A HIERARCHICAL LOW-RANK SCHUR COMPLEMENT PRECONDITIONER FOR INDEFINITE LINEAR SYSTEMS

GEOFFREY DILLON†, VASSILIS KALANTZIS†, YUANZHE XI†, AND YOUSEF SAAD†

Abstract. Nonsymmetric and highly indefinite linear systems can be quite difficult to solve by iterative methods. This paper combines ideas from the Multilevel Schur Low-Rank preconditioner developed by Y. Xi, R. Li, and Y. Saad [SIAM J. Matrix Anal., 37 (2016), pp. 235–259] with classic block preconditioning strategies in order to handle this case. The method to be described generates a tree structure $T$ that represents a hierarchical decomposition of the original matrix. This decomposition gives rise to a block structured matrix at each level of $T$. An approximate inverse of the original matrix based its block LU factorization is computed at each level via a low rank property that characterizes the difference between the inverses of the Schur complement and another block of the reordered matrix. The low rank correction matrix is computed by several steps of the Arnoldi process. Numerical results illustrate the robustness of the proposed preconditioner with respect to indefiniteness for a few discretized Partial Differential Equations (PDEs) and publicly available test problems.

Key words. block preconditioner, Schur complements, multilevel, low rank approximation, Krylov subspace methods, domain decomposition, Nested Dissection ordering.

AMS subject classifications. 65F08, 65F10, 65F50, 65N55, 65Y05

1. Introduction. This paper focuses on the solution of large nonsymmetric sparse linear systems

$$Ax = b$$

via Krylov subspace methods where $A \in \mathbb{C}^{n \times n}$ and $b \in \mathbb{C}^n$. When solving (1.1) it is often necessary to combine one of these Krylov methods with some form of preconditioning. For example, a right-preconditioning method would solve the system $AM^{-1}u = b, M^{-1}u = x$, in place of (1.1). Other variants include left and 2-sided preconditioners. Ideally, $M$ is an approximation to $A$ such that it is significantly easier to solve linear systems with it than with the original $A$.

A commonly used preconditioner is the Incomplete LU (ILU) factorization of $A$, where $A \approx LU = M$. ILU preconditioners can be very effective for certain types of linear systems. However, if the original matrix $A$ is poorly conditioned or highly indefinite ($A$ has eigenvalues on both sides of the imaginary axis) then ILU methods can fail due to very small pivots or unstable factors [12, 42]. Another disadvantage of ILU methods is their poor performance on high-performance computers, e.g., those with GPUs [33] or Intel Xeon Phi processors. Algebraic multigrid (AMG) is another popular technique for solving problems arising from discretized PDEs. Multigrid methods are provably optimal for a wide range of SPD matrices and also perform well in parallel. However, without specialization, multigrid will fail on even mildly indefinite problems. Sparse approximate inverses emerged in the 1990s as alternatives to ILU factorizations [9, 13, 22]. These methods were mostly abandoned due to their high cost both in terms of arithmetic and memory usage. A subsequent class of preconditioners were based on rank-structured matrices [10]. Two such types of matrices are $\mathcal{H}^2$-matrices [23, 24] and hierarchically semiseparable (HSS) matrices [49, 50, 51].

*This work was supported by NSF under grant DMS-1521573 and by the Minnesota Supercomputing Institute.

†Address: Department of Computer Science & Engineering, University of Minnesota, Twin Cities. {gdillon,kalan019,yxi,saad}@umn.edu
Both of these forms are the result of a partition of the original matrix where some of the off-diagonal blocks are approximated by low rank matrices. These ideas have been used to develop both sparse direct solvers and preconditioners [52]. Similarly, it also is possible to exploit preconditioners based on hierarchical LU factorizations [4].

In this paper we focus on approximate inverse preconditioners which are based on low rank corrections. Such approaches include the Multilevel Low-Rank (MLR) [32], the Schur complement low rank (SLR) preconditioner [34], and the Multilevel Schur complement Low-Rank (MSLR) preconditioner [46]. The idea behind the MSLR preconditioner is to combine a multilevel Hierarchical interface decomposition (HID) ordering [25] along with an efficient Schur complement approximation. This approach is shown to be much less sensitive to indefiniteness than the classical ILU and domain decomposition based methods. However, MSLR is designed for symmetric problems. This paper presents a preconditioner that incorporates a modified hierarchical low rank approximation of the inverse Schur complement from the MSLR preconditioner into a block preconditioner based on the block LU factorization of \( A \). The resulting method will be called a Generalized Multilevel Schur complement Low-Rank (GMSLR) preconditioner. Two characteristics of GMSLR are worth highlighting. First, **GMSLR is designed to be applicable to a wide range of problems.** The preconditioner is nonsymmetric and it changes at each iteration, since it incorporates inner solves, and flexible GMRES [40] is used as the accelerator. The method also performs well for symmetric matrices. As observed in [7, Section 10.1.2], the loss of symmetry incurred by application of a nonsymmetric preconditioner is not a major concern provided that good approximations to certain blocks of \( A \) are available. The numerical experiments will confirm this observation. Second, a property that is inherited from MSLR is that the **GMSLR preconditioner computes a recursive, multilevel approximation to the inverse of the Schur complement.** GMSLR is a block preconditioner with inner sub-solves required at every outer iteration. These inner solves can themselves be preconditioned in order to reduce computational costs. One of these required inner solves is with the Schur complement, i.e., we must solve \( Sy = g \). For most problems, this inverse Schur complement approximation turns out to be an effective preconditioner for these inner solves. Since an important goal of this paper is to deal with indefinite problems, we explored another improvement targeted specifically at such problems. This improvement consists of a well-established strategy [20, 35, 39, 45, 48] of adding complex shifts to the diagonal prior to performing any of the ILU factorizations required by GMSLR. In the case of GMSLR, this entails modifying the diagonal of coefficient matrix at each level by adding a complex scalar. As is the case for other (standard) preconditioners [39, 48] this strategy also has the effect of improving robustness while decreasing the fill-in required by GMSLR, especially for highly indefinite problems such that those arising from Helmholtz problems.

We note at this point that our focus is on a purely algebraic viewpoint where \( A \) is a general sparse indefinite matrix that does not necessarily originate from the discretization of a Partial Differential Equation. Therefore, we do not consider approaches based on hierarchical matrices.

This paper is organized as follows. In Section 2 we briefly review the HID ordering. Section 3 has a brief overview of block preconditioning that motivates the need for the low rank property of the inverse of the Schur complement. The details of the Schur complement approximation are given in Section 4. In Section 5 we present the preconditioner construction process. A two level analysis of the preconditioned eigenvalues is presented in Section 6. Then, in Section 7, we present some numerical
2. HID ordering. Reordering the original system matrix $A$ is essential for the performance of direct as well as iterative methods [8, 31, 38, 43]. GMSLR uses one such reordering technique known as the Hierarchical Interface Decomposition (HID) [25]. This ordering has also been used in the context of hierarchical linear system solvers [5] but is applicable to a wide class of sparse matrices, not just those that originate from PDEs. An HID ordering can be obtained in a number of ways. A particular method for obtaining such an ordering is the well known nested dissection method [21]. Nested dissection recursively partitions the adjacency graph of $A$ into two disjoint subgraphs and a vertex separator in such a way that the removal of the vertices of the separator from the original graph results in the two disjoint subgraphs. Each level of bisection produces a new separator and new subgraphs. This level information can be represented by an HID tree $T$. The matrix itself is reordered according to level, starting with level 0 and ending with level $L - 1$.

