

# Schur complement-based domain decomposition preconditioners with low-rank corrections

Ruipeng Li<sup>1,\*</sup>, Yuanzhe Xi<sup>2</sup> and Yousef Saad<sup>2</sup>

<sup>1</sup>Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P. O. Box 808, L-561, Livermore, CA 94551, U.S.A.

<sup>2</sup>Department of Computer Science & Engineering, University of Minnesota, Twin Cities, Minneapolis, MN 55455, U.S.A.

## SUMMARY

This paper introduces a robust preconditioner for general sparse matrices based on low-rank approximations of the Schur complement in a Domain Decomposition framework. In this ‘Schur Low Rank’ preconditioning approach, the coefficient matrix is first decoupled by a graph partitioner, and then a low-rank correction is exploited to compute an approximate inverse of the Schur complement associated with the interface unknowns. The method avoids explicit formation of the Schur complement. We show the feasibility of this strategy for a model problem and conduct a detailed spectral analysis for the relation between the low-rank correction and the quality of the preconditioner. We first introduce the SLR preconditioner for symmetric positive definite matrices and symmetric indefinite matrices if the interface matrices are symmetric positive definite. Extensions to general symmetric indefinite matrices as well as to nonsymmetric matrices are also discussed. Numerical experiments on general matrices illustrate the robustness and efficiency of the proposed approach. Copyright © 2016 John Wiley & Sons, Ltd.

Received 6 July 2015; Revised 10 December 2015; Accepted 11 March 2016

KEY WORDS: low-rank approximation; domain decomposition; general sparse linear system; parallel preconditioner; Krylov subspace method; the Lanczos algorithm

## 1. INTRODUCTION

We consider the problem of solving the linear system

$$Ax = b, \quad (1)$$

with  $A \in \mathbb{C}^{n \times n}$  a large sparse matrix. Krylov subspace methods preconditioned with a form of Incomplete LU (ILU) factorization can be quite effective for this problem, but there are situations where ILU-type preconditioners encounter difficulties. For instance, when the matrix is highly ill-conditioned or indefinite, the construction of the factors may not complete or may result in unstable factors. Another situation, one that initially motivated this line of work, is that ILU-based methods can yield exceedingly poor performance on certain high performance computers such as those equipped with graphics processing units (GPUs) [1] or Intel Xeon Phi processors. This is because building and using ILU factorizations is a highly sequential process. Blocking, which is a highly effective strategy utilized by sparse direct solvers to boost performance, is rarely exploited in the realm of iterative solution techniques.

In the late 1990s, a class of methods appeared as an alternative to ILUs that did not require forward and backward triangular solves. These were developed primarily as a means to bypass the

\*Correspondence to: Ruipeng Li, Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, P. O. Box 808, L-561, Livermore, CA 94551, U.S.A.

†E-mail: li50@llnl.gov

issues just mentioned and were based on finding an approximate inverse of the original matrix that was also sparse; see, e.g., [2–4] among others. These methods were, by and large, later abandoned as practitioners found them too costly in terms of preprocessing, iteration time, and memory usage.

Another line of work that emerged in recent years as a means of computing preconditioners is that of *rank-structured matrices*. The starting point is the work by W. Hackbusch and co-workers who introduced the notion of  $\mathcal{H}$ -matrices in the 1990s [5, 6]. These were based on some interesting rank-structure observed on matrices arising from the use of the fast multipole methods or the inverses of some partial differential operators. A similar rank-structure was also exploited by others in the so-called hierarchically semi-separable matrix format, which represents certain off-diagonal blocks by low-rank matrices [7–10]. Preconditioner-based rank-structured matrices were reported in [11–14].

More recent approaches did not exploit this rank structure but focused instead on a multilevel low-rank correction technique, which include the recursive multilevel low-rank preconditioner [15], Domain Decomposition (DD) based low-rank preconditioner [16], and the LORASC preconditioner [17]. This paper generalizes the technique developed in [15] to the classical Schur complement methods and proposes a Schur complement-based low-rank (SLR) correction preconditioner.

This paper first introduces the SLR preconditioner for symmetric matrices and proposes a spectral analysis that is restricted to symmetric positive definite (SPD) cases. This method can be extended to symmetric indefinite matrices as long as they have an SPD interface, that is, the submatrix associated with the interface unknowns resulting from the partitioning is SPD. This assumption usually holds for matrices arising from finite difference or finite element discretization of PDEs. For example, consider a (shifted) Laplacian operator discretized by nine-point stencils, in which case the interface matrix can still be (strongly) diagonally dominant even when the global matrix becomes indefinite. Extensions to general symmetric indefinite matrices as well as to nonsymmetric matrices are also presented.

It is useful to compare the advantages and the disadvantages of the proposed approach with those of the traditional ILU-type and the approximate inverse-type preconditioners. First, the SLR preconditioner is directly applicable to the class of distributed linear systems that arise in any standard DD approach, including all vertex-based, edge-based, or element-based partitionings. Second, it is well suited for single-instruction-multiple-data (SIMD) parallel machines. Thus, one can expect to implement this preconditioner on a multiprocessor system based on a multi(many)-core architecture exploiting two levels of parallelism. Third, as indicated by the experimental results, this method is not as sensitive to indefiniteness as ILUs or sparse approximate inverse preconditioners. A fourth appeal, shared by all the approximate inverse-type methods, is that an SLR preconditioner can be easily updated in the sense that if it does not yield satisfactory performance, it can easily be improved without forfeiting the work performed so far in building it.

The paper is organized as follows: In Section 2, we introduce the DD framework and Schur complement techniques. A spectral analysis will be proposed in Section 3. The symmetric SLR preconditioner will be discussed in Section 4 followed by implementation details in Section 5. The extension to nonsymmetric matrices will be presented in Section 6. Numerical results of model problems, Helmholtz problems, and general symmetric linear systems will be presented in Section 7, and we conclude in Section 8.

## 2. BACKGROUND: SPARSE LINEAR SYSTEMS AND THE DOMAIN DECOMPOSITION FRAMEWORK

Reference [15] introduced a method based on a divide-and-conquer approach that consisted in approximating the inverse of a matrix  $A$  by essentially the inverse of its  $2 \times 2$  block-diagonal approximation plus a low-rank correction. This principle was then applied recursively to each of the diagonal blocks. We observed that there is often a *decay property* when approximating the inverse of a matrix by the inverse of a close-by matrix in other contexts. By this we mean that the difference between the two inverses has very rapidly decaying eigenvalues, which makes it possible to

approximate this difference by small-rank matrices. The best framework where this property takes place is that of DD, which is emphasized in this paper.

### 2.1. Graph partitioning

Figure 1 shows two standard ways of partitioning a graph [18]. On the left side is a *vertex-based* partitioning that is common in the sparse matrix community where it is also referred to as graph partitioning by *edge-separators*. A vertex is an equation-unknown pair and the partitioner subdivides the vertex set into  $p$  partitions, that is,  $p$  non-overlapping subsets whose union is equal to the original vertex set. On the right side is an *edge-based* partitioning, which, in contrast, consists of assigning edges to subdomains. This is also called graph partitioning by *vertex separators* in the graph theory community.

From the perspective of a subdomain, one can distinguish three types of unknowns: (1) interior unknowns, (2) local interface unknowns, (3) and external interface unknowns. This is illustrated by Figure 2. In a vertex-based partitioning, interior unknowns are those coupled only with local unknowns, local interface unknowns are those coupled with both external and local unknowns, and external interface unknowns are those that belong to other subdomains and are coupled with local interface unknowns. In an edge-based partitioning the local and external interface are merged into one set consisting all the nodes that are *shared* by a subdomain and its neighbors while interior nodes are those nodes that are *not shared*.

If we reorder the equations by listing the interior unknowns of the  $p$  partitioned subdomains followed by the unknowns corresponding to the global interface, we can obtain a global system that

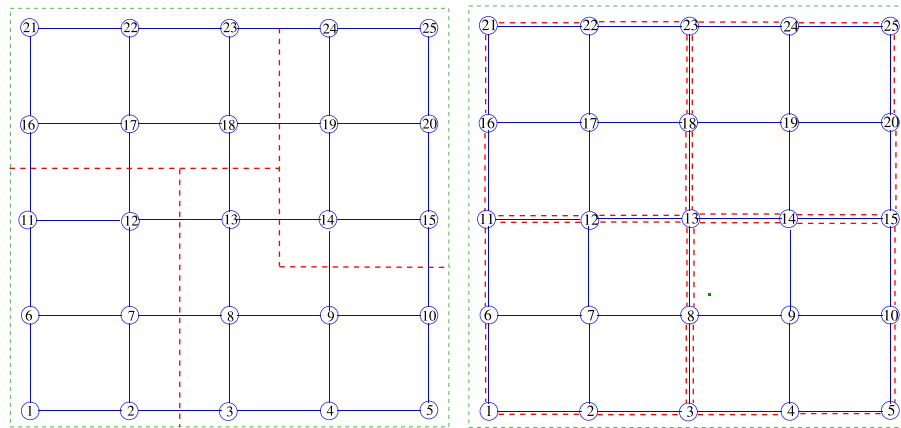


Figure 1. Two classical ways of partitioning a graph, vertex-based partitioning (left) and edge-based partitioning (right).

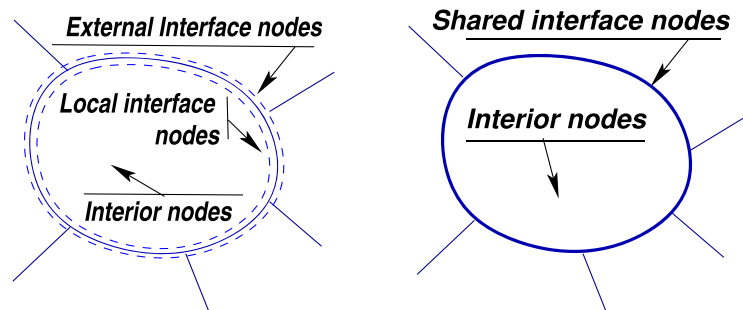


Figure 2. A local view of a partitioned domain: vertex-based partitioning (left), edge-based partitioning (right).

has the following form:

$$\left( \begin{array}{ccc|c} B_1 & & & E_1 \\ & B_2 & & E_2 \\ & & \ddots & \vdots \\ & & & B_p \\ \hline E_1^T & E_2^T & \dots & E_p^T \\ & & & C \end{array} \right) \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ y \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \\ g \end{pmatrix}, \quad (2)$$

or a more compact form,

$$\begin{pmatrix} B & E \\ E^T & C \end{pmatrix} \begin{pmatrix} u \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix}. \quad (3)$$

An illustration of the vertex-based and the edge-based partitionings is shown in Figure 3 for a 2D Laplacian. Each of these two partitioning methods has its advantages and disadvantages. In the present work, we focus on the edge-based partitioning, but this approach is also applicable to the situation of a vertex-based one.

A popular way of solving a global matrix in the form of (3) is to exploit the Schur complement techniques that eliminate the interior unknowns  $u_i$  first and then focus on computing in some way the interface unknowns. A novel approach based on this principle is proposed in the next section.

## 2.2. Schur complement techniques

To solve the system (3) obtained from a DD reordering, a number of techniques rely on the following basic block factorization

$$A = \begin{pmatrix} B & E \\ E^T & C \end{pmatrix} = \begin{pmatrix} I & \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ S & \end{pmatrix}, \quad \text{with } S = C - E^T B^{-1} E, \quad (4)$$

where  $S \in \mathbb{R}^{s \times s}$  is the ‘Schur complement’ matrix. If an approximate solve with the matrix  $S$  is available, then one can easily solve the original system by exploiting the previous factorization. In this case, note that this will require two solves with  $B$  and one solve with  $S$ . In classical ILU-type preconditioners, for example, in a two-level ARMS method [19], approximations to  $S$  are formed by dropping small terms and then ILU factorizations of  $S$  are computed. Moreover, the idea of constructing preconditioners for the reduced interface system with the Schur complement is also pursued in DD-type methods for solving substructuring problems, for example, the BDD [20] and BDDC [21, 22] methods.

