorithm 1. With right preconditioning, the resulting Krylov subspace now becomes

$$\mathcal{K}_k(\mathcal{A}\mathcal{P}_{\text{new}}^{-1}, b) = \text{span}\{b, \mathcal{A}\mathcal{P}_{\text{new}}^{-1}b, \ldots, (\mathcal{A}\mathcal{P}_{\text{new}}^{-1})^{k-1}b\}. \tag{18}$$

It is known that GMRES converges very fast if the approximation of the exact solution is found in a low-dimensional subspace. The next result reveals the intrinsic relation between the dimension of the Krylov subspace $\mathcal{K}(\mathcal{A}\mathcal{P}_{\text{new}}^{-1}, b)$ and the size of (2, 2)-block matrix in $\mathcal{A}$.

Theorem 3 *The minimal polynomial of the preconditioned matrix $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ has a degree not exceeding $m + 1$. Thus, the dimension of the corresponding Krylov subspace $\mathcal{K}(\mathcal{A}\mathcal{P}_{\text{new}}^{-1}, b)$ is at most $m + 1$.*

Proof From (17), we know that

$$\mathcal{P}_{\text{new}}^{-1}\mathcal{A} = \begin{bmatrix} I & -T_1 \\ 0 & I - T_2 \end{bmatrix}. \tag{19}$$

Algorithm 1 Restarted GMRES with the new preconditioner.

1: Compute $r_0^{(1)} = f - Ax_0 - B^T y_0$ and $r_0^{(2)} = -g + Bx_0 - Cy_0$, then $r_0 = [r_0^{(1)}, r_0^{(2)}]^T$. Set $\sigma = \|r_0\|_2$ and $v_1 = r_0/\sigma$.
2: **for** $j = 1, \ldots, k$ **do**
3: $z_j = \mathcal{P}_{\text{new}}^{-1} v_j$
4: $w = \mathcal{A} z_j$
5: **for** $i = 1, \ldots, j$ **do**
6: $h_{ij} = w^T v_i$
7: $w = w - h_{ij} v_i$
8: **end for**
9: Compute $h_{j+1,j} = \|w\|_2$ and $v_{j+1} = w/h_{j+1,j}$
10: Define $V_k = [v_1, \ldots, v_k]$, $\bar{H}_k = \{h_{ij}\}$, $1 \le i \le k+1$, $1 \le j \le k$
11: **end for**
12: Compute $y_k = \arg\min_{y \in \mathbb{R}^k} \|\sigma e_1 - \bar{H}_k y\|_2$, $x_k = x_0 + \mathcal{P}_{\text{new}}^{-1} V_k y_k$, where $e_1 = [1, 0, \ldots, 0]^T$.
13: If converged then stop; otherwise set $x_0 = x_k$ and goto line 1.

Since $\mathcal{A}\mathcal{P}_{\text{new}}^{-1} = \mathcal{P}_{\text{new}}(\mathcal{P}_{\text{new}}^{-1}\mathcal{A})\mathcal{P}_{\text{new}}^{-1}$, then $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ and $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$ are similar. Similar matrices have the same minimal polynomial (Horn and Johnson 1990, Corollary 3.3.3). Therefore, it reduces to prove that the minimal polynomial of $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$ has a degree at most $m+1$. It follows from Theorem 2 that the characteristic polynomial of $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$ is given by

$$(\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - I)^n \cdot \prod_{i=1}^{m} (\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - \lambda_i I), \tag{20}$$

where λ_i is the nonunit eigenvalue defined in (16). By expanding the polynomial $(\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - I) \cdot \prod_{i=1}^{m}(\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - \lambda_i I)$, we have

$$(\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - I) \cdot \prod_{i=1}^{m} (\mathcal{P}_{\text{new}}^{-1}\mathcal{A} - \lambda_i I) = \begin{bmatrix} 0 & -T_1 \prod_{i=1}^{m}[(1-\lambda_i)I - T_2] \\ & \\ 0 & -T_2 \prod_{i=1}^{m}[(1-\lambda_i)I - T_2] \end{bmatrix},$$

where $1 - \lambda_i$ are the eigenvalues of T_2 for $i = 1, \ldots, m$; see Theorem 2. By the Cayley–Hamilton theorem, we have

$$\prod_{i=1}^{m}[(1-\lambda_i)I - T_2] = 0.$$

In other words, the degree of the minimal polynomial of $\mathcal{P}_{\text{new}}^{-1}\mathcal{A}$ is at most $m+1$. Consequently, the degree of the minimal polynomial of the right preconditioned matrix $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ is at most $m+1$. From (Saad 2003, Proposition 6.1), it is known that the degree of the minimal polynomial is equal to the dimension of the corresponding Krylov subspace $\mathcal{K}(\mathcal{A}\mathcal{P}_{\text{new}}^{-1}, b)$ which completes the proof. □

Remark 2 As stressed in Remark 1, the nonunit eigenvalues λ_i approach 1 if the value of α is sufficiently large. Therefore, a further reduction in the iteration steps (much less than $m+1$) can be anticipated if some nonunit eigenvalues λ_i approximate 1 well. This is confirmed by numerical examples in Sect. 4.

Now let us have a close look at Algorithm 1. The major difference between Algorithm 1 and its unpreconditioned counterpart (Saad and Schultz 1986) is that an extra matrix-vector

Algorithm 2 Implementation of $z = \mathcal{P}_{\text{new}}^{-1} r$.

1: Solve

$$\begin{bmatrix} A^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} r_a \\ r_b \end{bmatrix} = \begin{bmatrix} u^{(1)} \\ v^{(1)} \end{bmatrix},$$

which involves solving $Au^{(1)} = r_a$ and $v^{(1)} = r_b$.

2: Solve

$$\begin{bmatrix} I & 0 \\ B & I \end{bmatrix} \begin{bmatrix} u^{(1)} \\ v^{(1)} \end{bmatrix} = \begin{bmatrix} u^{(2)} \\ v^{(2)} \end{bmatrix},$$

which reduces to $u^{(2)} = u^{(1)}$ and $v^{(2)} = Bu^{(1)} + v^{(1)}$.

3: Solve

$$\begin{bmatrix} I & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} u^{(2)} \\ v^{(2)} \end{bmatrix} = \begin{bmatrix} u^{(3)} \\ v^{(3)} \end{bmatrix},$$

which involves solving $Sv^{(3)} = v^{(2)}$ and updating $u^{(3)} = u^{(2)}$, where $S = C + BA^{-1}B^T(\alpha^{-1}C + I)$.

