

## AN ALGEBRAIC MULTILEVEL PRECONDITIONER WITH LOW-RANK CORRECTIONS FOR SPARSE SYMMETRIC MATRICES\*

YUANZHE XI<sup>†</sup>, RUIPENG LI<sup>‡</sup>, AND YOUSEF SAAD<sup>†</sup>

**Abstract.** This paper describes a multilevel preconditioning technique for solving sparse symmetric linear systems of equations. This “Multilevel Schur Low-Rank” (MSLR) preconditioner first builds a tree structure  $\mathcal{T}$  based on a hierarchical decomposition of the matrix and then computes an approximate inverse of the original matrix level by level. Unlike classical direct solvers, the construction of the MSLR preconditioner follows a top-down traversal of  $\mathcal{T}$  and exploits a low-rank property that is satisfied by the difference between the inverses of the local Schur complements and specific blocks of the original matrix. A few steps of the generalized Lanczos tridiagonalization procedure are applied to capture most of this difference. Numerical results are reported to illustrate the efficiency and robustness of the MSLR preconditioner with both two- and three-dimensional discretized PDE problems and with publicly available test problems.

**Key words.** low-rank approximation, Schur complements, multilevel preconditioner, domain decomposition, incomplete factorization, Krylov subspace methods, Nested Dissection ordering

**AMS subject classifications.** 65F08, 65N22, 65N55, 65Y05, 65Y20

**DOI.** 10.1137/15M1021830

**1. Introduction.** In this paper, we consider iterative methods for solving large symmetric sparse linear systems

$$(1.1) \quad Ax = b,$$

where  $A \in \mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$ . Krylov subspace methods belong to a class of general-purpose techniques that are usually combined with a preconditioner to solve the above system. Preconditioning consists of modifying the original system into, for example, the left-preconditioned system  $M^{-1}Ax = M^{-1}b$ . The preconditioner  $M$  is an approximation to  $A$  such that solving linear systems with it is relatively inexpensive. Often, the preconditioning matrix  $M$  is built from an Incomplete LU (ILU) factorization extracted from a form of approximate Gaussian elimination process in which “fill-ins,” i.e., nonzero entries appearing in zero locations in the original matrix, are removed. ILU preconditioners are fairly easy to implement and robust for a large class of problems.

The success of these ILU methods for solving certain systems arising from PDEs has been widely illustrated in the literature. At the same time it has also become clear in recent years that they fall short of providing robust and scalable solvers

---

\*Received by the editors May 18, 2015; accepted for publication (in revised form) by L. Giraud December 10, 2015; published electronically March 3, 2016. This work was supported by the NSF under grants DMS-1216366 and DMS-1521573 and by the Minnesota Supercomputing Institute. The U.S. Government retains a nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes. Copyright is owned by SIAM to the extent not limited by these rights.

<http://www.siam.org/journals/simax/37-1/M102183.html>

<sup>†</sup>Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN 55455-0154 (yxi@cs.umn.edu, saad@cs.umn.edu).

<sup>‡</sup>Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA 94551 (li50@llnl.gov). This author’s work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344 (LLNL-JRNL-680317).

for handling harder problems. The situation has been accentuated by the increased complexity of linear systems on the one hand and the parallelization requirements on the other. The preconditioners developed so far in the literature often face one or both of the following issues. First, they fail for highly indefinite systems because in these situations they will either encounter some small/zero pivots during the ILU factorization process or produce unstable triangular factors [5, 32]. Second, their performance in the new breed of highly parallel architectures, such as computers equipped with GPUs [23] or Intel Xeon Phi processors, is unacceptably poor due to the sequential nature of the construction and the application of the preconditioner. Approximate inverse preconditioners were developed in the 1990s as an alternative to ILUs [3, 6, 7] to address these issues, but their success has been limited because they rely on the inverse of  $A$  being well approximated by a sparse matrix. In the end, when considering construction cost, storage requirement, and number of iterations needed for convergence by these techniques, users found them noncompetitive for solving systems with general matrices.

Both of the issues just mentioned can be addressed by low-rank approximation techniques. To begin with, several rank structured representations of matrices have been developed for the fast solutions of structured systems, PDEs, integral equations, eigenvalue problems and for building preconditioners [8, 20, 21, 27, 43, 44, 48, 49]. They include the work on  $\mathcal{H}$ -matrices [14, 15] and hierarchically semiseparable (HSS) matrices [43, 47]. These matrix representations partition the given matrix into appropriate blocks and approximate certain off-diagonal blocks with low-rank matrices. They have also been integrated into sparse matrix techniques to speed up the intermediate dense matrix operations and provided a new class of sparse direct solvers [18, 46] and preconditioners [11, 45]. Although these methods have lower asymptotic complexity bounds than standard methods, the prefactors in these bounds are large, and the methods generally require much more sophisticated implementations. They often also require some information on the underlying physical structure [11].

More recently, a few articles have focused on a new class of approximate inverse preconditioners. Within the domain decomposition framework, these preconditioners seek data sparsity rather than the standard nonzero sparsity in the inverse matrices, and approximate these inverses with various low-rank correction techniques. The starting point is the Multilevel Low-Rank (MLR) preconditioner proposed in [22]. The Domain Decomposition based Low-Rank (DD-LR) preconditioner [24] is a variant geared towards distributed sparse matrices on massively distributed memory computers. The Schur complement based Low-Rank (SLR) preconditioner [25] further extends this framework to provide preconditioners designed for classical Schur complement methods. In general, these low-rank correction based approximate inverse preconditioners reach a good balance between robustness and efficiency and thus offer promising alternatives to the ILU-type preconditioners on modern high performance computers.

In this paper, we generalize the SLR preconditioner introduced in [25] and propose a Multilevel Schur complement based Low-Rank (MSLR) preconditioner. Its main advantages are outlined below.

(1) *Hierarchical interface decomposition (HID) ordering for general sparse matrices.* In contrast to the SLR preconditioner, where the adjacency graph of the matrix is only partitioned into subdomains and interface points, the MSLR preconditioner applies the HID ordering [16] to exploit the hierarchy among the interface points such that a block independent set structure [35] appears at each level of the HID tree  $\mathcal{T}$ . The MSLR preconditioner construction algorithm can then fully take advantage of

this structure. That is, all the diagonal blocks in the reordered matrix can be factored simultaneously. The HID ordering can be obtained from graph partitioning tools and is applicable to both two-dimensional (2D) and three-dimensional (3D) discretized PDEs as well as general sparse matrices.

(2) *Enhanced efficiency and robustness to indefiniteness.* After the HID ordering is obtained and applied, the MSLR preconditioner follows a top-down traversal of  $\mathcal{T}$  and constructs an approximate inverse by exploiting a low-rank property in the Schur complement inverse at each level of  $\mathcal{T}$ . This construction procedure traverses the tree  $\mathcal{T}$  in an order that is the reverse of that used in direct methods and significantly reduces fill-ins, especially for large 3D problems. We prove that, for some 2D/3D model problems, the storage for the MSLR preconditioner is nearly  $O(n)$ . In addition, the MSLR preconditioner gains in efficiency in the application phase since irregular computations, such as the backward and forward substitutions, are limited to the diagonal blocks where these operations can be performed independently for the blocks at the same level of  $\mathcal{T}$ . The remaining operations involve only the sparse/dense matrix-vector products. Extensive numerical experiments further show that the MSLR preconditioner is not as sensitive to indefinite systems as the classical ILU-type preconditioners.

(3) *Easy update.* The quality of the MSLR preconditioner is highly dependent on the rank used in the low-rank corrections. These low-rank corrections are computed by the generalized Lanczos algorithm. When the current preconditioner does not achieve good performance, it can be quickly updated by simply computing more eigenvectors and adding them into the constructed low-rank correction terms.

We would like to mention that there are similarities between the MSLR preconditioner proposed in this paper and the *hierarchical Poincaré–Steklov scheme* in [13] as well as the *BDDC preconditioner* in [26, 39]. Although the last two are not purely algebraic methods, they also exploit a hierarchy of separators to reduce the overall computational cost. Other connections exist between the MSLR preconditioner and other classes of domain decomposition methods, such as optimized Schwarz methods [40] and optimized 2-level Schwarz methods [10, 41]. These enhanced Schwarz methods use coarse grid correction ideas to improve the convergence rate of the classical Schwarz iteration when the number of subdomains increases. The low-rank corrections used in the MSLR preconditioner can be viewed as a means to incorporate “global” couplings between subdomains in a way that is similar to these enhanced Schwarz methods. Therefore, it may be possible to utilize the convergence theory available from these related studies to analyze the MSLR preconditioner in some specific cases.

The remaining sections are organized as follows. Section 2 reviews the background and motivation of the MSLR preconditioner. An HID ordering is introduced in section 3. We study the low-rank property associated with a sequence of local Schur complement inverses in section 4 and propose the MSLR preconditioner in section 5. The complexity of the MSLR preconditioner is analyzed in section 6. Numerical examples with some 2D and 3D PDEs as well as general sparse matrices are reported in section 7, and a conclusion is drawn in section 8. In the presentation, we use the following notation:

- $[W, \Sigma] = \text{eigs}(A, k)$  denotes the application of the Lanczos algorithm to compute the  $k$  largest eigenpairs of  $A$ , where the columns of  $W$  contain the desired eigenvectors and the diagonal entries of  $\Sigma$  are the  $k$  largest eigenvalues of  $A$ .
- $\text{diag}(\sigma_i)$  represents a diagonal matrix with  $\sigma_i$  along its diagonal.
- $|A|$  denotes the dimension of  $A$  for a square matrix.

**2. Background and motivation.** Assume  $A$  is a Symmetric Positive Definite (SPD) matrix, and let it be partitioned into  $2 \times 2$  block form as

$$(2.1) \quad A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix},$$

where  $C \in \mathbb{R}^{s \times s}$ . We define the Schur complement matrix  $S = C - E^T B^{-1} E$ . Since  $A$  is SPD,  $C$  is also SPD and admits a Cholesky factorization

$$C = LL^T.$$

Define the matrix

$$(2.2) \quad G = L^{-1} E^T B^{-1} E L^{-T} = L^{-1} (C - S) L^{-T},$$

and let  $\sigma_i$ ,  $i = 1, \dots, s$ , be its eigenvalues labeled in decreasing order. It can be easily shown that [25]

$$0 \leq \sigma_s \leq \sigma_{s-1} \leq \dots \leq \sigma_1 < 1.$$

We are interested in the separation of the eigenvalues of  $S^{-1} - C^{-1}$  or those of  $X \equiv L^T (S^{-1} - C^{-1}) L$ . The second is easier to analyze. The following result explains the observed “decay” property of the eigenvalues of  $X$ .

