EFFECTIVE AND ROBUST PRECONDITIONING OF GENERAL SPD MATRICES VIA STRUCTURED INCOMPLETE FACTORIZATION

JIANLIN XIA AND ZIXING XIN*

Abstract. For general symmetric positive definite (SPD) matrices, we present a framework for designing effective and robust black-box preconditioners via structured incomplete factorization. In a scaling-and-compression strategy, off-diagonal blocks are first scaled on both sides (by the inverses of the factors of the corresponding diagonal blocks) and then compressed into low-rank approximations. ULV-type factorizations are then computed. A resulting prototype preconditioner is always positive definite. Generalizations to practical hierarchical multilevel preconditioners are given. Systematic analysis of the approximation error, robustness, and effectiveness is shown for both the prototype preconditioner and the multilevel generalization. In particular, we show how local scaling and compression control the approximation accuracy and robustness, and how aggressive compression leads to efficient preconditioners that can significantly reduce the condition number and improve the eigenvalue clustering. A result is also given to show when the multilevel preconditioner preserves positive definiteness. The costs to apply the multilevel preconditioners are about $O(N)$, where N is the matrix size. Numerical tests on several ill-conditioned problems show the effectiveness and robustness even if the compression uses very small numerical ranks. In addition, significant robustness and effectiveness benefits can be observed as compared with a standard rank structured preconditioner based on direct off-diagonal compression.

Key words. effective preconditioning, approximation error, preservation of positive definiteness, structured incomplete factorization, scaling-and-compression strategy, multilevel scheme

AMS subject classifications. 15A23, 65F10, 65F30

1. Introduction.

Preconditioners play a key role in iterative solutions of linear systems. For symmetric positive definite (SPD) matrices, incomplete factorizations have often been used to construct robust preconditioners. See [1, 2, 3, 13, 19, 22] for some examples. In recent years, a strategy based on low-rank approximations has been successfully used to design preconditioners, where certain off-diagonal blocks are approximated by low-rank forms. Such rank structured preconditioners often target at some specific problems, usually PDEs and integral equations (see,. e.g., [6, 10, 11, 14]). For more general sparse matrices, some algebraic preconditioners that incorporate low-rank approximations appear in [15, 16, 17, 25].

For general (dense) SPD matrices A, limited work is available about the feasibility and effectiveness of preconditioning techniques based on low-rank off-diagonal approximations. In general, when the off-diagonal blocks of A are directly approximated by low-rank forms with very low accuracy, it is not clear how effective the resulting preconditioner is. The preconditioner may also likely fail to preserve the positive definiteness that is crucial in many applications. Preliminary studies are conducted in [12, 29, 23], where sequential Schur complement computations are used to ensure positive definiteness in an idea of Schur monotonicity or Schur compensation. In [29], the effectiveness for preconditioning is analyzed in terms of a special case.

In this paper, we consider the preconditioning of a general SPD matrix A based on incomplete factorizations that involve low-rank off-diagonal approximations. (For convenience, all our discussions are in terms of real dense SPD matrices.) We present a

*Department of Mathematics, Purdue University, West Lafayette, IN 47907 (xiaj@math.purdue.edu, zxin@purdue.edu). The research of Jianlin Xia was supported in part by NSF CAREER Award DMS-1255416.

framework to construct effective and robust *black-box* structured preconditioners that are convenient to analyze. In the framework, we compute a *structured incomplete factorization* (SIF) $A \approx \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$, where appropriate off-diagonal blocks are first *scaled on both sides* and then *compressed* (into low-rank approximations). The resulting preconditioners (called SIF preconditioners) can effectively reduce the condition number of A and bring the eigenvalues to cluster around 1. They are also robust in the sense that they can preserve the positive definiteness (either unconditionally in a prototype case or under a condition in a general case). More specifically, our main contributions include: (1) *a fundamental SIF preconditioning framework*, (2) the *multilevel generalization*, and (3) *systematic analysis*. The details are as follows.

We give a fundamental SIF framework for designing effective and robust structured preconditioners for A. The framework includes a *scaling-and-compression* strategy for matrix approximation and a ULV-type factorization. That is, instead of directly compressing an off-diagonal block, we scale the off-diagonal block on both sides with the inverses of the factors of the corresponding diagonal blocks and then perform the compression. A sequence of orthogonal and triangular factors (called ULV factors) is then computed. $\tilde{\mathbf{L}}$ is defined by these ULV factors and $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$ serves as the preconditioner. We show that this preconditioner has superior effectiveness and robustness.

The basic idea is illustrated in terms of a prototype preconditioner, which is shown to be *always positive definite* regardless of the accuracy used in the compression. We can also show that the local compression tolerance τ precisely controls the overall (relative) approximation accuracy of the matrix and the factor (see Theorem 2.4). Furthermore, we can justify the *effectiveness* of the preconditioner based on how τ impacts the eigenvalue distribution and the condition number of the preconditioned matrix $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$. In fact, the eigenvalues of $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ cluster inside $[1-\tau, 1+\tau]$. Moreover, as long as the singular values of the scaled off-diagonal block slightly decay, we can aggressively truncate them so as to get a compact preconditioner $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$ that significantly reduces the condition number of $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$. This is rigorously characterized in terms of a representation $\frac{1+\tau}{1-\tau}$ in Theorem 2.5.

We further generalize the prototype preconditioner to *multiple levels*, where multilevel scaling-and-compression operations are combined with a hierarchical ULV-type factorization. The robustness, accuracy, and effectiveness of the multilevel preconditioner $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$ are also shown. A condition for τ is given to ensure that the preconditioner is positive definite (Theorem 3.1). We also prove the condition number of $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ in a form similar to that in the prototype case, i.e., $\frac{1+\epsilon}{1-\epsilon}$ in Theorem 3.2, where ϵ is related to τ. The eigenvalues of $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ cluster inside $[\frac{1}{1+\epsilon}, \frac{1}{1-\epsilon}]$.

The accuracy and effectiveness analysis confirms that, with low-accuracy approximation, the preconditioners can significantly reduce the condition number and improve the eigenvalue clustering.

The SIF framework and preconditioning techniques have some attractive features.

1. The scaling strategy can enhance the compressibility of the off-diagonal blocks in some problems, where those blocks may have singular values very close to each other. An example is the five-point discrete Laplacian matrix from a 2D regular grid, which has the identity matrix as the nonzero subblock in its off-diagonal blocks.
2. The scaling-and-compression strategy also propagates diagonal block information to off-diagonal blocks, so that it is convenient to justify the robustness

and effectiveness. In the earlier work in [29], only the leading diagonal block information is propagated to an off-diagonal block.

3. Unlike the method in [29] that involves Schur complement computations in a sequential fashion, our SIF framework computes hierarchical ULV-type factorizations and avoids the use of sequential Schur complement computations. The construction of the preconditioner follows a binary tree, where the operations at the same level can be performed simultaneously. This potentially leads to much better scalability.
4. It is convenient to obtain a general multilevel preconditioner by repeatedly applying the scaling-and-compression strategy. The cost to apply the resulting preconditioner is $O(N \log N)$, where N is the order of A. A modified version is also provided and costs only $O(N)$ to apply. The $O(N \log N)$ version is generally more effective, while the $O(N)$ version is slightly more robust in practice.
5. Systematic analysis of the approximation accuracy, robustness, and effectiveness is given, not only for the prototype preconditioner, but also for the multilevel one. Multilevel error accumulation is analyzed. The eigenvalue clustering and the resulting condition number are also shown. The studies in [29] are restricted to a special 1-level case.

The performance of the preconditioners is illustrated in some numerical tests. In particular, we use *aggressive compression* (with *very small ranks* in the approximation of the scaled off-diagonal blocks) and get satisfactory convergence in iterative solutions. For some highly ill-conditioned problems such as some interpolation matrices in radial basis function methods, our preconditioners can still preserve positive definiteness and also greatly accelerate the convergence. On the other hand, a structured preconditioner based on direct off-diagonal compression (as in traditional rank structured methods) yields slower convergence, and also fails to be positive definite.

The outline of the paper is as follows. We first present the fundamental SIF preconditioning framework in Section 2. The prototype preconditioner and its analysis are included. The generalization of the framework and analysis to multiple levels is given in Section 3, followed by numerical tests in Section 4. The following notation will be used throughout the discussions.

- $\lambda(A)$ denotes an eigenvalue of A, and we usually use $\lambda(A)$ to mean any eigenvalue of A in general.
- $\lambda_1(A)$ and $\lambda_N(A)$ specifically denote the largest and smallest eigenvalues of A, respectively.
- $\rho(A)$ denotes the spectral radius of A.
- $\kappa(A)$ is the 2-norm condition number of A.
- $\mathrm{diag}(\cdot)$ represents a diagonal or block diagonal matrix with the given diagonal entries/blocks.

2. Fundamental SIF preconditioning framework and analysis. One essential idea of our preconditioners is a scaling-and-compression strategy: the off-diagonal blocks are first *scaled on both sides* by the inverses of the factors of the corresponding diagonal blocks and then *compressed*. The work in [29] only uses the scaling on the left by the inverse of the leading factor, and the scaling on both sides is mentioned without details. The double-sided scaling strategy is first formally mentioned in our presentation [26]. Here, we show that, with the double-sided scaling and compression, it becomes very convenient to justify the effectiveness (and also to generalize to multiple levels in the next section). Also unlike in [29], our schemes do

not need sequential Schur complement computations.

In this section, we present our scaling-and-compression strategy in a prototype preconditioner. Then we rigorously show its properties. The preconditioner and analysis will be generalized to multiple levels in Section 3.

2.1. Scaling-and-compression strategy. Consider an $N \times N$ SPD matrix A partitioned into a block 2×2 form

$$A \equiv \begin{pmatrix} A_{11} & A_{21}^T \\ A_{21} & A_{22} \end{pmatrix}, \tag{2.1}$$

where the two diagonal blocks A_{11} and A_{22} are square matrices. Suppose the Cholesky factorizations of the diagonal blocks look like

$$A_{11} = L_1 L_1^T, \quad A_{22} = L_2 L_2^T. \tag{2.2}$$

Then

$$A = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & C \\ C^T & I \end{pmatrix} \begin{pmatrix} L_1^T & \\ & L_2^T \end{pmatrix}, \tag{2.3}$$

where the identity matrices may have different sizes, and

$$C = L_1^{-1} A_{21}^T L_2^{-T}. \tag{2.4}$$

Compute an SVD

$$C = \begin{pmatrix} U_1 & \hat{U}_1 \end{pmatrix} \begin{pmatrix} \Sigma & \\ & \hat{\Sigma} \end{pmatrix} \begin{pmatrix} U_2^T \\ \hat{U}_2^T \end{pmatrix} = U_1 \Sigma U_2^T + \hat{U}_1 \hat{\Sigma} \hat{U}_2^T, \tag{2.5}$$

where $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$ is for the leading singular values $\sigma_1, \ldots, \sigma_r$ of C, and $\hat{\Sigma} = \mathrm{diag}(\sigma_{r+1}, \sigma_{r+2}, \ldots)$ is for the remaining singular values $\sigma_{r+1}, \sigma_{r+2}, \ldots$ of C. (Σ is a square matrix, and $\hat{\Sigma}$ may be rectangular. $\begin{pmatrix} U_1 & \hat{U}_1 \end{pmatrix}$ and $\begin{pmatrix} U_2 & \hat{U}_2 \end{pmatrix}$ are square matrices.) Suppose all the singular values are ordered from the largest to the smallest, and

$$\sigma_{r+1} \le \tau \le \sigma_r. \tag{2.6}$$

Later, τ will be used as a tolerance for the truncation of the singular values in $\hat{\Sigma}$.

Before we proceed, we give the following simple lemma.

