ROBUST AND EFFECTIVE ESIF PRECONDITIONING FOR GENERAL DENSE SPD MATRICES*

JIANLIN XIA†

Abstract. We propose an unconditionally robust and highly effective preconditioner for general dense symmetric positive definite (SPD) matrices based on structured incomplete factorization (SIF), called enhanced SIF (eSIF) preconditioner. The original SIF strategy proposed recently derives a structured preconditioner by applying block diagonal preprocessing to the matrix and then compressing appropriate scaled off-diagonal blocks. Here, we use an enhanced scaling-and-compression strategy to design the new eSIF preconditioner. Some subtle modifications are made, such as the use of two-sided block triangular preprocessing. A practical multilevel eSIF scheme is then designed. We give rigorous analysis for both the enhanced scaling-and-compression strategy and the multilevel eSIF preconditioner. The new eSIF framework has some significant advantages and overcomes some major limitations of the SIF strategy. (i) With the same tolerance for compressing the off-diagonal blocks, the eSIF preconditioner can approximate the original matrix to a much higher accuracy. (ii) The new preconditioner leads to much more significant reductions of condition numbers due to an accelerated magnification effect for the decay in the singular values of the scaled off-diagonal blocks. (iii) With the new preconditioner, the eigenvalues of the preconditioned matrix are much better clustered around 1. (iv) The multilevel eSIF preconditioner is further unconditionally robust or is guaranteed to be positive definite without the need of extra stabilization, while the multilevel SIF preconditioner has a strict requirement in order to preserve positive definiteness. Comprehensive numerical tests are used to show the advantages of the eSIF preconditioner in accelerating the convergence of iterative solutions.

Key words. eSIF preconditioning, SPD matrix, enhanced scaling-and-compression strategy, effectiveness, unconditional robustness, multilevel scheme

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

1. Introduction.

In this paper, we consider the design of an effective and robust preconditioning strategy for general dense symmetric positive definite (SPD) matrices. An effective preconditioner can significantly improve the convergence of iterative solutions. For an SPD matrix A, it is also desirable for the preconditioner to be robust or to preserve the positive definiteness. A commonly used strategy to design robust preconditioners is to apply modifications or incomplete/approximate Cholesky factorizations to A together with some robustness or stability enhancement strategies (see, e.g., [3, 4, 5, 11, 16]).

In recent years, a powerful tool has been introduced into the design of robust SPD preconditioners and it is to use low-rank approximations for certain dense blocks in A, A^{-1}, or some factors of A. A common way is to directly approximate A by rank-structured forms such as the ones in [2, 6, 7, 14, 34], but it is usually difficult to justify the performance of the resulting preconditioners. On the other hand, there are two types of methods that enable rigorous analysis of the effectiveness. One type is in [18, 19, 20, 28] based on low-rank strategies for approximating A^{-1}. Another type is in [1, 9, 12, 13, 21, 33, 35, 36] where approximate Cholesky factorizations are computed using low-rank approximations of relevant off-diagonal blocks. Both types of methods have been shown useful for many applications. A critical underlying reason (sometimes unnoticed in earlier work) behind the success of these preconditioners is actually to apply appropriate block diagonal scaling to A first and then compress the

*Submitted for review.

Funding: The research of Jianlin Xia was supported in part by an NSF grant DMS-1819166.

†Department of Mathematics, Purdue University, West Lafayette, IN 47907 (xiaj@math.purdue.edu).

resulting scaled off-diagonal blocks. A systematic way to formalize this is given in [35] as a so-called scaling-and-compression strategy and the resulting factorization is said to be a structured incomplete factorization (SIF). The preconditioning technique is called SIF preconditioning.

The basic idea of (one-level) SIF preconditioning is as follows [35]. Suppose A is $N \times N$ and is partitioned as

$$A \equiv \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}. \tag{1.1}$$

where the diagonal blocks A_{11} and A_{22} have Cholesky factorizations of the forms

$$A_{11} = L_1 L_1^T, \quad A_{22} = L_2 L_2^T. \tag{1.2}$$

Then the inverses of these Cholesky factors are used to scale the off-diagonal blocks. That is, let

$$C = L_1^{-1} A_{12} L_2^{-T}. \tag{1.3}$$

Suppose C has singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_k$ (which are actually all smaller than 1), where k is the smaller of the row and column sizes of C. Then the singular values σ_i are truncated aggressively so as to enable the quick computation of a rank structured approximate factorization of A.

Thus, the SIF technique essentially employs block diagonal scaling to preprocess A before relevant compression. This makes a significant difference as compared with standard rank-structured preconditioners that are based on direct off-diagonal compression. Accordingly, the SIF preconditioner has some attractive features, such as the convenient analysis of the performance, the convenient control of the approximation accuracy, and the nice effectiveness for preconditioning [35, 36]. In fact, if only r largest singular values of C are kept in its low-rank approximation, then the resulting preconditioner (called a one-level or prototype preconditioner) approximates A with a relative accuracy bound σ_{r+1}. The preconditioner also produces a condition number $\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$ for the preconditioned matrix. This idea can be repeatedly applied to the diagonal blocks to yield a practical multilevel SIF preconditioner.

A key idea for the effectiveness of the SIF preconditioner lies in a decay magnification effect [33, 35]. That is, although for a matrix A where the singular values σ_i of C may only slightly decay, the condition number $\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$ decays at a much faster rate to 1. Thus, it is possible to use a relatively small truncation rank r to get a structured preconditioner that is both effective and efficient to apply. A similar reason is also behind the effectiveness of those preconditioners in [18, 19, 20, 28, 33].

However, the SIF preconditioning has two major limitations. One is in the robustness. In the multilevel case, it needs a strict condition to avoid breakdown and ensure the existence or positive definiteness of the preconditioner. This condition needs either the condition number of A to be reasonably small, the low-rank approximation tolerance to be small, or the number of levels to be small. These mean the sacrifice of either the applicability or the efficiency of the preconditioner, as pointed out in [36].

Another limitation is in the effectiveness. Although the condition number form $\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$ has the decay magnification effect, if the decay of σ_i is too slow, using small r would not reduce the condition number too much. With small r, the eigenvalues of the preconditioned matrix may not closely cluster around 1 either. The performance of the preconditioner can then be less satisfactory.

Therefore, the motivation of this work is to overcome both limitations of the SIF technique. We make enhancements in several aspects. First, we would like get rid of the condition in the SIF scheme that avoids breakdown. That is, we produce a type of structured preconditioners that is *unconditionally robust* or always positive definite. Second, we would like to approximate A with *better accuracies* using the same truncation rank r. Next, we intend to *accelerate the decay magnification effect* in the condition number form. Lastly, we also try to *improve the eigenvalue clustering* of the preconditioned matrix. (We originally discussed how to achieve these enhancements in the presentation [32].)

Our idea to achieve these enhancements is to make some subtle changes to the original SIF scheme. Instead of block diagonal scaling, we use two-sided block triangular preprocessing which leads to an *enhanced scaling-and-compression strategy*. Then a low-rank approximation is still computed for C, but it is just used to accelerate computations related to Schur complements instead of off-diagonal blocks. (This will be made more precise in Section 2.) This strategy can be repeatedly applied to A_{11} and A_{22} in (1.1) so as to yield an efficient structured multilevel preconditioner.

This strategy makes it convenient to analyze the resulting preconditioners. The one-level preconditioner can now approximate A with a relative accuracy bound σ_{r+1}^2 (in contrast with the bound σ_{r+1} in the SIF case). The preconditioned matrix now has condition number $\frac{1}{1-\sigma_{r+1}^2}$, which is a significant improvement from $\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$ due to the quadratic form σ_{r+1}^2 and the smaller numerator. Similar improvements are also achieved with the multilevel preconditioner.

Moreover, the eigenvalues of the preconditioned matrix are now more closely clustered around 1. With the new one-level preconditioner, the eigenvalues are redistributed to $[1-\sigma_{r+1}^2, 1]$, with the eigenvalue 1 of multiplicity $N-(k-r)$. In comparison, the one-level SIF preconditioner only brings the eigenvalues to the interval $[1-\sigma_{r+1}, 1+\sigma_{r+1}]$, with the eigenvalue 1 of multiplicity $N-2(k-r)$. Similarly, the new multilevel preconditioner also greatly improves the eigenvalue clustering.

In addition, the multilevel generalization of the strategy always produces a positive definite preconditioner $\tilde{A}$ without the need of extra stabilization or diagonal compensation. In fact, the scheme has an automatic *positive definiteness enhancement effect*. That is, $\tilde{A}$ is equal to A plus a positive semidefinite matrix. Thus, the new multilevel preconditioner is unconditionally robust.

Due to all these enhancements, the new preconditioner is called an *enhanced SIF (eSIF) preconditioner*. We give comprehensive analysis of the accuracy, robustness, and effectiveness of both the one-level and the multilevel eSIF preconditioners in Theorems 2.1, 2.2, 3.1 and 3.2. All the benefits combined yield significantly better effectiveness than the SIF scheme. With the same number of levels and the same truncation rank r, although the eSIF preconditioner is slightly more expensive to apply in each iteration step, the total iterative solution cost is much lower.

We also show some techniques to design a practical multilevel eSIF scheme and then analyze the complexity and storage. The practical scheme avoids forming dense blocks like C in (1.3) while enabling the convenient low-rank approximation of these blocks. It also produces structured factors defined by compact forms such as Householder vectors.

The performance of the preconditioner is illustrated in terms of some challenging test matrices including some from [35]. As compared with the SIF preconditioner, the eSIF preconditioner yields dramatic reductions in the number of conjugate gradient iterations.

We would also like to mention some other relevant work. In earlier work [13, 33] where off-diagonal scaling and compression are used, although local Schur complement approximations have quadratic accuracy bounds like $O(\tau^2)$ in terms of a truncation tolerance τ, the overall accuracy (in their one-level scheme) is $O(\tau)$ due to the approximation of the scaled off-diagonal blocks. There is no accuracy analysis for the multilevel schemes in [13, 33]. An overall linear accuracy bound also arises in [37]. All these schemes have factorization complexity quadratic in N unless some structures are predetermined like in [38]. After the original submission of the current paper, an arXiv preprint [17] was posted and its latest version also cites the arXiv version [31] of our paper. The work in [17] deals with sparse SPD matrices instead of dense ones and uses a related strategy to achieve quadratic approximation accuracy. A condition number study for its one-level scheme is given in [17], but not for its multilevel one. Since the work in [17] approximates local Schur complements in the factorization of sparse matrices, the overall complexity is likely lower than quadratic, which is unclear from [17] though.

The organization of the remaining sections is as follows. The enhanced scaling-and-compression strategy and the one-level eSIF preconditioner will be presented and analyzed in Section 2. The techniques and analysis will then be generalized to multiple levels in Section 3. Section 4 further gives the practical multilevel design of the preconditioning scheme and also analyzes the storage and costs. Comprehensive numerical tests will be given in Section 5, following by some conclusions and discussions in Section 6. For convenience, we list frequently used notation as follows.

