

Improved New Block Preconditioner for Solving 3×3 Block Saddle Point Problems

Xin-Hui Shao * and Xin-Yang Liu

Department of Mathematics, College of Sciences, Northeastern University, Shenyang 110819, China; liuxy72@mails.neu.edu.cn

* Correspondence: shaoxinhui@mail.neu.edu.cn

Abstract

In order to overcome the computational challenges associated with block preconditioners for Krylov subspace methods, particularly those arising from Schur complement systems, this paper proposes an improved new block (INB) preconditioner for solving 3×3 block saddle point problems. A detailed semi-convergence analysis of the iterative scheme induced by the INB preconditioner is provided. Moreover, the spectral properties of the preconditioned matrix are analyzed, revealing strong eigenvalue clustering around one. Efficient formulas for selecting quasi-optimal parameters are derived based on Frobenius-norm minimization. Extensive numerical experiments demonstrate that the proposed INB preconditioner significantly reduces iteration counts and CPU time compared with several existing block preconditioners.

Keywords: three-by-three block saddle point problem; preconditioning; Krylov subspace methods; convergence; spectral analysis

MSC: 65F08; 65F10

1. Introduction

Consider an iterative approach to resolve the 3×3 block saddle point problem:

$$\mathcal{A}\mathbf{u} = \begin{pmatrix} A & B^T & 0 \\ B & 0 & C^T \\ 0 & C & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f \\ g \\ h \end{pmatrix} \equiv \mathbf{b}, \quad (1)$$

where, the matrix $A \in \mathbb{R}^{n \times n}$ is Symmetric Positive Definite (SPD), $B \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{l \times m}$ are the full row rank matrices. $f \in \mathbb{R}^n$, $g \in \mathbb{R}^m$ and $h \in \mathbb{R}^l$ are given known vectors, and $\mathbf{u} = (x, y, z)^T$ is the unknown vector. Many practical applications result in linear systems as shown in a linear system (1). For instance, the Picard iterative method is employed to address a class of stationary magnetohydrodynamic models using a hybrid finite element framework[1]. Furthermore, the finite element method is employed to address the time-dependent Maxwell equation with a discontinuous coefficient within a polyhedral region with a Lipschitz boundary. References [2,3] offer further examples and related literature. The numerical solution of such saddle point problems has been a central topic in numerical linear algebra, with a comprehensive theoretical

Academic Editor: István Faragó

Received: 25 December 2025

Revised: 6 February 2026

Accepted: 22 February 2026

Published: 27 February 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\) license](https://creativecommons.org/licenses/by/4.0/).

framework established in the seminal review work [4]. Consequently, it is pertinent to investigate the problem of double saddle points.

A linear system (1) has a symmetric coefficient matrix, which may reduce the effectiveness of traditional iterative solution techniques. Consequently, system (1) is often reformulated equivalently as the nonsymmetric but practically advantageous system:

$$\hat{\mathcal{A}}\mathbf{u} = \begin{pmatrix} A & B^T & 0 \\ -B & 0 & -C^T \\ 0 & C & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} f \\ -g \\ h \end{pmatrix} \equiv \hat{\mathbf{b}}, \quad (2)$$

Although the coefficient matrix in (2) is nonsymmetric, it possesses favorable structural properties. In particular, the leading block A remains SPD, and the overall matrix is symmetric positive semidefinite in the sense of the underlying bilinear form, which promotes good convergence behavior [5]. This reformulation is advantageous for Krylov subspace methods such as GMRES and FGMRES, which often exhibit improved convergence behavior when applied to such systems.

Although the linear system (2) is represented by a 3×3 block matrix structure, an alternative representation can be obtained by viewing it as a 2×2 block matrix, as illustrated in the following equation:

$$\begin{pmatrix} A & B^T & 0 \\ -B & 0 & -C^T \\ 0 & C & 0 \end{pmatrix} \text{ or } \begin{pmatrix} A & B^T & 0 \\ -B & 0 & -C^T \\ 0 & C & 0 \end{pmatrix}. \quad (3)$$

Many preconditioning schemes applicable to solving Equation (3) exhibit limitations due to the differing underlying characteristics of Equation (3) compared to a linear system (1). Consequently, it is necessary to construct novel preconditioners tailored to the specific structure of matrix A for the 3×3 block saddle-point problem.

$$\mathcal{P}_{BD} = \begin{pmatrix} A & 0 & 0 \\ 0 & S & 0 \\ 0 & 0 & CS^{-1}C^T \end{pmatrix},$$

Huang and Ma [6] proposed a block diagonal (BD) preconditioner, where $S = BA^{-1}B^T$. Given the computational complexity involved in solving the residual equations for the exact variant of this preconditioner, researchers subsequently introduced an inexact variant. Cao [7] later constructed a shift-splitting (SS) iteration method; deriving both the SS and the relaxed SS (RSS) preconditioners, the former taking the form:

$$\mathcal{P}_{SS} = \frac{1}{2} \begin{pmatrix} \alpha I + A & B^T & 0 \\ -B & \alpha I & -C^T \\ 0 & C & \alpha I \end{pmatrix}.$$

Reference [8] proposes a Generalized Alternating Splitting (GSS) preconditioner based on the SS preconditioner. Further extensions of generalized shift-splitting preconditioners for general saddle point systems were systematically studied in [9,10], which provide important methodological references for the construction of splitting-based preconditioners. Zhang et al. [11] construct an Unbalanced Alternating Splitting preconditioner for the saddle point problem (2). To enhance the preconditioning performance of the BD preconditioner, Xie et al. [12] investigate three efficient preconditioner structures. Aslani et al. [13] present a novel matrix splitting method and derive a block preconditioner

for solving the linear system (2). Salkuyeh et al. [14] develop an Alternating Positive Semi-definite Splitting (APSS) iterative method and propose the APSS preconditioner for problem (2).

