SIAM J. Matrix Anal. Appl.
Vol. 42, No. 2, pp. 659–682

A POWER SCHUR COMPLEMENT LOW-RANK CORRECTION PRECONDITIONER FOR GENERAL SPARSE LINEAR SYSTEMS[*]

QINGQING ZHENG[†], YUANZHE XI[‡], and YOUSEF SAAD[§]

Abstract. A parallel preconditioner is proposed for general large sparse linear systems that combines a power series expansion method with low-rank correction techniques. To enhance convergence, a power series expansion is added to a basic Schur complement iterative scheme by exploiting a standard matrix splitting of the Schur complement. One of the goals of the power series approach is to improve the eigenvalue separation of the preconditioner thus allowing an effective application of a low-rank correction technique. Experiments indicate that this combination can be quite robust when solving highly indefinite linear systems. The preconditioner exploits a domain-decomposition approach, and its construction starts with the use of a graph partitioner to reorder the original coefficient matrix. In this framework, unknowns corresponding to interface variables are obtained by solving a linear system whose coefficient matrix is the Schur complement. Unknowns associated with the interior variables are obtained by solving a block diagonal linear system where parallelism can be easily exploited. Numerical examples are provided to illustrate the effectiveness of the proposed preconditioner, with an emphasis on highlighting its robustness properties in the indefinite case.

Key words. low-rank correction, Schur complement, power series expansion, domain decomposition, parallel preconditioner, Krylov subspace method

AMS subject classification. 65F10

DOI. 10.1137/20M1316445

1. Introduction.

Consider the solution of the following linear system:

$$Az = b, \tag{1.1}$$

where $A \in \mathbb{R}^{n\times n}$ is a large sparse matrix and $b \in \mathbb{R}^n$ is a given vector. Preconditioned Krylov subspace methods are often used for solving such systems see, e.g., [26]. Among the most popular general-purpose preconditioners are the incomplete LU (ILU) techniques [16, 25]. However, ILU often fails, especially in situations when the matrix is highly indefinite [21, 29]. In addition, due to their sequential nature, ILU preconditioners will result in poor performance on massively parallel high-performance computers. Algebraic multigrid methods constitute another class of popular techniques for solving problems arising from some discretized elliptic PDEs. Often, algebraic multigrid methods also fail for indefinite problems. Finally, sparse approximate inverse preconditioners [5, 8, 14, 18] were developed to overcome these shortcomings but were later abandoned by practitioners due to their high memory demand.

[*]Received by the editors January 31, 2020; accepted for publication (in revised form) by L. Giraud January 5, 2021; published electronically April 27, 2021.
https://doi.org/10.1137/20M1316445

Funding: The work of the first author was supported by the National Natural Science Foundation of China grant 12001311 and the Shuimu Scholar of Tsinghua University. The work of the second author was supported by the National Science Foundation (NSF) grant OAC-2003720. The work of the third author was supported by the NSF grant DMS-1912048 and the Minnesota Supercomputing Institute.

[†]Department of Mathematical Sciences, Tsinghua University, Bejing, China (zheng1990@mail.tsinghua.edu.cn).
[‡]Department of Mathematics, Emory University, Atlanta, GA 30322 USA (yxi26@emory.edu).
[§]Computer Science & Engineering, University of Minnesota, Twin Cities, Minneapolis, MN 55455-0154 USA (saad@umn.edu).

Recently, a new class of approximate inverse preconditioners based on low-rank approximations has been studied. In [20], the Schur complement low-rank (SLR) preconditioner was presented, whose basic idea is that the difference between the inverse of the Schur complement and that of the (2,2) submatrix tends to be of low rank. The multilevel extensions of SLR such as multilevel SLR (MSLR) and generalized MSLR (GMSLR) address the issue that the Schur complement itself can be very large for 3D problems. They exploit the block diagonal structure of the Schur complement so as to enable recursivity to further reduce the costs. In the MSLR and GMSLR preconditioners, a hierarchical interface decomposition (HID) via the nested dissection algorithm was used. These preconditioners approximate the Schur complement or its inverse by exploiting various low-rank corrections, and because they are essentially approximate inverse methods they tend to perform rather well on indefinite linear systems. However, there are some issues for these methods. For example, it is hard to control the size of the interface and achieve high parallelism in the construction stage when nested dissection is used. One of the motivations of this paper is to propose a method that is a move away from nested dissection. Similar low-rank correction ideas have also been exploited in [11, 17, 1, 12]. A related class of methods is the class of rank structured matrix methods, which include the hierarchically off-diagonal low-rank matrix [3], the $\mathcal{H}$-matrix [4, 6, 7], the $\mathcal{H}^2$-matrix [15], and hierarchically semiseparable (HSS) matrices [32]. These methods partition the coefficient matrix A (or *intermediate* blocks that appear during a factorization, like in reference [22] or other low-rank multifrontal solvers) into several smaller blocks and approximate certain off-diagonal blocks by low-rank matrices. For example, the STRUctured Matrix PACKage solver [12] was proposed based on the HSS format. The block low-rank format [1] was used in [2] and [24] to develop structured Multifrontal Massively Parallel sparse direct Solver and Parallel Sparse matrix package solvers, respectively. The hierarchically block separable format was used in [13, 23] to develop the fast structured inversion algorithm. These techniques have recently been applied to precondition sparse linear systems, resulting in some rank structured sparse preconditioners. We refer the reader to [31, 32, 33, 34, 7, 22, 30] for details.

In this paper, we present a method that combines low-rank approximation methods with a simple Neumann polynomial expansion technique [26, section 12.3.1] aimed at improving robustness. We call the resulting method the *power–Schur complement low-rank* (PSLR) preconditioner. A straightforward way to apply the Neumann polynomial preconditioning technique to the Schur complement S is to approximate $(\omega S)^{-1}$ by an m-term polynomial expansion as [26, section 12.3.1]

$$\frac{1}{\omega}\big[I + N + N^2 + \cdots + N^m\big]D^{-1}, \tag{1.2}$$

where ω is a scaling parameter, D is the (block) diagonal of S, and $N = I - \omega D^{-1}S$. However, scheme (1.2) has a number of disadvantages. For example, it is difficult to choose an optimal value for the parameter ω. In addition, since the matrix series in (1.2) converges only when $\rho(N) < 1$, the approximation accuracy will improve as m increases only under this condition which may not be satisfied for a general matrix. Moreover, even if $\rho(N) < 1$, (1.2) is only a rough approximation to S^{-1} when m is small, and using a large m may become computationally expensive. The PSLR preconditioner seamlessly combines the power series expansion with a few low-rank correction techniques and can overcome these shortcomings. We summarize below the main advantages of the PSLR preconditioner over existing low-rank approximate inverse preconditioners.

1. **Improved robustness**. When $\rho(N) > 1$, the classical Neumann series defined by (1.2) diverges and the approximation accuracy deteriorates as m increases. However, low-rank correction techniques can be invoked to address this issue. More specifically, we exploit low-rank correction techniques as a form of deflation to move those eigenvalues of N with modulus larger than 1 closer to 0. The goal is to make the series (1.2) converge for the "deflated" Schur complement.
2. **Enhanced decay property**. The performance of each of the three previously developed methods, SLR, MSLR, and GMSLR, depends on the eigenvalue decay property associated with the Schur complement inverse S^{-1}. If the decay rate is slow, these preconditioners are not effective. On the other hand, the PSLR preconditioner can control the eigenvalue decay rate of the matrix to be approximated by adjusting the number of the expansion term m in (1.2), and this can significantly improve performance.
3. **High parallelism**. The low-rank correction terms used in the PSLR preconditioner can be computed by solving several linear systems with coefficient matrices that are block diagonal. This results in a much more efficient treatment than with the MSLR and GMSLR preconditioners since ILU factorizations and the resulting triangular solves can be applied efficiently in parallel. In addition, most of the important matrix-vector products of PSLR involve block diagonal matrices or dense matrices, leading to a high degree of parallelism in both the construction and the application stage.
4. **Suitability for general matrices**. PSLR is quite effective in handling general sparse problems. Unlike SLR and MSLR, it is not restricted to symmetric systems. Numerical experiments in section 4 illustrate that the PSLR preconditioner outperforms the other low-rank approximation based preconditioners on various tests.

The paper is organized as follows. Section 2 is a brief review of graph partitioning, which will be used to reorder the original coefficient matrix A. Section 3 shows how to build the PSLR preconditioner by exploiting low-rank approximations and a power series expansion associated with the inverse of a certain Schur complement S. A spectral analysis for the corresponding preconditioned matrix is also developed. Section 4 reports on numerical experiments to illustrate the efficiency and robustness of the PSLR preconditioner. Concluding remarks are stated in section 5.

2. Background: Graph partitioning.

Building the PSLR preconditioner begins with a reordering of the coefficient matrix A with the help of a graph partitioner [20, 26]. Specifically, in this paper, we invoke any vertex-based (also known as "edge separation") partitioner to reorder A. As there is no ambiguity, we will still use A and b to denote the reordered matrix and right-hand side, respectively.

Let s be the number of subdomains used in the partitioning. When the variables are labeled by subdomains and the interface variables are labeled last, the permuted linear system of (1.1) can be rewritten as

$$Az = \begin{pmatrix} B & E \\ F & C \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}, \tag{2.1}$$

where $B \in \mathbb{R}^{p\times p}, E \in \mathbb{R}^{p\times q}$, and $F \in \mathbb{R}^{q\times p}$ with $p + q = n$. The submatrices B, E, C have the following block diagonal structures:

$$B = \begin{pmatrix} B_1 & & & \\ & B_2 & & \\ & & \ddots & \\ & & & B_s \end{pmatrix}, \ E = \begin{pmatrix} E_1 & & & \\ & E_2 & & \\ & & \vdots & \\ & & & E_s \end{pmatrix}, \ C = \begin{pmatrix} C_1 & C_{12} & \cdots & C_{1s} \\ C_{21} & C_2 & \cdots & C_{2s} \\ \vdots & \vdots & \ddots & \vdots \\ C_{s1} & C_{s2} & \cdots & C_s \end{pmatrix},$$

while F has the same block structure as that of E^T.

