ROBUST STRUCTURED MULTIFRONTAL FACTORIZATION AND PRECONDITIONING FOR DISCRETIZED PDES

JIANLIN XIA*

Abstract. We present an approximate structured factorization method which is efficient, robust, and also relatively insensitive to ill conditioning, high frequencies, or wavenumbers for some discretized PDEs. Given a sparse symmetric positive definite discretized matrix A, we compute a structured approximate factorization $A \approx \mathcal{L}\mathcal{L}^T$ with a desired accuracy, where $\mathcal{L}$ is lower triangular and data sparse. This can be used in direct solution or preconditioning for linear systems. The method uses the idea that during the direct factorization of some discretized matrices, certain dense intermediate matrices have a low-rank property, or, their off-diagonal blocks can be approximated by compact low-rank matrices. In this paper, we organize the factorization with a supernodal version multifrontal method using nested dissection ordering of the matrix. Each dense intermediate matrix is formed explicitly and then partially factorized so that the leading factor is a hierarchically semiseparable matrix and the Schur complement remains dense. The use of explicit dense matrices makes the method much simpler than existing structured factorization methods. The overall factor remains structured and can be used to solve systems in nearly linear complexity. The factorization algorithm costs $O(rn\log_2 n)$, where n is the matrix size, and r is a parameter related to the tolerance and the problem. Schur complements are automatically compensated during the factorization so that $\mathcal{L}\mathcal{L}^T$ always exists for any accuracy and has enhanced positive definiteness. No extra stablization is needed. The method also works well as a preconditioner even if the low-rank property is not highly significant. We demonstrate the reliability and effectiveness of the method with various applications, including elliptic problems, linear elasticity equations, Helmholtz equations, Maxwell equations, etc.

Key words. supernodal multifrontal method, robust approximate factorization, Schur compensation, HSS structure, low-rank property

AMS subject classifications. 15A23, 65F05, 65F30

1. Introduction. Large sparse linear systems arise from numerical discretizations of PDEs in practical problems. Typically, there are two types of linear system solvers, direct methods and iterative methods. Direct methods are reliable and are efficient for multiple right-hand sides, but are often expensive due to the generation of fill-in or loss of sparsity. Iterative methods take good advantage of sparsity and require less storage, but may diverge or converge slowly without effective preconditioners. Classical preconditioners such as incomplete factorization or orthogonalization methods can break down due to numerical instability.

In this paper, we are interested in reliable and fast approximate factorizations of large discretized matrices. These factorizations can be used as direct solvers with specified accuracy, or as preconditioners. Assume we have a system

$$Ax = b \tag{1.1}$$

arising from the discretization of certain PDEs, where A is a symmetric positive definite (SPD) matrix. In direct solutions of the system, we compute the Cholesky factorization $A = \mathcal{L}\mathcal{L}^T$. The matrix A is usually sparse. However, the factorization may create nonzeros in $\mathcal{L}$ which are called fill-in. One way to reduce fill-in is to reorder the rows and columns of A, or to reorder the mesh points used in the discretization. For example, for an $N \times N$ regular mesh, the factorization with a natural rowwise or columnwise ordering needs $O(n^2)$ flops and $O(n^{3/2})$ storage space, where $n = N^2$ is the order of A. On the other hand, the nested dissection ordering [23] and

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

its generalizations lead to $O(n^{3/2})$ complexity factorizations with $O(n \log n)$ storage. For two-dimensional (2D) problems, these are shown to be lower bounds for exact factorizations with any ordering [30] (ignoring special techniques such as Strassen's algorithm [41]). Nested dissection uses separators to recursively divide the mesh into subregions. Mesh points are put into separators at different levels of the elimination.

On the other hand, some iterative methods such as multigrid converges with $O(n)$ complexity for some problems. In this work, we compute structured approximate factorizations of A based on a rank property of the problems. It has been indicated in [3, 4, 7, 25, 26, 44, etc.] that for some PDEs such as elliptic equations, during the direct factorization of A, the fill-in has a low-rank property, or, the off-diagonal blocks have small numerical ranks. This property can be used to improve the efficiency of the factorization. Rank structured matrices such as quasiseparable or semiseparable matrices [22, 43] can be used to approximate the dense intermediate matrices.

In fact, we can also simply form the dense intermediate matrices first and then approximately factorize them into rank structured matrices. In this paper, we organize the factorization with a supernodal multifrontal method together with the nest dissection ordering of mesh points. The multifrontal method is a very important direct methods for sparse matrix solutions [21, 33]. The multifrontal method keeps the propagation of information locally between nodes and their parents, even if nearby mesh points may locate at different levels of the elimination process after reordering. The supernodal version multifrontal method we use has nice data locality and takes good advantage of dense matrix operations. The factorization follows an elimination tree of the separators. The intermediate matrices are called frontal matrices and update matrices. A frontal matrix is formed before the elimination of each separator and carries the information of the separator and its upper level neighbors. An update matrix is the Schur complement after the elimination of a separator from the frontal matrix. If the problem has the low-rank property, the frontal matrices can be approximated with rank structured matrices such as semiseparable matrices. This is what [44] does. [44] uses semiseparable matrices in all stages of the structured factorization. However, this makes the assembly of structured matrices extremely complicated. The algorithm is not straightforward to understand and careful implementation is needed to ensure high efficiency.

Here, a dense frontal matrix corresponding to each separator is formed first and then approximately factorized into structured form. After the factorization, the Schur complement matrix (update matrix) is still a regular dense matrix. Thus, standard extend-add is used. That is, all frontal and update matrices are in dense forms, but the factors are structured. This makes the algorithm much simpler than the one in [44].

Furthermore, the algorithm in [44] may suffer from the problem of breakdown, or the lose of positive definiteness, especially when a large tolerance is used. Here, when directly factorizing dense frontal matrices, we use a robustness technique where Schur complements are automatically compensated. This compensation is done implicitly during the approximation of off-diagonal blocks. Thus, the low-rank approximation improves not only the efficiency but also the reliability. The total factorization cost of the new algorithm is only $O(rn \log n)$, where r is the maximum off-diagonal numerical rank. This has an extra $\log n$ factor compared to the complexity of the algorithm in [44]. However, the computation time is still very competitive due to the dense block operations.

In addition, even for problems where the low-rank property is not very signifi-

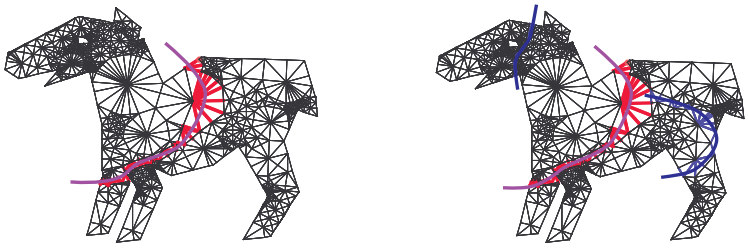
cant, our approximate factorization with a relatively large tolerance can still be used as a preconditioner whose effectiveness can be illustrated by numerical tests and preliminary analysis. That is, the factorization does not need strict low-rank structural requirement to be effective.

The semiseparable structure we use for the factors is a tree structured hierarchically semiseparable (HSS) matrix [8, 9, 14]. Thus, the overall Cholesky factor is given by two layers of tree structures, an outer elimination tree of separators, and an inner HSS tree corresponding to each separator. The complexity for solving a linear system with such a structured factor is nearly linear.