- $\lambda(A)$ is used to represent an eigenvalue of A (it is used in a general way and is not for any specific eigenvalue).
- $\kappa(A)$ denotes the 2-norm condition number of A.
- $\mathrm{diag}(\cdot)$ is used to mean a diagonal or block diagonal matrix constructed with the given diagonal entries or blocks.
- I_n is the $n \times n$ identity matrix and is used to distinguish identity matrices of different sizes in some contexts.

2. Enhanced scaling-and-compression strategy and prototype eSIF preconditioner. We first give the enhanced scaling-and-compression strategy and analyze the resulting prototype eSIF preconditioner in terms of the accuracy, robustness, and effectiveness.

In the SIF preconditioner in [35], A in (1.1) can be written as a factorized form as follows based on (1.2) and (1.3):

$$A = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & C \\ C^T & I \end{pmatrix} \begin{pmatrix} L_1^T & \\ & L_2^T \end{pmatrix}, \tag{2.1}$$

where $\begin{pmatrix} I & C \\ C^T & I \end{pmatrix}$ can be viewed as the result after the block diagonal preprocessing or scaling of A. C is then approximated by a low-rank form so as to obtain a rank-structured approximate factorization of A.

Here, we make some subtle changes which will turn out to make a significant difference. Rewrite (2.1) in the following form:

$$A = \begin{pmatrix} L_1 & \\ L_2C^T & L_2 \end{pmatrix} \begin{pmatrix} I & \\ & I - C^TC \end{pmatrix} \begin{pmatrix} L_1^T & CL_2^T \\ & L_2^T \end{pmatrix}. \tag{2.2}$$

Suppose C is $m \times n$ and a rank-r truncated SVD of C is

$$C \approx U_1 \Sigma_1 V_1^T, \tag{2.3}$$

where $\Sigma_1 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$ is for the largest r singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r$ of C. For later convenience, we also let the full SVD of C be

$$C = U\Sigma V^T = U_1\Sigma_1 V_1^T + U_2\Sigma_2 V_2^T, \tag{2.4}$$

where $U = \begin{pmatrix} U_1 & U_2 \end{pmatrix}$ and $V = \begin{pmatrix} V_1 & V_2 \end{pmatrix}$ are orthogonal and Σ_2 is a (rectangular) diagonal matrix for the remaining singular values $\sigma_{r+1} \geq \cdots \geq \sigma_{\min\{m,n\}}$. We further suppose τ is a tolerance for truncating the singular values in (2.3). That is,

$$\sigma_r \geq \tau \geq \sigma_{r+1}. \tag{2.5}$$

Note that all the singular values σ_i of C satisfy $\sigma_i < 1$ [35], so $\tau < 1$.

The apply (2.3) to C^TC in (2.2) to get

$$C^TC \approx V_1\Sigma_1^2 V_1^T.$$

In the meantime, we preserve the original form of C in the two triangular factors in (2.2). Accordingly,

$$A \approx \tilde{A} \equiv \begin{pmatrix} L_1 & \\ L_2C^T & L_2 \end{pmatrix} \begin{pmatrix} I & \\ & I - V_1\Sigma_1^2 V_1^T \end{pmatrix} \begin{pmatrix} L_1^T & CL_2^T \\ & L_2^T \end{pmatrix} \tag{2.6}$$

Suppose $\tilde{D}_2$ is the lower triangular Cholesky factor of $I - V_1\Sigma_1^2 V_1^T$:

$$I - V_1\Sigma_1^2 V_1^T = \tilde{D}_2\tilde{D}_2^T. \tag{2.7}$$

Let

$$\tilde{L} = \begin{pmatrix} L_1 & \\ L_2C^T & L_2 \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{D}_2 \end{pmatrix} = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & \\ C^T & I \end{pmatrix} \begin{pmatrix} I & \\ & \tilde{D}_2 \end{pmatrix}. \tag{2.8}$$

Then we get a *prototype (1-level) eSIF preconditioner*

$$\tilde{A} = \tilde{L}\tilde{L}^T. \tag{2.9}$$

This scheme can be understood as follows. Unlike in the SIF scheme where A is preprocessed by the block diagonal factor $\begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix}$, here we use a block triangular factor $\begin{pmatrix} L_1 & \\ L_2C^T & L_2 \end{pmatrix}$ to preprocess A. Note that it is still convenient to invert $\begin{pmatrix} L_1 & \\ L_2C^T & L_2 \end{pmatrix} = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & \\ C^T & I \end{pmatrix}$ in linear system solution so the form of C does not cause any substantial trouble. Also, we do not need to explicitly form or compress C. In addition, the Cholesky factor $\tilde{D}_2$ in (2.7) is only used for the purpose of analysis and does not need to be computed. The details will be given later in a more practical scheme in Section 4.

This leads to our *enhanced scaling-and-compression strategy*. We then analyze the properties of the resulting prototype eSIF preconditioner. Obviously, $\tilde{A}$ in (2.9) always exists and is positive definite. Furthermore, an additional benefit in the positive definiteness can be shown. We take a closer look at the positive definiteness of $\tilde{A}$ and also the accuracy of $\tilde{A}$ for approximating A.

THEOREM 2.1. *Let τ be the truncation tolerance in (2.5). $\tilde{A}$ in (2.9) satisfies*

$$\tilde{A} = A + E,$$

where E is a positive semidefinite matrix and

$$\frac{\|E\|_2}{\|A\|_2} \leq \sigma_{r+1}^2 \leq \tau^2. \tag{2.10}$$

In addition,

$$\frac{\|\tilde{L} - L\|_2}{\|L\|_2} \leq \frac{c\sqrt{1-\sigma_n^2}}{1-\sigma_1^2}\tau^2, \tag{2.11}$$

where L is the lower triangular Cholesky factor of A, $c = 1 + 2\lceil \log_2 n \rceil$, and σ_n is either the n-th singular value of C when $m \geq n$ or is 0 otherwise. On the other hand, if $\tilde{D}_2$ in $\tilde{L}$ in (2.8) is replaced by $(I - V_1\Sigma_1^2 V_1^T)^{1/2}$ and L is modified accordingly as $L = \begin{pmatrix} L_1 & \\ L_2 C^T & L_2(I - V\Sigma^T\Sigma V^T)^{1/2} \end{pmatrix}$ so that $A = LL^T$ still holds, then

$$\frac{\|\tilde{L} - L\|_2}{\|L\|_2} < \tau^2. \tag{2.12}$$

Proof. From (2.4) and (2.6), $\tilde{A}$ can be written as

$$\begin{aligned}
\tilde{A} &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & L_2 C^T C L_2^T + L_2(I - V_1\Sigma_1^2 V_1^T)L_2^T \end{pmatrix} \\
&= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} + L_2(C^T C - V_1\Sigma_1^2 V_1^T)L_2^T \end{pmatrix} \\
&= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} + L_2(V_2\Sigma_2^T\Sigma_2 V_2^T)L_2^T \end{pmatrix} = A + E,
\end{aligned}$$

where $E = \mathrm{diag}(0, L_2(V_2\Sigma_2^T\Sigma_2 V_2^T)L_2^T)$ is positive semidefinite and

$$\|E\|_2 = \|L_2(V_2\Sigma_2^T\Sigma_2 V_2^T)L_2^T\|_2 \leq \sigma_{r+1}^2\|L_2\|_2^2 = \sigma_{r+1}^2\|A_{22}\|_2 \leq \sigma_{r+1}^2\|A\|_2.$$

Also, let $D_2 D_2^T = I - V\Sigma^T\Sigma V^T$. Then $L = \begin{pmatrix} L_1 & \\ L_2 C^T & L_2 D_2 \end{pmatrix}$. Thus,

$$\begin{aligned}
\|\tilde{L} - L\|_2 &= \left\| \begin{pmatrix} L_1 & \\ L_2 C^T & L_2\tilde{D}_2 \end{pmatrix} - \begin{pmatrix} L_1 & \\ L_2 C^T & L_2 D_2 \end{pmatrix} \right\|_2 \\
&= \left\| \begin{pmatrix} 0 & \\ & L_2(\tilde{D}_2 - D_2) \end{pmatrix} \right\|_2 \leq \|L\|_2\|\tilde{D}_2 - D_2\|_2.
\end{aligned} \tag{2.13}$$

When D_2 is the lower triangular Cholesky factor of $I - V\Sigma^T\Sigma V^T$, an inequality in [35] gives

$$\|\tilde{D}_2 - D_2\|_2 \leq \frac{c\sqrt{1-\sigma_n^2}}{1-\sigma_1^2}\sigma_{r+1}^2, \quad c = 1 + 2\lceil \log_2 n \rceil.$$

This leads to (2.11).

If $\tilde{D}_2$ in $\tilde{L}$ is replaced by $(I-V_1\Sigma_1^2V_1^T)^{1/2}$ and D_2 is replaced by $(I-V\Sigma^T\Sigma V^T)^{1/2}$, then

$$\begin{aligned}\|\tilde{D}_2 - D_2\|_2 &= \|(I - V_1\Sigma_1^2V_1^T)^{1/2} - (I - V\Sigma^T\Sigma V^T)^{1/2}\|_2\\ &= \|(I - \mathrm{diag}(\Sigma_1^2, 0))^{1/2} - (I - \Sigma^T\Sigma)^{1/2}\|_2\\ &= 1 - \sqrt{1-\sigma_{r+1}^2} < \sigma_{r+1}^2.\end{aligned}$$

Then following (2.13), we get (2.12). ∎

This theorem gives both the accuracy and the robustness of the prototype eSIF preconditioner. Unlike the SIF framework where a similar prototype preconditioner has a relative accuracy bound τ, here the bound is τ^2 that is much more accurate. In addition, this theorem means the construction of $\tilde{A}$ automatically has a *positive definiteness enhancement effect*: it implicitly compensates A by a positive semidefinite matrix E. This is similar to ideas in [13, 33]. Later, we will show that this effect further carries over to the multilevel generalization, which is not the case for the SIF preconditioner.

The effectiveness of the prototype eSIF preconditioner can be shown as follows.