For each subdomain i, B_i denotes the matrix corresponding to the interior variables, C_i represents the matrix associated with local interface variables, and the matrices E_i and F_i denote the couplings to local interface variables and the couplings from local interface variables, respectively. A matrix C_{ij} is a nonzero matrix if and only if some interface variables of subdomain i are coupled with some interface variables of subdomain j.

After it is reordered, the solution to (2.1) can be found by solving two intermediate problems

$$\begin{cases} Sy = g - FB^{-1}f, \\ Bx = f - Ey, \end{cases} \tag{2.2}$$

where $S = C - FB^{-1}E$ is the *Schur complement* of the coefficient matrix in (2.1).

Since B and E are block diagonal, the second equation in (2.2) can be solved efficiently once the vector y becomes available. Many efforts have been devoted to develop preconditioners for solving linear systems associated with S in the first equation. Algebraic recursive multilevel solvers are a class of multilevel ILU-type preconditioners [26, 27] that consist of dropping small entries of S before applying an ILU factorization to it. The SLR preconditioner [20] developed more recently approximates S^{-1} by the sum of C^{-1} and a low-rank correction term. Here the low-rank correction term is computed by exploiting the eigenvalue decay property of $S^{-1} - C^{-1}$. A relative to SLR is the MSLR preconditioner [28] which approximates S^{-1} by applying the same idea as in SLR recursively in order to address the scalability issue. Finally GMSLR [10] was developed as a generalization of MSLR to nonsymmetric systems.

3. The PSLR preconditioner.

In this section, we first derive a power series expansion of S^{-1} and then discuss low-rank correction techniques whose goal is to improve its approximation accuracy.

3.1. Power series expansion of the inverse of the Schur complement.

The proposed power series expansion is applied to a splitting form of S rather than S itself. Specifically, we first write the Schur complement S as the difference of two matrices:

$$S = C_0 - E_s, \tag{3.1}$$

where

$$C_0 = \operatorname{diag}\begin{pmatrix} C_1, & C_2, & \ldots, & C_s \end{pmatrix}$$

is the block diagonal part of C and $E_s = C_0 - S$. Note that $E_s = (C_0 - C) + FB^{-1}E$. Then we have

$$S^{-1} = (I - C_0^{-1}E_s)^{-1}C_0^{-1}. \tag{3.2}$$

Next, we simply apply an $(m+1)$-term power series expansion of $(I - C_0^{-1}E_s)^{-1}$ to obtain the following approximation to S^{-1}:

$$S^{-1} \approx \sum_{i=0}^{m} (C_0^{-1}E_s)^i C_0^{-1}. \tag{3.3}$$

One immediate advantage of using (3.3) is that the application of $\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}$ on a vector only involves linear system solutions associated with C_0 and B, as well as matrix vector multiplications associated with E and F. The block diagonal structures in these three matrices make these operations extremely efficient.

Using results with standard norms it is straightforward to prove the following proposition which analyzes the approximation accuracy of (3.3).

PROPOSITION 3.1. *If the spectral radius of $C_0^{-1}E_s$ satisfies $\rho(C_0^{-1}E_s) < 1$, then*

$$S^{-1} = \sum_{i=0}^{m} (C_0^{-1}E_s)^i C_0^{-1} + R, \tag{3.4}$$

where the error matrix

$$R = \sum_{i=m+1}^{\infty} (C_0^{-1}E_s)^i C_0^{-1} \tag{3.5}$$

satisfies

$$\|R\| \leq \frac{\|C_0^{-1}E_s\|^{m+1}\|C_0^{-1}\|}{1 - \|C_0^{-1}E_s\|}. \tag{3.6}$$

A large class of matrices satisfy the condition $\rho(C_0^{-1}E_s) < 1$ as required in Proposition 3.1. For example, we can show that $\rho(C_0^{-1}E_s) < 1$ holds whenever A is symmetric positive definite (SPD) and its (2,2)-block C is diagonally dominant in the next lemma.

LEMMA 3.2. *If A in* (2.1) *is SPD, then*

$$\lambda(C_0^{-1}E_s) < 1. \tag{3.7}$$

Moreover, if the (2,2)*-block C of A is diagonally dominant, then*

$$\lambda(C_0^{-1}E_s) > -1. \tag{3.8}$$

Here $\lambda(\cdot)$ denotes any eigenvalue of a matrix.

Proof. Since A is SPD, S and C_0 are also SPD, and $C_0^{-\frac{1}{2}}SC_0^{-\frac{1}{2}}$ is SPD. Thus the eigenvalues of $C_0^{-1}S$, which is similar to $C_0^{-\frac{1}{2}}SC_0^{-\frac{1}{2}}$, are all real and positive. Here, for two given matrices $G_1, G_2 \in R^{n\times n}$, if there is a nonsingular matrix P such that $P^{-1}G_1P = G_2$, then matrices G_1 and G_2 are similar. Moreover,

$$C_0^{-1}E_s = C_0^{-1}(C_0 - S) = I - C_0^{-1}S, \tag{3.9}$$

and this shows that $\lambda(C_0^{-1}E_s) < 1$.

Now we prove the second part of this lemma. Let

$$C_g = C - C_0,$$

which is the matrix C stripped off its diagonal blocks, and note that $C_0-C_g = 2C_0-C$. Then we have

$$\begin{aligned} 2I - C_0^{-1}S &= C_0^{-1}(2C_0 - S) \\ &= C_0^{-1}(C_0 - C_g + E^TB^{-1}E), \end{aligned}$$

which is similar to

$$\Phi = C_0^{-\frac{1}{2}}(C_0 - C_g + E^TB^{-1}E)C_0^{-\frac{1}{2}}.$$

Since C is a diagonally dominant matrix, the matrix $C_0 - C_g$ is also diagonally dominant. This results in the symmetric positive definiteness of Φ. Hence, the eigenvalues of $2I - C_0^{-1}S$ are all positive, leading to

$$\lambda(C_0^{-1}S) < 2. \tag{3.10}$$

This along with (3.9) yields the desired result: $\lambda(C_0^{-1}E_s) = 1 - \lambda(C_0^{-1}S) > -1$. ◻

Lemma 3.2 shows that $\rho(C_0^{-1}E_s) < 1$, when A is SPD and C is diagonally dominant. As an example, we depict the eigenvalues of $C_0^{-1}E_s$ in Figure 1 for a 3D discretized Laplacian matrix A on a 20^3 grid where the numbers s of subdomains are set to $s = 5, 15, 25, 35$. It is easy to see that the absolute values of all the eigenvalues of $C_0^{-1}E_s$ are smaller than 1. In addition, the sizes of the Schur complement are $1824, 3340, 3944, 4522$ for $s = 5, 15, 25, 35$, respectively.

3.2. Low-rank approximations of S^{-1}. The power series expansion of S^{-1} in section 3.1 only provides a rough approximation to S^{-1}, especially when m is small or/and $\rho(C_0^{-1}E_s)$ is slightly smaller than 1. In this section, we will consider some low-rank correction techniques to improve the accuracy of this approximation. In addition, we will also consider the case when $\rho(C_0^{-1}E_s) > 1$.

First define

$$\widehat{R} = S^{-1} - \sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1}. \tag{3.11}$$

Notice that when $\rho(C_0^{-1}E_s) < 1$, $\widehat{R}$ is equal to the matrix R defined in (3.5).

Then we have

$$S^{-1} = \sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1} + \widehat{R}. \tag{3.12}$$

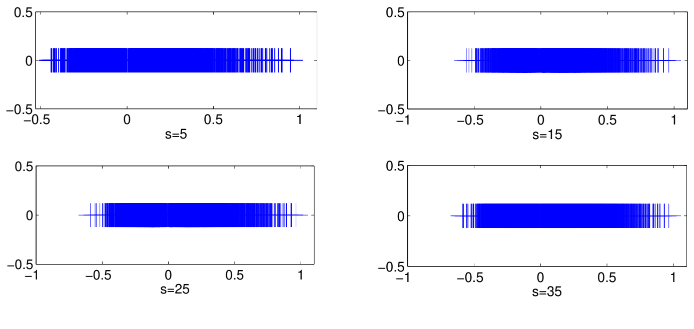

FIG. 1. *Eigenvalues (+) of $C_0^{-1}E_s$ for a 3D Laplacian matrix discretized on a 20^3 grid with the zero Dirichlet boundary condition where the numbers of subdomains $s = 5, 15, 25, 35$.*

Let

$$E_{rr}(m) := S\widehat{R} \in \mathbb{R}^{q\times q}. \tag{3.13}$$

Based on (3.12) we get

$$\begin{aligned} I &= S\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1} + S\widehat{R} \\ &= S\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1} + E_{rr}(m), \end{aligned} \tag{3.14}$$

which leads to

$$S^{-1} = \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}\right](I - E_{rr}(m))^{-1}. \tag{3.15}$$

Here, we assume $I - E_{rr}(m)$ is nonsingular.

Equation (3.15) provides another way to approximate S^{-1}. If a r_k-step Arnoldi procedure is performed on $E_{rr}(m)$, $E_{rr}(m)$ can be approximated by

$$E_{rr}(m) \approx V_{r_k}H_{r_k}V_{r_k}^T, \tag{3.16}$$

where $V_{r_k} \in \mathbb{R}^{q\times r_k}$ has orthonormal columns and $H_{r_k} = V_{r_k}^T E_{rr}(m)V_{r_k} \in \mathbb{R}^{r_k\times r_k}$ is an upper Hessenberg matrix whose eigenvalues can be used to approximate the largest eigenvalues of $E_{rr}(m)$. A simpler expression of $E_{rr}(m)$ is presented in Theorem 3.4 to perform the Arnoldi procedure. For a given m, it can be justified that the Frobenius norm $\|E_{rr}(m) - V_{r_k}H_{r_k}V_{r_k}^T\|_F$ decreases monotonically as r_k increases [26]. As a result, $V_{r_k}H_{r_k}V_{r_k}^T$ approximates $E_{rr}(m)$ more accurately as r_k increases.