$$\mathcal{P}_{APSS} = \begin{pmatrix} \alpha I + A & B^T & 0 \\ -B & \alpha I & 0 \\ 0 & 0 & \alpha I \end{pmatrix} \begin{pmatrix} \alpha I & 0 & 0 \\ 0 & \alpha I & -C^T \\ 0 & C & \alpha I \end{pmatrix}.$$

Furthermore, the APSS method has also been applied to solve large-scale sparse and singular 3×3 saddle point problems [15]. On this basis, Xiong and Li [16] further proposed a simplified relaxed alternating positive semi-definite splitting preconditioner for 3×3 block saddle point problems, which reduces the computational complexity of the APSS-type preconditioners while maintaining favorable convergence properties. Abdolmaleki et al. [17] proposed a block 3×3 diagonal preconditioner for the 3×3 block saddle point problem (2) from another perspective. Wang et al. [18] established an exact parameterized block SPD (Symmetric Positive Definite) preconditioner and considered its inexact form for the block 3×3 saddle point problem (2). For the broader class of nonsymmetric generalized saddle point problems, Li [19] and Zheng and Lu [20] respectively proposed extended shift-splitting iteration methods and corresponding preconditioners, which enrich the theoretical system of splitting-based preconditioning techniques for saddle point problems.

The following is a description of the structure of this article. Section 2 discusses the convergence of the iterative scheme produced by an enhanced block preconditioner, which is established first. A spectrum analysis of the preconditioning matrix is then shown in Section 3. Subsequently, Section 4 elucidates the selection of parameters essential for the preconditioner. Section 5 presents a series of numerical experiments, the objective of which is to illustrate the efficacy of the proposed preconditioner. The paper concludes in Section 6.

2. Improvement of a New Block Preconditioner

2.1. Construction of the INB Preconditioner

An efficient iterative solution to the double saddle point problem (2) is initially presented in this section. Based on the matrix splitting concept put forth in reference [21], we consider the following splitting of the coefficient matrix from (2):

$$A = \mathcal{P}_{INB} - \mathcal{R}_{INB} \quad (4)$$

where

$$\mathcal{P}_{INB} = \begin{pmatrix} A & B^T & 0 \\ 0 & \alpha I & -C^T \\ 0 & C & 0 \end{pmatrix} \text{ and } \mathcal{R}_{INB} = \begin{pmatrix} 0 & 0 & 0 \\ B & \alpha I & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The identity matrix is denoted by I in the equation. The nonsingularity of the matrix $\mathcal{P}_{INB}$ can be easily confirmed [5]. The following is the iterative strategy linked to the splitting (4):

$$\mathbf{u}^{(k+1)} = \mathcal{M} \mathbf{u}^{(k)} + \mathbf{g}, \quad k = 0, 1, 2, 3,$$

where $\mathbf{u}^{(0)}$ is the initial value, $\mathcal{M} = \mathcal{P}_{INB}^{-1} \mathcal{R}_{INB}$ is the iteration matrix, and $\mathbf{g} = \mathcal{P}_{INB}^{-1} \mathbf{b}$ is defined as follows:

We then show the implementation of the preconditioner $\mathcal{P}_{INB}$ when the parameters, $\alpha > 0$, are provided. The following type of linear system must be solved to apply the Krylov subspace method with the preconditioner $\mathcal{P}_{INB}$:

$$\begin{pmatrix} A & B^T & 0 \\ 0 & \alpha I & -C^T \\ 0 & C & 0 \end{pmatrix} \begin{pmatrix} s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix}$$

Consequently, we can derive Algorithm 1 below as a summary of how this preconditioner is implemented.

Algorithm 1. Computation of $s = \mathcal{P}_{INB}^{-1}r$

1. Solve $CC^T s_3 = \alpha r_3 - Cr_2$;
 2. Solve $s_2 = \alpha^{-1}(r_2 + C^T s_3)$;
 3. Solve $As_1 = r_1 - B^T s_2$.
-

Each iteration of the aforementioned algorithm requires solving linear systems with coefficient matrices CC^T and A . They are easier to solve than the Schur complement matrix $BA^{-1}B^T$ of the linear subsystem. It is important to note that for all $\alpha > 0$, both CC^T and A are symmetric positive definite, allowing us to utilize sparse Cholesky decomposition to solve them. Therefore, the per-iteration computational complexity is dominated by sparse triangular solves and sparse matrix vector products. Compared with block preconditioners that rely on approximate Schur complement inversions, the INB preconditioner achieves comparable per-iteration cost while significantly reducing the total number of Krylov iterations.

2.2. Semi-Convergence Analysis

Next, we will conduct a semi-convergence analysis. First, we present a useful lemma.

Lemma 1. ([12] Theorem 2) Let E and F be two $n \times n$ real symmetric matrices, where E is positive definite and F is positive semidefinite. Then $E \succeq F$ holds if and only if $\rho(E^{-1}F) \leq 1$, and $E \succ F$ holds if and only if $\rho(E^{-1}F) < 1$.

Theorem 1. Let $A \in \mathbb{R}^{n \times n}$ be symmetric positive definite (SPD), and let $B \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{l \times m}$ be row full-rank matrices. Let α be a positive constant. If $2\alpha I \succ BA^{-1}B^T$, then the iterative method converges to the unique solution of system (2) for any initial guess.