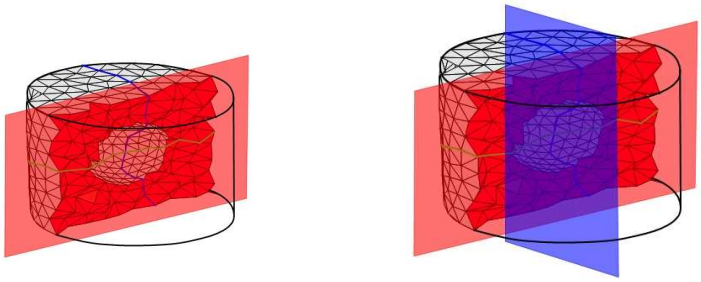
The rest of this paper is organized as follows. Section 2 reviews the nested dissection ordering and the multifrontal method. Section 4 presents the main factorization algorithm and its analysis. Numerical tests are given in Section 5. Problems such as elliptic problems, linear elasticity near incompressible limit, Helmholtz equations, Maxwell equations are considered.

2. Review of nested dissection ordering and multifrontal method.

2.1. Nested dissection ordering. Consider the mesh used in the discretization as an undirected graph $G = (V, E)$. Each vertex $i \in V$ corresponds to a row and a column of A, and each edge $(i, j) \in E$ corresponds to entries $a_{ij} = a_{ji} \neq 0$. Nested dissection recursively divides the mesh into subregions with separators [23]. At the first level, a separator divides the entire grid into two subregions (plus the separator itself). The two subregions are further divided recursively. See Figure 2.1.



Two levels of partition in 2D. Image based on MESHPART [24]



Three levels of partition in 3D. Image based on DISTMESH [**?**]

FIG. 2.1. *Nested dissection ordering for irregular meshes.*

Lower level separators are ordered before upper level ones. During the factorization of A, the elimination of a mesh point mutually connects points which are previously connected to it [39, 42]. This creates fill-in. For simplicity of presentations, we look at an $N \times N$ regular mesh such as the one in Figure 2.2. After reordering, the matrix A has nonzero pattern as shown in Figure 2.3. Then clearly, the nodes in the

top level separator are mutually connected and form a dimension $O(N)$ dense matrix whose exact factorization costs at least $O(N^3)$ flops.

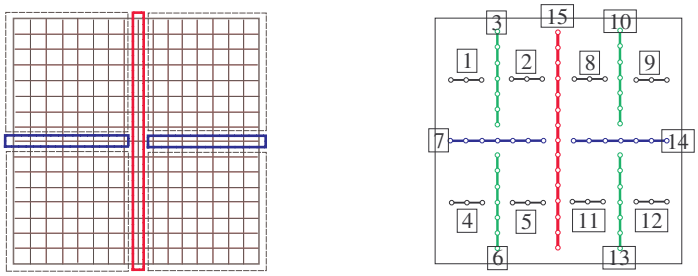


FIG. 2.2. *Separators in nested dissection.*

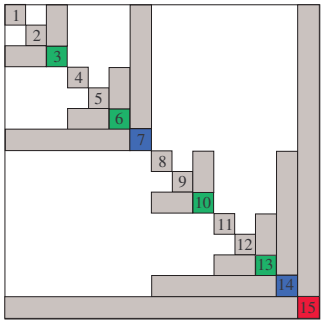


FIG. 2.3. *Nonzero pattern of A after the nested dissection ordering of the rows and columns.*

2.2. Multifrontal method. The multifrontal method reorganizes the factorization $A = \mathcal{L}\mathcal{L}^T$ into partial factorizations of intermediate dense matrices [21, 33]. The factorization is conducted following a tree called elimination tree. In general, an elimination tree $\mathcal{T}$ has n nodes and a node p is the parent of j if and only if

$$p = \min\{i > j | l_{ij} \neq 0\}.$$

Use $\mathcal{T}[j]$ to denote the subtree of $\mathcal{T}$ with root j. In the elimination, the contributions from child nodes are passed to parents in terms of outer products. Define the subtree update matrix corresponding to j as

$$\bar{\mathcal{U}}_j = - \sum_{k \in \mathcal{T}[j]-j} \mathcal{L}_{\{i_0,i_1,\ldots,i_r\},k} \mathcal{L}^T_{\{i_0,i_1,\ldots,i_r\},k},$$

where $\mathcal{L}_{\{i_0,i_1,\ldots,i_r\},k}$ is a column vector of all the nonzero entries in $\mathcal{L}_{:,k}$, the kth column of $\mathcal{L}$ and $\{i_0, i_1, \ldots, i_r\}$ is the set of row indices of the nonzero entries. The jth frontal matrix is defined to be

$$\mathcal{F}_j = \begin{pmatrix} a_{j,j} & A^T_{\{i_1,i_2,\ldots,i_r\},j} \\ A_{\{i_1,i_2,\ldots,i_r\},j} & 0 \end{pmatrix} + \bar{\mathcal{U}}_j.$$

One step of elimination applied to F_j provides the column $\mathcal{L}_{\{i_0,i_1,\ldots,i_r\},j}$

$$\mathcal{F}_j = \begin{pmatrix} l_{j,j} & 0 \\ \mathcal{L}_{\{i_1,i_2,\ldots,i_r\},j} & I \end{pmatrix} \begin{pmatrix} l_{j,j} & \mathcal{L}^T_{\{i_1,i_2,\ldots,i_r\},j} \\ 0 & U_j \end{pmatrix},$$

where U_j is the contribution from the $T[j]$ to the parent of j and is called the jth update matrix. U_j has the form

$$\mathcal{U}_j = -\sum_{k\in\mathcal{T}[j]} \mathcal{L}_{\{i_1,i_2,\ldots,i_r\},k}\mathcal{L}^T_{\{i_1,i_2,\ldots,i_r\},k}.$$

In the multifrontal method, lower level frontal matrices are formed and partially eliminated to provide some columns of L and the update matrices. The update matrices are then used to form upper level partial update matrices and thus upper level frontal matrices. This process is called extend-add, denoted

$$\mathcal{F}_j = \begin{pmatrix} a_{j,j} & A^T_{\{i_1,i_2,\ldots,i_r\},j} \\ A_{\{i_1,i_2,\ldots,i_r\},j} & 0 \end{pmatrix} \Updownarrow \mathcal{U}_{c_1} \Updownarrow \mathcal{U}_{c_2} \Updownarrow \cdots \Updownarrow \mathcal{U}_{c_q},$$

where nodes $c_1, c_2, \ldots, c_q$ are the children of j in the elimination tree. The process then repeats along the elimination tree.

3. Dense structured preconditioning. We introduce the preconditioner using block 2×2 cases. Generalizations to more blocks are shown in the remaining sections. Assume an $n \times n$ matrix A has the following partition:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{matrix} m \\ n-m \end{matrix}.$$

Without loss of generality, we assume $m \le n/2$, and any appropriate inverse in the following exists. First, compute an LU factorization $A_{11} = \mathbf{L}_{11}\mathbf{R}_{11}$, where $\mathbf{L}_{11}$ and $\mathbf{R}_{11}$ and lower and upper triangular matrices, respectively. The traditional block Cholesky factorization of A proceeds as

$$A = \begin{pmatrix} \mathbf{L}_{11} & 0 \\ \mathbf{L}_{21} & \mathbf{L}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ 0 & \mathbf{R}_{22} \end{pmatrix} \equiv \mathbf{LR}, \tag{3.1}$$

where $\mathbf{R}_{12} = \mathbf{L}_{11}^{-1}A_{12}$, $\mathbf{L}_{21} = A_{21}\mathbf{R}_{11}^{-1}$, and $\mathbf{L}_{22}\mathbf{R}_{22}$ is the LU factorization of the Schur complement

$$A^{(1)} = A_{22} - \mathbf{L}_{21}\mathbf{R}_{12}. \tag{3.2}$$

In our approximate factorization, before $\mathbf{R}_{22}$ is computed, we first compute the (τ-accurate) SVD of the matrix $\Omega \equiv \begin{pmatrix} \mathbf{R}_{12} & \mathbf{L}_{21}^T \end{pmatrix}$

$$\Omega = \begin{pmatrix} U & \hat{U} \end{pmatrix} \begin{pmatrix} \Sigma & \\ & \hat{\Sigma} \end{pmatrix} \begin{pmatrix} V^T \\ \hat{V}^T \end{pmatrix} = U\Sigma V^T + \hat{U}\hat{\Sigma}\hat{V}^T = U\Sigma V^T + O(\tau), \tag{3.3}$$

where τ is a tolerance, and $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_r)$, $\hat{\Sigma} = \mathrm{diag}(\sigma_{r+1}, \ldots, \sigma_m)$ with

$$\sigma_1 \ge \cdots \ge \sigma_r \ge \tau \ge \sigma_{r+1} \ge \cdots \ge \sigma_m. \tag{3.4}$$