In contrast, the SLR method introduced in this paper *approximates the inverse of  $S$  directly* by the sum of  $C^{-1}$  and a low-rank correction term, resulting in improved robustness for indefinite problems. Details on the low-rank property for  $S^{-1} - C^{-1}$  will be discussed in the next section.

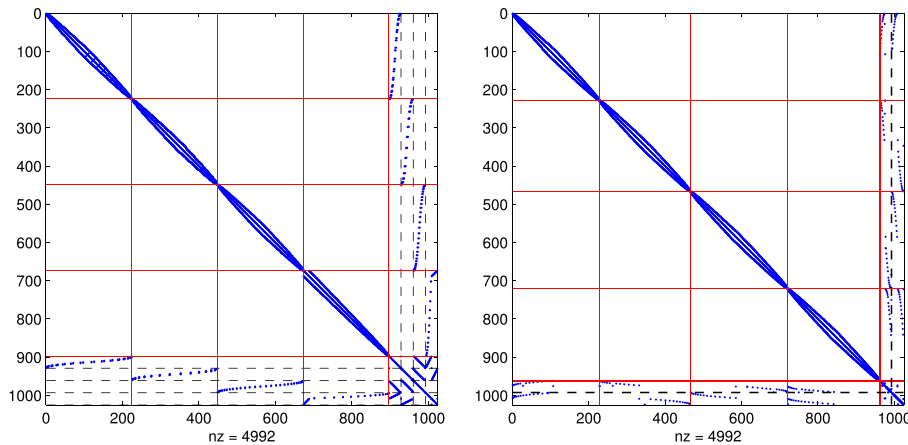


Figure 3. An example of a two-dimensional Laplacian matrix, which is partitioned into four subdomains with edge separators (left) and vertex separators (right), respectively.

## 3. SPECTRAL ANALYSIS

In this section, we study the fast eigenvalue decay property of  $S^{-1} - C^{-1}$ . In other words, our goal is to show that  $S^{-1} \approx C^{-1} + \text{LRC}$ , where LRC stands for a low-rank correction matrix.

3.1. Decay properties of  $S^{-1} - C^{-1}$ 

Assuming that the matrix  $C$  in (2) is SPD and  $C = LL^T$  is its Cholesky factorization, then we can write

$$S = L(I - L^{-1}E^T B^{-1}EL^{-T})L^T \equiv L(I - H)L^T. \quad (5)$$

Consider now the spectral factorization of  $H \in \mathbb{R}^{s \times s}$

$$H = L^{-1}E^T B^{-1}EL^{-T} = U\Lambda U^T, \quad (6)$$

where  $U$  is unitary, and  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_s)$  is the diagonal matrix of eigenvalues. When  $A$  is SPD, then  $H$  is at least symmetric positive semidefinite, and the following lemma shows that the eigenvalues  $\lambda_i$ 's are all less than one.

*Lemma 3.1*

Let  $H = L^{-1}E^T B^{-1}EL^{-T}$  and assume that  $A$  is SPD. Then we have  $0 \leq \lambda_i < 1$ , for each eigenvalue  $\lambda_i$  of  $H$ ,  $i = 1, \dots, s$ .

*Proof*

If  $A$  is SPD, then  $B$ ,  $C$ , and  $S$  are all SPD. Because an arbitrary eigenvalue  $\lambda(H)$  of  $H$  satisfies

$$\lambda(H) = \lambda(C^{-1}E^T B^{-1}E) = \lambda(C^{-1}(C - S)) = 1 - \lambda(C^{-1}S) < 1,$$

and  $H$  is at least symmetric positive semidefinite, we have  $0 \leq \lambda_i < 1$ . □

From (5), we know that the inverse of  $S$  reads

$$S^{-1} = L^{-T}(I - H)^{-1}L^{-1}. \quad (7)$$

Thus, we wish to show that *the matrix  $(I - H)^{-1}$  can be well approximated by an identity matrix plus a low rank matrix*, from which it would follow that  $S^{-1} \approx C^{-1} + \text{LRC}$  as desired. We have the following relations,

$$(I - H)^{-1} - I = L^T S^{-1} L - I = L^T (S^{-1} - C^{-1})L \equiv X, \quad (8)$$

and thus we obtain

$$S^{-1} = C^{-1} + L^{-T}XL^{-1}. \quad (9)$$

Note that the eigenvalues of  $X$  are the same as those of the matrix  $S^{-1}C - I$ . Thus, we will ask the question: Can  $X$  be well approximated by a low rank matrix? The answer can be found by examining the decay properties of the eigenvalues of  $X$ , which in turn can be assessed by checking the rate of change of the large eigenvalues of  $X$ . We can state the following result.

*Lemma 3.2*

The matrix  $X$  in (8) has the nonnegative eigenvalues  $\theta_k = \lambda_k/(1 - \lambda_k)$  for  $k = 1, \dots, s$ , where  $\lambda_k$  is the eigenvalue of the matrix  $H$  in (6).

*Proof*

From (8) the eigenvalues of the matrix  $X$  are  $(1 - \lambda_k)^{-1} - 1 = \lambda_k/(1 - \lambda_k)$ . These are nonnegative because the  $\lambda_k$ 's are between 0 and 1 from Lemma 3.1. □

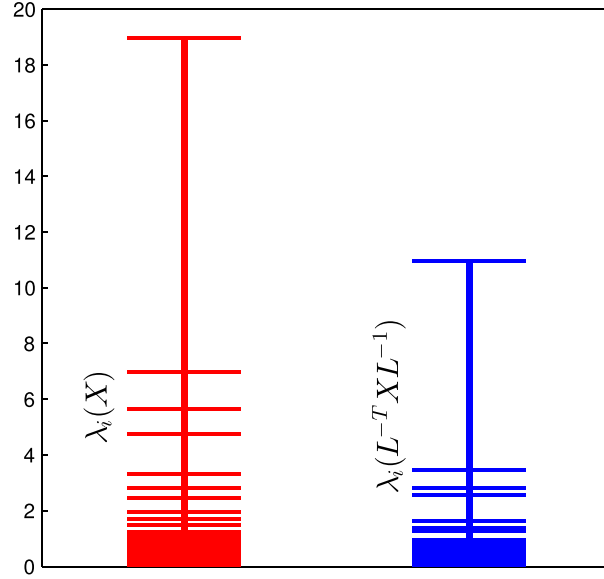


Figure 4. Illustration of the decay of eigenvalues of  $X$  (left) and  $S^{-1} - C^{-1} = L^{-T}XL^{-1}$  (right) for a two-dimensional Laplacian matrix with  $n_x = n_y = 32$ , where the domain is decomposed into four subdomains (i.e.,  $p = 4$ ), and the size of  $S$  is 127. Five eigenvectors will capture 82.5% of the spectrum of  $X$  and 85.1% of the spectrum of  $L^{-T}XL^{-1}$ , whereas 10 eigenvectors will capture 89.7% of the spectrum of  $X$  and 91.4% of the spectrum of  $L^{-T}XL^{-1}$ .

Now, we consider the derivative of  $\theta_k$  with respect to  $\lambda_k$ :

$$\frac{d\theta_k}{d\lambda_k} = \frac{1}{(1 - \lambda_k)^2}.$$

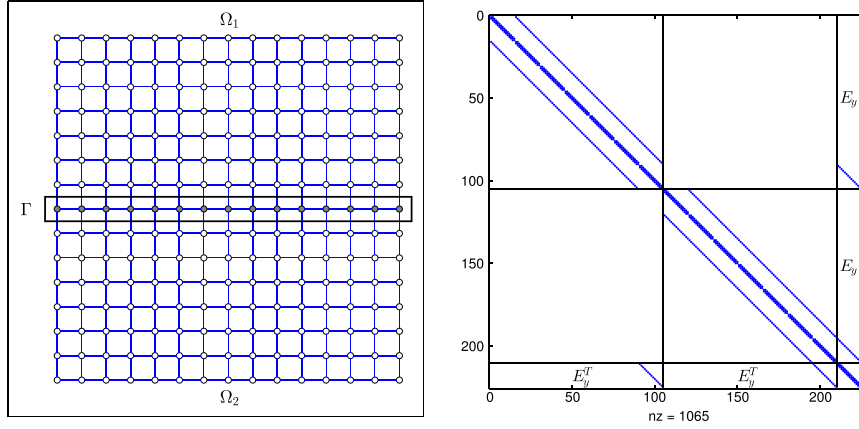
This indicates a rapid increase when  $\lambda_k$  increases toward one. In other words, this means that the largest eigenvalues of  $X$  tend to be well separated and  $X$  can be approximated accurately by a low-rank matrix in general. Figure 4 illustrates the decay of the eigenvalues of the matrix  $L^{-T}XL^{-1}$  and the matrix  $X$  for a 2D Laplacian matrix that is precisely the matrix shown in Figure 3. As can be seen, using just a few eigenvalues and vectors will represent the matrix  $X$  (or  $L^{-T}XL^{-1}$ ) quite well. In this particular situation, five eigenvectors (out of the total of 127) will capture 82.5% of  $X$  and 85.1% of  $L^{-T}XL^{-1}$ , whereas 10 eigenvectors will capture 89.7% of  $X$  and 91.4% of  $L^{-T}XL^{-1}$ .

### 3.2. Two-domain analysis in a two-dimensional model problem

The spectral analysis of the matrix  $S^{-1} - C^{-1}$  is difficult for general problems and general partitionings. In the simplest case when the matrix  $A$  originates from a 2D Laplacian on a regular grid, discretized by centered differences, and it is partitioned into two subdomains, the analysis becomes feasible. The goal of this section is to show that the eigenvalues of  $X$  and  $L^{-T}XL^{-1}$  decay rapidly.

Assume that  $-\Delta$  is discretized on a grid  $\Omega$  of size  $n_x \times (2n_y + 1)$  with Dirichlet boundary conditions and that the ordering is major along the  $x$  direction. The grid is partitioned horizontally into three parts: the two disconnected  $n_x \times n_y$  grids, namely  $\Omega_1$  and  $\Omega_2$ , which are the same, and the  $n_x \times 1$  separator denoted by  $\Gamma$ . See Figure 5(a) for an illustration. Let  $T_x$  be the tridiagonal matrix corresponding to  $\Gamma$  of dimension  $n_x \times n_x$  which discretizes  $-\partial^2/\partial x^2$ . The scaling term  $1/h^2$  is omitted so that  $T_x$  has the constant 2 on its main diagonal and  $-1$  on the co-diagonals. Finally, we denote by  $A$  the matrix which results from discretizing  $-\Delta$  on  $\Omega$  and reordered according to the partitioning  $\Omega = \{\Omega_1, \Omega_2, \Gamma\}$ . In  $\Omega_1$  and  $\Omega_2$ , the interface nodes are ordered at the end. Hence,  $A$  has the form:

$$A = \begin{pmatrix} A_y & E_y \\ & A_y & E_y \\ E_y^T & E_y^T & \hat{T}_x \end{pmatrix}, \quad (10)$$



(a) Partition of a regular mesh into 3 parts. (b) Nonzero pattern of the reordered matrix.

Figure 5. Illustration of the matrix  $A$  and the corresponding partitioning of the two-dimensional mesh.

where  $A_y$  corresponds to the  $n_x \times n_y$  grid (i.e.,  $\Omega_1$  or  $\Omega_2$ ),  $E_y$  defines the couplings between  $\Omega_1$  (or  $\Omega_2$ ) and  $\Gamma$ , and the matrix  $\hat{T}_x$  is associated with  $\Gamma$ , for which we have

$$\hat{T}_x = T_x + 2I. \quad (11)$$

Figure 5(b) is an illustration of the nonzero pattern of  $A$ .