Combining (3.15) with (3.16) gives rise to

$$\begin{aligned} S^{-1} &\approx \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}\right](I - V_{r_k}H_{r_k}V_{r_k}^T)^{-1} \\ &= \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}\right](I + V_{r_k}[(I-H_{r_k})^{-1} - I]V_{r_k}^T) \\ &= \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}\right](I + V_{r_k}G_{r_k}V_{r_k}^T), \end{aligned} \tag{3.17}$$

where $G_{r_k} = (I - H_{r_k})^{-1} - I \in \mathbb{R}^{r_k\times r_k}$. In the above process, we utilize the Sherman–Morrison–Woodbury formula to derive the expression of $(I - V_{r_k}H_{r_k}V_{r_k}^T)^{-1}$.

Thus, the final approximation to S^{-1} takes the form

$$S_{\text{app}}^{-1} = \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i C_0^{-1}\right](I + V_{r_k}G_{r_k}V_{r_k}^T). \tag{3.18}$$

Remark 3.3. Although the approximation (3.18) looks similar to the one used in SLR [20], they actually take completely different forms even when $m = 0$. For example, when $m = 0$, from (3.18), we have

$$S_{\text{app}}^{-1} = C_0^{-1}(I + V_{r_k}G_{r_k}V_{r_k}^T). \tag{3.19}$$

Here, $G_{r_k} = (I - H_{r_k})^{-1} - I \in \mathbb{R}^{r_k \times r_k}$ and V_{r_k}, H_{r_k} are obtained by a r_k-step Arnoldi procedure on $E_s C_0^{-1}$:

$$(V_{r_k}, H_{r_k}) = \text{Arnoldi}(E_s C_0^{-1}, r_k). \tag{3.20}$$

However, the low-rank correction terms of the SLR preconditioner are computed based on the decay properties of $S^{-1} - C^{-1}$, and the final approximation to S^{-1} is

$$\widetilde{S}^{-1} = U^{-1}[I + W_{r_k}[(I - R_{r_k})^{-1} - I]W_{r_k}^H]L^{-1}. \tag{3.21}$$

Here, L, U are the LU factors of C and

$$(W_{r_k}, R_{r_k}) = \text{Arnoldi}(L^{-1}FB^{-1}EU^{-1}, r_k). \tag{3.22}$$

Hence we can see that PSLR and SLR are two different preconditioners, whether from the expression form of the preconditioners or the computations of the low-rank correction terms.

3.2.1. Approximation accuracy analysis. In this section, we quantify the approximation accuracy of S_{app}^{-1} in terms of m and r_k. The next theorem first shows the relation between the eigenvalue decay rate of $E_{rr}(m)$ and the number of the power series expansion $m + 1$.

THEOREM 3.4. *For any matrix A, the matrix $E_{rr}(m)$ in (3.13) can be rewritten as*

$$E_{rr}(m) = (E_s C_0^{-1})^{m+1}. \tag{3.23}$$

Proof. Combining (3.1) with (3.14), we have

$$\begin{aligned}
E_{rr}(m) &= I - S\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i\right]C_0^{-1}\\
&= I - (C_0 - E_s)\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i\right]C_0^{-1}\\
&= I - C_0\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i\right]C_0^{-1} + E_s\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^i\right]C_0^{-1}\\
&= I - \sum_{i=0}^{m}(E_sC_0^{-1})^i + \sum_{i=1}^{m+1}(E_sC_0^{-1})^i\\
&= (E_sC_0^{-1})^{m+1}.
\end{aligned}$$

This completes the proof. ∎

Theorem 3.4 shows that the eigenvalues of $E_{rr}(m)$ decay faster as m increases. In fact, the eigenvalue decay rate of $E_{rr}(m)$ is $m+1$ times faster than that of $E_{rr}(0)$. In addition, from Theorem 3.4, we can see that the matrix $E_{rr}(m)$ does not need to be formed exactly during the computation of the low-rank correction terms. In fact, we just compute some sparse matrix-vector products and solve some linear systems with a coefficient matrix that is block-diagonal. These two processes can be implemented efficiently in parallel. Figure 2 and Figure 3 further justify Theorem 3.4 numerically on one symmetric 3D discretized Laplacian matrix (Figure 2) and one nonsymmetric

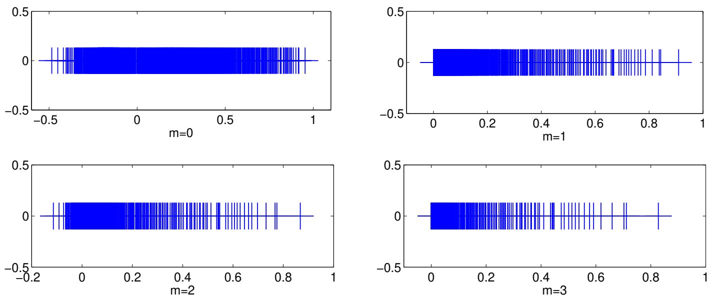

FIG. 2. *Spectrum of $E_{rr}(m)$ with different values of m, where the matrix A is taken as a 3D Laplacian matrix discretized on a 20^3 grid with the zero Dirichlet boundary condition. In this test, the number of subdomains is chosen as $s = 5$ and $\rho(E_sC_0^{-1}) = 0.9537$. Here, a blue $+$ denotes an eigenvalue of $E_sC_0^{-1}$. The size for the Schur complement is $1,824$.*

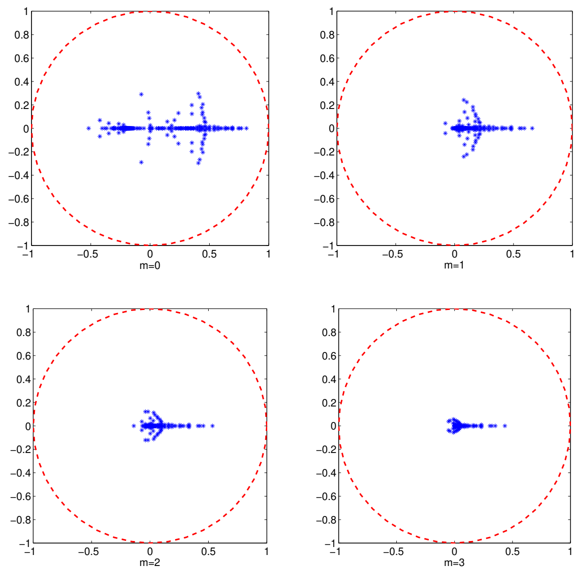

FIG. 3. *Spectrum of $E_{rr}(m)$ with different m, where the matrix A is the nonsymmetric* `pde900` *taken from the SuiteSparse collection* [9]*. The test matrix has the dimension of 900×900 and is indefinite. The number of subdomains used in the partition is $s = 5$ and $\rho(E_sC_0^{-1}) = 0.8117$. Here, a blue $*$ denotes an eigenvalue of $E_sC_0^{-1}$, and the red dashed circle has radius 1. The size for the Schur complement is 324.*

`pde900` matrix from the SuiteSparse collection [9] (Figure 3). The spectral radius of $E_sC_0^{-1}$ is equal to 0.9537 and 0.8117, respectively, in these two examples. As can be seen from Figure 2 and Figure 3, the eigenvalues of $E_{rr}(m)$ get more clustered around the origin when a larger m is used. Here, small values of m, i.e., $m = 0, 1, 2, 3$, are tested.

We then consider two indefinite matrices. The first one is the 3D shifted discretized Laplacian matrix (Figure 4) and the second one is the nonsymmetric `young1c` matrix from the SuiteSparse collection [9] (Figure 5). The indefiniteness causes the spectral radius of $E_sC_0^{-1}$ greater than 1 in both tests. But as can be seen from

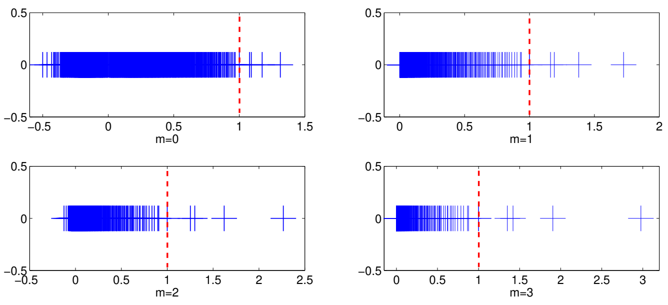

FIG. 4. *Spectrum of $E_{rr}(m)$ with different m, where the matrix A is taken as a shifted 3D Laplacian matrix discretized on a 20^3 grid with the zero Dirichlet boundary condition and the number of subdomains is chosen as $s = 5$. In this test, A is indefinite and has 7 negative eigenvalues, and $E_sC_0^{-1}$ has 4 eigenvalues larger than 1. Here, a blue + denotes an eigenvalue of $E_sC_0^{-1}$. The size for the Schur complement is 1,824.*

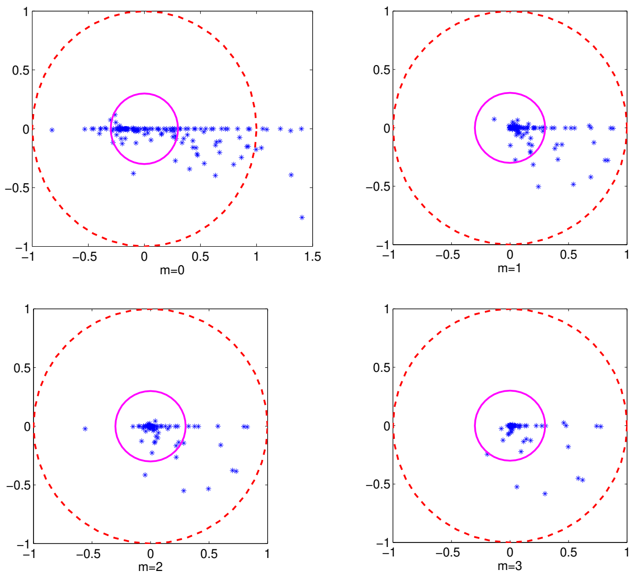

FIG. 5. *Spectrum of $E_{rr}(m)$ with different m, where the matrix A is the nonsymmetric* `young1c` *matrix from the SuiteSparse collection* [9]. *This matrix has the dimension of 841×841, and the number of subdomains used in the partition is $s = 5$. $E_sC_0^{-1}$ has 9 eigenvalues with modulus larger than 1 which are shown outside the red dashed circle in the top-left subfigure. Only the eigenvalues of $E_{rr}(m)$ within the unit red dashed circle are shown for the cases $m = 1, 2, 3$. Here, a blue $*$ denotes an eigenvalue of $E_sC_0^{-1}$; the red dashed circle has radius 1, while the pink solid circle has radius 0.3. For this test, the size for the Schur complement is 285.*

Figures 4–5, only a few eigenvalues have modulus greater than 1. As a result, the majority of the eigenvalues still get clustered around the origin as m increases. Based on this property, we can show that the approximation accuracy of (3.18) can be improved as m increases under mild conditions. In contrast, the classical Neumann series expansion (3.3) will diverge in this case.