Proof. Let λ represent the eigenvalue of the iteration matrix $\mathcal{M} = \mathcal{P}_{INB}^{-1}\mathcal{R}_{INB}$, $(u, v, w)^T$ denoting the corresponding eigenvector. When $\lambda = 0$ is given, the iterative method will necessarily converge. Therefore, we only consider the case where $\lambda \neq 0$ corresponds to $(u, v, w)^T \neq (0, 0, 0)^T$. Based on the relationship between eigenvalues and eigenvectors, we have:

$$\begin{pmatrix} 0 & 0 & 0 \\ B & \alpha I & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix} = \lambda \begin{pmatrix} A & B^T & 0 \\ 0 & \alpha I & -C^T \\ 0 & C & 0 \end{pmatrix} \begin{pmatrix} u \\ v \\ w \end{pmatrix}$$

The above expression can be equivalently transformed into:

$$\begin{cases} \lambda(Au + B^T v) = 0, & (5a) \\ \alpha(1-\lambda)Iv + \lambda C^T w + Bu = 0, & (5b) \\ \lambda Cv = 0. & (5c) \end{cases} \quad (5)$$

if $\lambda = 1$,

$$\begin{cases} Au + B^T v = 0, & (6a) \\ C^T w - Bu = 0, & (6b) \\ Cv = 0. & (6c) \end{cases} \quad (6)$$

Note that both u and v are not zero vectors; otherwise, $(u, v, w)^T = (0, 0, 0)^T$ would lead to a contradiction. By multiplying Equation (6b) by v^* and combining it with Equations (6a) and (6c), we obtain $u^* Au = 0$, which implies $u = 0$, and consequently leads to $v = 0$ and $w = 0$, resulting in a contradiction. Thus, $\lambda \neq 1$ must hold.

Next, we consider the case of $\lambda \neq 1$. Based on this assumption, it is easy to see that both u and v are non-zero vectors. From Equation (5c), we know that:

$$u = -AB^T v, v^* C^T = 0.$$

Left-multiply both sides of Equation (5b) by $\frac{v^*}{v^* v}$, where v^* is the conjugate transpose of v :

$$\lambda = 1 - \frac{v^* BA^{-1} B^T v}{v^* \alpha Iv}$$

By Theorem 1, it directly follows that $\lambda_{\max}(\alpha BA^{-1} B^T) = \rho(\alpha BA^{-1} B^T) < 2$ if and only if $2\alpha I \succ BA^{-1} B^T$, hence we can derive:

$$0 < \frac{v^* BA^{-1} B^T v}{v^* \alpha Iv} \leq \lambda_{\max}(\alpha BA^{-1} B^T) < 2,$$

Then,

$$\left| 1 - \frac{v^* BA^{-1} B^T v}{v^* \alpha Iv} \right| < 1.$$

□

Remark 1. The full row-rank assumption on B and C guarantees that the subsystems involving CC^T and the Schur complements are nonsingular, which is essential for the well-posedness of the preconditioning steps. In practice, if B or C is rank-deficient, a regularization (e.g., a small shift) can be applied[22,23].

3. Spectral Analysis of the Preconditioner Matrix

It follows from splitting (4) that $\mathcal{P}_{INB}^{-1}\mathcal{A} = I - \mathcal{P}_{INB}^{-1}\mathcal{R} = I - \mathcal{M}$.

Theorem 2. Suppose the matrix $A \in \mathbb{R}^{n \times n}$ is SPD, the matrix $B \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{l \times m}$ are full row rank. Let α be positive constants, $(\hat{u}, \hat{v}, \hat{w})^T$ be the eigenvector corresponding to the eigenvalue, θ is the eigenvalue of $\mathcal{P}_{INB}^{-1}\mathcal{A}$, and $\hat{v}^*$ is the conjugate transpose of $\hat{v}$. If $\theta = 1$ is an eigenvalue whose algebraic multiplicity is at least $n + l$, the corresponding eigenvectors take the form $\left(\hat{u}; -\frac{1}{\alpha}B\hat{u}; \hat{w}\right)$. The remaining eigenvalues are given by $\theta = \frac{\hat{v}^*BA^{-1}B^T\hat{v}}{\alpha\hat{v}^*\hat{v}}$, and for any nonzero vector $\hat{v} \in \ker(C) \subset \mathbb{R}^m$, their eigenvectors are $\left(-A^{-1}B^T\hat{v}; \hat{v}; \hat{w}\right)$, where $\hat{w} \in \mathbb{R}^l$ denotes any vector.

Proof. Let θ

represent the eigenvalues of the iterative matrix $\mathcal{P}_{INB}^{-1}\mathcal{A}$, with $(\hat{u}, \hat{v}, \hat{w})^T$ denoting the eigenvector corresponding to that eigenvalue:

$$\begin{pmatrix} 0 & 0 & 0 \\ B & \alpha I & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \hat{u} \\ \hat{v} \\ \hat{w} \end{pmatrix} = \lambda \begin{pmatrix} A & B^T & 0 \\ 0 & \alpha I & -C^T \\ 0 & C & 0 \end{pmatrix} \begin{pmatrix} \hat{u} \\ \hat{v} \\ \hat{w} \end{pmatrix}.$$

Therefore, we have

$$\begin{cases} (\theta - 1)(A\hat{u} + B^T\hat{v}) = 0, & (7a) \\ \alpha\theta\hat{v} - (\theta - 1)C^T\hat{w} + B\hat{u} = 0, & (7b) \\ (\theta - 1)C\hat{v} = 0. & (7c) \end{cases} \quad (7)$$

When $\theta = 1$, we have $\alpha\hat{u} + B^T\hat{v} = 0$. Clearly, $\hat{v} = -\frac{1}{\alpha}B\hat{u}$. Therefore, the linearly independent eigenvectors corresponding to the eigenvalue 1 take the form $\left(\hat{u}; -\frac{1}{\alpha}B\hat{u}; \hat{w}\right)$, and there are $n + l$ such vectors, where $\hat{u}$ and $\hat{w}$ are arbitrary linearly independent vectors.