LEMMA 2.1. *Suppose $\tilde{C}$ is an $m \times n$ matrix with singular values $\tilde{\sigma}_i,\ i = 1, 2, \ldots, \min\{m, n\}$. Let $H = \begin{pmatrix} I & \tilde{C} \\ \tilde{C}^T & I \end{pmatrix}$. Then H has an eigenvalue 1 with multiplicity $m + n - 2\min\{m, n\}$, and its remaining eigenvalues are given by*

$$1 \pm \tilde{\sigma}_i,\ i = 1, 2, \ldots, \min\{m, n\}.$$

Furthermore, H is SPD if and only if $\tilde{\sigma}_i < 1,\ i = 1, 2, \ldots, \min\{m, n\}$.

Proof. Suppose $\tilde{\Sigma}$ is an $m \times n$ diagonal matrix with the main diagonal entries given by $\tilde{\sigma}_i,\ i = 1, 2, \ldots, \min\{m, n\}$. Then H is similar to $\begin{pmatrix} I & \tilde{\Sigma} \\ \tilde{\Sigma}^T & I \end{pmatrix}$, whose eigenvalues are either 1 or $1 \pm \tilde{\sigma}_i$.

This result then immediately implies that H is positive definite if and only if $\tilde{\sigma}_i < 1$. ∎

Since A is SPD, so is $\begin{pmatrix} I & C \\ C^T & I \end{pmatrix}$ in (2.3). Then Lemma 2.1 leads to the following result.

LEMMA 2.2. *All the singular values σ_i of C in (2.4) satisfy $\sigma_i < 1,\ i = 1, 2, \ldots$*

To construct a structured incomplete preconditioner, we truncate the singular values of C in (2.4) by dropping $\hat{\Sigma}$ in (2.5). Then

$$A \approx \tilde{A} \equiv \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & U_1 \Sigma U_2^T \\ U_2 \Sigma U_1^T & I \end{pmatrix} \begin{pmatrix} L_1^T & \\ & L_2^T \end{pmatrix}. \tag{2.7}$$

We can continue to factorize the matrix in the middle of the right-hand side so as to obtain a preconditioner. For the analysis purpose, we tentatively use the Cholesky factorization. (Later in Section 2.4, this will be replaced by a more practical ULV factorization.) That is,

$$\begin{pmatrix} I & U_1 \Sigma U_2^T \\ U_2 \Sigma U_1^T & I \end{pmatrix} = \begin{pmatrix} I & \\ U_2 \Sigma U_1^T & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{S} \end{pmatrix} \begin{pmatrix} I & U_1 \Sigma U_2^T \\ & I \end{pmatrix}, \tag{2.8}$$

where $\tilde{S}$ is the Schur complement

$$\tilde{S} = I - U_2 \Sigma^2 U_2^T. \tag{2.9}$$

From Proposition 2.3 below, we can see that $\tilde{S}$ is positive definite, so assume $\tilde{S} = \tilde{D}_2 \tilde{D}_2^T$ is its Cholesky factorization. Thus,

$$\begin{pmatrix} I & U_1 \Sigma U_2^T \\ U_2 \Sigma U_1^T & I \end{pmatrix} = \begin{pmatrix} I & \\ U_2 \Sigma U_1^T & \tilde{D}_2 \end{pmatrix} \begin{pmatrix} I & U_1 \Sigma U_2^T \\ & \tilde{D}_2^T \end{pmatrix}.$$

Then a prototype preconditioner looks like

$$\tilde{A} = \tilde{\mathbf{L}} \tilde{\mathbf{L}}^T, \quad \text{with} \tag{2.10}$$

$$\tilde{\mathbf{L}} = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & \\ U_2 \Sigma U_1^T & \tilde{D}_2 \end{pmatrix} = \begin{pmatrix} L_1 & \\ L_2 U_2 \Sigma U_1^T & L_2 \tilde{D}_2 \end{pmatrix}.$$

REMARK 2.1. Our scheme here is more general than the one in [29], and there are some fundamental differences.

1. Double-sided scaling $L_1^{-1} A_{21}^T L_2^{-T}$ is used here in (2.4), while single-sided scaling $L_1^{-1} A_{21}^T$ is used in [29]. The double-sided scaling and compression make it convenient to control the approximation accuracy of $\tilde{A}$, and can be used to produce more general and comprehensive analysis. See the next subsection. The analysis in [29] is restricted to a special case.
2. Explicit Schur complement computation is needed in [29] to get a structured Cholesky factorization, which is essentially a sequential process. Our ULV factorization in Section 2.4 will avoid the Schur complements. The mentioning of the Schur complement above is only for the analysis purpose, and is not needed in the implementation.
3. The generalization of the prototype preconditioner (based on 1-level block 2×2 partitioning) to multiple levels is also more natural via repeated application. We will further provide accuracy, effectiveness, and robustness analysis for our multilevel version. See Section 3. No such type of analysis is available in [29].

2.2. Robustness and accuracy. We then discuss the robustness and accuracy of the prototype preconditioner $\tilde{A}$ in (2.7) (and also (2.10)). We show that $\tilde{A}$ is guaranteed to be positive definite, and it approximates A by a relative error bound precisely controlled by the (absolute) compression tolerance τ. The tolerance τ also controls the approximation accuracy of $\tilde{\mathbf{L}}$.

PROPOSITION 2.3. *$\tilde{A}$ in (2.7) resulting from the truncation of $\hat{\Sigma}$ in (2.5) is always positive definite.*

Proof. There are multiple ways to show the result. We prove it from an aspect that clearly reflects the robustness of the preconditioner. Since $\begin{pmatrix} I & C \\ C^T & I \end{pmatrix}$ is SPD, the following Schur complement is also SPD:

$$S \equiv I - C^T C = I - U_2 \Sigma^2 U_2^T - \hat{U}_2 \hat{\Sigma}^T \hat{\Sigma} \hat{U}_2^T. \tag{2.11}$$

(Note that $\hat{\Sigma}$ is allowed to be rectangular.) On the other hand, (2.9) means

$$\tilde{S} = S + \hat{U}_2 \hat{\Sigma}^T \hat{\Sigma} \hat{U}_2^T. \tag{2.12}$$

Thus, $\tilde{S}$ in (2.8) is SPD. Accordingly, $\begin{pmatrix} I & U_1 \Sigma U_2^T \\ U_2 \Sigma U_1^T & I \end{pmatrix}$ in (2.8) is SPD. (2.7) then indicates $\tilde{A}$ is SPD. (The relationship in (2.12) is consistent with the idea of Schur monotonicity or Schur compensation in [12, 29], and leads to the enhanced robustness of $\tilde{A}$.)

An alternative proof can be based on Lemma 2.1. ∎

We then show the approximation accuracy of $\tilde{A}$ and $\tilde{\mathbf{L}}$ in terms of the truncation tolerance τ.

THEOREM 2.4. *Suppose $\mathbf{L}$ is the exact lower triangular Cholesky factor of A. Then*

$$\|A - \tilde{A}\|_2 \le \tau \|A\|_2,$$
$$\|\mathbf{L} - \tilde{\mathbf{L}}\|_2 \le \tau \left(1 + \frac{2 d_n \sqrt{1-\sigma_n^2}}{1-\sigma_1^2} \tau\right) \|\mathbf{L}\|_2,$$

where $d_n = \frac{1}{2} + \lceil \log_2 n \rceil$ and n is the column size of C.

Proof. Note that (2.6) and Lemma 2.2 imply $\tau < 1$. According to (2.3), (2.5), and (2.7), the approximation error matrix is given by

$$\begin{aligned} A - \tilde{A} &= \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} 0 & \hat{U}_1 \hat{\Sigma} \hat{U}_2^T \\ \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T & 0 \end{pmatrix} \begin{pmatrix} L_1^T & \\ & L_2^T \end{pmatrix} \\ &= \begin{pmatrix} 0 & L_1 \hat{U}_1 \hat{\Sigma} \hat{U}_2^T L_2^T \\ L_2 \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T L_1^T & 0 \end{pmatrix}. \end{aligned}$$

Thus,

$$\begin{aligned} \|A - \tilde{A}\|_2 &= \|L_1 \hat{U}_1 \hat{\Sigma} \hat{U}_2^T L_2^T\|_2 \le \|L_1\|_2 \|\hat{\Sigma}\|_2 \|L_2\|_2 \\ &= \sigma_{r+1} \sqrt{\|A_{11}\|_2 \|A_{22}\|_2} \le \tau \|A\|_2, \end{aligned}$$

where we have used the relationship between the 2-norm of an SPD matrix and the 2-norm of its Cholesky factor.

Next, consider the approximation accuracy of $\tilde{\mathbf{L}}$. Suppose $S = D_2 D_2^T$ is the Cholesky factorization of S in (2.11). According to (2.3) and (2.5),

$$\mathbf{L} = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & \\ C^T & D_2 \end{pmatrix} = \begin{pmatrix} L_1 & \\ L_2(U_2 \Sigma U_1^T + \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T) & L_2 D_2 \end{pmatrix}. \tag{2.13}$$

This together with (2.10) yields

$$\mathbf{L} - \tilde{\mathbf{L}} = \begin{pmatrix} 0 & 0 \\ L_2 \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T & L_2(D_2 - \tilde{D}_2) \end{pmatrix}.$$

Thus,

$$\begin{aligned} \|\mathbf{L} - \tilde{\mathbf{L}}\|_2 &= \left\| \begin{pmatrix} L_2 \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T & L_2(D_2 - \tilde{D}_2) \end{pmatrix} \right\|_2 \\ &\le \|L_2\|_2 \left\| \begin{pmatrix} \hat{U}_2 \hat{\Sigma}^T \hat{U}_1^T & D_2 - \tilde{D}_2 \end{pmatrix} \right\|_2 \\ &\le \|L_2\|_2 (\tau + \|D_2 - \tilde{D}_2\|_2) \\ &= (\tau + \|D_2 - \tilde{D}_2\|_2) \sqrt{\|A_{22}\|_2} \\ &\le (\tau + \|D_2 - \tilde{D}_2\|_2) \sqrt{\|A\|_2} \\ &= (\tau + \|D_2 - \tilde{D}_2\|_2) \|\mathbf{L}\|_2. \end{aligned}$$

According to [9], the Cholesky factor $\tilde{D}_2$ of $\tilde{S}$ satisfies the approximation error bound

$$\|D_2 - \tilde{D}_2\|_2 \le 2 d_n \|S^{-1}\|_2 \|D_2\|_2 \|S - \tilde{S}\|_2 = 2 d_n \|S^{-1}\|_2 \|D_2\|_2 \sigma_{r+1}^2,$$

where $d_n = 1/2 + \lceil \log_2 n \rceil$ and n is the column size of C (and also the size of S). From (2.5) and (2.11) and noticing Lemmas 2.1–2.2, we can see that

$$\|S^{-1}\|_2 = \frac{1}{1 - \sigma_1^2}, \quad \|D_2\|_2 = \sqrt{1 - \sigma_n^2}.$$

Thus,

$$\begin{aligned} \|D_2 - \tilde{D}_2\|_2 &\le 2 d_n \frac{\sqrt{1 - \sigma_n^2}}{1 - \sigma_1^2} \sigma_{r+1}^2 \le \frac{2 d_n \sqrt{1 - \sigma_n^2}}{1 - \sigma_1^2} \tau^2, \\ \|\mathbf{L} - \tilde{\mathbf{L}}\|_2 &\le \left(\tau + \frac{2 d_n \sqrt{1 - \sigma_n^2}}{1 - \sigma_1^2} \tau^2 \right) \|\mathbf{L}\|_2 = \tau \left(1 + \frac{2 d_n \sqrt{1 - \sigma_n^2}}{1 - \sigma_1^2} \tau \right) \|\mathbf{L}\|_2. \end{aligned}$$

□

Theorem 2.4 and Lemma 2.2 indicate that the (absolute) tolerance τ in (2.6) for the compression of C essentially plays a role of a *relative error bound* in the approximation of A by $\tilde{A}$. τ also controls the relative approximation accuracy of the Cholesky factor $\mathbf{L}$. In fact, if τ is not too large and satisfies $\frac{2 d_n \sqrt{1 - \sigma_n^2}}{1 - \sigma_1^2} \tau = O(1)$, then $\|\mathbf{L} - \tilde{\mathbf{L}}\|_2 = O(\tau) \|\mathbf{L}\|_2$.