4: Solve

$$\begin{bmatrix} I & -A^{-1}B^T(\frac{1}{\alpha}C + I) \\ 0 & I \end{bmatrix} \begin{bmatrix} u^{(3)} \\ v^{(3)} \end{bmatrix} = \begin{bmatrix} z_a \\ z_b \end{bmatrix},$$

which involves solving $A\tilde{v}^{(3)} = B^T(\alpha^{-1}C + I)v^{(3)}$, $z_a = u^{(3)} - \tilde{v}^{(3)}$ and $z_b = v^{(3)}$.

Algorithm 3 Practical implementation of $z = \mathcal{P}_{\text{new}}^{-1} r$.

1: Solve $Au = r_a$.
2: Solve $Sv = Bu + r_b$, where $S = C + BA^{-1}B^T(\alpha^{-1}C + I)$.
3: Solve $A\tilde{v} = B^T(\alpha^{-1}C + I)v$, $z_a = u - \tilde{v}$ and $z_b = v$.

product involving $\mathcal{P}_{\text{new}}^{-1}$ in line 3 (thus line 12) requires to be computed. To this end, a naive approach is to compute $\mathcal{P}_{\text{new}}^{-1}$ explicitly and then do the resulting matrix-vector product. However, this is not preferred numerically and even prohibited in most situations. Instead, it is often accomplished by solving an equivalent linear systems. For example, we solve r from the linear systems $\mathcal{P}_{\text{new}} z = r$ if $\mathcal{P}_{\text{new}}^{-1} r$ is needed, where $r = (r_a^T, r_b^T)^T$ and $z = (z_a^T, z_b^T)^T$. Using the decomposition (9), we transform the problem of solving $z = \mathcal{P}_{\text{new}}^{-1} r$ into the following form:

$$\begin{bmatrix} z_a \\ z_b \end{bmatrix} = \begin{bmatrix} I & -A^{-1}B^T(\frac{1}{\alpha}C + I) \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ B & I \end{bmatrix} \begin{bmatrix} A^{-1} & 0 \\ 0 & I \end{bmatrix} \begin{bmatrix} r_a \\ r_b \end{bmatrix}, \quad (21)$$

whose implementation details are presented in Algorithm 2.

Algorithm 2 shows clearly how operations like $\mathcal{P}_{\text{new}}^{-1} r$ are done. From a numerical point of view, however, the process is not optimized and can be improved to be more efficient; for instance, the intermediate vectors $v^{(1)}$ (Step 1), $u^{(2)}$, $v^{(2)}$ (Step 2) and $u^{(3)}$ (Step 4) need not be stored. In light of this, we give a practical version of Algorithm 2 which is employed in our numerical experiments; see Algorithm 3.

3.3 Inexact variants

In Algorithm 3, the main computational overhead consists of solving three linear systems associated with two different coefficient matrices. In particular, one needs to solve a sub-linear system with the Schur complement coefficient matrix $S = C + BA^{-1}B^T(\alpha^{-1}C + I)$. In the context of solving saddle point linear systems, it can be done either directly or iteratively; see, for instance, (Cao et al. 2015; Rozložník 2018). Alternatively, an efficient work-around is to replace A with its approximation $\tilde{A}$ that is easier to implement. Many possible candidates for approximating A are available; for instance, A can be approximated by its diagonal/tridiagonal part or matrix factors derived from incomplete LU factorization. Thus, the approximation of the preconditioner $\mathcal{P}_{\text{new}}$ and the associated factorization (reminiscent of (8)) can be presented as

$$\begin{aligned}\widetilde{\mathcal{P}}_{\text{new}} &= \begin{bmatrix} \tilde{A} & 0 \\ 0 & I \end{bmatrix}\begin{bmatrix} I & 0 \\ -B & I \end{bmatrix}\begin{bmatrix} I & 0 \\ 0 & \tilde{S} \end{bmatrix}\begin{bmatrix} I & \tilde{A}^{-1}B^T(\frac{1}{\alpha}C + I) \\ 0 & I \end{bmatrix} \\ &= \begin{bmatrix} \tilde{A} & B^T(\frac{1}{\alpha}C + I) \\ -B & C \end{bmatrix}, \end{aligned} \tag{22}$$

where $\tilde{S} = C + B\tilde{A}^{-1}B^T(\alpha^{-1}C + I)$ with $\tilde{A}$ being the approximation to A. It is easy to check that solving $\widetilde{\mathcal{P}}_{\text{new}}^{-1}r$ is in essence the same as Algorithm 3 except for A being replaced by $\tilde{A}$.

The underlying idea of the aforementioned two approaches is either to solve the Schur complement linear systems with direct/iterative methods or to obtain an inexpensive approximation of the matrix A. In fact, we can also make use of the properties/structures of the saddle point matrix and develop efficient approximations to the Schur complement matrix. Below are some related ways that give us a hint to this end. In the context of solving PDE-constrained optimization problem, Pearson and Wathen (2012) take $\tilde{S} = (K + 1/\sqrt{\beta}M)M^{-1}(K + 1/\sqrt{\beta}M)$ as the approximation of the Schur complement S, where the saddle point problem is of the following form:

$$\begin{bmatrix} M & 0 & K \\ 0 & \beta M & -M \\ K & -M & 0 \end{bmatrix}\begin{bmatrix} y \\ u \\ p \end{bmatrix} = \begin{bmatrix} b \\ 0 \\ d \end{bmatrix}. \tag{23}$$

It is validated in Pearson and Wathen (2012) that the eigenvalues of $\tilde{S}^{-1}S$ are contained in the interval $[0.5, 1]$. In Axelsson (2019), Axelsson further tailors the approximation $\tilde{S}$ given in Pearson and Wathen (2012) to more standard type of problems. However, we cannot apply the approximation to our problem straightforwardly since the block matrices in (23) fail to satisfy restrictions needed in (1). One may also employ the Schur complement method (Rozložník 2018, p. 34) to solve the Schur complement linear systems. In spite of this, it should be noted that constructing an efficient approximation to the Schur complement is still a challenge for general saddle point problems and deserves a special treatment (Cao et al. 2019). Moreover, the focus of this subsection is on introducing a framework of inexact preconditioners, and a thorough analysis of approximation to Schur complement is beyond the scope of this work. Therefore, we will not dwell on this topic and regard it as an interesting future work. For more on approximations to the Schur complement from different engineering fields, we refer to Deuring (2009), Kay et al. (2002), Loghin and Wathen (2002), Olshanskii and Vassilevski (2007), Pearson and Wathen (2018) and references therein. Numerical examples in Sect. 4 indicate that the inexact preconditioner can be very practical if provided with suitable choices of $\tilde{A}$.

