Chapter 10
Robust and Efficient Multifrontal Solver for Large Discretized PDEs

Jianlin Xia

Abstract This paper presents a robust structured multifrontal factorization method for large symmetric positive definite sparse matrices arising from the discretization of partial differential equations (PDEs). For PDEs such as 2D and 3D elliptic equations, the method costs roughly $O(n)$ and $O(n^{4/3})$ flops, respectively. The algorithm takes advantage of a low-rank property in the direct factorization of some discretized matrices. We organize the factorization with a supernodal multifrontal method after the nested dissection ordering of the matrix. Dense intermediate matrices in the factorization are approximately factorized into hierarchically semiseparable (HSS) forms, so that a data-sparse Cholesky factor is computed and is guaranteed to exist, regardless of the accuracy of the approximation. We also use an idea of rank relaxation for HSS methods so as to achieve similar performance with flexible structures in broader types of PDE. Due to the structures and the rank relaxation, the performance of the method is relatively insensitive to parameters such as frequencies and sizes of discontinuities. Our method is also much simpler than similar structured multifrontal methods, and is more generally applicable (to PDEs on irregular meshes and to general sparse matrices as a black-box direct solver). The method also has the potential to work as a robust and effective preconditioner even if the low-rank property is insignificant. We demonstrate the efficiency and effectiveness of the method with several important PDEs. Various comparisons with other similar methods are given.

10.1 Introduction

Large sparse linear systems arise frequently from numerical and engineering problems, in particular, the discretization of partial differential equations (PDEs). Typically, there are two types of linear system solver, 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

J. Xia (✉)
Department of Mathematics, Purdue University, West Lafayette, IN, USA
e-mail: xiaj@math.purdue.edu

M.W. Berry et al. (eds.), *High-Performance Scientific Computing*,
DOI 10.1007/978-1-4471-2437-5_10,

converge slowly if no effective preconditioners are available. Also, classical ILU preconditioners may suffer from breakdown.

Assume we have a system

$$Ax = b, \tag{10.1}$$

where A is an $n \times n$ symmetric positive definite (SPD) matrix. If A arises from the discretization of some PDEs. It may be associated with a mesh. In the direct solution of the system, A or the mesh points can be reordered so as to reduce fill-in. For example, the nested dissection ordering [12] and its generalizations can be used to get nearly optimal exact factorization complexity, which is generally $O(n^{3/2})$ in 2D or $O(n^2)$ in 3D [17]. In nested dissection, a mesh is recursively divided with separators (small sets of mesh points). However, notice that some iterative methods such as multigrid converge with $O(n)$ complexity for some PDEs.

In the recent years, nearly linear complexity structured approximate factorization methods have been developed based on a *low-rank property*. It has been noticed that, during the direct solution of some PDEs such as elliptic equations, certain off-diagonal blocks of the intermediate dense matrices or fill-in have small numerical ranks [1, 2, 4, 19, 33, etc.]. This property is closely related to the idea of the fast multipole method [14] and the property of certain Green's functions which are smooth away from the diagonal singularity under certain conditions. This property can be used to improve the computational efficiency, with dense intermediate matrices approximated by rank structured matrices such as quasiseparable, semiseparable, or hierarchical matrices [2, 10, 15, 16, 28, etc.]. This idea is widely used in the development of new fast algorithms. Related techniques have also been shown very useful in high performance scientific computing [23, 24, 29].

Rank structured methods can be fully integrated into sparse matrix techniques to provide new fast solvers. In [33] and [25, 26], structured sparse factorization algorithms are proposed based on the multifrontal method [9, 20] and hierarchically semiseparable (HSS) matrices [3, 5, 34] or hierarchical matrices. The algorithms have nearly linear complexity and linear storage requirement for some problems. The method in [33] involves complicated HSS operations, and are mainly applicable to regular meshes. Later, more general structured multifrontal methods have been discussed in [30, 31] and [25]. The method in [25] also requires the mesh to be nearly regular (or the location and layout of the separators in nested dissection follow the patterns of those in a regular mesh). Both methods in [25, 33] only work for 2D problems. The 3D method in [26] only works for regular meshes. All these methods may suffer from the problem of breakdown, especially when a low accuracy is used, say, in preconditioning. In addition, these methods generally require bounded off-diagonal ranks in the low-rank property.

In this paper, we propose a more robust and more general structured multifrontal algorithm, following the preliminary discussions in the report [30]. We use a flexible nested dissection algorithm that works for irregular meshes in both 2D and 3D. In the meantime, a robust HSS Cholesky factorization algorithm in [35] is generalized to the context of the multifrontal method, so that

an approximate multifrontal factorization can always be computed without breakdown, in general. We also simplify the process by preserving certain dense operations, which keeps the performance to be similar to the fully structured version.

An optimization step is used in the multifrontal scheme (Theorem 10.1 below), so that the complexity can be lower than similar methods in [11, 25] by up to a factor of $O(\log n)$. Moreover, we relax the classical rank requirement in [25, 33], so that the structured sparse solution is fast even if the related numerical ranks are not bounded. Traditionally, HSS operations require the off-diagonal (numerical) ranks of a dense matrix to be bounded in order to achieve linear complexity. Here, the rank relaxation idea in [32] indicates that similar complexity can be achieved without this requirement. That is, the ranks are actually allowed to increase along the block sizes. This is then generalized to the rank relaxation in our robust sparse solution. It enhances the flexibility and applicability of structured multifrontal solvers, and is especially useful for difficult problems such as Helmholtz equations with high frequencies and 3D equations.

With the relaxed rank requirement, this new method has complexity similar to the one in [33], but applies to more general sparse matrices including 3D discretized ones. The factorization costs for some 2D and 3D discretized equations (elliptic, Helmholtz, etc.) are roughly $O(n)$ and $O(n^{4/3})$ flops, respectively (see Theorem 10.1 and Remark 10.1). In contrast, the exact factorization generally costs at least $O(n^{3/2})$ in 2D and $O(n^2)$ in 3D. We point out that, after the factorization, the solution cost and the storage requirement are both nearly $O(n)$, including for 3D. Furthermore, the rank structures and the rank relaxation idea indicate that the performance of the method is relatively insensitive to parameters such as frequencies in some problems.

Our method is especially useful for direct solutions of sparse linear systems with multiple right-hand sides, involving some parameters, and/or with only modest accuracy desired. It also has the potential to be used as a robust and effective preconditioner when the rank property is insignificant. The method uses two layers of tree structures, an outer one for the multifrontal method, and an inner one for each intermediate HSS matrix. It is thus suitable for parallel implementations. Several numerical examples are shown, including a Poisson equation, an interface problem, and a linear elasticity equation. The later two are ill conditioned, but our method (as a solver or a preconditioner) has similar performance for a large range of parameters. Both analytical and numerical comparisons with other similar methods are given.

The remaining sections are organized as follows. Section 10.2 reviews a dense HSS Cholesky factorization method. New structured multifrontal factorization and solution algorithms are developed in Sect. 10.4. Section 10.5 shows the algorithm and its complexity analysis. The numerical experiments are given in Sect. 10.6.

10.2 Review of HSS Cholesky Factorization of a Dense Matrix

10.2.1 Hierarchically Semiseparable Structures

HSS structures are very useful in handling dense matrices with the low-rank property. The definition of a postordering HSS form is as follows [34].