2.3. Effectiveness of the preconditioner. Theorem 2.4 indicates that we can use τ to control how accurately $\tilde{A}$ approximates A and $\tilde{\mathbf{L}}$ approximates $\mathbf{L}$. Moreover, since we are interested in preconditioning, we would like to show that τ does not have to be very small to yield an effective preconditioner. A larger tolerance τ means a

more compact SIF preconditioner that is more efficient to apply. Note that our scheme here is different and more general than the one in [29] (see Remark 2.1). However, we can still prove effectiveness results similar to those in [29] (and even stronger), though the double-sided scaling makes the analysis much less trivial here.

THEOREM 2.5. *For* $\tilde{\mathbf{L}}$ *in (2.10), we have*

$$\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T} = \begin{pmatrix} I & \hat{C} \\ \hat{C}^T & I \end{pmatrix},$$

where $\hat{C} = \hat{U}_1\hat{\Sigma}\hat{U}_2^T\tilde{D}_2^{-T}$ *and*

$$\|\hat{C}\|_2 = \sigma_{r+1} \le \tau. \tag{2.14}$$

Moreover,

$$\|\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T} - I\|_2 = \sigma_{r+1}, \tag{2.15}$$

$$|\lambda(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) - 1| \le \sigma_{r+1}, \tag{2.16}$$

$$\kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) = \frac{1+\sigma_{r+1}}{1-\sigma_{r+1}} \le \frac{1+\tau}{1-\tau}. \tag{2.17}$$

Proof. According to (2.10) and (2.13),

$$\begin{aligned}
\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T} &= \begin{pmatrix} I & \\ U_2\Sigma U_1^T & \tilde{D}_2 \end{pmatrix}^{-1} \begin{pmatrix} I & \\ C^T & D_2 \end{pmatrix} \begin{pmatrix} I & C \\ & D_2^T \end{pmatrix} \begin{pmatrix} I & U_1\Sigma U_2^T \\ & \tilde{D}_2^T \end{pmatrix}^{-1} \\
&= \begin{pmatrix} I & \\ & \tilde{D}_2^{-1} \end{pmatrix} \begin{pmatrix} I & \\ C^T - U_2\Sigma U_1^T & D_2 \end{pmatrix} \begin{pmatrix} I & C - U_1\Sigma U_2^T \\ & D_2^T \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{D}_2^{-T} \end{pmatrix} \\
&= \begin{pmatrix} I & \\ & \tilde{D}_2^{-1} \end{pmatrix} \begin{pmatrix} I & \\ \hat{U}_2\hat{\Sigma}^T\hat{U}_1^T & D_2 \end{pmatrix} \begin{pmatrix} I & \hat{U}_1\hat{\Sigma}\hat{U}_2^T \\ & D_2^T \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{D}_2^{-T} \end{pmatrix} \\
&= \begin{pmatrix} I & \hat{U}_1\hat{\Sigma}\hat{U}_2^T\tilde{D}_2^{-T} \\ \tilde{D}_2^{-1}\hat{U}_2\hat{\Sigma}^T\hat{U}_1^T & \tilde{D}_2^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T + S)\tilde{D}_2^{-T} \end{pmatrix} \\
&= \begin{pmatrix} I & \hat{U}_1\hat{\Sigma}\hat{U}_2^T\tilde{D}_2^{-T} \\ \tilde{D}_2^{-1}\hat{U}_2\hat{\Sigma}^T\hat{U}_1^T & \tilde{D}_2^{-1}\tilde{S}\tilde{D}_2^{-T} \end{pmatrix} = \begin{pmatrix} I & \hat{C} \\ \hat{C}^T & I \end{pmatrix}.
\end{aligned}$$

Next, we show (2.14). Notice that the 2-norm of a symmetric matrix is equal to its spectral radius. We have

$$\begin{aligned}
\|\hat{C}\|_2^2 = \|\hat{C}^T\hat{C}\|_2 &= \|\tilde{D}_2^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T)\tilde{D}_2^{-T}\|_2 \\
&= \rho(\tilde{D}_2^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T)\tilde{D}_2^{-T}) = \rho(\tilde{D}_2^{-T}\tilde{D}_2^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T)) \\
&= \rho(\tilde{S}^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T) \\
&= \rho((I - U_2\Sigma^2U_2^T)^{-1}(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T).
\end{aligned}$$

According to the Sherman-Morrison-Woodbury formula,

$$\begin{aligned}
(I - U_2\Sigma^2U_2^T)^{-1} &= [I - (U_2\Sigma)(U_2\Sigma)^T]^{-1} \\
&= I - (U_2\Sigma)[I + (U_2\Sigma)^T(U_2\Sigma)]^{-1}(U_2\Sigma)^T. \\
&= I - U_2\Sigma(I + \Sigma^2)^{-1}\Sigma^TU_2^T.
\end{aligned}$$

Thus,

$$\begin{aligned}\|\hat{C}\|_2^2 &= \rho([I - U_2\Sigma(I+\Sigma^2)^{-1}\Sigma^T U_2^T](\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T)) \\ &= \rho(\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T) = \|\hat{U}_2\hat{\Sigma}^T\hat{\Sigma}\hat{U}_2^T\|_2 = \sigma_{r+1}^2,\end{aligned}$$

where $U_2^T\hat{U}_2 = 0$ is used.

The remaining results can then be conveniently shown. (2.15) is obvious. Lemma 2.1 means that $\begin{pmatrix} I & \hat{C} \\ \hat{C}^T & I \end{pmatrix}$ has the largest eigenvalue $1+\sigma_{r+1}$ and the smallest eigenvalue $1-\sigma_{r+1}$. This leads to (2.16) and (2.17). ∎

Theorem 2.5 indicates that τ also controls how close the preconditioned matrix $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ is to I and how closely $\lambda(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T})$ clusters around 1.

Moreover, the effectiveness of the preconditioner (when τ is not small) can be interpreted from the theorem similarly to [29]. That is, when the singular values σ_i of C decay, the condition number $\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$ after truncation decays much faster. (In [29], this is illustrated in terms of a special case.) A similar study of the decay property is also done in [17] for some Schur complement based preconditioners. Here, we can further rigorously characterize this as follows. Suppose $s(t)$ is a differentiable and non-increasing function with $t > 0$, $0 \le s(t) < 1$. Its decay rate at t can be reflected by its slope $s'(t)$. Then for $f(t) = \frac{1+s(t)}{1-s(t)}$,

$$f'(t) = \theta(t)s'(t), \quad \text{with} \quad \theta(t) = \frac{2}{(1-s(t))^2}.$$

This indicates that, the slope of $f(t)$ at t is that of $s(t)$ enlarged by a *magnification factor* $\theta(t)$. The closer t is to 0 (or s is to 1), the larger $\theta(t)$ is and the faster $f(t)$ decays. Accordingly, this means that a relatively small t can already bring $f(t)$ to a reasonable magnitude.

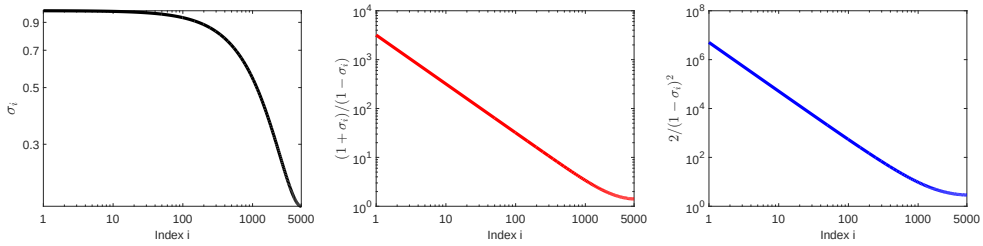
That is, by keeping a relatively small number of largest singular values σ_i of C ($\sigma_i \in [0,1)$), we can get a reasonable condition number after preconditioning. Notice that the singular value truncation is controlled by (2.6). We can thus use a relatively large tolerance τ for preconditioning. For example, for selected values of τ, the corresponding bounds for $\kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T})$ and the decay magnification factors are given in Table 2.1. For τ as large as 0.95, we already have a reasonable condition number bound $\frac{1+\tau}{1-\tau} = 39$. In practice, the faster σ_i decays, the better the preconditioner works.

TABLE 2.1
Understanding the decay behavior of the condition number of the preconditioned matrix in terms of the decay of the singular values σ_i when the singular values smaller than or equal to τ are truncated.

τ	0.999	0.99	0.95	0.9
$\frac{2}{(1-\tau)^2}$	2×10^6	2×10^4	8×10^2	2×10^2
$\frac{1+\tau}{1-\tau}$	1999	199	39	19

In particular, for a 5-point discrete 2D Laplacian matrix A evenly partitioned into a block 2×2 form, we can compute C and investigate its singular values σ_i, and then show the resulting condition number if all the singular values σ_i satisfying $\sigma_i \le \tau$ are truncated. See Figure 2.1. Clearly, for smaller indices i, the magnification factors θ

are larger and the condition number decays faster. By keeping only a small number of singular values or using a small numerical rank r, we can already get a reasonable condition number.



(a) Singular value σ_i (b) Condition number $\frac{1+\sigma_i}{1-\sigma_i}$ (c) Magnification factor $\theta_i = \frac{2}{(1-\sigma_i)^2}$

FIG. 2.1. *For a 5-point discrete Laplacian matrix A evenly partitioned into a block 2×2 form, (a) shows how the nonzero singular values σ_i of C decay, (b) shows how the condition number of the preconditioned matrix decays when the singular values smaller than or equal to σ_i are truncated, and (c) shows the factor that magnifies the decay.*

REMARK 2.2. For some model problems such as the 5-point discrete Laplacian matrix, the analytical derivation of the singular values of C will appear in [31]. It can further be shown that $\kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) = O(\sqrt{N})$ when r is set to be $O(1)$. This is beyond the focus here.

2.4. Basic ULV factorization scheme. As mentioned in Remark 2.1, the preconditioning scheme in [29] has several limitations, one of which is the sequential computation of Schur complements. This can be avoided by using a ULV-type factorization that is similar to (but simpler than) the ones in [5, 28] for hierarchically semiseparable (HSS) matrices. The idea is to introduce zeros into the scaled $(1,2)$ and $(2,1)$ off-diagonal blocks simultaneously and then obtain reduced matrices. More specifically, for $U_i, i = 1,2$ in (2.7), suppose Q_i is an orthogonal matrix such that $Q_i^T U_i = \begin{pmatrix} 0 \\ I \end{pmatrix}$. Then

$$\begin{pmatrix} I & U_1\Sigma U_2^T \\ U_2\Sigma U_1^T & I \end{pmatrix}$$
$$= \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \begin{pmatrix} I & \begin{pmatrix} 0 \\ I \end{pmatrix}\Sigma\begin{pmatrix} 0 & I \end{pmatrix} \\ \begin{pmatrix} 0 \\ I \end{pmatrix}\Sigma\begin{pmatrix} 0 & I \end{pmatrix} & I \end{pmatrix} \begin{pmatrix} Q_1^T & \\ & Q_2^T \end{pmatrix}.$$

Let Π_3 be an appropriate permutation matrix such that

$$\begin{pmatrix} I & U_1\Sigma U_2^T \\ U_2\Sigma U_1^T & I \end{pmatrix} = \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \Pi_3 \begin{pmatrix} I & \\ & D_3 \end{pmatrix} \Pi_3^T \begin{pmatrix} Q_1^T & \\ & Q_2^T \end{pmatrix},$$

where $D_3 = \begin{pmatrix} I & \Sigma \\ \Sigma & I \end{pmatrix}$ is a small $(2r \times 2r)$ matrix called *reduced matrix*. Π_3 is used to assemble D_3. Then directly compute a Cholesky factorization of $D_3 = L_3 L_3^T$ to

complete the factorization. This in turn gives a factorization of $\tilde{A}$ in (2.7) that looks like

$$\tilde{A} = \tilde{\mathbf{L}}_3 \tilde{\mathbf{L}}_3^T, \quad \text{with} \tag{2.18}$$
$$\tilde{\mathbf{L}}_3 = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} Q_1 & \\ & Q_2 \end{pmatrix} \Pi_3 \begin{pmatrix} I & \\ & L_3 \end{pmatrix}.$$

$\tilde{\mathbf{L}}_3$ is said to be a *ULV factor*, since it is given by a sequence of orthogonal and triangular matrices. (The term "ULV factor" here follows the name in HSS ULV factorizations [5], and differs from the traditional sense of ULV factors.) Notice that the operations associated with $i = 1, 2$ can be performed independently, and this avoids the sequential computation of a Schur complement.