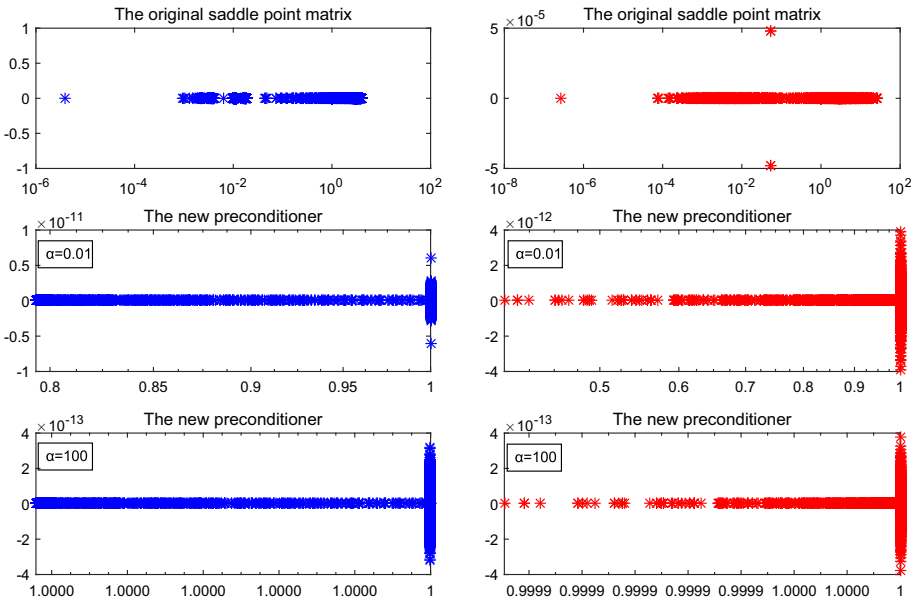

Fig. 1 Spectrum of $\mathcal{A}$ and $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ for 32×32 uniform (left) and stretched (right) grids

4 Numerical experiments

In this section, we present some numerical experiments to show the effectiveness of the new preconditioners for the generalized saddle point problem (1) in terms of the number of iteration steps/restarts (Iter), CPU time in seconds (CPU) and the relative residual norm (Res). The unpreconditioned GMRES with restarting (GMRES(k)) (Saad and Schultz 1986) and EVHSS (Zhang 2018) are used for comparison with the new preconditioners. For completeness, we record both the number of restarts and the number of inner steps in the last restart; see Tables 1, 2, 3, 4, 5. In what follows, the restarting frequency k is set to be 10. The initial guess is chosen to be the zero vector, and algorithms are terminated if $||r_i||_2/||b||_2 < 10^{-8}$ within 1500 restarts, where $r_i = b - Ax_i$ is the ith residual. All experiments are run on a PC using MATLAB 2014b under Windows 10 operating system.

We consider the generalized saddle point problems arising from the discretization of Stokes equation

$$\begin{cases} -\Delta u + \nabla p = f, \\ \nabla \cdot u = 0, \end{cases} \tag{24}$$

in a bounded domain $\Omega \in \mathbb{R}^2$ with suitable boundary conditions, where u is the velocity, p is the pressure, Δ represents the vector Laplacian operator, ∇ stands for the gradient, and $\nabla \cdot u$ is the divergence of u. As in Zhang (2018), the test problems generated here are "leaky" two-dimensional lid-driven cavity problems. The stabilized $Q1 - P0$ finite element is employed for discretization on both uniform and stretched grids. The IFISS software package (Elman et al. 2014) is used to generate the linear systems (1). We follow the practice in Zhang (2018) by deleting the first two rows of the original matrix B generated by IFISS (and thus the first two rows and columns of C) such that the resulting matrix B is of full rank.

Example 1 This example serves to illustrate the eigenvalue clustering effect by the new preconditioner. As stated in Theorem 2, the eigenvalues of the preconditioned matrix $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ are either 1 or of the form (16). If the value of α is sufficiently large, then the nonunit eigenvalues will approach 1; see Remark 1. Figure 1 depicts the eigenvalue distribution of the original saddle point matrix $\mathcal{A}$ and its right-preconditioned counterpart $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ for 32×32 grid of different types. It should be stressed that the five ticks 1.0000 on the x-axis in the lower-left subplot ($\alpha = 100$, 32×32 uniform grid) are different from the value 1. In fact, the real parts of all eigenvalues (corresponding to the x-axis values) therein lie in the interval [0.999973966055229, 1] which are rounded to 1.0000 and displayed. We are now in a position to see the clustering effect by the new preconditioner. Take the left subplots in Fig. 1 for example. Without preconditioning, the eigenvalues of the original matrix $\mathcal{A}$ are well spread between the two points (0,0) and (4,0). However, the eigenvalues become clustered even with a small value of α; see the subplot with $\alpha = 0.01$ on the left of Fig. 1. Such clustering effect becomes more pronounced when α is chosen to be moderately large, say $\alpha = 100$. A well-clustered spectrum is desirable for Krylov subspace methods. Apart from that, the conditioning of the original saddle point matrix $\mathcal{A}$ is also improved via the new preconditioner. For instance, the condition number of the original saddle point matrix $\mathcal{A}$ in the left subplot of Fig. 1 is 1.013×10^6, which indicates the associated linear systems make standard Krylov subspace methods cumbersome. With the proposed preconditioner, however, the condition number decreases to 1.125×10^4 (with $\alpha = 0.01$) and further to 1.011 (with $\alpha = 100$). As suggested in Benzi's survey (Benzi 2002, p. 420), a good preconditioner should be chosen such that the preconditioned coefficient matrix will have a smaller spectral condition number, and/or eigenvalues clustered around 1. In light of this, we conclude that the new preconditioner is appealing, and we shall look into how it affects the convergence of Krylov subspace methods in the following examples.

Example 2 The purpose of this example is threefold: (i) showing the accelerating effect of the new preconditioner when applied to the Krylov subspace method; (ii) analyzing the choice of the experimentally optimal parameter α for both $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$; (iii) comparing the numerical results when the inner sub-linear system of equations in Algorithm 3 are solved either directly or iteratively.