Therefore, the Schur complement associated with  $\Gamma$  in (10) reads

$$S_\Gamma = \hat{T}_x - 2E_y^T A_y^{-1} E_y, \quad (12)$$

and the eigenvalues of  $X$  and  $L^{-T}XL^{-1}$  correspond to those of  $S_\Gamma^{-1}\hat{T}_x - I$  and  $S_\Gamma^{-1} - \hat{T}_x^{-1}$ , respectively, in this case. The coupling matrix  $E_y$  has the form  $E_y^T = (0, I_x)$ , where  $I_x$  denotes the identity matrix of size  $n_x$ . Clearly, the matrix  $R_y = E_y^T A_y^{-1} E_y$  is simply the bottom right (corner), block of the inverse of  $A_y$ , which can be readily obtained from a standard block factorization. Noting that  $A_y$  is of the form

$$A_y = \begin{pmatrix} \hat{T}_x & -I & & & \\ -I & \hat{T}_x & -I & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -I \\ & & & -I & \hat{T}_x \end{pmatrix},$$

we write its block LU factorization as:

$$A_y = \begin{pmatrix} I & & & & \\ -D_1^{-1} & I & & & \\ & -D_2^{-1} & \ddots & & \\ & & \ddots & \ddots & \\ & & & -D_{n_y}^{-1} & I \end{pmatrix} \begin{pmatrix} D_1 & -I & & & \\ D_2 & -I & & & \\ & \ddots & \ddots & & \\ & & \ddots & -I & \\ & & & D_{n_y} \end{pmatrix}.$$

The  $D_i$ s satisfy the recurrence:  $D_k = \hat{T}_x - D_{k-1}^{-1}$ , for  $k = 2, \dots, n_y$ , starting with  $D_1 = \hat{T}_x$ . The result is that each  $D_k$  is a continued fraction in  $\hat{T}_x$ . As can be easily verified  $R_y$  is equal to  $D_{n_y}^{-1}$ . The scalar version of the above recurrence is of the form:

$$d_k = 2a - \frac{1}{d_{k-1}}, \quad k = 2, \dots, n_y, \quad \text{with} \quad d_1 \equiv 2a.$$

The  $d_i$ s are the diagonal entries of the U-matrix of an LU factorization similar to the one above but applied to the  $n_y \times n_y$  tridiagonal matrix  $T$  that has  $2a$  on the diagonal and  $-1$  on the co-diagonals. For reasons that will become clear we replaced the matrix  $\hat{T}_x$  by the scalar  $2a$ . We are interested in the inverse of the last entry, i.e.,  $d_{n_y}^{-1}$ . Using Chebyshev polynomials we can easily see that  $d_{n_y}^{-1} = U_{n_y-1}(a)/U_{n_y}(a)$ , where  $U_k(t)$  is the Chebyshev polynomial of the second kind (for details, see Appendix):

$$U_k(t) = \frac{\sinh((k+1)\cosh^{-1}(t))}{\sinh(\cosh^{-1}(t))}.$$

In terms of the original matrix  $A_y$ , the scalar  $a$  needs to be substituted by  $\hat{T}_x/2 = I + T_x/2$ . In the end, the matrix  $S^{-1} - C^{-1} = S_\Gamma^{-1} - \hat{T}_x^{-1}$  is a rational function of  $\hat{T}_x/2$ . We denote this rational function by  $s(t)$ , that is,  $S_\Gamma^{-1} - \hat{T}_x^{-1} = s(\hat{T}_x/2)$  and note that  $s$  is well defined in terms of the scalar  $a$ . Indeed, from earlier,

$$s(a) = \frac{1}{2a - 2\frac{U_{n_y-1}(a)}{U_{n_y}(a)}} - \frac{1}{2a} = \frac{U_{n_y-1}(a)}{a(2aU_{n_y}(a) - 2U_{n_y-1}(a))} = \frac{U_{n_y-1}(a)}{a[U_{n_y+1}(a) - U_{n_y-1}(a)]}.$$

Everything can now be expressed in terms of the eigenvalues of  $\hat{T}_x/2$ , which are

$$\eta_k = 1 + 2\sin^2 \frac{k\pi}{2(n_x + 1)}, \quad k = 1, \dots, n_x. \quad (13)$$

We can then state the following.

*Proposition 3.1*

Let  $\eta_k$  be defined in (13) and  $\theta_k = \cosh^{-1}(\eta_k)$ ,  $k = 1, \dots, n_x$ . Then, the eigenvalues  $\gamma_k$  of  $S_\Gamma^{-1} - \hat{T}_x^{-1}$  are given by the following:

$$\gamma_k = \frac{\sinh(n_y \theta_k)}{\eta_k [\sinh((n_y + 2)\theta_k) - \sinh(n_y \theta_k)]}, \quad k = 1, \dots, n_x. \quad (14)$$

Note that we have  $e^{\theta_k} = \eta_k + \sqrt{\eta_k^2 - 1}$  and  $\sinh(n\theta_k) = \left[ \left( \eta_k + \sqrt{\eta_k^2 - 1} \right)^n - \left( \eta_k + \sqrt{\eta_k^2 - 1} \right)^{-n} \right] / 2$ , which is well approximated by  $\left( \eta_k + \sqrt{\eta_k^2 - 1} \right)^n / 2$  for a large  $n$ . In the end, assuming  $n_y$  is large enough, we have

$$\gamma_k \approx \frac{1}{\eta_k \left[ \left( \eta_k + \sqrt{\eta_k^2 - 1} \right)^2 - 1 \right]} = \frac{1}{2\eta_k \left[ (\eta_k^2 - 1) + \eta_k \sqrt{\eta_k^2 - 1} \right]}. \quad (15)$$

This shows that for those eigenvalues of  $\hat{T}_x$  that are close to one, we would have a big amplification to the value  $1/\eta_k$ . These eigenvalues correspond to the smallest eigenvalues of  $T_x$ . We can also show that

$$\gamma_k \approx \frac{1}{2} \left[ \frac{1}{\sqrt{\eta_k^2 - 1}} - \frac{1}{\eta_k} \right],$$

while for the eigenvalues  $\zeta_k$  of  $S_\Gamma^{-1} \hat{T}_x - I$ , we have

$$\zeta_k = 2\eta_k \gamma_k \approx \frac{\eta_k}{\sqrt{\eta_k^2 - 1}} - 1.$$

An illustration of  $\gamma_k$ ,  $\zeta_k$  and  $1/\eta_k$  is shown in Figure 6.



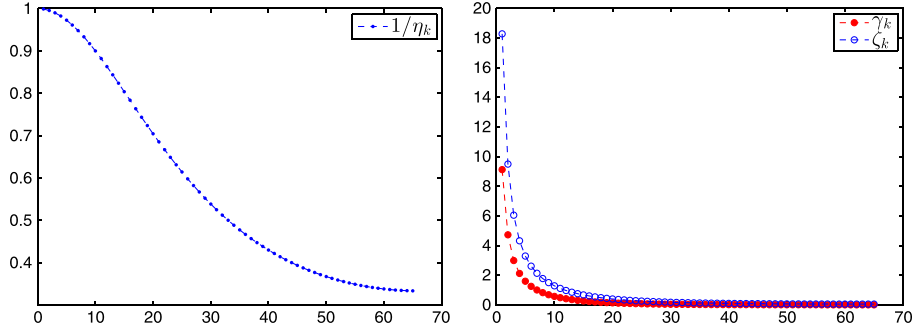


Figure 6. Illustration of the decay of the eigenvalues  $\gamma_k$  of the matrix  $S^{-1} - C^{-1}$  and the eigenvalues  $\zeta_k$  of the matrix  $S^{-1}C - I$ , and  $1/\eta_k$  for  $-\Delta$  on a two-dimensional grid of size  $n_x \times (2n_y + 1)$  with  $n_x = 65, n_y = 32$ , which is partitioned into two subdomains.

#### 4. SCHUR COMPLEMENT-BASED PRECONDITIONING WITH LOW-RANK CORRECTIONS

The goal of this section is to build a preconditioner for matrix  $A$  in the form (3) obtained from the DD method. The preconditioning matrix  $M$  is of the form

$$M = \begin{pmatrix} I & \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & E \\ & \tilde{S} \end{pmatrix}, \quad (16)$$

where  $\tilde{S}$  is an approximation to  $S$ . The aforementioned text is approximate factorization of (4) whereby (only)  $S$  is approximated. In fact, we will approximate directly the inverse of  $S$  instead of  $S$  by exploiting low-rank properties. Specifically, we seek an approximation of the form  $\tilde{S}^{-1} = C^{-1} + \text{LRC}$ , where LRC stands for a low-rank correction matrix. From a practical point of view, it will be difficult to compute directly an approximation to the matrix  $S^{-1} - C^{-1}$ , because  $S^{-1}$  is not available and we do not (yet) have an efficient means for solving linear systems with the matrix  $S$ . Instead, we will extract this approximation from that of the matrix  $X$  defined in Section 3.1; see (8). Recall the expression (5) and the eigen-decomposition of  $H$  in (6), which yield,

$$S = L(I - U\Lambda U^T)L^T = LU(I - \Lambda)U^T L^T. \quad (17)$$

The inverse of  $S$  is then

$$S^{-1} = L^{-T}U(I - \Lambda)^{-1}U^T L^{-1},$$

which we write in the form,

$$\begin{aligned} S^{-1} &= L^{-T} \left( I + U \left[ (I - \Lambda)^{-1} - I \right] U^T \right) L^{-1}, \\ &= C^{-1} + L^{-T}U \left[ (I - \Lambda)^{-1} - I \right] U^T L^{-1}. \end{aligned}$$

Now, assuming that  $H$  has an approximation of the following form,

$$\tilde{H} \approx U\tilde{\Lambda}U^T, \quad \tilde{\Lambda} = \text{diag}(\tilde{\lambda}_1, \dots, \tilde{\lambda}_s), \quad (18)$$

we will obtain the following approximation to  $S^{-1}$ :

$$\tilde{S}^{-1} = L^{-T}U(I - \tilde{\Lambda})^{-1}U^T L^{-1}, \quad (19)$$

$$= C^{-1} + L^{-T}U \left[ (I - \tilde{\Lambda})^{-1} - I \right] U^T L^{-1}. \quad (20)$$

##### Proposition 4.1

Let  $S$  and  $H$  be defined by (5) and (6), respectively, and let  $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_s)$  with the  $\sigma_i$ 's defined by the following:

$$\sigma_i = \frac{1 - \lambda_i}{1 - \tilde{\lambda}_i}, \quad i = 1, \dots, s. \quad (21)$$

Then, the eigendecomposition of  $S\tilde{S}^{-1}$  is given by the following:

$$S\tilde{S}^{-1} = (LU)\Sigma(LU)^{-1}. \quad (22)$$

*Proof*

From (17) and (19), we have

$$\begin{aligned} S\tilde{S}^{-1} &= LU(I - \Lambda)U^T L^T L^{-T} \left( U(I - \tilde{\Lambda})^{-1} U^T \right) L^{-1} \\ &= (LU)(I - \Lambda)(I - \tilde{\Lambda})^{-1} (U^T L^{-1}) = (LU)\Sigma(LU)^{-1}. \end{aligned}$$

□

The simplest selection of  $\tilde{\Lambda}$  is the one that ensures that the  $k$  largest eigenvalues of  $(I - \tilde{\Lambda})^{-1}$  match the largest eigenvalues of  $(I - \Lambda)^{-1}$ . Assume that the eigenvalues of  $H$  are  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_s$ , which means that the diagonal entries  $\tilde{\lambda}_i$  of  $\tilde{\Lambda}$  are selected such that

$$\tilde{\lambda}_i = \begin{cases} \lambda_i & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases}. \quad (23)$$

Proposition 4.1 indicates that in this case, the eigenvalues of  $S\tilde{S}^{-1}$  are

$$\begin{cases} 1 & \text{if } i \leq k \\ 1 - \lambda_i & \text{otherwise} \end{cases}.$$

Thus, we can infer that in this situation,  $k$  eigenvalues of  $S\tilde{S}^{-1}$  will take the value one and the other  $s - k$  eigenvalues  $\sigma_i$  satisfy  $0 < 1 - \lambda_{k+1} \leq \sigma_i < 1 - \lambda_s < 1$ .