Thus, the overall SIF framework includes the scaling-and-compression strategy followed by the ULV factorization. With the 1-level block 2×2 partitioning, we get the prototype preconditioner (2.18).

3. Practical multilevel SIF preconditioners and analysis. The fundamental ideas for the prototype preconditioner in the previous section can be conveniently generalized to practical preconditioners with multiple levels of partitioning. In the following, we describe our multilevel SIF framework, including a multilevel scaling-and-compression strategy combined with a multilevel ULV factorization. We have two versions for the multilevel generalization, with some differences in the performance. To describe the preconditioners and analysis, we list commonly used notation as follows.

- $\mathcal{I} = \{1 : N\}$ is the entire index set for A. For an index subset $s_i \subset \mathcal{I}$, $\mathcal{I} \backslash s_i$ is its complement.
- $\mathcal{T}$ represents a postordered binary tree with the nodes labeled as $i = 1, 2, \ldots,$ root($\mathcal{T}$), where root($\mathcal{T}$) denotes the root node. Suppose root($\mathcal{T}$) is at level 0, and the leaves are at the bottom level $\mathcal{L}$.
- sib(i) and par(i) denote the sibling and parent of node i in $\mathcal{T}$, respectively.
- Each node i of $\mathcal{T}$ is associated with a set s_i of consecutive indices. These sets satisfy $s_{\text{root}(\mathcal{T})} = \mathcal{I}$, and $s_i = s_{c_1} \cup s_{c_2}$, $s_{c_1} \cap s_{c_2} = \emptyset$ if i is a nonleaf node with children c_1 and c_2.
- $A|_{s_i \times s_j}$ represents the submatrix of A with row index set s_i and column index set s_j.
- $b|_{s_i}$ represents the vector selected from a vector b with the index set s_i.

3.1. Multilevel SIF preconditioner. To generalize the ideas in the previous section to multiple levels, we can apply the scheme repeatedly to the diagonal blocks A_{11} and A_{22} in (2.1). This can be done following a binary tree $\mathcal{T}$, and the operations corresponding to the nodes at the same level of $\mathcal{T}$ can be performed simultaneously. We can further avoid the top-down recursion by organizing all the local operations along a bottom-up traversal of $\mathcal{T}$. At each level of $\mathcal{T}$, suppose A is partitioned following the index set s_i associated with each node i at that level. The number of partition levels $\mathcal{L}$ will be decided at the end of this subsection.

In the traversal of $\mathcal{T}$, for a leaf node i, directly compute the Cholesky factorization of the corresponding diagonal block

$$A|_{s_i \times s_i} = L_i L_i^T. \tag{3.1}$$

Also for later notational consistency, set $\tilde{\mathbf{L}}_i \equiv L_i$. For a nonleaf node i, suppose c_1

and c_2 are its left and right children, respectively. Then

$$A|_{s_i\times s_i} = \begin{pmatrix} A|_{s_{c_1}\times s_{c_1}} & A|_{s_{c_1}\times s_{c_2}} \\ A|_{s_{c_2}\times s_{c_1}} & A|_{s_{c_2}\times s_{c_2}} \end{pmatrix}.$$

By induction as in (3.6) below, the diagonal blocks have been (approximately) factorized as

$$A|_{s_{c_1}\times s_{c_1}} \approx \tilde{\mathbf{L}}_{c_1}\tilde{\mathbf{L}}_{c_1}^T, \quad A|_{s_{c_2}\times s_{c_2}} \approx \tilde{\mathbf{L}}_{c_2}\tilde{\mathbf{L}}_{c_2}^T. \tag{3.2}$$

Thus,

(3.3)

$$A|_{s_i\times s_i} \approx \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} & \\ & \tilde{\mathbf{L}}_{c_2} \end{pmatrix} \begin{pmatrix} I & \tilde{\mathbf{L}}_{c_1}^{-1}A|_{s_{c_1}\times s_{c_2}}\tilde{\mathbf{L}}_{c_2}^{-T} \\ \tilde{\mathbf{L}}_{c_2}^{-1}A|_{s_{c_2}\times s_{c_1}}\tilde{\mathbf{L}}_{c_1}^{-T} & I \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{L}}_{c_1}^T & \\ & \tilde{\mathbf{L}}_{c_2}^T \end{pmatrix}.$$

Then just like the 1-level scheme in Section 2, we computed the compression (approximate SVD)

$$\tilde{\mathbf{L}}_{c_1}^{-1}A|_{s_{c_1}\times s_{c_2}}\tilde{\mathbf{L}}_{c_2}^{-T} \approx U_{c_1}\Sigma_{c_1}U_{c_2}^T, \tag{3.4}$$

and introduce zeros into this block by using orthogonal matrices Q_{c_1} and Q_{c_2} such that

$$Q_{c_1}^T U_{c_1} = \begin{pmatrix} 0 \\ I \end{pmatrix}, \quad Q_{c_2}^T U_{c_2} = \begin{pmatrix} 0 \\ I \end{pmatrix}. \tag{3.5}$$

Since $\tilde{\mathbf{L}}_{c_1}$ and $\tilde{\mathbf{L}}_{c_2}$ are structured factors, the formation of $\tilde{\mathbf{L}}_{c_1}^{-1}A|_{s_{c_1}\times s_{c_2}}\tilde{\mathbf{L}}_{c_2}^{-T}$ in (3.4) involves multiple structured solutions. This yields a factorization just like in (2.18):

$$A|_{s_i\times s_i} \approx \tilde{\mathbf{L}}_i\tilde{\mathbf{L}}_i^T, \quad \text{with} \tag{3.6}$$

$$\tilde{\mathbf{L}}_i = \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} & \\ & \tilde{\mathbf{L}}_{c_2} \end{pmatrix} \begin{pmatrix} Q_{c_1} & \\ & Q_{c_2} \end{pmatrix} \Pi_i \begin{pmatrix} I & \\ & L_i \end{pmatrix},$$

where Π_i is an appropriate permutation matrix to assemble the reduced matrix $D_i = \begin{pmatrix} I & \Sigma_{c_1} \\ \Sigma_{c_1} & I \end{pmatrix}$, and L_i is the lower triangular Cholesky factor of D_i. (Later in Section 3.2, we will discuss a condition that guarantees the positive definiteness.) The formula gives a recursive representation for $\tilde{\mathbf{L}}_i$.

Thus, it is clear that the algorithm generates a sequence of orthogonal matrices Q_i and triangular matrices L_i. These matrices Q_i, L_i (together with the rotations Π_i) define a ULV factor $\tilde{\mathbf{L}}$ in an approximate ULV factorization

$$A \approx \tilde{A} \equiv \tilde{\mathbf{L}}\tilde{\mathbf{L}}^T, \tag{3.7}$$

where $\tilde{\mathbf{L}} \equiv \tilde{\mathbf{L}}_{\mathrm{root}(\mathcal{T})}$. Our multilevel SIF preconditioner is then $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$.

The application of the preconditioner is also done following the traversal of $\mathcal{T}$. For example, consider the solution of the following linear system via forward substitution:

$$\tilde{\mathbf{L}}y = b.$$

Partition b into pieces $b|_{s_i}$ following the leaf level index sets s_i. For each leaf i, compute

$$y_i = Q_i^T L_i^{-1} b|_{s_i}.$$

For each nonleaf node i with children c_1 and c_2, compute

$$y_i = Q_i^T \begin{pmatrix} I & \\ & L_i^{-1} \end{pmatrix} \Pi_i^T \begin{pmatrix} y_{c_1} \\ y_{c_2} \end{pmatrix}.$$

When root($\mathcal{T}$) is reached, we obtain the solution $y \equiv y_{\mathrm{root}(\mathcal{T})}$. The backward substitution is done similarly.

The costs of the construction and application of the preconditioner can be conveniently counted. Suppose r is the maximum rank used for the compression or truncated SVD in (3.4). The repeated diagonal partition is done until the finest level diagonal block sizes are around r. Thus, $\mathcal{L} \approx \log \frac{N}{r}$. Then the approximate factorization costs $O(rN^2 \log \frac{N}{r})$ flops. (Notice that A is a general SPD matrix. This cost can be reduced when A is sparse or has some special properties.) To get a preconditioner, we set $r = O(1)$. The cost to apply the resulting preconditioner is $O(N \log N)$. The storage is also $O(N \log N)$. The $\log N$ term in the costs is due to the size of Q_i, which is roughly $\frac{N}{2^l}$ if i is at level l. (Q_i results from the extension of U_i into an orthogonal matrix, which can be done via Householder transformations for convenience. See (3.5). Thus, Q_i can be represented in terms of r Householder vectors.) We can remove the $\log N$ term in a modified scheme in Section 3.3.

3.2. Robustness, accuracy, and effectiveness of the multilevel preconditioner. We then provide analysis for the multilevel preconditioner. For convenience, introduce the notation $A^{(l)}$ corresponding to the levels $l = 0, 1, \ldots, \mathcal{L}$ of $\mathcal{T}$ so as to track the error accumulation at each level. Initially, $A^{(\mathcal{L})} \equiv A$. For each (leaf) node i at level $\mathcal{L}$, we compute the Cholesky factorization (3.1). At this level, no approximation is involved. Then for each node i at level $l < \mathcal{L}$, the scaling-and-compression strategy in (3.2)–(3.4) is applied, producing an approximate factorization in (3.6). This yields the approximation of $A^{(l+1)}$ by $A^{(l)}$, where the diagonal block $A^{(l+1)}|_{s_i \times s_i}$ is approximated by $A^{(l)}|_{s_i \times s_i} \equiv \tilde{\mathbf{L}}_i \tilde{\mathbf{L}}_i^T$. This process then continues, and produces a sequence of approximations to A:

$$A^{(\mathcal{L})} (\equiv A) \implies A^{(\mathcal{L}-1)} \implies \cdots \implies A^{(0)} (\equiv \tilde{A}), \tag{3.8}$$

where $\tilde{A}$ is the final preconditioner in a factorized form.

Since $A^{(\mathcal{L}-1)}$ approximates A with modified diagonal blocks, $A^{(\mathcal{L}-1)}$ may not be positive definite. Similarly, $A^{(l)}$ for $l < L$ may not be positive definite. We would like to study the levelwise error accumulation and give a condition for $A^{(l)}$ to remain positive definite.