We first prove an upper bound of the relative approximation accuracy of S_{app}^{-1} to S^{-1} in the next proposition.

PROPOSITION 3.5. *For any matrix norm $\| \cdot \|$, the approximation accuracy of S_{app}^{-1} to S^{-1} satisfies the following inequality:*

$$\frac{\| S^{-1} - S_{app}^{-1} \|}{\| S^{-1} \|} \leq \| X(m, r_k) \| \| Z(r_k)^{-1} \|, \tag{3.24}$$

where

$$X(m, r_k) = E_{rr}(m) - V_{r_k} H_{r_k} V_{r_k}^T, \ Z(r_k) = I - V_{r_k} H_{r_k} V_{r_k}^T. \tag{3.25}$$

Proof. From (3.15) and (3.17), we have

$$\begin{aligned} S^{-1} - S_{\text{app}}^{-1} &= \Big[\sum_{i=0}^{m} (C_0^{-1} E_s)^i C_0^{-1} \Big] \big[(I - E_{rr}(m))^{-1} - (I - V_{r_k} H_{r_k} V_{r_k}^T)^{-1} \big] \\ &= \Big[\sum_{i=0}^{m} (C_0^{-1} E_s)^i C_0^{-1} \Big] \Big[\big(Z(r_k) - X(m, r_k) \big)^{-1} - Z(r_k)^{-1} \Big] \\ &= \Big[\sum_{i=0}^{m} (C_0^{-1} E_s)^i C_0^{-1} \Big] \Big[\big(Z(r_k) - X(m, r_k) \big)^{-1} \big(Z(r_k) - (Z(r_k) - X(m, r_k)) \big) Z(r_k)^{-1} \Big] \\ &= \Big[\sum_{i=0}^{m} (C_0^{-1} E_s)^i C_0^{-1} \Big] (I - E_{rr}(m))^{-1} X(m, r_k) Z(r_k)^{-1} \\ &= S^{-1} X(m, r_k) Z(r_k)^{-1}. \end{aligned}$$

Using a matrix norm, this yields

$$\| S^{-1} - S_{\text{app}}^{-1} \| \leq \| S^{-1} \| \| X(m, r_k) \| \| Z(r_k)^{-1} \|,$$

from which (3.24) follows. ∎

Next, we provide two numerical experiments to illustrate Proposition 3.5. The Frobenius norm is employed for both tests, and we denote by $\Delta(m, r_k)$ the upper bound $\| X(m, r_k) \| \| Z(r_k)^{-1} \|$ in Proposition 3.5. The first test is a 3D Laplacian matrix, and the second one is a shifted 3D Laplacian matrix. Both matrices have size of $2,000 \times 2,000$. In the tests, we fix $r_k = 15$ and $s = 5$ and change numbers of terms used in the power series expansion from $m = 3$ to $m = 5$. For the Laplacian matrix, we have $\Delta(3, 15) = 0.49$ when $m = 3$ and $\Delta(5, 15) = 0.15$ when $m = 5$. For the shifted Laplacian matrix, $E_{rr}(m)$ has 11 eigenvalues with modulus larger than 1. Since r_k is larger than 11, when $m = 3$ and $m = 5$, we have $\Delta(3, 15) = 0.72$ and $\Delta(5, 15) = 0.665$, respectively. These two tests verify that the approximation is more accurate if m increases as long as the rank r_k is larger than the number of the eigenvalues of E_{rr} with modulus greater than 1. From the results of the above two specific problems, we can see that the upper bound $\Delta(m, r_k)$ is smaller in general for SPD matrices than for indefinite matrices.

3.2.2. Spectral analysis of the preconditioned Schur complement. The preconditioning effect of the proposed PSLR preconditioner depends directly on the eigenvalue distribution of $S_{\text{app}}^{-1} S$. When the eigenvalues of $S_{\text{app}}^{-1} S$ are clustered or close to one, one can expect a fast convergence for Krylov subspace methods.

From (3.15) and (3.17), we have

$$S_{\mathrm{app}}^{-1}S = \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1}\right](I - V_{r_k}H_{r_k}V_{r_k}^T)^{-1}(I - E_{rr}(m))\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1}\right]^{-1}$$

$$= \left[\sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1}\right]Z(r_k)^{-1}\big(Z(r_k) - X(m,r_k)\big)\left[\sum_{i=0}^{m}(C_0^{-1}E_s)^iC_0^{-1}\right]^{-1}, \tag{3.26}$$

where $Z(k)$ and $X(m, r_k)$ are the matrices defined by (3.25). Obviously, it follows from (3.26) that $S_{\mathrm{app}}^{-1}S$ is similar to

$$Z(r_k)^{-1}\big(Z(r_k) - X(m,r_k)\big) = I - Z(r_k)^{-1}X(m,r_k),$$

which implies that

$$\lambda(S_{\mathrm{app}}^{-1}S) = 1 - \lambda(Z(r_k)^{-1}X(m,r_k)).$$

When the eigenvalues of $X(m, r_k)$ (or $Z(r_k)^{-1}X(m, r_k)$) are close to zero, the eigenvalues of $S_{\mathrm{app}}^{-1}S$ are clustered around 1. To illustrate the influence of the approximation accuracy of S_{app}^{-1} on the eigenvalue distribution of $S_{\mathrm{app}}^{-1}S$, we display the eigenvalues of $S_{\mathrm{app}}^{-1}S$ for the same 3D Laplacian matrix presented in section 3.2.1 with $n = 2000$, $r_k = 15$, and $s = 5$ in Figure 6. The numbers of terms used in the power series expansion are $m = 3$ and $m = 5$ for two different cases, respectively. As can be seen from Figure 6, the eigenvalues in the right subfigure are more clustered than those in the left subfigure. This further illustrates the fact that the approximation is improved if m increases but the rank r_k is fixed. For this specific problem, the PSLR preconditioned GMRES method converges in 8 and 17 iterations, respectively, when m is set to 5 and 3 and iteration is stopped when the initial residual is reduced by 10^8. As another illustration, Figure 7 depicts the eigenvalues of $S_{\mathrm{app}}^{-1}S$ for the same shifted 3D Laplacian matrix presented in section 3.2.1 with $n = 2000$, $r_k = 15$, and $s = 5$. For this test, the PSLR preconditioned GMRES method with $m = 5$ converges in 13 iterations and the iteration number increases to 24 when m is reduced to 3, using the same stopping criterion as earlier.

3.3. Construction and application of the PSLR preconditioner.

This section provides a short description of the construction of the PSLR preconditioner

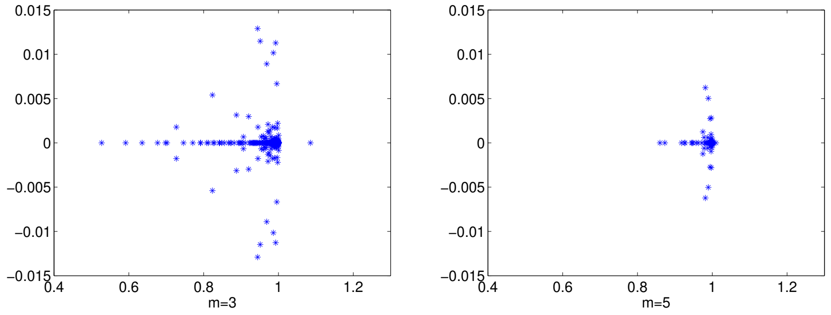

Fig. 6. *The eigenvalue distribution of $S_{app}^{-1}S$ for 3D Laplacian matrix with $r_k = 15$. The number of terms used in the power series expansion are 3 and 5 for left subfigure and right subfigure, respectively.*

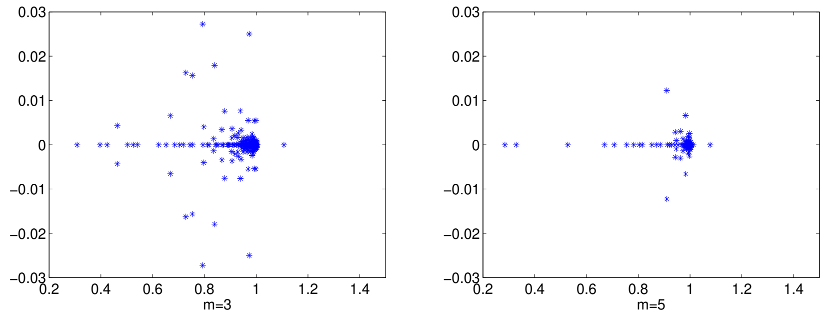

FIG. 7. *The eigenvalue distribution of $S_{app}^{-1}S$ for shifted* 3D *Laplacian matrix with $r_k = 15$. The number of terms used in the power series expansion are* 3 *and* 5 *for left subfigure and right subfigure, respectively.*

and its application. Recall from (2.2), the application of the PSLR preconditioner on a vector b follows the following two steps:

$$\begin{cases} y = S_{\text{app}}^{-1}(g - FB^{-1}f), \\ x = B^{-1}(f - Ey), \end{cases} \tag{3.27}$$

where b is partitioned into $(f^T, g^T)^T$ according to the sizes of B and C.