Another choice for  $\tilde{\Lambda}$ , inspired by [16], will make the eigenvalues of  $S\tilde{S}^{-1}$  larger than or equal to one. Consider defining  $\tilde{\Lambda}$  such that

$$\tilde{\lambda}_i = \begin{cases} \lambda_i & \text{if } i \leq k \\ \theta & \text{if } i > k \end{cases}. \quad (24)$$

Then, from (21), the eigenvalues of  $S\tilde{S}^{-1}$  are

$$\begin{cases} 1 & \text{if } i \leq k \\ (1 - \lambda_i)/(1 - \theta) & \text{if } i > k \end{cases}. \quad (25)$$

The earlier definition of  $\Lambda_k$  in (23), which truncates the lowest eigenvalues of  $H$  to zero, corresponds to selecting  $\theta = 0$ . This is essentially an *eigenvalue deflation* scheme to matrix  $SC^{-1}$ . Note that the eigenvalues of  $SC^{-1}$  are the same as those of  $I - H$ , which are  $1 - \lambda_i$ . In the previous scheme, the first  $k$  eigenvalues of  $SC^{-1}$  are moved to 1, and the others are scaled by  $1/(1 - \theta)$ . For  $i > k$ , the eigenvalues can be made greater than or equal to one by selecting  $\lambda_{k+1} \leq \theta < 1$ . In this case, the eigenvalues  $\sigma_i$  for  $i > k$ , which are equal to  $\sigma_i = (1 - \lambda_i)/(1 - \theta)$ , belong to the interval

$$\left[ 1, \frac{1 - \lambda_s}{1 - \theta} \right] \subseteq \left[ 1, \frac{1}{1 - \theta} \right]. \quad (26)$$

Thus, the spectral condition number of the preconditioned matrix is  $(1 - \lambda_s)/(1 - \theta)$ . The choice leading to the smallest two-norm deviation is letting  $\theta = \lambda_{k+1}$ . One question that may be asked is how does the condition number  $\kappa = \max \sigma_i / \min \sigma_i$  vary when  $\theta$  varies between 0 and 1?

First, observe that a general expression for the eigenvalues of  $S\tilde{S}^{-1}$  is given by (25) regardless of the value of  $\theta$ . When  $\lambda_{k+1} \leq \theta < 1$ , we just saw that the spectral condition number is equal to

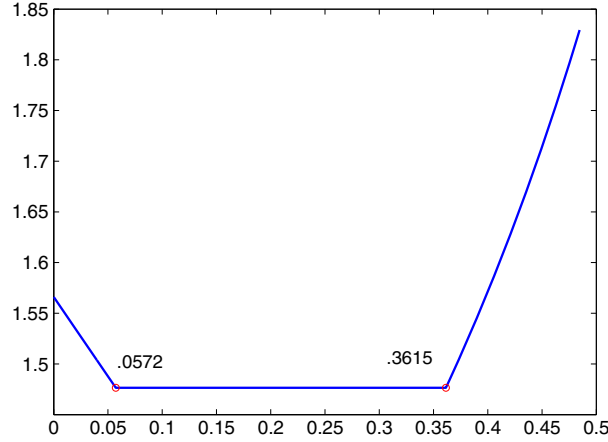


Figure 7. Illustration of the condition number  $\kappa(\theta)$  for the case of a two-dimensional Laplacian matrix with  $n_x = n_y = 256$  and the number of the subdomains  $p = 2$ , where 64 eigenvectors are used (*i.e.*,  $k = 64$ ).  $\lambda_s = .05719$ ,  $\lambda_{k+1} = .36145$ , and the optimal condition number is  $\kappa = 1.4765$ .

$(1 - \lambda_s)/(1 - \theta)$ . The smallest value of this condition number is reached when  $\theta$  takes the smallest value, which, recalling our restriction  $\lambda_{k+1} \leq \theta < 1$ , is  $\theta = \lambda_{k+1}$ . There is a second situation, which corresponds to when  $\lambda_s \leq \theta \leq \lambda_{k+1}$ . Here, the largest eigenvalue is still  $(1 - \lambda_s)/(1 - \theta)$ , which is larger than one. The smallest one is now smaller than one, which is  $(1 - \lambda_{k+1})/(1 - \theta)$ . So the condition number now is again  $(1 - \lambda_s)/(1 - \lambda_{k+1})$ , which is independent of  $\theta$  in the interval  $[\lambda_s, \lambda_{k+1}]$ . The third and final situation corresponds to the case when  $0 \leq \theta \leq \lambda_s$ . The largest eigenvalue is now one, because  $(1 - \lambda_s)/(1 - \theta) < 1$ , while the smallest one is still  $(1 - \lambda_{k+1})/(1 - \theta)$ . This leads to the condition number  $(1 - \theta)/(1 - \lambda_{k+1})$ , and the smallest spectral condition number for  $\theta$  in this interval is reached when  $\theta = \lambda_s$  leading to the same optimal condition number  $(1 - \lambda_s)/(1 - \lambda_{k+1})$ . This result is summarized in the following proposition.

*Proposition 4.2*

The spectral condition number  $\kappa(\theta)$  of  $S\tilde{S}^{-1}$  is equal to

$$\kappa(\theta) = \begin{cases} \frac{1 - \theta}{1 - \lambda_{k+1}} & \text{if } \theta \in [0, \lambda_s) \\ \frac{1 - \lambda_s}{1 - \lambda_{k+1}} & \text{if } \theta \in [\lambda_s, \lambda_{k+1}] \\ \frac{1 - \lambda_s}{1 - \theta} & \text{if } \theta \in (\lambda_{k+1}, 1) \end{cases} \quad (27)$$

It has a minimum value of  $(1 - \lambda_s)/(1 - \lambda_{k+1})$ , which is reached for any  $\theta$  in the second interval.

Figure 7 shows the variation of the condition number  $\kappa(\theta)$  as a function of  $\theta$ , for a 2D Laplacian matrix. One may conclude from this result that there is no reason for selecting a particular  $\theta \in [\lambda_s, \lambda_{k+1}]$  over another one as long as  $\theta$  belongs to the middle interval, because the spectral condition number  $\kappa(\theta)$  is the same. In fact, in practice when approximate eigenpairs are used that are computed, for example, by the Lanczos procedure, the choice  $\theta = \lambda_{k+1}$  often gives better performance than  $\theta = \lambda_s$  in this context because for the former choice, the perturbed eigenvalues are less likely to be close to zero. An example can be found in Figure 8, which shows that when using accurate enough eigenpairs, both choices of  $\theta$  will give the same condition number (which is also the optimal one), whereas when relatively inaccurate eigenpairs are used, setting  $\theta = \lambda_{k+1}$  can give a better condition number than that obtained from setting  $\theta = \lambda_s$ . In what follows, we assume that the approximation scheme (24) is used with  $\theta = \lambda_{k+1}$ , and we will denote by  $S_{k,\theta}^{-1}$  the related approximate inverse of  $S$ .

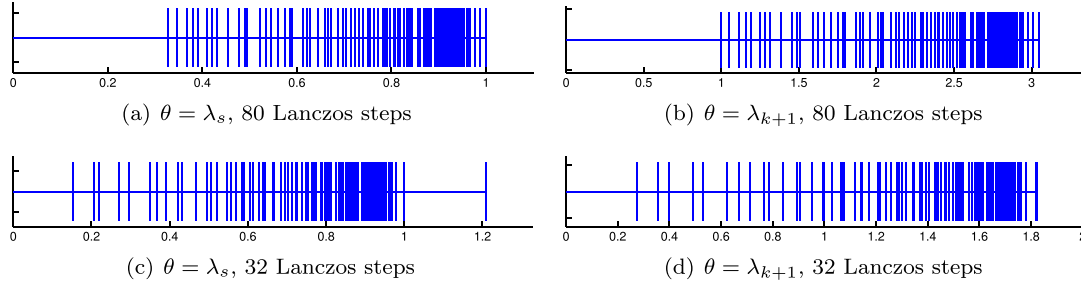


Figure 8. Illustration of the eigenvalues of  $S\tilde{S}^{-1}$  for the case of a two-dimensional Laplacian matrix with  $n_x = n_y = 128$ , the number of subdomains  $p = 2$  and the rank  $k = 16$ , such that the optimal spectral condition number  $\kappa(\theta) = 3.0464$ , for  $\lambda_s \leq \theta \leq \lambda_{k+1}$ . The two top figures show the eigenvalues of  $S\tilde{S}^{-1}$  with the Ritz values and vectors from 80 steps of the Lanczos iterations, where for both choices  $\theta = \lambda_s$  and  $\theta = \lambda_{k+1}$ ,  $\kappa(\theta) = 3.0464$ . The bottom two figures show the eigenvalues of  $S\tilde{S}^{-1}$  in the cases with 32 steps of the Lanczos iterations, where  $\kappa(\lambda_s) = 7.8940$  while  $\kappa(\lambda_{k+1}) = 6.6062$ .

From an implementation point of view, it is clear that only the  $k$  largest eigenvalues and the associated eigenvectors as well as the  $(k+1)$ -st largest eigenvalue of the matrix  $C^{-1}E^TB^{-1}E$  are needed. We prove this result in the following proposition.

*Proposition 4.3*

Let  $Z_k$  be the matrix whose column vectors are eigenvectors of  $C^{-1}E^TB^{-1}E$  associated with the  $k$  largest eigenvalues, and let  $\theta = \lambda_{k+1}$ . Then the following expression for  $S_{k,\theta}^{-1}$  holds:

$$S_{k,\theta}^{-1} = \frac{1}{1-\theta}C^{-1} + Z_k[(I - \Lambda_k)^{-1} - (1-\theta)^{-1}I]Z_k^T. \quad (28)$$

*Proof*

We write  $U = [U_k, W]$ , where  $U_k = [u_1, \dots, u_k]$  contains the eigenvectors of  $H$  associated with the largest  $k$  eigenvalues and  $W$  contains the remaining columns  $u_{k+1}, \dots, u_s$ . Note that  $W$  is not available but we use the fact that  $WW^T = I - U_kU_k^T$  for the purpose of this proof. With this, (20) becomes

$$\begin{aligned} S_{k,\theta}^{-1} &= C^{-1} + L^{-T}[U_k, W] \begin{pmatrix} (I - \Lambda_k)^{-1} - I & 0 \\ 0 & ((1-\theta)^{-1} - 1)I \end{pmatrix} [U_k, W]^T L^{-1} \\ &= C^{-1} + Z_k[(I - \Lambda_k)^{-1} - I]Z_k^T + [(1-\theta)^{-1} - 1]L^{-T}WW^TL^{-1} \\ &= C^{-1} + Z_k[(I - \Lambda_k)^{-1} - I]Z_k^T + [(1-\theta)^{-1} - 1]L^{-T}(I - U_kU_k^T)L^{-1} \\ &= \frac{1}{1-\theta}C^{-1} + Z_k[(I - \Lambda_k)^{-1} - (1-\theta)^{-1}I]Z_k^T. \end{aligned}$$

□

In a paper describing a similar technique, Grigori *et al.* [17] suggest another choice of  $\tilde{\Lambda}$ , which is as follows:

$$\tilde{\lambda}_i = \begin{cases} 1 - (1 - \lambda_i)/\varepsilon & \text{if } i \leq k \\ 0 & \text{otherwise} \end{cases}, \quad (29)$$

where  $\varepsilon$  is a parameter. Then the eigenvalues  $\sigma_i$ 's are

$$\begin{cases} \varepsilon & \text{if } i \leq k \\ 1 - \lambda_i & \text{otherwise} \end{cases}.$$

Note that the first choice in (23) is a special case of (29) when  $\varepsilon = 1$ . Writing the transformed eigenvalues as

$$\{\varepsilon, 1 - \lambda_{k+1}, 1 - \lambda_{k+2}, \dots, 1 - \lambda_s\},$$

the authors stated that the resulting condition number is  $\kappa = (1 - \lambda_s)/\varepsilon$ , with an implied assumption that  $\varepsilon \leq 1 - \lambda_{k+1}$ . In all cases, when  $1 - \lambda_{k+1} < \varepsilon \leq 1 - \lambda_s$ , the spectral condition number is the same as earlier, that is, equal to  $(1 - \lambda_s)/(1 - \lambda_{k+1})$ . On the other hand, when  $0 \leq \varepsilon \leq 1 - \lambda_{k+1}$ , then the condition number is now  $(1 - \lambda_s)/\varepsilon$ , and the best value will be reached again for  $\varepsilon = 1 - \lambda_{k+1}$ , which leads to the same condition number as mentioned earlier.

