A Purely Algebraic Approach to Preconditioning Based on Hierarchical *LU* Factorizations

M. Bebendorf and T. Fischer

Abstract The efficiency of hierarchical matrices depends on the quality of the block partition. We describe a nested dissection partitioning of the matrix into blocks that uses only the matrix graph and requires a logarithmic-linear number of operations. This block partition allows to compute a hierarchical *LU* decomposition with small fill-in. Furthermore, the algebraic approach admits, in contrast to the usual geometric partitioning, general grids for finite element discretization of elliptic boundary value problems.

1 Introduction

We consider large-scale finite element matrices $A \in \mathbb{R}^{I\times I}$, where I is an index set. Such matrices are usually treated by iterative solvers, which may converge slowly due to ill-conditioning. In order to accelerate the convergence, the FE system is preconditioned. We propose a preconditioning technique which is based on an approximated *LU* decomposition.

In the last years the structure of hierarchical matrices ($\mathscr{H}$-matrices) [5, 7, 2] has proved to be able to handle approximations of discrete solution operators of elliptic partial differential boundary value problems. Hierarchical matrices rely on low-rank approximations on each block of a partition P of the set of matrix indices $I \times I$. In order to guarantee the existence of such low-rank approximations, each block $b = t \times s \in P$ has to satisfy either the admissibility condition

$$\min\{\operatorname{diam} X_t, \operatorname{diam} X_s\} \leq \eta \operatorname{dist}(X_t, X_s) \tag{1}$$

Mario Bebendorf and Thomas Fischer
Mathematical Institute, Faculty of Mathematics and Computer Science, University Leipzig, Johannisgasse 26, 04103 Leipzig, e-mail: fischer@math.uni-leipzig.de

or $\min\{|t|,|s|\} \le n_{\min}$ for given parameters $\eta > 0$ and $n_{\min} \in \mathbb{N}$. Here, X_t denotes the support of a cluster t, which is the union of the supports of the basis functions corresponding to the indices in t:

$$X_t := \bigcup_{i\in t} X_i.$$

The partition is normally generated by recursive subdivision of $I \times I$. The recursion stops in blocks which satisfy (1) or which are small enough. For a given partition P the set of $\mathscr{H}$-matrices with blockwise rank k is defined by

$$\mathscr{H}(P,k) := \{M \in \mathbb{R}^{I\times I} : \operatorname{rank} M_b \le k \text{ for all } b \in P\}.$$

In [1] it was proved that the *LU* decomposition of FE matrices of uniformly elliptic operators can be approximated by $\mathscr{H}$-matrices with logarithmic-linear complexity. Up to now it was possible to guarantee logarithmic-linear complexity only for quasi-uniform discretizations and for some special grids (see [6]), since the generated cluster trees had to be balanced with respect to both, geometry and cardinality.

This article treats the set up of the approximated factors L and U in the hierarchical matrix format using only the matrix graph

$$G_A := \{(i,j) \in I \times I : a_{ij} \neq 0\} \tag{2}$$

of A. The construction of the partition is described such that instead of the geometric condition (1) the algebraic admissibility condition

$$\min\{\operatorname{diam} t, \operatorname{diam} s\} \le \eta \operatorname{dist}(t,s). \tag{3}$$

is satisfied on each large enough block, where

$$\operatorname{diam} t := \max_{i,j\in t} d_{ij} \quad \text{and} \quad \operatorname{dist}(t,s) := \min_{i\in t,\, j\in s} d_{ij}.$$

Here, d_{ij} is the shortest path between i and j in the matrix graph.