THEOREM 3.1. *Let τ be the tolerance of the truncated SVD in the scaling-and-compression strategy in (3.2)–(3.4) to generate the matrices $A^{(l)}$, $l = \mathcal{L}-1, \mathcal{L}-2, \ldots, 0$ in (3.8) with $\mathcal{L} \geq 1$. If*

$$(1+\tau)^{\mathcal{L}} < 1 + \frac{1}{\kappa(A)}, \tag{3.9}$$

then each $A^{(l)}$ is positive definite, and

$$\|A - \tilde{A}\|_2 \leq [(1+\tau)^{\mathcal{L}} - 1] \|A\|_2.$$

Proof. (3.9) means

$$[(1+\tau)^{\mathcal{L}} - 1] \|A\|_2 < \lambda_N(A), \tag{3.10}$$

where $\lambda_N(A)$ is the smallest eigenvalue of A. For $l = \mathcal{L}-1, \mathcal{L}-2, \ldots, 0$, we show

$$\|A - A^{(l)}\|_2 \leq [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2 < \lambda_N(A).$$

Then Weyl's Theorem means that each $A^{(l)}$ is SPD.

For $l = \mathcal{L}-1$, the matrix $A^{(\mathcal{L}-1)}$ is obtained from A through the 1-level scaling and compression, where $A|_{s_i \times s_i}$ is approximated as in (3.6) for each node i at level $\mathcal{L}-1$. Thus, Theorem 2.4 indicates

$$\|A|_{s_i \times s_i} - A^{(\mathcal{L}-1)}|_{s_i \times s_i}\|_2 \leq \tau \|A|_{s_i \times s_i}\|_2.$$

Accordingly,

$$\|A - A^{(\mathcal{L}-1)}\|_2 \leq \tau \max_{i \text{ at level } \mathcal{L}-1} \{\|A|_{s_i \times s_i}\|_2\} \leq \tau \|A\|_2.$$

Since $\tau \leq (1+\tau)^{\mathcal{L}} - 1$, (3.10) means

$$\|A - A^{(\mathcal{L}-1)}\|_2 \leq \tau\|A\|_2 \leq [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2 < \lambda_N(A).$$

Thus, $A^{(\mathcal{L}-1)}$ is SPD.

The factorization can then proceed to level $l = \mathcal{L}-2$ (when $\mathcal{L} \geq 2$). Similarly to the argument as above, we have

$$\|A^{(\mathcal{L}-1)} - A^{(\mathcal{L}-2)}\|_2 \leq \tau \|A^{(\mathcal{L}-1)}\|_2.$$

By the triangle inequality,

$$\begin{aligned} \|A - A^{(\mathcal{L}-2)}\|_2 &\leq \|A - A^{(\mathcal{L}-1)}\|_2 + \|A^{(\mathcal{L}-1)} - A^{(\mathcal{L}-2)}\|_2 \\ &\leq \|A - A^{(\mathcal{L}-1)}\|_2 + \tau\|A^{(\mathcal{L}-1)}\|_2 \\ &\leq \|A - A^{(\mathcal{L}-1)}\|_2 + \tau(\|A - A^{(\mathcal{L}-1)}\|_2 + \|A\|_2) \\ &\leq (1+\tau)\|A - A^{(\mathcal{L}-1)}\|_2 + \tau\|A\|_2 \\ &\leq [(1+\tau)+1]\tau\|A\|_2 = [(1+\tau)^2 - 1]\|A\|_2. \end{aligned} \tag{3.11}$$

Then by (3.10),

$$\|A - A^{(\mathcal{L}-2)}\|_2 \leq [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2 < \lambda_N(A).$$

This leads to the positive definiteness of $A^{(\mathcal{L}-2)}$.

The process then proceeds by induction. In fact, just like in (3.11), for level $l < \mathcal{L}-1$, we can get an error accumulation pattern

$$\|A - A^{(l)}\|_2 \leq (1+\tau)\|A - A^{(l+1)}\|_2 + \tau\|A\|_2.$$

Thus,

$$\begin{aligned} \|A - A^{(0)}\|_2 &\leq (1+\tau)\|A - A^{(1)}\|_2 + \tau\|A\|_2 \\ &\leq (1+\tau)[(1+\tau)\|A - A^{(2)}\|_2 + \tau\|A\|_2] + \tau\|A\|_2 \\ &\leq \cdots \leq [(1+\tau)^{L-1} + \cdots + (1+\tau) + 1]\tau\|A\|_2 \\ &= [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2 < \lambda_N(A). \end{aligned}$$

This completes the proof. ∎

Thus, if τ is smaller than roughly $\frac{1}{\mathcal{L}\kappa(A)}$, then the preconditioner $\tilde{A}$ is always positive definite, and it approximates A to the accuracy about $\mathcal{L}\tau$. Of course, this is still a very conservative estimate. In practice, the positive definiteness is often well preserved by $\tilde{A}$ for various ill-conditioned matrices A even with relative large τ. See the examples in Section 4. Our modified scheme in Section 3.3 has even better robustness in practice. (In our future work, we will investigate the feasibility of improving the methods and analysis to avoid the term $\kappa(A)$ in the condition.)

The following theorem shows the effectiveness of the multilevel preconditioner in terms of the eigenvalues and the condition number of the preconditioned matrix $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$. We also show how close $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ is to I.

THEOREM 3.2. *With the same conditions as in Theorem 3.1 and with* $\tilde{\mathbf{L}}$ *in (3.7), the eigenvalues of* $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ *satisfies*

$$\frac{1}{1+\epsilon} \le \lambda(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) \le \frac{1}{1-\epsilon}, \tag{3.12}$$

where $\epsilon = [(1+\tau)^{\mathcal{L}} - 1]\kappa(A) < 1$. *Accordingly,*

$$\begin{aligned} \|\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T} - I\|_2 &\le \frac{\epsilon}{1-\epsilon}, \\ \kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) &\le \frac{1+\epsilon}{1-\epsilon}. \end{aligned} \tag{3.13}$$

Proof. (3.9) means $\epsilon < 1$. Note that the eigenvalues of $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}$ are the inverses of the eigenvalues of $\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}$, where $A = \mathbf{L}\mathbf{L}^T$. By Theorem 3.1,

$$\begin{aligned} \|\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T} - I\|_2 &= \|\mathbf{L}^{-1}(\tilde{A} - A)\mathbf{L}^{-T}\|_2 \\ &\le \|A - \tilde{A}\|_2\|\mathbf{L}^{-1}\|_2\|\mathbf{L}^{-T}\|_2 \\ &\le [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2\|\mathbf{L}^{-1}\|_2\|\mathbf{L}^{-T}\|_2 \\ &= [(1+\tau)^{\mathcal{L}} - 1]\|A\|_2\|A^{-1}\|_2 = \epsilon. \end{aligned}$$

Note that

$$\|\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T} - I\|_2 = \max\{|\lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) - 1|,\ |\lambda_1(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) - 1|\}.$$

Thus,

$$|\lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) - 1| \le \epsilon. \tag{3.14}$$

This yields

$$\lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) \ge 1 - \epsilon.$$

(In fact, either $\lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) \ge 1$, or otherwise, (3.14) means $1 - \lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) \le \epsilon$.) Accordingly,

$$\lambda_1(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) = \frac{1}{\lambda_N(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T})} \le \frac{1}{1-\epsilon}. \tag{3.15}$$

On the other hand,

$$\lambda_1(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}) = \|\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T}\|_2 \le 1 + \|\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T} - I\|_2 \le 1 + \epsilon.$$

Accordingly,

$$\lambda_N(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) = \frac{1}{\lambda_1(\mathbf{L}^{-1}\tilde{A}\mathbf{L}^{-T})} \geq \frac{1}{1+\epsilon}. \tag{3.16}$$

Combining (3.15) and (3.16) yields (3.12) and (3.13).

As a consequence,

$$\begin{aligned}\|\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T} - I\|_2 &= \max\{|\lambda_1(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) - 1|,\ |\lambda_N(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{L}}^{-T}) - 1|\} \\ &\leq \max\{\frac{1}{1-\epsilon} - 1,\ 1 - \frac{1}{1+\epsilon}\} \\ &\leq \frac{\epsilon}{1-\epsilon}.\end{aligned}$$

□

Therefore, in the multilevel case, ϵ plays a role similar to τ in the 1-level case in Theorem 2.5. We can then similarly understand the effectiveness of the preconditioner when aggressive compression with relatively large τ is used, like in the end of Section 2.3 (Table 2.1 and Figure 2.1).

3.3. Modified multilevel SIF preconditioner. In the multilevel scheme in Section 3.1, the resulting Q_i factors have sizes equal to the sizes of the corresponding index sets s_i. Such sizes increase when we advance to upper level nodes. There are different ways to save some computational costs. One way is to instead scale and compress the off-diagonal blocks $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ in a hierarchical scheme. $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ is essentially called an HSS block row (a block row without the diagonal block) [28]. Some nested basis matrices will be produced. This can avoid the scaling and compression of large dense off-diagonal blocks since $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ may include subblocks that have already been compressed in previous steps. However, it makes the scaling on the right more delicate. Thus, we use a modified version. That is, we first apply the left scaling to $A|_{s_i\times(\mathcal{I}\setminus s_i)}$, perform compression, and compute ULV factorization. Since the entire block row $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ (instead of just $A|_{s_i\times s_j}$ for $j = \mathrm{sib}(i)$) is scaled on the left, the symmetry indicates that the right-side scaling is implicitly built in. This will become clear later in (3.21) and (3.22).

We would first like to mention a notation issue. In the algorithm, when $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ of A is compressed (by a rank-revealing QR factorization) as $A|_{s_i\times(\mathcal{I}\setminus s_i)} \approx U_iT_i$, the matrix T_i can reuse the storage for $A|_{s_i\times(\mathcal{I}\setminus s_i)}$. For convenience, we write this compression step as $A|_{s_i\times(\mathcal{I}\setminus s_i)} \approx U_i\hat{A}^{(i)}|_{\hat{s}_i\times(\mathcal{I}\setminus s_i)}$, where $\hat{A}^{(i)}|_{\hat{s}_i\times(\mathcal{I}\setminus s_i)}$ indicates the storage of T_i in A with the row index set $\hat{s}_i$ and column index set $\mathcal{I}\setminus s_i$. Such a notation scheme is used in [27] and makes it convenient to identify the subblocks that have been compressed in earlier steps. The symbols $\hat{A}^{(i)}$ are only for the purpose of organizing the compression steps, and do not imply any extra storage.

During the factorization, we try to take advantage of previous compression information as much as possible by keeping track of compressed subblocks. This is done with the aid of a *visited set* in [27].

DEFINITION 3.3. *For a node i of a postordered binary tree $\mathcal{T}$, a visited set $\mathcal{V}_i$ is defined to be*

$$\mathcal{V}_i = \{j:\ \ j < i \ \textit{and}\ \mathrm{sib}(j)\ \textit{is an ancestor of}\ i\}.$$

The visited set essentially corresponds to the roots of all the subtrees with indices smaller than i. Those nodes have been traversed, and correspond to blocks that are

highly compressed. A detailed explanation of this and a pictorial illustration of $\mathcal{V}_i$ can be found in [27]. In fact, $\mathcal{V}_i$ can be made more general to include the roots of all the subtrees previous visited, not necessarily those smaller than i. We follow the original definition in [27] just for the convenience of presentation.