Definition 10.1 Assume F is an $N \times N$ (real) matrix, and $\mathcal{I} = \{1, 2, \ldots, N\}$. Let T be a binary tree with k nodes, and $t_i \subset \mathcal{I}$ be an index set associated with each node i of T. Let $F|_{t_i \times t_j}$ denote the submatrix of F with a row index set t_i and a column index set t_j in $\mathcal{I}$. We say F is in an *HSS form* with the corresponding *HSS tree* T if:

1. T is a postordered full binary tree: each node i is either a leaf or is a non-leaf node with two children c_1 and c_2 which are ordered as $c_1 < c_2 < i$.
2. For each non-leaf node i, $t_{c_1} \cup t_{c_2} = t_i$, $t_{c_1} \cap t_{c_2} = \phi$, and $t_{2k-1} = \mathcal{I}$.
3. There exists matrices $D_i, U_i, V_i, R_i, W_i, B_i$ (called *HSS generators*) associated with each node i satisfying

$$D_i = \begin{pmatrix} D_{c_1} & U_{c_1} B_{c_1} V_{c_2}^T \\ U_{c_2} B_{c_2} V_{c_1}^T & D_{c_2} \end{pmatrix}, \qquad U_i = \begin{pmatrix} U_{c_1} & R_{c_1} \\ U_{c_2} & R_{c_2} \end{pmatrix}, \qquad V_i = \begin{pmatrix} V_{c_1} & W_{c_1} \\ V_{c_2} & W_{c_2} \end{pmatrix}, \tag{10.2}$$

   so that $D_i \equiv F|_{t_i \times t_i}$. Here, the generators associated with the root k are empty matrices except $D_k \equiv F$.

The HSS form of F is given by the generators. For a non-leaf node i, the generators D_i, U_i, V_i are recursively defined and are not explicitly stored. Clearly, U_i is a basis for the column space of $F_i^- = F|_{t_i \times (\mathcal{I} \setminus t_i)}$, and V_i^T is a basis for the row space of $F_i^| = F|_{(\mathcal{I} \setminus t_i) \times t_i}$. These off-diagonal blocks F_i^- and $F_i^|$ are called *HSS blocks*. The maximum (numerical) rank of all the HSS blocks is called the *HSS rank* of F. If F is symmetric, we can set [34]

$$D_i = D_i^T, \quad V_i = U_i, \quad B_j = B_i^T \quad (j\text{: sibling of } i).$$

10.2.2 Robust HSS Cholesky Factorization

Given a dense real SPD matrix F, we can use the method in [35] to compute an approximate Cholesky factorization $F \approx LL^T$, where L is a lower triangular HSS matrix. LL^T generally exists for any given approximation accuracy. The fundamental idea can be illustrated in terms of a block 2×2 SPD matrix. Factorize the $(1, 1)$

block of the following matrix:

$$F \equiv \begin{pmatrix} F_{1,1} & F_{2,1}^T \\ F_{2,1} & F_{2,2} \end{pmatrix} = \begin{pmatrix} D_1 & \\ F_{2,1} D_1^{-T} & I \end{pmatrix} \begin{pmatrix} D_1^T & D_1^{-1} F_{2,1}^T \\ & S \end{pmatrix},$$

where $F_{1,1} = D_1 D_1^T$ is the Cholesky factorization of $F_{1,1}$, and $S = F_{22} - (F_{2,1} D_1^{-T}) \cdot (D_1^{-1} F_{2,1}^T)^T$ is the Schur complement. Compute an SVD $F_{2,1} D_1^{-T} = U_2 B_1^T U_1^T + \hat{U}_2 \hat{B}_1^T \hat{U}_1^T$, where all the singular values greater than a tolerance τ are in B_1. (The number of singular values in B_1 is the off-diagonal *numerical rank* r.) Then

$$F \approx \begin{pmatrix} D_1 & \\ U_2 B_1^T U_1^T & I \end{pmatrix} \begin{pmatrix} D_1^T & U_1 B_1 U_2^T \\ & \tilde{S} \end{pmatrix},$$

where $\tilde{S}$ is an approximate Schur complement given by

$$\tilde{S} = F_{2,2} - U_2 B_1^2 U_2^T = S + O(\tau^2).$$

That is, a positive semidefinite term is implicitly added to the Schur complement. Then a Cholesky factorization $\tilde{S} = D_2 D_2^T$ yields

$$F \approx L L^T, \qquad L = \begin{pmatrix} D_1 & \\ U_2 B_1^T U_1^T & D_2 \end{pmatrix}.$$

Therefore, we obtain an approximate Cholesky factor L which is a block 2×2 HSS form. It is also shown in [35] that, with certain modifications, L can work as an effective preconditioner when the low-rank property is insignificant. That is, if the HSS rank of A for a small tolerance is large, L can be obtained by manually choosing a small rank r (and a large tolerance). The idea can be generalized to multiple blocks so that L is a general lower-triangular HSS matrix.

10.3 Nested Dissection for General Graphs

Before the numerical factorization of a sparse SPD matrix A, it is often reordered so as to reduce fill-in. Nested dissection generally leads to the optimal complexity for 2D and 3D discretized matrices [17].

In the following discussions, we focus on discretized matrices. For general sparse matrices, we can similarly consider the adjacency graph. Treat the mesh in the discretization as an undirected graph $(\mathcal{V}, \mathcal{E})$. Each mesh point $i \in \mathcal{V}$ corresponds to a row and a column of A, and each edge $(i, j) \in \mathcal{E}$ corresponds to the entries $A_{ij} = A_{ji} \neq 0$. A separator in $\mathcal{V}$ is found to divide the entire mesh into to two subregions, which are further divided recursively. Unlike the method in [33] which uses coordinates of mesh points, graph partition tools can be employed to handle more general meshes. Here, we use `METIS` [18], and follow the basic ideas in `Meshpart` [13]. See Figs. 10.1, 10.2 for some examples.

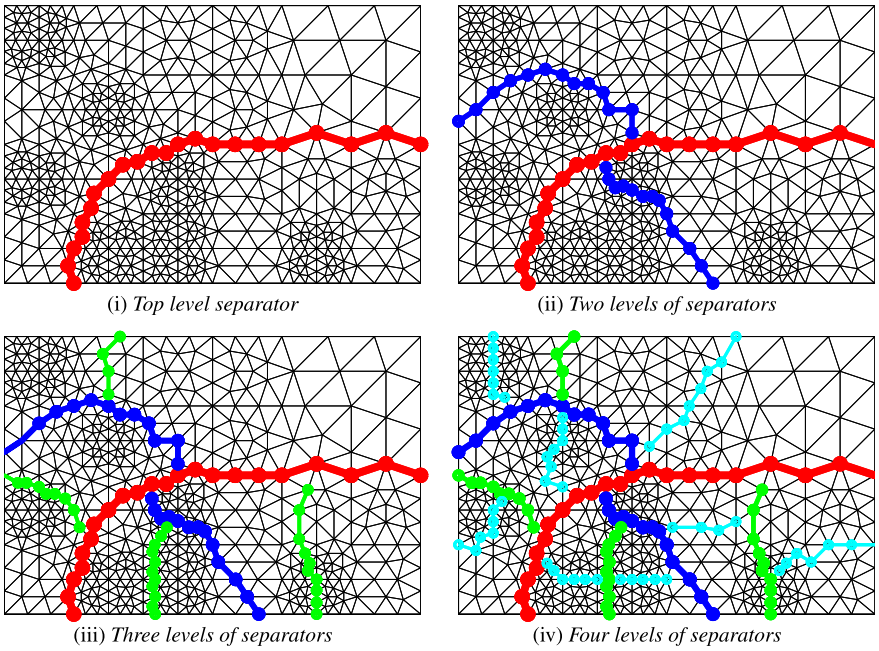


Fig. 10.1 Multiple levels of separators in nested dissection for an irregular mesh from `Meshpart` [13]