Since we assume that $A$ is large, sparse, and nonsymmetric, then an HID ordering has the multilevel, recursive structure

$$A_l = \begin{pmatrix} B_l & F_l \\ E_l & C_l \end{pmatrix} \text{ and } C_l = A_{l+1} \text{ for } l = 0 : L - 1. \quad (2.1)$$

In this notation, $A_0$ denotes the original matrix $A$ after HID ordering whereas $A_L$ is the submatrix associated with the $L$th level connector. The $B_l$ block itself has a block diagonal structure due to the block independent set ordering [43], making solves with $B_l$ ideally suited for parallel computation. Figure 2.1 shows an example of the HID ordering for a 3D convection-diffusion operator discretized with the standard 7-point finite difference stencil. The left subfigure plots the non-zero pattern of the entire matrix. The right subfigure is a close-up view of the non-zero pattern of the coupling of the interface variables at the root level.

3. Block Preconditioning. Domain decomposition reordering gives rise to linear systems of the form

$$A = \begin{pmatrix} B & F \\ E & C \end{pmatrix}, \quad (3.1)$$

see [2, 11]. Similar block structured matrices also arise from the discretization of systems of partial differential equations. In these coupled systems, the individual blocks usually correspond to differential/integral operators, however in this context they represent different sets of unknowns (interior, interface, coupling) that result from domain decomposition. There is a large body of work on preconditioning these systems mostly from the point of view of saddle point systems, see [6, 7, 28, 36, 37]. For examples of preconditioning other coupled systems of PDEs, see [14, 26, 27].

At the initial (root) level GMSLR uses a block triangular preconditioner of the form

$$P = \begin{pmatrix} \tilde{B}_0 & \tilde{E}_0 \\ 0 & \tilde{S}_0 \end{pmatrix}, \quad (3.2)$$

where $\tilde{B}_0$ is an approximation to the $(1,1)$ block of $A_0$ and $\tilde{S}_0$ is an approximation to the Schur complement $S_0 = C_0 - E_0B^{-1}_0F_0$. 

results from test problems and problems from the SuiteSparse matrix collection [17]. Concluding remarks and some ideas for future work can be found in Section 8.
In the ideal case where $\widetilde{B}_0 = B_0$ and $\widetilde{S}_0 = S_0$, it is well known that the matrix $A_0P_{\text{ideal}}^{-1}$ has a quadratic minimal polynomial, which means that GMRES will converge in two iterations [28, 37]. Therefore the total cost of the procedure based on the ideal form of (3.2) is 2 linear solves with $B_0$ and two linear solves with $S_0$ plus additional sparse matrix-vector products. This is made clear by looking at the factored form of $P_{\text{ideal}}^{-1}$:

$$P_{\text{ideal}}^{-1} = \left( \begin{array}{cc} B_0 & F_0 \\ S_0 & I \end{array} \right)^{-1} = \left( B_0^{-1} I \right) \left( \begin{array}{cc} I & -F_0 \\ I & I \end{array} \right) \left( \begin{array}{cc} I & I \\ S_0^{-1} & -E_0B_0^{-1} \end{array} \right). \quad (3.3)$$

This choice corresponds to using only the upper triangular part of the block $LU$ factorization of $A_0$ as a preconditioner. If both parts of this factorization are used, i.e., if our preconditioner is of the form

$$P^{-1} = \left( \begin{array}{cc} B_0^{-1} I \\ I \end{array} \right) \left( \begin{array}{cc} I & -F_0 \\ I & I \end{array} \right) \left( \begin{array}{cc} I & I \\ S_0^{-1} & -E_0B_0^{-1} \end{array} \right), \quad (3.4)$$

then in the ideal case we have an exact inverse of $A_0$ and a Krylov method will converge in a single iteration at the total cost of two solves with $B_0$ and one solve with $S_0$. Thus, in all, using (3.4) saves one $S_0$ solve over (3.3).

The scenario just described involves ideal preconditioners (3.3) and (3.4) which are however not practical since they involve the exact computation of $S_0^{-1}$. In practice, $B_0$ and $S_0$ are approximated, at the cost of a few extra outer iterations. With these approximations in place it turns out that there is little difference in practice between these two options and, based on our experience, we prefer to use (3.2). This issue will be revisited at the end of Section 7.1.1.

Similar to [34], we solve linear systems with the $B$ blocks by using incomplete LU (ILU) factorizations. Approximations to the Schur complement are typically tailored specifically to the problem being studied (e.g. the pressure convection diffusion [19] and least-squares commutator [18] preconditioners for Navier-Stokes). However, in
our framework, the block form of $A$ is the result of a reordering of the unknowns and so our Schur complement approximation is inherently algebraic and not based on the physics of the problem. We base our Schur complement approximation on ideas from [34, 46].

4. Schur complement approximation. GMSLR is an extension of the MSLR preconditioner of [46] based on approximating the block $LDU$ factorization of (2.1):

\[
A_l = \begin{pmatrix}
I & E_l B_l^{-1} \\
E_l B_l^{-1} & I
\end{pmatrix}
\begin{pmatrix}
B_l \\
S_l
\end{pmatrix}
\begin{pmatrix}
I & B_l^{-1} F_l \\
B_l^{-1} F_l & I
\end{pmatrix}
\]

(4.1)

at every level $l = 0, \ldots, L - 1$. We write the Schur complement as

\[
S_l = (I - E_l B_l^{-1} F_l C_l^{-1}) C_l \equiv (I - G_l) C_l.
\]

(4.2)

Let the complex Schur decomposition of $G_l$ be

\[
G_l = E_l B_l^{-1} F_l C_l^{-1} = W_l R_l W_l^H
\]

(4.3)

where $W_l$ is unitary and $R_l$ is an upper triangular matrix whose diagonal contains the eigenvalues of $G_l$. Substituting (4.3) into (4.2) we get that

\[
\]

(4.4)

Then, the Sherman-Morrison-Woodbury formula yields the inverse of $S_l$:

\[
S_l^{-1} = C_l^{-1} W_l (I - R_l)^{-1} W_l^H = C_l^{-1} [I + W_l ((I - R_l)^{-1} - I)] W_l^H
\]

(4.5)

which reduces to

\[
S_l^{-1} = C_l^{-1} + C_l^{-1} W_l [(I - R_l)^{-1} - I] W_l^H.
\]

(4.6)

Some observations about the matrix $S_l^{-1} - C_l^{-1}$ will be stated in the next section.

In our algorithm, we do not compute the full Schur decomposition of $G_l$, just the $k_l \times k_l$ leading submatrix of $R_l$ and the first $k_l$ Schur vectors. These choices give rise to the following inverse Schur complement approximation.