The modified multilevel SIF works as follows. For a leaf node i of $\mathcal{T}$, compute the Cholesky factorization as in (3.1) and set $\tilde{\mathbf{L}}_i \equiv L_i$. We then scale and compress the off-diagonal block $A|_{s_i\times(\mathcal{I}\setminus s_i)}$. Note that, due to the symmetry, some subblocks of $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ may have been compressed in the previous steps. Assume

$$\mathcal{V}_i = \{k_1, k_2, \ldots, k_\alpha\colon \quad k_1 < \cdots < k_\alpha\}, \tag{3.17}$$

where α is the number of elements in $\mathcal{V}_i$ and depends on i. Form

$$\Omega_i = \left(\begin{array}{cc} \left(\begin{array}{ccc} (\hat{A}^{(k_1)}|_{\hat{s}_{k_1}\times s_i})^T & \cdots & (\hat{A}^{(k_\alpha)}|_{\hat{s}_{k_\alpha}\times s_i})^T \end{array} \right) & A|_{s_i\times s_i^+} \end{array} \right),$$

where $s_i^+ \subset \mathcal{I}$ contains all indices larger than those in s_i, and $(\hat{A}^{(k_1)}|_{\hat{s}_{k_1}\times s_i})^T, \ldots, (\hat{A}^{(k_\alpha)}|_{\hat{s}_{k_\alpha}\times s_i})^T$ are results from earlier compression steps. (More specifically, each block $\hat{A}^{(k)}|_{s_k\times s_i}$ is previously compressed as $U_k\hat{A}^{(k)}|_{\hat{s}_k\times s_i}$ during the compression step associated with node $k \in \mathcal{V}_i$. Due to the symmetry, $(\hat{A}^{(k)}|_{s_k\times s_i})^T$ should be compressed at step i, and the row basis matrix U_k can be ignored. Thus, the factor $(\hat{A}^{(k)}|_{\hat{s}_k\times s_i})^T$ is collected into Ω_i. This can be understood like in Figure 3.3 in [27]. We skip the technical details.) Scale Ω_i as $\Theta_i \equiv L_i^{-1}\Omega_i$. Then compute a rank-revealing QR (RRQR) factorization

$$\Theta_i \approx U_i \left(\begin{array}{cc} \left(\begin{array}{ccc} \hat{A}^{(i)}|_{\hat{s}_i\times\hat{s}_{k_1}} & \cdots & \hat{A}^{(i)}|_{\hat{s}_i\times\hat{s}_{k_\alpha}} \end{array} \right) & \hat{A}^{(i)}|_{\hat{s}_i\times s_i^+} \end{array} \right).$$

where $\hat{s}_i$ is the row index set used to identify the second factor. Also, find an orthogonal matrix Q_i such that $Q_i^T U_i = \begin{pmatrix} 0 \\ I \end{pmatrix}$. Then set $\tilde{\mathbf{L}}_i = L_i Q_i$.

The process above indicates that $\tilde{\mathbf{L}}_i \begin{pmatrix} 0 \\ I \end{pmatrix} = L_i U_i$ and it is an approximate column basis matrix for Ω_i. Similarly, for $j = \mathrm{sib}(i)$, we also obtain an approximate column basis matrix $\tilde{\mathbf{L}}_j \begin{pmatrix} 0 \\ I \end{pmatrix}$ for Ω_j. These yield

$$A|_{s_i\times s_j} \approx \left(\tilde{\mathbf{L}}_i \begin{pmatrix} 0 \\ I \end{pmatrix}\right) B_i \left(\begin{pmatrix} 0 & I \end{pmatrix} \tilde{\mathbf{L}}_j^T \right) = \tilde{\mathbf{L}}_i \,\mathrm{diag}(0, B_i) \tilde{\mathbf{L}}_j^T, \tag{3.18}$$

where

$$B_i = \hat{A}^{(\iota)}|_{\hat{s}_i\times\hat{s}_j} \quad \text{with } \iota = \max\{i, j\}. \tag{3.19}$$

For a nonleaf node i with children c_1 and c_2, the induction process below shows that $\tilde{\mathbf{L}}_{c_1} \begin{pmatrix} 0 \\ I \end{pmatrix}$ and $\tilde{\mathbf{L}}_{c_2} \begin{pmatrix} 0 \\ I \end{pmatrix}$ are approximate column basis matrices for $A|_{s_{c_1}\times(\mathcal{I}\setminus s_{c_1})}$ and $A|_{s_{c_2}\times(\mathcal{I}\setminus s_{c_2})}$, respectively, and

$$A|_{s_{c_1}\times s_{c_1}} \approx \tilde{\mathbf{L}}_{c_1}\tilde{\mathbf{L}}_{c_1}^T, \quad A|_{s_{c_2}\times s_{c_2}} \approx \tilde{\mathbf{L}}_{c_2}\tilde{\mathbf{L}}_{c_2}^T, \tag{3.20}$$

$$A|_{s_{c_1}\times s_{c_2}} \approx \left(\tilde{\mathbf{L}}_{c_1} \begin{pmatrix} 0 \\ I \end{pmatrix}\right) B_{c_1} \left(\begin{pmatrix} 0 & I \end{pmatrix} \tilde{\mathbf{L}}_{c_2}^T \right) = \tilde{\mathbf{L}}_{c_1} \,\mathrm{diag}(0, B_{c_1}) \tilde{\mathbf{L}}_{c_2}^T, \quad \text{with} \tag{3.21}$$

$$B_{c_1} = \hat{A}^{(c_2)}|_{\hat{s}_{c_1}\times\hat{s}_{c_2}}.$$

It is thus clear that $A|_{s_i\times s_i}$ is approximated as

$$A|_{s_i\times s_i} \approx \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} & \\ & \tilde{\mathbf{L}}_{c_2} \end{pmatrix} \begin{pmatrix} I & \mathrm{diag}(0, B_{c_1}) \\ \mathrm{diag}(0, B_{c_1}^T) & I \end{pmatrix} \begin{pmatrix} \tilde{\mathbf{L}}_{c_1}^T & \\ & \tilde{\mathbf{L}}_{c_2}^T \end{pmatrix}. \tag{3.22}$$

(3.21) and (3.22) precisely explain how the right-side scaling is implicitly done in the process.

Again, with a permutation matrix Π_i, the matrix in the middle on the right-hand side of (3.22) can be assembled as $\Pi_i \begin{pmatrix} I & \\ & D_i \end{pmatrix}$, where $D_i = \begin{pmatrix} I & B_{c_1} \\ B_{c_1}^T & I \end{pmatrix}$. Compute a Cholesky factorization $D_i = L_i L_i^T$, then

$$A|_{s_i\times s_i} \approx \hat{\mathbf{L}}_i \hat{\mathbf{L}}_i^T, \quad \text{with} \quad \hat{\mathbf{L}}_i = \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} & \\ & \tilde{\mathbf{L}}_{c_2} \end{pmatrix} \Pi_i \begin{pmatrix} I & \\ & L_i \end{pmatrix}. \tag{3.23}$$

At this point, we can use $\hat{\mathbf{L}}_i^{-1}$ to scale the off-diagonal block row $A|_{s_i\times(\mathcal{I}\setminus s_i)}$ associated with i and then compress. Again by induction,

$$A|_{s_i\times(\mathcal{I}\setminus s_i)} = \begin{pmatrix} A|_{s_{c_1}\times(\mathcal{I}\setminus s_i)} \\ A|_{s_{c_2}\times(\mathcal{I}\setminus s_i)} \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} \begin{pmatrix} 0 \\ I \end{pmatrix} \Omega_{i,1} \\ \tilde{\mathbf{L}}_{c_2} \begin{pmatrix} 0 \\ I \end{pmatrix} \Omega_{i,2} \end{pmatrix}. \tag{3.24}$$

where

$$\Omega_i = \begin{pmatrix} \Omega_{i,1} \\ \Omega_{i,2} \end{pmatrix} \equiv \begin{pmatrix} \begin{pmatrix} (\hat{A}^{(c_1)}|_{\hat{s}_{k_1}\times\hat{s}_{c_1}})^T & \cdots & (\hat{A}^{(c_1)}|_{\hat{s}_{k_\alpha}\times\hat{s}_{c_1}})^T & \hat{A}^{(c_1)}|_{\hat{s}_{c_1}\times s_i^+} \\ (\hat{A}^{(c_2)}|_{\hat{s}_{k_1}\times\hat{s}_{c_2}})^T & \cdots & (\hat{A}^{(c_2)}|_{\hat{s}_{k_\alpha}\times\hat{s}_{c_2}})^T & \hat{A}^{(c_2)}|_{\hat{s}_{c_2}\times s_i^+} \end{pmatrix} \end{pmatrix},$$

and the notation in (3.17) is assumed. Then

$$\begin{aligned} \hat{\mathbf{L}}_i^{-1} A|_{s_i\times(\mathcal{I}\setminus s_i)} &= \hat{\mathbf{L}}_i^{-1} \,\mathrm{diag}\left(\tilde{\mathbf{L}}_{c_1} \begin{pmatrix} 0 \\ I \end{pmatrix}, \tilde{\mathbf{L}}_{c_2} \begin{pmatrix} 0 \\ I \end{pmatrix} \right) \Omega_i \\ &= \begin{pmatrix} I & \\ & L_i^{-1} \end{pmatrix} \Pi_i^T \,\mathrm{diag}\left(\begin{pmatrix} 0 \\ I \end{pmatrix}, \begin{pmatrix} 0 \\ I \end{pmatrix} \right) \Omega_i \\ &= \begin{pmatrix} I & \\ & L_i^{-1} \end{pmatrix} \begin{pmatrix} 0 \\ \mathrm{diag}(I, I) \end{pmatrix} \Omega_i = \begin{pmatrix} 0 \\ L_i^{-1} \end{pmatrix} \Omega_i. \end{aligned}$$

Thus, we just need to compress

$$\Theta_i \equiv L_i^{-1} \Omega_i.$$

We can then perform an RRQR factorization

$$\Theta_i \approx U_i \begin{pmatrix} \hat{A}^{(i)}|_{\hat{s}_i\times\hat{s}_{k_1}} & \cdots & \hat{A}^{(i)}|_{\hat{s}_i\times\hat{s}_{k_\alpha}} & \hat{A}^{(i)}|_{\hat{s}_i\times s_i^+} \end{pmatrix}.$$

Again, find an orthogonal matrix Q_i such that $Q_i^T U_i = \begin{pmatrix} 0 \\ I \end{pmatrix}$. Let

$$\tilde{\mathbf{L}}_i \equiv \hat{\mathbf{L}}_i \begin{pmatrix} I & \\ & Q_i \end{pmatrix} = \begin{pmatrix} \tilde{\mathbf{L}}_{c_1} & \\ & \tilde{\mathbf{L}}_{c_2} \end{pmatrix} \Pi_i \begin{pmatrix} I & \\ & L_i Q_i \end{pmatrix}. \tag{3.25}$$

Then we can verify that $\tilde{\mathbf{L}}_i \begin{pmatrix} 0 \\ I \end{pmatrix} = \hat{\mathbf{L}}_i \begin{pmatrix} 0 \\ U_i \end{pmatrix}$ and

$$A|_{s_i \times (\mathcal{I} \setminus s_i)} \approx \tilde{\mathbf{L}}_i \begin{pmatrix} 0 \\ I \end{pmatrix} \begin{pmatrix} \hat{A}^{(i)}|_{\hat{s}_i \times \hat{s}_{k_1}} & \cdots & \hat{A}^{(i)}|_{\hat{s}_i \times \hat{s}_{k_\alpha}} & \hat{A}^{(i)}|_{\hat{s}_i \times s_i^+} \end{pmatrix}.$$

This confirms the induction about the column basis matrices of the forms $\tilde{\mathbf{L}}_i \begin{pmatrix} 0 \\ I \end{pmatrix}$. (See, e.g., (3.21) and (3.24).) Moreover, (3.23) can be rewritten as

$$A|_{s_i \times s_i} \approx \tilde{\mathbf{L}}_i \tilde{\mathbf{L}}_i^T, \quad \text{with } \tilde{\mathbf{L}}_i \text{ in (3.25).}$$

This confirms the induction process in (3.20). In addition, we can extract B_i just like in (3.19) so as to obtain an approximation like in (3.18).

The factorization process then continues until root($\mathcal{T}$) is reached, and then we have an approximate factorization like in (3.7). A major difference is that $\tilde{\mathbf{L}}$ is now given by the recursion (3.25), where Q_i is a small matrix with size $2r$ (when r is the numerical rank in the compression).

For a general dense matrix A, this scheme costs $O(rN^2)$ to compute the approximate factorization. The solution procedure can be designed similarly to that at the end of Section 3.1. With $r = O(1)$ in preconditioning, the cost to apply the resulting preconditioner is $O(N)$, and the storage is $O(N)$. This scheme shares some similarities as the one in Section 3.1, but is much more sophisticated. We thus skip the other analysis. We expect slightly reduced effectiveness, but enhanced robustness, as confirmed by numerical tests in the next section.