If $\theta \neq 1$ holds, it is straightforward to observe that $\hat{v}$ cannot be the zero vector, and we can therefore obtain:

$$\begin{cases} \hat{u} = -A^{-1}B^T\hat{v}, & (8a) \\ \alpha\theta\hat{v} + (\theta - 1)C^T\hat{w} - BA^{-1}B^T\hat{v} = 0, & (8b) \\ C\hat{v} = 0. & (8c) \end{cases} \quad (8)$$

Left-multiply Equation (8b) by $\frac{\hat{v}^*}{\hat{v}^* \hat{v}}$, where $\hat{v}^*$ denotes the conjugate transpose of $\hat{v}$, yielding $\theta = \frac{\hat{v}^* B A^{-1} B^T \hat{v}}{\alpha \hat{v}^* \hat{v}}$. Since C is row full-rank, for any right-hand side, there exists $\hat{v} \in \mathbb{R}^l$ satisfying Equation (8b). Thus, the eigenvector is $(-A^{-1} B^T \hat{v}; \hat{v}; \hat{w})$. $\square$

Remark 2. By Theorem 3., the non-unit eigenvalues of the preconditioned matrix $\mathcal{P}_{INB}^{-1} \mathcal{A}$ satisfy:

$$0 < \frac{\lambda_{\min}(B A^{-1} B^T)}{\alpha} \leq \lambda \leq \frac{\lambda_{\max}(B A^{-1} B^T)}{\alpha}.$$

Specifically, when α satisfies $\alpha = \lambda_{\max}(B A^{-1} B)$, all non-unit eigenvalues $\lambda \in [\kappa^{-1}, 1]$, where $\kappa = \kappa(B A^{-1} B^T) = \frac{\lambda_{\max}(B A^{-1} B^T)}{\lambda_{\min}(B A^{-1} B^T)}$. The above expression clearly demonstrates that the non-unit eigenvalues depend continuously on the parameter α . By selecting an appropriate α , eigenvalues can be clustered around a desired value, leveraging spectral concentration to enhance iterative method convergence.

4. Parameter Selection

The choice of parameters α has a significant impact on the preconditioner's efficiency. The techniques used to choose the preconditioner's parameters α will be covered and examined in this section.

The selection of parameters is a rather intricate process for the majority of preconditioners, including SS-type preconditioners [6,24,25]. Therefore, it often requires extensive experimentation to choose the optimal parameters. Fortunately, by adopting certain approximation strategies, effective formulas for approximately optimal parameters can be derived. This paper offers a workable approach to choosing the parameters in the preconditioner under discussion, drawing inspiration from Chen's work [26].

First, we establish a function $\Gamma(\alpha) = \|\mathcal{R}_{INB}\|_F^2$ that depends on α . Then, we minimize $\Gamma(\alpha)$ to choose these parameters, where $\mathcal{R}_{INB}$ is provided by Equation (4) and $\|\cdot\|_F$ stands for the Frobenius norm.

From direct calculations, we obtain:

$$\Gamma(\alpha) = \|\mathcal{R}_{INB}\|_F^2 = \text{tr}(\mathcal{R}_{INB} \mathcal{R}_{INB}^T) = \alpha^2 m + \text{tr}(B^T B)$$

By solving $\frac{d\Gamma(\alpha)}{d\alpha} = 0$ considering the scale effects of B and C in practical problems, we obtain the practical formula for quasi-optimal parameters:

$$\alpha_{opt} = \sqrt{\frac{\text{tr}(B^T B)}{m}}. \quad (9)$$

5. Numerical Experiments

In order to accelerate the Krylov subspace iterative approach for the resolution of double saddle point problems, this section demonstrates that the proposed preconditioner is both viable and efficacious.

A personal computer running Windows 10 with a 2.40 GHz Intel(R) Core(TM) i5-9300H CPU, 8.00 GB of RAM (HP laptop, made in China), and MATLAB (version R2019a) was used to acquire all numerical results. The preconditioned FGMRES iterative method's iteration count (IT) and CPU time (CPU) were recorded. Every experiment was run ten times, and the related averages are represented by CPU and IT, with IT values rounded to the closest whole number.

In practical computations, if the initial vector $\mathbf{u}^{(0)} = \mathbf{0} \in \mathbb{R}^{n+m+l}$ is chosen, the iteration will stop when the current iterate $\mathbf{u}^{(k)}$ satisfies the following condition:

$$\text{RES} = \frac{\|\mathbf{b} - \mathcal{A}\mathbf{u}^{(k)}\|_2}{\|\mathbf{b}\|_2} < 10^{-6}.$$

Alternatively, if $\text{IT} > 1500$ is satisfied by the number of iterations. If the number of steps in the numerical table exceeds 1500, it will display as '-'. The two scenarios listed below determine how the right-hand side vector $\mathbf{b} \in \mathbb{R}^{n+m+l}$ is set:

Case I: The vector $\mathbf{b} \in \mathbb{R}^{n+m+l}$ is selected so that $\mathbf{u}^* = (1, 1, \dots, 1)^T \in \mathbb{R}^{n+m+l}$ is the precise solution to the linear system (2).

Case II: The vector $\mathbf{b} \in \mathbb{R}^{n+m+l}$ is selected so that $\mathbf{u}^* = \text{rand}(n+m+l, 1) \in \mathbb{R}^{n+m+l}$ is the precise solution to the linear system (2), where $\text{rand}(\cdot, 1)$ represents a column vector comprising components generated at random.

The theoretical approximate optimal value of the parameter α , denoted as α_{opt} , was calculated according to formula (9). For the SS preconditioner and the APSS preconditioner, the parameters are set as $\alpha = 0.01$ and $\alpha = 1.5$, respectively.

Example 1. Consider Example 4.1 from reference [27].