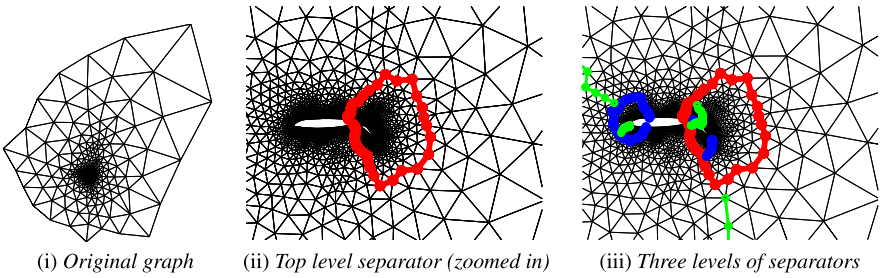


Fig. 10.2 Multiple levels of separators in nested dissection for an irregular mesh with a missing piece, where the matrix is from the University of Florida sparse matrix collection [7]

Lower level separators are ordered before upper level ones. For example, Fig. 10.3 shows the nonzero pattern of a discretized matrix A after the reordering of a 3D mesh. During the factorization of A, the elimination of a mesh point mutually connects points which are previously connected to it [22, 27]. This creates fill-in.

As compared with the methods in [25, 33], our work has more flexibility:

1. The domains can be in any shape, such as with missing pieces (Fig. 10.2).
2. Both 2D and 3D domains can be handled (Fig. 10.3).

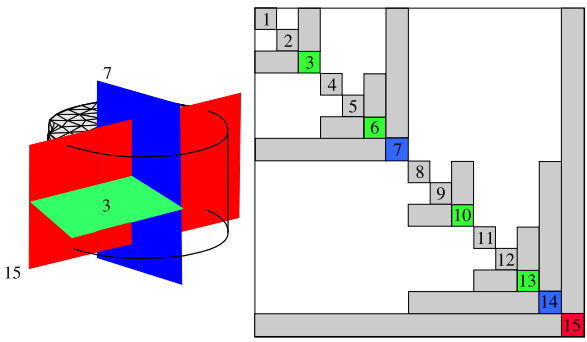


Fig. 10.3 Three levels of partition of a 3D mesh and the corresponding nonzero pattern of A after the nested dissection ordering

3. The mesh points and separators can be arbitrarily located, and the separators can be arbitrarily connected to each other.
4. Our method can also work as a black box for general sparse matrices.

10.4 Robust Structured Multifrontal Factorization

In this section, we consider the direct factorization of sparse SPD matrices with the multifrontal method [9, 20], which is one of the most important sparse factorization algorithms. During the factorization, if the dense intermediate matrices have the low-rank property, we use fast robust HSS methods to replace the dense operations.

10.4.1 Multifrontal Method

The multifrontal method [9, 20] reorganizes the sparse Cholesky factorization $A = \mathcal{L}\mathcal{L}^T$ into local factorizations of intermediate dense matrices, where $\mathcal{L}$ is lower triangular. The factorization is conducted following a tree called *elimination tree*, or more generally, an *assembly tree*. In general, an elimination tree $\mathcal{T}$ has n nodes and a node p is the parent of i if and only if

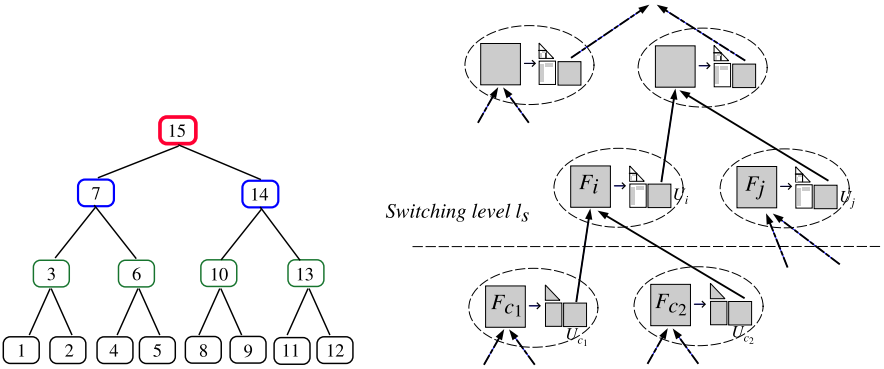
$$p = \min\{j > i | \mathcal{L}|_{j\times i} \neq 0\},$$

where $\mathcal{L}|_{j\times i}$ represents the (j, i) entry of $\mathcal{L}$. Use $\mathcal{T}[i]$ to denote the subtree of $\mathcal{T}$ with root i. Let $\mathcal{N}_i \equiv \{j_1, j_2, \ldots, j_d\}$ be the set of row indices of nonzeros in $\mathcal{L}_{:,i}$ (the ith column of $\mathcal{L}$) with i excluded. The ith *frontal matrix* is defined to be

$$\mathcal{F}_i = \begin{pmatrix} A|_{i\times i} & (A|_{\mathcal{N}_i\times i})^T \\ A|_{\mathcal{N}_i\times i} & 0 \end{pmatrix} - \sum_{j\in\mathcal{T}[i]\setminus i} \mathcal{L}|_{(i\cup\mathcal{N}_i)\times j}(\mathcal{L}|_{(i\cup\mathcal{N}_i)\times j})^T.$$

One step of elimination applied to $\mathcal{F}_i$ provides the column $\mathcal{L}_{(i\cup\mathcal{N}_i)\times i}$:

$$\mathcal{F}_i = \begin{pmatrix} \mathcal{L}|_{i\times i} & 0 \\ \mathcal{L}|_{\mathcal{N}_i\times i} & I \end{pmatrix} \begin{pmatrix} (\mathcal{L}|_{i\times i})^T & (\mathcal{L}|_{\mathcal{N}_i\times i})^T \\ 0 & \mathcal{U}_i \end{pmatrix},$$



(i) *Supernodal elimination tree* $\mathcal{T}$ (ii) *Robust structured multifrontal factorization*

Fig. 10.4 Supernodal version elimination tree for the problem in Fig. 10.3, and also a general pattern of triangular structured multifrontal factorization, where a switching level $\mathbf{l}_s$ is marked

where $\mathcal{U}_i$ is the contribution from $\mathcal{T}[i]$ to p and is called the ith *update matrix*:

$$\mathcal{U}_i = -\sum_{j \in \mathcal{T}[i]} \mathcal{L}|_{\mathcal{N}_i \times j} (\mathcal{L}|_{\mathcal{N}_i \times j})^T.$$