Let us begin with examining the accelerating effect of the new preconditioner when applied to the Krylov subspace method. Two other candidates, including (unpreconditioned) GMRES (Saad and Schultz 1986) and GMRES preconditioned by EVHSS (Zhang 2018), are used for comparison. For the moment, the inner sub-linear system of equations in Algorithm 3 are solved by the direct method. The numerical results are tabulated in Tables 1 and 2. It should be noted that the unpreconditioned GMRES fails to reach the required accuracy within 1500 restarts for most cases here. Therefore, we only display the results of $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ in the two tables. As observed from Tables 1 and 2, both $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ are superior to their unpreconditioned peer. When α is relatively small, the new preconditioner is on par with EVHSS regarding CPU time; sometimes EVHSS is slightly better than the new preconditioner or vice versa. As α grows, however, the new preconditioner outperforms EVHSS for different grid sizes; see, for instance, EVHSS even fails to reach the prescribed accuracy within 1500 restarts (denoted by "–") in Table 2 for 32×32 stretched grid. As pointed out earlier, the quantity "Iter" includes both the number of restarts and the number of inner steps in the last restart. For instance, the notation "3(1)" denotes that EVHSS-preconditioned GMRES uses three restarts and the number of inner iterations in the last restart is 1. To put

it another way, the total number of (inner) steps for EVHSS-preconditioned GMRES is 21[1]. We also note that GMRES preconditioned by $\mathcal{P}_{\text{new}}$ often takes only one restart due to the favorable eigenvalue clustering of $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ which is echoed by Fig. 1 and Remark 2. A similar conclusion can also be drawn from Table 2.

Next we give an in-depth analysis of the influence of the parameter α upon EVHSS and the new preconditioner. In Zhang (2018, Table 1), Zhang presents the experimentally optimal parameters to illustrate the potential of EVHSS. For fairness, we compare EVHSS (using the experimentally optimal parameters) with the new preconditioner. As stated in Remark 1, large values of α often bring about a tight spectrum. However, this does not imply that we have to use unduly large values of α. Instead, a moderately large value, say $\alpha = 100$, is sufficient for practical use; see Fig. 1. To further justify our statement, we carry out more detailed comparisons between $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ with varying values of α for different uniform grids in Figs. 2, 3, 4. As given in Zhang (2018, Table 1), the experimentally optimal parameters of EVHSS for 16×16, 32×32, and 64×64 are found in the interval $(0, 0.01)$ (Zhang 2018). Therefore, we plot the iteration step and CPU time curves on different scales, that is, the intervals in which α locates in Figs. 2, 3, 4 are, respectively, $(0, 0.01]$, $(0.01, 1]$ and $(1, 100]$. Some remarks are in order. Owing to a clustered spectrum, the new preconditioner requires fewer iteration steps than EVHSS for α ranging from 1 to 100. In Fig. 2, EVHSS achieves its optimal performance in terms of the CPU time and is slightly better than the new preconditioner when α varies from 0 to 0.01. However, the advantage of the new preconditioner dominates when α is greater than 0.01; see Figs. 3, 4. Besides, there are many oscillations in the iteration step and CPU time curves of EVHSS which implies EVHSS is sensitive to the choice of α (especially when α becomes large). Fortunately, this undesirable property does not carry over to the new preconditioner. In this sense, the new preconditioner is more applicable than EVHSS. To figure out the practical choice of α for the new preconditioner, we take for example the 32×32 case (bottom subplots) in Fig. 4. In that case, the improvement regarding total iteration steps and CPU time becomes negligible for α greater than 20. Thus, each value between 20 and 100 can be a reasonable choice of α for the new preconditioner. In accordance with Remark 2 and Fig. 1, we adopt a relatively large α, that is, $\alpha = 100$ as the experimental optimal value for the new preconditioner. The results are listed in Table 3. In Zhang (2018), the restarting frequency in GMRES(k) is $k = 30$ (instead of $k = 10$ as chosen here) and the stopping tolerance is 10^{-6} (instead of 10^{-8} as chosen here), which explains the difference in the number of iterations between Zhang (2018, Tables 2–3) and Table 3. For most tests, the new preconditioner reaches a higher accuracy with shorter CPU time than that of EVHSS; see Table 3. This verifies the effectiveness of the new preconditioner.

Finally, we shall touch upon the matter of inner linear systems solvers in Algorithm 3. We use previously the direct methods, say the MATLAB command backslash to solve the inner sub-linear systems, the numerical results of which are tabulated in Table 3. However, we can also use iterative methods to obtain approximations to these sub-linear systems. This inner iteration, together with the outer GMRES iteration, yields the well-known inner–outer iteration (Simoncini and Szyld 2003). Here we use the restarted GMRES method to solve the sub-linear systems with an inner tolerance 10^{-4} and a maximum 100 restarts. For fair comparison, the inner linear systems in EVHSS are also solved in an iterative manner. Choices of α are the same as in Table 3. The results are presented in Table 4. Some remarks are available by comparing Table 4 with Table 3. Loosely speaking, the preconditioners

[1] In this case, EVHSS-preconditioned GMRES uses two full restarts (each of 10 inner steps) and a restart with just 1 inner step. Thus, the total number of inner steps is $(3 - 1) \times 10 + 1 = 21$.

Table 1 Numerical results for Stokes equation with different uniform grids

Grids	α	EVHSS			$\mathcal{P}_{\text{new}}$		
		Iter	CPU	Res	Iter	CPU	Res
16×16	0.01	3(1)	0.0537	4.5867e−09	1(10)	0.1358	7.3705e−09
	1	14(2)	0.2930	9.4243e−09	1(3)	0.0480	6.4302e−09
	100	54(2)	1.126	9.757e−09	1(2)	0.0344	3.4185e−10
32×32	0.01	7(1)	1.8365	7.955e−09	1(6)	1.8372	7.2844e−09
	1	59(2)	17.0561	9.9688e−09	1(3)	0.9124	7.4029e−11
	100	162(8)	47.1439	9.9797e−09	1(2)	0.6199	1.5619e−11
64×64	0.01	114(1)	864.3565	9.8015e−09	1(5)	56.2271	1.4616e−10
	1	350(10)	2648.0636	9.9819e−09	1(2)	22.4823	6.9718e−09
	100	251(10)	1897.4545	9.9995e−09	1(2)	22.4291	6.9777e−13