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

$$B = (I \otimes F \quad F \otimes I) \in \mathbb{R}^{p^2 \times 2p^2},$$

and

$$C = E \otimes F \in \mathbb{R}^{p^2 \times p^2},$$

where

$$T = \frac{1}{h^2} \cdot \text{tridiag}(-1, 2, -1) \in \mathbb{R}^{p \times p}, F = \frac{1}{h} \cdot \text{tridiag}(0, 1, -1) \in \mathbb{R}^{p \times p},$$

$$E = \text{diag}(1, p+1, 2p+1, \dots, p^2 - p + 1),$$

where the discretization mesh size is given by $h = \frac{1}{(p+1)}$ and Kronecker is denoted by $\otimes$.

In Tables 1 and 2, as well as Tables 3 and 4, we list the iteration steps IT, relative errors RES, and corresponding CPU times of the preconditioned FGMRES iterative method for Example 1 and Example 2 under different dimensional conditions.

Table 1. Numerical Results for Example 1 (Case I).

	$p(n)$	16(1024)	32(4096)	64(16,384)	80(25,600)	128(65,536)
<i>FGMRES</i>	IT	995	—	—	—	—
	CPU	9.3984	—	—	—	—
	RES	3.46×10^{-7}	—	—	—	—
$\mathcal{P}_{BD}$	IT	4	5	6	7	8
	CPU	0.0849	3.0379	97.4431	333.7245	1329.4921
	RES	1.21×10^{-7}	8.93×10^{-7}	2.25×10^{-6}	2.98×10^{-6}	1.14×10^{-6}
$\mathcal{P}_{SS}$	IT	3	3	3	3	3
	CPU	0.0531	1.2941	36.2971	91.1133	443.3911
	RES	1.99×10^{-8}	2.92×10^{-9}	2.98×10^{-7}	2.44×10^{-7}	1.92×10^{-7}
$\mathcal{P}_{APSS}$	IT	10	10	10	10	10
	CPU	0.2207	6.4175	41.6397	117.3392	543.1127
	RES	9.77×10^{-7}	1.16×10^{-7}	3.90×10^{-7}	3.54×10^{-7}	2.36×10^{-7}
$\mathcal{P}_{INB}$	IT	2	2	2	2	2
	CPU	0.0426	1.1416	11.5256	29.9146	195.5428
	RES	5.15×10^{-16}	8.98×10^{-16}	9.73×10^{-16}	9.68×10^{-16}	1.23×10^{-15}

Table 2. Numerical Results for Example 1 (Case II).

	$p(n)$	16(1024)	32(4096)	64(16,384)	80(25,600)	128(65,536)
<i>FGMRES</i>	IT	998	—	—	—	—
	CPU	9.8351	—	—	—	—
	RES	2.97×10^{-7}	—	—	—	—
$\mathcal{P}_{BD}$	IT	5	5	8	9	10
	CPU	0.1036	5.4263	100.3112	335.2901	1391.1103
	RES	5.57×10^{-6}	5.18×10^{-6}	9.49×10^{-5}	3.21×10^{-5}	2.39×10^{-5}
$\mathcal{P}_{SS}$	IT	3	3	4	4	4
	CPU	0.0842	2.8921	44.1113	113.3951	480.9231
	RES	9.74×10^{-7}	3.11×10^{-7}	2.59×10^{-7}	1.93×10^{-7}	9.57×10^{-6}
$\mathcal{P}_{APSS}$	IT	10	10	13	39	71
	CPU	0.4588	8.546	56.1103	146.4782	601.0024
	RES	8.26×10^{-7}	3.29×10^{-7}	9.42×10^{-6}	8.81×10^{-6}	6.236×10^{-6}
$\mathcal{P}_{INB}$	IT	2	2	2	3	3
	CPU	0.0592	1.9475	23.5596	36.1291	199.0237
	RES	4.19×10^{-16}	5.92×10^{-16}	3.51×10^{-16}	6.03×10^{-15}	2.11×10^{-15}

Table 3. Numerical Results for Example 2 (Case I).

	$p(n)$	16(2080)	32(8256)	48(18,528)	64(32,896)
<i>FGMRES</i>	IT	263	548	1149	—
	CPU	2.6859	66.9866	1135.3281	—
	RES	9.64×10^{-7}	9.87×10^{-7}	9.96×10^{-7}	—
$\mathcal{P}_{BD}$	IT	6×10^{-7}	6	6	6
	CPU	0.3015	6.2943	46.8373	183.1471
	RES	1.34×10^{-11}	1.98×10^{-14}	1.52×10^{-16}	8.68×10^{-16}
	IT	4	3	2	2

$\mathcal{P}_{SS}$	CPU	0.2688	2.9105	16.0167	66.7651
	RES	2.85×10^{-8}	3.53×10^{-8}	5.94×10^{-7}	1.94×10^{-7}
$\mathcal{P}_{APSS}$	IT	39	31	24	20
	CPU	2.0574	30.1911	188.0799	501.3303
	RES	9.57×10^{-7}	7.63×10^{-7}	9.83×10^{-7}	7.58×10^{-7}
$\mathcal{P}_{INB}$	IT	2	2	2	2
	CPU	0.1012	1.81095	12.9366	54.1412
	RES	4.81×10^{-12}	1.65×10^{-13}	3.27×10^{-14}	1.33×10^{-14}

Table 4. Numerical Results for Example 2 (Case II).