The power of condition (3) is that it does not involve the geometry of the discretization. Hence, clustering has to account only for the cardinality of the clusters. This directly generalizes the theory of $\mathscr{H}$-matrix approximations to arbitrary grids including adaptively refined ones. Additionally, the algebraic approach allows to minimize the interface in nested dissection reorderings. Since the size of the interface determines the quality of the partition P, one can expect an acceleration of the hierarchical *LU* factorization algorithm. Condition (3), however, involves the distance $\operatorname{dist}(t,s)$ of two clusters t and s in the matrix graph and their diameters $\operatorname{diam} t$ and $\operatorname{diam} s$. The efficient (i.e., with complexity of order $|t| + |s|$) computation of these quantities is a challenge. One should, however, keep in mind that for matrix partitioning it is not required to know their exact values. In this article we will therefore present efficient multilevel algorithms for the computation of approximations of these quantities.

The structure of this article is as follows. The algebraic construction of the cluster tree is presented in the Section 2.1. In the Sections 2.2 and 2.3 we describe the efficient evaluation of (3). The last section contains numerical results which compare $\mathscr{H}$-matrix *LU* factorizations based on geometric and algebraic matrix partitioning with the direct solver PARDISO [11].

2 Algebraic Matrix Partitioning

To construct a partition one usually generates a *cluster tree* $T_I = (V_{T_I}, E_{T_I})$, which is a graph satisfying the following conditions:

1. the index set I is the root of T_I,
2. $t = \cup_{t' \in S_I(t)} t'$ for all $t \in V_{T_I} \backslash \mathscr{L}(T_I)$ and t' are pairwise disjoint,
3. $|S_I(t)| > 1$ for all $t \in V_{T_I} \backslash \mathscr{L}(T_I)$,

where the elements of the set of sons $S_I(t) := \{t' \in V_{T_I} : (t,t') \in E_{T_I}\}$ are pairwise disjoint and $\mathscr{L}(T_I) := \{t \in V_{T_I} : |S_I(t)| = 0\}$ denotes the set of leafs.

Condition (3) does not contain any information about the geometry of the discretization. Hence, the assumption that the cluster tree T_I is geometrically balanced can be omitted, which allows to treat general grids including adaptively refined ones. Therefore, we use a cardinality balanced cluster tree and assume that the diameter of a generated cluster is equivalent to its cardinality in the sense that there are constants $c_1, c_2 > 0$ such that

$$c_1|t| \leq (\operatorname{diam} t)^d \leq c_2|t| \quad \text{for all } t \in T_I. \tag{4}$$

2.1 Algebraic Construction of the Cluster Tree

In order to reduce *fill-in* during *LU* factorization, I is decomposed using the nested dissection method [4]. Nested dissection is based on the matrix graph $G_A = (V,E)$. In each step it partitions the vertex set V into V_1, V_2, S such that V_1, V_2 are of approximately equal size and S separates V_1, V_2 and additionally satisfies $|S| \ll |V_1|$. The vertex sets V_1 and V_2, corresponding to $t_1, t_2 \subset I$, are recursively partitioned, and we achieve a nested dissection cluster tree (see Fig. 1).

Each nested dissection step can be separated in two phases.

(1) The vertices are divided in two disjoint sets V_1', V_2'. The bipartition can be computed using *spectral bisection* based on the Fiedler vector, which is the eigenvector to the second smallest eigenvalue; see [3]. Since computing eigenvectors of large matrices is computationally expensive, multilevel ideas have been introduced to accelerate the process [9]. For this purpose the graph G_A is coarsened into a sequence $G^{(1)}, \ldots, G^{(\kappa)}$ such that $|V| > |V^{(1)}| > \cdots > |V^{(\kappa)}|$. Spectral bisection can then be applied to the smallest graph $G^{(\kappa)}$. The resulting partition P_κ is projected back to

G_A by going through the intermediate partitions $P_{\kappa-1},\dots,P_1$. The partition P_{i+1} can be improved by refinement heuristics such as the Kernighan-Lin algorithm [10].

Subdividing V in this manner in some sense minimizes the *edge cut* C, i.e., a set of edges $C \subset E$ such that $G' = (V, E\backslash C)$ is no longer connected. The size of the edge cut is in $O(|V|^{1-1/d})$.