**PROPOSITION 2.1.** *The eigenvalues  $\theta_s \leq \theta_{s-1} \leq \dots \leq \theta_1$  of  $X$  are related to those of  $G = L^{-1} (C - S) L^{-T}$  by*

$$(2.3) \quad \theta_i = \frac{\sigma_i}{1 - \sigma_i}, \quad i = 1, \dots, s.$$

*In particular the gaps  $\theta_i - \theta_{i+1}$  between two consecutive eigenvalues of  $X$  are given by*

$$(2.4) \quad \theta_i - \theta_{i+1} = \frac{\sigma_i - \sigma_{i+1}}{(1 - \sigma_{i+1})(1 - \sigma_i)}, \quad i = 1, \dots, s - 1.$$

*These gaps are bigger than the gaps  $\sigma_i - \sigma_{i+1}$  associated with the matrix  $G$ , and they are much bigger for those eigenvalues  $\sigma_i$  close to 1, i.e., for the largest ones.*

*Proof.* We have

$$S = L(I - L^{-1} E^T B^{-1} E L^{-T}) L^T = L(I - G) L^T.$$

From the above expression and the Cholesky factorization of  $C$  we have

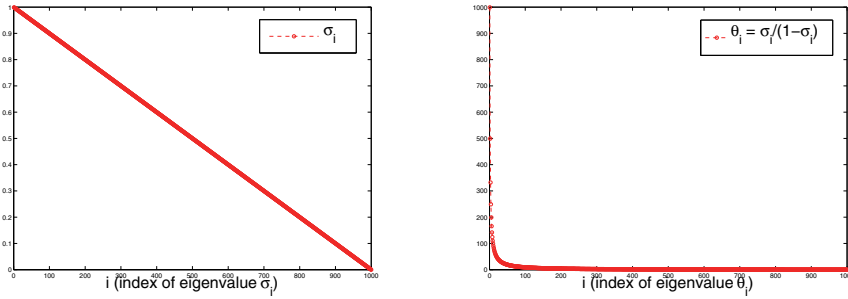
$$S^{-1} - C^{-1} = L^{-T} [(I - G)^{-1} - I] L^{-1} = L^{-T} [G(I - G)^{-1}] L^{-1}.$$

Therefore,  $X = L^T (S^{-1} - C^{-1}) L = G(I - G)^{-1}$  and so the eigenvalues of  $X$  and those of  $G$  are related by (2.3). The gap formula (2.4) results from a straightforward calculation.  $\square$

Proposition 2.1 establishes a relation between the eigenvalues of  $L^{-1} (C - S) L^{-T}$  and  $L^T (S^{-1} - C^{-1}) L$ . It shows that as long as the largest  $\sigma_i$ 's do not cluster around 1, the largest  $\theta_i$ 's will be very well separated in general.

We were initially interested in the difference between  $S^{-1}$  and  $C^{-1}$ , which is equal to  $L^{-T} X L^{-1}$ . First we observe that  $X$  has the same eigenvalues as the matrix  $(S^{-1} - C^{-1}) C$ :

$$(2.5) \quad \lambda_i((S^{-1} - C^{-1}) C) = \theta_i, \quad i = 1, \dots, s,$$



(a)  $\sigma_i$  decays linearly between  $[0, 0.999]$ . (b)  $\theta_i = \frac{\sigma_i}{1-\sigma_i}$  decays between  $[0, 1000]$ .

FIG. 2.1. Example: Eigenvalue decay rate comparison between  $\sigma_i$  and  $\sigma_i/(1-\sigma_i)$ . Here we assume  $\sigma_i$ 's decay linearly from 0.999 to 0 for 1000 points.

where  $\lambda_i(\cdot)$  denotes the  $i$ th largest eigenvalue. Based on the eigenvalue inequality (Lidskii's product inequalities and their further developments [28]) and (2.5), we know that

$$\theta_i \lambda_s(C^{-1}) \leq \lambda_i(S^{-1} - C^{-1}) \leq \theta_i \lambda_1(C^{-1}).$$

Assume that

$$(2.6) \quad \theta_{k+1} \leq \tau \theta_1 \quad \text{and} \quad \lambda_1(C^{-1}) = c \lambda_s(C^{-1})$$

for some constants  $\tau$  and  $c$ ; then the following inequalities hold:

$$(2.7) \quad \lambda_{k+1}(S^{-1} - C^{-1}) \leq \tau \theta_1 \lambda_1(C^{-1}) = c \tau \theta_1 \lambda_s(C^{-1}) \leq c \tau \lambda_1(S^{-1} - C^{-1}).$$

Therefore, if the gap between  $\theta_1$  and  $\theta_{k+1}$  is relatively big, which means  $\tau$  in (2.6) is small, then the gap between  $\lambda_1(S^{-1} - C^{-1})$  and  $\lambda_{k+1}(S^{-1} - C^{-1})$  will be preserved except that it will be damped to a certain degree by the condition number of  $C$ . In many applications when the matrix  $A$  originates from a discretized PDE and the matrix  $C$  is associated with the interface unknowns in the domain decomposition framework,  $C$  is usually strongly diagonally dominant and has a small condition number. In these cases,  $S^{-1} - C^{-1}$  inherits the decay property of  $X$ .

The above analysis can be generalized to the case when  $B$  is indefinite and  $C$  is SPD in (2.1). In this situation the eigenvalues  $\sigma_i$  of  $G$  in (2.2) are no longer bounded above by 1, and the gaps  $|\theta_i - \theta_{i+1}|$  are much bigger than the gaps  $\sigma_i - \sigma_{i+1}$  for those  $\sigma_i$  around 1 but not necessarily the largest ones when compared with the SPD case in Proposition 2.1. Then we can apply the singular value inequality (Theorem 9 in [28]) and prove a result similar to that in (2.7) with the eigenvalues  $\lambda_i$  and  $\theta_i$  replaced by the corresponding singular values. Thus,  $S^{-1}$  can still be approximated by  $C^{-1}$  plus a low-rank matrix in this case.

It is not easy to measure the decay rate of  $S^{-1} - C^{-1}$  for a general matrix. It was explicitly analyzed in [25] for a 2D Laplacian on a regular grid for the two-domain case. It is also possible to illustrate Proposition 2.1 with a simple example where 1000 eigenvalues  $\sigma_i$  are assumed to decay linearly between  $(0, 0.999)$ ; see Figure 2.1(a). The values of  $\theta_i = \sigma_i/(1-\sigma_i)$ , which are also plotted in Figure 2.1(b) for comparison, decay much faster than the  $\sigma_i$ 's. Thus, when  $k = 11$ ,  $\sigma_k$  is 99% as large as  $\sigma_1$ , while  $\theta_k$  is only 9% of  $\theta_1$ . This means that  $S^{-1}$  can be well approximated by  $C^{-1}$  plus a rank-10 matrix in this case. In contrast,  $C - S$  cannot be approximated

by a low-rank matrix. This property is also demonstrated through some examples in section 4.

The SLR preconditioner proposed in [25] exploits the result of Proposition 2.1 in a domain decomposition framework. The idea is to first partition the adjacency graph of the input matrix into  $p$  disjoint subdomains. The matrix  $B$  in (2.1) corresponds to the interior points of all subdomains and is in a block-diagonal form. The matrices  $B$  and  $C$  are factored using, e.g., the incomplete Cholesky factorization, so we have  $B \approx L_B L_B^T$  and  $C \approx L_C L_C^T$ . The SLR preconditioner  $M$  then has the following form:

$$M = \begin{pmatrix} L_B & \\ E^T L_B^{-T} & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{S} \end{pmatrix} \begin{pmatrix} L_B^T & L_B^{-1} E \\ & I \end{pmatrix},$$

where  $\tilde{S}^{-1} = L_C^{-T} (I + W T W^T) L_C^{-1}$  and  $L_C^{-T} W T W^T L_C^{-1}$  is a low-rank approximation of  $S^{-1} - C^{-1}$  obtained from approximating  $L_C^{-1} E^T B^{-1} E L_C^{-T}$  using the Lanczos algorithm.

The numerical tests reported in [25] indicated that the SLR preconditioner is quite robust, making it an attractive alternative to ILU-type preconditioners for certain types of problems. However, its efficiency deteriorates for large 3D problems mainly because the number of the interface points can be relatively big for those problems (see Figure 4.3 for an illustration) and a direct factorization of the matrix  $C$  in (2.1) becomes costly. At the same time, in order to control the cost associated with factoring the matrix  $B$  in (2.1), a large number of subdomains is required for those problems. This in turn leads to a larger number of interface unknowns. As a result, SLR scales poorly for large 3D problems, and this provides the main motivation for exploring an extension into a multilevel scheme.

**3. HID ordering.** An interesting class of domain decomposition methods, which exploits a hierarchy of “interfaces,” is based on so-called *wirebasket* orderings [37, 38]. These techniques take advantage of cross points in the partitioned mesh to derive preconditioners with good convergence properties. In [16] a similar idea is introduced for general sparse matrices not necessarily originating from PDEs. This paper defines a so-called *Hierarchical Interface Decomposition (HID)* ordering for a general graph through “connectors.” In an HID ordering with  $L$  levels, connectors of level  $l$  are a set of subsets of vertices that are disjoint with other connectors on the same level and separated in the original graph by connectors of higher levels. When labeled by levels, the resulting matrix will have a block-diagonal structure for each level, and very efficient parallel preconditioners based on parallel Gaussian elimination can be developed. However, we will not seek to develop ILU-type preconditioners, but rather to incorporate low-rank approximations in this context as is explained in the following sections.

