Research Article

Wei-Hua Luo*, Bruno Carpentieri, and Jun Guo

A dimension expanded preconditioning technique for block two-by-two linear equations

https://doi.org/10.1515/dema-2023-0260
received December 1, 2022; accepted June 17, 2023

Abstract: In this article, we introduce a novel block preconditioner for block two-by-two linear equations by expanding the dimension of the coefficient matrix. Theoretical results on the eigenvalues distribution of the preconditioned matrix are obtained, and a feasible implementation is discussed. Some numerical examples, including the solution of the Navier-Stokes equations, are presented to support the theoretical findings and demonstrate the preconditioner's efficiency.

Keywords: block two-by-two, linear equations, preconditioner, Navier-Stokes equations

MSC 2020: 65F08, 65F10

1 Introduction

In this article, we focus on solving the block two-by-two system of linear equations of the form

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \end{bmatrix}, \tag{1.1}$$

where $A \in \mathbb{R}^{m\times m}$, $B \in \mathbb{R}^{m\times n}$, $C \in \mathbb{R}^{n\times m}$, and $D \in \mathbb{R}^{n\times n}$ are sparse matrices with $m \geq n$, A is nonsingular, $\boldsymbol{x}_1, \boldsymbol{b}_1 \in \mathbb{R}^{m\times 1}$, $\boldsymbol{x}_2, \boldsymbol{b}_2 \in \mathbb{R}^{n\times 1}$. Many engineering problems are discretized into linear systems of form (1.1) using numerical methods such as finite difference methods (FDMs), finite element methods (FEMs), and spectral methods. Using FEMs to solve the Navier-Stokes equations, for instance, yields (1.1), where A and D are symmetric positive definite (SPD) and symmetric positive semi-definite (SPSD), respectively, and $C = -B^T$ [1,2]. The discretization of some electromagnetic scattering and quantum mechanics problems [3,4] results in a complex symmetric indefinite linear system

$$(W + iT)\mathbf{x} = b, \quad i = \sqrt{-1}, W, \quad T \in \mathbb{R}^{n\times n}, \quad \mathbf{x}, b \in \mathbb{C}^n, \tag{1.2}$$

which can be transformed into a special case of (1.1) (see [5,6]), namely,

$$\begin{bmatrix} T & W \\ W & T \end{bmatrix} \begin{bmatrix} x \\ -y \end{bmatrix} = \begin{bmatrix} g \\ f \end{bmatrix} \equiv b. \tag{1.3}$$

* **Corresponding author: Wei-Hua Luo,** School of Mathematics and Physics, Hunan University of Arts and Science, Changde, Hunan 415000, P. R. China, e-mail: huaweiluo2012@163.com
Bruno Carpentieri: Faculty of Engineering, Free University of Bozen-Bolzano, Bolzano, Italy, e-mail: bcarpentieri@gmail.com
Jun Guo: College of Applied Mathematics, Chengdu University of Information Technology, Chengdu, Sichuan 611130, P. R. China, e-mail: junguo0407@cuit.edu.cn

Further examples of (1.1) with $B \neq \pm C^T$ and $B \neq \pm C$ can be found in the fields of Economics, Computational Fluid Dynamics, and so on. Table 1 contains some examples taken from the University of Florida Sparse Matrix Collection (URL: https://sparse.tamu.edu/, see, [7,8]).

When m and n are large enough, researchers are more likely to adopt iterative approaches to solve (1.1), since they can alleviate the high memory and computing costs of direct methods, assuming some effective preconditioners are available to lower the number of iterations. For some special cases of problem (1.1), many excellent preconditioning techniques have been proposed so far. For example, Benzi and Golub [9] presented a Hermitian and skew-Hermitian splitting (HSS) preconditioner based on the Hermitian and skew-Hermitian splitting iterative method for the case of SPD A, SPSD D, and $C = \pm B^T$ (also known as a generalized saddle point problem). Bai and Cao et al. [10,11] investigated a simplified HSS preconditioner (SHSS) that approximates the coefficient matrix more closely than HSS. Also, Cao et al. [12,13] successively presented a modified dimensional split (MDS) preconditioner and relaxed splitting preconditioner, which can be seen as excellent generalizations of the preconditioners from [14,15]. Especially, in [12] a complete convergence theorem of the corresponding MDS stationary iteration method was proved. Zhang [16] presented an efficient variant of HSS (EVHSS) preconditioner by combining it with the relaxation preconditioning technique. Other strategies can be found in [17,18] for constraint preconditioners, [19] for splitting preconditioner, [20] for incomplete LU (ILU) preconditioner, and so on. Zhang and Dai [5,21] and Shen and Shi [22] designed a block preconditioner, a splitting preconditioner, and a variant HSS preconditioner for cases (1.2) or (1.3). Many more results about preconditioning methods have been explored for the case of $D = 0, B = \pm C^T$ (called a saddle problem), including but not limited to dimensional split (DS) [14], relaxed dimensional factorization [15], relaxed physical factorization [23], dimension-wise splitting iteration with selective relaxation [24], HSS-types [9,25], and the references therein. Bai [26] studied a class of structured preconditioners through matrix transformation and matrix approximations for the general case (1.1). Other methods include product (PS) preconditioner [27], alternating splitting preconditioner [28,29], extended positive-definite and skew-Hermitian splitting (EPSS) preconditioner [30], block positive semi-definite splitting (BPS) preconditioner [31], and so on.