Update matrices are used to form upper level frontal matrices. This process is called an *extend–add* operation, which matches indices and add entries, denoted

$$\mathcal{F}_i = \begin{pmatrix} A|_{j\times j} & (A|_{\mathcal{N}_i \times i})^T \\ A|_{\mathcal{N}_i \times i} & 0 \end{pmatrix} \Diamond\!\!\!\!\!+\, \mathcal{U}_{c_1} \Diamond\!\!\!\!\!+\, \mathcal{U}_{c_2} \Diamond\!\!\!\!\!+\, \cdots \Diamond\!\!\!\!\!+\, \mathcal{U}_{c_q},$$

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

10.4.2 Structured Supernodal Multifrontal Factorization

Here, we use nested dissection to reorder A and to produce a binary tree $\mathcal{T}$ as the assembly tree in a supernodal version of the multifrontal method, where each separator is treated as a node in the tree. Figure 10.4(i) shows the assembly tree for the mesh in Fig. 10.3.

Assume the root of $\mathcal{T}$ is at level 0, and the leaves are at the largest level. For a separator i, let $\mathcal{N}_i \equiv \{j_1, j_2, \ldots, j_d\}$ be the set of neighbor separators of i at the same or upper levels of i in $\mathcal{T}$. Also let t_j denote the index set of the neighbor j in A, and let $\hat{t}_j$ denote the subset of t_j that is connected to i due to lower level eliminations. The frontal matrix $\mathcal{F}_i$ is formed by the block form extend–add operation

$$\mathcal{F}_i = \mathcal{F}_i^0 \Diamond\!\!\!\!\!+\, \mathcal{U}_{c_1} \Diamond\!\!\!\!\!+\, \mathcal{U}_{c_2}, \quad \mathcal{F}_i^0 \equiv \begin{pmatrix} A|_{t_i \times t_i} & \left(A|_{(\cup_{j=1}^d \hat{t}_j)\times t_i}\right)^T \\ A|_{(\cup_{j=1}^d \hat{t}_j)\times t_i} & 0 \end{pmatrix}, \tag{10.3}$$

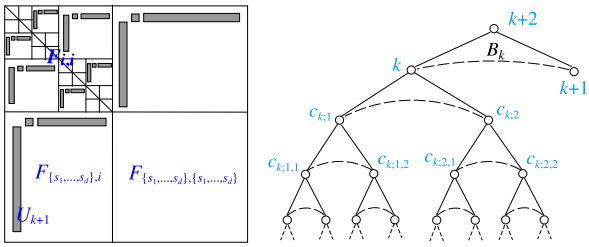


Fig. 10.5 Partial factorization of a frontal matrix $\mathcal{F}_i$ and the HSS tree used. $\mathcal{F}_i$ is shown with its leading block $F_{i,i}$ represented by structured factors. The subtree $T[k]$ is the HSS tree for $F_{i,i}$

where c_1 and c_2 are the children of i. For notational convenience, rewrite $\mathcal{F}_i$ as

$$\mathcal{F}_i \equiv \begin{pmatrix} F_{i,i} & F_{\mathcal{N}_i,i}^T \\ F_{\mathcal{N}_i,i} & F_{\mathcal{N}_i,\mathcal{N}_i} \end{pmatrix}. \tag{10.4}$$

In our structured multifrontal method, we set a *switching level* l_s so that if a separator i is at level l of $\mathcal{T}$ and $l > l_s$, we use exact Cholesky factorizations, and otherwise, we use HSS Cholesky factorizations. Similar to [33], we can show that this can help minimize the cost, which is smaller than using structured factorizations at all levels as in [25] by a factor up to $O(\log n)$. This is justified by Theorem 10.1 below. See Fig. 10.4(ii).

Here, we only need to describe the structured factorization part, which includes:

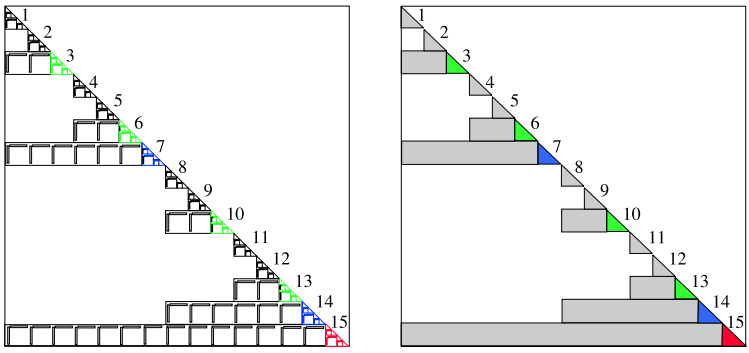
1. Factorizing $\mathcal{F}_i$ with the robust HSS method in Sect. 10.2.2, except that the last diagonal block $F_{\mathcal{N}_i,\mathcal{N}_i}$ is not factorized. Then $F_{i,i} \approx L_{i,i} L_{i,i}^T$.
2. In the meantime, $L_{i,i}^{-1} F_{\mathcal{N}_i,i}$ is compressed into a low-rank form.
3. Computing the update matrix or Schur complement $\mathcal{U}_i$ with a low-rank update.

Then $\mathcal{U}_i$ participates in the extend–add operation to form the parent frontal matrix. The details are elaborated as follows.

In order to perform the partial factorization of $\mathcal{F}_i$, we use a full HSS tree T with $k+2$ nodes, where the left and right children of the root are k and $k+1$ respectively. See Fig. 10.5. The subtree $T[k]$ is used as the HSS tree for completely factorizing $F_{i,i}$, and the single node $k+1$ is for the unfactorized part $F_{\mathcal{N}_i,\mathcal{N}_i}$. The algorithm in [35] is applied to $\mathcal{F}_i$ following the postordering traversal of the nodes of T. The factorization stops after the entire $T[k]$ is visited (or after the Schur complement $\mathcal{U}_i$ is computed). At the point, we have an approximate HSS Cholesky factorization

$$F_{i,i} \approx L_{i,i} L_{i,i}^T, \quad L_{i,i} = \left(\begin{array}{c|c} \left(\begin{array}{c|c} \ddots & 0 \\ \hline U_{c_k,1,2} B_{c_k,1,2} U_{c_k,1,1}^T & \ddots \end{array}\right) & 0 \\ \hline U_{c_k,2} B_{c_k,2} U_{c_k,1}^T & \left(\begin{array}{c|c} \ddots & 0 \\ \hline U_{c_k,2,2} B_{c_k,2,2} U_{c_k,2,1}^T & \ddots \end{array}\right) \end{array}\right), \tag{10.5}$$

where $L_{i,i}$ is a lower-triangular HSS matrix and $c_{k,1}, c_{k,2}, \ldots$ are the children of appropriate nodes as shown in Fig. 10.5.