An HID ordering can be obtained in a number of ways; see [16] for an example from a standard graph partitioning. They can also be obtained from the standard Nested Dissection (ND) algorithm [12]. Let  $\mathcal{G} = (V, E)$  be the adjacency graph associated with a symmetric sparse matrix  $A$ . The main idea of the ND ordering is to recursively bi-partition the graph using *vertex separators*. Here, a vertex separator is a small set of nodes in  $V$  whose removal separates the graph into two disjoint subgraphs. Suppose we have  $L$  levels. The first bisection produces two subsets and a separator. At this level ( $L = 3$ ) we will have one connector, and it is just the separator. This set is the line of vertices labeled 15 in Figure 3.1(a). Descending to the next level ( $L - 1 = 2$ ), we now have two connectors consisting of the two separators of each of the resulting subsets. These are labeled 13 and 14 in Figure 3.1(a). The separators produced at

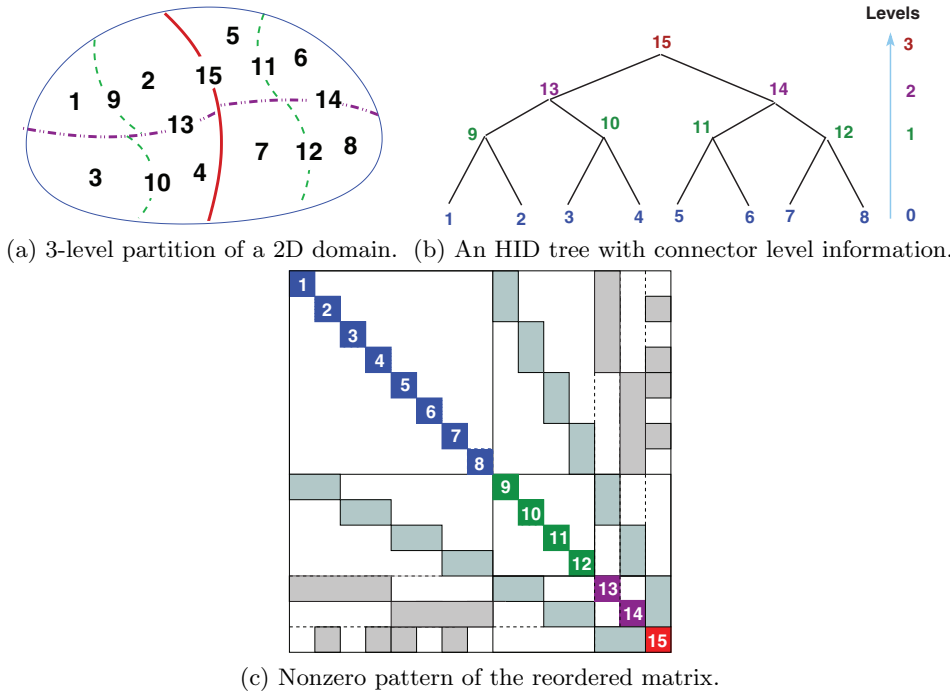


FIG. 3.1. An illustration of the HID ordering for a 2D problem.

step  $i$  of the ND ordering ( $i = 1, \dots, L - 1$ ) are the connectors of level  $L - i + 1$ . The zeroth level consists of the subsets of points interior to each subdomain. The level information of these connectors can be represented by an HID tree  $\mathcal{T}$ . See Figure 3.1(b) for an example. The matrix is ordered by level, starting with nodes of level 0 and ending with nodes of level  $L$ . Figure 3.1(c) shows the nonzero pattern for the mesh in Figure 3.1(a) when this ordering is used.

The reordered matrix with the HID ordering has the following multilevel recursive form:

$$(3.1) \quad A_l = \begin{pmatrix} B_l & E_l \\ E_l^T & C_l \end{pmatrix} \quad \text{and} \quad C_l = A_{l+1} \quad \text{for} \quad l = 0 : L - 1,$$

where  $A_0$  is the reordered matrix of  $A$  from the HID ordering and  $A_L$  represents the submatrix associated with the top-level connector. Each leading block  $B_l$  in  $A_l$  has the desired block-diagonal structure resulting from the block independent set [35] nature of this ordering. In the example of Figure 3.1(c),  $B_0$ ,  $B_1$ , and  $B_2$  correspond to the matrices containing the blocks labeled 1–8, 9–12, and 13–14, respectively.

The HID ordering forms the block independent sets in a preprocessing stage. In contrast, the methods in [4, 31, 35, 36] determine these block independent sets dynamically since each  $A_{l+1}$ , which is an approximate Schur complement matrix at level  $l$ , is available only after the factorization of  $A_l$ . Once the block independent set at each level  $l$  is determined, classical multilevel algorithms [31, 35, 36] explore different strategies to approximate the following factorization of  $A_l$ :

$$(3.2) \quad A_l = \begin{pmatrix} I & \\ E_l^T B_l^{-1} & I \end{pmatrix} \begin{pmatrix} B_l & \\ & S_l \end{pmatrix} \begin{pmatrix} I & B_l^{-1} E_l \\ & I \end{pmatrix},$$

where  $S_l = C_l - E_l^T B_l^{-1} E_l$  is the Schur complement. The next section discusses the low-rank property of  $S_l^{-1} - C_l^{-1}$  for each  $l$  and derives an efficient multilevel preconditioner based on this property.

**4. Low-rank property of  $S_l^{-1} - C_l^{-1}$ .** In this section, we examine the rank of the matrix  $S_l^{-1} - C_l^{-1}$  obtained from the HID ordering for some model problems. We start our analysis from the exact rank of  $S_l^{-1} - C_l^{-1}$ . First we need to define the *cross points* in the HID ordering since these points are closely related to the rank of  $S_l^{-1} - C_l^{-1}$ . Similar definitions can also be found in [16, 38].

**DEFINITION 4.1.** *Cross points at level  $l$  are subsets of connectors of level  $l$  that intersect with connectors of higher levels.*

In Figure 3.1(a), for example, the cross points of level 2 are those points at the intersection of the lines labeled 13 and 14 with the line labeled 15. Similarly, cross points of level 1 are those points at the intersection of the lines labeled 9, 10, 11, 12 with the lines labeled 13, 14. The relation between the cross points and the rank of  $S_l^{-1} - C_l^{-1}$  is shown in the following lemma.

**LEMMA 4.2.** *The exact rank of  $S_l^{-1} - C_l^{-1}$  is bounded by the number of the cross points at level  $l$ .*

*Proof.* Since  $E_l$  in (3.1) represents the couplings between the connectors at level  $l$  and upper levels  $k$  for  $k > l$ , the rank of  $E_l$  should equal the number of cross points at level  $l$ , which is generally smaller than the dimension of  $C_l$ . This means  $E_l$  admits a low-rank representation

$$E_l = U_l V_l^T,$$

where the column size of  $V_l$  equals the number of cross points at level  $l$ . Thus,

$$S_l = C_l - V_l \underbrace{U_l^T B_l^{-1} U_l}_{G_l} V_l^T.$$

Applying the Sherman–Morrison–Woodbury formula to  $S_l$ ,

$$S_l^{-1} = C_l^{-1} + C_l^{-1} V_l (G_l^{-1} - V_l^T C_l^{-1} V_l)^{-1} V_l^T C_l^{-1},$$

shows clearly that the rank of  $S_l^{-1} - C_l^{-1}$  is bounded by the rank of  $V_l$ , which is equal to the rank of  $E_l$ .  $\square$

Lemma 4.2 shows that we can bound the rank of  $S_l^{-1} - C_l^{-1}$  based on the rank of  $E_l$ . For the 2D and 3D PDE problems discretized on the regular grids ( $N \times N$  for 2D or  $N \times N \times N$  for 3D), the rank of  $E_l$  can be analytically derived.

**LEMMA 4.3.** *Suppose an  $L$ -level HID ordering is applied to a 2D/3D regular grid. Then the rank of  $S_l^{-1} - C_l^{-1}$  has order  $O(2^{L-l})$  for 2D problems, and  $O(2^{\frac{2(L-l)}{3}} N)$  for 3D problems when  $l > 0$ .*

*Proof.* Based on Lemma 4.2, we know that we only need to count the number of the cross points at each level. For the 2D case, each connector at level  $l$  connects to upper level connectors through at most two points. There are  $2^{L-l}$  connectors in all at level  $l$ , so the rank of  $S_l^{-1} - C_l^{-1}$  is bounded by  $O(2^{L-l})$ . The ranks in the 3D case are much larger since the connectors are all 2D planes. Each connector at level  $l$  connects to upper-level connectors through at most four lines. The length of those lines has order  $O(N/2^{\frac{L-l}{3}})$ . Thus, the rank of  $S_l^{-1} - C_l^{-1}$  has order  $O(2^{\frac{2(L-l)}{3}} N)$ .  $\square$



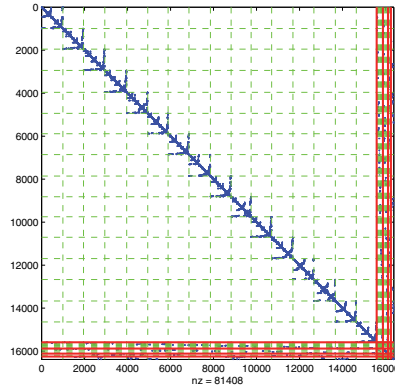


FIG. 4.1. Nonzero pattern of a reordered discretized negative 2D Laplacian with the zero Dirichlet boundary condition and a 4-level HID ordering. The original matrix is discretized on a  $128 \times 128$  regular grid with 5-point stencil.

Lemma 4.3 shows that the rank of  $S_l^{-1} - C_l^{-1}$  can be considered small when  $l$  is near  $L$ . In the extreme case when  $l = 0$ , the rank of  $S_l^{-1} - C_l^{-1}$  no longer satisfies the rank bound estimation in Lemma 4.2 and is proportional to the number of boundary points of all subdomains at level 0, which is  $O(2^{\frac{L}{2}} N)$  for the 2D case, and  $O(2^{\frac{L}{3}} N^2)$  for the 3D case. See Figures 4.1–4.4 for two Laplacian examples. We apply a 4-level HID ordering to both problems. In Figures 4.1 and 4.3, we use red solid lines to separate the unknowns at different levels and dashed green lines to indicate the diagonal blocks in each  $B^{(l)}$  ( $0 \leq l \leq 3$ ) (color available online). The eigenvalues in Figures 4.2 and 4.4 are computed by the MATLAB built-in function `eig`, and the resulting ranks are slightly different from the estimations in Lemma 4.3. The reason is that we use the graph partitioning tool METIS [19] to achieve the HID ordering so that the connectors may have irregular connectivity and do not strictly follow the assumptions in the theoretical analysis.

Figures 4.1 and 4.3 illustrate two different scenarios of interface points in 2D and 3D problems. Apparently, the ratio between the number of the interface points (which correspond to the unknowns at levels 1–4) and the number of the interior points (which correspond to the unknowns at level 0) in the 3D case is much larger than that in the 2D case, which indicates that a multilevel scheme becomes a necessity especially for 3D problems. Figures 4.2 and 4.4 numerically support Lemma 4.3 and show that the exact rank of  $S_l^{-1} - C_l^{-1}$  increases almost at the same rate as the estimation in Lemma 4.3. But a more interesting property is in regard to the numerical rank. Based on Proposition 2.1, we can expect that the numerical rank  $r_l$  of  $S_l^{-1} - C_l^{-1}$  should be much smaller. We report these numerical ranks with respect to different tolerances  $\tau$  in Tables 4.1 and 4.2 for the Laplacian examples in Figures 4.1 and 4.3. Here the numerical rank  $r_l = k$  with respect to  $\tau$ , or relative to  $\tau$ , just means that the  $k$ th singular value is larger than  $\tau \times$  the 1st singular value, while the  $(k + 1)$ st singular value is less than or equal to  $\tau \times$  the 1st singular value. (In these Laplacian examples, the eigenvalues and the singular values are the same.)