Definition 4.1. Let $G_l = E_l B_l^{-1} F_l C_l^{-1}$, $l = 0 \ldots L - 1$ and $G_l = W_l R_l W_l^H$ be its Schur decomposition at level $l$. Let $W_l, k_l$ be the matrix of the first $k_l$ Schur vectors of $W_l$. If we define $R_l, k_l$ to be the $k_l \times k_l$ leading principal submatrix of $R_l$, then the approximate $l^{th}$ level inverse Schur complement $\overline{S}_{l,k_l}^{-1}$ is given by

\[
\overline{S}_{l,k_l}^{-1} = C_l^{-1} (I + W_l, k_l H_l, k_l W_l^H).
\]

(4.7)

where

\[
H_l, k_l = [(I - R_l, k_l)^{-1} - I].
\]

(4.8)

The inverse Schur complement approximation in (4.7) will be used at every level $l = 0, \ldots, L - 1$. Due to the potential large size of the $C_l$ blocks, we can only afford to factor $C_{L-1}$ (i.e., at the top level) since it is the smallest of all the $C_l$ blocks. For $l \neq L - 1$ we use a slightly modified version of the recursive scheme of [46] for approximating the action of $C_l^{-1}$ on a vector. The details of this approximation will be shown in Section 5.
4.1. **Low rank property of $S_l^{-1} - C_l^{-1}$**. Consider the inverse Schur complement formula given by (4.6). In this section we claim that for certain problems, the matrix $S_l^{-1} - C_l^{-1}$ is of low rank. If this is the case, then (4.7) will be a good approximation to (4.6). The only assumption we make on the blocks $B_l, C_l$ is that they have LU factorizations, i.e.,

$$B_l = L_l B_l, \quad C_l = L_l C_l.$$  \hspace{1cm} (4.9)

In practice we will use incomplete LU factorizations, so instead

$$B_l \approx L_l B_l, \quad C_l \approx L_l C_l.$$ 

Note that for large, 3D problems, the number of interface points (i.e., the size of the $C_l$ block) can be quite large, making this factorization too costly. This is part of the motivation for the multilevel decomposition.

To see that $S_l^{-1} - C_l^{-1}$ is usually of low rank, again define the matrix $G_l$ by

$$G_l = E_l B_l^{-1} F_l C_l^{-1} = (C_l - S_l) C_l^{-1}.$$  \hspace{1cm} (4.10)

Let $\gamma_i, i = 1, \ldots, s$ be the eigenvalues of $G_l$ (and also $R_l$) and define $X_l = C_l (S_l^{-1} - C_l^{-1})$. By equation (4.6) the eigenvalues $\theta_1, \theta_2, \cdots, \theta_s - 1, \theta_s$ of $X_l$ are given explicitly by

$$\theta_i = \frac{\gamma_i}{1 - \gamma_i}, \quad i = 1, \ldots, s$$  \hspace{1cm} (4.11)

since $(I - G_l)^{-1} - I = G_l (I - G_l)^{-1}$.

As long as the eigenvalues $\gamma_i$ of $G_l$ are not clustered at 1, the eigenvalues $\theta_i$ of $X_l$ will be well separated. This in turn means that $S_l^{-1} - C_l^{-1}$ can be approximated by a low rank matrix. This was studied in detail in [46, Section 2] for the symmetric case, where a theoretical bound for the numerical rank was established.

4.2. **Building the low rank correction.** We use Arnoldi’s method [1] to build the low rank correction matrices in (4.7). This approximation can be efficient if the desired eigenpairs of $G_l$ are on the periphery of the spectrum. However, as we shall see in the numerical results, this is simply not the case for some of the more indefinite problems. A particular remedy is to take more steps of Arnoldi’s method.

Taking $m$ steps of Arnoldi’s method on $G_l$ yields the Krylov factorizations:

$$G_l U_m = U_m H_m + h_{m+1,m} u_{m+1} e_m^T,$$

$$U_m^T G_l U_m = H_m$$

where $U_m$ is an orthonormal matrix and $H_m$ is a Hessenberg matrix whose eigenvalues (also called Ritz values) are good estimates to the extreme eigenvalues of $G_l$. We then take the complex Schur factorization of $H_m$

$$Q^H H_m Q = T.$$  \hspace{1cm} (4.12)

We can reorder the $k_l$ eigenvalues closest to 1 we wish to deflate so that they appear as the first $k_l$ diagonal entries of $T$ [3, 44]. The low rank matrices in (4.7) are approximated by:

$$R_{l,k_l} \approx T_{1:1,k_l}, \quad W_{l,k_l} \approx U_m Q_{1:1,k_l}.$$  \hspace{1cm} (4.13)
5. Preconditioner construction process. In this section we show how the low rank property discussed in the previous section is used to build an efficient preconditioner. The only assumption we make is that each of the $B_l, C_l$ blocks are non-singular. This assumption is typically satisfied unless the original matrix has a block of all zeros (e.g. a saddle point system). At the end of this section we also present an analysis of the computational and memory costs of this preconditioner.

5.1. 3-level scheme. We illustrate the steps taken to solve $Ax = b$ with a 3-level example.

Step 0: Apply a 3-level HID ordering to the original matrix $A$ and right hand side $b$.

Call the resulting reordered matrix and right hand side $A_0, b_0$ respectively.

Step 1: At this level (only) we use the block triangular matrix

\[ U_0^{-1} = \begin{pmatrix} B_0 & F_0 \\ S_0 \end{pmatrix}^{-1} = \begin{pmatrix} B_0^{-1} \\ I \\ -F_0 \\ I \end{pmatrix} \begin{pmatrix} I \\ -F_0 \\ I \end{pmatrix} \begin{pmatrix} I \\ S_0^{-1} \end{pmatrix} \]

as a right preconditioner for $A_0$ i.e., we solve $A_0 U_0^{-1} u = b_0$. Here we approximately factor $B_0$ by ILU and approximate the Schur complement by

\[ S_0^{-1} \approx \tilde{S}_0^{-1} = C_0^{-1} (I + W_0 H_0 W_0^T) \]

where $H_0$ and $W_0$ are taken from (4.8) and (4.13) respectively. To solve with $C_0$, we refer to (2.1) and move from level 0 to level 1.

Step 2: At level 1, we have

\[ C_0^{-1} = A_1^{-1} = \begin{pmatrix} I & -B_1^{-1} F_1 \\ I \end{pmatrix} \begin{pmatrix} B_1^{-1} \\ I \end{pmatrix} \begin{pmatrix} I \\ -E_1 B_1^{-1} I \end{pmatrix} \]

where $S_1^{-1}$ is approximated by $C_1^{-1}$ plus a low rank correction:

\[ S_1^{-1} \approx \tilde{S}_1^{-1} = C_1^{-1} (I + W_1 H_1 W_1^T). \]

Next we move up a level again to define an approximate inverse for $C_1$, referring again to (2.1).

Step 3: At level 2 we have:

\[ C_1^{-1} = A_2^{-1} = \begin{pmatrix} I & -B_2^{-1} F_2 \\ I \end{pmatrix} \begin{pmatrix} B_2^{-1} \\ I \end{pmatrix} \begin{pmatrix} I \\ -E_2 B_2^{-1} I \end{pmatrix} \]

Similarly to Step 2, we now approximate $S_2^{-1}$ by $C_2^{-1}$ plus a low rank correction term, i.e.,

\[ S_2^{-1} \approx \tilde{S}_2^{-1} = C_2^{-1} (I + W_2 H_2 W_2^T). \]

At this level, we decide that $C_2$ is sufficiently small and compute its ILU factorization: $C_2 \approx L_{C_2} U_{C_2}$.