(i) *Factor from our structured factorization* (ii) *Factor from exact factorization*

Fig. 10.6 The nonzero pattern of a structured multifrontal factor of the matrix A in Fig. 10.3, as compared with the factor from the exact factorization

Then the frontal matrix $\mathcal{F}_j$ in Eq. (10.3) is (approximately) factorized as

$$\mathcal{F}_i \approx \begin{pmatrix} L_{i,i} & \\ L_{\mathcal{N}_i,i} & I \end{pmatrix} \begin{pmatrix} I & \\ & \mathcal{U}_i \end{pmatrix} \begin{pmatrix} L_{i,i}^T & L_{\mathcal{N}_i,i}^T \\ & I \end{pmatrix}, \tag{10.6}$$

where

$$L_{\mathcal{N}_i,i} = U_{k+1} B_k^T U_k^T, \tag{10.7}$$

Note that U_{k+1} and B_k are explicitly available, and U_k is implicitly represented by lower level U and R generators and has orthonormal columns (see Eq. (10.2)). Thus, $\mathcal{U}_i$ can be formed explicitly with a low-rank update

$$\begin{aligned} \mathcal{U}_i &= F_{\mathcal{N}_i,\mathcal{N}_i} - \left(U_{k+1} B_k^T U_k^T\right)\left(U_{k+1} B_k^T U_k^T\right)^T \\ &= F_{\mathcal{N}_i,\mathcal{N}_i} - \left(U_{k+1} B_k^T\right)\left(U_{k+1} B_k^T\right)^T. \end{aligned} \tag{10.8}$$

Here, for simplicity, we keep $\mathcal{U}_i$ as a dense matrix so that the extend–add operation is the same as in the standard supernodal multifrontal method (see Remark 10.2 below for more explanations). According to the idea of Schur compensation in [35], $\mathcal{U}_i$ is roughly equal to the exact Schur complement of $F_{i,i}$ plus a positive semi-definite term, and is always positive definite, in general. $\mathcal{U}_i$ then participates in the construction of the parent frontal matrix just like in the standard multifrontal method. Similarly, the parent frontal matrix is also guaranteed to be positive definite.

This process then proceeds along the assembly tree $\mathcal{T}$. After the elimination, we have an approximate factorization

$$A \approx \mathcal{L}\mathcal{L}^T,$$

where $\mathcal{L}$ always exists and is called a *triangular structured multifrontal factor*. See Fig. 10.6 for an example. $\mathcal{L}$ is associated with two layers of postordering trees, the

outer layer assembly tree $\mathcal{T}$, and an inner layer HSS tree T for each node of $\mathcal{T}$. For a node i of $\mathcal{T}$, we store a triangular HSS form $L_{i,i}$ and also the U and B generators in Eq. (10.8). These are used in the structured multifrontal solution.

10.4.3 Structured Multifrontal Solution

Next, we consider the solution of Eq. (10.1) with the structured multifrontal factor. We solve two structured triangular systems

$$\mathcal{L}y = b, \tag{10.9}$$

$$\mathcal{L}^T x = y. \tag{10.10}$$

For convenience, assume b, x, y are partitioned conformably according to the sizes of the separators. For example, $b = (b_1^T, b_2^T, \ldots, b_K^T)^T$ with the length of b_i equal to the number of mesh points in separator i. Since the situation for a node i at a level l greater than the switching level l_s is trivial (with regular dense solutions), we only focus on structured solutions when describing the algorithm.

The solution of Eq. (10.9) with forward substitution involves forward (or post-ordering) traversal of the assembly tree $\mathcal{T}$. For a node i of $\mathcal{T}$, according to Eq. (10.6), we need to solve a system of the following form for y_i:

$$\begin{pmatrix} L_{i,i} & \\ L_{\mathcal{N}_i,i} & I \end{pmatrix} \begin{pmatrix} y_i \\ \tilde{b}_{\mathcal{N}_i} \end{pmatrix} = \begin{pmatrix} b_i \\ b_{\mathcal{N}_i} \end{pmatrix}, \tag{10.11}$$

where $b_{\mathcal{N}_i}$ is related to $b_{j_1}, b_{j_2}, \ldots, b_{j_d}$ (the separators $j_1, j_2, \ldots, j_d \in \mathcal{N}_i$ are connected to i and partially contribute to $b_{\mathcal{N}_i}$). Here, b_i is either from b (when i is a leaf), or is an updated vector due to the solution steps associated with lower level nodes. (We still use b_i for notational convenience. See Eq. (10.12).)

We first solve $L_{i,i}y_i = b_i$ with a lower triangular HSS solver in [21]. Then $L_{\mathcal{N}_i,i}y_i$ is the contribution of separator i to its neighbors. That is, we update $b_{\mathcal{N}_i}$ by

$$b_{\mathcal{N}_i} \leftarrow b_{\mathcal{N}_i} - L_{\mathcal{N}_i,i}y_i = b_{\mathcal{N}_i} - U_{k+1}\big(B_k^T\big(U_k^T y_i\big)\big), \tag{10.12}$$

where Eq. (10.7) is used. Again, U_{k+1} and B_k are explicitly available, and $U_k^T y_i$ can be quickly computed since it is partially formed in the HSS solution of $L_{i,i}y_i = b_i$ [21]. Thus, $b_{\mathcal{N}_i}$ can be convenient computed, and is then used to update $b_{j_1}, b_{j_2}, \ldots, b_{j_d}$.

In the backward substitution stage for solving Eq. (10.10), we traverse the elimination tree top-down. Similarly for each node i, according to Eqs. (10.6)–(10.7), we need to solve a system of the following form for x_i:

$$\begin{pmatrix} L_{i,i}^T & U_k B_k U_{k+1}^T \\ & I \end{pmatrix} \begin{pmatrix} x_i \\ x_{\mathcal{N}_i} \end{pmatrix} = \begin{pmatrix} y_i \\ x_{\mathcal{N}_i} \end{pmatrix},$$

Algorithm 10.1: Robust structured multifrontal factorization (RSMF)

for *nodes (separators)* $i = 1, 2, \ldots$ *of* $\mathcal{T}$ **do**
  if i *is a leaf* **then**
    form $\mathcal{F}_i \equiv \mathcal{F}_i^0$, where $\mathcal{F}_i^0$ is given in Eq. (10.3)
  if i *is at level* $l > l_s$ **then**
    Compute traditional Cholesky factorization $F_{i,i} = \hat{L}_i \hat{L}_i^T$ of $F_{i,i}$ in Eq. (10.4);
    Compute the Schur complement $\mathcal{U}_i$
  else
    Apply the robust HSS Cholesky factorization to $\mathcal{F}_i$ so that Eqs. (10.5) and (10.7) in Eq. (10.6) are computed;
    Compute $\mathcal{U}_i$ with a low-rank update as in Eq. (10.8)
  if i *is a left node* **then**
    push $\mathcal{U}_i$ onto the update matrix stack
  else
    Pop $\mathcal{U}_j$ from the update matrix stack;
    $\mathcal{F}_p = \mathcal{F}_p^0 \Diamond \mathcal{U}_i \Diamond \mathcal{U}_j$, where p is the parent of i

where $x_{\mathcal{N}_i}$ is already available from the solution steps associated with the upper level separators. This just needs the solution of an upper triangular HSS system $L_{i,i}^T x_i = y_i - U_k(B_k(U_{k+1}^T x_{\mathcal{N}_i}))$.

Note that the space of b can be used to store y and then x. After all the updates and solutions are performed, b is transformed into x.