Partition $V^T = \begin{pmatrix} V_1^T & V_2^T \end{pmatrix}$ and $\hat{V}^T = \begin{pmatrix} \hat{V}_1^T & \hat{V}_2^T \end{pmatrix}$ following the column partition of $\begin{pmatrix} \mathbf{R}_{12} & \mathbf{L}_{21}^T \end{pmatrix}$. The Schur complement (4.4) is now

$$\begin{aligned} A^{(1)} &= A_{22} - (U\Sigma V_2^T + \hat{U}\hat{\Sigma}\hat{V}_2^T)^T(U\Sigma V_1^T + \hat{U}\hat{\Sigma}\hat{V}_1^T) \\ &= A_{22} - V_2\Sigma^2 V_1^T - \hat{V}_2\hat{\Sigma}^2\hat{V}_1^T. \end{aligned}$$

With the approximation $\Omega \approx U\Sigma V^T$, the Schur complement $A^{(1)}$ is approximated by

$$\tilde{A}^{(1)} = A_{22} - V_2\Sigma^2 V_1^T \tag{3.5}$$

Next, compute the LU factorization $\tilde{A}^{(1)} = \tilde{\mathbf{L}}_{22}\tilde{\mathbf{R}}_{22}$ and obtain an approximate factorization of A

$$A \approx \tilde{\mathbf{L}}\tilde{\mathbf{R}}, \quad \tilde{\mathbf{L}} = \begin{pmatrix} \mathbf{L}_{11} & \\ V_2\Sigma U^T & \tilde{\mathbf{L}}_{22} \end{pmatrix}, \ \tilde{\mathbf{R}} = \begin{pmatrix} \mathbf{R}_{11} & U\Sigma V_1^T \\ 0 & \tilde{\mathbf{R}}_{22} \end{pmatrix}. \tag{3.6}$$

We then study the effectiveness of using $\tilde{\mathbf{L}}\tilde{\mathbf{R}}$ as a preconditioner for A by considering the condition number of the preconditioned matrix $\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{R}}^{-1} = \tilde{\mathbf{L}}^{-1}\mathbf{L}\mathbf{R}\tilde{\mathbf{R}}^{-1}$.

LEMMA 3.1. *Let $\tilde{\mathbf{L}}\tilde{\mathbf{R}}$ in (3.6) be a preconditioner for A in (3.1). Then preconditioned matrix is*

$$\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{R}}^{-1} = \begin{pmatrix} I & C_1 \\ C_2 & I \end{pmatrix},$$

where $C_1 = \hat{U}\hat{\Sigma}\hat{V}_1^T\tilde{\mathbf{R}}_{22}^{-1}$ and $C_2 = \tilde{\mathbf{L}}_{22}^{-1}\hat{V}_2\hat{\Sigma}\hat{U}^T$.

Proof. Clearly,

$$\mathbf{R}\tilde{\mathbf{R}}^{-1} = \begin{pmatrix} \mathbf{R}_{11} & U\Sigma V_1^T + \hat{U}\hat{\Sigma}\hat{V}_1^T \\ 0 & \mathbf{R}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{R}_{11} & U\Sigma V_1^T \\ 0 & \tilde{\mathbf{R}}_{22} \end{pmatrix}^{-1} = \begin{pmatrix} I & \hat{U}\hat{\Sigma}\hat{V}_1^T\tilde{\mathbf{R}}_{22}^{-1} \\ 0 & \mathbf{R}_{22}\tilde{\mathbf{R}}_{22}^{-1} \end{pmatrix},$$

and $\tilde{\mathbf{L}}^{-1}\mathbf{L} = \begin{pmatrix} I & 0 \\ \tilde{\mathbf{L}}_{22}^{-1}\hat{V}_2\hat{\Sigma}\hat{U}^T & \tilde{\mathbf{L}}_{22}^{-1}\mathbf{L}_{22} \end{pmatrix}$. Thus,

$$\begin{aligned} \tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{R}}^{-1} &= \begin{pmatrix} I & 0 \\ \tilde{\mathbf{L}}_{22}^{-1}\hat{V}_2\hat{\Sigma}\hat{U}^T & \tilde{\mathbf{L}}_{22}^{-1}\mathbf{L}_{22} \end{pmatrix} \begin{pmatrix} I & \hat{U}\hat{\Sigma}\hat{V}_1^T\tilde{\mathbf{R}}_{22}^{-1} \\ 0 & \mathbf{R}_{22}\tilde{\mathbf{R}}_{22}^{-1} \end{pmatrix} \\ &= \begin{pmatrix} I & \hat{U}\hat{\Sigma}\hat{V}_1^T\tilde{\mathbf{R}}_{22}^{-1} \\ \tilde{\mathbf{L}}_{22}^{-1}\hat{V}_2\hat{\Sigma}\hat{U}^T & \tilde{\mathbf{L}}_{22}^{-1}(\hat{V}_2\hat{\Sigma}^2\hat{V}_1^T + \mathbf{L}_{22}\mathbf{R}_{22})\tilde{\mathbf{R}}_{22}^{-1} \end{pmatrix}. \end{aligned}$$

Noticing (3.5), we have

$$\tilde{\mathbf{L}}_{22}^{-1}(\hat{V}_2\hat{\Sigma}^2\hat{V}_1^T + \mathbf{L}_{22}\mathbf{R}_{22})\tilde{\mathbf{R}}_{22}^{-1} = \tilde{\mathbf{L}}_{22}^{-1}\tilde{A}^{(1)}\tilde{\mathbf{R}}_{22}^{-1} = I.$$

□

THEOREM 3.2. *The 2-norm condition number $\kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{R}}^{-1})$ of the preconditioned matrix satisfies*

$$\kappa(\tilde{\mathbf{L}}^{-1}A\tilde{\mathbf{R}}^{-1})$$

Since

$$\tilde{\mathbf{R}}^T\tilde{\mathbf{R}} = A + \begin{pmatrix} 0 & 0 \\ 0 & \hat{V}\hat{\Sigma}^2\hat{V}^T \end{pmatrix} = A + \begin{pmatrix} 0 & 0 \\ 0 & O(\tau^2) \end{pmatrix}, \tag{3.7}$$

the approximate matrix $\tilde{\mathbf{R}}^T\tilde{\mathbf{R}}$ is guaranteed to be positive definite. The matrix $\tilde{\mathbf{R}}$ can be used as a preconditioner and the preconditioned matrix $\tilde{A} = \tilde{\mathbf{R}}^{-T}A\tilde{\mathbf{R}}^{-1}$ has a form in the following result.

4. Structured supernodal multifrontal factorization.

4.1. Supernodal multifrontal method with nested dissection. We first generalize a supernodal version multifrontal method used in [44] for sparse problems.

In the classical multifrontal method, each node of the elimination tree represents one row or column of the matrix. On the other hand, supernodal factorization allows dense vector operations and reduces symbolic overhead. Here we use nested dissection to order the mesh points. See Figure 2.2. The separators are then treated as nodes in the elimination tree. Figure 4.1 shows the elimination tree for the mesh in Figure 2.2.