It is not difficult to find that almost all of the aforementioned results require the solution of a subproblem that includes at least a Schur complement structure. In [11,16,21], for example,

$$\begin{bmatrix} \alpha I & B^T \\ -B & C \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \begin{bmatrix} A & B^T \\ -B & \alpha I \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \begin{bmatrix} \alpha I & -W \\ W & T \end{bmatrix}\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{1.4}$$

need to be computed, respectively. In [26,27], the subsystems

$$(I - \overline{C} \cdot \overline{B})\overline{u} = \overline{v}, (D - CG^{-1}B)\overline{u} = \overline{v} \tag{1.5}$$

should be dealt with, respectively, where G is an approximation of A, and

$$\overline{B} = L_A^{-1} B R_D^{-1}, \quad \overline{C} = L_D^{-1} C R_A^{-1},$$

where L_A, R_A, L_D, and R_D satisfy

$$A = L_A J_A R_A, \quad D = L_D J_D R_D,$$

with J_A and J_D two matrices approximating the identity matrices of $\mathbb{R}^{m\times m}$ and $\mathbb{R}^{n\times n}$, respectively. Solving the systems (1.4) or (1.5) may be costly in terms of computer memory and CPU time when the density of B and C is relatively high and m and n are sufficiently large.

Table 1: Features of matrices in Example 3

Name	Background	m	n	nnz	Condition number
dw256A	Electromagnetics problem	400	112	2,480	3.7165×10^4
ck656	2D/3D problem	500	156	3,884	1.1802×10^7
poli	Economic problem	3,000	1,008	8,188	6.3993×10^1
polilarge	Economic problem	13,575	2,000	33,033	48.0022
venkat25	Computational fluid dynamics problem	40,000	22,424	1,717,763	1.2165×10^8
poisson3Db	Computational fluid dynamics problem	45,000	40,623	2,374,949	1.6555×10^5

In this article, we present a novel preconditioner inspired by the idea of [32]. The key idea of this technique is to expand the dimension of the coefficient matrix and transform it into an augmented equivalent system. Implementing this dimension expanding (DE) preconditioner saves us the trouble of solving the aforementioned subproblems that are similar to (1.4) or (1.5); instead, we only need to solve the subsystems

$$Au_1 = v_1, [(1-\alpha)I - \alpha D]u_2 = v_2. \tag{1.6}$$

The rest of this article is structured as follows. Section 2 describes in detail the novel preconditioner and discusses its practical implementation. The eigenvalue distributions of the preconditioned matrix is theoretically examined. Later, in Section 3, some practical numerical examples, including the Navier-Stokes equations, are provided to demonstrate the efficiency of the novel preconditioner when compared to other existing techniques. Finally, Section 4 draws conclusions.

2 Description of the dimension expanded preconditioner (shortly, P_{DE})

To begin, we transform (1.1) into an equivalent system of the form

$$\begin{bmatrix} \tilde{B} & \tilde{A} \\ D & C \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{x}_1 \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{b}}_1 \\ \boldsymbol{b}_2 \end{bmatrix}, \tilde{A} = A + BC, \tilde{B} = B + BD, \tilde{\boldsymbol{b}}_1 = \boldsymbol{b}_1 + B\boldsymbol{b}_2. \tag{2.1}$$

Second, we augment (2.1) into a 3×3 block system of the form

$$Hu = \begin{bmatrix} I & \mathbf{0} & I \\ \alpha_1 B + BD & A + BC & (\alpha_1 - 1)B \\ I + D & C & I \end{bmatrix} \begin{bmatrix} \boldsymbol{x}_2 \\ \boldsymbol{x}_1 \\ \boldsymbol{x}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{0} \\ \tilde{\boldsymbol{b}}_1 \\ \boldsymbol{b}_2 \end{bmatrix} \tag{2.2}$$

by taking $\alpha_1 = \frac{\alpha_2 - 2}{\alpha_2 - 1}$, where α_2 is a chosen parameter close to 1.

It is worth noting here that the goal of performing an equivalent transformation from (1.1) into (2.1) and augmenting (2.1) into (2.2) is only to obtain the following preconditioner (abbreviated as P_{DE}), for which we will see that solving $P_{\mathrm{DE}}u = v$ can be easily and cheaply implemented.

The preconditioner P_{DE} can then be taken as follows:

$$P_{\mathrm{DE}} = \begin{bmatrix} I & \mathbf{0} & \alpha_2 I \\ \alpha_1 B + BD & A + BC & (\alpha_1 - 1)B \\ I + D & C & I \end{bmatrix} \tag{2.3}$$

for the system (2.2). The difference between H and P_{DE} is clearly

$$\Delta H = H - P_{\mathrm{DE}} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & (1-\alpha_2)I \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{2.4}$$

Furthermore, the following results apply to the distributions of the eigenvalues of the preconditioned matrix $P_{\mathrm{DE}}^{-1}H$.

Theorem 1. *The preconditioned matrix $P_{\mathrm{DE}}^{-1}H$ has an eigenvalue of 1 with a multiplicity of at least $n + m$, and all of the remaining eigenvalues λ's are eigenvalues of $V^{-1}(CA^{-1}B - D)$, where $V = (1-\alpha_2)I - \alpha_2 D$.*

Proof. By using some simple computations, and noting that $\alpha_1 + \alpha_2 - \alpha_1\alpha_2 - 2 = 0$, it is easy to obtain

$$P_{\mathrm{DE}}^{-1} = \begin{bmatrix} I & \alpha_2 V^{-1}CA^{-1} & -\alpha_2 V^{-1} \\ \mathbf{0} & A^{-1} & \mathbf{0} \\ \mathbf{0} & -V^{-1}CA^{-1} & V^{-1} \end{bmatrix} \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ (1-\alpha_1)B & I & -B \\ -(I+D) & \mathbf{0} & I \end{bmatrix}.$$