	$p(n)$	16(2080)	32(8256)	48(18,528)	64(32,896)
$FGMRES$	IT	316	616	1172	—
	CPU	3.1572	99.3511	1210.3271	—
	RES	8.49×10^{-7}	6.62×10^{-7}	6.32×10^{-7}	—
$\mathcal{P}_{BD}$	IT	6	7	7	7
	CPU	0.3563	8.1930	59.9631	228.0478
	RES	9.35×10^{-10}	7.32×10^{-13}	2.34×10^{-15}	9.98×10^{-15}
$\mathcal{P}_{SS}$	IT	4	3	2	2
	CPU	0.2841	4.3952	18.4137	76.7519
	RES	1.19×10^{-8}	3.17×10^{-8}	4.79×10^{-7}	1.43×10^{-7}
$\mathcal{P}_{APSS}$	IT	41	36	28	20
	CPU	2.3332	40.9712	400.3982	603.1199
	RES	7.22×10^{-7}	4.93×10^{-7}	7.22×10^{-7}	7.58×10^{-7}
$\mathcal{P}_{INB}$	IT	2	2	2	2
	CPU	0.1633	2.1116	15.3334	61.0019
	RES	4.51×10^{-12}	1.13×10^{-13}	3.03×10^{-14}	1.15×10^{-14}

Specifically, if the iteration steps exceed 1500, the value is omitted.

It is clear from Tables 1 and 2 that the INB method demonstrates remarkable performance in numerical computations, featuring fewer iteration steps, shorter computational time, and higher solution accuracy. Its advantages are particularly prominent in large-scale problems. The efficiency, stability, and high precision of the INB method make it a preferable algorithm compared to preconditioners such as BD, SS, and APSS.

Example 2. Consider *Example 2* from reference [28]. The present study is concerned with the 3×3 block saddle point problem (2), which can be defined as follows:

$$A = \text{diag}(2W^T W + D_1, D_2, D_3) \in \mathbb{R}^{n \times n},$$

is a block-diagonal matrix;

$$B = \begin{bmatrix} E, -I_{2\tilde{p}}, I_{2\tilde{p}} \end{bmatrix} \in \mathbb{R}^{m \times n}, \text{ and } C = E^T \in \mathbb{R}^{l \times m}$$

are both full rank matrices, where, $\tilde{p} = p^2$, and $\hat{p} = p(p+1)$; $W = (w_{ij}) \in \mathbb{R}^{\hat{p} \times \hat{p}}$; has en-

tries $w_{ij} = e^{-2((i/3)^2 + (j/3)^2)}$ $D_1 = I_{\hat{p}}$ is an identity matrix, the diagonal matrices

$D_i = \text{diag}(d_j^{(i)}) \in \mathbb{R}^{2\tilde{p} \times 2\tilde{p}}$, $i = 2, 3$, are defined by

$$d_j^2 = \begin{cases} 1, & \text{for } 1 \leq j \leq \tilde{p} \\ 10^{-5} (j - \tilde{p})^2, & \text{for } \tilde{p}+1 \leq j \leq 2\tilde{p} \end{cases}$$

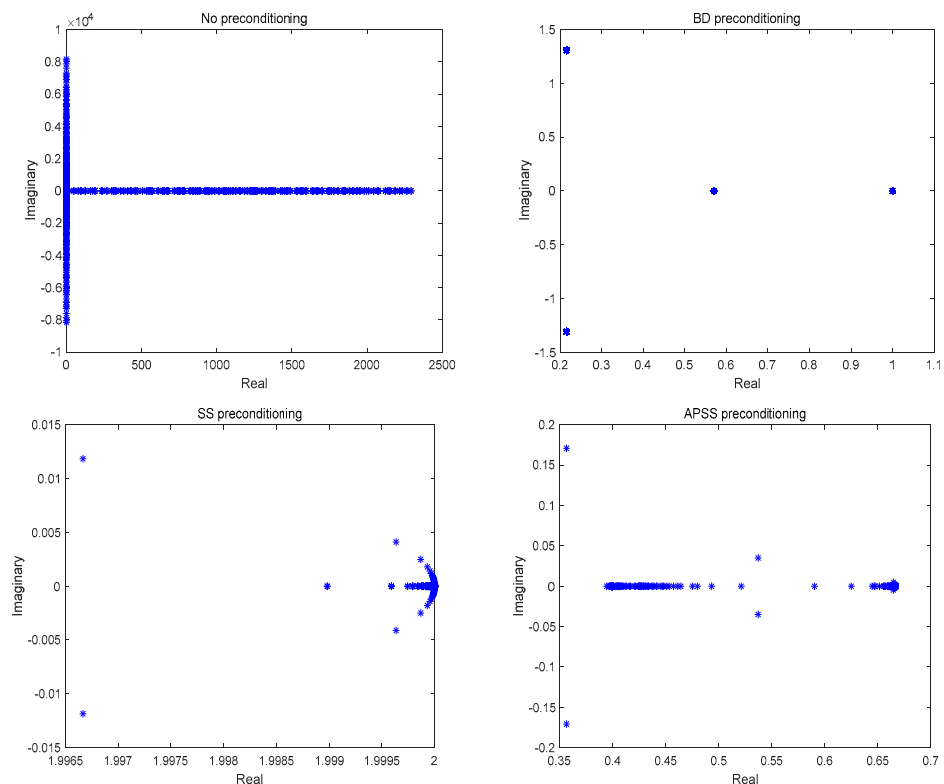
$$d_j^3 = 10^{-5} (j + \tilde{p})^2, \quad \text{for } 1 \leq j \leq 2\tilde{p}.$$

Finally

$$E = \begin{pmatrix} \hat{E} \otimes I_p \\ I_p \otimes \hat{E} \end{pmatrix}, \quad \hat{E} = \begin{pmatrix} 2 & -1 & & \\ & 2 & -1 & \\ & & \ddots & \ddots \\ & & & 2 & -1 \end{pmatrix} \in \mathbb{R}^{p \times (p+1)},$$

where Kronecker is denoted by $\otimes$.