(2) The vertex set S, which separates V_1 and V_2, is computed. To this end we consider the boundaries

$$\begin{aligned}\partial V_1' &:= \{u \in V_1' : \exists v \in V_2' \text{ and } (u,v) \in E\},\\ \partial V_2' &:= \{v \in V_2' : \exists u \in V_1' \text{ and } (v,u) \in E\}\end{aligned}$$

of V_1' and V_2' and the edge set $E_{12} = \{(u,v) \in E, u \in \partial V_1', v \in \partial V_2'\}$ between $\partial V_1'$ and $\partial V_2'$. The bipartite graph

$$B := (\partial V_1' \cup \partial V_2', E_{12})$$

is constructed which takes $O(|V|^{1-1/d})$ operations.

In order to get a small separator S, the *minimal vertex cover algorithm* [8] is applied to B. A minimal vertex cover for bipartite graphs can be calculated with complexity $O(|V|^{3/2\cdot(1-1/d)})$. Finally, the vertices belonging to the minimal vertex cover are moved out of V_1' and V_2' to S to obtain a partition V_1, V_2, S of V.

Since the partitioning algorithm ensures that the cardinality of each cluster from the same level in T_I is of the same order of magnitude, i.e., $|t| \sim |I|2^{-\ell}$ for $t \in T_I^{(\ell)}$, we can guarantee logarithmic depth of T_I.

Our algorithm extends the nested dissection cluster tree to ensure that every level of the cluster tree stores a partition of the index set I. To this end the separator index set s of the ℓ-th level of T_I is copied to the next level of the cluster tree as long as the cardinality of s is smaller than $|I|2^{-\ell'}$, $\ell' > \ell$. A sequence $s^{(1)}, s^{(2)}, \dots, s^{(\ell'-\ell)}$ of separators is obtained, each containing the same index set but in different levels. We say $s^{(1)}, s^{(2)}, \dots, s^{(\ell'-\ell)}$ have the same *virtual depth* ℓ, the depth of $s^{(1)}$. If $|s| \geq |I|2^{-\ell'}$, the separator s is recursively partitioned as described in phase one of the nested dissection algorithm. In Fig. 2 separators s_0 and s_0' contain the same index sets and are in different levels but have the same virtual depth.

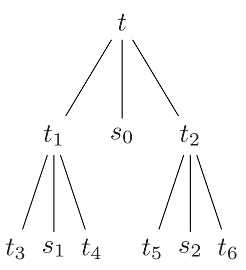

Fig. 1 nested dissection cluster tree

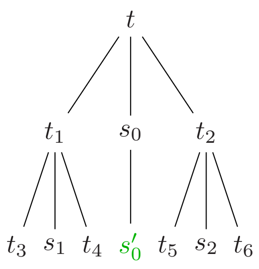

Fig. 2 subtree of T_I rooted at $t \in T_I^{(\ell)}$

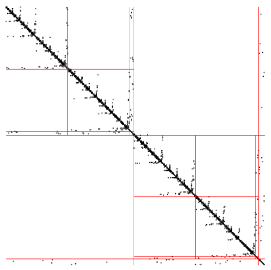
Fig. 3 matrix structure after two nested dissection steps

2.2 The Algebraic Admissibility Condition

In this section we present the evaluation of the admissibility condition (3) based on the cluster tree described in Section 2.1.

Let $t \in T_I^{(\ell-1)}$ be decomposed into $t_1, t_2, s \in T_I^{(\ell)}$, where t_1 and t_2 are separated by s, using the algorithm described in Section 2.1; see Fig. 2. This means that there does not exist any edge between t_1 and t_2. As consequence, $a_{ij} = 0$ and $a_{ji} = 0$ for $i \in t_1, j \in t_2$; see Fig. 3. Most of the information of the matrix is contained in the interface blocks $t_1 \times s$, $t_2 \times s$, $s \times t_1$, $s \times t_2$, and $s \times s$. In order to guarantee logarithmic-linear complexity, these blocks are decomposed into sub-blocks that either can be approximated or are small. The systematical search for pairs of index sets in T_I that form a block which can be approximated by a low rank matrix creates the so called *block cluster tree* $T_{I\times I}$, where the admissible blocks can be found in the leafs $\mathscr{L}(T_{I\times I})$ of $T_{I\times I}$.