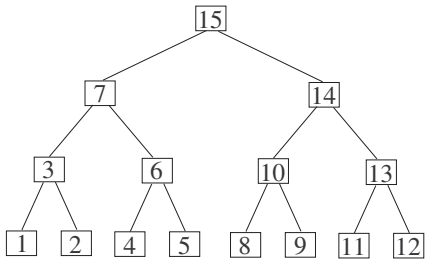


FIG. 4.1. *Supernodal version elimination tree for the mesh in Figure 2.2*

Typically, when the bottom level separator sizes are specified, we can decide a binary tree as the elimination tree of separators. Without loss of generality, we can use a full binary tree, that is, each node other than the leaves has exactly two children. In fact, we have

THEOREM 4.1. *For any odd number $2k-1$ with $k \in \mathbb{N}$, there always exists a full binary tree with totally $2k-1$ nodes, and k of them are leaf nodes.*

Proof. Induction or construction. ∎

Let $\{i_1, i_2, \ldots, i_r\}$ be the set of neighbor separators of separator j at the same or upper levels of j. The frontal matrix $\mathcal{F}_j$ has the following form:

$$\mathcal{F}_j = \begin{pmatrix} F_{j,j} & F^T_{\{i_1,i_2,\ldots,i_r\},j} \\ F_{\{i_1,i_2,\ldots,i_r\},j} & F_{\{i_1,i_2,\ldots,i_r\},\{i_1,i_2,\ldots,i_r\}} \end{pmatrix} \equiv \begin{pmatrix} F_{j,j} & F^T_{i_1,j} & \cdots & F^T_{i_r,j} \\ F_{i_1,j} & F_{i_1,i_1} & \cdots & F^T_{i_r,i_1} \\ \vdots & \vdots & \ddots & \vdots \\ F_{i_r,j} & F_{i_r,i_1} & \cdots & F_{i_r,i_r} \end{pmatrix}.$$

Let c_1 and c_2 be the children of j. $\mathcal{F}_j$ is formed by the block form extend-add

$$\mathcal{F}_j = \begin{pmatrix} A_{j,j} & A^T_{\{i_1,i_2,\ldots,i_r\},j} \\ A_{\{i_1,i_2,\ldots,i_r\},j} & 0 \end{pmatrix} \updownarrow \mathcal{U}_{c_1} \updownarrow \mathcal{U}_{c_2}, \tag{4.1}$$

where we call the first matrix on the righthand side the jth initial frontal matrix. The major operations include, forming the frontal matrix $\mathcal{F}_j$ by extend-add (when j is a leaf node, no extend-add is necessary), eliminating separator j or factorizing $F_{j,j}$, and forming $\mathcal{U}_j$ or computing the Schur complement of $F_{j,j}$. All these are done in block form.

4.2. Semiseparable structures. It has been shown that during the factorization of the discretized matrices arising from some problems such as elliptic PDEs, the off-diagonal of the Schur complements have bounded numerical ranks independent of

mesh sizes [3, 4, 7]. The Green's functions for these problems are smooth away from the diagonal singularity. As discussed in [44], when the factorization is conducted with the supernodal multifrontal method, the frontal and update matrices then have this low-rank property also, that is, the off-diagonal blocks of the frontal and update matrices have small numerical ranks, with approximate definitions of off-diagonal blocks. Therefore, in [44], all the frontal and update matrices are approximated by semiseparable matrices and the extend-add operation (4.1) is replaced by a structured one. The semiseparable structure used in [44] is called hierarchically semiseparable (HSS) structure [8, 9, 14]. A block 4×4 HSS matrix looks like

$$\begin{pmatrix} D_1 & U_1B_1V_2^T & U_1R_1B_3W_4^TV_4^T & U_1R_1B_3W_5^TV_5^T \\ U_2B_2V_1^T & D_2 & U_2R_2B_3W_4^TV_4^T & U_2R_2B_3W_5^TV_5^T \\ U_4R_4B_6W_1^TV_1^T & U_4R_4B_6W_2^TV_2^T & D_4 & U_4B_4V_5^T \\ U_5R_5B_6W_1^TV_1^T & U_5R_5B_6W_2^TV_2^T & U_5B_5V_4^T & D_5 \end{pmatrix}, \tag{4.2}$$

where the matrices $D_i, U_i, B_i, \ldots$ are called generators. (4.2) is a simplified HSS form in [14] based on the original form. An HSS matrix has a natural multi-level tree structure called HSS tree. Figure 4.2 shows the HSS tree for (4.2) and the generators in (4.2) are indexed according to the postordering of the tree nodes. Upper level generators are defined in terms of lower level ones. For example, $U_3 = \begin{pmatrix} U_1R_1 \\ U_2R_2 \end{pmatrix}$.

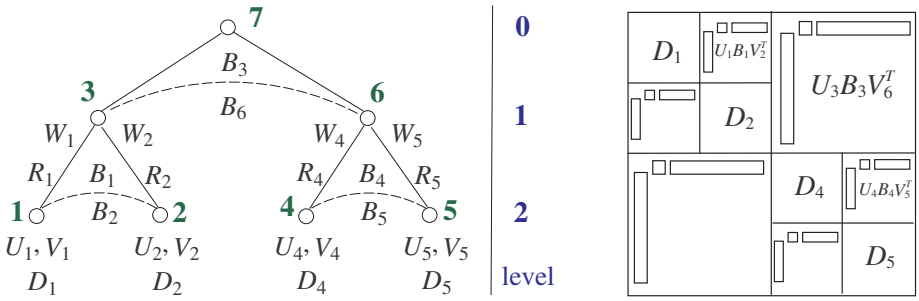
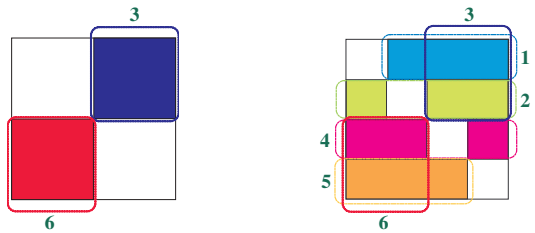


FIG. 4.2. *HSS tree for (??) and the multi-level representation.*

Correspondly, the matrix (4.2) can be partitioned into multiple levels and the off-diagonal blocks at different levels are show in Figure 4.3. These off-diagonal blocks are called HSS (off-diagonal) blocks. The bottom level diagonal blocks correspond to the leaf nodes of the HSS tree.



(i) One level of HSS blocks (ii) Two levels of HSS blocks

FIG. 4.3. *Two levels of HSS (off-diagonal) blocks.*

4.3. Robust structured factorization. The structured factorization in [44] leads to a nearly linear complexity solver for the problem. However, the structured extend-add process is extremely complicated due to HSS operations such as permutation, splitting, merging, and compression. In this paper, we use the robust structured factorization method in [45] which is for general dense matrices. It highly simplifies the overall solver.