In order to apply the preconditioner $U_0^{-1}$, the actual algorithm starts at level 2 and proceeds up to level 0. For this particular example, that means we start forward-backward solving with the ILU factorization of $C_2$ since $C_2^{-1}$ is needed in order to apply $S_2^{-1}$. Now that the action of $S_2^{-1}$ is available we can then approximate $A_2^{-1}$ and the pattern continues until we hit level 0, i.e.,

\[ L_{C_2} U_{C_2} \rightarrow C_2^{-1} \rightarrow \tilde{S}_2^{-1} \rightarrow A_2^{-1} \rightarrow \tilde{S}_1^{-1} \rightarrow A_1^{-1} \rightarrow \tilde{S}_0^{-1} \rightarrow U_0^{-1}. \]

Once $C_i^{-1}$ (or its action on a vector) is available, the low rank correction matrices $W_l, H_l$ can be computed.
5.2. General Case. When computing the partial Schur decomposition of the matrix $G_l$, we need to be able to compute matrix vector products with the matrix $E_l B_l^{-1} F_l C_l^{-1}$ at each level $l$. We already have the factors of $B_l$, so any matrix-vector product with $B_l^{-1}$ can be computed with one forward and one backward substitution. The same does not hold true for $C_l$, since we only compute its factorization at level $L - 1$. However, we already have an approximate factorization of $A_{l+1}$ and since $C_l^{-1} = A_{l+1}^{-1}$ we can use this approximation to apply $C_l^{-1}$ to a vector. The construction of the preconditioner is summarized in Algorithm 1. The details of the recursively defined product of $C_l^{-1}$ with a vector $b$ are given in Algorithm 2.

Algorithm 1 Generalized Multilevel Schur Low-Rank (Construction phase)

1: procedure GMSLR
2: Apply an $L$-level reordering to $A$ ($A_0$ = reordered matrix).
3: for level $l$ from $L - 1$ to 0 do
4: if $l = L - 1$ then
5: Compute ILU factorization of $C_{L-1}$, $C_{L-1} \approx L_{C_{L-1}} U_{C_{L-1}}$
6: end if
7: Compute ILU factorization of $B_l$, $B_l \approx L_{B_l} U_{B_l}$
8: Perform $k_l$ steps of the Arnoldi process $\triangleright$ Call Algorithm 2 to apply $C_l^{-1}$

$$[V_l, K_l] = \text{Arnoldi}(E_l U_{B_l}^{-1} L_{B_l}^{-1} F_l C_l^{-1}, k_l)$$

9: Compute the complex Schur decomposition $K_l = WTW^T$.
10: Compute $W_{l,k_l} = V_l W$ and set $R_l,k_l = T_{l; k_l; 1:k_l}$.
11: Compute $H_l = (I - R_l,k_l)^{-1} - I = R_l,k_l(0 - R_l,k_l)^{-1}$.
12: end for
13: end procedure

Algorithm 2 Approximation of $y = C_l^{-1}b$ for $l \geq 1$ and $y = U_0^{-1}b$

1: procedure RecursiveSolve($l, b$)
2: if $l = L - 1$ then
3: return $y = U_{C_{L-1}}^{-1} C_{L-1}^{-1} b$
4: else
5: Split $b = (b_1^T, b_2^T)^T$ conformingly with the blocking of $C_l$
6: Compute $z_1 = U_{B_l}^{-1} L_{B_l}^{-1} b_1$
7: Compute $z_2 = b_2 - E_{l+1} z_1$
8: if $1 \leq l < L - 1$ then
9: Compute $w_2 = W_{l+1,k_l+1} H_{l+1} W_{l+1,k_l+1}^T z_2$
10: Compute $y_2 = \text{RecursiveSolve}(l + 1, z_2 + w_2)$
11: Compute $y_1 = z_1 - U_{B_{l+1}}^{-1} L_{B_{l+1}}^{-1} F_{l+1} y_2$
12: else
13: Solve the system $S_0 y_2 = z_2$ with $S_0^{-1}$ as a right preconditioner
14: Compute $y_1 = U_{B_0}^{-1} L_{B_0}^{-1} (b_1 - F_{0} y_2)$
15: end if
16: return $y = (y_1^T, y_2^T)^T$
17: end if
18: end procedure
Similarly to MSLR the HID ordering gives rise to $B_l$ matrices that are block-diagonal in structure, and so all of these blocks can be factored in parallel. Furthermore, the triangular solves associated with $B_l$ can also be done in parallel for each block. In addition, while Algorithm 2 generally provides an accurate approximation to $C_l^{-1}$, we must point out that due to the presence of the inner solve at level $l = 0$ (Line 13 of Algorithm 2), GMSLR is (potentially) more expensive per iteration than MSLR. This expense can be lessened somewhat by the fact that the inner solve can only require 1-2 digits of accuracy without radically affecting the convergence rate of the outer solve.

5.3. Computational and memory complexity of the preconditioner. Let $\text{mem}(\text{ILU}(B_l))$ denote the memory cost associated with the storage of the incomplete factorization of $B_l$. Then the total memory cost $\mu^L_{\text{GMSLR}}$ of the GMSLR preconditioner using $L$ levels is:

$$\mu^L_{\text{GMSLR}} = \left( \sum_{l=0}^{L-1} \left[ \text{mem}(\text{ILU}(B_l)) + \max\{2s_lk_l + k_l^2, 2s_l^2 + s_l^2\} \right] \right) + \text{mem}(\text{ILU}(C_{L-1})), $$

where the second term inside the summation accounts for the memory cost associated with the partial Schur decompositions of order $1 \leq k_l \leq s_l$ at levels $0 \leq l \leq L-1$, and $s_l$ denotes the number of interface variables at level $l$, i.e., the leading dimension of each $C_l$. For simplicity, we treat the upper triangular matrix $H_{l,k_l}$ as a dense matrix.

In the case where the incomplete factorization of matrices $B_l$, $l = 0, \ldots, L-1$, and $C_{L-1}$ are obtained by a thresholded version of ILU, with a maximum number of non-zero entries per row equal to $\tau$, the above memory cost is bounded by

$$\mu^L_{\text{GMSLR}} \leq \left( \sum_{l=0}^{L-1} \left[ 2\tau d_l + \max\{2s_lk_l + k_l^2, 2s_l^2 + s_l^2\} \right] \right) + 2\tau s_{L-1},$$

where $d_l$ denotes the leading dimension of $B_l$.

To obtain an estimate of the computational cost to apply the GMSLR preconditioner at the root level $l \equiv 0$, we need to consider the computational cost associated with all intermediate levels. Let $\text{trisol}(\text{ILU}(B_l))$ and $\text{trisol}(\text{ILU}(C_{L-1}))$ denote the cost of the triangular solves with $B_l$ and $C_{L-1}$ respectively and let $\gamma^{(L-1)}_{\text{GMSLR}}$ denote the cost associated with level $l = L - 1$. At level $l = L - 2$, the cost to apply the GMSLR preconditioner is equal to the sum of the cost to apply the preconditioner at level $l + 1 = L - 1$ and the cost $2\text{trisol}(\text{ILU}(B_{L-2})) + O(s_{L-2}k_{L-2})$, where we assumed that $k_{L-2} \ll s_{L-2}$. Continuing in the same spirit, we finally get that the cost to apply the GMSLR preconditioner at level $l$, $\gamma^{(l)}_{\text{GMSLR}}$, is equal to

$$\gamma^{(l)}_{\text{GMSLR}} = \gamma^{(l+1)}_{\text{GMSLR}} + 2\text{trisol}(\text{ILU}(B_l)) + O(s_l k_l), \ l = 0, \ldots, L-2,$$

where

$$\gamma^{(L-1)}_{\text{GMSLR}} = \text{trisol}(\text{ILU}(C_{L-1})) + 2\text{trisol}(\text{ILU}(B_{L-1})) + O(s_{L-1}k_{L-1}).$$

6. Eigenvalue Analysis. This section studies the spectra of linear systems preconditioned by GMSLR. We only consider a 2 level decomposition since the recursive nature of both algorithms makes the analysis difficult. In what follows, let $\bar{B}_0$ denote
an approximation to $B_0$ and $\tilde{S}_0$ the GMSLR approximation to the Schur complement
$S_0 = C_0 - E_0 B_0^{-1} F_0$ respectively. GMSLR starts with a $2 \times 2$ block partition of the
original matrix $A$, i.e.,

$$A_0 = \begin{pmatrix} B_0 & F_0 \\ E_0 & C_0 \end{pmatrix}$$

(6.1)

where $B_0$ is $n_B \times n_B$ and $C_0$ is $s \times s$.