Table 2 Numerical results for Stokes equation with different stretched grids

Grids	α	EVHSS			$\mathcal{P}_{\text{new}}$		
		Iter	CPU	Res	Iter	CPU	Res
16×16	0.01	4(4)	0.3942	9.1448e−09	2(6)	0.6060	5.8924e−09
	1	500(6)	10.0706	9.9994e−09	1(4)	0.0585	8.1187e−10
	100	753(8)	14.9417	9.9992e−09	1(2)	0.0347	1.2047e−09
32×32	0.01	178(9)	51.3762	9.9967e−09	1(10)	3.1710	5.9088e−09
	1	1330(7)	380.4173	9.997e−09	1(3)	0.9144	1.7114e−09
	100	–	–	–	1(2)	0.6108	4.5475e−11
64×64	0.01	37(10)	279.4618	9.9006e−09	1(7)	78.8612	1.2548e−09
	1	312(3)	2334.8392	9.9753e−09	1(3)	33.8358	2.0001e−11
	100	436(9)	3283.5052	9.9917e−09	1(2)	22.5768	1.3983e−12

$\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ with a direct inner solver usually ends up with a higher accuracy than their counterparts with an iterative inner solver, though there exist occasional exceptions. Besides, preconditioners with a direct inner solver appear more suitable for the stretched grids, while preconditioners with an iterative inner solver seem more appropriate for the uniform grids. Moreover, choosing an iterative or a direct inner linear systems solver has less impact on $\mathcal{P}_{\text{new}}$ than on $\mathcal{P}_{\text{EVHSS}}$ in terms of iteration steps and CPU time; see, for instance, the stretched cases in Tables 3 and 4. For either direct or iterative approach, the new preconditioner surpasses EVHSS regarding iteration steps and CPU time.

Example 3 Example 2 shows that the new preconditioner is competitive with EVHSS in speeding up the convergence of GMRES. As reported in Sect. 3.3 and Algorithm 3, however, a sub-linear system with the coefficient matrix S needs to be solved per iteration. If A is large and relatively dense, then solving the linear systems involving S can be rather time-consuming; it can be monitored by evoking the MATLAB profiling tool `Profiler`. As noted in (22), it poses no difficulty if inexpensive approximations to the matrix A in S are available. Several candidates are at our disposal, including the diagonal, triangular or incomplete LU factorization matrices of A (Bai et al. 2005b; Benzi et al. 2005).

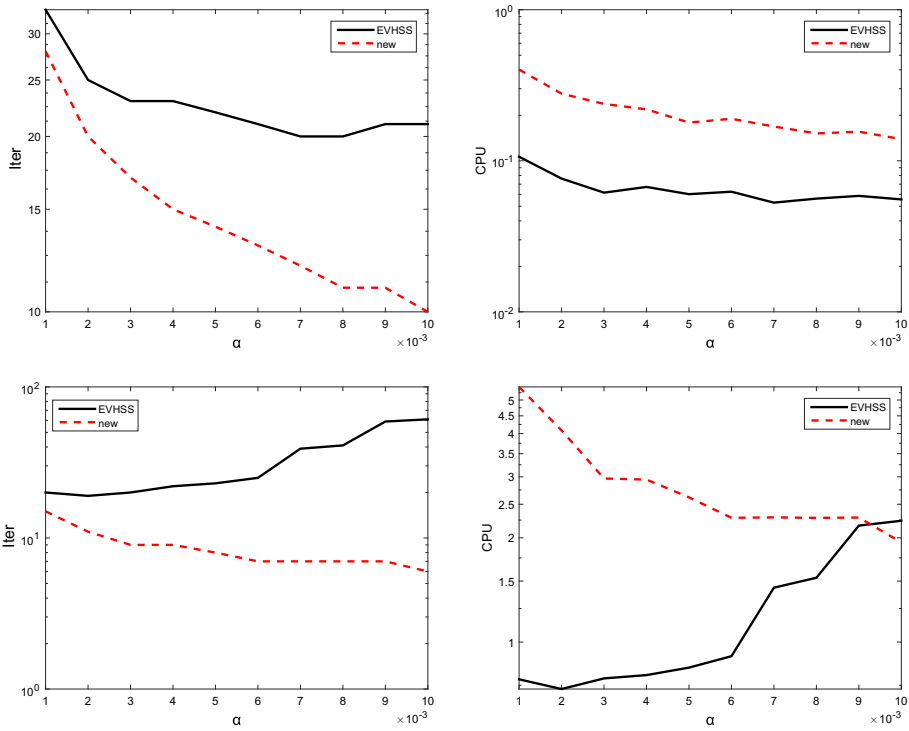

Fig. 2 Total iteration steps and CPU time of GMRES preconditioned by $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ with α ranging from 0 to 0.01 for 16×16 (top) and 32×32 (bottom) uniform grids, respectively

Table 3 Numerical results of EVHSS and $\mathcal{P}_{\text{new}}$ with practical optimal values of α, where the inner linear systems are solved by a direct method

Grids		EVHSS			$\mathcal{P}_{\text{new}}$		
Type	Size	Iter	CPU	Res	Iter	CPU	Res
Uniform	16×16	3(2)	0.0779	3.2715e−09	1(2)	0.0354	3.4185e−10
	32×32	2(9)	0.5667	4.6684e−09	1(2)	0.6035	1.5619e−11
	64×64	4(10)	31.4061	6.6043e−09	1(2)	22.4257	6.9777e−13
Stretched	16×16	4(4)	0.0909	5.8081e−09	1(2)	0.0355	1.2047e−09
	32×32	15(4)	4.0856	9.966e−09	1(2)	0.6061	4.5475e−11
	64×64	4(9)	30.0677	9.0157e−09	1(2)	22.5537	1.3983e−12

This example serves to demonstrate the potential of the inexact preconditioner $\widetilde{\mathcal{P}}_{\text{new}}$ when compared with the exact preconditioner $\mathcal{P}_{\text{new}}$. For the moment, we only consider the simplest case by superseding A with its diagonal part, i.e., $\tilde{A} = \texttt{diag}(\texttt{diag}(A))$ in (22). It does not imply that using the diagonal part to replace A is optimal. Numerical practice with other inexact variants or finding efficient approximations of the Schur complement matrix needs further investigation and thus is regarded as an important future work. The numerical results are given in Table 5, where the inner linear systems are solved by the direct method. For coarse grids, as indicated in Table 5, the exact preconditioner $\mathcal{P}_{\text{new}}$ is sometimes better than