From the expression of V, we deduce that $\alpha_2 V^{-1} - V^{-1} + \alpha_2 V^{-1} D = -I$, and hence,

$$P_{\mathrm{DE}}^{-1}\Delta H = \begin{bmatrix} I & \alpha_2 V^{-1}CA^{-1} & -\alpha_2 V^{-1} \\ \mathbf{0} & A^{-1} & \mathbf{0} \\ \mathbf{0} & -V^{-1}CA^{-1} & V^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & (1-\alpha_2)I \\ \mathbf{0} & \mathbf{0} & -B \\ \mathbf{0} & \mathbf{0} & (\alpha_2 - 1)(I + D) \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{0} & W_1 \\ \mathbf{0} & \mathbf{0} & W_2 \\ \mathbf{0} & \mathbf{0} & W_3 \end{bmatrix},$$

where

$$W_1 = (1-\alpha_2)I + \alpha_2(1-\alpha_2)V^{-1}[(1-\alpha_1)CA^{-1}B + I + D], \quad W_2 = -A^{-1}B, \quad W_3 = V^{-1}[CA^{-1}B - D] - I.$$

As a result,

$$P_{\mathrm{DE}}^{-1}H = \begin{bmatrix} I & \mathbf{0} & W_1 \\ \mathbf{0} & I & W_2 \\ \mathbf{0} & \mathbf{0} & V^{-1}[CA^{-1}B - D] \end{bmatrix} \tag{2.5}$$

holds, and we know that the remaining eigenvalues λ's are all the eigenvalues of $V^{-1}(CA^{-1}B - D)$. This completes the proof of Theorem 1. □

Remark 1. System (1.1) becomes a generalized saddle problem when A and D are all SPD and $B = -C^T$. Let $SP(W)$ denote the spectrum of a matrix W, and suppose $SP(D) \in [\mu_1, \mu_n]$, $SP(-CA^{-1}B) \in [\sigma_1, \sigma_n]$, then, from Theorem 1, we can easily obtain that the remaining eigenvalues λ's of $P_{\mathrm{DE}}^{-1}H$ will be located in $\left[\frac{\sigma_1+\mu_1}{\alpha_2-1+\alpha_2\mu_n}, \frac{\sigma_n+\mu_n}{\alpha_2-1+\alpha_2\mu_1}\right]$ as $\alpha_2 > 1$. When D is an indefinite matrix, we can ensure the invertibility of V by using a positive and small enough α_2 (say, $0 < \alpha_2 < \frac{1}{1+\rho(D)}$), where $\rho(D)$ is the spectral radius of D. In this case, the remaining eigenvalues λ's of $P_{\mathrm{DE}}^{-1}H$ will be found in $\left[\frac{\lambda_{\min}(CA^{-1}B-D)}{1-\alpha_2-\alpha_2\rho(D)}, \frac{\lambda_{\max}(CA^{-1}B-D)}{1-\alpha_2-\alpha_2\rho(D)}\right]$.

Figures 1–3 depict the eigenvalue distributions of $P_{\mathrm{DE}}^{-1}H$, where H are, respectively, taken from Examples 1–3 in the following section. From these figures, we can see relatively clustered distribution of the eigenvalues of the preconditioned matrix $P_{\mathrm{DE}}^{-1}H$, and this can be especially noticed in Figure 2.

When using exact arithmetic in P_{DE}, the following theorem demonstrates that the iterative solution of linear system (2.2) preconditioned by P_{DE} lies in the $(n+1)$-dimensional space $\mathcal{K}(P_{\mathrm{DE}}^{-1}H, P_{\mathrm{DE}}^{-1}\boldsymbol{r})$, allowing for fast generalized minimal residual method (GMRES) convergence.

Theorem 2. *The degree of the minimal polynomial of preconditioned matrix $P_{\mathrm{DE}}^{-1}H$ is at most $n+1$, and thus, the dimension of the Krylov subspace $\mathcal{K}(P_{\mathrm{DE}}^{-1}H, \boldsymbol{b})$ is at most $n+1$ for any given vector $\boldsymbol{b}$.*

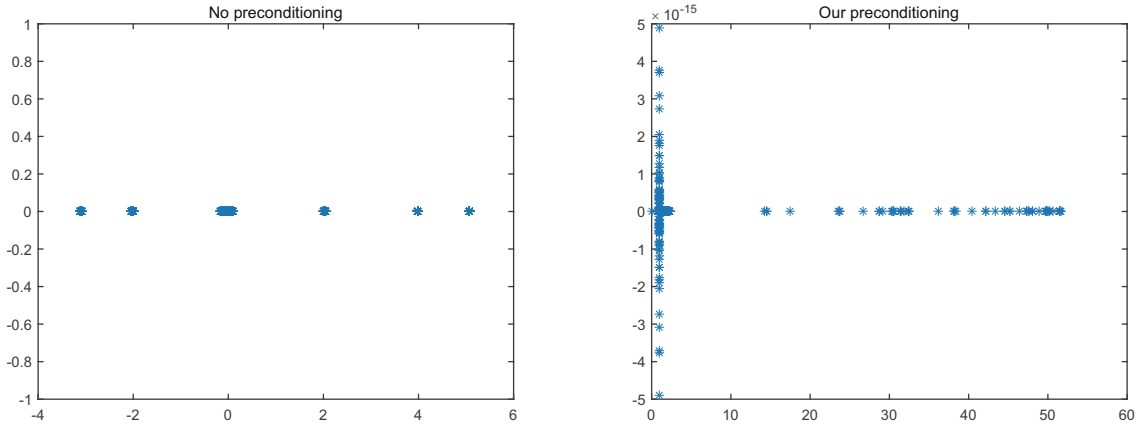

Figure 1: The eigenvalue distributions of the un-preconditioned/preconditioned matrix $P_{\mathrm{DE}}^{1}H$ in Example 1, $\alpha_2 = 1.3$, $h = \frac{1}{16}$, $\gamma = 0.01$, $\nu = 1$, left: no preconditioner; right: DE preconditioner.