As was already seen, the GMSLR preconditioner is based on the block-LU factorization of (6.1), so at level 0 we have

$$A_0 = \begin{pmatrix} B_0 & F_0 \\ E_0 & C_0 \end{pmatrix} = \begin{pmatrix} I & 0 \\ E_0 B_0^{-1} & I \end{pmatrix} \begin{pmatrix} B_0 & F_0 \\ 0 & S_0 \end{pmatrix} = L_0 U_0,$$

and the preconditioner $\tilde{U}_0^{-1}$ is

$$\tilde{U}_0^{-1} = \begin{pmatrix} B_0^{-1} & 0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & -F_0 \\ 0 & I \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \tilde{S}_0^{-1} \end{pmatrix}.$$  

A simple calculation shows that

$$A_0 \tilde{U}_0^{-1} = \begin{pmatrix} B_0 \tilde{B}_0^{-1} \\ E_0 \tilde{B}_0^{-1} \end{pmatrix} \begin{pmatrix} I - B_0 \tilde{B}_0^{-1} & F_0 \tilde{S}_0^{-1} \\ 0 & S_0 \tilde{S}_0^{-1} \end{pmatrix}.$$  

(6.2)

If we assume that $\tilde{B}_0 = B_0$, then (6.2) simplifies to

$$A_0 \tilde{U}_0^{-1} = \begin{pmatrix} I \\ E_0 B_0^{-1} \end{pmatrix} \begin{pmatrix} 0 & \tilde{S}_0^{-1} \end{pmatrix},$$

(6.3)

which has eigenvalues $\lambda(A_0 \tilde{U}_0^{-1}) = \{1, \lambda(S_0 \tilde{S}_0^{-1})\}$.

Convergence will be rapid if the eigenvalues of $S_0 \tilde{S}_0^{-1}$ are also close to 1. To illustrate the influence the rank has on convergence, we show in Figure 6.1 the spectra of $S_0 \tilde{S}_0^{-1}$ for a small test problem. Here $A$ is the discretized shifted Laplacian operator $-\Delta u - cu = f$ with $c = 0.0$ (left figure) and $c = 0.5$ (right figure) and homogeneous

Fig. 6.1: The spectrum of $S_0 \tilde{S}_0^{-1}$ for different values of $k_0 \equiv k_1$ ($L = 2$). Left: $c = 0.0$. Right: $c = 0.5$. 

\[\text{Re} \lambda(S_0 \tilde{S}_0^{-1})\]
Dirichlet boundary conditions. For reference, when $c = 0.5$, this $8000 \times 8000$ matrix has 35 negative eigenvalues. This is a matrix selected for illustrative purposes, so we use two levels with equal ranks $k_0, k_1$ and compute the exact $LU$ factorization of $B_0$. As the ranks $k_0, k_1$ increase, the real part of the eigenvalues of $S_0\tilde{S}_0^{-1}$ clusters more tightly around 1.

7. Numerical experiments. All experiments were run on a single node of the 

Mesabi Linux cluster at the Minnesota Supercomputing Institute. This node has a memory of 64 GBs and consists of two sockets each having a twelve core 2.5 GHz Intel Haswell processor. The GMSLR preconditioner was written in C++ and compiled by Intel’s C++ compiler using $-O3$ optimization. Simple thread-level parallelism was achieved with OpenMP with a maximum of 24 threads. The $B_l$ blocks are factored by the ILUT routine from ITSOL. The Intel Math Kernel Library (MKL) was used for the BLAS and LAPACK routines. We use flexible GMRES [40] with a fixed restart size of 40 as the outer solver, denoted by GMRES(40). The inner solve in step 14 of Algorithm 2 is also done with FGMRES. Unless otherwise noted, we follow the methodology of [32, 41, 46] where the right hand side vector $b$ is given by $Ae = b$ where $e$ is the vector of all ones.

The HID ordering was obtained by the function PartGraphRecursive from the METIS [30] package. The diagonal blocks of each $B_l, C_l$, $l = 0, \ldots, L - 1$, were reordered using the approximate minimum degree (AMD) ordering [15, 16] in order to reduce the fill-in generated by their ILU factorizations. In our experiments the reported preconditioner construction time comes from the factorization of the $B_l$ blocks and the computation of the low rank correction matrices. The reordering time is regarded as preprocessing and is therefore not reported. Similarly, the iteration time is the combined time spent on the inner and outer solves.

The parameters we are most interested in varying are: the number $L$ of levels in the HID and the maximum rank used in the low rank correction, i.e., the number of steps of Arnoldi’s method (see Section 4.2), denoted by $rk$. In particular we set $k_l$ in (4.13) to be equal to $rk$ for any $l = 0, \ldots, L - 1$, and thus all Arnoldi vectors are included in the low rank correction terms.

We use the following notation in the results that follow:

- $\text{fill} = \frac{nnz(prec)}{nnz(A)}$, where $nnz$ denotes the number of non-zero entries of the input matrix;
- $\text{p-t}$: wall clock time to build the preconditioner (in seconds);
- $\text{its}$: number of outer iterations of preconditioned GMRES(40) required for $\|r_k\|_2 < 10^{-6}$. We use “F” to indicate that GMRES(40) did not converge after 500 iterations;
- $\text{i-t}$: wall clock time for the iteration phase of the solver.
- $\text{rk}$: max rank used in building the low rank corrections.

The value of $nnz(prec)$ is the sum of the non-zero entries associated with the incomplete factorizations (ILU) and Low-Rank Correction (LRC) terms. These quantities are computed as $\sum_{l=0}^{L-1} [(nnz(U_{B_l}) + nnz(L_{B_l})] + (nnz(U_{C_{L-1}}) + nnz(L_{C_{L-1}}))$ for ILU, and $\sum_{l=0}^{L-1} [2d_l rk + rk^2]$ for LRC, respectively.

7.1. Problem 1. We begin our tests with the symmetric indefinite problem:

$$-\Delta u - cu = f \quad \text{in } \Omega,$$
$$u = 0 \quad \text{on } \partial \Omega,$$

(7.1)
where $\Omega = (0,1)^3$. The discretization is via finite differences with the standard 7-point stencil in 3D. This test problem is useful for testing robustness with respect to definiteness. For reference, GMRES preconditioned by standard AMG fails to converge when applied to (7.1) with even a small positive shift on a $32 \times 32$ regular mesh.

7.1.1. Varying the number of levels. First, we study the effect of adding more levels to the preconditioner. We solve (7.1) with $c > 0$ in order to make the problem indefinite. In the cases where $c > 0$, we shift the discretized Laplacian operator by $sI$, where $s = h^2c$ for mesh size $h$. For this first example, we set $s = 0.5$. The associated coefficient matrix has 163 negative eigenvalues. The maximum rank was fixed at 50. By Table 7.1 we can see that, as $L$ grows larger, the ILU fill-factor decreases monotonically while the low rank correction fill-factor increases monotonically. The optimal number of levels occurs when these two quantities are roughly equal. For this particular example, we pick $L_{opt} = 6$ as it strikes the right balance of fill, iteration count, and total computational time. Note that as $L$ increases, so does the number of interface variables at the root level, $s_0$. This can be verified immediately by looking at Table 7.2 where we list $s_0$ for all values of $L$ from $L = 2$ to $L = 6$. Figure 7.1 plots the number of inner iterations performed by GMSLR as a function of the rank $rk$ for different values of the drop tolerance (denoted by tol) in the ILU factorizations.