THEOREM 2.2. *The eigenvalues of* $\tilde{L}^{-1}A\tilde{L}^{-T}$ *are*

$$\lambda(\tilde{L}^{-1}A\tilde{L}^{-T}) = 1-\sigma_{r+1}^2, \ldots, 1-\sigma_k^2, \underbrace{1, \ldots, 1}_{N-(k-r)},$$

where $k = \min\{m, n\}$. *Accordingly,*

$$\begin{aligned}\|\tilde{L}^{-1}A\tilde{L}^{-T} - I\|_2 &= \sigma_{r+1}^2 \le \tau^2,\\ \kappa(\tilde{L}^{-1}A\tilde{L}^{-T}) &= \frac{1}{1-\sigma_{r+1}^2} \le \frac{1}{1-\tau^2}.\end{aligned}$$

Proof. It is not hard to verify

$$\tilde{L}^{-1}A\tilde{L}^{-T} = \mathrm{diag}(I_{N-n}, \tilde{D}_2^{-1}(I_n - V\Sigma^T\Sigma V^T)\tilde{D}_2^{-T}). \tag{2.14}$$

The eigenvalues of $\tilde{D}_2^{-1}(I_n - V\Sigma^T\Sigma V^T)\tilde{D}_2^{-T}$ are

$$\begin{aligned}\lambda(\tilde{D}_2^{-1}(I_n - V\Sigma^T\Sigma V^T)\tilde{D}_2^{-T}) &= \lambda(\tilde{D}_2^{-T}\tilde{D}_2^{-1}(I_n - V\Sigma^T\Sigma V^T))\\ &= \lambda((I_n - V_1\Sigma_1^2V_1^T)^{-1}(I_n - V\Sigma^T\Sigma V^T)).\end{aligned}$$

Further derivations can be done via the Sherman-Morrison-Woodbury formula or in the following way:

$$\begin{aligned}&(I_n - V_1\Sigma_1^2V_1^T)^{-1}(I_n - V\Sigma^T\Sigma V^T)\\ &= (V(I_n - \mathrm{diag}(\Sigma_1^2, 0))V^T)^{-1}V(I_n - \Sigma^T\Sigma)V^T\\ &= V\,\mathrm{diag}((I_r - \Sigma_1^2)^{-1}, I_{n-r})(I_n - \Sigma^T\Sigma)V^T\\ &= V\,\mathrm{diag}(I_r, I_{n-r} - \Sigma_2^T\Sigma_2)V^T.\end{aligned}$$

Thus,

$$\lambda(\tilde{D}_2^{-1}(I_n - V\Sigma^T\Sigma V^T)\tilde{D}_2^{-T}) = \lambda(\mathrm{diag}(I_r, I_{n-r} - \Sigma_2^T\Sigma_2)), \tag{2.15}$$

which are just $1-\sigma_{r+1}^2,\ldots,1-\sigma_k^2,1$. The eigenvalue 1 is a multiple eigenvalue. If $k=n$, then the eigenvalue 1 in (2.15) has multiplicity r. If $k=m$, $I_{n-r}-\Sigma_2^T\Sigma_2$ also has $n-k$ eigenvalues equal to 1 so the eigenvalue 1 in (2.15) has multiplicity $n-(k-r)$. For both cases, the eigenvalue 1 of $\tilde{L}^{-1}A\tilde{L}^{-T}$ has multiplicity $N-(k-r)$ according to (2.14). ∎

To give an idea on the advantages of the prototype eSIF preconditioner over the corresponding prototype SIF preconditioner in [35], we compare the results in Table 2.1 with $\tilde{L}$ and $\tilde{A}$ from the eSIF or SIF scheme. The eSIF scheme yields a much higher approximation accuracy than SIF (τ^2 vs. τ) for both $\frac{\|A-\tilde{A}\|_2}{\|A\|_2}$ and $\|\tilde{L}^{-1}A\tilde{L}^{-T}-I\|_2$. The eigenvalues of the preconditioned matrix $\tilde{L}^{-1}A\tilde{L}^{-T}$ from eSIF are also much more closely clustered around 1 and eSIF produces a lot more eigenvalues equal to 1 than SIF. This is further illustrated in Figure 2.1.

TABLE 2.1
Comparison of prototype SIF and eSIF preconditioners that are used to produce $\tilde{L}$ and $\tilde{A}$, where $k=\min\{m,n\}$ and the results for the SIF preconditioner are from [35, 36].

	SIF	eSIF
$\frac{\|\tilde{A}-A\|_2}{\|A\|_2}$	$\le \tau$	$\le \tau^2$
$\frac{\|\tilde{L}-L\|_2}{\|L\|_2}$	$\le \tau + \frac{c\sqrt{1-\sigma_n^2}}{1-\sigma_1^2}\tau^2$	$\le \frac{c\sqrt{1-\sigma_n^2}}{1-\sigma_1^2}\tau^2$
$\lambda(\tilde{L}^{-1}A\tilde{L}^{-T})$	$1\pm\sigma_{r+1},\ldots,1\pm\sigma_k,\underbrace{1,\ldots,1}_{N-2(k-r)}$	$1-\sigma_{r+1}^2,\ldots,1-\sigma_k^2,\underbrace{1,\ldots,1}_{N-(k-r)}$
$\|\tilde{L}^{-1}A\tilde{L}^{-T}-I\|_2$	$\sigma_{r+1}\le\tau$	$\sigma_{r+1}^2\le\tau^2$
$\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$	$\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}\le\frac{1+\tau}{1-\tau}$	$\frac{1}{1-\sigma_{r+1}^2}\le\frac{1}{1-\tau^2}$

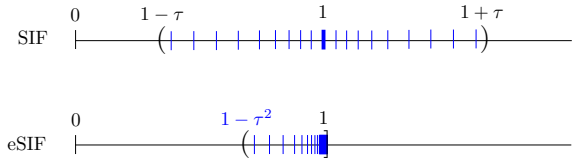


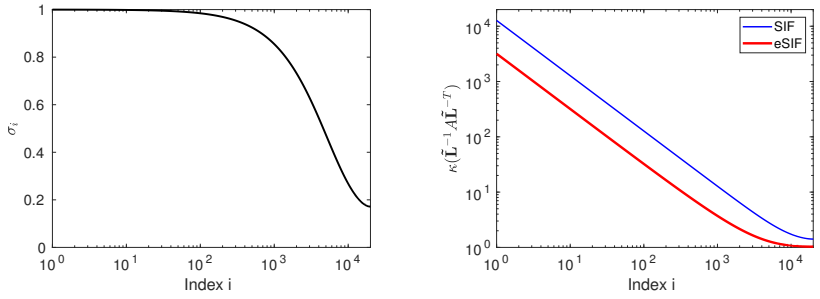
FIG. 2.1. *How the eigenvalues $\lambda(\tilde{L}^{-1}A\tilde{L}^{-T})$ cluster around 1 when $\tilde{L}\tilde{L}^T$ is obtained with the prototype SIF and eSIF preconditioners.*

Specifically, SIF produces $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})=\frac{1+\sigma_{r+1}}{1-\sigma_{r+1}}$, while eSIF leads to much smaller $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})=\frac{1}{1-\sigma_{r+1}^2}$. (Notice the quadratic term σ_{r+1}^2 in the denominator and the smaller numerator.) To further illustrate the difference in $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$, we use an example like in [35]. In the example, the singular values of C look like those in Figure 2.2(a) and are based on the analytical forms from a 5-point discrete Laplacian matrix [36]. The singular values of C in (1.3) only slowly decay. Figure 2.2(b) shows $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$ from both schemes. We can observe two things.

1. Like in SIF, the modest decay of the nonzero singular values σ_i of C is further dramatically magnified in $\frac{1}{1-\sigma_i^2}$. That is, even if σ_i decays slowly, $\frac{1}{1-\sigma_i^2}$ decays much faster so that σ_i can still be aggressively truncated so as to produce reasonably small $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$. This is the *decay magnifying effect* like in

[35].

2. Furthermore, the decay magnification effect from eSIF is more dramatic since $\frac{1}{1-\sigma_i^2}$ is smaller than $\frac{1+\sigma_i}{1-\sigma_i}$ by a factor of $(1+\sigma_i)^2$. For a large range of r values, eSIF gives much better condition numbers than SIF.



(a) Nonzero singular values σ_i of C (b) $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$

FIG. 2.2. *For an example where the singular values σ_i of C slowly decay, how $\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$ decays when $\tilde{L}$ is from the prototype SIF or eSIF preconditioner obtained by truncating σ_i with r set to be i in (b).*

Remark 2.3. The approximation accuracy σ_{r+1} depends on the ordering and partitioning of A. It is desirable to reorder and partition A so as to make σ_{r+1} as small as possible. Since it is generally unknown in advance what σ_{r+1} would be like (other than $\sigma_{r+1} < 1$), we may try to reduce the numerical rank of A_{12} as much as possible. However, just like most other hierarchical rank-structured methods, there is no quick way to reorder a general dense matrix to reduce its off-diagonal numerical ranks. For some cases, heuristics might be used. For example, if A corresponds to certain underlying mesh or data points, then we may permute and partition A following a partitioning of the mesh or point set so that the connection or interaction between the resulting subsets is as weak as possible. Sometimes, this may also be combined with randomized processes (see, e.g, [10]). Since we deal with general dense SPD matrices, our studies do not require any specific ordering and the ordering issue is expected to be considered in future work. In addition, one thing that is worth mentioning is that, as pointed out in [35, 36], the scaling of the off-diagonal blocks often has an effect of enhancing the decay of off-diagonal singular values. For instance, for the matrix example used in Figure 2.2, the original A_{12} block has a negative identity matrix and the nonzero singular values do not decay at all. After scaling, the nonzero singular values σ_i of C have reasonable decay. See Figure 2.2(a). This is also a feature exploited in [18, 19, 20, 28].

3. Multilevel eSIF preconditioner. The prototype preconditioner in the previous section still has two dense Cholesky factors L_1 and L_2 in (2.8). To get an efficient preconditioner, we generalize the prototype preconditioner to multiple levels. That is, apply it repeatedly to the diagonal blocks of A. For convenience, we use eSIF(1) to denote the prototype 1-level eSIF scheme. A 2-level eSIF scheme or eSIF(2) uses eSIF(1) to obtain approximate factors $\tilde{L}_1 \approx L_1$ and $\tilde{L}_2 \approx L_2$ for (1.2). Similarly, an l-level eSIF scheme or eSIF(l) uses eSIF($l-1$) to approximate L_1 and L_2. With a sufficient number of levels (usually $l = O(\log N)$), the finest level diagonal blocks are small enough and can be directly factorized. The overall resulting factor $\tilde{L}$ is an eSIF(l) factor. The resulting approximation matrix $\tilde{A}$ is an eSIF(l) preconditioner.

We prove that the eSIF(l) preconditioner $\tilde{A}$ is always positive definite and show how accurate $\tilde{A}$ is for approximating A.