The factorization method in [45] computes an approximate Cholesky factorization for a dense SPD matrix $F \approx LL^T$ where L is a lower triangular HSS matrix. The approximation LL^T can be accurate to any specified accuracy, and it has better positive definiteness than F due to the automatic Schur complement compensation. (It is not hard to show that in the HSS Cholesky factor, the U, V generators satisfy $U_i \equiv V_i$.) The fundamental idea can be illustrated by a block 2×2 example. Partition F as

$$F \equiv \begin{pmatrix} F_{11} & F_{21}^T \\ F_{21} & F_{22} \end{pmatrix}$$

and compute a Cholesky factorization $F_{11} = D_1 D_1^T$. We have

$$F = \begin{pmatrix} D_1 & \\ F_{21} D_1^{-T} & I \end{pmatrix} \begin{pmatrix} D_1^T & D_1^{-1} F_{21}^T \\ & S \end{pmatrix}, \tag{4.3}$$

where $S = F_{22} - (F_{21} D_1^{-T})(D_1^{-1} F_{21}^T)$ is the Schur complement. Compute an SVD of the off-diagonal block

$$D_1^{-1} F_{21}^T = \begin{pmatrix} U_1 & \hat{U}_1 \end{pmatrix} \begin{pmatrix} B_1 & \\ & \hat{B}_1 \end{pmatrix} \begin{pmatrix} V_2^T \\ \hat{V}_2^T \end{pmatrix} = U_1 B_1 V_2^T + \hat{U}_1 \hat{B}_1 \hat{V}_2^T,$$

where the partition of the SVD factors is decided by a tolerance τ so that $||\hat{B}_1||_2 = O(\tau)$. If $D_1^{-1} F_{21}^T$ is approximated by $U_1 B_1 V_2^T$, then $S = F_{22} - V_2^T B_1^T V_2 - \hat{V}_2^T \hat{B}_1^T \hat{V}_2$ can be approximated by

$$\tilde{S} = F_{22} - V_2^T B_1^T V_2. \tag{4.4}$$

Clearly, $\tilde{S}$ has enhanced positive definiteness. On the other hand, dropping $\hat{U}_1 \hat{B}_1 \hat{V}_2^T$ also saves the cost. Then compute a Cholesky factorization $\tilde{S} = D_2 D_2^T$ and we have

$$F \approx LL^T = \begin{pmatrix} D_1 & \\ U_2 B_2 U_1^T & D_2 \end{pmatrix} \begin{pmatrix} D_1^T & U_1 B_1 U_2^T \\ & D_2^T \end{pmatrix}.$$

In general, the factorization is conducted along the postordering traversal of a given full binary tree which becomes the HSS tree. See [45] for the details. The algorithm gives an HSS form L where the U generators all have orthonormal columns.

In this paper, we use this method to partially factorize the frontal matrices. That is, for a node j of the elimination tree, its frontal matrix $\mathcal{F}_j$ in (4.1) is (approximately) factorized as

$$\mathcal{F}_j \approx \left(\begin{array}{c|c} L_{j,j} & 0 \\ \hline L_{\{i_1,i_2,\ldots,i_r\},j} & I \end{array}\right) \left(\begin{array}{c|c} L_{j,j}^T & L_{\{i_1,i_2,\ldots,i_r\},j}^T \\ \hline 0 & \mathcal{U}_j \end{array}\right), \tag{4.5}$$

where $L_{j,j}$ is an HSS matrix, $\begin{pmatrix} L_{i_1,j}^T & \cdots & L_{i_r,j}^T \end{pmatrix}$ is in structured form given in the factorization of $F_{j,j}$, and $\mathcal{U}_j$ is dense. This partial factorization can be done as follows.

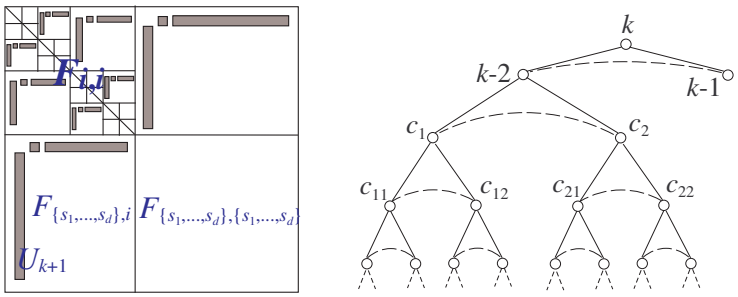


FIG. 4.4. *Partial factorization of a frontal matrix F_j and the HSS tree used. F_j is shown with its leading block $F_{j,j}$ represented by structured factors. The subtree $T[k-2]$ is for $F_{j,j}$.*

Set a full HSS tree T with k nodes where the left and right children of the root are $k-2$ and $k-1$ respectively. The entire subtree $T[k-2]$ is used as the HSS tree for completely factorizing $F_{j,j}$, and the single node $k-1$ is for the unfactorized part $F_{\{i_1,i_2,\ldots,i_r\},\{i_1,i_2,\ldots,i_r\}}$ of $\mathcal{F}_j$.

The algorithm in [45] is applied to $\mathcal{F}_j$ with following the postordering traversal of the nodes of the tree T. But the factorization stops when the entire $T[k-2]$ is visited. At the point, $F_{j,j} \approx L_{j,j}L_{j,j}^T$ where $L_{j,j}$ is an HSS matrix. Assume $L_{j,j}$ has generators D_i, U_i, R_i. Let nodes $c_1 < c_2$ be the children of $k-2$, and $c_{i1} < c_{i2}$ be the children of $c_i, i = 1, 2$. Then use the multi-level HSS representation as in Figure 4.2 we have a structured form of (4.5)

$$L_{j,j} = \left(\begin{array}{c|c} \left(\begin{array}{c|c} \ddots & 0 \\ \hline U_{c_{12}}B_{c_{12}}U_{c_{11}}^T & \ddots \end{array}\right) & 0 \\ \hline U_{c_2}B_{c_1}U_{c_1}^T & \left(\begin{array}{c|c} \ddots & 0 \\ \hline U_{c_{22}}B_{c_{22}}U_{c_{21}}^T & \ddots \end{array}\right) \end{array}\right), \tag{4.6}$$

$$L_{\{i_1,i_2,\ldots,i_r\},j} = H_j U_{k-2}^T, \tag{4.7}$$

$$\mathcal{U}_j \approx F_{\{i_1,i_2,\ldots,i_r\},\{i_1,i_2,\ldots,i_r\}} - H_j H_{k-1}^T, \tag{4.8}$$

where H_j is a factor obtained in the compression of $F_{\{i_1,i_2,\ldots,i_r\},j}$, and more specifically, $U_{k-2}H_{k-1}^T$ is a rank-revealing QR factorization of $L_{j,j}^{-1}F_{\{i_1,i_2,\ldots,i_r\},j}^T$. Since U_{k-2} has orthonormal columns, the approximation (4.8) to $\mathcal{U}_j$ is obtained by a low-rank update only. Furthermore, the approximation has better positive-definiteness than the exact $\mathcal{U}_j$, similar to the situation that $\tilde{S}$ in (4.4) approximates S in (4.3). The approximate $\mathcal{U}_j$ then participate in the construction of the frontal matrix of the parent frontal matrix just like in the standard multifrontal method. Each node j of the elimination tree is associated with an HSS form $L_{j,j}$ and a full rank matrix H_j. These are used in the structured multifrontal solution.

This process then proceeds along the elimination tree. After the elimination, we have an approximate factor $\mathcal{L}$ of A in structured form. See Figure 4.5 for an example. $\mathcal{L}$ is associated with two layers of postordering trees, the outer layer elimination tree, and an inner layer HSS tree for each node of the elimination tree. For better

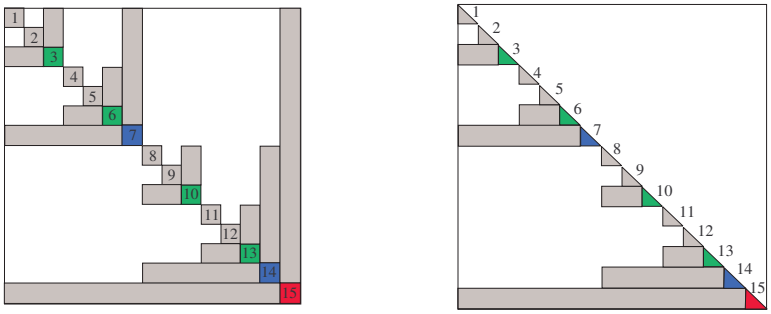


FIG. 4.5. *A sparse matrix in nested dissection ordering, and its structured factor.*

performance, at some bottom levels of the elimination, we can use the traditional multifrontal method. After a switching level l_s, structured factorizations are used.