Finally, recall that we could have used the inexact version of (3.4) instead of (3.2). For SPD problems there is not a significant difference in the results obtained by either preconditioner. However, as shown in Table 7.2, for an indefinite problem such as (7.1) with $s = 0.5$, (3.2) performs better. The likely explanation for this behavior is that (3.2) involves fewer solves with the $B_l$ matrices which are highly indefinite and therefore admit poor ILU factorizations.

7.1.2. Varying the maximum rank in the low rank corrections. Next, we keep the number of levels fixed, but increase the maximum rank. We again solve (7.1)
Table 7.1: The fill factor and iteration counts for solving (7.1) with $s = 0.5$ on a $32^3$ grid with the FGMRES-GMSLR method. Here, the maximum rank for the LRC matrices was fixed at 50.

<table>
<thead>
<tr>
<th>$L$</th>
<th>ILU fill</th>
<th>LRC fill</th>
<th>fill</th>
<th>p-t</th>
<th>i-t</th>
<th>its</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>34.61</td>
<td>.23</td>
<td>34.84</td>
<td>5.16</td>
<td>1.25</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>21.03</td>
<td>.68</td>
<td>21.71</td>
<td>.986</td>
<td>2.69</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>15.64</td>
<td>1.35</td>
<td>16.99</td>
<td>.382</td>
<td>1.03</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>8.69</td>
<td>2.46</td>
<td>11.15</td>
<td>.169</td>
<td>.97</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>5.56</td>
<td>3.96</td>
<td>9.52</td>
<td>.172</td>
<td>.95</td>
<td>17</td>
</tr>
</tbody>
</table>

Table 7.2: Comparison between GMSLR with only using $U^{-1}_0$ and GMSLR with $L^{-1}_0$ and $U^{-1}_0$ on (7.1) with $s = 0.5$ on a $32^3$ grid. The maximum rank was fixed at 50.

<table>
<thead>
<tr>
<th>$L$</th>
<th>$s_0$</th>
<th>GMSLR - $U^{-1}_0$ only</th>
<th>GMSLR - $U^{-1}_0 L^{-1}_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p-t</td>
<td>i-t</td>
<td>its</td>
</tr>
<tr>
<td>2</td>
<td>1.024</td>
<td>5.16</td>
<td>1.25</td>
</tr>
<tr>
<td>3</td>
<td>2.016</td>
<td>.986</td>
<td>2.69</td>
</tr>
<tr>
<td>4</td>
<td>2.977</td>
<td>.382</td>
<td>1.03</td>
</tr>
<tr>
<td>5</td>
<td>4.955</td>
<td>.169</td>
<td>.97</td>
</tr>
<tr>
<td>6</td>
<td>6.699</td>
<td>.172</td>
<td>.95</td>
</tr>
</tbody>
</table>

with $s = 0.5$ discretized on a $32^3$ regular grid. The ILU fill factor is constant because we are keeping the number of levels fixed at 6. The fill factor from the low rank corrections increases at an almost constant rate. Increasing the maximum rank has the unfortunate effect of increasing the fill-factor and the preconditioner construction time. As we see in Table 7.3, the effect of increasing the rank (at least for this model problem) is difficult to predict. As a general rule, it seems as though a large maximum rank is unavoidable for highly indefinite problems.

Table 7.3: Iteration counts for solving (7.1) with $s = 0.5$ on a $32^3$ grid with the FGMRES-GMSLR method. The number of levels was fixed at 6.

<table>
<thead>
<tr>
<th>rk</th>
<th>ILU fill</th>
<th>LRC fill</th>
<th>fill</th>
<th>p-t</th>
<th>i-t</th>
<th>its</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>5.56</td>
<td>1.58</td>
<td>7.14</td>
<td>.091</td>
<td>1.34</td>
<td>24</td>
</tr>
<tr>
<td>30</td>
<td>5.56</td>
<td>2.37</td>
<td>7.93</td>
<td>.118</td>
<td>1.14</td>
<td>19</td>
</tr>
<tr>
<td>40</td>
<td>5.56</td>
<td>3.17</td>
<td>8.73</td>
<td>.139</td>
<td>1.04</td>
<td>18</td>
</tr>
<tr>
<td>50</td>
<td>5.56</td>
<td>3.96</td>
<td>9.52</td>
<td>.174</td>
<td>.972</td>
<td>17</td>
</tr>
<tr>
<td>60</td>
<td>5.56</td>
<td>4.75</td>
<td>10.31</td>
<td>.208</td>
<td>1.29</td>
<td>22</td>
</tr>
<tr>
<td>70</td>
<td>5.56</td>
<td>5.24</td>
<td>10.8</td>
<td>.221</td>
<td>1.35</td>
<td>24</td>
</tr>
<tr>
<td>80</td>
<td>5.56</td>
<td>5.99</td>
<td>11.55</td>
<td>.291</td>
<td>.968</td>
<td>15</td>
</tr>
</tbody>
</table>

7.1.3. Increasingly indefinite problems. The model problem (7.1) becomes significantly more difficult to solve as $s$ increases. Here, we increase $s$ from 0 to 1 while tuning the maximum rank and number of levels to compensate for solving this increasingly difficult problem. We report the results that give the best balance
between iteration count and fill in Table 7.4. The fill factor increases dramatically for
two reasons: first, we must increase the rank of the low rank correction and second,
we must keep the number of levels low, which, as was observed in Section 7.1.1, leads
to increased fill-in for the same drop tolerance. If the rank is too low or the number
of levels is too high, GMRES(40) simply will not converge. Recall that the construction
of the low rank correction is based on finding approximate eigenvalues of the matrix
\( E_lU_l^{-1}L_l^{-1}F_lC_l^{-1} \) using Arnoldi’s method. When \( B_0 \) is indefinite, as is the case here,
the eigenvalues we seek get pushed deeper inside the spectrum, i.e. they become
interior eigenvalues. Since the Arnoldi process does a poorer job for these interior
eigenvalues than it does for extreme ones, we are forced to take more Arnoldi steps
in order to approximate them.

Table 7.4: Results of solving symmetric linear systems with increasing shift values \( s \)
on a \( 32^3 \) regular mesh with GMSLR.

<table>
<thead>
<tr>
<th>( s )</th>
<th>( L )</th>
<th>max rank</th>
<th>fill</th>
<th>p-t</th>
<th>i-t</th>
<th>its</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8</td>
<td>20</td>
<td>5.89</td>
<td>.109</td>
<td>.068</td>
<td>3</td>
</tr>
<tr>
<td>.25</td>
<td>6</td>
<td>30</td>
<td>7.59</td>
<td>.117</td>
<td>.449</td>
<td>8</td>
</tr>
<tr>
<td>.5</td>
<td>6</td>
<td>50</td>
<td>9.52</td>
<td>.174</td>
<td>.973</td>
<td>17</td>
</tr>
<tr>
<td>.75</td>
<td>5</td>
<td>80</td>
<td>12.77</td>
<td>.291</td>
<td>.826</td>
<td>13</td>
</tr>
<tr>
<td>1.0</td>
<td>5</td>
<td>120</td>
<td>13.73</td>
<td>.406</td>
<td>1.87</td>
<td>29</td>
</tr>
</tbody>
</table>

7.2. Problem 2. The second problem of interest is nonsymmetric:

\[-\Delta u - \alpha \cdot \nabla u - cu = f \quad \text{in } \Omega,\]
\[u = 0 \quad \text{on } \partial \Omega,\]

(7.2)

where \( \Omega = (0,1)^3, \alpha \in \mathbb{R}^3 \). This problem is simply a shifted convection-diffusion
equation, again discretized by the 7-point finite difference stencil. As before we shift
the discretized convection-diffusion operator by \( sI \) where \( s = h^2c \).