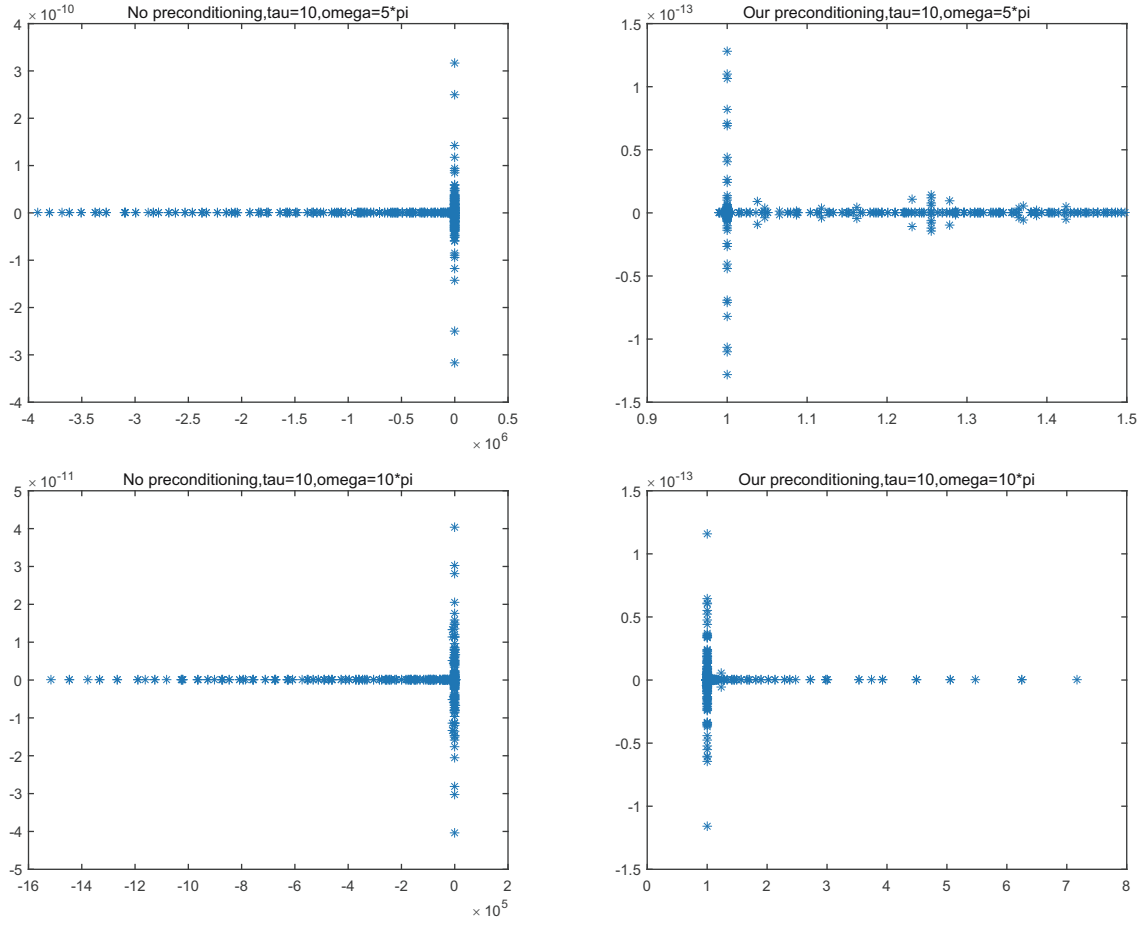

Figure 2: The eigenvalue distributions of the un-preconditioned/preconditioned matrix $P_{\mathrm{DE}}^{1}H$ in Example 2, $\alpha_2 = 1.01$, $h = \frac{1}{16}$, $\tau = 10$, upper left: no preconditioner, $\omega = 5\pi$; upper right: DE preconditioner, $\omega = 5\pi$; lower left: no preconditioner, $\omega = 10\pi$; lower right: DE preconditioner, $\omega = 10\pi$.

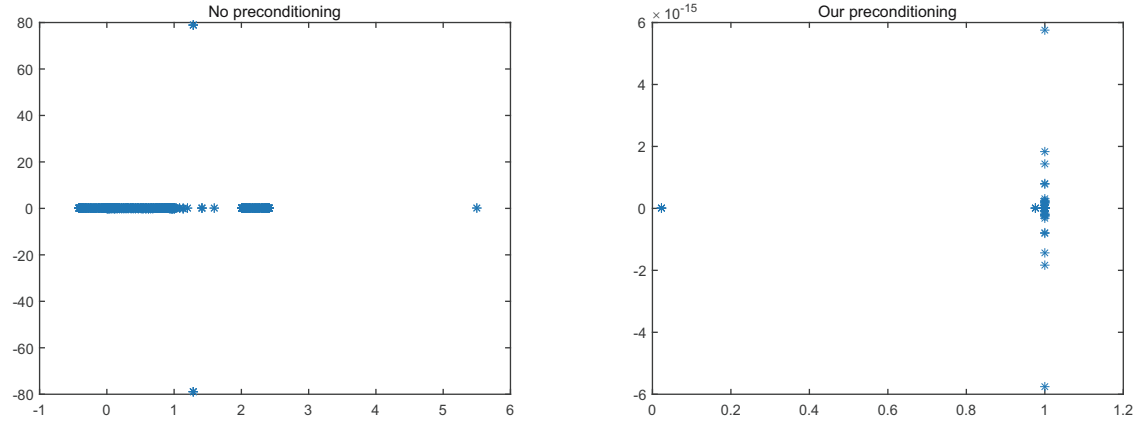

Figure 3: The eigenvalue distributions of the un-preconditioned/preconditioned matrix $P_{\mathrm{DE}}^{1}H$ in Example 3, $\alpha_2 = 1.00001$, H corresponds to "ck656." left: no preconditioner; right: DE preconditioner.