In all cases, if we want the spectral condition number of the matrix  $S\tilde{S}^{-1}$ , which is  $\kappa = (1 - \lambda_s)/(1 - \lambda_{k+1})$ , to be bounded from above by a constant  $K$ , we can only guarantee this by having  $k$  large enough so that  $1/(1 - \lambda_{k+1}) \leq K$ , or equivalently,  $\lambda_{k+1} \leq 1 - 1/K$ . In other words, we would have to select the rank  $k$  large enough such that

$$\lambda_{k+1} \leq 1 - \frac{1}{K}. \quad (30)$$

Of course, the required rank  $k$  depends primarily on the eigenvalue decay of the  $\lambda_i$ 's. In general, however, this means that the method will require a sufficient number of eigenvectors to be computed and that this number must be increased if we wish to decrease the spectral condition number to a given value. For problems arising from PDEs, it is expected that in order to keep the spectral condition number constant,  $k$  must have to be increased as the problem sizes increase.

## 5. PRACTICAL IMPLEMENTATION

In this section, we will address the implementation details for computing and applying an SLR preconditioner.

### 5.1. Computing the low-rank approximations

One of the key issues in setting up the preconditioner (16) is to extract a low-rank approximation to the matrix  $C^{-1}E^T B^{-1}E$ . Assuming that  $C$  is SPD, we use the Lanczos algorithm [23] on the matrix  $L^{-1}E^T B^{-1}EL^{-T}$ , where  $L$  is obtained from an incomplete Cholesky factorization of  $C$ . In the case when only a few extreme eigenpairs are needed, the Lanczos algorithm can efficiently approximate these without forming the matrix explicitly because the procedure only requires the matrix for performing the matrix-vector product of the form  $y = L^{-1}E^T B^{-1}EL^{-T}x$ , which requires solves with the block diagonal matrix  $B$ . The solves with  $B$  and  $C$  are performed inexactly as described in the next section. As it is well known, in the presence of rounding errors, orthogonality in the Lanczos procedure is quickly lost, and a form of reorthogonalization is needed in practice. In our approach, the partial reorthogonalization scheme [24, 25] is used. The cost of this step will not be an issue to the overall performance when a small number of steps are performed to approximate a few eigenpairs.

### 5.2. Solves with $\mathbf{B}$ and $\mathbf{C}$

A solve with the matrix  $B$  amounts to  $p$  local and independent solves with the matrices  $B_i$ ,  $i = 1, \dots, p$ . These can be carried out efficiently either by a direct solver or by Krylov subspace methods with more traditional ILU preconditioners for example. On the other hand, the matrix  $C$ , associated with all the interface unknowns, often has some diagonal dominance properties for discretized PDEs problems, so that ILU-based methods can typically work well. However, as we will see, this method cannot scale well. For solving large problems, especially the ones issued from 3-D PDEs, we need to have a large number of subdomains such that the local solves with  $B_i$  can be inexpensive. As a result, the number of interface unknowns will increase rapidly with the problem size and the number of subdomains.

For example, consider a five-point stencil discretization of the 2D Laplacian on a regular mesh of size  $n_x \times n_x$  and assume a 2D geometric partitioning is used. As shown in the leftmost side of the illustration in Figure 9, when we start with four partitions, we have about  $2n_x$  interface points ( $2n_x - 1$  to be exact). Each time we partition further, halving each subdomain in each direction, we multiply the number of subdomains by a factor of four and add roughly  $2^{k-1}n_x$  interface points in each direction at the  $k$ th division ( $k = 1$  corresponds to the initial partitioning on the left side of

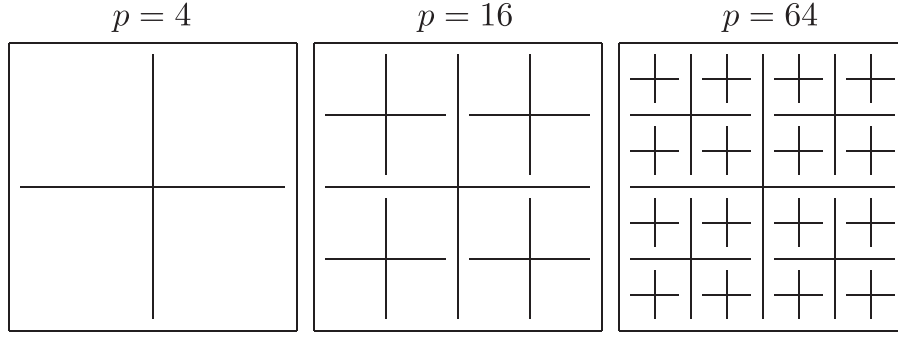


Figure 9. A two-dimensional finite difference mesh recursively partitioned into 64 subdomains.

Figure 9. At the  $k$ th division, we would have  $p = (2^k)^2$  subdomains and  $1 + 2 + \dots + 2^{k-1} = 2^k - 1$  lines in each direction, that is,  $\approx 2^{k+1}n_x$  interface points, when  $k$  is large. So the number of interface points evolves like  $2\sqrt{p}n_x = 2\sqrt{pN}$  where  $N = n_x^2$  is the total number of points. For a 3D mesh of size  $n_x \times n_x \times n_x$ , with a 3-D partitioning, the number of interface points is about  $3(\sqrt{p}N)^{2/3}$ , where  $N = n_x^3$ .

An alternative is to apply the SLR method recursively to  $C$ . This requires that the interface unknowns be ordered in a way that  $C$  has the same structure as that of  $A$ , that is, that the leading block is block diagonal. This is a property that can be satisfied by the hierarchical interface decomposition (HID) method discussed in [26]. This idea essentially yields a multilevel scheme of the SLR method that is currently being investigated by the authors. In the current SLR method, we simply use ILU factorizations for  $C$ .

### 5.3. Improving an Schur Low Rank preconditioner

One of the main weaknesses of standard, that is, ILU-type, preconditioners is that they are difficult to update. For example, suppose we compute a preconditioner to a given matrix and find that it is not accurate enough to yield convergence. In the case of ILUs, we would have essentially to start from the beginning. However, for SLR, improving a given preconditioner is essentially trivial. For example, the heart of the SLR method consists of obtaining a low-rank approximation the matrix  $H$  defined in (6) or (32). Improving this approximation would consist in merely adding a few more vectors (increasing  $k$ ), and this can be easily achieved in a number of ways, for example, by resorting to a form of deflation, without having to throw away the vectors already computed. Details of incrementally updating low-rank correction based preconditioners can be found in [15, §2.6].

## 6. EXTENSION TO NONSYMMETRIC MATRICES

Consider the nonsymmetric equivalent of (4),

$$A = \begin{pmatrix} B & F \\ E^T & C \end{pmatrix} = \begin{pmatrix} I & \\ E^T B^{-1} & I \end{pmatrix} \begin{pmatrix} B & F \\ & S \end{pmatrix}, \quad \text{with } S = C - E^T B^{-1} F.$$

Let  $C = LU$  be the LU factorization of  $C$ , so that we have

$$S = L(I - L^{-1}E^T B^{-1}FU^{-1})U \equiv L(I - H)U, \quad (31)$$

as in (5). Then, let the *complex* Schur decomposition of  $H$  be

$$H = L^{-1}E^T B^{-1}FU^{-1} = WRW^H, \quad (32)$$

where  $W$  is unitary and  $R$  is an upper triangular matrix that contains the eigenvalues of  $H$ , denoted by  $\lambda_i$ , on the diagonal. It follows by substituting (32) in (31) that

$$S = L(I - WRW^H)U = LW(I - R)W^H U, \quad (33)$$

and the inverse of  $S$  is then

$$S^{-1} = U^{-1} [W(I - R)^{-1} W^H] L^{-1}.$$

Let  $\tilde{R}$  be a triangular matrix that is an approximation to  $R$ , with the diagonal entries  $\tilde{R}_{i,i} = \tilde{\lambda}_i$ ,  $i = 1, 2, \dots, s$ . Then, an approximate inverse of  $S$  can be obtained from

$$\tilde{S}^{-1} = U^{-1} W (I - \tilde{R})^{-1} W^H L^{-1} = U^{-1} [I + W ((I - \tilde{R})^{-1} - I) W^H] L^{-1}. \quad (34)$$

Analogously to Proposition 4.1, the following proposition shows the eigenvalues of  $S\tilde{S}^{-1}$ .

*Proposition 6.1*

Let  $S$  and  $\tilde{S}^{-1}$  be given by (33) and (34), respectively. The Schur decomposition of  $L^{-1} S \tilde{S}^{-1} L$  is given by the following:

$$L^{-1} S \tilde{S}^{-1} L = W \Sigma W^H, \quad (35)$$

where  $\Sigma = (I - R)(I - \tilde{R})^{-1}$  is an upper triangular matrix that has  $\sigma_1, \dots, \sigma_s$  on the diagonal with  $\sigma_i$  defined by the following:

$$\sigma_i = \frac{1 - \lambda_i}{1 - \tilde{\lambda}_i}, \quad i = 1, \dots, s. \quad (36)$$

*Proof*

By multiplying (33) with (34), we have

$$S \tilde{S}^{-1} = LW(I - R)W^H U U^{-1} W (I - \tilde{R})^{-1} W^H L^{-1} = LW(I - R)(I - \tilde{R})^{-1} W^H L^{-1}.$$

Therefore,

$$L^{-1} S \tilde{S}^{-1} L = W(I - R)(I - \tilde{R})^{-1} W^H = W \Sigma W^H,$$

$$\text{with } \sigma_i = (1 - R_{ii}) / (1 - \tilde{R}_{ii}) = (1 - \lambda_i) / (1 - \tilde{\lambda}_i). \quad \square$$

Clearly, the eigenvalues of  $S\tilde{S}^{-1}$  are  $\sigma_i$ ,  $i = 1, \dots, s$ . As before, we let the first  $k$  diagonal entries of  $\tilde{R}$  match those of  $R$  and all the others equal to a constant  $\theta$ , that is,

$$\begin{cases} \tilde{\lambda}_i = \lambda_i & i = 1, \dots, k \\ \tilde{\lambda}_i = \theta & i = k + 1, \dots, s \end{cases}. \quad (37)$$

With this, the eigenvalues of  $S\tilde{S}^{-1}$  become

$$\begin{cases} 1 & i = 1, \dots, k \\ (1 - \lambda_i) / (1 - \theta) & i = k + 1, \dots, s \end{cases}. \quad (38)$$

In practice, we need not to compute the entire Schur decomposition of  $H$ . Only the  $k \times k$  leading principal submatrix of  $R$  and the first  $k$  Schur vectors are needed. This is shown in the following proposition, where we denote by  $S_{k,\theta}^{-1}$  the resulting approximate inverse of  $S$ .