From Table 4.1, we find that the numerical ranks in the 2D Laplacian example are all less than 7 for  $\tau = 2 \times 10^{-1}$ . From Table 4.2, we observe that the ranks in the 3D problem are slightly larger than those in the 2D problem. They are all less than

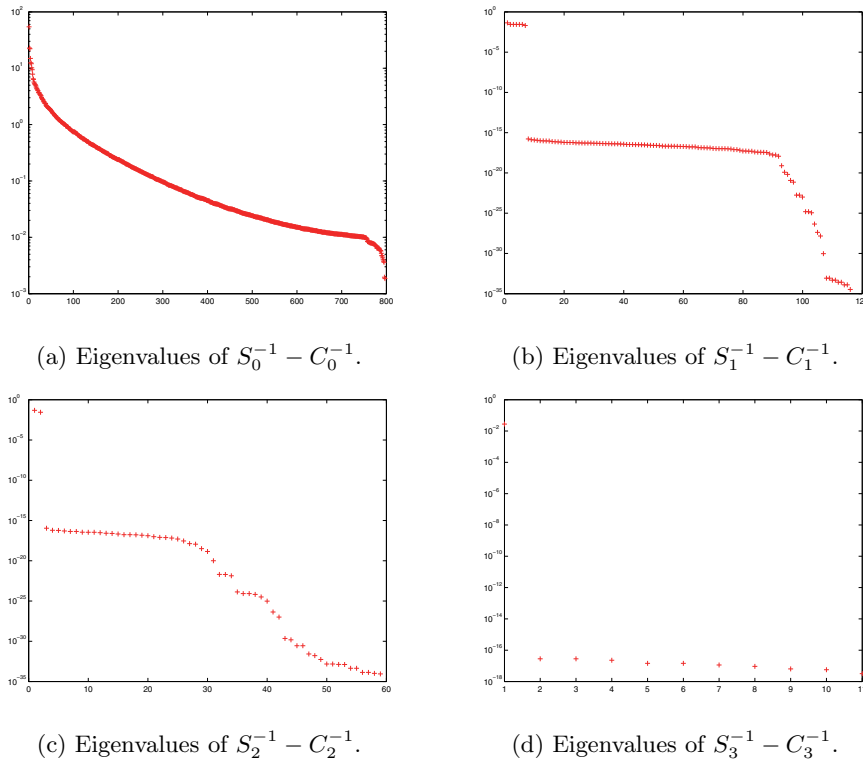


FIG. 4.2. The eigenvalues of  $S_l^{-1} - C_l^{-1}$  at different levels  $l$  for the 2D example in Figure 4.1.

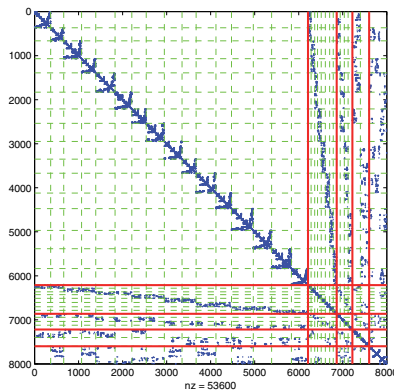


FIG. 4.3. Nonzero pattern of a reordered discretized negative 3D Laplacian with the zero Dirichlet boundary condition and a 4-level HID ordering. The original matrix is discretized on a  $20 \times 20 \times 20$  regular grid with 7-point stencil.

38 when  $\tau = 2 \times 10^{-1}$ . Therefore,  $S_l^{-1} - C_l^{-1}$  ( $0 \leq l \leq 3$ ) can be well approximated by a low-rank matrix in these Laplacian examples. Next, we will derive an efficient method to compute such an approximation without having to form  $S_l^{-1}$  explicitly. This low-rank approximation method will be used extensively in the construction of the MSLR preconditioner introduced in the next section.

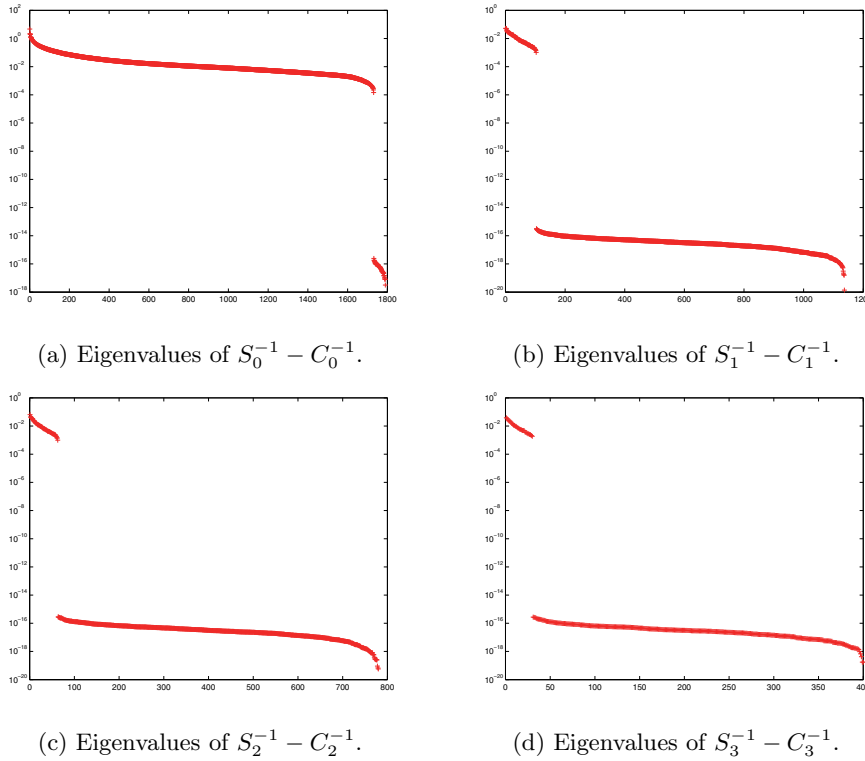


FIG. 4.4. The eigenvalues of  $S_l^{-1} - C_l^{-1}$  at different levels  $l$  for the 3D example in Figure 4.3.

TABLE 4.1

Numerical rank  $r_l$  of  $S_l^{-1} - C_l^{-1}$  for the 2D Laplacian example in Figure 4.1 relative to different tolerance  $\tau$ .

Matrix	$S_0^{-1} - C_0^{-1}$	$S_1^{-1} - C_1^{-1}$	$S_2^{-1} - C_2^{-1}$	$S_3^{-1} - C_3^{-1}$
dim	797	504	269	135
$\tau$	$r_0$	$r_1$	$r_2$	$r_3$
$2 \times 10^{-1}$	6	7	2	1
$10^{-1}$	12	7	2	1
$10^{-2}$	126	7	2	1
$10^{-3}$	371	7	2	1

TABLE 4.2

Numerical rank  $r_l$  of  $S_l^{-1} - C_l^{-1}$  for the 3D Laplacian example in Figure 4.3 relative to different tolerance  $\tau$ .

Matrix	$S_0^{-1} - C_0^{-1}$	$S_1^{-1} - C_1^{-1}$	$S_2^{-1} - C_2^{-1}$	$S_3^{-1} - C_3^{-1}$
dim	1790	1137	780	400
$\tau$	$r_0$	$r_1$	$r_2$	$r_3$
$2 \times 10^{-1}$	10	38	17	13
$10^{-1}$	28	63	32	21
$10^{-2}$	275	103	63	30
$10^{-3}$	1268	103	63	30

Based on the proof of Proposition 2.1, we know that

$$(4.1) \quad S_l^{-1} - C_l^{-1} = L_l^{-T} U_l (\Sigma_l (I - \Sigma_l)^{-1}) U_l^T L_l^{-1},$$

where  $L_l$  is the Cholesky factor of  $C_l$  and  $U_l \Sigma_l U_l^T$  is the eigendecomposition of  $L_l^{-1} E_l^T B_l^{-1} E_l L_l^{-T}$ . Notice that if we compute an approximation based on (4.1) which is adopted in [25], we would have to compute the incomplete Cholesky factorization of each  $C_l$ . Obviously, this would be highly inefficient because it is not possible to reuse the factorization information among these factors even though  $C_l$  is a submatrix of  $C_{l-k}$  for  $k < l$ .

The other way to compute such an approximation is based on the observation that  $C_l^{-1} E_l^T B_l^{-1} E_l$  has the same eigenvalues as  $L_l^{-1} E_l^T B_l^{-1} E_l L_l^{-T}$  and the columns of  $L_l^{-T} U_l$  in (4.1) are the eigenvectors of  $C_l^{-1} E_l^T B_l^{-1} E_l$  due to the following relation:

$$L_l^{-1} E_l^T B_l^{-1} E_l L_l^{-T} U_l = U_l \Sigma_l \iff C_l^{-1} E_l^T B_l^{-1} E_l L_l^{-T} U_l = L_l^{-T} U_l \Sigma_l.$$

Thus, the computation of a low-rank approximation to (4.1) can be obtained through the following generalized eigenvalue problem:

$$E_l^T B_l^{-1} E_l x = \lambda C_l x.$$

We apply the Lanczos algorithm to compute the  $k$  largest eigenpairs of the above problem and denote this computation as

$$(4.2) \quad [W_l, \Sigma_l] = \text{eigs}(C_l^{-1} E_l^T B_l^{-1} E_l, k).$$

For the Lanczos algorithm of matrix pencils, see, e.g., Chapter 9 in [33] for more details. Then, a rank- $k$  approximation to  $S_l^{-1} - C_l^{-1}$  can be chosen as

$$(4.3) \quad S_l^{-1} - C_l^{-1} \approx W_l H_l W_l^T,$$

where the diagonal matrix  $H_l$  is computed by

$$H_l = \Sigma_l (I - \Sigma_l)^{-1}.$$

This low-rank approximation method has the advantage that it only requires a system solution with the matrix  $C_l$  in the eigenvalue computation. In the next section, we will introduce a recursive formula to solve systems with  $C_l$  based on the approximation of  $C_k$  for  $k > l$ .

**5. Multilevel low-rank correction preconditioner construction.** In this section, we will show how to utilize the low-rank property discussed in section 4 to develop an efficient multilevel Schur complement based Low-Rank (MSLR) preconditioner for symmetric sparse matrices. As shown in section 2, the proposed method can also be extended to indefinite systems where only  $B_0$  is indefinite and the remaining  $B_l$ 's for  $l > 0$  are all SPD. This situation often occurs in discretized PDE problems [25].

**5.1. Basic idea of the 3-level scheme.** The basic idea of the MSLR preconditioner can be illustrated with a 3-level algorithm. The major steps are as follows:

1. Apply a 3-level HID ordering to reorder the matrix  $A$  such that the top-level connector is at level 3.

2. At level 0, factorize  $A_0$  as

$$A_0 = \begin{pmatrix} I & \\ E_0^T B_0^{-1} & I \end{pmatrix} \begin{pmatrix} B_0 & \\ & S_0 \end{pmatrix} \begin{pmatrix} I & B_0^{-1} E_0 \\ & I \end{pmatrix}.$$

Thus,

$$A_0^{-1} = \begin{pmatrix} I & -B_0^{-1} E_0 \\ & I \end{pmatrix} \begin{pmatrix} B_0^{-1} & \\ & S_0^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -E_0^T B_0^{-1} & I \end{pmatrix}.$$

This shows that the original system  $A$  can be easily solved if  $S_0^{-1}$  is available. Based on the discussion in section 4, we know that  $S_0^{-1}$  can be approximated by  $C_0^{-1}$  plus a low-rank correction term; i.e., we have

$$S_0^{-1} \approx C_0^{-1} + W_0 H_0 W_0^T.$$

Notice that this low-rank term also involves information from  $C_0^{-1}$ ; i.e., it must utilize  $C_0^{-1}$  in some way. A direct factorization of  $C_0$  is too costly if its dimension is large, so we go upward to the next level (level 1) and try to find an approximation to  $C_0^{-1}$ .