Proof. Because we know from (2.5) that the characteristic polynomial $f(\lambda)$ of $P_{\mathrm{DE}}^{-1}H$ is

$$f(\lambda) = \det(\lambda I_{(m+2n)} - P_{\mathrm{DE}}^{-1}H) = (\lambda - 1)^{m+n}\det[\lambda I_n - V^{-1}(CA^{-1}B - D)] = (\lambda - 1)^{m+n}(\lambda - \lambda_1^*)\cdots(\lambda - \lambda_n^*)$$

with λ_i^*, $i = 1, 2, \ldots, n$ the eigenvalues of $V^{-1}(CA^{-1}B - D)$, $f(\lambda) = 0$ has at most $n + 1$ distinct roots $\lambda_i = \lambda_i^*$, $i = 1, 2, \ldots, n$, $\lambda_{n+1} = 1$, and the degree of the corresponding minimal polynomial of the preconditioned matrix is at most $n + 1$. We can deduce from [33] that the dimension of the Krylov subspace $\mathcal{K}(P_{\mathrm{DE}}^{-1}H, \boldsymbol{b})$ is therefore at most $n + 1$. This completes the proof of Theorem 2. □

We provide Algorithm 1 for practically implementing P_{DE}^{-1}, which is obtained by considering the following factorization:

$$P_{\mathrm{DE}}^{-1} = P_4P_3P_2P_1 = \begin{bmatrix} -\alpha_2V^{-1} & & \\ & I & \\ & & -\alpha_2V^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{0} & -C & I \\ \mathbf{0} & I & \mathbf{0} \\ \dfrac{-V}{\alpha_2^2} & \dfrac{C}{\alpha_2} & \dfrac{-I}{\alpha_2} \end{bmatrix} \begin{bmatrix} I & & \\ & A^{-1} & \\ & & I \end{bmatrix} \begin{bmatrix} I & \mathbf{0} & \mathbf{0} \\ \dfrac{(\alpha_1 - 2)B}{-\alpha_2} & I & -B \\ \dfrac{-I}{\alpha_2} & \mathbf{0} & I \end{bmatrix}. \tag{2.6}$$

Algorithm 1. Preconditioning operation $P_{\mathrm{DE}}u = v$

Step 1. Compute an incomplete LU or a Cholesky factorization of reordered matrices A and V using techniques such as approximate minimum degree permutation (amd).

Step 2. Calculate $[\boldsymbol{u}_1^{(1)\top}, \boldsymbol{u}_2^{(1)\top}, \boldsymbol{u}_3^{(1)\top}]^\top = P_1[\boldsymbol{v}_1^\top, \boldsymbol{v}_2^\top, \boldsymbol{v}_3^\top]^\top$.

Step 3. Solve $A\boldsymbol{u}_2^* = \boldsymbol{u}_2^{(1)}$ for $\boldsymbol{u}_2^*$ using the factorization of A obtained in Step 1 as a preconditioner.

Step 4. Calculate $[\boldsymbol{u}_1^{(2)\top}, \boldsymbol{u}_2^{(2)\top}, \boldsymbol{u}_3^{(2)\top}]^\top = P_3[\boldsymbol{u}_1^{(1)\top}, \boldsymbol{u}_2^{*\top}, \boldsymbol{u}_3^{(1)\top}]^\top$.

Step 5. Solve $V\boldsymbol{u}_1^* = \boldsymbol{u}_1^{(2)}$, $V\boldsymbol{u}_3^* = \boldsymbol{u}_3^{(2)}$ for $\boldsymbol{u}_1^*, \boldsymbol{u}_3^*$ using the factorization of V obtained in Step 1 as a preconditioner, and calculate $[\boldsymbol{u}_1^\top, \boldsymbol{u}_2^\top, \boldsymbol{u}_3^\top]^\top = [-\alpha_2\boldsymbol{u}_1^{*\top}, \boldsymbol{u}_2^{(2)\top}, -\alpha_2\boldsymbol{u}_3^{*\top}]^\top$.

Remark 2. From (2.6), we can understand the relationship between α_1 and α_2 in Eqs. (2.2) is crucial. Evidently, without the equality $\alpha_1 = \frac{\alpha_2 - 2}{\alpha_2 - 1}$, we would be unable to obtain the factorization of P_{DE} and hence, P_{DE}^{-1} in (2.6).

Remark 3. We can see from Algorithm 1 that implementing P_{DE} can avoid solving linear subsystems like (1.4)–(1.5), which can significantly reduce the average cost of the entire iterative solution when using a Krylov iterative method.

3 Numerical results

In this section, we give three numerical examples to investigate the practical efficiency of the P_{DE} method (denoted by "DE") using Matlab R2016a on a PC equipped with an Intel(R)Core(TM)i5-8265U processor (CPU@1.60GHz). We compare our method to four existing techniques, denoted by the letters "SHSS" (taken from [11]), "BS" (taken from paper [5]), "PS" (taken from [27]), and "BPS" (taken from [31]).