*Proposition 6.2*

Let  $H = WRW^H$  be the complex Schur decomposition of  $H$ .  $R_k$  is the  $k \times k$  leading principal submatrix of  $R$ , and  $W_k$  have the first  $k$  Schur vectors as columns. With  $\tilde{R}$  defined by

$$\tilde{R} = \begin{pmatrix} R_k & \\ & \theta I \end{pmatrix},$$

the following expression for  $S_{k,\theta}^{-1}$  holds

$$S_{k,\theta}^{-1} = \frac{1}{1 - \theta} C^{-1} + U^{-1} W_k [(I - R_k)^{-1} - (1 - \theta)^{-1} I] W_k^H L^{-1}. \quad (39)$$

*Proof*

The proof is a straightforward extension of that of Proposition 4.1.  $\square$

### 6.1. Computing the low-rank approximations

We use the Arnoldi process [27] to compute the low-rank matrices in (39). When the low-rank approximation is associated with the extreme eigenvalues of the matrix  $H$ , that is, the eigenvalues on the periphery of the spectrum, this computation can be efficient. Performing  $m$ ,  $m > k$ , steps of the Arnoldi process on  $H$ , leads to the standard Krylov factorization equations:

$$\begin{aligned} HU_m &= U_m H_m + h_{m+1,m} u_{m+1} e_m^T, \\ U_m^T H U_m &= H_m, \end{aligned}$$

where  $u_i$ , for  $i = 1, \dots, m+1$ , are orthonormal and  $U_m = [u_1, \dots, u_m]$ . The eigenvalues of  $H_m$  are good estimates of the extreme eigenvalues of  $H$ . Let the complex Schur decomposition of  $H_m$  be

$$Q^H H_m Q = T. \quad (40)$$

Furthermore, the  $k$  eigenvalues that we want to deflate can be ordered to the first  $k$  entries on the diagonal of  $T$  [28, 29]. This ordering method is provided by the LAPACK [30] subroutine DTRSEN. Thus, the low-rank matrices in (39) can be approximated by the following:

$$R_k \approx T_{1:k,1:k} \quad \text{and} \quad W_k \approx U_m Q_{:,1:k}.$$

When used in finite precision arithmetic, the basis vectors computed in the Arnoldi procedure with the Gram–Schmidt (G-S) orthogonalization process will lose their orthogonality [31] at some point. A simple remedy is to perform a reorthogonalization. As indicated in [32, 33], only one reorthogonalization step with the classical G-S algorithm is needed to preserve the orthogonality to the machine precision level. Note that the orthogonality issue of the Arnoldi vectors typically does not have a big impact in the GMRES method, for which the modified G-S performs well. On the other hand, when computing the low-rank correction in the SLR preconditioner, we cannot forego reorthogonalization in the Arnoldi process.

The results shown in section also apply to the symmetric case when both  $A$  and  $C$  are indefinite. The extension to complex non-Hermitian matrices is straightforward.

## 7. NUMERICAL EXPERIMENTS

The experiments were conducted on a machine at Minnesota Supercomputing Institute, equipped with two Intel Xeon X5560 processors (8 MB Cache, 2.8 GHz, quad-core) and 24 GB of main memory. A preliminary implementation of the SLR preconditioner was written in C/C++, and the code was compiled by the Intel C compiler with the -O2 optimization level. BLAS and LAPACK routines from Intel Math Kernel Library (MKL) were used to enhance the performance on multiple cores. Thread-level parallelism was realized by OpenMP [34].

For symmetric linear systems, the accelerators used were the conjugate gradient (CG) method for the SPD cases, and the generalized minimal residual (GMRES) method with a restart dimension of 40, denoted by GMRES(40) for the indefinite cases. For the nonsymmetric cases, GMRES(40) was used. Two types of preconditioning methods were tested in our experiments: ILU-type preconditioners with threshold dropping that include the incomplete Cholesky factorization (ICT), the incomplete LDL factorization (ILDLT), and ILUTP, and the SLR method. ILUTP uses a dual threshold dropping scheme with column pivoting [35], and its symmetric variants ICT and ILDLT compute ILU factorizations in the Crout form of Gaussian eliminations [36].

For all the problems, we used the graph partitioner PartGraphRecursive from METIS [37, 38] to partition the domains. The time for the graph partitioning will not be included in the time of building the preconditioners. For each subdomain  $i$ ,  $B_i$  was reordered by the approximate minimum



Table I. The required ranks of the SLR method for bounding the condition number of  $S\tilde{S}^{-1}$  by  $K$  and with different numbers of subdomains, for two-dimensional/three-dimensional Laplacians with  $c = 0$  in (41).

(a) ranks change with grid sizes				(b) condition numbers change with ranks						
Grid	$K \approx 36$ rk	Grid	$K \approx 12$ rk	1024 <sup>2</sup>	rk $K$	5 168	10 114	20 65	30 46	40 36
128 <sup>2</sup>	3	25 <sup>3</sup>	1	100 <sup>3</sup>	rk	5	10	20	30	40
256 <sup>2</sup>	6	40 <sup>3</sup>	4		$K$	32	25	17	14	13
512 <sup>2</sup>	16	64 <sup>3</sup>	12							
1024 <sup>2</sup>	40	100 <sup>3</sup>	42							

(c) ranks change with numbers of subdomains						
1024 <sup>2</sup>	nd	2	4	8	16	32
$K \approx 36$	rk	9	19	40	56	90
100 <sup>3</sup>	nd	8	16	32	64	128
$K \approx 12$	rk	21	30	42	56	75

degree (AMD) ordering [39–41], and ILU was used as the local solver. Prior to computing the ILU preconditioners, the AMD ordering was applied to the original matrix. In the SLR method, matrix  $C$  was factored by ILU. In the Lanczos/Arnoldi procedure, we set the maximum number of steps as five times the number of requested eigenvalues.

Based on the experimental results, we can state that in general, building an SLR preconditioner, especially when using larger ranks, requires much more time than an ICT/ILDLT preconditioner. Nevertheless, experimental results indicated that the SLR preconditioner is more robust and can also achieve great time savings in the iterative phase. In the case of systems with a large number of right-hand sides, expensive but effective preconditioners may be justified because their cost is amortized. In this section, we first report on the results of solving symmetric linear systems from a 2D/3D PDE on regular meshes. Then, we report the performance of the SLR preconditioner for solving complex non-Hermitian linear systems from 2D and 3D Helmholtz equations. Last, we will show results for solving a sequence of general sparse symmetric linear systems. Except for the Helmholtz problems, the iterations were stopped whenever the residual norm had been reduced by eight orders of magnitude or the maximum number of iterations allowed, which is 300, was exceeded. The results are summarized in Tables II – VI and VIII, where all times are reported in seconds. When comparing the preconditioners, the following factors are considered: (1) fill-ratio, that is, the ratio of the number of nonzeros required to store a preconditioner to the number of nonzeros in the original matrix, (2) time for building preconditioners, (3) the number of iterations, and (4) time for the iterations. In all tables, ‘F’ indicates non-convergence within the maximum allowed number of steps.

### 7.1. Two-dimensional/three-dimensional model problems

We consider 2D and 3D PDE problems,

$$\begin{aligned} -\Delta u - cu &= f \text{ in } \Omega, \\ u &= \phi(x) \text{ on } \partial\Omega, \end{aligned} \quad (41)$$

where  $\Omega = (0, 1)^2$  and  $\Omega = (0, 1)^3$  are the domains and  $\partial\Omega$  is the boundary. We take the five-point or seven-point centered difference approximation on the regular meshes.

To begin with, we examine the required ranks of the SLR method in order to bound the spectral condition number of the matrix  $S\tilde{S}^{-1}$  by a constant  $K$ . Recall from (30) that this requires that the  $(k + 1)$ -st largest eigenvalue,  $\lambda_{k+1}$ , of the matrix  $C^{-1}E^T B^{-1}E$  be less than  $1 - 1/K$ . The results for 2D/3D Laplacians are shown in Table I (a), where eight subdomains were used for the 2D case and it was 32 for the 3D case. From there, we can see that for the 2D problems, and the required rank is about doubled when the step-size is reduced by half, while for the 3D cases, the rank needs

Table II. Performance of the ICT and the SLR preconditioners for solving SPD linear systems from the two-dimensional/three-dimensional PDE with CG.

Grid	ICT-CG				SLR-CG					
	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
256 <sup>2</sup>	4.5	0.074	51	0.239	32	16	4.3	0.090	67	0.145
512 <sup>2</sup>	4.6	0.299	97	1.93	64	32	4.9	0.650	103	1.01
1024 <sup>2</sup>	5.4	1.44	149	14.2	128	32	5.7	5.23	175	7.95
40 <sup>3</sup>	4.4	0.125	25	0.152	32	16	4.0	0.182	31	0.104
64 <sup>3</sup>	6.8	0.976	32	1.24	64	32	6.3	1.52	38	0.633
100 <sup>3</sup>	7.3	4.05	47	7.52	128	32	6.5	5.50	67	4.48

ICT, incomplete Cholesky factorization; CG, conjugate gradient; SLR, Schur low rank.

to be increased by a factor of roughly 3.5. Table I (b) shows how the condition number varies with the rank for the largest 2D and 3D problems. Table I (c) shows that the required rank for a fixed condition number increases with the number of subdomains, which is indicated by ‘nd’.

In the next set of experiments, we solve (41) with  $c = 0$ , so that the coefficient matrices are SPD and we use the SLR preconditioner with the CG method. Numerical experiments were carried out to compare the performance of the SLR preconditioner with the ICT preconditioner. The results are shown in Table II. The sizes of the grids, the fill-ratios (fill), the numbers of iterations (its), the time for building the preconditioners (p-t), and the time for iterations (i-t) are tabulated. For the SLR preconditioners, the number of subdomains (nd) and the rank (rk) are also listed. The fill-ratios of the three preconditioners were controlled to be roughly equal. The ICT factorizations were used for the solves with the matrices  $B$  and  $C$  in the SLR method. As shown in Table II, we tested the problems on three 2D grids and three 3D grids of increasing sizes, where for the SLR method, the domain was partitioned into 32, 64, and 128 subdomains, respectively, and the ranks 16 or 32 were used in the SLR preconditioners.

Compared with the ICT preconditioner, building an SLR preconditioner required more CPU time (up to four times more for the largest 2D case). For these problems, the SLR-CG method achieved convergence in slightly more iterations than those with the ICT preconditioner, but SLR still achieved performance gains in terms of significantly reduced iteration times. The CPU time for building an SLR preconditioner is typically dominated by the cost of the Lanczos algorithm. Furthermore, this cost is actually governed by the cost of the solves with  $B_i$ 's and  $C$ , which are required at each iteration. Moreover, when the rank  $k$  used is large, the cost of reorthogonalization will also become significant. Some simple thread-level parallelism has been exploited using OpenMP for the solves with the  $B_i$ 's, which can be performed independently. The multi-threaded MKL routines also helped speedup the vector operations in the reorthogonalizations. We point out that there is a room for substantial improvements in the performance of these computations. In particular, they are very suitable for the SIMD type parallel machines such as computers equipped with GPUs or with the Intel Xeon Phi processors. These features have not yet been implemented in the current code.

Next, we consider solving the symmetric indefinite problems by setting  $c > 0$  in (41), which corresponds to shifting the discretized negative Laplacian (a positive definite matrix) by subtracting  $sI$  with a certain  $s > 0$ . In this set of experiments, we solve the 2D problems with  $s = 0.01$  and the 3D problems with  $s = 0.05$ . The SLR method is compared with ILDLT with GMRES (40).

Results are shown in Table III. For most problems, the ILDLT/GMRES failed even with high fill-ratios. In contrast, the SLR method appears to be more effective, achieving convergence for all cases, and great savings in the iteration time. In contrast with the SPD case, a few difficulties were encountered. For the 2D problems, an SLR preconditioner with a large number of subdomains (say, 64 or 128) often failed to converge. As a result, the sizes of the subdomains were still quite large, and factoring the matrices  $B_i$ 's was quite expensive in terms of both the CPU time and the memory requirement. Furthermore, for both the 2D and 3D problems, approximations of higher ranks were required compared with those used in the SPD cases. This only increased the memory requirement

Table III. Performance of the ILDLT and the SLR preconditioners for solving symmetric indefinite linear systems from the two-dimensional/three-dimensional PDE with GMRES (40).