In contrast to the usual definition of $T_{I\times I}$, we define the set of sons

$$S_{I\times I}(t \times s) := \begin{cases} \emptyset, & \text{if } S_I(t) = \emptyset \text{ or } S_I(s) = \emptyset, \\ \emptyset, & \text{if } t \neq s \text{ and neither } t \text{ nor } s \text{ are separators,} \\ \emptyset, & \text{if } t \text{ or } s \text{ are separators and satisfy (3),} \\ S_I(t) \times S_I(s), & \text{else.} \end{cases}$$

The block cluster tree is generated by recursively applying $S_{I\times I}$ to the root $I \times I$. The following definition helps to accelerate the admissibility test, i.e., the evaluation of (3).

Definition 1. Two index sets $t_1, t_2 \subset I$ are denoted as *neighbored* if there exists an edge in G_A connecting indices of t and t', i.e., $\exists i \in t_1, \exists j \in t_2$ such that $(i,j) \in E$.

As cluster t is called contiguous if there are two indices $i_{\min}$ and $i_{\max}$ such that

$$t_1 = \{i : i_{\min} \leq i < i_{\max}\}.$$

Note that checking whether two contiguous clusters t_1 and t_2 are neighbored can be be done with $O(\min\{|t_1|, |t_2|\})$ operations.

If $t_1, t_2 \subset I$ are neighbored, it holds that $t_1 \in \mathscr{N}_\eta(t_2)$ and $t_2 \in \mathscr{N}_\eta(t_1)$, where

$$\mathscr{N}_\eta(t) := \{t' \in T_I^{(\ell)} : \operatorname{diam} t > \eta \operatorname{dist}(t, t')\}$$

denotes the *near-field* of t. A block is admissible if $t_2 \notin \mathscr{N}_\eta(t_1)$ or $t_1 \notin \mathscr{N}_\eta(t_2)$. If they are not neighbored, it is necessary to compute the distance $\operatorname{dist}(t_1, t_2)$ between them. The evaluation of $\operatorname{dist}(t_1, t_2)$ involves the computation of $|t_1| \cdot |t_2|$ shortest paths in the matrix graph, each of which takes $O(|I|)$ operations with breadth-first search. Since our aim is to preserve the logarithmic-linear complexity, it is necessary to approximate the distance between t_1 and t_2.

2.3 Approximation of Distance and Diameter

Assume that the father of $t_1 \times t_2 \in T_{I\times I}^{(\ell)}$ is not admissible and t_1, t_2 are not neighbored.

The first step to accelerate the computation of the distance is to calculate all neighbors of the same level in the cluster tree. Assume that the neighbors in the level ℓ of the tree T_I are known. Obviously, for each cluster $t \in T_I^{(\ell)}$ the pairs (t_1, s_0) and (s_0, t_2) (so-called "a-priori neighbors") are neighbored, where $S(t) = \{t_1, s_0, t_2\}$. Since two clusters can be neighbored only if their parents are neighbored, we can restrict the search for neighbors to the set $S_I(t_1) \times S_I(t_2)$, where t_1 and t_2 are neighbored clusters in the ℓ-th level.

Example 1. In Fig. 4 a-priorily known neighbors are symbolized by dashed lines. Computed neighbors are characterized by dotted lines.