ALGORITHM 1. (Robust structured multifrontal factorization)

1. Mesh ordering: Use nested dissection to order the mesh points. Build a postordering elimination tree with m nodes.
2. Symbolic factorization: Decide the switching level l_s (see Theorem 4.2 below). Based on the connections of separators and off-diagonal numerical ranks (or storage restriction), predict the structure of the factor. Allocate storage for update matrix stacks and the structured factor.
3. Numerical factorization: For separators $j = 1, \ldots, m$
   (a) If j is a leaf node, set j to be the jth initial frontal matrix.
   (b) If j is at level $l_j > l_s$
      i. Apply the traditional Cholesky factorization to partially factorize $\mathcal{F}_{j,j}$ until separator j is eliminated.
      ii. Compute $\mathcal{U}_j$, the Schur complement.
   (c) If j is at level $l_j \le l_s$
      i. Apply the structured Cholesky factorization above to partially factorize $\mathcal{F}_{j,j}$ until separator j is eliminated. Store $L_{j,j}$ in (4.6) and H_j in (4.7).
      ii. Compute $\mathcal{U}_j$ with a low-rank update as in (4.8).
   (d) If j is a left node, push $\mathcal{U}_j$ onto the update matrix stack.
   (e) If j is a right node, pop an update matrix $\mathcal{U}_i$ from the update matrix stack (i is the sibling of j). Use the traditional extend-add procedure to obtain the frontal matrix of the parent node of j.

About the complexity of the algorithm, we have the following theorem.

THEOREM 4.2. *Assume Algorithm 1 is used to factorize a discretized matrix A on an $N \times N$ regular mesh, and the elimination tree is a complete binary tree. Also assume the numerical ranks of all HSS off-diagonal blocks of the frontal matrices in the factorization are bounded by r. Then the complexity of Algorithm 1 is $O(rN^2 \log_2 N)$, if the elimination tree has $l = 2\lfloor \log_2 N \rfloor$ levels with the switch level l_s chosen such that the cost for the bottom level standard factorizations is the same as the cost for upper level structured factorizations. In such a situation, the storage for the structured factor is $O(N^2 \log_2(r \log_2 N))$.*

Proof. Among the total l levels of the elimination tree, the $l - l_s$ bottom levels are standard factorizations, and the upper l_s levels are structured. Level i has 2^i separators each with size $N/2^{\lfloor i/2 \rfloor}$. Since the mesh is regular, any separator at level

i has at most four upper level neighbors. The dimension of $\mathcal{F}_j$ is $O(N/2^{\lfloor i/2 \rfloor})$.

The major steps are the partial factorization step 3b or 3c, and the extend-add step 3e. Clearly, the extend-add step 3e costs $O((N/2^{\lfloor i/2 \rfloor})^2)$ flops for each right node j at level i.

If separator j is at level $i > l_s$, the partial factorization of $\mathcal{F}_j$ costs $O((N/2^{\lfloor i/2 \rfloor})^3)$ flops. This includes the factorization of $F_{j,j}$, the update of the block lower triangular part, and the computation of the Schur complement $\mathcal{U}_j$. If separator j is at level $i \leq l_s$, the partial factorization of $\mathcal{F}_j$ is structured. The factorizatin of $F_{j,j}$ costs $O(r(N/2^{\lfloor i/2 \rfloor})^2)$ flops, according to the HSS factorization algorithm in [45]. The update of the block lower triangular part (4.7) and the computation of $\mathcal{U}_j$ in (4.8) are are done by low-rank updates and their costs are bounded by $O(r(N/2^{\lfloor i/2 \rfloor})^2)$.

	Traditional factorizations	Structured factorizations
$l = O(\log_2 N)$ levels	$l - l_s$ bottom levels	l_s upper levels
Each level ($i = 0, 1, \ldots, l$)	2^i separators, each of size $O(N/2^{\lfloor i/2 \rfloor})$	
Cost for each separator	$O((N/2^{\lfloor i/2 \rfloor})^3)$	$O(r(N/2^{\lfloor i/2 \rfloor})^2)$
Total cost	$\sum_{i=l_s+1}^{l} 2^i O((N/2^{\lfloor i/2 \rfloor})^3)$	$\sum_{i=0}^{l_s} 2^i O(r(N/2^{\lfloor i/2 \rfloor})^2)$

TABLE 4.1
Flop count for Algorithm 1.

Table 4.1 shows the count of the complexity. Therefore, we have the total cost for the algorithm by adding the costs in the last row of Table 4.1

$$\xi(l_s) = \sum_{i=l_s+1}^{l} 2^i O((N/2^{\lfloor i/2 \rfloor})^3) + \sum_{i=0}^{l_s} 2^i O(r(N/2^{\lfloor i/2 \rfloor})^2)$$

$$\sum_{i=l_s/2+1}^{l/2} 2^{2i}(2^{l/2-i})^3 + \sum_{i=l_s/2+1}^{l/2} 2^{2i+1}(2^{l/2-i})^3 + \sum_{i=0}^{l_s/2}(l/2-i)r2^{2i}(2^{l/2-i})^2 + \sum_{i=0}^{l_s/2}(l/2-i)r2^{2i+1}(2^{l/2-i})^2$$

$$= O(2^l 2^{\frac{1}{2}(l-l_s)}) + O(2^l r l_s)$$

$$: -\tfrac{3}{8}2^l r\,(l_s+2)\,(l_s-2l)$$

$$\sum_{i=0}^{l/2}(l/2-i)2^{2i}(2^{l/2-i})^2 + \sum_{i=0}^{l/2}(l/2-i)2^{2i+1}(2^{l/2-i})^2$$

$$: \tfrac{3}{8}2^l l\,(l+2)$$

This cost is optimized if we choose l_s so that

$$2^{\frac{1}{2}(l-l_s)} \approx r l_s. \tag{4.9}$$

Since $r l_s \leq r l = O(r \log_2 N)$,

$$\xi(l_s) \leq O(2^l r l) = O(r N^2 \log_2 N).$$

This means, the optimal complexity is achieved if the cost before the switch level is equal to that after.

With the optimality condition (4.9), the storage requirement is

$$\begin{aligned}\sigma(l_s) &= \sum_{i=l_s+1}^{l} 2^i O((N/2^{\lfloor i/2 \rfloor})^2) + \sum_{i=0}^{l_s} 2^i O(r(N/2^{\lfloor i/2 \rfloor})) \\ &= O(2^l(l-l_s)) + O(r2^{\frac{1}{2}(l+l_s)}) \\ &= O(N^2 \log_2(rl_s)) + O(\frac{N^2}{l_s}) \\ &\lessapprox O(N^2 \log_2(r \log_2 N)).\end{aligned}$$

□

If we have knowledge about the maximum numerical rank r in advance, we can use the optimality condition (4.9) to find l_s in the symbolic factorization step. In fact, we can use (4.9) to verify that l_s satisfies $\alpha(l - 2\log_2 r) \lessapprox l_s \lessapprox l - 2\log_2 r$, where $\alpha = \frac{e \ln 2}{2+e \ln 2} \approx 0.49$ with e the natural exponential. The larger l is, the closer l_s is to $l - 2\log_2 r$. In such a situation, we can verify that the storage requirement $\sigma(l_s)$ is nearly $O(N^2 \log_2 r)$, or linear in the matrix size N^2.

As compared with other structured algorithms such as the one in [44], Algorithm 1 has some advantages. One is that this new algorithm is much simpler because the extend-add operation only uses the regular dense matrix addition. Another advantage is that it is more robust. For a symmetric matrix A, the structured factor is always guaranteed to exist and has enhanced positive definiteness. The tradeoff is that there is an extra $\log_2 N$ factor in the flop count, and an extra $O(N^2 \log_2 \log_2 N)$ memory requirement. However, since the implementation is significantly simpler, the actual running time is still competive when taking advantage of BLAS3 dense matrix operations. The storage count is nearly linear in N^2 in practice. For example, for N as large as 10^{300}, we still have $\log_2 \log_2 N < 10$.