3.1 Characteristics of the model problems

The matrices in Example 1 are obtained by discretizing the well-known Navier-Stokes equations employing the mixed finite element of the bilinear pressure $Q_1 - P_0$ pair with local stabilization. All matrices are generated by the IFISS software package provided by Davis and Hu [8]. When generating the required matrices, we run the driver "navier-testproblem" and select the problem "Lid driven cavity." The viscosity parameter is set to $\nu = 1, 0.01, 0.0001$, the mesh steps are set to $h = 1/16, 1/32, 1/64, 1/128$, and all other parameters are set automatically by selecting the "default" option. Because the obtained sub-matrix B is rank deficient, we replace B with $B(1:n, 1:n) = B(1:n, 1:n) + 0.0001I_{n\times n}$ to ensure that the resulting matrix $\begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is nonsingular. In

this example, the sub-matrix A is SPD, the sub-matrix D is SPSD, and $B = -C^T$. To compensate for the influence of the small viscosity ν, we scale the sub-matrices A and D into $\gamma A, D/\gamma$ while maintaining equivalence between the scaled and original systems.

Example 2 is the complex symmetric linear system (from [5])

$$[(K - (3 - \sqrt{3})\omega^2 I) + i(K + (3 + \sqrt{3})\tau^2 I)]x = b, \tag{3.1}$$

in which ω and τ are two positive parameters and K is the five-point centered difference matrix approximating the negative Laplacian operator with homogeneous Dirichlet boundary conditions, namely,

$$K = I_p \otimes V_p + V_p \otimes I_p, h = \frac{1}{p+1}, V_p = h^{-2}\text{tridiag}(-1, 2, -1) \in \mathbb{R}^{p\times p}.$$

If we denote $W = K - (3 - \sqrt{3})\omega^2 I$ and $T = K + (3 + \sqrt{3})\tau^2 I$, then (3.1) is easily transformed into (1.3).

The matrices in Example 3 are taken from the University of Florida Sparse Matrix Collection (URL: https://sparse.tamu.edu/, see [7,8]), and they all represent the case of $B \neq \pm C^T$ and A, D are all nonsymmetric. The details of these tested matrices are shown in Table 1.

3.2 Experimental design

In all of these examples, the exact solution x_{exa} is $x = \text{ones}(n + m, 1)$. All of the subsystems $M\boldsymbol{u} = \boldsymbol{v}$ involved in our numerical experiments are inexactly solved by GMRES or PCG; concretely, we first use the Matlab code "symamd" to obtain an approximate minimum degree permutation for M. Then, if M is not SPD, we compute an incomplete LU factorization with "droptol=1e-5" for M. Alternatively, if M is SPD, we compute an incomplete Cholesky factorization with "droptol=1e-5." Then, we solve $M\boldsymbol{u} = \boldsymbol{v}$ using either the GMRES method preconditioned by the triangular factors L and U (when M is not SPD) or the PCG method preconditioned by the triangular factors L, L^T (when M is SPD). The convergence tolerance to terminate the iterations is set equal to "tol=1e-5." We use the flexible GMRES (FGMRES) for Example 1 and GMRES for Examples 2 and 3 as the outer iteration, with a convergence tolerance of 1×10^{-8} and a maximum number of iterations of 1,000. Both relative errors between the exact and numerical solutions of these two iterative methods are reported.

In the following tables, we denote by "`iter`" the number of iterations required by the FGMRES and GMRES methods, and by "`time`" the elapsed CPU time in seconds, which includes the time for computing the "ilu"/"ichol" factorization as well as the time for iteratively solving the global preconditioned linear system. The symbol "*" indicates that the FGMRES or GMRES method fails to converge. Furthermore, we report on the average cost of the five methods, denoted by "`aver`" in the tables and calculated as "time/iter."

3.3 Analyses of the numerical results

The results presented in Tables 2–8 show that the proposed method is feasible.

Table 2: Numerical results of Example 1 ($\nu = 1, \gamma = 0.001$)

	DE ($\alpha_2 = 1.3$)				SHSS				BPS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/16	60	2.2×10^{-7}	0.12	0.00	15	3.5×10^{-6}	0.02	0.00	39	3.0×10^{-1}	1.61	0.04
1/32	50	1.7×10^{-6}	0.19	0.00	26	2.2×10^{-6}	0.12	0.00	54	3.0×10^{-1}	4.44	0.08
1/64	33	6.3×10^{-7}	0.38	0.01	41	5.9×10^{-6}	0.95	0.02	69	3.2×10^{-1}	28.59	0.41
1/128	30	4.9×10^{-7}	1.19	0.04	66	3.4×10^{-6}	8.63	0.13	143	5.1×10^{-2}	148.54	1.03

Table 3: Numerical results of Example 1 ($\nu = 0.01, \gamma = 0.2$)

	DE ($\alpha_2 = 1.3$)				SHSS				BPS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/16	55	1.0×10^{-6}	0.11	0.00	19	6.5×10^{-8}	0.04	0.00	29	1.1×10^{-6}	0.11	0.00
1/32	40	7.8×10^{-6}	0.15	0.00	32	7.9×10^{-8}	0.12	0.00	49	2.1×10^{-6}	0.80	0.02
1/64	30	3.2×10^{-6}	0.36	0.01	53	6.8×10^{-8}	1.12	0.02	87	5.1×10^{-8}	29.45	0.33
1/128	31	7.7×10^{-7}	1.24	0.04	87	5.1×10^{-8}	10.40	0.12	126	1.4×10^{-7}	72.81	0.57

Table 4: Numerical results of Example 1 ($\nu = 0.0001, \gamma = 20$)

	DE ($\alpha_2 = 1.3$)				SHSS				BPS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/16	55	1.0×10^{-6}	0.12	0.00	19	3.1×10^{-7}	0.03	0.00	210	2.0×10^{-6}	2.35	0.01
1/32	40	7.8×10^{-6}	0.15	0.00	32	1.7×10^{-7}	0.12	0.00	576	2.8×10^{-6}	16.69	0.02
1/64	30	3.2×10^{-6}	0.36	0.01	54	1.3×10^{-7}	1.12	0.02	636	4.2×10^{-6}	52.15	0.08
1/128	31	8.0×10^{-7}	1.30	0.04	90	2.4×10^{-8}	11.73	0.13	*	*	*	*