The scheme (3.27) requires three linear system solutions, two associated with B and one associated with S_{app}. Applying S_{app}^{-1} on a vector based on (3.18) involves solving $m+1$ linear systems associated with C_0. Since both B and C_0 are block diagonal, the construction of the PSLR preconditioner starts with the ILU factorization of these diagonal blocks. The computed ILU factors can then be used in the Arnoldi procedure to compute V_{r_k} and H_{r_k} associated with S_{app}^{-1}. The construction algorithm is summarized in Algorithm 3.1. Theorem 3.4 is referenced to perform the matrix-vector product associated with the matrix $E_{rr}(m)$.

Algorithm 3.1. Construction of PSLR preconditioner.

```
1: Apply domain decomposition to reorder $A$ with $s$ subdomains
2: **For** $i = 1 : s$ Do
3:     $[L_i^B, U_i^B] = \text{ilu}(B_i)$
4:     $[L_i^{C_0}, U_i^{C_0}] = \text{ilu}(C_i)$
5: **EndDo**
6: Apply Arnoldi procedure to compute ▷ *Based on Theorem* 3.4
7:         $[V_{r_k}, H_{r_k}] = \text{Arnoldi}(E_{rr}(m), r_k)$
8: Compute $G_{r_k} = (I - H_{r_k})^{-1} - I$
```

The computational cost of the PSLR construction process is dominated by ILU factorization and the triangular solves involved in applying the operator $E_{rr}(m)$ in the Arnoldi process. Since these operations can be performed independently among different diagonal blocks in B and C_0, Algorithm 3.1 is highly parallelizable.

Algorithm 3.2 describes the application of the PSLR preconditioner on a vector b. Besides linear system solutions associated with B and C_0, the remaining operations are matrix-vector multiplications associated with sparse matrices E, F and dense matrices V_{r_k} and G_{r_k}. Since both E and F are in block diagonal forms, this application algorithm is also highly parallizable.

Algorithm 3.2. Computing $z = \text{PSLR}(b)$.

```
1: Partition $b = \begin{pmatrix} f \\ g \end{pmatrix}$
2: Compute $y = (g - FB^{-1})f$
3: Update $y \leftarrow y + V_{r_k}(G_{r_k}(V_{r_k}^T y))$
4: Compute
       $$y \leftarrow \sum_{i=0}^{m} (C_0^{-1}E_s)^i C_0^{-1} y$$
5: Solve $Bx = f - Ey$
6: Set $z = \begin{pmatrix} x \\ y \end{pmatrix}$
```

3.4. Complexity analysis. In this section, we compare the computational complexity among PSLR, MSLR, and GMSLR under mild assumptions. Let trisol(ILU($\cdot$)) denote the cost of the triangular solve with the corresponding matrix. From Algorithm 3.2, we can see that the total cost γ_{PSLR} of applying PSLR on a vector is

$$\gamma_{PSLR} = (m+1)\text{trisol}(\text{ILU}(C_0)) + 2\text{trisol}(\text{ILU}(B)) + O(\text{sz}(C)r_k). \tag{3.28}$$

Here, sz($\cdot$) is the size of the associated matrix. So we have

$$\gamma_{PSLR} = (m+3)s\gamma_1 + O(\text{sz}(C)r_k),$$

where γ_1 is the cost of trisol(ILU(C_i)) $(i = 1, 2, \ldots, s)$. Here, we assume that the cost of ILU solve for each block in C_0 is the same as that for each block in B. Let the multilevel recursive form of the reordered matrix with the HID ordering be

$$A^{(l)} = \begin{pmatrix} B^{(l)} & E^{(l)} \\ F^{(l)} & C^{(l)} \end{pmatrix}.$$

Assume the total level of a binary HID tree of GMSLR is L and the rank r_k used in PSLR and GMSLR are the same. Also, the size of each block of $B^{(l)}$ $(l = 0, 1, \ldots, L-2)$ is the same as that of $C^{(L-1)}$, and the cost of ILU solve for each block in $B^{(l)}$ $(l = 0, 1, \ldots, L-2)$ is the same as that of $C^{(L-1)}$. Since the number of blocks of $B^{(l)}$ is 2^{L-l-1}, we can obtain the total cost γ_{GMSLR} of applying GMSLR on a vector as

$$\gamma_{GMSLR} = \sum_{l=0}^{L-1} \gamma_{GMSLR}^{l} \tag{3.29}$$

$$= L\text{trisol}(\text{ILU}(C^{(L-1)})) + 2\sum_{l=1}^{L} l\text{trisol}(\text{ILU}(B^{(L-1)})) + \sum_{l=0}^{L-1} O(\text{sz}(C^{(l)})r_k) \tag{3.30}$$

$$= (2^{L+2} - 3L - 4)\gamma_1 + \sum_{l=0}^{L-1} O(\text{sz}(C)r_k).$$

Details about the derivation of γ_{GMSLR}^{l} can be found in section 5.3 of [10]. When $l = 0$, the number of blocks of $B^{(0)}$ is just the number of subdomains used in PSLR and $C^{(0)} = C$, so s in the PSLR is equal to 2^{L-1} in this case. Now we compare γ_{GMSLR} and γ_{PSLR}. Obviously,

$$\begin{aligned}
&\gamma_{GMSLR} - \gamma_{PSLR}\\
&\quad = (2^{L+2} - 3L - 4)\gamma_1 - (m+3)s\gamma_1 + \sum_{l=1}^{L-1} O(\mathrm{sz}(C^{(l)})r_k)\\
&\quad = (5s - ms - 3\log_2 s - 7)\gamma_1 + \sum_{l=1}^{L-1} O(\mathrm{sz}(C^{(l)})r_k)\\
&\quad \geq (5s - ms - 3\log_2 s - 7)\gamma_1.
\end{aligned}$$

Denote $f(s) = (5s - ms - 3\log_2 s - 7)\gamma_1$; then we have the following
(a) if $m = 0$, then $f(s) > 0$ when $s > 2$;
(b) if $m = 1$, then $f(s) > 0$ when $s \geq 3$;
(c) if $m = 2$, then $f(s) > 0$ when $s \geq 5$;
(d) if $m = 3$, then $f(s) > 0$ when $s > 8$;
(e) if $m = 4$, then $f(s) > 0$ when $s \geq 20$.
For cases (a) – (e), $\gamma_{GMSLR} > \gamma_{PSLR}$ holds, which illustrates that the cost of PSLR is less than that of GMSLR. In addition, if $m \geq 5$, then $f(s) < 0$ for any s. In this case, it is possible that $\gamma_{GMSLR} < \gamma_{PSLR}$.

Similar to the GMSLR, we can obtain the following results for MSLR:
(a) if $m = 0$, then $\gamma_{MSLR} > \gamma_{PSLR}$ when $s \geq 2$;
(b) if $m = 1$, then $\gamma_{MSLR} > \gamma_{PSLR}$ when $s \geq 3$;
(c) if $m = 2$, then $\gamma_{MSLR} > \gamma_{PSLR}$ when $s \geq 6$;
(d) if $m = 3$, then $\gamma_{MSLR} > \gamma_{PSLR}$ when $s \geq 8$;
(e) if $m \geq 4$, then $\gamma_{MSLR} < \gamma_{PSLR}$ for any s. In this case, it is possible that $\gamma_{MSLR} < \gamma_{PSLR}$.
The construction cost comparisons among PSLR, MSLR, and GMSLR are similar to the application cost analysis.

4. Numerical examples. In this section, we report numerical experiments to show the efficiency and robustness of the PSLR preconditioner. The test problems include symmetric and nonsymmetric cases. The PSLR preconditioner was implemented in C++ and compiled with the -O3 optimization option. All the experiments were run on a single node of the `Mesabi` Linux cluster at the Minnesota Supercomputing Institute, which has 64 GB or memory and two Intel 2.5 GHz Haswell processors with 12 cores each. The `PartGraphKway` from the METIS [19] package was used to partition matrices. BLAS and LAPACK routines from Intel Math Kernel Library were used to enhance the performance on multiple cores. Thread-level parallelism was realized by OpenMP. The preconditioner construction time consists of the ILU factorizations of matrices B_i, C_i, $i = 1, 2, \ldots, s$, and the computation of V_{r_k} and G_{r_k}. In actual computations, the right-hand side b was chosen randomly such that $Ax = b$ with x being a random vector, and the initial guess on x was always taken as a zero vector in the Krylov subspace methods. In addition, the maximum number of fill-ins for each row of B_i, C_i $(i = 1, 2, \ldots, s)$ was set to be 100.

For the SPD problems, we compare the PSLR preconditioner with the MSLR preconditioner [28] and the incomplete Cholesky factorization preconditioner (ICT) with threshold dropping, and the conjugate gradient (CG) method as the accelerator. For general problems, we compare PSLR with the GMSLR preconditioner and the ILU factorization preconditioner (ILUT) with threshold dropping, using GMRES [26] as the accelerator. BLAS and LAPACK routines from Intel Math Kernel Library were used in incomplete factorizations and MSLR and GMSLR precoditioners. MSLR and GMSLR preconditioners were also parallelized with OpenMP.

In the rest of this section, the following notation is used:

- its: the number of iterations of GMRES or CG to reduce the initial residual norm by 10^8. Moreover, the "F" indicates that GMRES or CG failed to converge within 500 iterations;
- o-t: wall clock time to reorder the matrix;
- p-t: wall clock time for the preconditioner construction;
- i-t: wall clock time for the iteration procedure. If GMRES or CG fails to converge within 500 iterations, then we denote this time by "–";
- t-t: total wall clock time, i.e., the sum of the preconditioner construction time and the iteration time;
- r_k: the rank used in the low-rank correction terms;
- m: the number of terms used in the power series expansion;
- fill (total): the total fill-factor defined as $\frac{\text{nnz}(prec)}{\text{nnz}(A)}$;
- fill (ILU): the fill-factor comes from ILU decompositions defined as $\frac{\text{nnz}(ILU)}{\text{nnz}(A)}$;
- fill (low-rank): fill-factor comes from the low-rank correction terms defined as $\frac{\text{nnz}(LRC)}{\text{nnz}(A)}$.