THEOREM 3.1. *Let τ be the tolerance for any singular value truncation like (2.3)–(2.5) in the eSIF(l) scheme. The approximate matrix $\tilde{A}$ resulting from eSIF(l) is always positive definite and satisfies*

$$\tilde{A} = A + E, \tag{3.1}$$

where E is a positive semidefinite matrix and

$$\frac{\|E\|_2}{\|A\|_2} \leq (1+\tau^2)^l - 1.$$

Proof. We prove this by induction. $l = 1$ corresponds to eSIF(1) and the result is in Theorem 2.1. Suppose the result holds for eSIF($l-1$) with $l > 1$. Apply eSIF($l-1$) to A_{11} and A_{22} to get approximate Cholesky factors $\tilde{L}_1$ and $\tilde{L}_2$, respectively. By induction, we have

$$\tilde{L}_1 \tilde{L}_1{}^T = A_{11} + E_1, \quad \tilde{L}_2 \tilde{L}_2{}^T = A_{22} + E_2,$$

where E_1 and E_2 are positive semidefinite matrices satisfying

$$\|E_1\|_2 \leq \left[(1+\tau^2)^{l-1} - 1\right] \|A_{11}\|_2 \leq \left[(1+\tau^2)^{l-1} - 1\right] \|A\|_2,$$
$$\|E_2\|_2 \leq \left[(1+\tau^2)^{l-1} - 1\right] \|A_{22}\|_2 \leq \left[(1+\tau^2)^{l-1} - 1\right] \|A\|_2.$$

Thus,

$$A \approx \begin{pmatrix} \tilde{L}_1 \tilde{L}_1{}^T & A_{21}^T \\ A_{21} & \tilde{L}_2 \tilde{L}_2{}^T \end{pmatrix} = A + \operatorname{diag}(E_1, E_2) \equiv \hat{A}.$$

Clearly, $\hat{A}$ is always positive definite.

Then apply eSIF(1) to $\hat{A}$ to yield

$$\hat{A} \approx \tilde{A} \equiv \tilde{L}\tilde{L}^T,$$

where $\tilde{L}$ is the eSIF(l) factor. With Theorem 2.1 applied to $\hat{A}$, we get

$$\tilde{A} = \hat{A} + \tilde{E},$$

where $\tilde{E}$ is a positive semidefinite matrix satisfying $\|\tilde{E}\|_2 \leq \tau^2 \|\hat{A}\|_2$. Then

$$\tilde{A} = A + (\operatorname{diag}(E_1, E_2) + \tilde{E}) \equiv A + E,$$

where $E = \operatorname{diag}(E_1, E_2) + \tilde{E}$ is positive semidefinite. Thus, $\tilde{A}$ is positive definite and

$$\begin{aligned}
\|E\|_2 &\leq \|\operatorname{diag}(E_1, E_2)\|_2 + \|\tilde{E}\|_2 \\
&\leq \|\operatorname{diag}(E_1, E_2)\|_2 + \tau^2 \|\hat{A}\|_2 \\
&= \|\operatorname{diag}(E_1, E_2)\|_2 + \tau^2 \|A + \operatorname{diag}(E_1, E_2)\|_2 \\
&\leq \tau^2 \|A\|_2 + (1+\tau^2) \|\operatorname{diag}(E_1, E_2)\|_2 \\
&\leq \tau^2 \|A\|_2 + (1+\tau^2) \left[(1+\tau^2)^{l-1} - 1\right] \|A\|_2 \\
&= \left[(1+\tau^2)^l - 1\right] \|A\|_2.
\end{aligned}$$

The result then holds by induction. ∎

Thus, $\frac{\|E\|_2}{\|A\|_2}$ is roughly $O(l\tau^2)$ for reasonable τ, which indicates a very slow levelwise approximation error accumulation. Moreover, like eSIF(1), eSIF(l) also has a positive definiteness enhancement effect so that $\tilde{A}$ remains positive definite. In contrast, the multilevel SIF scheme in [35] may breakdown due to the loss of positive definiteness.

Then we can look at the effectiveness of the eSIF(l) preconditioner.

THEOREM 3.2. *Let τ be the tolerance for any singular value truncation like (2.3)–(2.5) in the eSIF(l) scheme and $\epsilon = \left[(1+\tau^2)^l - 1\right]\kappa(A)$. Let $\tilde{L}$ be the eSIF(l) factor. Then the eigenvalues of the preconditioned matrix $\tilde{L}^{-1}A\tilde{L}^{-T}$ satisfy*

$$\frac{1}{1+\epsilon} \le \lambda(\tilde{L}^{-1}A\tilde{L}^{-T}) \le 1. \tag{3.2}$$

Accordingly,

$$\|\tilde{L}^{-1}A\tilde{L}^{-T} - I\|_2 \le \frac{\epsilon}{1+\epsilon},$$
$$\kappa(\tilde{L}^{-1}A\tilde{L}^{-T}) \le 1+\epsilon.$$

Proof. Let $A = LL^T$ be the Cholesky factorization of A. With (3.1),

$$L^{-1}\tilde{A}L^{-T} = I + L^{-1}(\tilde{A} - A)L^{-T} = I + L^{-1}EL^{-T},$$

According to Theorem 3.1, $L^{-1}EL^{-T}$ is positive semidefinite. Thus, $\lambda(L^{-1}\tilde{A}L^{-T}) \ge 1$.

Theorem 3.1 also yields

$$\begin{aligned}\|L^{-1}EL^{-T}\|_2 &\le \|E\|_2\|L^{-1}\|_2\|L^{-T}\|_2 \\ &\le \left[(1+\tau^2)^l - 1\right]\|A\|_2\|A^{-1}\|_2 = \epsilon.\end{aligned}$$

Therefore,

$$1 \le \lambda(L^{-1}\tilde{A}L^{-T}) \le 1+\epsilon.$$

Since the eigenvalues of $\tilde{L}^{-1}A\tilde{L}^{-T}$ are the inverses of those of $L^{-1}\tilde{A}L^{-T}$, we get (3.2). ∎

A comparison of the multilevel eSIF and SIF preconditioners is given in Table 3.1. The multilevel eSIF preconditioner has several significant advantages over the SIF one.

1. The multilevel eSIF preconditioner is unconditionally robust or is guaranteed to be positive definite, while the SIF one needs a strict (or even impractical) condition to ensure the positive definiteness of the approximation. That is, the SIF one needs $\hat{\epsilon} \equiv \left[(1+\tau)^l - 1\right]\kappa(A) < 1$. This means τ needs to be small and/or the magnitudes of l and $\kappa(A)$ cannot be very large.
2. The eSIF one gives a more accurate approximation to A with a relative error bound $(1+\tau^2)^l - 1$ instead of $(1+\tau)^l - 1$.
3. The eSIF one produces a much better condition number for the preconditioned matrix ($1+\epsilon$ vs. $\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}$ with ϵ further much smaller than $\hat{\epsilon}$).
4. The eSIF one further produces better eigenvalue clustering for the preconditioned matrix. The eigenvalues of the preconditioned matrix from eSIF lie in $[\frac{1}{1+\epsilon}, 1]$, while those from SIF lie in a much larger interval $[\frac{1}{1+\hat{\epsilon}}, \frac{1}{1-\hat{\epsilon}}]$.

A combination of these advantages makes the eSIF preconditioner much more effective, as demonstrated later in numerical tests.

TABLE 3.1
Comparison of l-level SIF and eSIF preconditioners that are used to produce $\tilde{L}$ and $\tilde{A}$, where the results for the SIF preconditioner are from [35].

	SIF	eSIF
Existence/ Positive definiteness	Conditional $(\hat{\epsilon} \equiv \left[(1+\tau)^l - 1\right] \kappa(A) < 1)$	Unconditional
$\frac{\|\tilde{A}-A\|_2}{\|A\|_2}$	$\leq (1+\tau)^l - 1$	$\leq (1+\tau^2)^l - 1$
$\lambda(\tilde{L}^{-1}A\tilde{L}^{-T})$	$\in [\frac{1}{1+\hat{\epsilon}}, \frac{1}{1-\hat{\epsilon}}]$	$\in [\frac{1}{1+\epsilon}, 1]$
$\|\tilde{L}^{-1}A\tilde{L}^{-T} - I\|_2$	$\leq \frac{\hat{\epsilon}}{1-\hat{\epsilon}}$	$\leq \frac{\epsilon}{1+\epsilon}$
$\kappa(\tilde{L}^{-1}A\tilde{L}^{-T})$	$\leq \frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}$	$\leq 1+\epsilon$

4. Practical eSIF(l) scheme. In our discussions above, some steps are used for convenience and are not efficient for practical preconditioning. In the design of a practical scheme for eSIF(l), we need to take care of the following points.

1. Avoid expensive dense Cholesky factorizations like in (2.7).
2. Avoid the explicit formation of C in (1.3) (needed in (2.8)) which is too costly.
3. Compute the low-rank approximation of C without the explicit form of C.

For the first point, we can let Q be an orthogonal matrix extended from V_1 in (2.3) so that

$$Q^T V_1 = \begin{pmatrix} I \\ 0 \end{pmatrix}.$$

Since V_1 has column size r which is typically small for the purpose of preconditioning, Q can be conveniently obtained with the aid of r Householder vectors. Due to this, Q is generally different from V in (2.4). Then (2.7) can be replaced by

$$I - V_1 \Sigma_1^2 V_1^T = Q(I - \mathrm{diag}(\Sigma_1^2, 0))Q^T.$$

Accordingly, $\tilde{A}$ in (2.6) can be rewritten as

$$\tilde{A} = \begin{pmatrix} L_1 & \\ L_2 C^T & L_2 \end{pmatrix} \begin{pmatrix} I & \\ & Q \end{pmatrix} \begin{pmatrix} I & \\ & I - \mathrm{diag}(\Sigma_1^2, 0) \end{pmatrix} \begin{pmatrix} I & \\ & Q^T \end{pmatrix} \begin{pmatrix} L_1^T & C L_2^T \\ & L_2^T \end{pmatrix}.$$

Thus, we can let

$$\tilde{L} = \begin{pmatrix} L_1 & \\ & L_2 \end{pmatrix} \begin{pmatrix} I & \\ C^T & I \end{pmatrix} \begin{pmatrix} I & \\ & Q\tilde{\Sigma}_1 \end{pmatrix}, \quad \text{with} \tag{4.1}$$

$$\tilde{\Sigma}_1 = \mathrm{diag}((I - \Sigma_1^2)^{1/2}, I) = \mathrm{diag}(\sqrt{1-\sigma_1^2}, \ldots, \sqrt{1-\sigma_r^2}, 1, \ldots, 1),$$

so that (2.9) still holds.