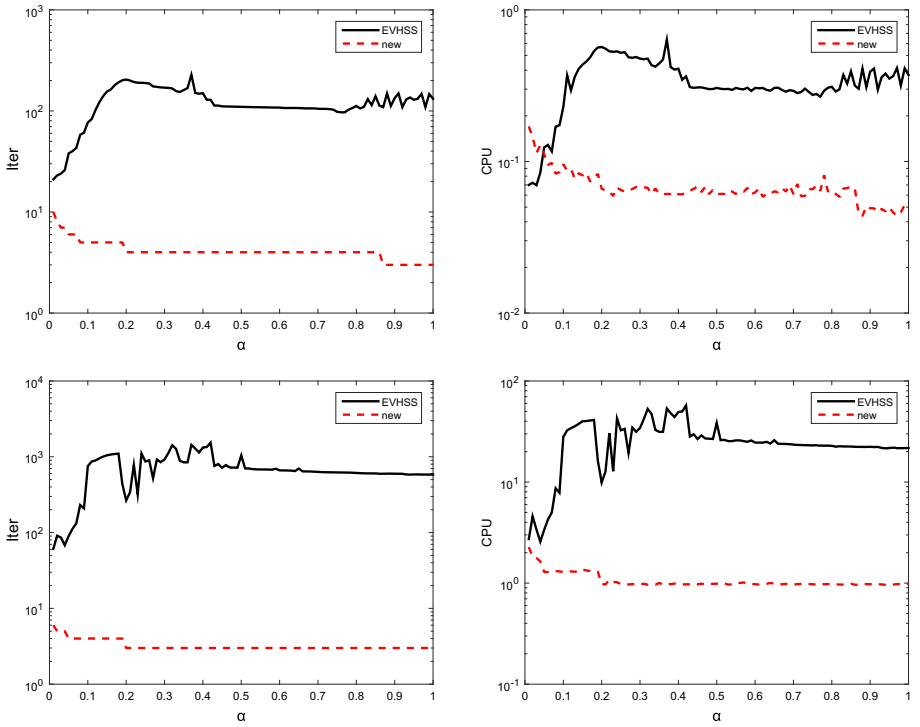

Fig. 3 Total iteration steps and CPU time of GMRES preconditioned by $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ with α ranging from 0.01 to 1 for 16×16 (top) and 32×32 (bottom) uniform grids, respectively

Table 4 Numerical results of EVHSS and $\mathcal{P}_{\text{new}}$ with practical optimal values of α, where the inner linear systems are solved by an iterative method

Grids		EVHSS			$\mathcal{P}_{\text{new}}$		
Type	Size	Iter	CPU	Res	Iter	CPU	Res
Uniform	16×16	3(1)	0.1316	6.6153e−09	1(2)	0.0341	5.9771e−09
	32×32	2(9)	0.8015	3.8036e−09	1(2)	0.4573	9.5494e−09
	64×64	5(9)	22.571	9.3383e−09	1(2)	12.0615	8.7442e−09
Stretched	16×16	4(4)	0.2169	7.9022e−09	1(3)	0.0701	1.5547e−12
	32×32	15(10)	8.9815	9.5483e−09	1(2)	0.8052	7.8152e−09
	64×64	4(9)	41.1991	9.5302e−09	1(2)	25.7432	9.9792e−09

its inexact counterpart $\widetilde{\mathcal{P}}_{\text{new}}$ (in CPU time) by a small margin or vice versa; see, for instance, the 16×16 case in Table 5. As the grids get finer, however, the inexact preconditioner becomes much better than the exact one in terms of CPU time, albeit with more iteration steps. This can be exemplified by the case 128×128 grid with $\alpha = 100$ where the CPU time ratio of the inexact preconditioner to its exact counterpart is merely 10.3%.

Another interesting observation is that the CPU time exhausted by $\widetilde{\mathcal{P}}_{\text{new}}$ is less sensitive to the grid size compared with those by EVHSS and $\mathcal{P}_{\text{new}}$; cf. Tables 1, 2, 3, 4, and 5. This drops us a hint that the inexact preconditioner is more appropriate for solving large-scale linear

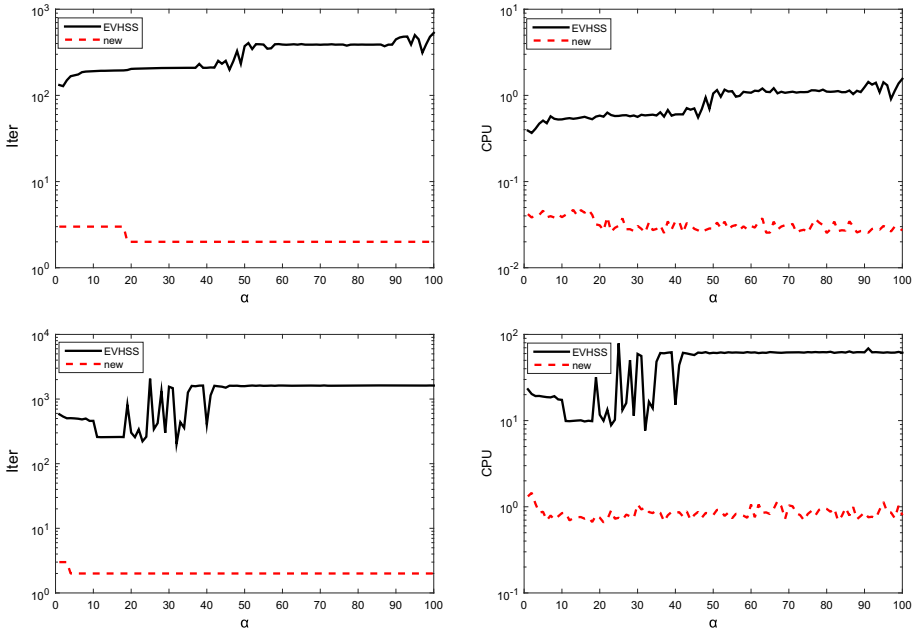

Fig. 4 Total iteration steps and CPU time of GMRES preconditioned by $\mathcal{P}_{\text{EVHSS}}$ and $\mathcal{P}_{\text{new}}$ with α ranging from 1 to 100 for 16×16 (top) and 32×32 (bottom) uniform grids, respectively