Grid	ILDLT-GMRES				SLR-GMRES					
	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
256 <sup>2</sup>	6.5	0.125	F	—	8	32	6.4	0.213	33	0.125
512 <sup>2</sup>	8.4	0.702	F	—	16	64	7.6	2.06	93	1.50
1024 <sup>2</sup>	12.6	5.14	F	—	8	128	10.8	24.5	50	4.81
40 <sup>3</sup>	6.7	0.249	54	0.540	64	32	6.7	0.490	23	0.123
64 <sup>3</sup>	9.0	1.39	F	—	128	64	9.1	3.94	45	1.16
100 <sup>3</sup>	14.7	10.9	F	—	128	180	14.6	62.9	88	13.9

ILDLT, incomplete LDL factorization; SLR, Schur low rank.

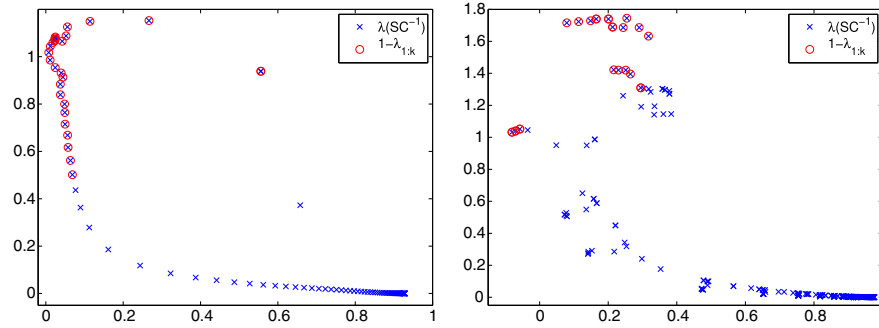


Figure 10. The eigenvalues of  $SC^{-1}$  and the  $k$  eigenvalues associated with the low-rank correction for two-dimensional and three-dimensional Helmholtz problems on  $100^2$  (left) and  $20^3$  (right) meshes.

Table IV. Results of the SLR and ILUTP preconditioners for solving two-dimensional Helmholtz problems with GMRES (40). 16 points per wavelength.

$\omega/(2\pi)$	$q$	$n = N^2$	SLR				ILUTP	
			nd	rk	fill	its	fill	its
6.25	16	100 <sup>2</sup>	4	32	3.3	27	3.3	55
12.5	16	200 <sup>2</sup>	8	64	6.8	28	6.7	36
25	16	400 <sup>2</sup>	8	64	8.7	61	8.6	270
31.25	16	500 <sup>2</sup>	8	128	9.6	44	9.7	F
37.5	16	600 <sup>2</sup>	8	200	9.6	39	10.1	F
43.75	16	700 <sup>2</sup>	8	250	10.1	46	11.7	F

Left: the solution with  $\frac{\omega}{2\pi} = 25$ . SLR, Schur low rank.

slightly, but it significantly increased the CPU time required by the Lanczos algorithm. An example is the largest 3D problem in Table III, where a rank of 180 was used.

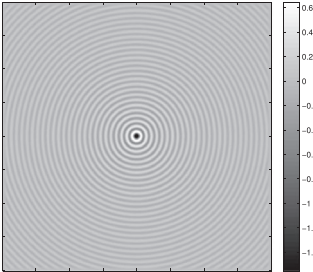
## 7.2. Two-dimensional/three-dimensional Helmholtz problems

This section, discusses the performance of the SLR preconditioner for solving problems from a 2D/3D Helmholtz equation of the form

$$\left(-\Delta - \frac{\omega^2}{v(x)^2}\right)u(x, \omega) = s(x, \omega) \quad (42)$$

where  $\Delta$  is the Laplacian operator,  $\omega$  is the angular frequency,  $v(x)$  is the velocity field, and  $u(x, \omega)$  is called the time-harmonic wave field solution to the external forcing term  $s(x, \omega)$ . The domain of interest is the unit box,  $(0, 1)^d$  with  $d = 2, 3$ , where we take the five-point or the seven-point

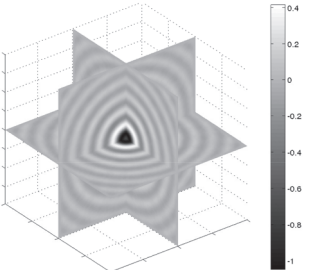
Table V. Results of the SLR and ILUTP preconditioners for solving 2-D Helmholtz problems with GMRES (40).



$\omega/(2\pi)$	$q$	$n = N^2$	SLR				ILUTP	
			nd	rk	fill	its	fill	its
12.5	8	$100^2$	4	64	4.9	10	4.8	58
25	8	$200^2$	8	128	6.6	21	6.9	F
50	8	$400^2$	8	128	8.3	100	8.2	F
62.5	8	$500^2$	8	256	9.2	84	9.3	F
75	8	$600^2$	8	400	10.3	153	12.2	F
87.5	8	$700^2$	8	500	10.6	236	10.7	F

SLR, Schur low rank. Eight points per wavelength. Left: the solution with  $\frac{\omega}{2\pi} = 50$ .

Table VI. Results of the SLR and ILUTP preconditioners for solving 3-D Helmholtz problems with GMRES(40).



$\omega/(2\pi)$	$q$	$n = N^3$	SLR				ILUTP	
			nd	rk	fill	its	fill	its
2.5	8	$20^3$	4	32	3.4	28	3.3	33
3.75	8	$30^3$	8	32	4.1	49	3.9	60
5	8	$40^3$	16	32	4.5	88	4.5	125
6.25	8	$50^3$	16	32	8.5	85	8.7	145
7.5	8	$60^3$	16	128	11.6	88	12.2	F
10	8	$80^3$	16	128	15.6	176	16.6	F

SLR, Schur low rank. Eight points per wavelength. Left: the solution with  $\frac{\omega}{2\pi} = 10$ .

centered difference approximation with regular meshes. The PML boundary condition is used at all sides. The resulting coefficient matrices are *complex non-Hermitian*. In (42), we assume that the mean of  $v(x)$  is equal to 1. Then,  $\omega/(2\pi)$  is the wave number, and  $\lambda = 2\pi/\omega$  is the wavelength. When the sampling rate of  $q$  points per wavelength is used, the number of mesh points for each dimension is  $N = q\omega/(2\pi)$ , and the coefficient matrix is of the order  $n = N^d$ , which is a negative Laplacian shifted by  $-sI$  with  $s \approx \omega^2/N^2 = (2\pi/q)^2$ .

To begin with, we examine the eigenvalues of the preconditioned Schur complement with the SLR preconditioner. Figure 10 shows the eigenvalues of  $SC^{-1}$  and the eigenvalues that correspond to the low-rank correction,  $\lambda_{1:k}$ . Observe that in both cases, many eigenvalues of  $SC^{-1}$  are clustered around 1, and that most are far away from the origin. Thus, we chose the eigenvalues that are far from 1 and the corresponding Schur vectors for the low-rank correction. These eigenvalues will be deflated to 1, while the remaining ones will be scaled by a scalar  $(1 - \theta)^{-1}$ . Recall that the eigenvalues of  $SC^{-1}$  are  $1 - \lambda_i$ , where  $\lambda_i$  are the eigenvalues of the matrix  $H$  in (32). Therefore, the eigenvalues of  $SC^{-1}$  that are far from 1 correspond to the eigenvalues of  $H$  that are large in magnitude. These eigenvalues and the Schur vectors can be efficiently computed by the Arnoldi process.

We first tested the performance of the SLR preconditioner compared with the ILUTP preconditioner for solving the 2D Helmholtz problem discretized with  $q = 16$  points per wavelength. The corresponding shift is about  $-0.15I$ . GMRES (40) was used with the relative residual tolerance set to be  $10^{-5}$ . For the SLR preconditioner,  $S_{k,\theta}^{-1}$ , in (39), we set  $\theta = \lambda_{k+1}$ . The results are shown in Table IV. The picture on the left shows the solution with  $\omega/(2\pi) = 25$ . We tested the problem on six meshes of increasing sizes, where the wave number  $\omega/(2\pi)$  is proportional to  $N$ . For all the problems, convergence was achieved with the SLR method using higher ranks for larger problems, whereas the ILUTP preconditioner failed for the three large problems.

Next, we consider 2D Helmholtz problems with  $q = 8$ . This set of problems is harder than the previous one, as indicated by the higher wave numbers and the larger shift, which is about  $-0.62I$ , for the coefficient matrix. The results are shown in Table V. The picture on the left shows the solution with  $\omega/(2\pi) = 50$ . Compared with the cases with  $q = 16$ , higher ranks were used, and more iterations were required for convergence. For these problems, the SLR method still outperformed the ILUTP preconditioner. It succeeded for all the cases and required fewer iterations for the smallest case where the ILUTP method also worked.

Next, we solve 3D Helmholtz problems on six cubes with  $q = 8$ . The results are shown in Table VI. The picture on the left shows the solution with  $\omega/(2\pi) = 10$ . Compared with the 2D problems, larger numbers of subdomains ('nd') were used in order to keep the cost of factoring the matrices  $B_i$  inexpensive. The ranks required by the SLR method were smaller than those for the 2D problems. The SLR method showed a performance that is superior to that of the ILUTP preconditioner. It led to convergence for all the cases and required fewer iterations than ILUTP when both worked.

Table VII. Names, orders ( $N$ ), NNZ, positive definiteness, and short descriptions of the test matrices.

Matrix	$N$	NNZ	SPD	Description
Williams/cant	62,451	4,007,383	yes	FEM cantilever
UTEP/dubcova2	65,025	1,030,225	yes	2D/3D PDE problem
UTEP/dubcova3	146,689	3,636,643	yes	2D/3D PDE problem
Rothberg/cfd1	70,656	1,825,580	yes	CFD problem
Rothberg/cfd2	123,440	3,085,406	yes	CFD problem
Schmid/thermal1	82,654	574,458	yes	thermal problem
Schmid/thermal2	1,228,045	8,580,313	yes	thermal problem
Wissgott/parabolic_fem	525,825	3,674,625	yes	CFD problem
CEMW/tmt_sym	726,713	5,080,961	yes	electromagnetics problem
McRae/ecology2	999,999	4,995,991	yes	landscape ecology problem
Lin/Lin	256,000	1,766,400	no	structural problem
Cote/vibrobox	12,328	301,700	no	vibroacoustic problem
Cunningham/qa8fk	66,127	1,660,579	no	3D acoustics problem
Koutsovasilis/F2	71,505	5,294,285	no	structural problem
GHS_indef/helm2d03	392,257	2,741,935	no	2D Helmholtz problem

SPD, symmetric positive definite; NNZ, numbers of nonzeros.

Table VIII. Performance of the ICT/ILDLT and the SLR preconditioners for solving general symmetric linear systems.

Matrix	ICT/ILDLT					SLR				
	fill	p-t	its	i-t	nd	rk	fill	p-t	its	i-t
cant	4.7	3.87	150	9.34	32	90	4.9	5.58	82	1.92
dubcova2	2.7	0.300	47	0.492	16	32	2.8	0.280	19	0.080
dubcova3	2.2	1.01	46	1.44	16	32	1.8	0.677	19	0.212
cfd1	6.9	2.89	295	11.9	32	32	6.9	2.13	64	1.07
cfd2	9.9	13.5	F	—	32	80	8.8	7.62	178	5.75
thermal1	5.1	0.227	68	0.711	16	32	5.0	0.277	59	0.231
thermal2	6.9	5.10	178	39.3	64	90	6.6	14.8	184	15.0
para_fem	6.1	2.04	58	4.68	32	80	6.9	6.05	86	3.03
tmt_sym	6.0	1.85	122	11.6	64	80	5.9	6.61	127	5.23
ecology2	8.4	2.64	142	18.5	32	96	8.0	12.3	90	5.58
Lin	11	1.93	F	—	64	64	9.9	3.78	73	1.75
vibrobox	6.0	0.738	F	—	4	64	3.8	0.437	226	0.619
qa8fk	4.2	0.789	22	0.507	16	64	4.5	1.94	28	0.309
F2	5.1	9.66	F	—	8	80	3.9	6.25	72	2.14
helm2d03	14	14.4	F	—	16	128	11	11.9	63	2.63

SLR, Schur low rank; ICT, incomplete Cholesky factorization.