Here $\text{nnz}(X)$ denotes the number of nonzero entries of a matrix X. Moreover,

$$\begin{aligned}
\text{nnz}(ILU) &= \sum_{i=1}^{s} \left[\text{nnz}(L_i^B) + \text{nnz}(U_i^B) + \text{nnz}(L_i^{C_0}) + \text{nnz}(U_i^{C_0})\right],\\
\text{nnz}(LRC) &= \text{nnz}(V_{r_k}) + \text{nnz}(G_{r_k}),\\
\text{nnz}(prec) &= \text{nnz}(ILU) + \text{nnz}(LRC).
\end{aligned}$$

Note that we employ the notation $\text{nnz}(V_{r_k})$ and $\text{nnz}(G_{r_k})$ for dense matrices. The term *fill-factor*, which is meant to reflect memory usage, mixes traditional fill-in (ILU) along with the additional memory needed to store the (dense) low-rank correction matrices.

4.1. Test 1. Consider the following symmetric problem:

$$\begin{aligned}
-\triangle u - \beta u &= f \text{ in } \Omega,\\
u &= 0 \text{ on } \partial\Omega.
\end{aligned} \tag{4.1}$$

Here $\Omega = [0,1]^3$, and these PDEs were discretized by the 7-point stencil. The discretized operation is equivalent to shifting the discretized Laplacian by a shift of $h^2\beta I$ for a mesh spacing of h.

4.1.1. Effect of s. In this subsection, we look into the effect of the number s of subdomains on the effectiveness of the PSLR preconditioner. We solve (4.1) with the shift of 0.05 on a 50^3 grid by the GMRES-PSLR method. The resulting coefficient matrix is indefinite (the largest and the smallest eigenvalues are 11.9386 and -0.0045, respectively). The number of terms used in the power series expansion is $m = 3$, and the rank for the low-rank correction terms is fixed at 15.

We can see from Table 1 that the fill-factor from ILU/LU decompositions decreases monotonically, while the fill-factor from low-rank correction terms increases when s increases from 5 to 55. This is because the size of each B_i and C_i, $i = 1, 2, \ldots, s$, is smaller from a larger s, which reduces the storage and the computational cost for the ILU factorizations. A larger s also results in a larger Schur complement S, which implies that the matrix V_{r_k} has more rows. That is why the fill-factor from the low-rank correction terms increases when s becomes larger. size(S) denotes the

TABLE 1
The fill-factor, iteration counts, and CPU time for solving (4.1) *with shift* $= 0.05$ *on a* 50^3 *grid by the GMRES-PSLR method (*$m = 3$*). Here, the rank in the low-rank correction terms is* 15*, and the dropping threshold in the ILU factorizations is* 10^{-2}.

(ILU) s	size(S)	fill (ILU)	fill (low-rank)	fill (total)	its	o-t	p-t	i-t	t-t
5	11383	2.68	0.20	2.88	90	0.11	0.11	1.10	1.21
15	21454	2.45	0.37	2.83	89	0.10	0.13	0.64	0.77
25	28337	2.30	0.49	2.79	88	0.11	0.15	0.57	0.72
35	31547	2.23	0.55	2.78	86	0.10	0.16	0.54	0.70
45	34817	2.17	0.60	2.77	86	0.10	0.19	0.57	0.76
55	37846	2.11	0.66	2.77	86	0.10	0.20	0.59	0.79
(LU) s	size(S)	fill (ILU)	fill (low-rank)	fill (total)	its	o-t	p-t	i-t	t-t
5	11383	162.78	0.20	162.98	31	0.11	21.38	42.06	63.44
15	21454	84.67	0.37	85.04	54	0.10	9.78	11.98	21.76
25	28337	55.16	0.49	55.65	59	0.11	3.40	7.14	9.54
35	31547	38.23	0.55	38.78	62	0.10	3.57	5.68	9.25
45	34817	31.20	0.60	31.80	62	0.10	3.91	5.38	9.28
55	37846	25.04	0.66	25.70	62	0.10	3.72	6.01	9.73

size of the Schur complement S. Compared with the ILU case, we see that the times with the use of LU are much larger (although the iteration number can be reduced when LU is used). This illustrates that we should not take the dropping threshold to be small. More details about the effect of the dropping threshold will be presented in section 4.2.1. These experiments also illustrate the fact that the performance of the PSLR preconditioner does not vary much with the number of subdomains used.

4.1.2. Effect of m. The number of terms used in the power series expansion is also an important factor, as was previously discussed. We investigate this factor by solving the same problem as in section 4.1.1 with the rank used in the low-rank correction part being fixed at 15. For this test, the size for the Schur complement S is $31,547$, and the ordering time (o-t) is 0.10 second. The iteration counts and CPU times for different m's are given in Table 2. As can be observed, the iteration number decreases from 171 to 78 when m increases from 0 to 5. This can be attributed to the improved clustering of the spectrum of the preconditioned Schur complement as the number of terms used in the power series expansion increases. Meanwhile, the time to construct the PSLR preconditioner increases slightly. Since the iteration number is reduced considerably when m increases from 0 to some positive constant and then reduced slightly after that, we expect that the iteration time decreases first and then increases. This is verified by the numerical results in Table 2. As is seen from Table 2, the iteration time first goes down from 0.90 to 0.57 as m increases from 0 to 3 and then increases from 0.57 to 0.63 when m increases from 3 to 5. The total time has the same trend as that of the iteration time. The results in Table 2 are plotted in Figure 8. In the figure we can see that $m = 3$ is optimal for this test, in terms of CPU time. In general, there is a similar pattern, and m should not be taken too large for the sake of a better overall performance.

4.1.3. Effect of r_k. In this subsection, we consider the effect of the rank used in the low-rank correction terms on the PSLR preconditioner. Here we consider the same test problem used in the previous two subsections but with different r_k's. We observe from Table 3 that the iteration number decreases as r_k increases from 0 to 75. The fill-factor from ILU decompositions keeps the same value 2.44 since we fix the number of subdomains. On the other hand, the fill-factor from the low-rank terms and the time to compute low-rank correction terms increase as the rank becomes larger.

TABLE 2
Iteration counts and CPU times for solving (4.1) *with shift* $= 0.05$ *on a* 50^3 *grid by the GMRES-PSLR method, in which* $s = 35$*, the dropping threshold in the ILU factorizations is* 10^{-2}*, and the rank in the low-rank correction part is* 15.

m	its	p-t	i-t	t-t
0	171	0.11	0.90	1.01
1	109	0.12	0.61	0.73
2	96	0.13	0.59	0.72
3	86	0.14	0.57	0.71
4	81	0.16	0.60	0.76
5	78	0.18	0.63	0.81

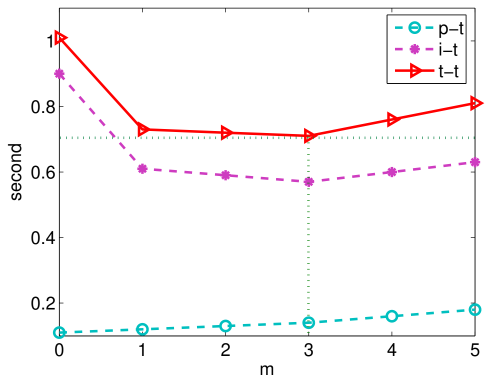

FIG. 8. *The preconditioner construction time, the iteration time, and the total time for solving* (4.1) *with shift* $= 0.05$ *on a* 50^3 *grid with different* m*'s by the GMRES-PSLR method.*

TABLE 3
The fill-factor, iteration counts, and CPU time for solving (4.1) *with shift* $= 0.05$ *on a* 50^3 *grid by the GMRES-PSLR method with* $m = 3$*, in which* $s = 35$ *and the dropping threshold in the ILU factorizations is* 10^{-2}.

r_k	fill (ILU)	fill (low-rank)	its	p-t	i-t	t-t
0	2.24	0.00	92	0.08	0.81	0.89
15	2.24	0.55	86	0.13	0.57	0.70
30	2.24	1.10	83	0.19	0.59	0.78
45	2.24	1.65	80	0.31	0.61	0.92
60	2.24	2.20	78	0.33	0.60	0.93
75	2.24	2.75	75	0.39	0.60	0.99

In the meantime, the iteration time and the total CPU time decrease as r_k increases from 0 to 15 and then increase. This indicates that there is no need to take a very large rank in practice.

In addition, Table 4 shows the benefit of incorporating low-rank corrections in the PSLR preconditioner when solving highly indefinite linear systems. Note that the PSLR preconditioner reduces to the Neumann polynomial preconditioner when the rank r_k is equal to zero. Results in Table 4 show that the low-rank correction technique can greatly improve the performance and robustness of the classical Neumann polynomial preconditioner even when the rank r_k is smaller than the number of the eigenvalues of E_{rr} with modulus greater than 1. For example, the GMRES-PSLR combination (full GMRES is used) fails to converge when there is no low-rank correction applied. Here, the shift for the grid 50^3 is set to 0.14, in which case the

Table 4

The fill-factor, iteration counts, and CPU time for solving (4.1) *with shift* $= 0.14$ *on a* 50^3 *grid by the GMRES-PSLR method with* $m = 3$*, in which* $s = 35$ *and the dropping threshold in the ILU factorizations is* 10^{-2}.

r_k	fill (ILU)	fill (low-rank)	its	t-t
0	3.07	0.00	F	–
15	3.07	0.55	346	8.90
30	3.07	1.10	310	8.06
45	3.07	1.65	266	7.01
60	3.07	2.20	220	5.65
75	3.07	2.75	199	5.69

Table 5

Execution time as a function of the number of threads for solving (4.1) *with shift* $= 0.05$ *on a* 50^3 *grid by the GMRES-PSLR method with* $m = 3$*, in which* $s = 35$*,* $r_k = 15$*, and the dropping threshold in the ILU factorizations is* 10^{-2}.

Threads	p-t	i-t	t-t
1	0.56	4.06	4.62
2	0.32	2.29	2.61
4	0.18	1.35	1.53
8	0.17	1.11	1.27
16	0.15	0.92	1.06
24	0.14	0.56	0.70

shifted discretized operator has 78 negative eigenvalues (the largest and the smallest eigenvalues are 11.8486 and -0.0040, respectively).