Next, we try to avoid the explicit formation of C in (1.3) which is too expensive. Note (4.1) means

$$\tilde{L}^{-1} = \begin{pmatrix} I & \\ & \tilde{\Sigma}_1^{-1} Q^T \end{pmatrix} \begin{pmatrix} I & \\ -C^T & I \end{pmatrix} \begin{pmatrix} L_1^{-1} & \\ & L_2^{-1} \end{pmatrix}.$$

If C is not formed but kept as the form in (1.3), then the application of $\tilde{L}^{-1}$ to a vector involves four smaller solution steps: one application of L_1^{-1} to a vector, one

application of L_1^{-T} to a vector, and two applications of L_2^{-1} to vectors. To reduce the number of such solutions, we rewrite $\tilde{L}$ in (4.1) as

$$\tilde{L} = \begin{pmatrix} L_1 & \\ & I \end{pmatrix} \begin{pmatrix} I & \\ A_{12}^T L_1^{-T} & L_2 \end{pmatrix} \begin{pmatrix} I & \\ & Q\tilde{\Sigma}_1 \end{pmatrix} \tag{4.2}$$

$$= \begin{pmatrix} L_1 & \\ & I \end{pmatrix} \begin{pmatrix} I & \\ A_{12}^T L_1^{-T} & I \end{pmatrix} \begin{pmatrix} I & \\ & L_2 Q\tilde{\Sigma}_1 \end{pmatrix}.$$

$\tilde{L}^{-1}$ now has the following form and can be conveniently applied to a vector:

$$\tilde{L}^{-1} = \begin{pmatrix} I & \\ & \tilde{\Sigma}_1^{-1} Q^T L_2^{-1} \end{pmatrix} \begin{pmatrix} I & \\ -A_{12}^T L_1^{-T} & I \end{pmatrix} \begin{pmatrix} L_1^{-1} & \\ & I \end{pmatrix}. \tag{4.3}$$

In fact, the application of $\tilde{L}^{-1}$ to a vector now just needs the applications of L_1^{-1}, L_1^{-T}, L_2^{-1} to vectors. In the eSIF(l) scheme, L_1 and L_2 are further approximated by structured factors from the eSIF($l-1$) scheme. In addition, Q^T is a Householder matrix defined by r Householder vectors and can be quickly applied to a vector. A_{12}^T is just part of A. With (4.2), there is no need to form C explicitly. From these discussions, it is also clear how $\tilde{L}^{-1}$ can be applied to vectors in actual preconditioning as structured solution.

Remark 4.1. With the form of $\tilde{L}$ in (4.2), it is clear that (2.9) still holds for $\tilde{A}$ in (2.6). Thus, the approximation error result (2.10) in Theorem 2.1 and the effectiveness results in Theorem 2.2 remain the same. This further means that Theorems 3.1 and 3.2 for the multilevel scheme still hold.

Thirdly, although C needs not to be formed, it still needs to be compressed so as to produce $\tilde{\Sigma}_1$ and Q in (4.2). We use randomized SVD [22] that is based on matrix-vector products. That is, let

$$Y = C^T Z = L_2^{-1}(A_{12}^T(L_1^{-T} Z)), \tag{4.4}$$

where Z is an appropriate skinny random matrix with column size $r + \alpha$ and α is a small constant oversampling size. Y can be used to extract an approximate row basis matrix $\hat{V}_1^T$ for C. After this, let

$$T = C\hat{V}_1 = L_1^{-1}(A_{12}(L_2^{-T}\hat{V}_1)). \tag{4.5}$$

$T\hat{V}_1^T$ essentially provides a low-rank approximation to C. Many studies of randomized SVDs in recent years have shown the reliability of this process. The tall and skinny matrix T can then be used to quickly extract r approximate leading singular values of C. Accordingly, this process provides an efficient way to get approximate Q and Σ_1. That is, we can compute an SVD $T = U_1\Sigma_1\tilde{V}_1^T$ and set $V_1 = \hat{V}_1\tilde{V}_1$. To improve the quality of the randomized approximation, a power iteration may also be used [15].

Computing Y in (4.4) and T in (4.5) uses linear solves in terms of L_1 and L_2 and matrix-vector multiplications in terms of A_{12}. When $\tilde{L}$ results from the eSIF(l) scheme, L_1 and L_2 are approximated by structured eSIF($l-1$) factors.

Algorithms 4.1 and 4.2 show the construction and application of the eSIF(l) preconditioner, respectively. The construction algorithm uses the solution algorithm. Algorithm 4.1 includes a simple randomized SVD scheme without the use of power iterations. To make it convenient to understand, the l-level schemes are constructed by calling the $(l-1)$-level schemes. In practical implementations, this may be changed

to the traversal of a binary tree so as to get scalable algorithms. Operations associated with each diagonal block correspond to a node of the binary tree. Operations associated with an off-diagonal block correspond to a pair of sibling nodes. Thus, at each level of the tree, the operations can be performed in parallel. This is very similar to the situations in various existing hierarchical rank-structured methods so that the parallelization can conveniently take advantage of techniques well developed in, say, [23, 24, 27, 39]. For example, like in the parallel randomized algorithms in [23] for hierarchically semiseparable (HSS) matrices [7, 34], a process grid can be used for the operations associated with each node of the tree. Distributed structured operations can then be conveniently designed. Since our focus here is on the design of the eSIF preconditioner and the theoretical analysis, the reader is referred to those references for relevant techniques for parallel implementations.

Algorithm 4.1 eSIF(l) factorization scheme (for constructing the preconditioner)

procedure $\tilde{L} = \mathsf{eSIF}(l, A, r, \alpha)$
  if $l = 0$ **then** ▷ *Finest level diagonal block*
    $A = \tilde{L}\tilde{L}^T$ ▷ *Cholesky factorization*
  else ▷ *Structured factorization*
    Partition A into a block 2×2 form like in (1.1)
    $\tilde{L}_1 \leftarrow \mathsf{eSIF}(l-1, A_{11}, r, \alpha)$, $\tilde{L}_2 \leftarrow \mathsf{eSIF}(l-1, A_{22}, r, \alpha)$
      ▷ *Diagonal block factorizations with eSIF($l-1$)*
    $Z \leftarrow$ skinny random matrix with column size $r + \alpha$
      ▷ *Lines 7–14: randomized SVD*
    $Y \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_1, Z, \text{'bwd'})$ ▷ $\tilde{L}_1^{-T} Z$
    $Y \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_2, A_{12}^T Y, \text{'fwd'})$ ▷ *Y like in (4.4)*
    $\hat{V}_1 \leftarrow$ leading r left singular vectors of Y
    $T \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_2, \hat{V}_1, \text{'bwd'})$ ▷ $\tilde{L}_2^{-T} \hat{V}_1$
    $T \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_1, A_{12} T, \text{'fwd'})$ ▷ *T like in (4.5)*
    $T = U_1 \tilde{\Sigma}_1 \tilde{V}_1^T$ ▷ *SVD*
    $V_1 \leftarrow \hat{V}_1 \tilde{V}_1$
    Extend V_1 to an orthogonal matrix Q
      ▷ *Q given in terms of Householder vectors*
    $\tilde{L} \leftarrow \{\tilde{L}_1, \tilde{L}_2, \Sigma_1, Q\}$ ▷ *$\tilde{L}$ given in terms of a series of structured factors*
  end if
end procedure

We then study the costs to construct and apply the eSIF(l) factor $\tilde{L}$ and the storage of $\tilde{L}$. In practice, we specify r instead of τ in low-rank compression so as to explicitly control the cost. Also see Remark 4.3 below.

PROPOSITION 4.2. *Suppose A is repeatedly bipartitioned into $l = \lfloor \log N \rfloor$ levels with the diagonal blocks at each partition level having the same size (for convenience). Let ξ_f be the complexity to compute the eSIF(l) factor $\tilde{L}$ where each intermediate low-rank approximation step uses rank r. Let ξ_s be the complexity to apply $\tilde{L}^{-1}$ to a vector. Then*

$$\xi_f = 6(r+\alpha)N^2 + O(r(r+\alpha)N^{\log_2 3}), \quad \xi_s = 2N^2 + O(rN^{\log_2 3}), \tag{4.6}$$

where α is a small constant oversampling size in randomized SVDs. (Here, we suppose no power iteration is used in randomized SVDs. Otherwise, the number of iterations

Algorithm 4.2 eSIF(l) solution via forward or backward substitution

procedure $x = \mathsf{eSIFsol}(l, \tilde{L}, b, s)$
  ▷ *Solving $\tilde{L}x = b$ or $\tilde{L}^T x = b$, depending on the variable s*
  if $s =$ 'fwd' **then** ▷ *Forward substitution for solving $\tilde{L}x = b$*
    if $l = 0$ **then** ▷ *Finest level dense solution*
      $x \leftarrow \tilde{L}^{-1} b$
    else ▷ *Structured solution (see (4.3))*
      $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \leftarrow b$ ▷ *Conformable partition following the sizes of $\tilde{L}_1, \tilde{L}_2$*
      $x_1 \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_1, b_1, \text{'fwd'})$
      $x_2 \leftarrow b_2 - A_{12}^T \cdot (\mathsf{eSIFsol}(l-1, \tilde{L}_1, x_1, \text{'bwd'}))$
      $x_2 \leftarrow Q^T \cdot (\mathsf{eSIFsol}(l-1, \tilde{L}_2, x_2, \text{'fwd'}))$
      $x_2(1:r) \leftarrow (I - \Sigma_1^2)^{-1/2} x_2(1:r)$
      ▷ *$\tilde{\Sigma}_1 = \mathrm{diag}((I - \Sigma_1^2)^{1/2}, I)$ like in (4.1); $x_2(1:r)$: first r entries of x_2*
      $x \leftarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
    end if
  else if $s =$ 'bwd' **then** ▷ *Backward substitution for solving $\tilde{L}^T x = b$*
    if $l = 0$ **then** ▷ *Finest level dense solution*
      $x \leftarrow \tilde{L}^{-T} x$
    else ▷ *Structured solution (see the transpose of (4.3))*
      $\begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \leftarrow b$ ▷ *Conformable partition following the sizes of $\tilde{L}_1, \tilde{L}_2$*
      $b_2(1:r) \leftarrow (I - \Sigma_1^2)^{-1/2} b_2(1:r)$
      ▷ *$\tilde{\Sigma}_1 = \mathrm{diag}((I - \Sigma_1^2)^{1/2}, I)$ like in (4.1); $b_2(1:r)$: first r entries of b_2*
      $x_2 \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_2, Qb_2, \text{'bwd'})$
      $x_1 \leftarrow b_1 - \mathsf{eSIFsol}(l-1, \tilde{L}_1, A_{12}x_2, \text{'fwd'})$
      $x_1 \leftarrow \mathsf{eSIFsol}(l-1, \tilde{L}_1, x_1, \text{'bwd'})$
      $x \leftarrow \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$
    end if
  end if
end procedure

will appear in ξ_f.) The storage of $\tilde{L}$ is

$$\theta = O(rN \log N),$$

excluding any storage for the blocks of A.