### 7.3. General symmetric matrices

We selected 15 symmetric matrices from the University of Florida sparse matrix collection [42] that are listed in Table VII. Among these 10 matrices are SPD matrices, and five matrices are symmetric indefinite. If a right-hand side is not provided, an artificial one was created by  $b = Ae$ , where  $e$  is a random vector of unit two-norm.

Table VIII shows the performance of the three preconditioning methods. The CG method and the GMRES method achieved convergence for all the cases with the SLR preconditioner, whereas for many cases, they failed to converge with the ICT and ILDLT preconditioners. Similar to the experiment results for the PDE problems, the SLR preconditioner often required more CPU time to build than the other two counterparts when the memory requirements of these methods are similar. In the iteration phase, the SLR preconditioner required fewer iterations for most of the problems, and achieved significant CPU time savings for all the cases where the other two methods also worked.

## 8. CONCLUSION

This paper presented a preconditioning method, named SLR, based on a Schur complement approach with low-rank corrections for solving general sparse linear systems. Like the method presented in [15], the new method uses a low-rank approximation to build a preconditioner, exploiting some decay property of eigenvalues. The major difference with [15] is that SLR is not recursive. It focuses on the Schur complement in any standard domain decomposition framework and tries to approximate its inverse by exploiting low-rank approximations. As a result, the method is much easier to implement.

Experimental results indicate that in terms of iteration times, the proposed preconditioner can be a more efficient alternative to the ones based on incomplete factorizations, namely, the ILU-type or block ILU-type methods for SPD systems. Moreover, this preconditioner appears to be more robust than the incomplete factorization-based methods for indefinite problems. Recall that ILU-based methods often deliver unstable and, in some cases, quite dense factors when the original matrix is highly indefinite, and this renders them ineffective for such cases. In contrast, SLR is essentially a form of approximate inverse technique and as such it is not prone to these difficulties. The results for the shifted Laplacian matrices and the Helmholtz problems support this finding. On the negative side, building an SLR preconditioner can be time consuming, although several mitigating factors should be taken into account. These are similar to those pointed out in [15], which also exploits low-rank approximation, and we summarize them here. The first is that a big part of the computations to build the SLR preconditioner, which is the computation of the approximate eigenvectors, can be easily vectorized and this is especially attractive for massively parallel machines, such as those equipped with GPUs or with the Intel Xeon Phi processors. The set-up phase is likely to be far more advantageous than a factorization-based one, which tends to be much more sequential; see, for example, [1]. The second is that there are situations in which many systems with the same matrix must be solved in which case more expensive but more effective preconditioners may be justified as their cost will be amortized. Finally, as discussed in Section 5.3, these preconditioners are more easily updatable than traditional ones based on ILU-type factorizations.

## APPENDIX

Let

$$T = \begin{pmatrix} 2a & -1 & & & \\ -1 & 2a & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2a \end{pmatrix}$$

and

$$T = \begin{pmatrix} 1 & & & \\ -d_1^{-1} & 1 & & \\ & -d_2^{-1} & \ddots & \\ & & \ddots & \ddots \\ & & & -d_n^{-1} & 1 \end{pmatrix} \begin{pmatrix} d_1 & -1 & & \\ & d_2 & -1 & \\ & & \ddots & \ddots \\ & & & \ddots & -1 \\ & & & & d_n \end{pmatrix}$$

be the LU factorization of  $T$ . We are interested in  $d_n^{-1}$ . If we solve  $Tx = e_n$  where  $e_n$  is the  $n$ th canonical basis vector for  $\mathbb{R}^n$ , and  $x = [\xi_0, \dots, \xi_{n-1}]^T$ , then clearly  $\xi_{n-1} = 1/d_n$ , which is what we need to calculate. Let  $\xi_k = U_k(a)$ , for  $k = 0, 1, \dots, n-1$ , where  $U_k$  is the  $k$ th degree Chebyshev polynomial of the second kind. These polynomials satisfy the recurrence relation:  $U_{k+1}(t) = 2tU_k(t) - U_{k-1}(t)$ , starting with  $U_0(t) = 1$  and  $U_1(t) = 2t$ . Then clearly, equations  $k = 1, \dots, n-1$  of the system  $Tx = e_n$  are satisfied. For the last equation, we obtain  $U_n(a)$  instead of the wanted value of 1. Scaling  $x$  by  $U_n(a)$  yields the result  $1/d_n = \xi_{n-1} = U_{n-1}(a)/U_n(a)$ .

#### ACKNOWLEDGEMENTS

The authors would like to thank Jianlin Xia for providing the test cases of Section 7.2 and the anonymous reviewers for their valuable comments and suggestions. The authors are grateful to the University of Minnesota Supercomputing Institute for providing them with computational resources and assistance with the computations. This work performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 (LLNL-JRNL-679681), and supported by Minnesota Supercomputing Institute and NSF under Contract NSF/DMS-1216366, NSF/DMS-1521573.

#### REFERENCES

1. Li R, Saad Y. GPU-accelerated preconditioned iterative linear solvers. *The Journal of Supercomputing* 2013; **63**: 443–466.
2. Benzi M, Meyer CD, Tũma. A sparse approximate inverse preconditioner for the conjugate gradient method. *SIAM Journal on Scientific Computing* 1996; **17**:1135–1149.
3. Benzi M, Tũma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1998; **19**(3):968–994.
4. Grote MJ, Huckle T. Parallel preconditionings with sparse approximate inverses. *SIAM Journal on Scientific Computing* 1997; **18**:838–853.
5. Hackbusch W. A sparse matrix arithmetic based on  $\mathcal{H}$ -matrices. Part I: Introduction to  $\mathcal{H}$ -matrices. *Computing* 1999; **62**:89–108.
6. Hackbusch W, Khoromskij BN. A sparse  $\mathcal{H}$ -matrix arithmetic. Part II: application to multi-dimensional problems. *Computing* 2000; **64**:21–47.
7. Chandrasekaran S, Gu M, Pals T. A fast ULV decomposition solver for Hierarchically Semiseparable Representations. *SIAM Journal on Matrix Analysis and Applications* 2006; **28**(3):603–622.
8. Xia J, Chandrasekaran S, Gu M, Li XS. Fast algorithms for hierarchically semiseparable matrices. *Numerical Linear Algebra with Applications* 2010; **17**(6):953–976.
9. Ambikasaran S, Darve E. An  $\mathcal{O}(n \log n)$  fast direct solver for partial hierarchically semi-separable matrices. *Journal of Scientific Computing* 2013; **57**(3):477–501.
10. Wang S, de Hoop MV, Xia J. On 3D modeling of seismic wave propagation via a structured parallel multifrontal direct Helmholtz solver. *Geophysical Prospecting* 2011; **59**(5):857–873.
11. Le Borne S.  $\mathcal{H}$ -matrices for convection–diffusion problems with constant convection. *Computing* 2003; **70**(3): 261–274.
12. Le Borne S, Grasedyck L.  $\mathcal{H}$ -matrix preconditioners in convection-dominated problems. *SIAM Journal on Matrix Analysis and Applications* 2006; **27**(4):1172–1183.
13. Xia J, Gu M. Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. *SIAM Journal on Matrix Analysis and Applications* 2010; **31**(5):2899–2920.
14. Engquist B, Ying L. Sweeping preconditioner for the Helmholtz equation: hierarchical matrix representation. *Communications on Pure and Applied Mathematics* 2011; **64**(5):697–735.
15. Li R, Saad Y. Divide and conquer low-rank preconditioners for symmetric matrices. *SIAM Journal on Scientific Computing* 2013; **35**(4):A2069–A2095.
16. Li R, Saad Y. *Low-rank Correction Methods for Algebraic Domain Decomposition Preconditioners*, 2014. Submitted.
17. Grigori L, Nataf F, Yousef S. Robust algebraic Schur complement preconditioners based on low rank corrections. *Technical Report RR-8557*, INRIA, 2014.

18. Saad Y. *Iterative Methods for Sparse Linear Systems* (2nd edition). SIAM: Philadelphia, PA, 2003.
19. Saad Y, Suchomel B. ARMS: An algebraic recursive multilevel solver for general sparse linear systems. *Numerical Linear Algebra with Applications* 2002; **9**:359–378.
20. Mandel J. Balancing domain decomposition. *Communications in Numerical Methods in Engineering* 1993; **9**(3): 233–241.
21. Dohrmann CR. A preconditioner for substructuring based on constrained energy minimization. *SIAM Journal on Scientific Computing* 2003; **25**(1):246–258.
22. Mandel J, Dohrmann CR. Convergence of a balancing domain decomposition by constraints and energy minimization. *Numerical Linear Algebra with Applications* 2003; **10**(7):639–659.
23. Lanczos C. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards* 1950; **45**:255–282.
24. Parlett BN, Scott DS. The Lanczos algorithm with selective orthogonalization. *Mathematics of Computation* 1979; **33**(145):pp. 217–238.
25. Simon HD. The Lanczos algorithm with partial reorthogonalization. *Mathematics of Computation* 1984; **42**(165):115–142.
26. Hénon P, Saad Y. A parallel multistage ilu factorization based on a hierarchical graph decomposition. *SIAM Journal on Scientific Computing* 2006; **28**(6):2266–2293.
27. Arnoldi WE. The principle of minimized iterations in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics* 1951; **9**(17):17–29.
28. Bai Z, Demmel JW. On swapping diagonal blocks in real Schur form. *Linear Algebra and its Applications* 1993; **186**:75–95.
29. Stewart GW. Algorithm 506: Hqr3 and exchng: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix. *ACM Transactions on Mathematical Software* September 1976; **2**(3):275–280.
30. Anderson E, Bai Z, Bischof C, Blackford S, Demmel J, Dongarra J, Du Croz J, Greenbaum A, Hammarling S, McKenney A, Sorensen D. *LAPACK users' guide* (3rd edn). SIAM: Philadelphia, PA, 1999.
31. Giraud L, Langou J, Rozložník M, van den Eshof J. Rounding error analysis of the classical gram-schmidt orthogonalization process. *Numerische Mathematik* 2005; **101**(1):87–100.
32. Giraud L, Langou J, Rozložník M. The loss of orthogonality in the gram-schmidt orthogonalization process. *Computers & Mathematics with Applications* 2005; **50**(7):1069–1075. Numerical Methods and Computational Mechanics.
33. Watkins D. *The Matrix Eigenvalue Problem*. SIAM, 2007.
34. OpenMP Architecture Review Board. *OpenMP application program interface version 3.1*, 2011. Available from: <http://www.openmp.org/mp-documents/OpenMP3.1.pdf> [Accessed on July 2011].
35. Saad Y. Ilut: A dual threshold incomplete lu factorization. *Numerical Linear Algebra with Applications* 1994; **1**(4):387–402.
36. Li N, Saad Y. Crout versions of the ILU factorization with pivoting for sparse symmetric matrices. *Electronic Transactions on Numerical Analysis* 2006; **20**:75–85.
37. Karypis G, Kumar V. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 1998; **20**(1):359–392.
38. Karypis G, Kumar V. A parallel algorithm for multilevel graph partitioning and sparse matrix ordering. *Journal of Parallel and Distributed Computing* 1998; **48**(1):71–95.
39. Amestoy PR, Davis TA, Duff IS. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications* 1996; **17**(4):886–905.
40. Amestoy PR, Davis TA, Duff IS. Algorithm 837: an approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software* 2004; **30**(3):381–388.
41. Davis TA. *Direct methods for sparse linear systems (fundamentals of algorithms 2)*. Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2006.
42. Davis TA, Hu Y. The University of Florida Sparse Matrix Collection. *ACM Transactions on Mathematical Software* 2011; **38**(1):1:1–1:25.