10.5 Algorithm, Complexity, and Rank Relaxation

The structured multifrontal factorization algorithm is summarized as follows.

In a parallel implementation, we can traverse the assembly tree levelwise. The algorithm can be applied to general sparse SPD matrices. But we only consider its complexity in terms of sparse matrices arising from 2D and 3D discretized PDEs. The detailed flop count uses an idea of rank relaxation. The following lemma is a simple extension of the results in [32].

Lemma 10.1 (Dense rank relaxation) *Suppose an order N matrix F is hierarchically partitioned into $O(\log N)$ levels of HSS blocks following a perfect binary tree. Let $N_l = O(N/2^l)$ be the row dimension of the HSS block rows at level l, and r_l be their maximum numerical rank. Then for a given r_l, a triangular HSS factorization of A can be computed in ξ_{fact} flops, the HSS system can be solved in ξ_{sol} flops, and the HSS form needs memory size σ_{mem}, where the values are given in Table 10.1.*

Table 10.1 Costs and storage of dense-to-triangular-HSS factorization and solution with rank relaxation, where $p \in \mathbb{N}$, and $r = \max r_l$ is the HSS rank

r_l		$r = \max r_l$	ξ_{fact}	ξ_{sol}	σ_{mem}
$O(1)$		$O(1)$	$O(N^2)$	$O(N)$	$O(N)$
$O((\log_2 N_l)^p)$, $p \geq 0$		$O((\log_2 N)^p)$			
$O(N_l^{1/p})$	$p > 3$	$O(N^{1/p})$			
	$p = 3$	$O(N^{1/3})$	$O(N^2)$	$O(N \log N)$	$O(N)$
	$p = 2$	$O(N^{1/2})$	$O(N^2 \log N)$	$O(N^{3/2})$	$O(N \log N)$

Table 10.2 Factorization cost ξ_{fact}, solution cost ξ_{sol}, and storage σ_{mem} of the structured multifrontal method applied to a discretized matrix A of order n on a 2D $n^{1/2} \times n^{1/2}$ mesh, where $p \in \mathbb{N}$

r_l		$r = \max_i \max r_l$	ξ_{fact}	ξ_{sol}	σ_{mem}
$O(1)$		$O(1)$	$O(n \log n)$	$O(n \log \log n)$	$O(n \log \log n)$
$O((\log N_l)^p)$, $p \geq 0$		$O((\log N)^p)$			
$O(N_l^{1/p})$	$p \geq 3$	$O(N^{1/p})$			
	$p = 2$	$O(N^{1/2})$	$O(n \log^2 n)$		

Lemma 10.1 and an extension of the derivations in [31] yield the following results.

Theorem 10.1 (Sparse rank relaxation) *Suppose the robust structured multifrontal factorization method* (*Algorithm* 10.1) *and the solution method* (*Sect.* 10.4.3) *are applied to a discretized matrix A of order n on a regular mesh. Assume each frontal matrix $\mathcal{F}_i$ has order $O(N)$ and is treated as F in Lemma* 10.1 *so that the HSS blocks at level l of the HSS tree of $\mathcal{F}_i$ has row dimension N_l and rank r_l. Let the factorization cost, solution cost, and memory size of the structured multifrontal method be ξ_{fact}, ξ_{sol}, and σ_{mem}, respectively. Then if r_l satisfies the patterns as in Lemma* 10.1,

- *If A is obtained from a 2D $n^{1/2} \times n^{1/2}$ mesh, the results are given in Table* 10.2. *The switching level $l_s = O(\log n^{1/2})$ is chosen so that the* factorization costs *before and after the switching level are the same.*
- *If A is obtained from a 3D $n^{1/3} \times n^{1/3} \times n^{1/3}$ mesh, the results are given in Table* 10.3. *The switching level $l_s = O(\log n^{1/3})$ is chosen so that the* solution costs *before and after the switching level are the same.*

As an example, for problems such as 2D discrete Poisson's equations, it is shown that the maximum rank bound for all nodes i of $\mathcal{T}$ is $r = \max_i \max_l = O(1)$ [4]. Thus, Table 10.2 applies and our solver has nearly linear complexity and nearly linear storage. In contrast, the factorization method in [11] costs $O(n \log^2 n)$. Moreover, Theorem 10.1 indicates that we can relax the rank requirement to get similar complexity. For 3D discrete Poisson's equations, it is shown that $r = O(n^{1/3})$ [4].

Table 10.3 Factorization cost ξ_{fact}, solution cost ξ_{sol}, and storage σ_{mem} of the structured multifrontal method applied to a discretized matrix A of order n on a 3D $n^{1/3} \times n^{1/3} \times n^{1/3}$ mesh, where $p \in \mathbb{N}$

r_l		$r = \max_i \max r_l$	ξ_{fact}	ξ_{sol}	σ_{mem}
$O(1)$		$O(1)$	$O(n^{4/3})$	$O(n)$	$O(n)$
$O((\log_2 N_l)^p)$, $p \geq 0$		$O((\log_2 N)^p)$			
$O(N_l^{1/p})$, $p > 3$		$O(N^{1/p})$			
$O(N_l^{1/p})$	$p = 3$	$O(N^{1/3})$	$O(n^{4/3})$	$O(n \log^{1/2} n)$	$O(n \log^{1/2} n)$
	$p = 2$	$O(N^{1/2})$	$O(n^{4/3} \log n)$	$O(n \log n)$	$O(n \log n)$

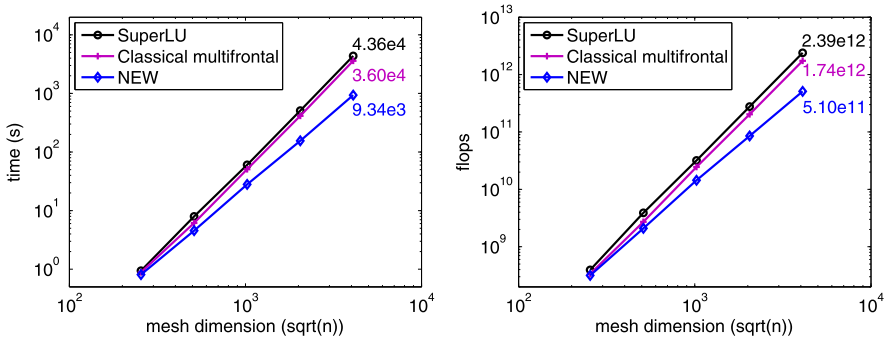
In some numerical tests, the pattern of r_l is observed to follow the last row of Table 10.3.

Remark 10.1 For 2D Helmholtz equations, the rank bound is $r = O(\log n)$ with certain assumptions [11], which only depends on the logarithm of the frequency. Thus, not only our method has nearly $O(n)$ complexity, but also its performance is relatively insensitive to the frequency, because of the rank bound and the rank relaxation. Similar results are also observed for other problems with parameters such as sizes of discontinuities and Poisson's ratios. See Sect. 10.6 for some examples.