Proof. Let $\tilde{L}_1$ and $\tilde{L}_2$ be the eSIF($l-1$) factors that approximate L_1 and L_2, respectively. For the eSIF(l) factor $\tilde{L}$, we use $\xi_s(N)$ to denote the cost to apply $\tilde{L}^{-1}$ to a vector. According to (4.3),

$$\xi_s(N) = 3\xi_s(\frac{N}{2}) + 2(\frac{N}{2})^2 + O(rN),$$

where the first term on the right-hand side is for applying $\tilde{L}_1^{-1}$, $\tilde{L}_1^{-T}$, $\tilde{L}_2^{-1}$ to vectors, the second term is the dominant cost for multiplying A_{12}^T in (4.3) to a vector, and the third term is for the remaining costs (mainly to multiple Q^T to a vector). This gives

a recursive relationship which can be expanded to yield

$$\xi_s(N) = \frac{2}{3}N^2 \sum_{i=1}^{l} \frac{3^i}{4^i} + O(rN \sum_{i=1}^{l} \frac{3^i}{2^i}) \tag{4.7}$$
$$= 2N^2 + O(r3^l) = 2N^2 + O(rN^{\log_2 3}).$$

Then consider the cost $\xi_f(N)$ to compute $\tilde{L}$. We have

$$\xi_f(N) = 2\xi_f(\frac{N}{2}) + 4(r+\alpha)\xi_s(\frac{N}{2}) + 4(r+\alpha)(\frac{N}{2})^2 + O(r^2N),$$

where the first term on the right-hand side is for constructing $\tilde{L}_1$ and $\tilde{L}_2$, the second term is for applying the relevant inverses of these factors as in (4.4) and (4.5) during the randomized SVD, the third term is the dominant cost for multiplying A_{12}^T and A_{12} to vectors as in (4.4) and (4.5), and the last term is for the remaining costs. According to (4.7),

$$\xi_f(N) = 2\xi_f(\frac{N}{2}) + 3(r+\alpha)N^2 + O(r(r+\alpha)N^{\log_2 3}).$$

Based on this recursive relationship, we can obtain the count ξ_f in (4.6).

Finally, the storage $\theta(N)$ for $\tilde{L}$ (excluding the blocks of A) mainly includes the storage for $\tilde{L}_1, \tilde{L}_2$ and the r Householder vectors for Q in (4.2):

$$\theta(N) = 2\theta(\frac{N}{2}) + O(rN).$$

At the finest level of the partitioning of A, it also needs the storage of $O(rN)$ for the Cholesky factors of the small diagonal blocks. Essentially, the actual storage at each level is then $O(rN)$ and the total storage is $\theta = O(rlN)$. ∎

We can see that the storage for the structured factors is roughly linear in N since r is often fixed to be a small constant in preconditioning. The cost of applying $\tilde{L}^{-1}$ to a vector has a leading term $2N^2$. However, note that it costs about $2N^2$ to multiply A with a vector in each iteration anyway. For the SIF case in [35], the application cost is lower but each iteration step still costs $O(N^2)$ due to the matrix-vector multiplication. It also costs $O(rN^2)$ to construct the multilevel SIF preconditioners. The precise constant factor of the flop count is not given in [35]. There are two SIF versions in [35]. One also uses repeated block 2×2 partitioning of A like above and uses randomized SVDs. We can similarly show that the leading term of the complexity is $2(r+\alpha)N^2$. The second version involves nested off-diagonal basis matrices and has better robustness. Its cost is slightly higher in general, based on some counts from [30]. Thus, the construction of the eSIF preconditioner is a little more expensive. Nevertheless, the construction cost is just a one-time expense and the preconditioner can be used for multiple solves. Furthermore, SIF preconditioners may not exist for some cases due to the loss of positive definiteness. In the next section, we can see that the eSIF preconditioner can often dramatically reduce the number of conjugate gradient iterations so that it saves the solution cost significantly.

Remark 4.3. During the construction of the preconditioner, we specify r so as to explicitly control the cost of the preconditioner. Since the practical scheme uses randomized SVDs to avoid forming large dense blocks, it is actually not very convenient

to control the approximation accuracy via a tolerance τ for singular values. This is because there is not a direct mechanism to explicitly monitor the accuracy of singular values in the randomized process. With a certain number of random vectors, if the resulting singular values from randomized SVDs do not reach the desired tolerance, more random vectors are used, but then it is not immediate to get the next singular values. Instead, it needs to go through some reorthogonalizations, multiplications, and moreover, SVD updates. In other words, adaptive sampling with more random vectors does not immediately produce new (smaller) singular values on top of existing singular values, and the monitoring of the approximation accuracy is then not very convenient. This is why a probabilistic strategy is used to roughly estimate the approximation accuracy in a somewhat nontrivial adaptive scheme in previous work such as [15, 23, 24, 29]. To ensure reasonable reliability of the error estimate, if the estimated error satisfies a certain bound for a consecutive number of times, it assumes the approximation error meets the desired accuracy. This not only needs extra costs but can also highly overestimate the actual numerical rank for a desired accuracy. It may lead to r much larger than necessary and also varying a lot for different runs and different tolerances. This would then defeat the purpose of designing an efficient preconditioner since we want r to be quite small. Thus, directly using a prespecified r is much more convenient.

5. Numerical experiments. We then show the performance of the multilevel eSIF preconditioner in accelerating the convergence of the preconditioned conjugate gradient method (PCG). We compare the following three preconditioners.

- bdiag: the block diagonal preconditioner.
- SIF: an SIF preconditioner from [35] (for the two versions of SIF preconditioners in [35], we use the one with better robustness).
- eSIF: the multilevel eSIF preconditioner.

In [35], it has been shown that SIF is generally much more effective than a preconditioner based on direct approximations by HSS forms. Here, we would like to show how eSIF further outperforms SIF. The following notation is used to simplify the presentation of the test results.

- $\gamma = \frac{\|Ax-b\|_2}{\|b\|_2}$: 2-norm relative residual for a numerical solution x, with b generated using the exact solution vector of all ones.
- n_{iter}: total number of iterations to reach a certain accuracy for the relative residual.
- A_{prec}: matrix preconditioned by the factors from the preconditioners (for example, $A_{\text{prec}} = \tilde{L}^{-1}A\tilde{L}^{-T}$ in the eSIF case).
- r: numerical rank used in any low-rank approximation step in constructing SIF and eSIF.
- l: total number of levels in SIF and eSIF.

When SIF and eSIF are constructed, we use the same parameters r, l, and finest level diagonal block size. Also in the construction of eSIF, one step of power iteration is used in randomized SVDs and the oversampling size is set to be 3. The preconditioner bdiag is constructed with the same diagonal block sizes as those of the finest level diagonal block sizes of SIF and eSIF. Just like in [35], all the test matrices are treated as general dense SPD matrices and are not specifically reordered.

EXAMPLE 1. We first test the methods on the matrix A with the (i, j) entry

$$A_{ij} = \frac{(ij)^{1/4}\pi}{20 + 0.8(i-j)^2},$$

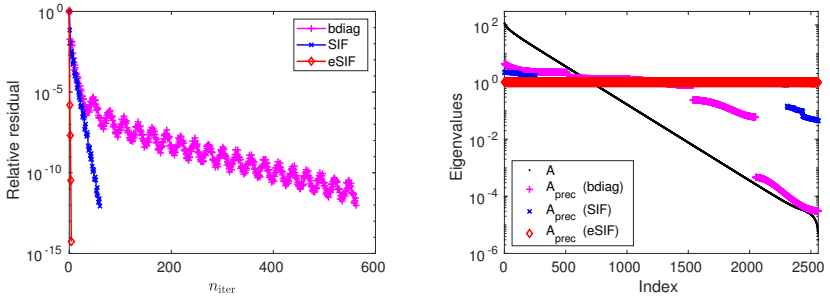
which is modified from a test example in [35] to make it more challenging.

In the construction of SIF and eSIF, we use $r = 5$. With the matrix size N increases, l increases accordingly for SIF and eSIF so that the finest level diagonal block size is fixed. Table 5.1 shows the results of PCG iterations to reach the tolerance 10^{-12} for the relative residual γ. Both SIF and eSIF help significantly reduce the condition numbers. The both make PCG converge much faster than using bdiag. eSIF is further much more effective than SIF and leads to $\kappa(A_{\text{prec}})$ close to 1. PCG with eSIF only needs few steps to reach the desired accuracy. The numbers of iterations are lower than with SIF by about 12 to 15 times.

TABLE 5.1
Example 1. *Convergence results of PCG with bdiag, SIF, and eSIF preconditioners. (For the two largest matrices, it is very slow to form A_{prec}, so the condition numbers are not computed.)*

N		1280	2560	5120	10,240	20,480	40,960
l		8	9	10	11	12	13
$\kappa(A)$		2.66e7	3.85e7	5.55e7	7.95e7		
$\kappa(A_{\text{prec}})$	bdiag	1.41e5	1.42e5	1.42e5	1.42e5		
	SIF	5.03e1	5.03e1	5.03e1	5.03e1		
	eSIF	1.01	1.01	1.02	1.02		
n_{iter}	bdiag	570	562	546	551	526	525
	SIF	57	60	61	60	60	60
	eSIF	4	4	4	4	4	5
γ	bdiag	9.65e−13	9.49e−13	9.50e−13	6.33e−13	7.89e−13	7.93e−13
	SIF	8.02e−13	8.42e−13	3.54e−13	9.36e−13	7.36e−13	8.28e−13
	eSIF	5.90e−15	5.48e−15	1.34e−13	4.28e−13	5.00e−14	9.61e−15

Figure 5.1(a) shows the actual convergence behaviors for one matrix and Figure 5.1(b) reflects how the preconditioners change the eigenvalue distributions. With eSIF, the eigenvalues of A_{prec} are all closely clustered around 1.

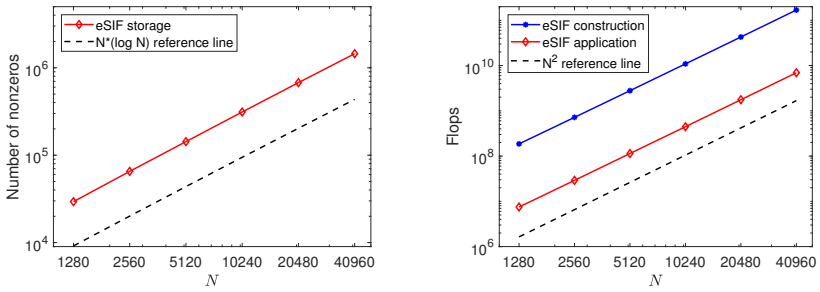


(a) Convergence (b) Eigenvalues

FIG. 5.1. Example 1. *Convergence of PCG with bdiag, SIF, and eSIF preconditioners and eigenvalues of the preconditioned matrices for $N = 2560$ in Table 5.1.*

To confirm the efficiency of eSIF, we plot the storage requirement of eSIF and the costs to construct and apply the preconditioner in each step. Since r is fixed, the storage of eSIF is $O(N \log N)$ and the construction and application costs are $O(N^2)$, which is confirmed in Figure 5.2. (Some tests for SIF can be found in [35]. We are

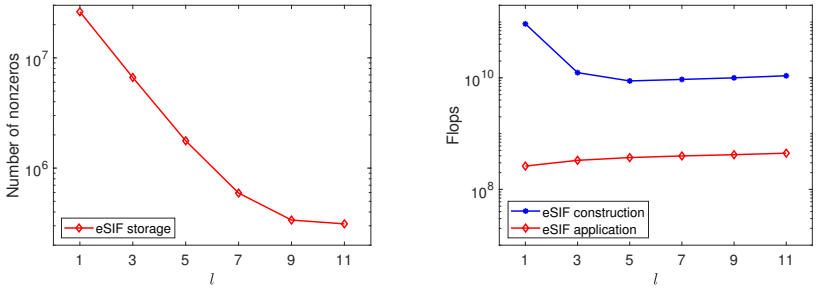
then not showing the results for SIF. In fact, here SIF has storage roughly comparable to that of eSIF.)