REMARK 3.1. In this scheme and also the one in Section 3.1, we may replace the truncated SVDs or RRQR factorizations by other appropriate methods to improve the efficiency. In particular, if A is a sparse matrix or a dense one that admits fast matrix-vector multiplications, randomized construction schemes similar to those in [18, 20, 30] can be used so that the cost to construct the preconditioner can be reduced to roughly $O(N)$. Relevant details will appear in [31], as this work focuses on general SPD matrices. Here for the compression of dense blocks, either truncated SVD or a randomized method is used, depending on the block sizes.

4. Numerical experiments. In this section, we demonstrate the performance of the new preconditioners, which are used in the preconditioned conjugate gradient method (PCG) to solve some ill-conditioned linear systems. We compare the effectiveness and robustness of the following preconditioners:

- bdiag: the block diagonal preconditioner;
- HSS: HSS approximation (in ULV factors [28]) based on direct off-diagonal compression;
- NEW0: the multilevel SIF preconditioner in Section 3.1;
- NEW1: the modified multilevel SIF preconditioner in Section 3.3.

For convenience, the following notation is used in the discussions:

- r: the numerical rank bound used in off-diagonal compression;
- A_{prec}: the preconditioned matrix (e.g., $\tilde{\mathbf{L}}^{-1} A \tilde{\mathbf{L}}^{-T}$ when a preconditioner $\tilde{\mathbf{L}}\tilde{\mathbf{L}}^T$ is used; for some cases where HSS loses positive definiteness, factors from nonsymmetric HSS factorizations are used);
- $\gamma = \frac{\|Ax-b\|_2}{\|b\|_2}$: the relative residual, where b is generated with the vector of all ones as the exact solution;

- $\mathcal{N}_{\rm it}$: the number of iterations to reach a given accuracy.

Several ill-conditioned matrices are tested. Since our studies are concerned with general SPD matrices, all the test matrices are treated as dense ones, and we do not take advantage of possible special structures within some examples.

In all the tests, we keep the numerical rank bound r very small, so that the costs of applying our preconditioners (by substitution) are nearly linear in N. Such costs are almost negligible as compared with dense matrix-vector multiplication costs. Thus, it is important to reduce the number of iterations in PCG. For consistency, the diagonal block sizes of bdiag and also the finest level diagonal block sizes of HSS, NEW0, and NEW1 are all set to be r.

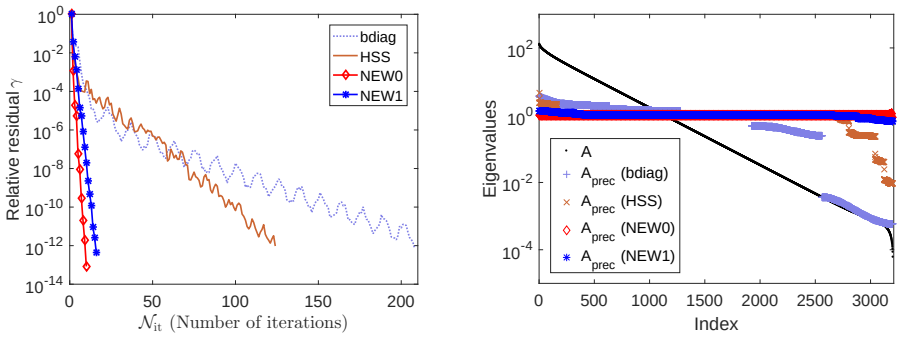
EXAMPLE 1. Consider a matrix A with entries

$$A_{ij} = \frac{(ij)^{1/4}\pi}{16 + (i-j)^2}.$$

The matrix can be known to be SPD and ill conditioned based on a result in [24].

We test different matrix sizes N, but fix $r = 5$ when generating the preconditioners. PCG with the four preconditioners is used to reach the relative accuracy $\gamma \le 10^{-12}$. The tests are in Matlab and run on an Intel Xeon-E5 processor with 64GB memory. Table 4.1 shows the results. We report the condition numbers before and after preconditioning, the numbers of iterations $\mathcal{N}_{\rm it}$, and the relative residuals γ. For all the tests, NEW0 and NEW1 remain SPD, although r is very small. However, HSS fails to preserve the positive definiteness. Thus, a nonsymmetric HSS factorization is used in the solution.

NEW0 and NEW1 significantly reduce the condition numbers and lead to much faster convergence than bdiag and HSS. HSS gives reasonable reduction in the condition numbers, but the numbers of iterations are higher. The performance can also be observed from Figure 4.1(a), which shows the actual convergence behaviors for $N = 3200$. The fast convergence with the new preconditioners can also be confirmed by the eigenvalue distribution of $A_{\rm prec}$. See Figure 4.1(b), where the eigenvalues of $A_{\rm prec}$ with NEW0 or NEW1 closely cluster around 1. It is not the case for bdiag or HSS.



(a) Convergence of PCG (b) $\lambda(A)$ and $\lambda(A_{\rm prec})$

FIG. 4.1. *Example 1. Convergence of PCG with the preconditioners for the matrix with $N = 3200$, and the eigenvalues before and after preconditioning.*

In addition, NEW0 fully respects the scaling-and-compression strategy and is the most effective. NEW1 needs less memory than NEW0 ($O(N)$ vs. $O(N \log N)$). See

TABLE 4.1

Example 1. Performance of the preconditioners in PCG. (The condition numbers are not evaluated for $N = 25,600$ and $51,200$ since it is too slow to form A_{prec}.)

N		1600	3200	6400	12,800	25,600	51,200
$\kappa(A)$		1.48e6	2.14e6	3.06e6	4.38e6		
$\kappa(A_{\text{prec}})$	bdiag	6.18e3	6.20e3	6.21e3	6.22e3		
	HSS	4.89e2	4.89e2	4.81e2	4.76e2		
	NEW0	1.30	1.31	1.31	1.32		
	NEW1	2.05	2.05	2.05	2.05		
$\mathcal{N}_{\text{it}}$	bdiag	213	207	209	198	197	185
	HSS	119	123	125	125	124	123
	NEW0	9	9	8	8	8	8
	NEW1	15	15	15	15	15	15
γ	bdiag	8.10e−13	7.56e−13	6.60e−13	9.76e−13	8.83e−13	8.90e−13
	HSS	8.78e−13	9.54e−13	7.83e−13	4.82e−13	9.69e−13	8.55e−13
	NEW0	6.89e−14	8.34e−14	8.85e−13	6.76e−13	5.43e−13	4.40e−13
	NEW1	5.18e−13	4.61e−13	3.82e−13	3.36e−13	3.46e−13	8.68e−13
Positive definiteness	bdiag	✓					
	HSS	✗					
	NEW0	✓					
	NEW1	✓					

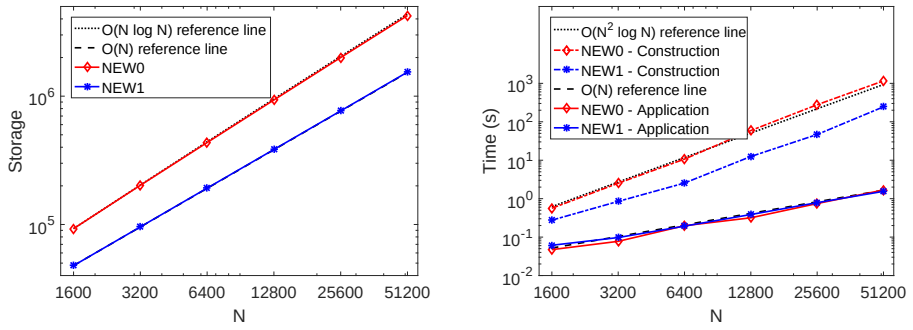
Figure 4.2(a). The storage of HSS is nearly the same as that of NEW1 and is not shown.

We also report the runtime to construct the two new preconditioners and to apply them (to one vector) in Figure 4.2(b). The construction of HSS is about 3 times faster than that of NEW1, and the application of HSS has runtime close to that of NEW1. However, the iteration time of PCG with HSS is much longer due to the larger number of iterations. The iteration time of PCG with bdiag is also quite larger than with the new preconditioners. (The construction and application of bdiag are more efficient in Matlab due to the simple block diagonal structure, while the new preconditioners involve more sophisticated structured matrix operations such as the application of lots of local Householder matrices. Thus, the advantage of the new preconditioners (as compared with bdiag) in iteration time is not as large as the advantage in the numbers of iterations. In practice, the reduction of the number of iterations is essential, since each dense matrix-vector multiplication costs $O(N^2)$, while each application of these preconditioners costs only about $O(N)$.)

EXAMPLE 2. Then consider some interpolation matrices A for the following radial basis functions (RBFs) of t:

$$f_1(t,\mu) = e^{-\mu^2 t^2}, \quad f_2(t,\mu) = \frac{1}{\sqrt{\mu^2 t^2 + 1}}, \quad f_3(t,\mu) = \operatorname{sech} \mu t,$$

where μ is the shape parameter. For convenience, choose t to be between 0 and $N-1$. It is well known that the RBF matrices are very ill conditioned for small shape parameters. For some cases, the condition numbers are nearly exponential in $\frac{1}{\mu}$ or $\frac{1}{\mu^2}$ (see, e.g., [4]).



(a) Storage (number of nonzeros) (b) Construction and application time

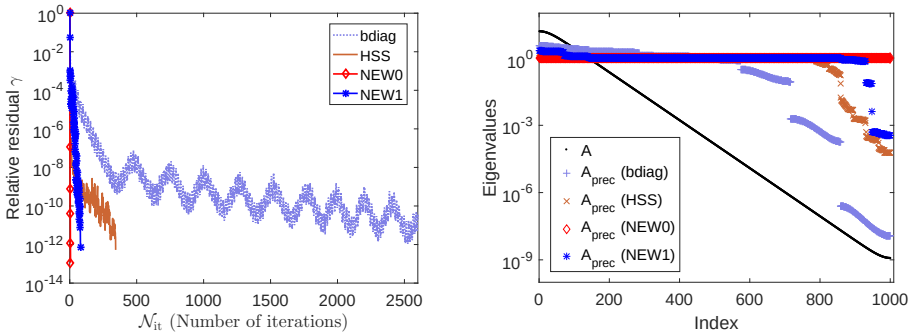
FIG. 4.2. *Example 1. Storage for NEW0 and NEW1 and the runtime to construct them and to apply them to one vector.*

We test several matrices with different parameters μ. The matrix sizes are set to be $N = 1000$. (Similar results are observed for other sizes.) We fix $r = 7$ when generating the preconditioners. See Table 4.2 for the results. PCG with NEW0 or NEW1 converges quickly, and NEW0 is the most effective. On the other hand, NEW1 remains positive definite for all the test cases, while NEW0 loses positive definiteness for one case (indicated by "almost" in Table 4.2). In that case, two small intermediate reduced matrices in the construction of the preconditioner become indefinite. A very small diagonal shift is added to make these reduced matrices positive definite. In addition, HSS again becomes indefinite for all the tests.

TABLE 4.2
Example 2. Performance of the preconditioners in PCG.