In Figures 1 and 2, we depict the eigenvalue distributions of the original coefficient matrix for Example 1 and the coefficient matrices preconditioned by BD, SS, APSS, and INB. More quantitatively, the eigenvalues of the preconditioned matrices are observed to lie in a narrow neighborhood of one, with a spectral radius close to unity and only small imaginary components. This indicates that the non-unit eigenvalues are well separated from the origin and strongly clustered, which is consistent with the spectral analysis in Section 3 and explains the robust and fast convergence behavior of the INB preconditioner. This advantage translates into efficient convergence of iterative methods, optimized utilization of computational resources, and high-precision solutions, providing powerful algorithmic support for solving large-scale linear systems.



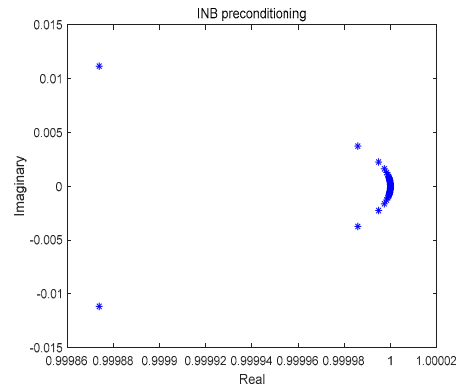
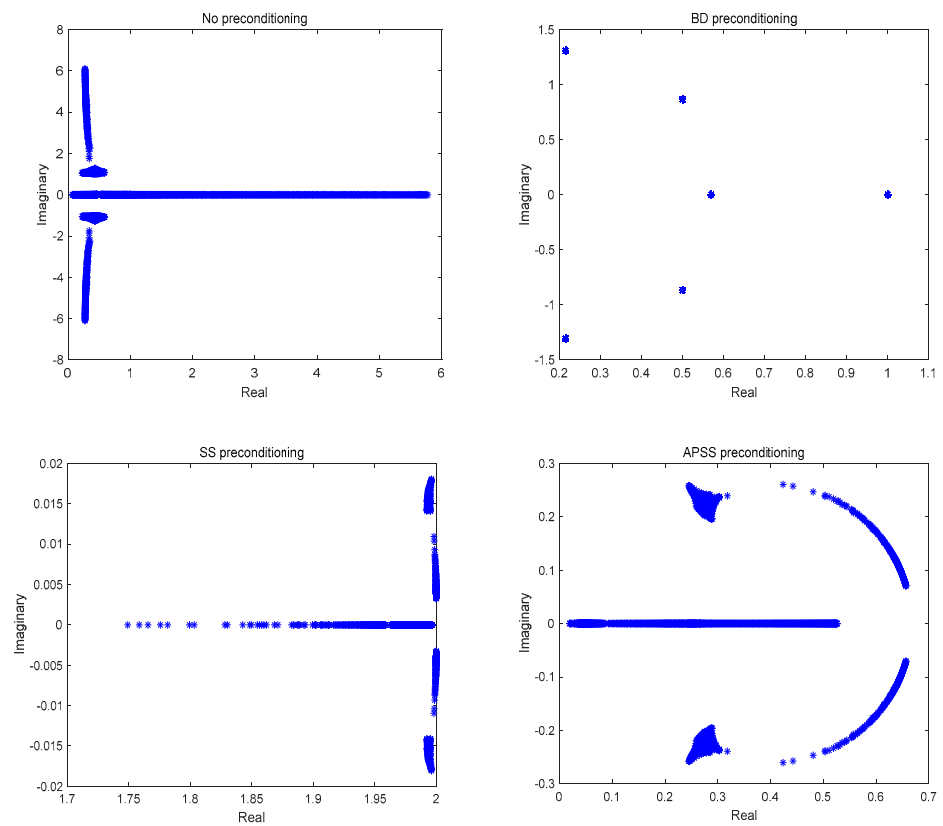


Figure 1. Spectral distribution for Case I when $p = 16$ in Example 1: No preconditioner, B D preconditioner, SS preconditioner, APSS preconditioner, INB preconditioner.



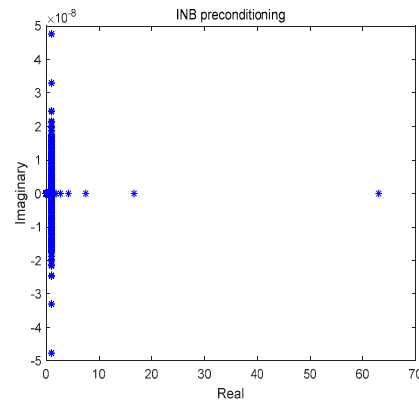


Figure 2. Spectral distribution for Case I when $p = 10$ in Example 2: No preconditioner, B D preconditioner, SS preconditioner, APSS preconditioner, INB preconditioner.

6. Conclusions

When solving block linear systems with Krylov subspace iterative methods, the use of an effective preconditioner is of paramount importance. The main novelty of this work lies in the construction of an improved new block (INB) preconditioner that (i) avoids explicit Schur complement solves, (ii) admits a rigorous semi-convergence and spectral analysis, and (iii) allows for efficient and robust parameter selection through approximate analytical formulas. Numerical results demonstrate that the INB preconditioner achieves superior eigenvalue clustering and significantly reduces iteration counts compared with several state-of-the-art methods.

The present analysis assumes that the leading block A is symmetric positive definite and that the matrices B and C are full row-rank. For highly ill-conditioned problems or singular leading blocks, the Frobenius-norm-based parameter selection may not be globally optimal, although it remains robust in practice. Extensions to singular A , variable-coefficient systems, and 4×4 block structures are possible and form part of ongoing work. The method is amenable to parallel implementation due to its block structure and favorable memory footprint compared with block triangular preconditioners.