(a) Storage (b) Costs

FIG. 5.2. Example 1. *Storage for the structured factors of the eSIF preconditioner (excluding the storage for A) and the construction and application costs with varying N.*

To see how the efficiency is related to the number of levels l, we vary l for the matrix with size $N = 10240$. See Figure 5.3. A larger l leads to lower storage for the structured factors. When l is too small, the finest level diagonal blocks are large and it is costly to factorize these diagonal blocks and store the factors. When l increases, the cost for constructing the preconditioner decreases quickly at the beginning. The cost for applying the preconditioner slightly increases initially (since more levels need multiplications involving dense off-diagonal blocks of A), but then remains roughly steady (since the dominant cost is from higher levels). For larger l, the cost for the construction also becomes roughly steady. Thus, it makes sense to use relatively larger l so as to reduce the storage.



(a) Storage (b) Costs

FIG. 5.3. Example 1. *Storage for the structured factors of the eSIF preconditioner (excluding the storage for A) and the construction and application costs with varying l for the matrix with size* $N = 10240$.

EXAMPLE 2. In the second example, we consider to precondition some RBF (radial basis function) interpolation matrices which are known to be notoriously challenging for iterative methods due to the ill condition with some shape parameters (see, e.g., [8]). We consider the following four types of RBFs:

$$e^{-\varepsilon^2 t^2}, \quad \operatorname{sech} \varepsilon t, \quad \frac{1}{\sqrt{1+\varepsilon^2 t^2}}, \quad \frac{1}{1+\varepsilon^2 t^2},$$

where ε is the shape parameter. The interpolation matrices are obtained with grid points $0, 1, \ldots, N-1$.

We test the RBF interpolation matrices A with various different shape parameters. With $N = 1280$, $r = 6$, and $l = 8$, the performance of PCG to reach the tolerance 10^{-12} for γ is given in Table 5.2. When the shape parameter ε reduces, the condition numbers of the interpolation matrices increase quickly. SIF improves the condition numbers more significantly than bdiag. However, for smaller ε, the condition numbers resulting from both bdiag and SIF get much worse and the convergence of PCG slows down.

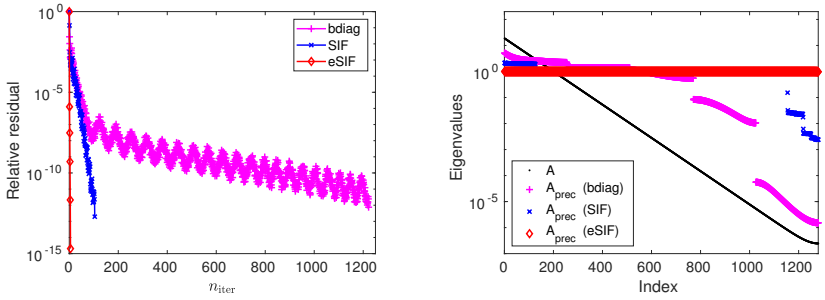
TABLE 5.2
Example 2. *Convergence results of PCG using* bdiag, SIF, *and* eSIF *preconditioners with* $r = 6$ *in* SIF *and* eSIF.

RBF		$e^{-\varepsilon^2 t^2}$			$\operatorname{sech} \varepsilon t$		
ε		0.4	0.36	0.32	0.3	0.25	0.2
$\kappa(A)$		2.49e6	9.27e7	1.46e10	3.48e6	9.34e7	1.30e10
$\kappa(A_{\text{prec}})$	bdiag	1.26e5	4.50e6	7.11e8	1.52e5	4.24e6	6.28e8
	SIF	2.38	2.11e3	2.14e6	1.34	5.02e2	7.58e5
	eSIF	1.00	1.00	1.00	1.00	1.00	1.30
n_{iter}	bdiag	700	2193	4482	547	1271	3211
	SIF	15	107	549	9	52	282
	eSIF	1	1	2	1	1	3
γ	bdiag	8.82e−13	8.62e−13	8.97e−13	7.97e−13	9.28e−13	8.25e−13
	SIF	4.94e−13	5.16e−13	9.86e−13	4.02e−13	9.44e−13	9.91e−13
	eSIF	6.16e−16	7.34e−15	2.63e−16	6.96e−15	1.85e−13	4.91e−14

RBF		$\frac{1}{\sqrt{1+\varepsilon^2 t^2}}$			$\frac{1}{1+\varepsilon^2 t^2}$		
ε		0.3	0.25	0.2	1/4	1/5	1/6
$\kappa(A)$		2.64e5	2.27e6	5.62e7	1.42e5	3.29e6	7.59e7
$\kappa(A_{\text{prec}})$	bdiag	1.15e4	9.64e4	2.40e6	6.18e3	1.41e5	3.34e6
	SIF	1.74	6.30	2.22e2	1.94	2.66e1	8.91e2
	eSIF	1.00	1.00	1.26	1.00	1.00	1.03
n_{iter}	bdiag	195	375	937	190	541	1222
	SIF	13	27	86	14	43	104
	eSIF	3	3	6	2	3	5
γ	bdiag	9.21e−13	7.19e−13	8.92e−13	9.84e−13	9.16e−13	7.52e−13
	SIF	4.23e−13	5.14e−13	6.20e−13	2.72e−13	7.15e−13	1.95e−13
	eSIF	1.77e−15	1.62e−15	8.16e−15	2.36e−13	5.58e−13	2.05e−15

On the other hand, eSIF performs significantly better for all the cases. Dramatic reductions in the numbers of iterations can be observed. In Table 5.2, the number of PCG iterations with eSIF is up to 274 times lower than with SIF and up to 2241 times lower than with bdiag. Overall, PCG with eSIF takes just few iterations to reach the desired accuracy.

Figure 5.4(a) shows the actual convergence behaviors for one case and Figure 5.4(b) illustrates how the preconditioners improve the eigenvalue distribution. Again, the eigenvalue clustering with eSIF is much better.



(a) Convergence (b) Eigenvalues

FIG. 5.4. Example 2. *Convergence of PCG and eigenvalues of the preconditioned matrices for the case with RBF* $\frac{1}{1+\varepsilon^2 t^2}$, $\varepsilon = \frac{1}{6}$ *in Table 5.2.*

We also try different numerical ranks r and the results are reported in Table 5.3. SIF is more sensitive to r. For some cases, SIF with $r = 4$ leads to quite slow convergence of PCG. In contrast, eSIF remains very effective for the different r choices and yields much faster convergence.

EXAMPLE 3. In the last example, we compare eSIF with SIF in terms of the following test matrices from different application backgrounds.

- `MHD3200B` ($N = 3200$, $\kappa(A) = 1.60e13$): The test matrix MHD3200B from the Matrix Market [25] treated as a dense matrix. $r = 9$ and $l = 8$ are used in the test.
- `ElasSchur` ($N = 3198$, $\kappa(A) = 8.91e6$): A dense Schur complement in the factorization of a discretized linear elasticity equation as used in [33]. The ratio of the so-called Lamé constants is 10^5. The original sparse discretized matrix has size $5,113,602$ and the Schur complement A corresponds to the last separator in the nested dissection ordering of the sparse matrix. $r = 5$ and $l = 9$ are used in the test.
- `LinProg` ($N = 2301$, $\kappa(A) = 2.09e11$): A test example in [35] from linear programming. The matrix is formed by $A = BDB^T$, where B is from the linear programming test matrix set Meszaros in [26] and D is a diagonal matrix with diagonal entries evenly located in $[10^{-5}, 1]$. $r = 3$ and $l = 9$ are used in the test.
- `Gaussian` ($N = 4000$, $\kappa(A) = 1.41e10$): a matrix of the form $sI + G$ with G from the discretization of the Gaussian kernel $e^{-\frac{\|t_i - t_j\|_2}{2\mu^2}}$. Such matrices frequently appear in applications such as Gaussian processes. Here, $s = 10^{-9}$, $\mu = 2.5$ and the t_i points are random points distributed in a long three dimensional rectangular parallelepiped. $r = 20$ and $l = 8$ are used in the test.

The convergence behaviors of PCG with SIF and eSIF preconditioners are given in Figure 5.5. Much faster convergence of PCG can be observed with eSIF. For the four matrices listed in the above order, the numbers of PCG iterations with SIF are about 11, 7, 7, and 21 times of those with eSIF, respectively.

6. Conclusions. We have presented an eSIF framework that enhances a recent SIF preconditioner in multiple aspects. During the construction of the preconditioner, two-sided block triangular preprocessing is followed by low-rank approximations in appropriate computations. Analysis of both the prototype preconditioner and the

TABLE 5.3
Example 2. *Convergence results of PCG using* SIF *and* eSIF *preconditioners with different* r.

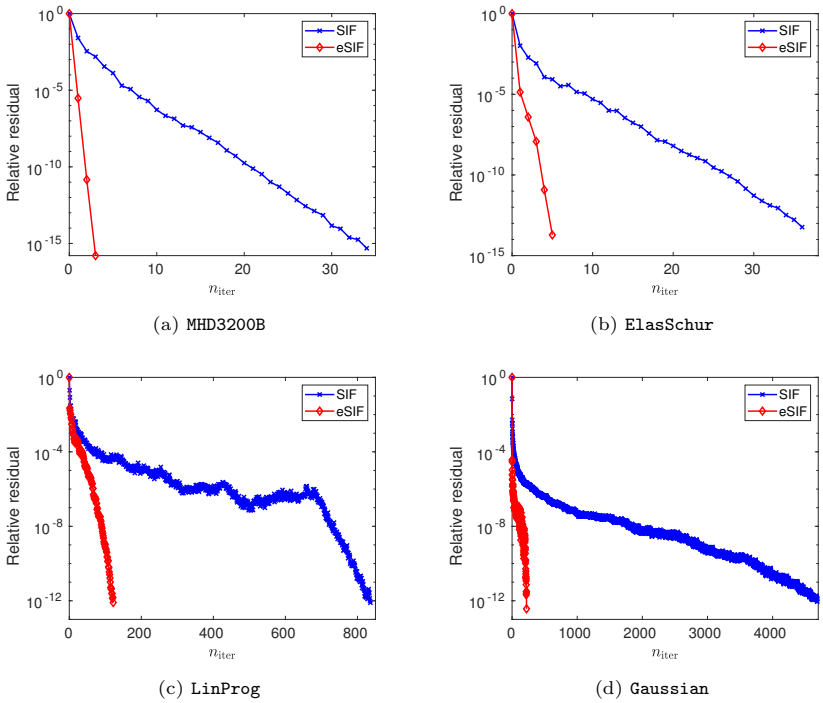
RBF			$e^{-\varepsilon^2 t^2}$			$\operatorname{sech} \varepsilon t$		
ε			0.3	0.25	0.2	1/4	1/5	1/6
$\kappa(A_{\text{prec}})$	$r=8$	SIF	1.01	2.35	3.64e4	1.00	1.23	4.80e3
		eSIF	1.00	1.00	1.00	1.00	1.00	1.06
	$r=4$	SIF	5.17e2	7.51e4	6.94e7	1.41e2	4.61e4	1.82e7
		eSIF	1.00	1.00	5.58	1.00	1.01	1.58e2
n_{iter}	$r=8$	SIF	5	13	245	4	7	69
		eSIF	1	1	1	1	1	2
	$r=4$	SIF	178	751	3972	92	410	1613
		eSIF	2	3	17	2	3	14
γ	$r=8$	SIF	7.95e−15	2.90e−13	4.95e−13	2.64e−15	3.18e−13	4.28e−13
		eSIF	6.89e−16	1.08e−15	1.23e−14	6.28e−15	1.85e−13	8.59e−13
	$r=4$	SIF	9.09e−13	9.42e−13	4.36e−11	8.11e−13	6.92e−13	6.06e−13
		eSIF	1.20e−15	4.63e−15	7.58e−13	9.14e−16	8.64e−14	6.33e−13