7.2.1. Varying the number of levels. In this next set of experiments we fix
\( \alpha = [1,1,1] \) and solve (7.2 ) in 3D with no shift and then with a shift of \( s = .25 \). As before, we start by increasing the number of levels. The results of the first
problem with a maximum rank of 20 are in Table 7.5. These results are comparable
to those obtained from the SPD problem (7.1) with \( s = 0 \), i.e., for this problem, the
convergence rate is not adversely affected by the loss of symmetry.

Next, we solve (7.2) with \( s = .25 \). The shift significantly increases the number
of eigenvalues with negative real parts, so we increase the maximum rank to 50. The
results can be found in Table 7.6. It is interesting to note that the fill from the low
rank correction is almost exactly the same as in Table 7.1. This is due to the fact
that both problems used a maximum rank of 50 to build the low rank corrections.

7.3. Problem 3. The third model problem is a Helmholtz equation of the form

\[( -\Delta - \frac{\omega^2}{v(x)^2} ) u(x, \omega) = s(x, \omega). \]

(7.3)

In this formulation, \( \Delta \) is the Laplacian operator, \( \omega \) the angular frequency, \( v(x) \) the
velocity field, and \( s(x, \omega) \) is the external forcing function with corresponding
time-harmonic wave field solution \( u(x, \omega) \). The computational domain is the unit cube
Table 7.5: The fill factor and iteration counts for solving (7.2) with no shift and \( \alpha = [1, 1, 1] \) on a \( 32^3 \) grid with the FGMRES-GMSLR method. Here, the maximum rank for the LRC matrices was fixed at 20.

<table>
<thead>
<tr>
<th>L</th>
<th>ILU fill</th>
<th>LRC fill</th>
<th>fill</th>
<th>p-t</th>
<th>i-t</th>
<th>its</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>11.69</td>
<td>.092</td>
<td>11.78</td>
<td>.505</td>
<td>.159</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>10.13</td>
<td>.272</td>
<td>10.4</td>
<td>.234</td>
<td>.079</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>8.8</td>
<td>.539</td>
<td>9.34</td>
<td>.126</td>
<td>.044</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>6.47</td>
<td>.983</td>
<td>7.46</td>
<td>.09</td>
<td>.041</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4.89</td>
<td>1.58</td>
<td>6.47</td>
<td>.086</td>
<td>.074</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3.8</td>
<td>2.34</td>
<td>6.14</td>
<td>.092</td>
<td>.066</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>2.53</td>
<td>3.35</td>
<td>5.88</td>
<td>.116</td>
<td>.066</td>
<td>3</td>
</tr>
</tbody>
</table>

Table 7.6: The fill factor and iteration counts for solving (7.2) with \( s = .25 \) and \( \alpha = [1, 1, 1] \) on a \( 32^3 \) grid with the FGMRES-GMSLR method. Here, the maximum rank for the LRC matrices was fixed at 50.

<table>
<thead>
<tr>
<th>L</th>
<th>ILU fill</th>
<th>LRC fill</th>
<th>fill</th>
<th>p-t</th>
<th>i-t</th>
<th>its</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>24.11</td>
<td>.23</td>
<td>24.34</td>
<td>2.03</td>
<td>.88</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>15.44</td>
<td>.681</td>
<td>16.12</td>
<td>.58</td>
<td>.61</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>11.64</td>
<td>1.35</td>
<td>12.99</td>
<td>.237</td>
<td>.381</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>7.25</td>
<td>2.46</td>
<td>9.71</td>
<td>.149</td>
<td>.91</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>5.16</td>
<td>3.96</td>
<td>9.12</td>
<td>.167</td>
<td>.741</td>
<td>13</td>
</tr>
<tr>
<td>7</td>
<td>3.91</td>
<td>5.86</td>
<td>9.77</td>
<td>.214</td>
<td>1.00</td>
<td>14</td>
</tr>
<tr>
<td>8</td>
<td>2.56</td>
<td>8.39</td>
<td>10.95</td>
<td>.288</td>
<td>4.54</td>
<td>53</td>
</tr>
</tbody>
</table>

\( \Omega = (0, 1)^3 \) where we again use the seven-point finite difference discretization on a regular mesh. The Perfectly Matched Layer (PML) boundary condition is used on all faces of \( \Omega \). The resulting linear systems are complex non-Hermitian. If we assume that the mean of \( v(x) \) is 1 in (7.3), then the wave number is \( \omega/(2\pi) \) and \( \lambda = 2\pi/\omega \) is the wavelength. The number of grid points in each dimension is \( N = q\omega/(2\pi) \) where \( q \) is the number of points per wavelength. As a result, the discretized system is of size \( n = N^3 \times N^3 \).

Table 7.7: Results from solving (7.3) on a sequence of 3D meshes with GMSLR. All problems have \( q = 8 \) points per wavelength. The second set of results is with a small complex shift added to the \( B_l \) matrices.

<table>
<thead>
<tr>
<th>( \omega/(2\pi) )</th>
<th>( n = N^3 )</th>
<th>L</th>
<th>rk</th>
<th>GMSLR - no shift</th>
<th>GMSLR w/ complex shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>fill</td>
<td>p-t</td>
</tr>
<tr>
<td>2.5</td>
<td>20^4</td>
<td>5</td>
<td>16</td>
<td>3.79</td>
<td>.063</td>
</tr>
<tr>
<td>3</td>
<td>30^4</td>
<td>6</td>
<td>16</td>
<td>5.18</td>
<td>.156</td>
</tr>
<tr>
<td>5</td>
<td>40^4</td>
<td>7</td>
<td>16</td>
<td>6.19</td>
<td>.282</td>
</tr>
<tr>
<td>6</td>
<td>50^4</td>
<td>7</td>
<td>16</td>
<td>8.16</td>
<td>.768</td>
</tr>
<tr>
<td>8</td>
<td>60^4</td>
<td>8</td>
<td>16</td>
<td>7.73</td>
<td>.867</td>
</tr>
<tr>
<td>10</td>
<td>80^4</td>
<td>9</td>
<td>16</td>
<td>7.85</td>
<td>1.57</td>
</tr>
</tbody>
</table>

15
We test the performance of the GMSLR preconditioner on 6 cubes, setting $q = 8$, and report the results in Table 7.7. Since $q$ is fixed, an increase in the wave number means an increase in $N$, so the higher frequency problems lead to much larger linear systems. In these experiments, we set the inner solve tolerance to $10^{-2}$ or a maximum of 10 iterations. Results reported under the legend “GMSLR - no shift” stand for the regular GMSLR preconditioner. Results reported under the legend “GMSLR w/ complex shift” stand for runs where the GMSLR preconditioner was built by first shifting $B_l$ by $\sigma = \left(\sum_{j=1}^{d_l} (B_l)_{jj} / d_l \right) * 0.05 * i$. Without a complex shift, these problems can be much more difficult, especially as the matrix size grows. Indeed, for the last two examples no convergence was achieved after 300 outer iterations. On the other hand, the shift benefits all test problems as it allows for an increased number of levels (and thus less fill-in introduced by ILU) while also keeping the number of outer iterations relatively small (the number of outer iterations only increased from 17 to 28 as the matrix size grew from $20^3$ to $80^3$).