Table 5: Numerical results of Example 2 ($\omega = 5\pi, \tau = 10$)

	DE ($\alpha_2 = 1.01$)				BS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/32	10	2.5×10^{-9}	0.04	0.00	22	1.5×10^{-8}	0.06	0.00
1/64	11	1.7×10^{-9}	0.12	0.01	43	8.4×10^{-8}	0.51	0.01
1/128	11	3.4×10^{-9}	0.42	0.03	87	5.6×10^{-7}	3.35	0.03
1/256	11	4.2×10^{-9}	1.64	0.14	189	3.3×10^{-6}	34.74	0.18

Table 6: Numerical results of Example 2 ($\omega = 10\pi, \tau = 10$)

	DE ($\alpha_2 = 1.01$)				BS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/32	12	3.2×10^{-8}	0.05	0.00	13	4.3×10^{-9}	0.04	0.00
1/64	13	2.4×10^{-8}	0.12	0.01	28	6.9×10^{-8}	0.28	0.01
1/128	14	2.3×10^{-8}	0.52	0.03	54	3.3×10^{-7}	2.40	0.04
1/256	14	3.0×10^{-8}	2.51	0.17	112	1.9×10^{-6}	28.59	0.25

- When the dimension of H is large enough, "DE" is expected to outperform the other four methods in terms of iterations. This is reflected in Example 1 (Tables 2–4) and Example 2 (Tables 5–7).
- When looking at the increasing rate of iterations, "DE" is relatively stable when compared to "SHSS," "BS," "PS," and "BPS." This is demonstrated in Examples 1 (Tables 2–4), 2 (Tables 5–7), and 3 (Table 8).
- From the perspective of relative errors, "DE" is relative stable and accurate, similar to "SHSS," "BS," and "PS."
- In the case of high dimensional H, "DE" requires less CPU time than "SHSS," "BS," "PS," and "BPS." This is particularly evident in Examples 1 (Tables 2–4) and 2 (Tables 5–7).
- "DE" is feasible and efficient for all $B = \pm C^T$ (Example 1), $B = C$ (Example 2), and $B \neq \pm C^T$ (Example 3) cases.

Table 7: Numerical results of Example 2 ($\omega = 15\pi$, $\tau = 10$)

	DE ($\alpha_2 = 1.01$)				BS			
h	iter	errors	time (s)	aver	iter	errors	time (s)	aver
1/32	26	1.0×10^{-7}	0.09	0.00	10	4.2×10^{-9}	0.03	0.00
1/64	26	1.2×10^{-7}	0.20	0.01	18	6.7×10^{-8}	0.21	0.01
1/128	29	1.2×10^{-7}	1.03	0.03	37	3.5×10^{-7}	1.40	0.03
1/256	29	1.5×10^{-7}	4.43	0.15	74	2.4×10^{-6}	15.01	0.20

Table 8: Numerical results of Example 3

	DE ($\alpha_2 = 1.00001$)				PS			
Matrices	iter	errors	time (s)	aver	iter	errors	time (s)	aver
dw256A	11	6.8×10^{-8}	0.03	0.00	9	1.4×10^{-6}	0.03	0.00
ck656	7	2.6×10^{-6}	0.03	0.00	10	5.2×10^{-5}	0.05	0.00
poli	6	2.8×10^{-9}	0.05	0.00	3	1.0×10^{-11}	0.04	0.01
poli-large	6	4.9×10^{-6}	0.55	0.09	9	6.1×10^{-6}	0.68	0.07
venkat25	64	9.9×10^{-7}	19.55	0.30	58	1.7×10^{-6}	19.87	0.34
poisson3db	108	7.5×10^{-6}	32.17	0.29	*	*	*	*

4 Conclusion

In this article, we constructed a preconditioner for a general block two-by-two linear system (1.1) by expanding the dimension of the coefficient matrix. Theoretical analyses show that the presented preconditioner can result in a relatively well clustered distribution of the eigenvalues of the preconditioned matrix. For the inner subproblems, a concrete implementation scheme and an inexact solver are provided. To investigate the efficiency of the proposed preconditioner, numerical examples are provided. Theoretical results and numerical examples demonstrate that the preconditioner presented here is applicable for both $B = \pm C^T$ and $B \neq \pm C^T$.

Funding information: This research was supported by the Scientific Research Fund of Hunan Provincial Science and Technology Department (2022JJ30416) and the Scientific Research Funds of Hunan Provincial Education Department (20A345 and 22A0483). The second author is a member of Gruppo Nazionale per il Calcolo Scientifico (GNCS) of Istituto Nazionale di Alta Matematica (INdAM), and this work was partially supported by INdAM-GNCS under Progetti di Ricerca 2023. The work of Jun Guo was supported by the Sichuan National Applied Mathematics co-construction project (2022ZX004), CUIT (KYTD202243), and the Scientific Research Foundation (KYTZ202184).

Conflict of interest: The authors declare no potential conflict of interest.

References

[1] V. Girault and P. A. Raviart, *Finite Element Methods for Navier-Stokes Equations: Theory and Algorithms*, Springer Science & Business Media, NewYork, 2012.

[2] Z. Chen, *Finite Element Methods and Their Applications*, Springer, Berlin, 2005.

[3] G. Bao and W. Sun, *A fast algorithm for the electromagnetic scattering from a large cavity*, SIAM J. Sci. Comput. **27** (2005), 553–574.