RBF		$e^{-\epsilon^2 t^2}$		$\operatorname{sech} \epsilon t$		$\frac{1}{\sqrt{\epsilon^2 x^2+1}}$	
ϵ		0.4	0.34	0.3	0.2	0.3	0.2
$\kappa(A)$		2.49e6	9.30e8	3.48e6	1.30e10	2.52e5	5.36e7
$\kappa(A_{\text{prec}})$	bdiag	8.83e4	2.75e7	8.02e4	3.18e8	6.51e3	1.20e6
	HSS	1.60e1	5.42e4	2.94e1	2.79e4	6.23e1	7.50e4
	NEW0	1.00	1.00	1.00	1.15	1.02	4.17e1
	NEW1	1.88	2.33e3	1.27	5.77e3	1.51	1.48e2
$\mathcal{N}_{\text{it}}$	bdiag	456	1973	299	2738	146	587
	HSS	53	332	39	342	38	1028
	NEW0	3	3	2	5	4	13
	NEW1	13	105	8	79	11	78
γ	bdiag	6.67e−13	9.99e−13	9.43e−13	9.36e−13	4.57e−13	7.83e−13
	HSS	8.63e−13	8.03e−13	7.42e−13	5.27e−13	2.70e−13	8.70e−13
	NEW0	2.49e−16	5.46e−14	2.93e−13	1.14e−13	2.77e−13	2.41e−14
	NEW1	3.82e−13	5.15e−13	8.05e−13	6.98e−13	4.69e−13	7.02e−13
Positive definiteness	bdiag			✓			
	HSS			✗			
	NEW0	✓	✓	✓	✓	✓	Almost
	NEW1			✓			

Figure 4.3 also shows the actual convergence and eigenvalue distributions for one matrix. The new preconditioners yield much faster convergence and better eigenvalue clustering.



(a) Convergence of PCG (b) $\lambda(A)$ and $\lambda(A_{\rm prec})$

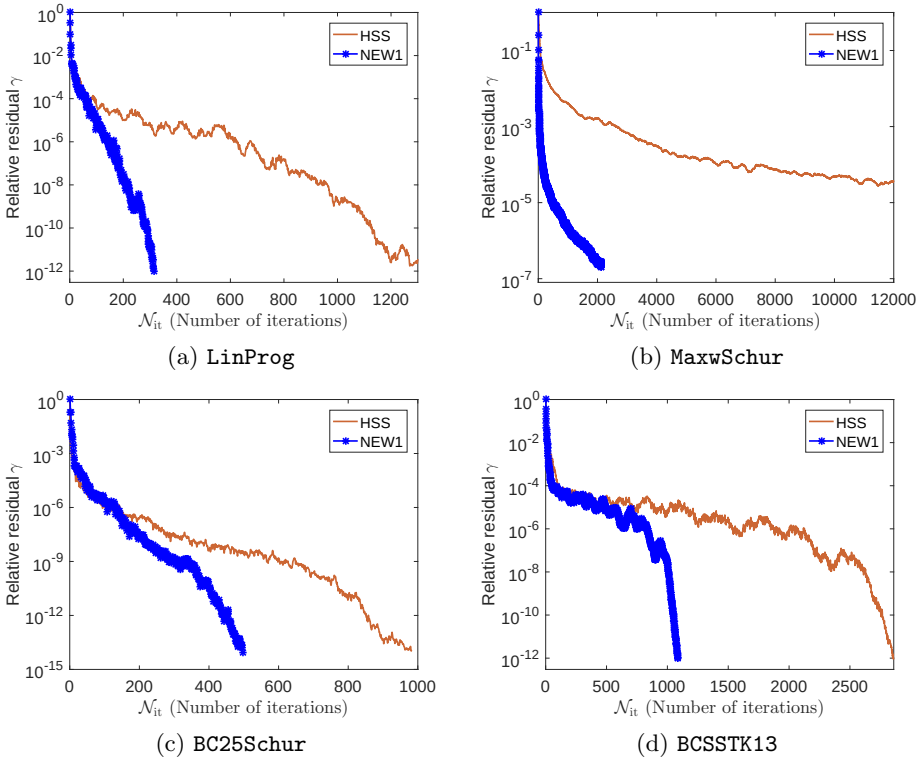
FIG. 4.3. *Example 2. Convergence of PCG with the preconditioners for the matrix corresponding to* $\operatorname{sech}\mu t$, $\mu = 0.2$ *in Table 4.2, and the eigenvalues before and after preconditioning.*

EXAMPLE 3. We further test some highly ill-conditioned matrices from different applications, for which NEW1 still demonstrates superior robustness and also leads to much faster convergence than HSS.

- `LinProg` ($N = 2301,\ \kappa(A) = 2.09e11$)
  A matrix of the form $C\Lambda C^T$ as used in some linear programming problems, where C is a rectangular matrix nemspmm2 from the linear programming test matrix set Meszaros in [7], and Λ is a diagonal matrix with diagonal entries even spaced between 10^{-5} and 1. We set $r = 9$.
- `MaxwSchur` ($N = 3947,\ \kappa(A) = 4.78e9$)
  A dense normal matrix in [29] constructed based on a Schur complement in the direct factorization of a discretized 3D Maxwell equation. We set $r = 20$.
- `BC25Schur` ($N = 1826,\ \kappa(A) = 3.28e11$)
  A dense intermediate Schur complement in the multifrontal factorization [8] of the BCS structural engineering matrix BCSSTK25 from the Harwell-Boeing Collection [21]. We set $r = 7$.
- `BCSSTK13` ($N = 2003,\ \kappa(A) = 1.10e10$)
  The matrix BCSSTK13 (directly treated as dense) from the Harwell-Boeing Collection [21]. We set $r = 10$.

For each matrix, NEW1 remains positive definite, while HSS loses the positive definiteness. The convergence of PCG with NEW1 is also faster, as shown in Figure 4.4. In Table 4.3, we also show the convergence results for one matrix with different compression rank bounds r. A larger rank bound r leads to better convergence, but the preconditioner construction and application cost more.

5. Conclusions. We have presented the SIF framework and its analysis for preconditioning general SPD matrices. The SIF techniques include the scaling-and-compression strategy and the ULV factorization. A prototype preconditioner is used to illustrate the basic ideas and fundamental analysis. Generalizations to practical multilevel schemes are then made and also analyzed. Systematic analysis for the robustness, accuracy control, and effectiveness of the preconditioners is given.



(a) `LinProg` (b) `MaxwSchur`

(c) `BC25Schur` (d) `BCSSTK13`

FIG. 4.4. *Example 3. Convergence of PCG with the preconditioners NEW1 and HSS.*

TABLE 4.3

Example 3. Performance of NEW1 and HSS as the preconditioners in PCG for the matrix `LinProg`, where different compression rank bounds r are used to construct the preconditioners.

r		3	6	9	12
$\mathcal{N}_{\rm it}$	HSS	2165	1499	1331	1147
	NEW1	598	378	313	282
γ	HSS	$7.39e-13$	$8.99e-13$	$7.91e-13$	$9.56e-13$
	NEW1	$9.58e-13$	$7.74e-13$	$9.80e-13$	$9.03e-13$

Our discussions mainly focus on general dense SPD matrices, though the techniques are also applicable to sparse ones and some special dense ones (see Remark 3.1). The techniques in this paper can also be applied to dense Schur complements in sparse preconditioning. Such work will appear in [31]. We also expect to produce efficient black-box implementations of the preconditioners. Similar techniques for the preconditioning of more general matrices such as indefinite ones are under development.

Acknowledgements. We thank Lieven Vandenberghe for suggesting a linear programming test example, and thank the two anonymous referees for the valuable suggestions. The research of Jianlin Xia was supported in part by NSF CAREER Award DMS-1255416.

REFERENCES

[1] M. A. Ajiz and A. Jennings, *A robust incomplete Choleski-conjugate gradient algorithm*, Internat. J. Numer. Methods Engrg., 20 (1984), pp. 949–966.

[2] M. Benzi, J. K. Cullum, and M. Tuma, *Robust approximate inverse preconditioning for the conjugate gradient method*, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[3] M. Benzi and M. Tuma, *A robust incomplete factorization preconditioner for positive definite matrices*, Numer. Linear Algebra Appl., 10 (2003), pp. 385–400.

[4] J. P. Boyd and K. W. Gildersleeve, *Numerical experiments on the condition number of the interpolation matrices for radial basis functions*, Appl. Numer. Math., 61 (2011), pp. 443–459.

[5] S. Chandrasekaran, M. Gu, and T. Pals, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622.

[6] P. Coulier, B. Quaife, and E. Darve, *An efficient preconditioner for the fast simulation of a 2D Stokes flow in porous media*, Internat. J. Numer. Methods Engrg., to appear, (2017), DOI: 10.1002/nme.5626.

[7] T. Davis and Y. Hu, *The SuiteSparse Matrix Collection*, http://www.cise.ufl.edu/research/sparse/matrices.

[8] I. S. Duff and J. K. Reid, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM Tans. Math. Software, 9 (1983), pp. 302–325.

[9] Z. Drmac, M. Omladic, and K. Veselic, *On the perturbation of the Cholesky factorization*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1319–1332.

[10] B. Engquist and L. Ying, *Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation*, Pure Appl. Math., LXIV (2011), pp. 0697–0735.

[11] L. Grasedyck, S. Le Borne, and R. Kriemann, *Parallel blackbox H-LU preconditioning for elliptic boundary value problems*, Comput. Visual. Sci. 11 (2008), pp. 273–291.

[12] M. Gu, X. S. Li, and P. Vassilevski, *Direction-preserving and Schur-monotonic semiseparable approximations of symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2650–2664.

[13] I. E. Kaporin, *High quality preconditioning of a general symmetric positive definite matrix based on its $U^TU + U^TR + R^TU$-decomposition*, Numer. Linear Algebra Appl., 5 (1998), pp. 483–509.

[14] S. Le Borne, L. Grasedyck, and R. Kriemann, *Domain-decomposition based $\mathcal{H}$-LU preconditioners*, in Domain Decomposition Methods in Science and Engineering XVI, O. B. Widlund and D. E. Keyes, eds., Lect. Notes Comput. Sci. Eng. 55, Springer, Berlin, 2006, pp. 667–674.

[15] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013), pp. A2069–A2095.

[16] R. Li and Y. Saad, *Low-rank correction methods for algebraic domain decomposition preconditioners*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 807–828.

[17] R. Li, Y. Xi, and Y. Saad, *Schur complement based domain decomposition preconditioners with low-rank corrections*, Numer. Linear Algebra Appl., 23(4) (2016), pp. 706-729.

[18] L. Lin, J. Lu, and L. Ying, *Fast construction of hierarchical matrix representation from matrix-vector multiplication*, J. Comput. Phys., 230 (2011), pp. 4071–4087.

[19] T. A. Manteuffel, *An incomplete factorization technique for positive definite linear systems*, Math. Comp., 34 (1980), pp. 473–497.

[20] P. G. Martinsson, *A fast randomized algorithm for computing a hierarchically semiseparable representation of a matrix*, SIAM. J. Matrix Anal. Appl., 32 (2011), pp. 1251–1274.

[21] *Matrix Market*, http://math.nist.gov/MatrixMarket.

[22] J. A. Meijerink and H. A. van der Vorst, *An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix*, Math. Comp., 31 (1977), pp. 148–162.

[23] A. Napov, *Conditioning analysis of incomplete Cholesky factorizations with orthogonal dropping*, SIAM J. Matrix Anal. Appl., 34 (2013), pp. 1148–1173.

[24] C. V. M. van der Mee and S. Seatzu, *A method for generating infinite positive self-adjoint test matrices and Riesz bases*, SIAM J. Matrix Anal. Appl., 26 (2005), pp. 1132–1149.

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

[26] J. Xia, *Some new developments on efficient structured solvers and preconditioners*, Presentation in 2011 SIAM CSE Conference, February 28, 2011.

[27] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388–410.

[28] J. Xia, S. Chandrasekaran, M. Gu, and X. S. Li, *Fast algorithms for hierarchically semiseparable matrices*, Numer. Linear Algebra Appl., 17 (2010), pp. 953–976.
[29] J. Xia and M. Gu, *Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl., 31 (2010), pp. 2899–2920.
[30] J. Xia, Y. Xi, and M. Gu, *A superfast structured solver for Toeplitz linear systems via randomized sampling*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 837–858.
[31] J. Xia, Z. Xin, S. Cauley, and V. Balakrishnan, *Effective and robust sparse preconditioning via structured incomplete factorization*, preprint, 2017.