4.1.4. Effect of the number of threads. We now examine the effect of the number of threads on the performance when parallelization is achieved through OpenMP. Table 5 shows the total execution time as the number of threads increases from 4 to 24, when solving problem (4.1) with shift $= 0.05$ on a 50^3 grid. The rank here is taken as $r_k = 15$. As one can see from Table 5, the total wall clock time decreases as the number of threads increases. For this case, the total fill-factor is 2.79 (2.24 for ILU and 0.55 for the low-rank part), and the iteration number is 86 (regardless of the number of threads). As expected, the execution time for GMRES-PSLR is reduced when more threads are used, due to parallelism. So, the number of threads used in our numerical experiments is taken as the number of cores, i.e., 24.

4.1.5. Laplacian matrices. We now test some general 3D Laplacian matrices to show the efficiency of the PSLR preconditioner. We solve (4.1) with $\beta > 0$, where the corresponding problems are symmetric indefinite. For these problems, the discretized Laplacian was shifted by $h^2\beta I$ for mesh size h. The numbers of negative eigenvalues are $20, 69, 133$ for grids $32^3, 64^3$, and 128^3, respectively. Here, we set $m = 3$, $r_k = 15$, and $s = 35$ in the PSLR preconditioner. As we see from Table 6, the PSLR preconditioner outperforms ILUT, SLR, and GMSLR preconditioners for solving the resulting indefinite problems.

This is because the iteration number and the construction time of the PSLR preconditioner are much lower than those used by the other three preconditioners. We found that the ILUT and GMSLR preconditioners cannot even converge when the mesh size is 128^3 and shift $= 0.03$, in which case the number of negative eigenvalues is 133.

To further show the effect of the power series expansion on preconditioning, we solve (4.1) with the shift of 0.16 on a 32^3 grid by the GMRES-PSLR method and

Table 6
Comparisons of PSLR *with $m = 3$, $r_k = 15$, and $s = 35$,* ILUT, SLR*, and* GMSLR *preconditioners for solving symmetric indefinite linear systems from the 3D shifted Laplacians* (4.1).

Mesh	Shift	ILUT				GMSLR						
		fill	its	p-t	i-t	lev	r_k	fill	its	o-t	p-t	i-t
32^3	0.16	2.80	109	0.03	0.93	7	16	2.75	106	0.03	0.08	0.53
64^3	0.08	2.86	341	0.29	25.72	10	16	2.89	315	0.22	0.93	18.57
128^3	0.03	3.15	F	2.16	–	13	16	3.17	F	0.51	5.32	–

Mesh	Shift	PSLR					SLR					
		fill	its	o-t	p-t	i-t	r_k	fill	its	o-t	p-t	i-t
32^3	0.16	2.76	97	0.02	0.03	0.22	32	2.75	134	0.03	0.73	0.76
64^3	0.08	2.85	288	0.15	0.26	5.42	32	2.87	346	0.15	0.20	20.54
128^3	0.03	3.15	318	0.42	3.45	26.62	32	3.16	F	0.04	6.12	–

Table 7
Demonstration of r_k as a function of m for solving symmetric indefinite linear systems from the 3D shifted Laplacians (4.1).

$m = 0$					$m = 1$					$m = 2$				
r_k	fill	p-t	i-t	t-t	r_k	fill	p-t	i-t	t-t	r_k	fill	p-t	i-t	t-t
290	19.70	0.64	0.28	0.92	100	7.49	0.12	0.22	0.34	52	5.14	0.08	0.21	0.29
$m = 3$					$m = 4$					$m = 5$				
r_k	fill	p-t	i-t	t-t	r_k	fill	p-t	i-t	t-t	r_k	fill	p-t	i-t	t-t
15	2.76	0.03	0.22	0.25	8	2.38	0.04	0.28	0.32	4	2.19	0.04	0.33	0.37

report the results in Table 7. Here, we fix the number of subdomains to 35 and vary the values of m and r_k to make the GMRES-PSLR method reach the same accuracy (1e-8) at 97 iterations. As we can see from Table 7, a bigger rank r_k is needed when the number of terms used in the power series expansion is small. That is, rank r_k is a monotonically decreasing function of m.

4.2. Test 2.

We consider the shifted convection-diffusion equation below:

$$-\triangle u - \gamma \cdot \nabla u - \beta u = f \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega, \tag{4.2}$$

which is a nonsymmetric problem. This equation is discretized by the standard 7-point stencil in three dimensions, where $\Omega = [0, 1]^3$ and $\gamma \in \mathbb{R}^3$.

4.2.1. Effect of the dropping threshold.

The dropping threshold used for the ILU decompositions to the matrices B_i and C_i ($i = 1, 2, \ldots, s$) is an important factor that influences the performance of the PSLR preconditioner. To illustrate its effect, we solve problem (4.2) with the shift $= 0.16$ and $\gamma = (0.1, 0.1, 0.1)$ on a 32^3 grid by using different thresholds. The corresponding results are presented in Table 8, where the fill-factor from the low-rank correction part is 0.83 for all the different thresholds. Since the rank in the low-rank correction terms is set to 15, we do not repeat those data in Table 8. If a smaller threshold is used, the matrix B and the block diagonal part C_0 of C will be more accurate approximations to C. This implies that the iteration

TABLE 8
The fill-factor, iteration counts, and CPU time for solving (4.2) *with* $\gamma = (0.1, 0.1, 0.1)$ *on a* 32^3 *grid by the GMRES-PSLR method with* $m = 3$*, where* $s = 35$ *and the rank used in the low-rank correction part is taken as* 15*.*

Threshold	fill (ILU)	its	p-t	i-t	t-t
0.1	2.11	97	0.03	0.26	0.29
0.05	2.21	92	0.04	0.24	0.28
0.01	2.92	88	0.05	0.22	0.27
0.005	3.53	83	0.06	0.22	0.28
0.001	4.58	73	0.08	0.21	0.29
0.0005	5.54	67	0.09	0.21	0.30
0.0001	7.80	66	0.10	0.22	0.32

TABLE 9
Comparisons of PSLR *with* $m = 3$, $r_k = 15$*, and* $s = 35$*,* ILUT *and* GMSLR *preconditioners for nonsymmetric indefinite linear systems from the discretized* 3*D shifted convection-diffusion equation* (4.2)*.*

Mesh	Shift	PSLR					ILUT				GMSLR						
		fill	its	o-t	p-t	i-t	fill	its	p-t	i-t	lev	r_k	fill	its	o-t	p-t	i-t
32^3	0.16	2.78	88	0.02	0.05	0.22	2.79	89	0.03	0.54	7	16	2.73	86	0.03	0.09	0.45
64^3	0.08	2.86	260	0.15	0.27	5.54	2.88	270	0.28	25.05	10	16	2.89	266	0.22	1.04	16.73
128^3	0.03	3.13	309	0.41	3.77	24.98	3.10	F	4.26	–	13	16	3.12	F	0.52	3.56	–

number decreases and the fill-factor from the ILU decompositions increases as the threshold becomes smaller. This is illustrated by the results in Table 8. Similar to the performance of the rank r_k used in the low-rank correction part in section 4.1.3, we should not take the dropping threshold to be small since a reduction of a few iterations is not offset by a higher cost in the preconditioner construction.

Now we present more tests to illustrate the efficiency of PSLR when solving shifted convection-diffusion equations. Here, γ is set to $(0.1, 0.1, 0.1)$, and the shift is taken as $0.16, 0.08, 0.03$ for grid $32^3, 64^3, 128^3$, respectively. Here, we fixed $m = 3$, $r_k = 15$, and $s = 35$ in the PSLR preconditioner. As is seen from Table 9, the PSLR preconditioner outperforms GMSLR and ILUT preconditioners. In addition, from the results associated with the 64^3 grid in Table 9, we can see that PSLR needs less time per iteration compared with other preconditioners. Again GMRES does not converge with the GMSLR and ILUT preconditioners for the case when the shift = 0.03 and the mesh size is 128^3.

4.3. Test 3. Next we test PSLR for some general sparse linear systems including symmetric and nonsymmetric ones to show that the method can work quite well for general systems. The test matrices are from SuiteSparse matrix collection [9], and Table 10 provides a brief description.

Numerical results are presented in Table 11. Here, we fixed $s = 35$, $m = 3$, and $r_k = 15$ in the PSLR preconditioner for all the experiments. From this table, we can see that the GMRES-PSLR method converges for all the test problems without tuning its parameters. Moreover, the iteration time is much less than that of MSLR, ICT, GMSLR, and ILUT preconditioners. The GMRES, accelerator failed to converge within 500 iterations when used in conjunction with the ICT and MSLR preconditioners for the CFD problem `cfd1`.

TABLE 10
Some details on the test matrices. size(S) in the last column is the size of the Schur complement in the PSLR preconditioner.

Matrix	Order	nnz	Symmetric	Description	size(S)
cfd1	70,656	1,825,580	yes	CFD problem	26,522
ecology1	1,000,000	4,996,000	yes	landscape ecology problem	325,020
ecology2	999,999	4,995,991	yes	landscape ecology problem	300,230
thermal1	82,654	574,458	yes	thermal problem	39,567
thermal2	1,228,045	8,580,313	yes	thermal problem	409,348
Dubcova3	146,689	3,636,643	yes	2D/3D problem	6,998
CurlCurl_4	2,380,515	26,515,867	yes	model reduction problem	893,505
nlpkkt240	27,993,600	760,648,352	yes	optimization Problem	6,998,400
CoupCons3D	416,800	17,277,420	no	structural problem	104,200
Atmosmodd	1,270,432	8,814,880	no	atmospheric model	508,173
Atmosmodl	1,489,752	10,319,760	no	atmospheric model	496,584
Cage14	1,505,785	27,130,349	no	directed weighted graph	376,448
Transport	1,602,111	23,500,731	no	structural problem	500,538
FullChip	2,987,012	26,621,983	no	circuit simulation problem	853.432
cage15	5,154,859	99,199,551	no	directed weighted graph	1,718,286
circuit5M	5,558,326	59,524,291	no	circuit simulation problem	1,600,450

TABLE 11
Comparisons of PSLR *with $m = 3$, $r_k = 15$, and $s = 35$,* ICT/ILUT *and* MSLR/GMSLR *preconditioners.*