In order to compute the approximate distance between $t_1, t_2 \in T_I^{(\ell)}$, we construct a graph G_D. There are predecessors $pre(t_1)$ and $pre(t_2)$ of t_1 and t_2 such that $pre(t_1)$ and $pre(t_2)$ are neighbored. The vertices of G_D consist of the descendants of $pre(t_1)$ and $pre(t_2)$ in the ℓ-th level of T_I. G_D contains a weighted edge between two vertices if and only if the clusters are neighbored. The weight is the difference between ℓ and the virtual depth of the neighbor node. Since $pre(t)$ and $pre(t')$ are neighbored, the graph G_D is connected and it is possible to calculate the approximate distance using Dijkstra's algorithm [12]. In a forthcoming article it is proved that the number of nodes in G_D is bounded from above by a constant.

Example 2. In Fig. 4 assume that $t_7 \times s_0''$ is not admissible. Therefore, the admissibility of the pair (t_{15}, s_0'''), for instance, is checked. Using $pre(t_{15}) = t_3$ and $pre(s_0''') = s_0'$, we obtain the vertex set $\{t_{15}, s_7, t_{16}, s_3', t_{17}, s_8, t_{18}, s_0'''\}$ of G_D, which is depicted in Fig. 5. Dijkstra's algorithm results in an approximate distance between t_{15} and s_0''' of seven.

This approach can be improved by the following iterative refinement procedure. Dijkstra's algorithm not only computes the distance between t_1 and t_2 but also the nodes of the shortest path. We construct a new graph consisting of vertices from level $\ell + m$ for some m rooted at the shortest path nodes. Its edges are defined as in the previous graph G_D. The computation of the shortest path between t_1 and t_2 in this graph will lead to an improved approximation of $\operatorname{dist}(t_1, t_2)$.

Example 3. Assume in Example 2 that Dijkstra's algorithm calculated the shortest path $t_{15}, s_3', t_{18}, s_0'''$. The subtrees rooted at the path nodes are depicted in Fig. 6. We choose vertices of level $\ell + 2$ and determine the edge set considering the neighbors; see Fig. 7. Dijkstra's algorithm is then applied to this refined graph.

It remains to compute an approximation to the diameter of a cluster t. This can be done by a breadth-first search [12]. The result is bounded from below by the radius $r(t) := \min_{i\in t} \max_{j\in t} d_{ij}$ of t and bounded from above by $\operatorname{diam} t$.

In a forthcoming article we prove that it is possible to generate the approximate *LU* factorization in almost linear time using this matrix.

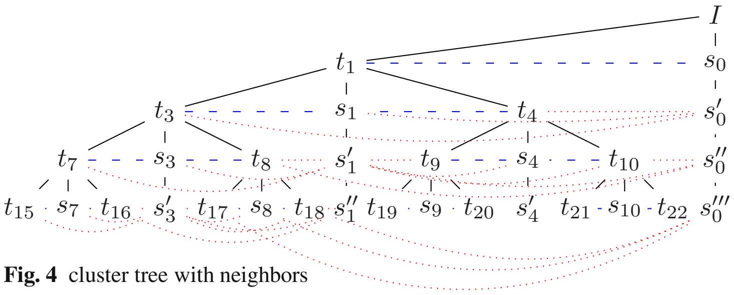
Fig. 4 cluster tree with neighbors

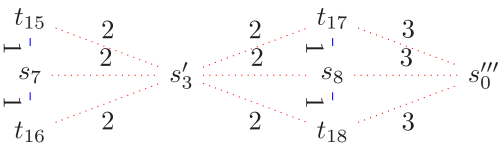
Fig. 5 G_D for computing the approximation for $\mathrm{dist}(t_{15}, s_0''')$

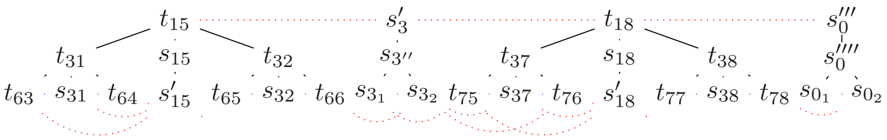
Fig. 6 subtrees and neighbors between clusters