3. At level 1, we have

$$C_0^{-1} \equiv A_1^{-1} = \begin{pmatrix} I & -B_1^{-1} E_1 \\ & I \end{pmatrix} \begin{pmatrix} B_1^{-1} & \\ & S_1^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -E_1^T B_1^{-1} & I \end{pmatrix}.$$

Following the same reasoning as at level 0, the matrix of  $S_1^{-1}$  can be approximated by  $C_1^{-1}$  plus a low-rank correction:

$$S_1^{-1} \approx C_1^{-1} + W_1 H_1 W_1^T.$$

We then move to level 2 to compute  $C_1^{-1}$ .

4. At level 2, we have

$$C_1^{-1} \equiv A_2^{-1} = \begin{pmatrix} I & -B_2^{-1} E_2 \\ & I \end{pmatrix} \begin{pmatrix} B_2^{-1} & \\ & S_2^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -E_2^T B_2^{-1} & I \end{pmatrix},$$

and similarly to previous levels,

$$S_2^{-1} \approx C_2^{-1} + W_2 H_2 W_2^T.$$

We assume that the size of  $C_2$  is small enough that we can compute its incomplete Cholesky factorization,

$$C_2 \approx L_2 L_2^T.$$

5. Therefore, a 3-level scheme to compute an approximation to  $A^{-1}$  follows the dependencies as below:

$$A^{-1} \rightarrow A_0^{-1} \rightarrow S_0^{-1} \rightarrow A_1^{-1} \rightarrow S_1^{-1} \rightarrow A_2^{-1} \rightarrow S_2^{-1} \rightarrow C_2^{-1} \rightarrow L_2.$$

See Figure 5.1 for an illustration of these dependencies.

Once an approximation to  $C_l^{-1}$  is available, the low-rank correction matrices  $W_l$  and  $H_l$  can be computed with the process described by (4.2).

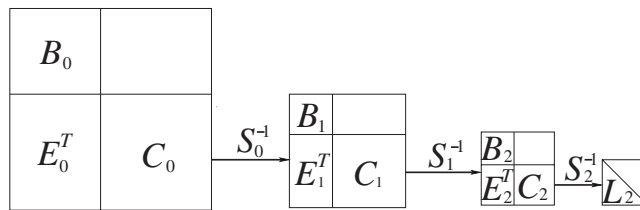


FIG. 5.1. An illustration of the dependencies in the 3-level algorithm.

**5.2. The multilevel scheme.** The 3-level algorithm introduced in the previous section illustrates the basic idea of a multilevel algorithm. In a real implementation, the MSLR preconditioner construction follows a reverse order and proceeds from level  $L - 1$  to level 0. This section provides a few details on this construction process.

First we apply an  $L$ -level HID ordering to reorder the matrix  $A$  and compute an incomplete Cholesky factorization of  $C_{L-1}$ :

$$C_{L-1} \approx M_{L-1} M_{L-1}^T.$$

We then traverse the HID tree from level  $L - 1$ . We compute an incomplete Cholesky factorization of  $B_{L-1}$ ,

$$B_{L-1} \approx L_{L-1} L_{L-1}^T,$$

and derive the low-rank correction by computing the largest  $k$  eigenpairs of the matrix  $C_{L-1}^{-1} E_{L-1}^T B_{L-1}^{-1} E_{L-1}$ ,

$$[W_{L-1}, \Sigma_{L-1}] = \text{eigs}(C_{L-1}^{-1} E_{L-1}^T B_{L-1}^{-1} E_{L-1}, k).$$

A diagonal matrix  $H_{L-1}$  is then modified from  $\Sigma_{L-1}$  and is given by

$$H_{L-1} = \Sigma_{L-1} (I - \Sigma_{L-1})^{-1}.$$

Based on the analysis in section 4, we know that

$$S_{L-1}^{-1} \approx C_{L-1}^{-1} + W_{L-1} H_{L-1} W_{L-1}^T.$$

We then go downwards of the HID tree and repeat the same operations performed on level  $L - 1$  for the remaining levels. At level 0, if  $B_0$  is still an SPD matrix, we apply the incomplete Cholesky factorization to it; otherwise we apply the incomplete LDL factorization.

When applying the generalized Lanczos algorithm to compute  $W_l$  and  $\Sigma_l$  at each level  $l$ , we need to compute products of the matrix  $C_l^{-1} E_l^T B_l^{-1} E_l$  with arbitrary vectors. Since we have the factors of  $B_l$ , the matrix-vector products associated with  $B_l^{-1}$  can be carried out by one forward and one backward substitution. However, the same strategy cannot be applied to  $C_l^{-1}$  since we do not have the incomplete factors of  $C_l$  except at level  $L - 1$ . Recall that  $C_l^{-1} \equiv A_{l+1}^{-1}$  for  $l < L - 1$ , and an approximate factorization of  $A_{l+1}$  has already been available after the computation at level  $l + 1$ . Thus, in order to reduce computational costs, we can use this approximation to apply  $C_l^{-1}$  to an arbitrary vector. More specifically, we take the following approximation form:

$$C_l \equiv A_{l+1} \approx \begin{pmatrix} L_{l+1} & \\ Y_{l+1}^T & I \end{pmatrix} \begin{pmatrix} D_{l+1} & \\ & T_{l+1} \end{pmatrix} \begin{pmatrix} L_{l+1}^T & Y_{l+1} \\ & I \end{pmatrix},$$

---

ALGORITHM 1. *Multilevel Schur complement based Low-Rank (MSLR) preconditioner construction.*

---

- 1: **procedure** MSLR
- 2:     Apply an L-level HID ordering to  $A$  and denote the reordered matrix as  $A_0$
- 3:     Factor  $C_{L-1}$  by an incomplete Cholesky factorization

$$C_{L-1} \approx M_{L-1} M_{L-1}^T$$

- 4:     **for** level  $l$  from  $L - 1$  to  $0$  **do**
- 5:         Factor  $B_l$  by an incomplete Cholesky/LDL factorization

$$B_l \approx L_l D_l L_l^T$$

- 6:         Compute the  $k$  largest eigenpairs  $\triangleright$  *Call Algorithm 2 to apply  $C_l^{-1}$  except when  $l = L - 1$*

$$[W_l, \Sigma_l] = \text{eigs}(C_l^{-1} E_l^T B_l^{-1} E_l, k)$$

by the generalized Lanczos algorithm

- 7:         Compute  $H_l = \Sigma_l(I - \Sigma_l)^{-1}$
  - 8:     **end for**
  - 9: **end procedure**
- 

---

ALGORITHM 2. *A recursive formula for the approximation of  $y = C_l^{-1}b$ .*

---

- 1: **procedure** MSLRSolve( $l, b$ )
  - 2:     **if**  $l = L - 1$  **then**
  - 3:         return  $y = M_{L-1}^{-T} M_{L-1}^{-1} b$
  - 4:     **else**
  - 5:         Split  $b = (b_1^T, b_2^T)^T$  conformingly with the blocking of  $C_l$
  - 6:         Compute  $z_1 = L_{l+1}^{-1} b_1$
  - 7:         Compute  $z_2 = b_2 - E_{l+1}^T L_{l+1}^{-T} D_{l+1}^{-T} z_1$
  - 8:         Compute  $z_1 = D_{l+1}^{-1} z_1$
  - 9:         Compute  $y_2 = \text{MSLRSolve}(l + 1, z_2)$
  - 10:         Compute  $y_2 = y_2 + W_{l+1} H_{l+1} W_{l+1}^T z_2$
  - 11:         Compute  $y_1 = L_{l+1}^{-T} (z_1 - D_{l+1}^{-1} L_{l+1}^{-1} E_{l+1} y_2)$
  - 12:         return  $y = (y_1^T, y_2^T)^T$
  - 13:     **end if**
  - 14: **end procedure**
- 

where  $Y_{l+1} = D_{l+1}^{-1} L_{l+1}^{-1} E_{l+1}$  and  $T_{l+1}^{-1} = C_{l+1}^{-1} + W_{l+1} H_{l+1} W_{l+1}^T \approx S_{l+1}^{-1}$  for  $l < L - 1$ , and  $C_{L-1} \approx M_{L-1} M_{L-1}^T$ . Note that  $D_{l+1}$  is equal to the identity matrix if  $B_{l+1}$  is SPD and  $C_{l+1}$  is solved in a recursive way. Algorithm 1 provides a detailed illustration of the multilevel construction scheme. A recursive scheme for computing the product of  $C_l^{-1}$  with a vector  $b$  is described in Algorithm 2.

Note that the block independent set structure from the HID ordering can greatly improve the efficiency of the MSLR preconditioner in a number of ways. First, in the MSLR construction algorithm, all the diagonal blocks in  $B_l$  can be factored simultaneously. Second, in the application of the MSLR preconditioner to a vector  $v$ , which corresponds to applying Algorithm 2 by setting  $l = -1$ , the triangular solves

associated with  $B_l$  can be performed independently for each diagonal block in  $B_l$ . The computation of the matrix-vector products in the generalized Lanczos algorithm can also take advantage of this structure.

**6. Complexity analysis.** In this section, we show the computational complexity and the storage cost of the MSLR preconditioner for some model problems. For simplicity, we consider the case where the maximal numerical rank used in the approximation of  $S_l^{-1} - C_l^{-1}$  is bounded by a constant  $r$ . We also want to emphasize that the prefactors in these bounds can be large.