Remark 10.2 In our discussions, we keep $\mathcal{U}_i$ as a dense matrix. It turns out that this is at most $O(\log n)$ times slower than a fully structured version where an HSS form of $\mathcal{U}_i$ is used, but it significantly simplifies the descriptions and implementations. Moreover, our solution cost is very close to $O(n)$ (such as $O(n \log \log n) \sim O(n \log^{1/2} n)$ in Theorem 10.1). The storage for a stack needed for the update matrices is about the same as the factor size. A fully structured robust multifrontal solver with HSS form extend–add operations will appear in our future work.

Remark 10.3 Moreover, our method has various other advantages:

- Our method applies to PDEs on irregular grids and general sparse problems. The method in [26, 33] is mainly designed for regular grids, and the one in [25] requires that the mesh is nearly regular, or the separators in the partition roughly follow the layout in a regular grid.
- Our method applies to both 2D and 3D PDEs, while it is not clear how the ones in [11, 25, 33] perform in 3D.
- We use a switching level to optimize the cost, and the factorization cost in 2D is faster than the one in [11] by a factor of $O(\log n)$.
- We incorporate robustness enhancement so that, for an SPD matrix A, the structured factor $\mathcal{L}$ always exists and $\mathcal{L}\mathcal{L}^T$ is positive definite, in general. This does not hold for the methods in [11, 25, 26, 33].
- The algorithm is parallelizable, while the one in [11] is sequential.



(i) Timing (ii) Flops

Fig. 10.7 Performance of the new robust structured multifrontal method (NEW) for Eq. (10.13), as compared with SuperLU [8] and the classical multifrontal method, where NEW uses a relative tolerance $\tau = 10^{-6}$, the total number of levels in $\mathcal{T}$ increases from 14 to 21 when n increases, and there are about nine levels below the switching level l_s

10.6 Numerical Experiments

The method is implemented in Fortran 90, and can work as a fast direct solver. If the low-rank property is insignificant, the method can serve as an efficient and effective preconditioner. We test it on various important discretized PDE examples.

Example 10.1 We first demonstrate the efficiency of the solver for the standard five-point discretized Laplacian from the 2D Poisson equation with a Dirichlet boundary condition:

$$-\Delta \mathbf{u} = \mathbf{f}, \quad \mathbf{u} \in \mathbb{R}^2. \tag{10.13}$$

Here, we let the matrix size n range from 255^2 to 4095^2. Every time n nearly quadruples. See Fig. 10.7 for the timing and flops of the factorizations. We see that the robust structured method is much faster than both SuperLU [8] and the exact multifrontal method when n is large.

The results of the structured solution are shown in Table 10.4. We observe that both the storage and the solution cost scale nearly linearly in terms of n. The accuracy is also well controlled. In addition, with few steps of iterative refinement, the full computer precision is reached.

Example 10.2 Next, we solve a 3D interface problem with jumps in the coefficient:

$$\begin{aligned} -\nabla \cdot (c(\delta)\nabla \mathbf{u}) &= \mathbf{f}, \quad \mathbf{u} \in \mathbb{R}^3, \\ c(\delta) &= 1 \text{ or } \delta. \end{aligned} \tag{10.14}$$

We follow the choice of $c(\delta)$ and the boundary condition in i FEM [6]. The smaller δ is, the more ill conditioned the problem is. As compared with the exact

Table 10.4 Storage (number of nonzero entries in $\mathcal{L}$), solution cost, and relative residual of the new robust structured multifrontal solution for Eq. (10.13)

n	250^2	500^2	1000^3	2000^3	4000^2
Solution time (s)	7.37*e*–2	2.52*e*–1	1.06*e*0	4.38*e*0	1.80*e*1
Solution flops	9.63*e*6	4.62*e*7	2.00*e*8	8.38*e*8	3.54*e*8
Storage	2.55*e*6	1.25*e*7	5.30*e*7	2.19*e*8	9.36*e*8
$\frac{\|Ax-b\|_2}{\|b\|_2}$	7.95*e*–9	1.74*e*–8	2.31*e*–8	2.29*e*–8	1.85*e*–8

Table 10.5 Solution of Eq. (10.14) in 3D (with discontinuities in the coefficient) using the classical multifrontal factorization (MF) and our new robust structured factorization (NEW) with a relative tolerance $\tau = 10^{-3}$, where $\delta = 10^{-8}$

n	Flops				Storage (Number of nonzeros in L)			
	1.70*e*5	2.75*e*5	5.37*e*5	12.7*e*5	1.70*e*5	2.75*e*5	5.37*e*5	12.7*e*5
MF	0.91*e*11	4.21*e*11	12.5*e*11	41.0*e*11	1.11*e*8	2.01*e*8	4.20*e*8	9.21*e*8
NEW	0.80*e*11	2.44*e*11	6.44*e*11	17.2*e*11	1.05*e*8	1.39*e*8	2.34*e*8	4.60*e*8

Table 10.6 Solution of Eq. (10.14) in 3D (with discontinuities in the coefficient) using our new robust structured factorization with a relative tolerance $\tau = 10^{-3}$ for different δ, where the storage is measured by the number of nonzero entries in $\mathcal{L}$

δ	10^{-2}	10^{-4}	10^{-6}	10^{-8}
Flops	9.84*e*11	8.90*e*11	8.15*e*11	6.44*e*11
Storage	2.71*e*8	2.60*e*8	2.50*e*8	2.34*e*8

multifrontal method, our robust structured factorization attains satisfactory speedup with modest accuracy τ. See Table 10.5. We also test the structured method for different δ. Table 10.6 indicates that the performance is relatively insensitive to δ.

Example 10.3 Finally, we consider the preconditioning of a linear elasticity equation

$$-(\mu\Delta\mathbf{u} + (\lambda+\mu)\nabla\nabla\cdot\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega = (0,1)\times(0,1). \tag{10.15}$$

This equation is frequently solved in structural mechanics. Standard solvers including multigrid often suffer from the deterioration of the convergence rate for large Poisson's ratios λ/μ or near the incompressible limit. When λ/μ grows, the condition number of the discretized matrix A grows quickly. Here, we demonstrate the effectiveness of our structured solver as a preconditioner (although direct factorizations may cost less). For A with size $n \approx 1.28 \times 10^6$, we manually specify a numerical rank $r = 40$ in the structured multifrontal preconditioner. The convergence results for λ/μ varying from 1 to 10^6 is shown in Table 10.7. The convergence behavior is illustrated in Fig. 10.8. We observe that the preconditioned conjugate gra-

Table 10.7 Convergence of direct CG and preconditioned CG with our robust structured multifrontal solver (CG-RSMF) as a preconditioner for solving Eq. (10.15), where direct CG is set to stop when certain number of iterations is reached, and N_{iter} is the number of iterations.