### 7.4. General sparse matrices

To further illustrate the robustness of the GMSLR preconditioner, we tested it on several large, nonsymmetric matrices from the SuiteSparse Matrix Collection [17]. These matrices come from a wide range of application areas, not just PDEs. As a benchmark, we also tested ILUT for these nonsymmetric matrices. Information about the matrices is shown in Table 7.8. Table 7.9 shows the results of these experiments. The ILUT parameters were chosen such that the fill of both methods was comparable.

#### Table 7.8: Set of nonsymmetric test matrices from the SuiteSparse Matrix Collection where nnz is the number of nonzero entries in the matrix.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>Order</th>
<th>nnz</th>
<th>SPD</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>cbuckle</td>
<td>13,681</td>
<td>676,515</td>
<td>yes</td>
<td>structural problem</td>
</tr>
<tr>
<td>epb2</td>
<td>25,228</td>
<td>175,027</td>
<td>no</td>
<td>thermal problem</td>
</tr>
<tr>
<td>wang4</td>
<td>26,068</td>
<td>177,196</td>
<td>no</td>
<td>semiconductor device problem</td>
</tr>
<tr>
<td>barrier2−1</td>
<td>113,076</td>
<td>3,805,068</td>
<td>no</td>
<td>semiconductor device problem</td>
</tr>
<tr>
<td>Cage12</td>
<td>130,228</td>
<td>2,032,536</td>
<td>no</td>
<td>directed weighted graph</td>
</tr>
<tr>
<td>offshore</td>
<td>259,789</td>
<td>4,242,673</td>
<td>yes</td>
<td>electromagnetics problem</td>
</tr>
<tr>
<td>CoupCons</td>
<td>416,800</td>
<td>22,322,336</td>
<td>no</td>
<td>structural problem</td>
</tr>
<tr>
<td>AtmosModd</td>
<td>1,270,432</td>
<td>8,814,880</td>
<td>no</td>
<td>atmospheric model</td>
</tr>
<tr>
<td>AtmosModL</td>
<td>1,489,752</td>
<td>10,319,760</td>
<td>no</td>
<td>atmospheric model</td>
</tr>
<tr>
<td>Cage14</td>
<td>1,505,785</td>
<td>27,130,349</td>
<td>no</td>
<td>directed weighted graph</td>
</tr>
<tr>
<td>Transport</td>
<td>1,602,111</td>
<td>23,500,731</td>
<td>no</td>
<td>CFD problem</td>
</tr>
</tbody>
</table>

Results are shown in Table 7.9, where F indicates a failure to converge in 500 iterations. As can be seen, for these problems, GMSLR is superior to ILUT. It is worth adding that ILUT is a highly sequential preconditioner both in its construction and its application. In contrast, GMSLR is by design a domain decomposition-type preconditioner that offers potential for excellent parallelism.

Figure 7.2 plots the value of i-t and p-t as both $L$ and the drop tolerance “tol” of the incomplete factorizations are varied for matrices “barrier2-1” and “offshore”. In agreement with the results reported so far, an increase in the value of $L$ reduces the time to construct and apply the preconditioner. On the other hand, an increase in the value of $L$ might also lead to a larger number of inner iterations necessary to
Table 7.9: Comparison between GMSLR and ILUT preconditioners for solving the above problems. ILUT parameters were chosen so that the fill factor was close to that of GMSLR. Both sets of tests use the same reordered matrix.

<table>
<thead>
<tr>
<th>Matrix</th>
<th>GMSLR</th>
<th>ILUT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>fill  L  rk  p-t  i-t  its</td>
<td>fill  p-t  i-t  its</td>
</tr>
<tr>
<td>cbuckle</td>
<td>2.13  5  5  .22  .10  9</td>
<td>2.27  .38  .32  22</td>
</tr>
<tr>
<td>epb2</td>
<td>3.63  5  15  .23  .13  4</td>
<td>3.43  .98  .76  19</td>
</tr>
<tr>
<td>wang4</td>
<td>4.83  2  35  .14  .08  13</td>
<td>4.92  .45  .41  18</td>
</tr>
<tr>
<td>barrier2−1</td>
<td>3.72  5  10  .60  1.91  6</td>
<td>3.69  44.19  14.32  F</td>
</tr>
<tr>
<td>Cage12</td>
<td>0.95  5  25  .23  .28  4</td>
<td>1.00  .24  .31  5</td>
</tr>
<tr>
<td>offshore</td>
<td>0.99  12  35  1.27  2.35  5</td>
<td>1.09  1.02  1.60  10</td>
</tr>
<tr>
<td>CoupCons</td>
<td>1.82  10  16  1.68  .64  5</td>
<td>1.64  17.49  2.03  23</td>
</tr>
<tr>
<td>AtmosModd</td>
<td>5.86  10  16  1.23  3.05  11</td>
<td>5.68  8.10  8.60  47</td>
</tr>
<tr>
<td>AtmosModL</td>
<td>5.81  11  4  1.67  2.12  7</td>
<td>6.03  11.35  6.37  30</td>
</tr>
<tr>
<td>Cage14</td>
<td>1.54  6  4  3.10  .89  4</td>
<td>1.57  5.09  .70  4</td>
</tr>
<tr>
<td>Transport</td>
<td>2.52  11  4  1.85  7.45  23</td>
<td>2.59  27.91  59.7  116</td>
</tr>
</tbody>
</table>

Fig. 7.2: Amount of time spent on building the GMSLR preconditioner and solving the linear system as the number of levels $L$ and the drop tolerance $tol$ in the incomplete factorizations vary. The rank $rk$ was fixed to 10.

8. Conclusion. The GMSLR preconditioner combines several ideas. First is the HID ordering method, which has a recursive multilevel structure. The $(1, 1)$ block of each level of this structure is block diagonal, which means that solves with this block are easily parallelizable. Motivated by the block $LU$ factorization of the reordered matrix, we use a block triangular preconditioner at the bottom level of the HID tree. For the other levels, we use approximate inverse factorizations exploiting a recursive relationship between the different levels. Finally, we approximate the inverse Schur complement at each level of the HID tree via a low rank correction technique.
Because it is essentially an approximate inverse preconditioner, GMSLR is capable of solving a wide range of highly indefinite problems that would be difficult to solve by standard methods such as ILU. The numerical experiments we showed confirm this. Additional benefits of GMSLR include its inherent parallelism and its fast construction.

GMSLR is also promising for use in eigenvalue computations, especially in the context of rational filtering eigenvalue solvers where complex, indefinite linear systems need be solved [29, 47]. The factorization of these systems can be slow and costly for large 3D problems. We plan on investigating the use of Krylov subspace methods preconditioned by GMSLR to solve such systems. Among other objectives, we also plan to implement and publicly release a fully parallel, domain-decomposition based version of GMSLR.

Acknowledgements. The authors would like to thank the Minnesota Supercomputing Institute for the use of their extensive computing resources and the anonymous referees for their careful reading of this paper and helpful suggestions.

REFERENCES