**THEOREM 6.1.** *Suppose the matrix  $A$  is discretized on a 2D  $N \times N$  grid ( $n = N^2$ ) or a 3D  $N \times N \times N$  grid ( $n = N^3$ ), and the maximal numerical rank used in the low-rank approximation of  $S_l^{-1} - C_l^{-1}$  is bounded by a constant  $r$ . Then the optimal application cost  $\xi_{\text{appl}}$  and storage cost  $\sigma_{\text{mem}}$  for the MSLR preconditioner on this matrix  $A$  satisfy the estimates listed in Table 6.1.*

TABLE 6.1

*Application cost  $\xi_{\text{appl}}$  and storage cost  $\sigma_{\text{mem}}$  of the MSLR preconditioner for the matrix  $A$  discretized on a regular grid, where  $r$  is the maximal rank used in the low-rank corrections.*

Grid	$\xi_{\text{appl}}$	$\sigma_{\text{mem}}$
2D ( $N \times N$ , $n = N^2$ )	$O(\sqrt{rn})$	$O(\sqrt{rn})$
3D ( $N \times N \times N$ , $n = N^3$ )	$O((r^{\frac{2}{3}} + \log n)n)$	$O((r^{\frac{2}{3}} + \log n)n)$

*Proof.* The MSLR preconditioner requires storing the triangular factors and the low-rank correction matrices at each level. For the 2D case, each diagonal block in  $B_l$  has the size of  $O(\frac{N}{2^{\lfloor \frac{L-l+1}{2} \rfloor}})$ , and there are  $2^{L-l}$  blocks in total; thus  $|B_l|$  is  $O(2^{\lfloor \frac{L-l}{2} \rfloor} N)$ . The row size of  $W_l$  is equal to the row size of  $C_l$ , which is  $N$  when  $l = L - 1$  and  $\sum_{k=l+1}^{L-1} |B_k| + N$  for  $l < L - 1$ . Therefore, the storage for the low-rank correction  $\sigma_{\text{lrkmem}}$  can be estimated as

$$\sigma_{\text{lrkmem}} = \sum_{l=0}^{L-2} \left( \sum_{k=l+1}^{L-1} |B_k| + N \right) r + Nr + Lr = O(2^{\frac{L}{2}} Nr),$$

which is an increasing function of  $L$ . On the other hand, the storage  $\sigma_{\text{lumem}}$  for the triangular factors can be estimated in the following way:

$$\begin{aligned} \sigma_{\text{lumem}} &= \sum_{l=1}^L 2^{L-l} O\left(\frac{N}{2^{\lfloor \frac{L-l+1}{2} \rfloor}}\right) + 2^L O\left(\left(\frac{N}{2^{\lfloor \frac{L+1}{2} \rfloor}}\right)^3\right) \\ &= O(2^{\frac{L}{2}} N) + O\left(\frac{N^3}{2^{\frac{L}{2}}}\right). \end{aligned}$$

Here, we assume that each diagonal block in  $B_l$  for  $l > 0$  is a tridiagonal matrix and each diagonal block in  $B_0$  is a banded matrix with bandwidth  $O(\frac{N}{2^{\lfloor \frac{L+1}{2} \rfloor}})$ . Therefore, the optimal  $\sigma_{\text{mem}} = O(\sqrt{rn})$  is obtained when  $2^L = O(\frac{N^2}{r})$  for the 2D case.

The complexity for the 3D case can be analyzed similarly:

$$\sigma_{\text{lrkmem}} = \sum_{l=0}^{L-2} \left( \sum_{k=l+1}^{L-1} |B_k| + N^2 \right) r + N^2 r + Lr = O(2^{\frac{L}{3}} N^2 r)$$



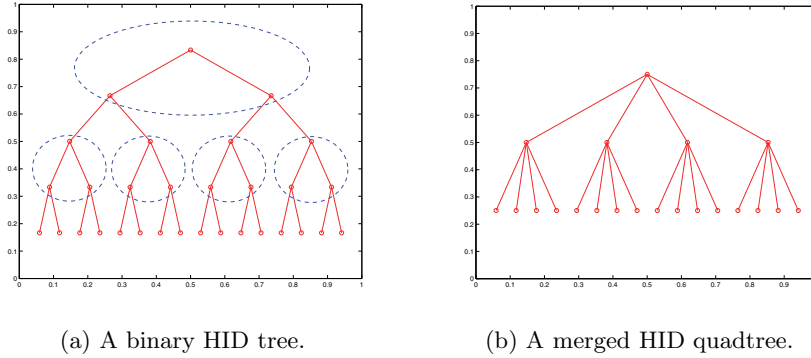


FIG. 6.1. An illustration of the merged HID tree from a binary HID tree. Here, every three nonleaf nodes in the same blue dotted ellipse in the left binary tree are merged into a new node in the right quadtree.

and

$$\begin{aligned} \sigma_{\text{lumem}} &= \sum_{l=1}^L 2^{L-l} O\left(\left(\frac{N}{2^{\lfloor \frac{L-l+1}{3} \rfloor}}\right)^3\right) + 2^L O\left(\left(\frac{N}{2^{\lfloor \frac{L+1}{3} \rfloor}}\right)^5\right) \\ &= O(N^3 L) + O\left(\frac{N^5}{2^{\frac{2L}{3}}}\right). \end{aligned}$$

The optimal  $\sigma_{\text{men}} = O((r^{\frac{2}{3}} + \log n)n)$  is obtained when  $2^L = O(\frac{N^3}{r})$ . The application cost  $\xi_{\text{appl}}$  of the MSLR preconditioner is the same as that of memory.  $\square$

Note that in practice, we can merge a few nodes on the binary HID tree into a new node to further improve the performance of the MSLR preconditioner. More specifically, for those nonleaf nodes, we merge one node and its two children nodes into a new node for 2D problems, and merge one node, its two children nodes, and four children nodes of those two children nodes into one new node for 3D problems. See Figure 6.1 for a 2D example. This modification, in fact, corresponds to a quad/octa-partitioning of the mesh for 2D/3D cases, which will create fewer but larger connectors than the original binary HID ordering. The performance is then improved on two fronts. First, the depth of the recursion in the MSLR construction algorithm is reduced, and this will lead to a more accurate approximation. Second, the number of nodes near the root node increases, and this leads to more parallelism in the upper levels of the HID tree.

**7. Numerical experiments.** In this section, we illustrate the efficiency and robustness of the MSLR preconditioner with some sparse matrices from both 2D and 3D simulations. The testing platform consisted of two Intel Xeon X5560 processors (8 MB cache, 4 cores each at 2.8 GHz) and 24 GB of memory. The MSLR preconditioner was implemented in C/C++, and the code was compiled by the Intel C compiler using the `-O2` optimization level. We also used BLAS and LAPACK routines from the Intel Math Kernel Library.

The HID ordering in the MSLR preconditioner was implemented through the `PartGraphRecursive` from METIS [19]. Each diagonal block in an MSLR preconditioner was reordered by the approximate minimum degree (AMD) ordering [1, 2] to reduce the fill-ins in the incomplete Cholesky or LDL factorizations. We consider

those reordering procedures as a preprocessing step and do not include these in the MSLR preconditioner construction time. In the generalized Lanczos algorithm, we used the full reorthogonalization algorithm and set the maximal number of Lanczos steps at less than 5 times the number of required eigenvalues.

We compared the MSLR preconditioner with the incomplete Cholesky factorization with threshold dropping (ICT), the incomplete LDL factorization with threshold dropping (ILDLT), and the restricted additive Schwarz (RAS) method with 1-level overlapping in the following tests. The accelerators we used included the conjugate gradient (CG) method [17] for the SPD cases, and the generalized minimal residual (GMRES) [30, 34] method with a restart dimension of 40, denoted by GMRES(40) for the indefinite cases. Since the RAS preconditioner is nonsymmetric even for the symmetric matrices, we used it along with the GMRES(40) for all of the numerical tests. For the ICT and the ILDLT preconditioners, the test matrices were first reordered by the AMD ordering. We followed the numerical experiments in [22, 31] and formed the right-hand side by taking  $b = Ae$ , where  $e$  is the vector of all ones.

Applications of the MSLR and the RAS preconditioners were parallelized using OpenMP [29], where the number of threads used is the number of cores, which is 8. For the MSLR preconditioner, the solve with each B-matrix, which is block-diagonal, is performed in parallel with a thread assigned to each diagonal block. Vectorizing the computation in the application of RAS is straightforward, where one thread is used for each subdomain. An exhaustive comparison with more advanced Schwarz methods such as those in [10, 40] is premature at this stage. They will be undertaken in future work once a fully optimized code based on the MSLR preconditioner is available.

The following notation is used throughout the section:

- fill: ratio of the number of nonzeros in the preconditioners over the number of nonzeros in the original matrix;
- p-t: wall clock time to build the preconditioners in seconds;
- its: number of iterations using CG or GMRES(40) with preconditioners to reduce the initial residual by a factor of  $10^{-6}$ . We use “F” to indicate non-convergence within 300 iterations;
- i-t: wall clock time required in the iteration phase. Following [22, 35], we do not report the iteration time in the case of nonconvergence and use “-” to indicate it;
- rk: maximal rank used in the low-rank corrections;
- lev: number of levels in the binary tree from the HID ordering;
- mg: type of the HID ordering used in the MSLR preconditioner construction. We use 1, 2, and 3 to denote the HID ordering associated with a binary tree, a merged quadtree (as in Figure 6.1), and a merged octatree, respectively.

**7.1. Model problems.** We begin our tests of the MSLR preconditioner with the following model problem:

$$(7.1) \quad \begin{aligned} -\Delta u - cu &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega, \end{aligned}$$

where the PDEs are defined over  $\Omega = (0, 1)^d$  with  $d = 2$  or  $3$  and with the zero Dirichlet boundary conditions. These PDEs are discretized by the 5-point stencil in two dimensions and the 7-point stencil in three dimensions.

**7.1.1. Effect of the number of levels.** One important consideration when using the MSLR preconditioner is the number of levels. We studied this effect by

TABLE 7.1

The fill-factor and iteration counts for solving a discretized negative Laplacian on a  $50^3$  grid with the CG-MSLR method. Here, the rank used in the low-rank correction is 16, and the threshold used in the incomplete Cholesky factorization is  $10^{-3}$ .

lev	fill from ICT	fill from low-rank	fill	its
5	11.6	.629	12.2	16
6	7.77	1.01	8.79	18
7	5.74	1.50	7.24	22
8	3.50	2.16	5.66	23
9	2.32	3.00	5.32	27
10	1.65	4.01	5.66	30
11	1.12	5.26	6.38	31

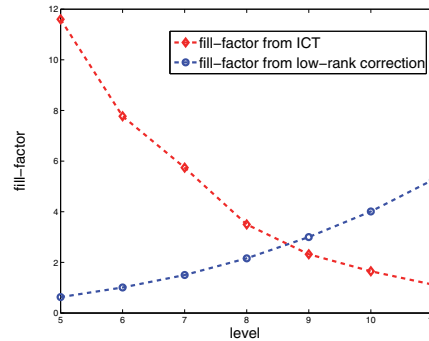


FIG. 7.1. Illustration of fill-factors from ICT and low-rank correction with respect to different levels in Table 7.1.

solving (7.1) with  $c = 0$  and discretizing it on a  $50^3$  grid. We fixed the rank to 16 and the threshold in the ICT to  $10^{-3}$  and gradually increased the number of levels from 5 to 11. We did not merge the nodes of the binary HID tree in this example. As can be seen from Table 7.1, the total fill-factor first decreases from 12.2 to 5.32 as the number of levels increases from 5 to 9 and then increases to 6.38 as the number of levels further increases to 11. Figure 7.1 shows that the fill-factor from the incomplete factorization is a decreasing function with lev, while the fill-factor from the low-rank corrections is an increasing function with lev. The optimal fill-factor was obtained when the fill-factors from these two parts were almost the same, which is when lev = 9 in this example. The results in Table 7.1 and Figure 7.1 are consistent with our complexity analysis in section 6.