	λ/μ	1	10^2	10^4	10^6
Direct CG	N_{iter}	3917	7997	14950	31004
	Flops	1.28e16	2.61e16	4.87e16	1.01e17
	$\frac{\|Ax-b\|_2}{\|b\|_2}$	1.00e–12	1.33e–9	3.92e–9	4.89e–10
CG-RSMF	N_{iter}	40	47	72	141
	Flops	1.31e14	1.54e14	2.36e14	4.61e14
	$\frac{\|Ax-b\|_2}{\|b\|_2}$	8.17e–16	6.54e–16	7.73e–16	4.66e–16

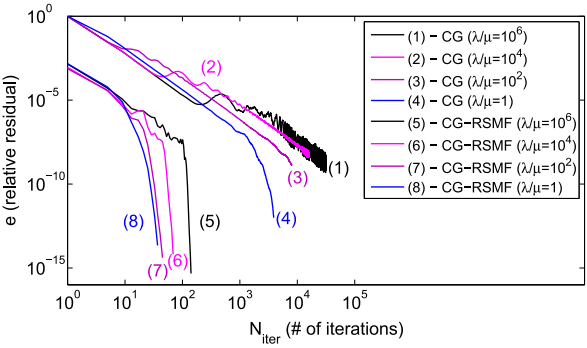


Fig. 10.8 Convergence of direct CG and preconditioned CG with our robust structured multifrontal solver (CG-RSMF) as a preconditioner for solving Eq. (10.15)

dient (CG) method converges quickly for all λ/μ. In comparison, direct CG costs more and has difficulty converging for large λ/μ.

Acknowledgements The author thanks Ming Gu, Xiaoye S. Li, and Jie Shen for some useful discussions, and thanks Zhiqiang Cai and Long Chen for providing two test examples. This research was supported in part by NSF grant CHE-0957024.

References

1. Bebendorf, M.: Efficient inversion of the Galerkin matrix of general second-order elliptic operators with nonsmooth coefficients. Math. Comput. **74**(251), 1179–1199 (2005)
2. Bebendorf, M., Hackbusch, W.: Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with l^∞-coefficients. Numer. Math. **95**, 1–28 (2003)
3. Chandrasekaran, S., Dewilde, P., Gu, M., Lyons, W., Pals, T.: A fast solver for HSS representations via sparse matrices. SIAM J. Matrix Anal. Appl. **29**, 67–81 (2006)
4. Chandrasekaran, S., Dewilde, P., Gu, M., Somasunderam, N.: On the numerical rank of the off-diagonal blocks of Schur complements of discretized elliptic PDEs. SIAM J. Matrix Anal. Appl. **31**, 2261–2290 (2010)
5. Chandrasekaran, S., Gu, M., Pals, T.: A fast ULV decomposition solver for hierarchically semiseparable representations. SIAM J. Matrix Anal. Appl. **28**, 603–622 (2006)

6. Chen, L.: *i*FEM: An innovative finite element methods package in MATLAB. Technical Report (2008). http://math.uci.edu/~chenlong/Papers/iFEMpaper.pdf
7. Davis, T.A., Hu, Y.: The university of Florida sparse matrix collection. ACM Trans. Math. Softw.
8. Demmel, J.W., Gilbert, J.R., Li, X.S.: SuperLU Users' Guide (2003). http://crd.lbl.gov/~xiaoye/SuperLU/
9. Duff, I.S., Reid, J.K.: The multifrontal solution of indefinite sparse symmetric linear. ACM Trans. Math. Softw. **9**, 302–325 (1983)
10. Eidelman, Y., Gohberg, I.C.: On a new class of structured matrices. Integral Equ. Oper. Theory **34**, 293–324 (1999)
11. Engquist, B., Ying, L.: Sweeping preconditioner for the Helmholtz equation: Hierarchical matrix representation. Commun. Pure Appl. Math. **64**, 697–735 (2011)
12. George, A.: Nested dissection of a regular finite element mesh. SIAM J. Numer. Anal. **10**, 345–363 (1973)
13. Gilbert, J.R., Teng, S.H.: MESHPART, a Matlab mesh partitioning and graph separator toolbox (2002). http://aton.cerfacs.fr/algor/Softs/MESHPART/
14. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comp. Physiol. **73**, 325–348 (1987)
15. Hackbusch, W.: A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. Computer **62**, 89–108 (1999)
16. Hackbusch, W., Börm, S.: Data-sparse approximation by adaptive $\mathcal{H}^2$-matrices. Computer **69**, 1–35 (2002)
17. Hoffman, A.J., Martin, M.S., Rose, D.J.: Complexity bounds for regular finite difference and finite element grids. SIAM J. Numer. Anal. **10**, 364–369 (1973)
18. Karypis, G.: METIS: family of multilevel partitioning algorithms (1998). http://glaros.dtc.umn.edu/gkhome/metis/metis/overview
19. Le Borne, S., Grasedyck, L., Kriemann, R.: Domain-decomposition based $\mathcal{H}$-LU preconditioners. Domain Decomposition Methods in Science and Engineering XVI. in O.B. Widlund, D.E. Keyes (Eds) **55**, 661–668 (2006)
20. Liu, J.W.H.: The multifrontal method for sparse matrix solution: Theory and practice. SIAM Rev. **34**, 82–109 (1992)
21. Lyons, W.: Fast algorithms with applications to PDEs. Ph.D. Thesis, UCSB (2005)
22. Parter, S.: The use of linear graphs in Gauss elimination. SIAM Rev. **3**, 119–130 (1961)
23. Polizzi, E., Sameh, A.H.: A parallel hybrid banded system solver: the SPIKE algorithm. Parallel Comput. **32**, 177–194 (2006)
24. Sambavaram, S.R., Sarin, V., Sameh, A.H., Grama, A.: Multipole-based preconditioners for large sparse linear systems. Parallel Comput. **29**, 1261–1273 (2003)
25. Schmitz, P.G., Ying, L.: A fast direct solver for elliptic problems on general meshes in 2D. J. Comput. Phys. (2011). doi:10.1016/j.jcp.2011.10.013
26. Schmitz, P.G., Ying, L.: A fast direct solver for elliptic problems on Cartesian meshes in 3D (2011, submitted). http://www.math.utexas.edu/users/lexing/publications/direct3d.pdf
27. Tewarson, R.P.: On the product form of inverses of sparse matrices. SIAM Rev. **8** (1966)
28. Vandebril, R., Barel, M.V., Golub, G., Mastronardi, N.: A bibliography on semiseparable matrices. Calcolo **42**, 249–270 (2005)
29. Wang, S., Li, X.S., Xia, J., Situ, Y., de Hoop, V.M.: Efficient scalable algorithms for hierarchically semiseparable matrices. Preprint (2011). http://www.math.purdue.edu/~xiaj/work/parhss.pdf
30. Xia, J.: Robust structured multifrontal factorization and preconditioning for discretized PDEs. Preprint (2008). http://www.math.purdue.edu/~xiaj/work/mfprec.pdf
31. Xia, J.: Efficient structured multifrontal factorization for large sparse matrices. Preprint (2010). http://www.math.purdue.edu/~xiaj/work/mfhss.pdf
32. Xia, J.: On the complexity of some hierarchical structured matrices. SIAM J. Matrix Anal. Appl. (2011, submitted). http://www.math.purdue.edu/~xiaj/work/hsscost.pdf

33. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Superfast multifrontal method for large structured linear systems of equations. SIAM J. Matrix Anal. Appl. **31**, 1382–1411 (2009)
34. Xia, J., Chandrasekaran, S., Gu, M., Li, X.S.: Fast algorithms for hierarchically semiseparable matrices. Numer. Linear Algebra Appl. **17**, 953–976 (2010)
35. Xia, J., Gu, M.: Robust approximate Cholesky factorization of rank-structured symmetric positive definite matrices. SIAM J. Matrix Anal. Appl. **31**, 2899–2920 (2010)