[4] W. Van Dijk and F. M. Toyama, *Accurate numerical solutions of the time-dependent Schrödinger equation*, Phys. Rev. E. **75** (2007), 036707, 1–10.

[5] J.-H. Zhang and H. Dai, *A new block preconditioner for complex symmetric indefinite linear systems*, Numer. Algorithms **74** (2017), 889–903.
[6] Y. Cao and Z.-R. Ren, *Two variants of the PMHSS iteration method for a class of complex symmetric indefinite linear systems*, Appl. Math. Comput. **264** (2015), 61–71.
[7] K. Chen, *Matrix Preconditioning Techniques and Applications*, Cambridge University Press, Cambridge, 2005.
[8] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software **38** (2011), no. 1, 1–25.
[9] M. Benzi and G. H. Golub, *A preconditioner for generalized saddle point problems*, SIAM J. Matrix. Anal. Appl. **26** (2004), no. 1, 20–41.
[10] 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), no. 3, 603–626.
[11] Y. Cao, Z.-R. Ren, and Q. Shi, *A simplified HSS preconditioner for generalized saddle point problems*, BIT Numer. Math. **56** (2016), 423–439.
[12] Y. Cao, L.-Q. Yao, and M.-Q. Jiang, *A modified dimensional split preconditioner for generalized saddle point problems*, J. Comput. Appl. Math. **250** (2013), 70–82.
[13] Y. Cao, S.-X. Miao, and Y.-S. Cui, *A relaxed splitting preconditioner for generalized saddle point problems*, Comput. Appl. Math. **34** (2015), 865–879.
[14] M. Benzi and X.-P. Guo, *A dimensional split preconditioner for Stokes and linearized Navier-Stokes equations*, Appl. Numer. Math. **61** (2011), no. 1, 66–76.
[15] M. Benzi, M. Ng, N. Qiang, and W. Zhen, *A relaxed dimensional factorization preconditioner for the incompressible Navier-Stokes equations*, J. Comput. Phys. **230** (2011), 6185–6202.
[16] J.-L. Zhang, *An efficient variant of HSS preconditioner for generalized saddle point problems*, Numer. Linear Algebra Appl. **25** (2018), no. 4, 1–14.
[17] Z.-Z. Bai, M. K. Ng, and Z.-Q. Wang, *Constraint preconditioners for symmetric indefinite matrices*, SIAM J. Matrix. Anal. Appl. **31** (2009), no. 2, 410–433.
[18] G.-F. Zhang, Z.-R. Ren, and Y.-Y. Zhou, *On HSS-based constraint preconditioners for generalized saddle point problems*, Numer. Algorithms, **57** (2011), 273–287.
[19] Y. Cao, M.-Q. Jiang, and Y.-L. Zheng, *A splitting preconditioner for saddle point problems*, Numer. Linear Algebra Appl. **18** (2011), no. 5, 875–895.
[20] I. N. Konshin, M. A. Olshanskii, and Y. V. Vassilevski, *ILU preconditioners for nonsymmetric saddle-point matrices with application to the incompressible Navier-Stokes equations*, SIAM J. Sci. Comput. **37** (2015), no. 5, A2171–A2197.
[21] J.-H. Zhang and H. Dai, *A new splitting preconditioner for the iterative solution of complex symmetric indefinite linear systems*, Appl. Math. Lett. **49** (2015), 100–106.
[22] Q.-Q. Shen and Q. Shi, *A variant of the HSS preconditioner for complex symmetric indefinite linear systems*, Comput. Math. Appl. **75** (2018), no. 3, 850–863.
[23] M. Frigo, N. Castelletto, and M. Ferronato, *A relaxed physical factorization preconditioner for mixed finite element coupled poromechanics*, SIAM J. Sci. Comput. **41** (2019), no. 4, B694–B720.
[24] M. J. Gander, Q. Niu, and Y. Xu, *Analysis of a new dimension-wise splitting iteration with selective relaxation for saddle point problems*, BIT Numer. Math. **56** (2016), no. 2, 441–465.
[25] Y. Cao, J.-L. Dong, and Y.-M. Wang, *A relaxed deteriorated PSS preconditioner for nonsymmetric saddle point problems from the steady Navier-Stokes equation*, J. Comput. Appl. Math. **273** (2015), 41–60.
[26] Z.-Z. Bai, *Structured preconditioners for nonsingular matrices of block two-by-two structures*, Math. Comput. **75** (2006), 791–815.
[27] W. Chao, T.-Z. Huang, and C. Wen, *A new preconditioner for indefinite and asymmetric matrices*, Appl. Math. Comput. **219** (2013), 11036–11043.
[28] H. Chen, X. Li, and Y. Wang, *A splitting preconditioner for a block two-by-two linear system with applications to the bidomain equations*, J. Comput. Appl. Math. **321** (2017), 487–498.
[29] Q. Zheng and L. Lu, *A shift-splitting preconditioner for a class of block two-by-two linear systems*, Appl. Math. Lett. **66** (2016), no. 3, 54–60.
[30] M. Masoudi and D. K. Salkuyeh, *An extension of the positive-definite and skew-Hermitian splitting method for preconditioning of generalized saddle point problems*, Comput. Math. Appl. **79** (2020), no. 8, 2304–2321.
[31] Y. Cao, *A block positive-semidefinite splitting preconditioner for generalized saddle point linear systems*, J. Comput. Appl. Math. **374** (2020), 112787.
[32] W.-H. Luo, X.-M. Gu, and B. Carpentieri, *A dimension expanded preconditioning technique for saddle point problems*, BIT Numer. Math. **62** (2022), no. 4, 1983–2004.
[33] Y. Saad, *Iterative Methods for Sparse Linear Systems*, SIAM, Philadelphia, PA, 2003.