**7.1.2. Effect of the rank in the low-rank corrections.** We next varied the rank used in the MSLR preconditioner and compared the iteration number and the corresponding fill-factor to solve a discretized negative Laplacian on a  $50^3$  grid with the CG method. We fixed the threshold to  $10^{-3}$  for the incomplete Cholesky factorization, lev to 7, and mg to 3 for this test matrix. The convergence results with respect to different ranks can be found in Table 7.2.

As the rank increases from 10 to 50, one can observe a steady improvement in the convergence in Table 7.2. However, larger ranks also lead to larger fill-factors and more computational time to compute the low-rank corrections. In the following numerical examples, we chose a moderate rank for the SPD problems and a relatively larger rank for the indefinite problems.

TABLE 7.2

Ranks and iteration counts for solving a discretized negative Laplacian on a  $50^3$  grid with the CG-MSLR method. Here we fix the grid size to  $50^3$ , lev to 7, and the threshold used in the incomplete Cholesky factorization to  $10^{-3}$ .

rk	fill from ICT	fill from low-rank	fill	its
10	3.56	.826	4.38	24
20	3.56	1.65	5.21	21
30	3.56	2.48	6.03	18
40	3.56	3.30	6.56	18
50	3.56	4.13	7.69	16

TABLE 7.3

Comparison of the ICT, the RAS, and the MSLR preconditioners for solving SPD linear systems from the 2D/3D PDE in (7.1) with the CG or the GMRES method when  $c = 0$ .

Grid	ICT-CG				RAS-GMRES				MSLR-CG						
	fill	p-t	its	i-t	fill	p-t	its	i-t	lev	mg	rk	fill	p-t	its	i-t
$256^2$	4.86	0.08	34	0.17	4.96	0.19	83	4.96	5	2	16	4.80	0.05	49	0.11
$512^2$	4.55	0.31	67	1.33	4.74	0.86	211	1.87	7	2	16	4.72	0.27	78	0.74
$1024^2$	4.58	1.20	128	11.1	5.21	4.38	F	–	9	2	16	4.74	1.16	159	6.49
$32^3$	4.27	0.06	16	0.05	4.31	0.17	26	0.03	7	3	16	4.13	0.10	17	0.03
$64^3$	4.14	0.46	30	0.76	4.12	1.63	49	0.53	10	3	16	4.07	0.85	35	0.47
$128^3$	4.29	3.67	57	12.40	4.20	21.90	121	19.90	13	3	16	4.16	10.18	66	8.21

**7.1.3. Preconditioning model problems.** We first solve (7.1) with  $c = 0$  and various grid sizes. We compared the convergence behavior between the MSLR, the ICT, and the RAS preconditioners. The results are reported in Table 7.3. As the grid size increased, we increased lev accordingly to control the fill-ins from the incomplete Cholesky factorization. For the 2D problems, we set lev to 5, 7, and 9 for the grid size  $256^2$ ,  $512^2$ , and  $1024^2$ , respectively. For the 3D problems, we chose lev to be 7, 10, and 13 for the grid size  $32^3$ ,  $64^3$ , and  $128^3$ , respectively. Here, the rank in the low-rank corrections was fixed to 16.

Table 7.3 shows that with almost the same fill-factors, the CPU time to construct an MSLR preconditioner is the smallest among these three for the 2D problems and becomes more expensive than the ICT preconditioner for the 3D problems. In fact, the construction time of the MSLR preconditioner is dominated by the generalized Lanczos algorithm since the computation of each Lanczos vector needs a sequence of triangular solves and sparse/dense matrix-vector products. But the MSLR preconditioner can greatly reduce the CPU time in the iteration phase in both cases. We find that even though the iteration number of the MSLR-CG method is slightly larger than that of the ICT-CG method in Table 7.3, the MSLR-CG method costs only about 60% of the iteration time used by the ICT-CG method to achieve the same accuracy. This can be explained by two factors. First, all the triangular solves associated with the diagonal blocks in the same  $B_l$  can be performed independently. We have exploited this property with the simple thread-level parallelism by OpenMP. Second, a large portion of the fill-factor in the MSLR preconditioner is from the low-rank corrections. These dense matrices participate in the application phase through very efficient matrix-vector product operations. Therefore, we can expect a more substantial performance improvement when implementing the MSLR preconditioner on SIMD-type parallel machines such as those equipped with GPUs or Intel Xeon Phi

TABLE 7.4  
Performance of the SLR-CG on the 3D model problems in Table 7.3.

Grid	fill	p-t	its	i-t
32 <sup>3</sup>	4.18	0.15	16	0.03
64 <sup>3</sup>	4.14	1.05	37	0.61
128 <sup>3</sup>	4.17	10.80	78	10.9

TABLE 7.5  
Comparison of the ILDLT, the RAS, and the MSLR preconditioners for solving symmetric indefinite linear systems from the 2D/3D PDE in (7.1) with the GMRES method when  $c > 0$ .

Grid	ILDLT-GMRES				RAS-GMRES				MSLR-GMRES						
	fill	p-t	its	i-t	fill	p-t	its	i-t	lev	mg	rk	fill	p-t	its	i-t
256 <sup>2</sup>	8.18	0.18	F	-	7.56	0.26	F	-	4	2	64	6.58	0.29	20	0.07
512 <sup>2</sup>	8.39	0.71	F	-	7.84	1.19	F	-	5	2	80	7.68	3.17	36	0.60
1024 <sup>2</sup>	12.6	5.34	F	-	19.40	22.90	F	-	6	2	180	9.13	41.09	76	6.30
32 <sup>3</sup>	5.89	0.11	21	0.11	5.78	0.09	40	0.06	7	3	32	5.60	0.25	17	0.04
64 <sup>3</sup>	7.05	1.03	F	-	11.40	3.01	F	-	10	3	64	7.06	7.44	187	3.97
128 <sup>3</sup>	9.35	12.20	F	-	10.2	31.20	F	-	13	3	64	8.07	80.20	F	-

processors in the future.

It is also interesting to see a comparison between the MSLR and the SLR [25] preconditioners on these model problems. We controlled lev, rk, and the threshold such that the SLR preconditioner had roughly the same fill-factors as the MSLR, and reported its performance in Table 7.4.

We find that when the grid size is 32<sup>3</sup>, the SLR-CG method requires almost the same number of iterations and iteration time to converge as the MSLR-CG method. But as the grid size increases, the SLR-CG method becomes less efficient. The MSLR-CG method not only requires less CPU time to construct the preconditioner, but it also requires slightly fewer iterations and less iteration time to converge.

We then consider solving indefinite systems by setting  $c > 0$  in (7.1). Here, we shifted the discretized negative Laplacian by  $sI$ , where  $s$  is related to  $c$  by  $s = h^2c$  and  $h$  is the grid spacing. We fixed  $s = 0.01$  for the 2D problems, and  $s = 0.04$  for the 3D problems. The convergence results are tabulated in Table 7.5. We can see that the ILDLT-GMRES and the RAS-GMRES fail on 5 out of the 6 test matrices, while the MSLR-GMRES method fails only on the largest 3D test matrix. The drop tolerance for the ILDLT preconditioners was set as 0.01 for the 2D problems, and 0.06 for the 3D problems. Compared with the SPD matrices in Table 7.3, we observe that the MSLR preconditioner requires larger fill-factors to converge for these indefinite systems. There are two reasons for this. First, the fill-in in the incomplete factorizations of indefinite matrices is generally more substantial than for SPD matrices, and this leads to denser factors. We also noticed that the MSLR-GMRES method failed to converge when we chose the same number of levels as in the SPD cases in Table 7.3. This forced us to reduce the number of levels. Thus, the fill-factors from the incomplete factorization are larger than those in the SPD cases. The second reason is that the MSLR preconditioner also requires larger ranks in the low-rank corrections for indefinite systems. This factor not only contributes to a larger fill-factor but also significantly increases its construction time. The reason why higher ranks are needed for indefinite problems can be understood by referring to the analysis in section 2. It shows that the most relevant low-rank approximation to  $S_l^{-1} - C_l^{-1}$  is that related to

TABLE 7.6

Example 2: Test matrices from the University of Florida Sparse Matrix Collection, where  $nnz$  stands for the number of nonzeros in the matrix.

Matrix	Order	nnz	SPD	Origin
<code>cf1</code>	70,656	1,825,580	yes	CFD problem
<code>cf2</code>	123,440	3,085,406	yes	CFD problem
<code>Dubcova3</code>	146,689	3,636,643	yes	2D/3D PDE problem
<code>thermal1</code>	82,654	574,458	yes	thermal problem
<code>thermal2</code>	1,228,045	8,580,313	yes	thermal problem
<code>F2</code>	71,505	5,294,285	no	structural problem
<code>Lin</code>	256,000	1,766,400	no	structural problem
<code>qa8fk</code>	66,127	1,660,579	no	3D acoustics problem
<code>vibrobox</code>	12,328	301,700	no	vibroacoustic problem

TABLE 7.7

Comparison of the ICT or the ILDLT, the RAS, and the MSLR preconditioners for solving general symmetric linear systems with the CG or GMRES(40) method.

Matrix	ICT/ILDLT				RAS				MSLR						
	fill	p-t	its	i-t	fill	p-t	its	i-t	lev	mg	rk	fill	p-t	its	i-t
<code>cf1</code>	5.81	3.40	298	11.7	5.61	1.83	F	-	7	1	64	5.00	1.42	85	0.96
<code>cf2</code>	4.47	3.20	271	13.60	4.52	2.65	F	-	8	2	50	4.47	1.53	155	2.58
<code>Dubcova3</code>	1.19	0.47	45	0.93	1.18	1.17	54	0.45	5	1	64	1.17	0.39	20	0.18
<code>thermal1</code>	4.57	0.20	52	0.50	4.55	0.38	235	0.77	6	3	50	4.50	0.18	38	0.14
<code>thermal2</code>	5.98	4.21	90	17.80	5.87	8.11	F	-	8	3	64	6.02	3.29	87	6.02
<code>F2</code>	2.82	3.43	F	-	2.90	3.46	F	-	6	2	64	2.49	1.24	79	1.51
<code>Lin</code>	4.61	0.53	F	-	5.36	4.63	F	-	10	3	22	4.55	0.88	185	2.96
<code>qa8fk</code>	1.90	0.26	17	0.20	4.56	1.16	25	0.20	7	2	32	1.89	0.44	21	0.14
<code>vibrobox</code>	4.13	0.43	F	-	4.35	0.36	F	-	5	1	16	3.86	0.18	57	0.16

the eigenpairs of the matrix pair  $(E_l^T B_l^{-1} E_l, C_l)$  that are near 1. When  $B_l$  is SPD, these eigenpairs correspond to the largest ones. However, when  $B_0$  is indefinite, these eigenpairs become *interior*. When computing the desired eigenpairs by the generalized Lanczos algorithm, the extreme eigenpairs generally converge before the interior ones. As the matrix size increases, the number of eigenvalues between the largest one of  $(E_0^T B_0^{-1} E_0, C_0)$  and 1 increases accordingly. Therefore, we need to compute more eigenpairs before those close to 1 reach convergence. Here, we chose the best rank by gradually increasing the rank's value and simply keeping all the computed eigenpairs needed for convergence of the low-rank correction terms.