RBF			$\frac{1}{\sqrt{1+\varepsilon^2 t^2}}$			$\frac{1}{1+\varepsilon^2 t^2}$		
ε			0.3	0.25	0.2	1/4	1/5	1/6
$\kappa(A_{\text{prec}})$	$r=8$	SIF	1.39	3.66	1.06e2	1.45	6.32	6.21e1
		eSIF	1.00	1.00	1.00	1.00	1.00	1.00
	$r=4$	SIF	6.96e1	7.44e2	2.47e4	2.98	9.42e1	1.91e4
		eSIF	1.03	1.56	1.18	1.00	1.06	4.34
n_{iter}	$r=8$	SIF	10	19	75	11	27	64
		eSIF	2	2	2	2	2	3
	$r=4$	SIF	77	224	761	19	87	368
		eSIF	5	8	19	4	5	14
γ	$r=8$	SIF	9.73e−14	7.71e−13	4.63e−13	1.11e−13	2.50e−13	6.97e−13
		eSIF	1.78e−15	2.19e−14	1.09e−13	1.44e−15	3.02e−15	1.95e−15
	$r=4$	SIF	5.93e−13	9.84e−13	9.21e−13	4.81e−13	9.20e−13	5.71e−13
		eSIF	8.38e−14	9.19e−13	1.87e−13	3.84e−15	2.67e−13	1.05e−13

practical multilevel extension is given. We are able to not only overcome a major bottleneck of potential loss of positive definiteness in the SIF scheme but also significantly improve the accuracy bounds, condition numbers, and eigenvalue distributions. Thorough comparisons in terms of the analysis and the test performance are given.

In our future work, we expect to explore new preprocessing and approximation strategies that can further improve the eigenvalue clustering and accelerate the decay magnification effect in the condition number. The current work successfully improves the relevant accuracy, condition number, and eigenvalue bounds by a significant amount (e.g., from $\frac{1+\hat{\epsilon}}{1-\hat{\epsilon}}$ to $1+\epsilon$ in Table 3.1 with ϵ much smaller than $\hat{\epsilon}$). We expect to further continue this trend and in the meantime keep the preconditioners convenient to apply. We will also explore the feasibility of extending our ideas to nonsymmetric and indefinite matrices.

Acknowledgements. Thank the two anonymous referees for providing useful suggestions that help improve this paper.



(a) MHD3200B (b) ElasSchur

(c) LinProg (d) Gaussian

FIG. 5.5. Example 3. *Convergence of PCG with SIF and eSIF preconditioners.*

REFERENCES

[1] E. AGULLO, E. DARVE, L. GIRAUD, AND Y. HARNESS, *Low-rank factorizations in data sparse hierarchical algorithms for preconditioning symmetric positive definite matrices*, SIAM J. Matrix Anal. Appl. 39 (2018), pp. 1701–1725.

[2] P. AMESTOY, C. ASHCRAFT, O. BOITEAU, A. BUTTARI, J.-Y. L'EXCELLENT, AND CLÉMENT WEISBECKER, *Improving multifrontal methods by means of block low-rank representations*, SIAM J. Sci. Comput., 37 (2015), pp. A1451–A1474.

[3] O. AXELSSON AND L. KOLOTILINA, *Diagonally compensated reduction and related preconditioning methods*, Numer. Linear Algebra Appl., 1 (1994), pp. 155–177.

[4] M. BENZI, J. K. CULLUM, AND M. TŮMA, Robust approximate inverse preconditioning for the conjugate gradient method, SIAM J. Sci. Comput., 22 (2000), pp. 1318–1332.

[5] M. BENZI AND M. TŮMA, A robust incomplete factorization preconditioner for positive definite matrices, Numer. Linear Algebra Appl., 10 (2003), pp. 385–400.

[6] S. BÖRM AND W. HACKBUSCH, *Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices*, Computing, 69 (2002), pp. 1–35.

[7] S. CHANDRASEKARAN, P. DEWILDE, M. GU, AND T. PALS, *A fast ULV decomposition solver for hierarchically semiseparable representations*, SIAM J. Matrix Anal. Appl., 28 (2006), pp. 603–622.

[8] 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.

[9] L. CAMBIER, C. CHEN, E. G. BOMAN, S. RAJAMANICKAM, R. S. TUMINARO, AND E. DARVE, *An algebraic sparsified nested dissection algorithm using low-rank approximations*, SIAM J. Matrix Anal. Appl., 41 (2020), pp. 715–746.

[10] G. CHÁVEZ, Y. LIU, P. GHYSELS, X. S. LI, AND E. REBROVA, *Scalable and memory-efficient kernel ridge regression*, 2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pp. 956–965.

[11] Z. DRMAC, M. OMLADIC, AND K. VESELIC, *On the perturbation of the Cholesky factorization*, SIAM J. Matrix Anal. Appl., 15 (1994), pp. 1319–1332.

[12] J. Feliu-Fabà, K. L. Ho, and L. Ying, *Recursively preconditioned hierarchical interpolative factorization for elliptic partial differential equations*, Commun. Math. Sci., 18 (2020), pp. 91–108.
[13] 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.
[14] W. Hackbusch, *A Sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices*, Computing 62 (1999), pp. 89–108.
[15] N. Halko, P.G. Martinsson, and J. Tropp, *Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions*, SIAM Review, 53 (2011), pp. 217–288.
[16] D. S. Kershaw, *The incomplete Cholesky–conjugate gradient method for the iterative solution of systems of linear equations*, J. Comput. Phys., 26 (1978), pp. 43–65.
[17] B. Klockiewicz, L. Cambier, R. Humble, H. Tchelepi, and E. Darve, *Second order accurate hierarchical approximate factorization of sparse SPD matrices*, arXiv:2007.00789, (2020).
[18] R. Li and Y. Saad, *Divide and conquer low-rank preconditioners for symmetric matrices*, SIAM J. Sci. Comput., 35 (2013); pp. A2069–A2095.
[19] R. Li and Y. Saad, *Low-rank correction methods for algebraic domain decomposition preconditioners*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 807–828.
[20] R. Li, Y. Xi, and Y. Saad, *Schur complement based domain decomposition preconditioners with low-rank corrections*, Numer. Linear Algebra Appl., 23 (2016), pp. 706–729.
[21] S. Li, M. Gu, C. Wu, and J. Xia, *New efficient and robust HSS Cholesky factorization of SPD matrices*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 886–904.
[22] E. Liberty, F. Woolfe, P. G. Martinsson, V. Rokhlin, and M. Tygert, *Randomized algorithms for the low-rank approximation of matrices*, Proc. Natl. Acad. Sci. USA, 104 (2007), pp. 20167–20172.
[23] X. Liu, J. Xia, and M. V. de Hoop, *Parallel randomized and matrix-free direct solvers for large structured dense linear systems*, SIAM J. Sci. Comput., 38 (2016), S508–S538.
[24] F.-H. Rouet, X. S. Li, P. Ghysels, and A. Napov, *A distributed-memory package for dense hierarchicallysemi-separable matrix computations using randomization*, ACM Trans. Math. Software, 42 (2016).
[25] *The Matrix Market*, https://math.nist.gov/MatrixMarket.
[26] *The SuiteSparse Matrix Collection*, http://faculty.cse.tamu.edu/davis/suitesparse.html.
[27] S. Wang. X. S. Li, F. H. Rouet, J. Xia, and M. V. de Hoop, *A parallel geometric multifrontal solver using hierarchically semiseparable structure*, ACM Trans. Math. Software, 42 (2016), Article 21.
[28] Y. Xi, R. Li, and Y. Saad, *An algebraic multilevel preconditioner with low-rank corrections for sparse symmetric matrices*, SIAM J. Matrix Anal. Appl., 37 (2016), pp. 235–259.
[29] Y. Xi, J. Xia, and R. Chan, *A fast randomized eigensolver with structured LDL factorization update*, SIAM J. Matrix Anal. Appl., 35 (2014), pp. 974-996.
[30] J. Xia, *On the complexity of some hierarchical structured matrix algorithms*, SIAM J. Matrix Anal. Appl., 33 (2012), pp. 388-410.
[31] J. Xia, *Robust and effective eSIF preconditioning for general SPD matrices*, arXiv:2007.03729, (2020).
[32] J. Xia, *Effective eSIF Preconditioners with Guaranteed Positive Definiteness for General SPD Matrices*, Presentation in the 2019 Preconditioning Conference.
[33] 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.
[34] 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.
[35] J. Xia and Z. Xin, *Effective and robust preconditioning of general SPD matrices via structured incomplete factorization*, SIAM J. Matrix Anal. Appl., 38 (2017), pp. 1298–1322.
[36] Z. Xin, J. Xia, S. Cauley, and V. Balakrishnan, *Effectiveness and robustness revisited for a preconditioning technique based on structured incomplete factorization*, Numer. Linear Algebra Appl., 27 (2020), e2294.
[37] X. Xing and E. Chow, *Preserving positive definiteness in hierarchically semiseparable matrix approximations*, SIAM J. Matrix Anal. Appl., 39 (2018), pp. 829–855.
[38] X. Xing, H. Huang, and E. Chow, *Efficient construction of an HSS preconditioner for symmetric positive definite $\mathcal{H}^2$ matrices*, arXiv:2011.07632, (2020).
[39] I. Yamazaki, A. Ida, R. Yokota, and J. Dongarra, *Distributed-memory lattice $\mathcal{H}$-matrix factorization*, The International Journal of High Performance Computing Applications, 33 (2019), pp. 1046–1063.