systems (1). Apart from that, it appears that the value of α has much less impact on the inexact preconditioner than it does on the exact prototype. Since the spectrum of the preconditioned matrix is closely related to GMRES convergence, it is inspiring to ask: how well do the eigenvalues cluster after using an inexact preconditioner and how does it affect the related convergence? To answer this question, we plot the eigenvalue distribution of $\mathcal{A}\widetilde{\mathcal{P}}_{\text{new}}^{-1}$ in Fig. 5. As shown in Fig. 5, the eigenvalues of $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ become more clustered as α goes up, which in turn leads to a faster convergence regarding iteration steps. Nevertheless, this does not apply to the spectrum of $\mathcal{A}\widetilde{\mathcal{P}}_{\text{new}}^{-1}$. In fact, the spectrum of $\mathcal{A}\widetilde{\mathcal{P}}_{\text{new}}^{-1}$ does not change too much even if α increases from 0.01 to 100; see the right subplots in Fig. 5. This explains why the acceleration is not so remarkable for $\widetilde{\mathcal{P}}_{\text{new}}$ when compared with $\mathcal{P}_{\text{new}}$ in terms of iteration steps. However, since the computational overhead involved in using the inexact preconditioner $\widetilde{\mathcal{P}}_{\text{new}}$ is far less than that of the exact preconditioner $\mathcal{P}_{\text{new}}$, then the CPU time of the former can be expected to be much less than that of the latter. In most practical situations, it is the CPU time that we are really concerned with. With this insight in hand, we conclude that $\widetilde{\mathcal{P}}_{\text{new}}$ can be an efficient alternative for $\mathcal{P}_{\text{new}}$ when the linear systems (1) is large and sparse.

5 Conclusion

We present a fast HSS-type iterative scheme for solving the generalized saddle point problem. Theoretical results on unconditional convergence, eigenvalue distribution of the preconditioned saddle point matrix and the speed-up effect on GMRES have been established. Based on the new preconditioner, a framework for developing practical inexact preconditioners is also proposed. Numerical experiments show that the new preconditioner and its practical

Table 5 Numerical results for exact and inexact preconditioners on uniform grids

Grids	α	$\mathcal{P}_{\text{new}}$			$\widetilde{\mathcal{P}}_{\text{new}}$		
		Iter	CPU	Res	Iter	CPU	Res
16×16	0.01	1(10)	0.1580	7.3705e−09	4(10)	0.0688	7.4726e−09
	1	1(3)	0.0407	6.4302e−09	4(10)	0.0837	8.0367e−09
	100	1(2)	0.0311	3.4185e−10	4(10)	0.0679	8.0227e−09
32×32	0.01	1(6)	1.9770	7.2844e−09	10(4)	0.6789	9.7079e−09
	1	1(3)	0.9280	7.4029e−11	10(4)	0.6863	8.7563e−09
	100	1(2)	0.6185	1.5619e−11	10(4)	0.6962	8.7604e−09
64×64	0.01	1(5)	57.4575	1.4616e−10	26(7)	9.6544	9.9047e−09
	1	1(2)	22.7494	6.9718e−09	26(10)	9.6785	9.915e−09
	100	1(2)	22.6678	6.9777e−13	26(10)	9.7584	9.9139e−09
128×128	0.01	1(3)	2462.1403	8.9629e−09	83(8)	170.9418	9.9581e−09
	1	1(2)	1696.4994	3.0923e−10	83(8)	171.3698	9.9678e−09
	100	1(2)	1665.8702	3.1361e−14	83(8)	171.2660	9.9652e−09

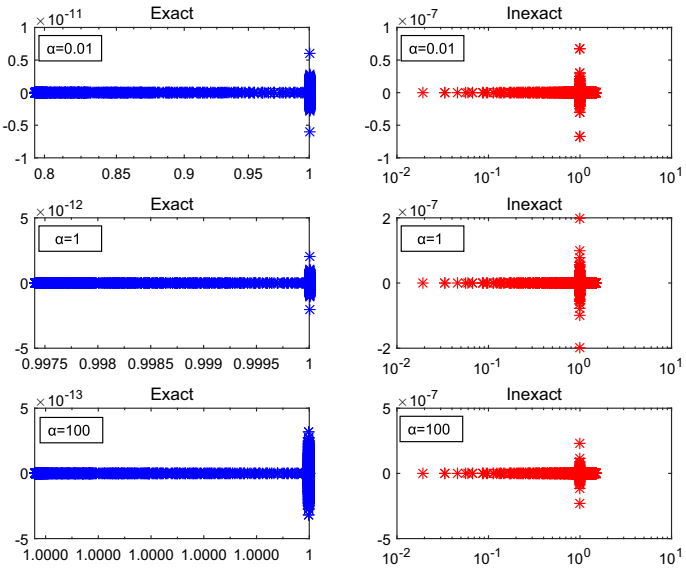

Fig. 5 Eigenvalue distributions of $\mathcal{A}\mathcal{P}_{\text{new}}^{-1}$ and $\mathcal{A}\widetilde{\mathcal{P}}_{\text{new}}^{-1}$ for 32×32 uniform grid

inexact variants are superior to EVHSS for most cases in terms of the CPU time. Future work includes constructing efficient approximations to the Schur complement for large, sparse and structured saddle point linear systems from different applied fields.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Nos. 11601323,11801362) and the Key Discipline Fund (Multivariate Statistical Analysis 2018-2019) of College of Arts and Sciences in Shanghai Maritime University. The authors would like to thank Dr. Ju-Li Zhang of Shanghai University of Engineering Science for helpful discussions. The first author also would like to thank Professor Jun-Feng Yin for gracious invitation and hospitality during his visit to Tongji University.

References

Axelsson O (2019) Optimality properties of a square block matrix preconditioner with applications. Comput Math Appl. https://doi.org/10.1016/j.camwa.2019.09.024

Bai Z-Z (2015) Motivations and realizations of Krylov subspace methods for large sparse linear systems. J Comput Appl Math 283:71–78

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

Bai Z-Z, Golub GH, Ng MK (2003) Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. SIAM J Matrix Anal Appl 24:603–626

Bai Z-Z, Golub GH, Lu L-Z (2005a) Block triangular and skew-Hermitian splitting methods for positive-definite linear systems. SIAM J Sci Comput 26:844–863

Bai Z-Z, Parlett BN, Wang Z-Q (2005b) On generalized successive overrelaxation methods for augmented linear systems. Numer Math 102:1–38

Bai Z-Z, Golub GH, Ng MK (2008) On inexact Hermitian and skew-Hermitian splitting methods for non-Hermitian positive definite linear systems. Linear Algebra Appl 428:413–440

Beik FPA, Benzi M, Chaparpordi SHA (2017) On block diagonal and block triangular iterative schemes and preconditioners for stabilized saddle point problems. J Comput Appl Math 326:15–30