**7.2. Preconditioning general matrices.** To show that the MSLR preconditioner is generally applicable, we further tested it on 9 symmetric matrices from the University of Florida Sparse Matrix Collection [9]. These matrices arise from different backgrounds. See Table 7.6 for a short description.

The convergence results of the three preconditioned methods are reported in Table 7.7. For the MSLR preconditioner, we chose `lev` and `mg` according to the matrix order, and `nnz` such that most fill-factors are around 4.5 and the largest one is 6.02 for the matrix `thermal2`. As can be seen, the Krylov methods with the MSLR preconditioner achieve convergence for all the test matrices, using nearly the same fill-factors, whereas they fail to converge with the other two preconditioners in many cases, especially for

those indefinite matrices. Although the MSLR preconditioner requires slightly more CPU time to construct than the ILDLT preconditioner for the matrices `Lin` and `qa8fk`, the Krylov methods with the MSLR preconditioner take the lowest iteration times to converge for all cases.

**8. Conclusion.** The MSLR preconditioner presented in this paper exploits a hierarchical graph decomposition called HID, which reorders the matrix into a multilevel block form where at each level the (1,1) block is block-diagonal. One such structure can be obtained from adapting the Nested Dissection ordering, for example, and this is what was used in this paper. The preconditioner is built by obtaining approximate inverses of certain Schur complements by exploiting a low-rank property associated with these Schur complements at each level of the HID tree.

One of the biggest attractions of the MSLR preconditioner is its robustness for indefinite problems. As was illustrated in the experiments, MSLR is able to handle problems that are out of reach of the standard ILU factorizations. Yet, this clear advantage comes with the added benefit of an inexpensive construction of the preconditioner and superior parallelism.

For these reasons we plan on extending these techniques to nonsymmetric and to complex (non-Hermitian) systems. We also plan to explore other low-rank approximation methods to further reduce preprocessing costs. In particular, newly developed rational filtering techniques [42] will be explored for computing all the eigenpairs of  $(E_l^T B_l^{-1} E_l, C_l)$  inside the interval centered at 1 with radius  $\tau$ . It is also possible to make the algorithm more adaptive by controlling the low-rank approximation accuracy at different levels with a specified threshold  $\tau$ . Finally, another topic that is worth investigating further concerns the obtention of effective HID orderings, other than those extracted from Nested Dissection.

**Acknowledgment.** The authors would like to thank three anonymous referees for their useful suggestions which led to substantial improvements of the original version of this paper.

#### REFERENCES

- [1] P. R. AMESTOY, T. A. DAVIS, AND I. S. DUFF, *An approximate minimum degree ordering algorithm*, SIAM J. Matrix Anal. Appl., 17 (1996), pp. 886–905.
- [2] P. R. AMESTOY, T. A. DAVIS, AND I. S. DUFF, *Algorithm 837: An approximate minimum degree ordering algorithm*, ACM Trans. Math. Software, 30 (2004), pp. 381–388.
- [3] M. BENZI AND M. TŮMA, *A sparse approximate inverse preconditioner for nonsymmetric linear systems*, SIAM J. Sci. Comput., 19 (1998), pp. 968–994.
- [4] M. BOLLHÖFER, J. I. ALIAGA, A. F. MARTÍN, AND E. S. QUINTANA-ORTÍ, *ILUPACK*, in Encyclopedia of Parallel Computing, Springer, New York, 2011, pp. 917–926.
- [5] E. CHOW AND Y. SAAD, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.
- [6] E. CHOW AND Y. SAAD, *Approximate inverse preconditioners via sparse-sparse iterations*, SIAM J. Sci. Comput., 19 (1998), pp. 995–1023.
- [7] E. CHOW AND Y. SAAD, *Preconditioned Krylov subspace methods for sampling multivariate Gaussian distributions*, SIAM J. Sci. Comput., 36 (2014), pp. A588–A608.
- [8] E. CORONA, P.-G. MARTINSSON, AND D. ZORIN, *An  $O(N)$  direct solver for integral equations on the plane*, Appl. Comput. Harmon. Anal., 38 (2015), pp. 284–317.
- [9] T. A. DAVIS AND Y. HU, *The University of Florida Sparse Matrix Collection*, ACM Trans. Math. Software, 38 (2011), 1.
- [10] O. DUBOIS AND M. J. GANDER, *Convergence behavior of a two-level optimized Schwarz preconditioner*, in Domain Decomposition Methods in Science and Engineering XVIII, Springer, Berlin, Heidelberg, 2009, pp. 177–184.

- [11] B. ENGQUIST AND L. YING, *Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation*, Comm. Pure Appl. Math., 64 (2011), pp. 697–735.
- [12] A. GEORGE, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.
- [13] A. GILLMAN AND P. G. MARTINSSON, *A direct solver with  $O(N)$  complexity for variable coefficient elliptic PDEs discretized via a high-order composite spectral collocation method*, SIAM J. Sci. Comput., 36 (2014), pp. A2023–A2046.
- [14] W. HACKBUSCH, *A sparse matrix arithmetic based on  $\mathcal{H}$ -matrices. Part I: Introduction to  $\mathcal{H}$ -matrices*, Computing, 62 (1999), pp. 89–108.
- [15] W. HACKBUSCH AND B. N. KHOROMSKIJ, *A sparse  $\mathcal{H}$ -matrix arithmetic. Part II: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.
- [16] P. HÉNON AND Y. SAAD, *A parallel multistage ILU factorization based on a hierarchical graph decomposition*, SIAM J. Sci. Comput., 28 (2006), pp. 2266–2293.
- [17] M. R. HESTENES AND E. L. STIEFEL, *Methods of conjugate gradients for solving linear systems*, J. Research Nat. Bur. Standards, 49 (1952), pp. 409–436.
- [18] K. L. HO AND L. YING, *Hierarchical Interpolative Factorization for Elliptic Operators: Differential Equations*, preprint, arXiv:1307.2895v1 [math.NA], 2013.
- [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] S. LE BORNE,  *$\mathcal{H}$ -matrices for convection-diffusion problems with constant convection*, Computing, 70 (2003), pp. 261–274.
- [21] S. LE BORNE AND L. GASEDYCK,  *$\mathcal{H}$ -matrix preconditioners in convection-dominated problems*, SIAM J. Matrix Anal. Appl., 27 (2006), pp. 1172–1183.
- [22] R. LI AND Y. SAAD, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.
- [23] R. LI AND Y. SAAD, *GPU-accelerated preconditioned iterative linear solvers*, J. Supercomput., 63 (2013), pp. 443–466.
- [24] R. LI AND Y. SAAD, *Low-Rank Correction Methods for Algebraic Domain Decomposition Preconditioners*, Technical Report ys-2014-4, Department of Computer Science, University of Minnesota, Minneapolis, MN, 2014; available online at <http://arxiv.org/abs/1505.04341>.
- [25] R. LI, Y. XI, AND Y. SAAD, *Schur Complement Based Domain Decomposition Preconditioners with Low-Rank Corrections*, Technical Report ys-2014-3, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 2014; available online at <http://arxiv.org/abs/1505.04340>.
- [26] J. MANDEL, B. SOUSEDÍK, AND J. ŠÍSTEK, *Adaptive BDDC in three dimensions*, Math. Comput. Simulation, 82 (2012), pp. 1812–1831.
- [27] 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.
- [28] J. K. MERIKOSKI AND R. KUMAR, *Inequalities for spreads of matrix sums and products*, Appl. Math. E-Notes, 4 (2004), pp. 150–159.
- [29] OPENMP ARCHITECTURE REVIEW BOARD, *OpenMP Application Program Interface, Version 3.1*, 2011.
- [30] Y. SAAD, *A flexible inner-outer preconditioned GMRES algorithm*, SIAM J. Sci. Comput., 14 (1993), pp. 461–469.
- [31] Y. SAAD, *ILUM: A multi-elimination ILU preconditioner for general sparse matrices*, SIAM J. Sci. Comput., 17 (1996), pp. 830–847.
- [32] Y. SAAD, *Iterative Methods for Sparse Linear Systems*, 2nd ed., SIAM, Philadelphia, 2003.
- [33] Y. SAAD, *Numerical Methods for Large Eigenvalue Problems*, revised ed., SIAM, Philadelphia, 2011.
- [34] Y. SAAD AND M. H. SCHULTZ, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems*, SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856–869.
- [35] 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.
- [36] Y. SAAD AND J. ZHANG, *BILUM: Block versions of multielimination and multilevel ILU preconditioner for general sparse linear systems*, SIAM J. Sci. Comput., 20 (1999), pp. 2103–2121.
- [37] B. SMITH, *Domain Decomposition Algorithms for the Partial Differential Equations of Linear Elasticity*, Ph.D. thesis, Courant Institute, New York University, New York, 1990.
- [38] B. SMITH, P. BJØRSTAD, AND W. GROPP, *Domain Decomposition: Parallel Multilevel Methods for Elliptic Partial Differential Equations*, Cambridge University Press, New York, 1996.
- [39] B. SOUSEDÍK, J. ŠÍSTEK, AND J. MANDEL, *Adaptive-multilevel BDDC and its parallel implementation*, Computing, 95 (2013), pp. 1087–1119.



- [40] A. ST-CYR, M. J. GANDER, AND S. J. THOMAS, *Optimized multiplicative, additive, and restricted additive Schwarz preconditioning*, SIAM J. Sci. Comput., 29 (2007), pp. 2402–2425.
- [41] A. TOSELLI AND O. B. WIDLUND, *Domain Decomposition Methods: Algorithms and Theory*, Springer Ser. Comput. Math. 34, Springer-Verlag, Berlin, 2005.
- [42] Y. XI AND Y. SAAD, *Least-Squares Rational Filters for the Solution of Interior Eigenvalue Problems*, Technical Report ys-2015-5, Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, 2015.
- [43] Y. XI, J. XIA, S. CAULEY, AND V. BALAKRISHNAN, *Superfast and stable structured solvers for Toeplitz least squares via randomized sampling*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 44–72.
- [44] Y. XI, J. XIA, AND R. CHAN, *A fast randomized eigensolver with structured LDL factorization update*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 974–996.
- [45] J. XIA, *Efficient structured multifrontal factorization for general large sparse matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A832–A860.
- [46] J. XIA, *Randomized sparse direct solvers*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 197–227.
- [47] J. XIA, S. CHANDRASEKARAN, M. GU, AND X. S. LI, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.
- [48] 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 (2009), pp. 1382–1411.
- [49] J. XIA AND M. GU, *Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2899–2920.