4.4. Structured multifrontal solution. After the factorization, we have $A \approx \mathcal{L}\mathcal{L}^T$ where the diagonal blocks of $\mathcal{L}$ are in HSS forms and the lower off-diagonal blocks are in compressed forms. We then solve two structured triangular systems

$$\mathcal{L}y = b, \tag{4.10}$$

$$\mathcal{L}^T x = y. \tag{4.11}$$

For convenience, assume b, x, y are partitioned comfomally according to the sizes of the separators so that, say, $b = \left(b_1^T, b_2^T, \ldots, b_m^T\right)^T$ with the length of b_j equal to the number of nodes in separator j. Since the situation when $L_{j,j}$ is nonstructured (node j is before the switching level) is trivial, when describing the algorithm we assume the factorization for each node is structured.

The solution of (4.10) with forward substitution involves forward (or postordering) traversal of the elimination tree. Initially, for node $j = 1$ of the elimination tree, we solve a lower triangular HSS system $L_{1,1}y_1 = b_1$. A triangular HSS solver similar to the one in [34] is used. This is done by a forward (or postordering) traversal of the HSS tree for $L_{1,1}$. In such a solver, b_1 is partitioned again according to the according to the leaf nodes of the HSS tree. Assume $\{i_1, i_2, \ldots, i_r\}$ be the set of upper level neighbor separators of separator 1. Then $b_{i_1}, b_{i_2}, \ldots, b_{i_r}$ should be updated. Let

$$t_1 = H_1(U_{k-2}^T y_1),$$

where $k-2$ is the total number of nodes in the HSS tree for $L_{1,1}$. Partition t_1 as $\left(t_{1,i_1}^T, t_{1,i_2}^T, \ldots, t_{1,i_r}^T\right)^T$ according to the sizes of separators $i_1, i_2, \ldots, i_r$. Then the righthand side vector b is updated via

$$b_i \leftarrow b_i - t_{1,i},\ i = i_1, i_2, \ldots, i_r.$$

Since U_{k-2} may only be available implicitly in terms of lower level HSS generators, an HSS matrix vector product may be needed [17]. The process then applies similarly to $j = 2, 3, \ldots, m$.

In the backward substitution stage for solving (4.11), we traverse the elimination tree top-down. Initially, for node $j = m$ of the elimination tree, solve an upper triangular HSS system $L_{m,m}x_m = y_m$. Then assume $\{i_1, i_2, \ldots, i_s\}$ be the set of *lower* level neighbor separators of separator n. We should update $y_{i_1}, y_{i_2}, \ldots, y_{i_s}$. Let

$$t_m = U_{k-2}(H_m^T x_m),$$

where $k-2$ is the total number of nodes in the HSS tree for $L_{m,m}$. Partition t_m as $\left(t_{m,i_1}^T, t_{1,i_2}^T, \ldots, t_{m,i_s}^T\right)^T$ according to the sizes of separators $i_1, i_2, \ldots, i_s$. Then the righthand side vector y is updated via

$$y_i \leftarrow y_i - t_{1,i},\ i = i_1, i_2, \ldots, i_s.$$

Again, an HSS matrix vector product may be needed to compute t_m. The process applies similarly to $j = m-1, m-2, \ldots, 1$.

The complexity of the solution process is also almost linear in N^2 in practice.

THEOREM 4.3. *Under the same conditions as in Theorem 4.2, the cost to solve (4.10) or (4.11) is* $O(N^2 \log_2(r \log_2 N))$.

5. Numerical experiments.

5.1. Implementation issues. The new algorithm applies to general irregular meshes. In fact, it can even factorize arbitrary SPD sparse matrices, since the ordering of the rows and columns can be done on the connectivity graph of the matrix. Various ordering packages can be used. We build a nested dissection code based on the toolbox `MESHPART` [24] which uses `METIS` [38].

5.2. Elliptic problems. *(Show complexity and storage)*

5.3. Spectral method for elliptic equations and Helmholtz equations. Nelx = Nely = 20

5.4. Linear elasticity near incompressible limit. Consider a linear elasticity equation

$$-(\mu\overrightarrow{\Delta u} + (\lambda+\mu)\nabla\overrightarrow{\nabla \cdot u}) = \overrightarrow{f} \text{ in } \Omega = [0,1]\times[0,1], \quad \overrightarrow{u} = \overrightarrow{0} \text{ on } \partial\Omega, \tag{5.1}$$

where $\overrightarrow{u}$ is the displacement vector field, and λ and μ are the Lamé constants. This PDE is very ill conditioned when λ/μ is large. This limit is known as the incompressible limit which, and for example, is associated with the mechanical behavior of elastomeric materials and plastic flow in metals. It is an important situation in practice.

To demonstrate the existence of the low-rank property in this problem, we consider a frontal matrix F corresponding to the top level separator in nested dissection. For different λ/μ and different tolerances in the approximation, the maximum off-diagonal numerical ranks for F are given in Table 5.1. The numerical ranks are relatively small as compared with the matrix size. This indicates that the dense fill-in can be compressed. More significant is that, the numerical ranks are relatively insensitive to λ/μ, even if it is very large. This means, low-rank structured approximate factorizations of the problems are suitable for ill-conditioned situations. Or, our method is a good choice near the incompressible limit.

τ		10^{-2}	10^{-4}	10^{-6}	10^{-8}
$\frac{\lambda}{\mu}$	1	8	22	34	43
	10^2	4	17	31	40

τ		10^{-2}	10^{-4}	10^{-6}	10^{-8}
$\frac{\lambda}{\mu}$	10^4	3	11	22	33
	10^6	3	10	16	27

TABLE 5.1

Maximum off-diagonal numerical ranks of an order 201 *frontal matrix in the supernodal multifrontal method for (5.1) with different compression tolerances* τ.

5.5. Time harmonic Maxwell equations. The problems of electromagnetic fields are governed by Maxwell equations. The numerical solution of time harmonic Maxwell equations (and also Helmholtz equations, etc.) has a lot of challenges. One is that, the solution is smooth and oscillatory away from boundaries and interfaces. The approximation of oscillations generally requires sufficiently small mesh element sizes compared to the wavelengh of the solution. This leads to a large number of mesh points, especially for high wave numbers [**?**, **?**, **?**, **?**, **?**, **?**]. In addition, the resulting discretized linear systems are often severely ill conditioned. Iterative methods with many existing preconditioners including multigrid may fail to converge quickly for these problems. It is well-known that for high wave numbers, standard multigrid often encounters difficulty in approximating the problem on coarse grids, and generally, complicated implementations of multigrid are needed [**?**, **?**, **?**].

On the other hand, our preconditioners are based on structured approximate factorizations and approximate information at all grid points are used to predict the solutions. Our preliminary tests indicate that our algorithms are relatively insensitive to wave numbers, when a discretized matrix is obtained for a wave number with appropriate mesh sizes. For example, consider the following time harmonic Maxwell equations

$$\begin{aligned} \nabla \times \nabla \times \mathbf{E} - k^2 \mathbf{E} &= \mathbf{J} \text{ in } \Omega, \\ \mathbf{E} \times \mathbf{n} &= \mathbf{g} \text{ on } \partial\Omega, \end{aligned} \tag{5.2}$$

where $\mathbf{n}$ is the unit outer normal of $\partial\Omega$, and appropriate parameters are defined similar to those in [18]. *To be done.....*

5.6. Helmholtz equations in seismic-imaging problems. Seismic-imaging problems often requires to solve large discretized Helmholtz equations for different frequencies. Each frequency leads to one coefficient matrix. The problem is then solved against many sources and receivers which correspond to multiple right-hand side vectors for each same coefficient matrix. The discretized matrices are usually indefinite and the solutions are highly damped for high frequencies. Multigrid methods or similar are generally used to solve these problems [**?**, **?**, **?**]. However, standard multigrid may not work well for all frequencies or multiple sources and receivers.