Benzi M (2002) Preconditioning techniques for large linear systems: a survey. J Comput Phys 182:418–477

Benzi M, Golub GH, Liesen J (2005) Numerical solution of saddle point problems. Acta Numer 14:1–137

Betts JT (2001) Practical methods for optimal control using nonlinear programming. SIAM, Philadelphia

Cao Y, Yi S-C (2016) A class of Uzawa-PSS iteration methods for nonsingular and singular non-Hermitian sddle point problems. Appl Math Comput 275:41–49

Cao Y, Yao L-Q, Jiang M-Q (2013) A modified dimensional split preconditioner for generalized saddle point problems. J Comput Appl Math 250:70–82

Cao Y, Dong J-L, Wang Y-M (2015) A relaxed deteriorated PSS preconditioner for nonsymmetric saddle point problems from the steady Navier–Stokes equation. J Comput Appl Math 273:41–60

Cao Y, Ren Z-R, Shi Q (2016) A simplified HSS preconditioner for generalized saddle point problems. BIT Numer Math 56:423–439

Cao Y, Ren Z, Yao L (2019) Improved relaxed positive-definite and skew-Hermitian splitting preconditioners for saddle point problems. J Comput Math 37:95–111

Chen F (2018) On convergence of EVHSS iteration method for solving generalized saddle-point linear systems. Appl Math Lett 86:30–35

de Sturler E, Liesen J (2005) Block-diagonal and constraint preconditioners for nonsymmetric indefinite linear systems. Part I: theory. SIAM J Sci Comput 26:1598–1619

Deuring P (2009) Eigenvalue bounds for the Schur complement with a pressure convection-diffusion preconditioner in incompressible flow computations. J Comput Appl Math 228:444–457

Elman HC, Ramage A, Silvester DJ (2014) IFISS: a computational laboratory for investigating incompressible flow problems. SIAM Rev 56:261–273

Horn RA, Johnson CR (1990) Matrix analysis. Cambridge University Press, Cambridge

Huang T-Z, Wu S-L, Li C-X (2009) The spectral properties of the Hermitian and skew-Hermitian splitting preconditioner for generalized saddle point problems. J Comput Appl Math 229:37–46

Kay D, Loghin D, Wathen AJ (2002) A preconditioner for the steady-state Navier–Stokes equations. SIAM J Sci Comput 24:237–256

Ke Y-F, Ma C-F (2017) The dimensional splitting iteration methods for solving saddle point problems arising from time-harmonic eddy current models. Appl Math Comput 303:146–164

Keller C, Gould NIM, Wathen AJ (2000) Constraint preconditioning for indefinite linear systems. SIAM J Matrix Anal Appl 21:1300–1317

Li C-X, Wu S-L (2015) A single-step HSS method for non-Hermitian positive definite linear systems. Appl Math Lett 44:26–29

Liang Z-Z, Zhang G-F (2016) Two new variants of the HSS preconditioner for regularized saddle point problems. Comput Math Appl 72:603–619

Liao L-D, Zhang G-F (2019) A generalized variant of simplified HSS preconditioner for generalized saddle point problems. Appl Math Comput 346:790–799

Ling S-T, Liu Q-B (2017) New local generalized shift-splitting preconditioners for saddle point problems. Appl Math Comput 302:58–67

Loghin D, Wathen AJ (2002) Schur complement preconditioners for the Navier–Stokes equations. Int J Numer Methods Fluids 40:403–412

Murphy MF, Golub GH, Wathen AJ (2000) A note on preconditioning for indefinite linear systems. SIAM J Sci Comput 21:1969–1972

Olshanskii MA, Vassilevski YV (2007) Pressure Schur complement preconditioners for the discrete Oseen problem. SIAM J Sci Comput 29:2686–2704
Pan J-Y, Ng MK, Bai Z-Z (2006) New preconditioners for saddle point problems. Appl Math Comput 172:762–771
Pearson JW, Wathen AJ (2012) A new approximation of the Schur complement in preconditioners for PDE-constrained optimization. Numer Linear Algebra Appl 19:816–829
Pearson JW, Wathen AJ (2018) Matching Schur complement approximations for certain saddle-point systems. Contemporary computational mathematics—a celebration of the 80th birthday of Ian Sloan
Rozložník M (2018) Saddle-point problems and their iterative solution. Birkhäuser, Basel
Saad Y (2003) Iterative methods for sparse linear systems. SIAM, Philadelphia
Saad Y, Schultz MH (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 7:856–869
Shen S-Q (2014) A note on PSS preconditioners for generalized saddle point problems. Appl Math Comput 237:723–729
Shen Q-Q, Shi Q (2016) Generalized shift-splitting preconditioners for nonsingular and singular generalized saddle point problems. Comput Math Appl 72:632–641
Shen H-L, Wu H-Y, Shao X-H, Song X-D (2019) The PPS method-based constraint preconditioners for generalized saddle point problems. Comput Appl Math 38(1):21
Simoncini V (2004) Block triangular preconditioners for symmetric saddle-point problems. Appl Numer Math 49:63–80
Simoncini V, Szyld DB (2003) Flexible inner-outer Krylov subspace methods SIAM. J Numer Anal 40:2219–2239
Wathen AJ (2015) Preconditioning. Acta Numer 24:329–376
Yun JH (2013) Variants of the Uzawa method for saddle point problem. Comput Math Appl 65:1037–1046
Zeng M-L, Ma C-F (2016) A parameterized SHSS iteration method for a class of complex symmetric system of linear equations. Comput Math Appl 71:2124–2131
Zhang J-L (2018) An efficient variant of HSS preconditioner for generalized saddle point problems. Numer Linear Algebra Appl 25:e2166. https://doi.org/10.1002/nla.2166
Zhang J, Shang J (2010) A class of Uzawa-SOR methods for saddle point problems. Appl Math Comput 216:2163–2168
Zhang G-F, Ren Z-R, Zhou Y-Y (2011) On HSS-based constraint preconditioners for generalized saddle-point problems. Numer Algorithms 57:273–287
Zhang J-L, Gu C-Q, Zhang K (2014) A relaxed positive-definite and skew-Hermitian splitting preconditioner for saddle point problems. Appl Math Comput 249:468–479
Zhang K, Zhang J-L, Gu C-Q (2017) A new relaxed PSS preconditioner for nonsymmetric saddle point problems. Appl Math Comput 308:115–129
Zhang C-H, Wang X, Tang X-B (2019) Generalized AOR method for solving a class of generalized saddle point problems. J Comput Appl Math 350:69–79