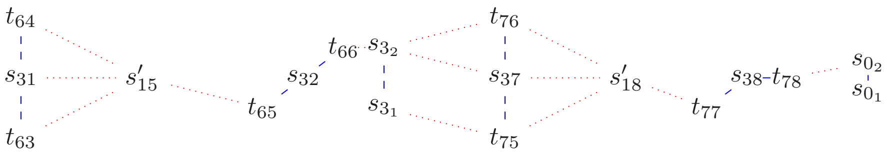
Fig. 7 refined graph

3 Numerical Results

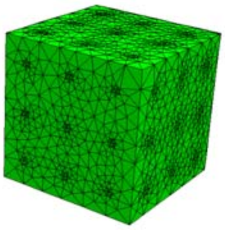
Fig. 8 Computational domain

size	algebraic			PCG		geometric		PARDISO	
	partitioning	$\mathscr{H}$-Cholesky				$\mathscr{H}$-Cholesky		Cholesky	
	t in s	t in s	MB	#it	t in s	t in s	MB	t in s	MB
32 429	0.72	0.86	25	18	0.37	1.85	29	0.50	38
101 296	3.06	3.42	77	23	1.62	8.68	105	4.17	198
658 609	25.35	31.38	726	35	19.38	91.61	805	147.81	2659
2 539 954	106.40	158.68	2920	61	142.27	471.54	3507	–	–

Table 1 Algebraic $\mathscr{H}$-Cholesky preconditioner, geometric $\mathscr{H}$-Cholesky preconditioner, PARDISO Cholesky factorization

The results shown in Table 1 were obtained for the Laplacian on the computational domain shown in Fig. 8. The computation were done on an Intel Xeon 3.0 GHz with 16 GB of core memory. The time required to compute the matrix partition based on the algebraic admissibility condition (1) scales almost linearly with the number of degrees of freedom. The accuracy of the approximate *LU* factorization was chosen to $\delta = 0.5$. Compared with the usual geometric approach to matrix partitioning, the algebraic method leads to a significantly faster computation of the preconditioner. Additionally, the memory consumption of the approximation is reduced.

References

1. Bebendorf, M.: Why finite element discretizations can be factored by triangular hierarchical matrices. SIAM J. Num. Anal. **45**(4), 1472–1494 (2007)
2. Bebendorf, M.: Hierarchical matrices: a means to efficiently solve elliptic boundary value problems, *LNCSE*, vol. 63. Springer (2008)
3. Fiedler, M.: A property of eigenvectors of nonnegative symmetric matrices and its application to graph theory. Czech. Math. J. **25**, 619–633 (1975)
4. George, A.: Nested dissection of a regular finite element mesh. SIAM J. Numer. Anal. **10**(2), 345–363 (1973)
5. Hackbusch, W.: A sparse matrix arithmetic based on $\mathscr{H}$-matrices. Part I: Introduction to $\mathscr{H}$-matrices. Computing **62**(2), 89–108 (1999)
6. Hackbusch, W., Khoromskij, B.N.: $\mathscr{H}$-matrix approximation on graded meshes. In: J.R. Whiteman (ed.) The Mathematics of Finite Elements and Applications X, pp. 307–316. Elsevier (2000)
7. Hackbusch, W., Khoromskij, B.N.: A sparse $\mathscr{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. Computing **64**(1), 21–47 (2000)
8. Hopcroft, J.E., Karp, R.M.: An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing **2**(4), 225–231 (1973)
9. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing **20**(1), 359–392 (1999)
10. Kernighan, B., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal **29** (1970)
11. Schenk, O., Gärtner, K.: Solving unsymmetric sparse systems of linear equations with PARDISO. Future Gener. Comput. Syst. **20**(3), 475–487 (2004)
12. Sedgewick, R.: Part 5, graph algorithms. In: Algorithms in C, 5, 3 edn. Addison-Wesley (2002)