On the other hand, a preliminary implementation of our algorithm as an approximate factorization algorithm gives superior test results. Our algorithm is attractive in various aspects:

1. Our structured factorization is highly efficient for multiple right-hand sides with the same coefficient matrix (i.e., multiple sources and receivers corresponding to each frequency), since only one factorization is needed for each frequency.
2. All the matrices have the same nonzero pattern, so our symbolic factorization and graph partitioning need only to be done once for all frequencies, sources, and receivers.
3. The dense fill-in in the factorization turns out to be reasonably compressible, and our structured factorizations give very satisfactory approximate solutions without high compression rate (or, related off-diagonal numerical ranks need not to be very small).
4. The structured factorization is robust and relatively insensitive to frequencies.

To be done.....

Acknowledgments.

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] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. Vorst, eds., *Templates for the solution of algebraic eigenvalue problems: a practical guide*, SIAM, Philadelphia, 2000.

[3] M. Bebendorf, *Efficient inversion of Galerkin matrices of general second-order elliptic differential operators with nonsmooth coefficients*, Math. Comp., 74 (2005), pp. 1179–1199.

[4] M. Bebendorf and W. Hackbusch, *Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-Coefficients*, Numer. Math., 95 (2003), pp. 1–28.

[5] 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.

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

[7] S. Chandrasekaran, P. Dewilde, and M. Gu, *On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs*, preprint, 2007.

[8] S. Chandrasekaran, P. Dewilde, M. Gu, W. Lyons, and T. Pals, *A fast solver for HSS representations via sparse matrices*, SIAM J. Matrix Anal. Appl., 29 (2006), pp. 67–81.

[9] 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.

[10] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, and A.-J. van der Veen, *Fast stable solver for sequentially semi-separable linear systems of equations*, in High Performance Computing - HiPC 2002: 9th International Conference, Lecture Notes in Comput. Sci. 2552, Springer-Verlag, Heidelberg, 2002, pp. 545–554.

[11] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Fast stable solvers for sequentially semi-separable linear systems of equations and least squares problems*, Technical report, University of California, Berkeley, CA, 2003.

[12] S. Chandrasekaran, P. Dewilde, M. Gu, T. Pals, X. Sun, A.-J. van der Veen, and D. White, *Some fast algorithms for sequentially semiseparable representations*, SIAM J. Matrix Anal. Appl, 27 (2005), pp. 341-364.

[13] S. Chandrasekaran, M. Gu, X.S. Li, and P. Vesselevski, *Schur-monotonic semi-separable approximations of symmetric positive definite matrices*, Research Notes, 2006.

[14] S. Chandrasekaran, M. Gu, X.S. Li, and J. Xia, *Some fast algorithms for hierarchically semiseparable matrices*, Technical Report, CAM 08-24, UCLA.

[15] S. Chandrasekaran, M. Gu, X. Sun, J. Xia, and J. Zhu, *A superfast algorithm for Toeplitz systems of linear equations*, SIAM J. Matrix Anal. Appl., 29 (2007), pp. 1247–1266.

[16] S. Chandrasekaran, M. Gu, and W. Lyons, *A fast and stable adaptive solver for hierarchi-*

cally semi-separable representations, Technical Report, UCSB Math 2004-20, U.C. Santa Barbara, 2004.

[17] S. Chandrasekaran, M. Gu, and T. Pals, *Fast and stable algorithms for hierarchically semi-separable representations*, Technical Report, UCSB, April 2004.

[18] Z. Chen, L. Wang, and W. Zheng, *An adaptive multilevel method for time-harmonic Maxwell equations with singularities*, SIAM J. Sci. Comput., 29 (2007), pp. 118–138.

[19] J. Demmel, *The condition number of equivalence transformations that block diagonalize matrix pencils*, SIAM J. Numer. Anal., 20 (1983), pp. 599–610.

[20] J. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

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

[22] Y. Eidelman and I. Gohberg, *On a new class of structured matrices*, Integral Equations Operator Theory, 34 (1999) pp. 293–324.

[23] J. A. George, *Nested dissection of a regular finite element mesh*, SIAM J. Numer. Anal., 10 (1973), pp. 345–363.

[24] J. R. Gilbert and S.-H. Teng, *MESHPART, A Matlab Mesh Partitioning and Graph Separator Toolbox*, `http://aton.cerfacs.fr/algor/Softs/MESHPART/`.

[25] L. Grasedyck, R. Kriemann, and S. Le Borne, *Parallel black box domain decomposition based $\mathcal{H}$-LU preconditioning*, Technical Report 115, Max Planck Institute for Mathematics in the Sciences, Leipzig, 2005.

[26] L. Grasedyck, R. Kriemann, and S. Le Borne, *Domain-decomposition based $\mathcal{H}$-LU preconditioners*, in Domain Decomposition Methods in Science and Engineering XVI, O.B.Widlund and D.E.Keyes (eds.), Springer LNCSE, 55 (2006), pp. 661–668.

[27] W. Hackbusch, *A Sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: introduction to $\mathcal{H}$-matrices*, Computing, 62 (1999), pp. 89–108.

[28] W. Hackbusch and B. N. Khoromskij, *A sparse $\mathcal{H}$-matrix arithmetic. Part-II: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.

[29] I. Hladik, M. B. Reed, and G. Swoboda, *Robust preconditioners for linear elasticity FEM analysis*, Internat. J. Numer. Methods Engng., 40 (1997), pp. 2109-2127.

[30] A. J. Hoffman, M. S. Martin, and D. J. Rose, *Complexity bounds for regular finite difference and finite element grids*, SIAM J. Numer. Anal., 10 (1973), pp. 364–369.

[31] 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.

[32] D. S. Kershaw, *The incomplete Cholesky conjugate gradient method for the iterative solution of systems of linear equations*, J. Comp. Phys., 26 (1977), pp. 43–65.

[33] J. W. H. Liu, *The multifrontal method for sparse matrix solution: Theory and practice*, SIAM Review, 34 (1992), pp. 82–109.

[34] W. Lyons, S. Chandrasekaran, and M. Gu, *Fast LU decomposition for operators with hierarchically semiseparable structure*, preprint submitted to Elsevier Science, 2005.

[35] P. G. Martinsson, V. Rokhlin, and M. Tygert, *A fast algorithm for the inversion of general Toeplitz matrices*, Comput. Math. Appl. 50 (2005), pp. 741–752.

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

[37] 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. Comput. 31 (1977), pp. 148–162.

[38] METIS, *Family of Multilevel Partitioning Algorithms*, `http://glaros.dtc.umn.edu/gkhome/views/metis`.

[39] S. V. Parter, *The use of linear graphs in gaussian elimination*, SIAM Rev., 3 (1961), pp. 119–130.

[40] M. Rezghi and S. M. Hosseini, *An ILU preconditioner for nonsymmetric positive definite matrices by using the conjugate Gram-Schmidt process*, J. Comput. Appl. Math., 188 (2006), pp. 150–164.

[41] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math., 13 (1969), pp. 354–356.

[42] R. P. Tewarson, *On the product form of inverses of sparse matrices and graph theory*, SIAM Rev., 9 (1967), pp. 91–99.

[43] R. Vandebril, M. Van Barel, G. Golub, and N. Mastronardi, *A bibliography on semiseparable matrices*, Calcolo, 42 (2005), pp. 249–270.

[44] S. Chandrasekaran, M. Gu, and X. S. Li, J. Xia, *Superfast multifrontal method for structured linear systems of equations*, submitted to SIAM J. Matrix Anal. Appl., 2009, `http://www.math.purdue.edu/~xiaj/work/supermf.pdf`.

[45] M. Gu and J. Xia, *Robust structured factorization and preconditioning for SPD matrices*, submitted to SIAM J. Matrix Anal. Appl., 2009.