Matrix	PSLR						ICT					MSLR							
	fill	its	o-t	p-t	i-t	t-t	fill	its	p-t	i-t	t-t	lev	r_k	fill	its	o-t	p-t	i-t	t-t
cfd1	3.15	245	0.12	0.51	3.92	**4.43**	3.14	F	11.48	–	–	7	180	3.15	F	0.20	10.90	–	–
ecology1	2.68	119	0.25	1.24	8.33	**9.57**	2.67	87	0.60	17.40	18.00	7	32	2.68	318	0.36	11.60	9.97	21.57
ecology2	2.68	107	0.25	1.25	9.57	**10.82**	2.67	402	0.61	26.87	27.48	8	35	2.67	399	0.35	10.40	15.30	25.70
thermal1	2.38	103	0.15	0.15	0.38	**0.53**	2.38	138	0.16	2.84	3.00	6	24	2.39	181	0.20	1.09	0.68	1.77
thermal2	2.44	156	0.30	2.06	12.17	**14.23**	2.45	317	2.56	26.66	29.22	8	32	2.46	497	0.38	20.90	22.80	43.70
Dubcova3	3.62	61	0.17	0.87	1.67	**2.54**	3.59	52	1.98	2.50	4.48	8	64	3.60	23	0.24	1.58	2.21	3.79
CurlCurl_4	6.89	135	0.43	0.50	24.51	**25.01**	6.89	479	4.01	55.78	59.79	9	64	6.88	399	0.54	0.70	38.21	38.91
nlpkkt240	7.78	180	1.15	3.87	89.61	**93.48**	7.78	487	10.01	140.20	150.21	13	64	7.77	221	1.30	5.01	101.10	106.11
Matrix	PSLR						ILUT					GMSLR							
	fill	its	o-t	p-t	i-t	t-t	fill	its	p-t	i-t	t-t	lev	r_k	fill	its	o-t	p-t	i-t	t-t
CoupCons3D	1.54	19	0.20	1.76	2.40	**4.16**	1.53	12	8.64	4.21	12.85	10	16	1.53	17	0.29	2.35	3.51	5.86
Atmosmodd	4.24	36	0.35	2.40	6.88	**9.28**	4.28	45	12.73	17.11	29.84	10	16	4.26	38	.47	4.00	15.78	19.78
Atmosmodl	4.65	18	0.40	4.04	10.46	**14.50**	4.66	27	8.87	19.09	27.96	11	16	4.62	25	0.51	5.33	16.22	21.55
cage14	2.13	4	0.42	4.13	5.79	**9.92**	2.11	6	6.95	10.18	17.13	6	4	2.13	38	0.51	5.73	8.89	14.62
Transport	2.67	99	0.48	5.27	19.72	**24.99**	2.67	100	24.38	40.94	65.32	11	16	2.66	53	0.60	6.09	31.94	38.03
FullChip	3.98	120	0.74	9.01	39.98	**48.99**	3.98	324	40.89	70.50	111.39	9	64	3.96	240	0.87	12.45	50.70	63.16
cage15	4.61	9	1.87	6.76	14.21	**20.97**	4.60	15	13.50	25.18	38.68	8	64	4.62	22	2.04	6.96	19.78	26.74
circuit5M	6.71	122	1.90	7.01	36.80	**43.81**	6.70	254	29.85	56.76	86.61	9	64	6.71	156	2.19	7.10	55.40	62.50

5. Conclusion. We have presented an effective Schur complement-based parallel preconditioner for solving general large sparse linear systems. The method utilizes a standard Schur complement viewpoint and exploits a power series expansion along with a low-rank correction technique to approximate the inverse of the Schur complement. The main difference between PSLR and other Schur complement techniques proposed earlier is that PSLR relies on the power series expansion to reduce the rank needed to obtain a good approximation of the inverse of the Schur complement. The number m of terms used in the power series expansion and the rank used in the low-rank correction part control the approximation accuracy of the preconditioner. In practice, small values for these two parameters are sufficient to yield a reasonably good approximation to S^{-1}.

As was illustrated in the experiments, a big advantage of PSLR is its high level of parallelism. Another advantage is its robustness when solving indefinite linear systems. Finally, PSLR is fairly easy to build and apply and is quite general. All that is required at the outset is a problem that is partitioned into subdomains. In our future work, we will develop a general-purpose distributed memory version of our current code.

Acknowledgments. We would like to thank both referees for their detailed comments and suggestions that led to considerable improvements in the quality of the present manuscript. In addition, the authors would like to thank Professor Zhongxiao Jia for his feedback on an early draft of this manuscript.

REFERENCES

[1] P. Amestoy, C. Ashcraft, O. Boiteau, A. Buttari, J. Y. L'Excellent, and C. Weisbecker, *Improving multifrontal methods by means of block low-rank representations*, SIAM J. Sci. Comput., 37 (2015), pp. A1451–A1474.

[2] P. Amestoy, A. Buttari, J. Y. L'Excellent, and T. Mary, *Performance and scalability of the block low-rank multifrontal factorization on multicore architectures*, ACM Trans. Math. Software, 45 (2019), pp. 1–26.

[3] A. Aminfar, S. Ambikasaran, and E. Darve, *A fast block low-rank dense solver with applications to finite-element matrices*, J. Comput. Phys., 304 (2016), pp. 170–188.

[4] M. Bebendorf, *Hierarchical Matrices: A Means to Efficiently Solve Elliptic Boundary Value Problems*, Lect. Notes Comput. Sci. Eng., Springer, Berlin, 2008.

[5] M. Benzi and M. Tuma, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.

[6] S. L. Borne and L. Grasedyck, *$\mathcal{H}$-matrix preconditioners in convection-dominated problems*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1172–1183.

[7] D. Cai, E. Chow, L. Erlandson, Y. Saad, and Y. Xi, *SMASH: Structured matrix approximation by separation and hierarchy*, Numer. Linear Algebra Appl., 25 (2018), e2204.

[8] E. Chow and Y. Saad, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.

[9] T. A. Davis and Y. Hu, *The University of Florida sparse matrix collection*, ACM Trans. Math. Software, 38 (2011) 1.

[10] G. Dillon, V. Kalantzis, Y. Xi, and Y. Saad, *A hierarchical low-rank Schur complement preconditioner for indefinite linear systems*, SIAM J. Sci. Comput., 40 (2018), pp. A2234–A2252.

[11] A. Franceschini, V. A. P. Magri, M. Ferronato, and C. Janna, *A robust multilevel approximate inverse preconditioner for symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 123–147.

[12] P. Ghysels, X. S. Li, F. H. Rouet, S. Williams, and A. Napov, *An efficient multicore implementation of a novel HSS-structured multifrontal solver using randomized sampling*, SIAM J. Sci. Comput., 38 (2016), pp. S358–S384.

[13] A. Gillman, P. Young, and P. G. Martinsson, *A direct solver with $\mathcal{O}(n)$ complexity for integral equations on one-dimensional domains*, Front. Math. China, 7 (2012), pp. 217–247.

[14] M. Grote and T. Huckle, *Parallel preconditioning with sparse approximate inverses*, SIAM J. Sci. Comput., 18 (1997), pp. 838–853.

[15] W. Hackbusch and S. Börm, *$\mathcal{H}^2$-matrix approximation of integral operators by interpolation*, Appl. Numer. Math., 43 (2002), pp. 129–143.

[16] J. C. Haws, M. Benzi, and M. Tuma, *Preconditioning highly indefinite and nonsymmetric matrices*, SIAM J. Sci. Comput., 22 (2000), pp. 1333–1353.

[17] N. J. Higham and T. Mary, *A new preconditioner that exploits low-rank approximations to factorization error*, SIAM J. Sci. Comput., 41 (2019), pp. A59–A82.

[18] Z. Jia and W. J. Kang, *A residual based sparse approximate inverse preconditioning procedure for large sparse linear systems*, Numer. Linear Algebra Appl., 24 (2017), e2080.

[19] G. Karypis and V. Kumar, *A fast and high quality multilevel scheme for partitioning irregular graphs*, SIAM J. Sci. Comput., 20 (1998), pp. 359–392.

[20] R. Li, Y. Xi, and Y. Saad, *Schur complement-based domain decomposition preconditioners with low-rank corrections*, Numer. Linear Algebra Appl., 23 (2016), pp. 706–729.

[21] X. Liu, Y. Xi, Y. Saad, and M. V. de Hoop, *Solving the three-dimensional high-frequency Helmholtz equation using contour integration and polynomial preconditioning*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 58–82.

[22] X. Liu, J. Xia, and M. V. de Hoop, *Parallel randomized and matrix-free direct solvers for large structured dense linear systems*, SIAM J. Sci. Comput., 38 (2016), pp. S508–S538.

[23] P. G. Martinsson and V. Rokhlin, *A fast direct solver for boundary integral equations in two dimensions*, J. Comput. Phys., 205 (2005), pp. 1–23.

[24] G. Pichon, E. Darve, M. Faverge, P. Ramet, and J. Roman, *Sparse supernodal solver using block low-rank compression: Design, performance and analysis*, J. Comput. Sci., 27 (2018), pp. 255–270.

[25] Y. Saad, *ILUT: A dual threshold incomplete ILU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[26] Y. Saad, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelpha, PA, 2003.

[27] Y. Saad and B. Suchomel, *ARMS: An algebraic recursive multilevel solver for general sparse linear systems*, Numer. Linear Algebra Appl., 9 (2002), pp. 359–378.

[28] Y. Xi, R. Li, and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.

[29] Y. Xi and Y. Saad, *A rational function preconditioner for indefinite sparse linear systems*, SIAM J. Sci. Comput., 39 (2017), pp. A1145–A1167.

[30] Y. Xi and J. Xia, *On the stability of some hierarchical rank structured matrix alogrithms*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 1279–1303.

[31] J. Xia, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.

[32] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Superfast multifrontal method for large structured linear systems of equations*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 1382–1411.

[33] J. Xia, Y. Xi, S. Cauley, and V. Balakrishnan, *Fast sparse selected inversion*, SIAM J. Matrix Anal. Appl., 36 (2015), pp. 1283–1314.

[34] J. Xia and Z. Xin, *Effective and robust preconditioning of general spd matrices via structured incomplete factorization*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1298–1322.