Author Contributions: X.-H.S.: Writing review & editing, Visualization, Methodology, Investigation, Formal analysis, Resources, Funding acquisition, Supervision, Project administration. X.-Y.L.: Writing original draft, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by [the Fundamental Research Funds for the Central Universities] grant number [N2224005-1] And The APC was funded by [the Fundamental Research Funds for the Central Universities] grant number [N2224005-1].

Data Availability Statement: Data will be made available on request.

Acknowledgments: The project is supported by the Fundamental Research Funds for the Central Universities (N2224005-1).

Conflicts of Interest: The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Hu, K.; Xu, J. Structure-preserving finite element methods for stationary MHD models. *Math. Comput.* **2019**, *88*, 553–581.
2. Monk, P. Analysis of a finite element method for Maxwell's equations. *SIAM J. Numer. Anal.* **1992**, *29*, 714–729.
3. Degond, P.; Raviart, P.A. An analysis of the Darwin model of approximation to Maxwell's equations. *Forum Math.* **1992**, *4*, 13–44.
4. Benzi, M.; Golub, G.H.; Liesen, J. Numerical solution of saddle point problems. *Acta Numer.* **2005**, *14*, 1–137.
5. Bai, Z.Z. Eigenvalue estimates for saddle point matrices of Hermitian and indefinite leading blocks. *J. Comput. Appl. Math.* **2013**, *237*, 295–306.
6. Huang, N.; Ma, C.F. Spectral analysis of the preconditioned system for the 3×3 block saddle point problem. *Numer. Algorithms* **2019**, *81*, 421–444.
7. Cao, Y. Shift-splitting preconditioners for a class of block three-by-three saddle point problems. *Appl. Math. Lett.* **2019**, *96*, 40–46.
8. Wang, L.; Zhang, K. Generalized shift-splitting preconditioner for saddle point problems with block three-by-three structure. *Open Access Libr. J.* **2019**, *6*, 1–14.
9. Chen, C.; Ma, C. A generalized shift-splitting preconditioner for saddle point problems. *Appl. Math. Lett.* **2015**, *43*, 49–55.
10. Cao, Y.; Li, S.; Yao, L.Q. A class of generalized shift-splitting preconditioners for nonsymmetric saddle point problems. *Appl. Math. Lett.* **2015**, *49*, 20–27.
11. Zhang, N.; Li, R.X.; Li, J. Lopsided shift-splitting preconditioner for saddle point problems with three-by-three structure. *Comput. Appl. Math.* **2022**, *41*, 261.
12. Xie, X.; Li, H.B. A note on preconditioning for the 3×3 block saddle point problem. *Comput. Math. Appl.* **2020**, *79*, 3289–3296.
13. Aslani, H.; Salkuyeh, D.K.; Beik, F.P.A. On the preconditioning of three-by-three block saddle point problems. *arXiv* **2020**, arXiv:2003.10522.
14. Salkuyeh, D.K.; Aslani, H.; Liang, Z.Z. An alternating positive semidefinite splitting preconditioner for the three-by-three block saddle point problems. *Math. Commun.* **2021**, *26*, 177–195.
15. Aslani, H.; Khojasteh Salkuyeh, D. On the preconditioned APSS iterative method for singular coupled saddle point problems. *Math. Commun.* **2024**, *29*, 21–37.
16. Xiong, X.; Li, J. A simplified relaxed alternating positive semi-definite splitting preconditioner for saddle point problems with three-by-three block structure. *J. Appl. Math. Comput.* **2023**, *69*, 2295–2313.
17. Abdolmaleki, M.; Karimi, S.; Salkuyeh, D.K. A new block-diagonal preconditioner for a class of 3×3 block saddle point problems. *Mediterr. J. Math.* **2022**, *19*, 43.
18. Ma, H.; Zhang, N. A note on block-diagonally preconditioned PIU methods for singular saddle point problems. *Int. J. Comput. Math.* **2011**, *88*, 3448–3457.
19. Li, J. Extended Shift-Splitting Iteration Method For Nonsymmetric Generalized Saddle Point Problems. *Appl. Math. E-Notes* **2018**, *18*, 106–115.
20. Zheng, Q.; Lu, L. Extended shift-splitting preconditioners for saddle point problems. *J. Comput. Appl. Math.* **2017**, *313*, 70–81.
21. Balani, F.B.; Hajarian, M.; Bergamaschi, L. Two block preconditioners for a class of double saddle point linear systems. *Appl. Numer. Math.* **2023**, *190*, 155–167.
22. Bai, Z.Z. On semi-convergence of Hermitian and skew-Hermitian splitting methods for singular linear systems. *Computing* **2010**, *89*, 171–197.
23. Ayman-Mursaleen, M. Approximation and Convergence Analysis of Blending-Type q -Baskakov Operators Using Wavelet Transformations. *Filomat* **2025**, *39*, 11117–11130.
24. Cao, Y.; Du, J.; Niu, Q. Shift-splitting preconditioners for saddle point problems. *J. Comput. Appl. Math.* **2014**, *272*, 239–250.
25. Zhou, S.W.; Yang, A.L.; Dou, Y.; Wu, Y.J. The modified shift-splitting preconditioners for nonsymmetric saddle-point problems. *Appl. Math. Lett.* **2016**, *59*, 109–114.
26. Chen, F.; Ren, B.C. A modified alternating positive semidefinite splitting preconditioner for block three-by-three saddle point problems. *Electron. Trans. Numer. Anal.* **2023**, *58*, 84–100.

27. Li, J.; Xiong, X. A new block triangular preconditioner for three-by-three block saddle-point problem. *Appl. Math.* **2024**, *69*, 67–91.
28. Li, J.; Li, Z. A modified preconditioner for three-by-three block saddle point problems. *Jpn. J. Ind. Appl. Math.* **2024**, *41*, 659–680.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.