Domain-decomposition based $\mathcal{H}$-LU preconditioners

Sabine Le Borne[1], Lars Grasedyck[2], and Ronald Kriemann[2]

[1] Tennessee Technological University, Cookeville, TN 38505. This material is based upon work supported by the US Department of Energy under Grant No. DE-FG02-04ER25649. sleborne@tntech.edu

[2] Max-Planck-Institute for Mathematics in the Sciences, Leipzig, Germany lgr,rok@mis.mpg.de

1 Introduction

Hierarchical matrices (in short: $\mathcal{H}$-matrices) have first been introduced in 1998 [Hac99] and since then have entered into a wide range of applications. They provide a format for the data-sparse representation of fully populated matrices. The key idea is to approximate certain subblocks of a matrix by low rank approximations which are represented by a product of two low rank matrices: Let $A \in R^{n\times n}$ with rank(A)=k and $k \ll n$. Then there exist matrices $B, C \in R^{n\times k}$ such that $A = BC^T$. Whereas A has n^2 entries, B and C together have $2kn$ entries which results in significant savings in storage if $k \ll n$. A new $\mathcal{H}$-matrix arithmetic has been developed which allows (approximate) matrix-vector multiplication and matrix-matrix operations such as addition, multiplication and inversion of matrices in this format in nearly optimal complexity $\mathcal{O}(n \log^\alpha n)$ [GH03].

In finite element methods, the stiffness matrix is sparse but its inverse is fully populated and can be approximated by an $\mathcal{H}$-matrix. Such an approximate inverse may then be used as a preconditioner in iterative methods [LeB04]. Even though the complexity of the $\mathcal{H}$-matrix inversion is nearly optimal, there are relatively large constants involved in these complexity estimates which in the past have prevented $\mathcal{H}$-matrix based preconditioners to be competitive with other state-of-art methods. The following recent developments have addressed this drawback successfully and allowed $\mathcal{H}$-matrix based preconditioners to be competitive also in the FEM context: 1) a weak admissibility condition [HKK04] yielding coarser block structures and therefore reduced constants in the complexity estimates, 2) the introduction of an $\mathcal{H}$-LU decomposition [Lin04, GL04] which is computed significantly faster than an approximate inverse and provides an (in general more accurate) preconditioner, and 3) the parallelization of $\mathcal{H}$-matrix arithmetic [Kri05]. In this paper, we will add another improvement to these three components: We will

introduce (recursive) domain decompositions with an interior boundary, also known as *nested dissection*, into the construction of the index cluster tree of an $\mathcal{H}$-matrix. The new clustering algorithm will yield a block structure in which large subblocks are zero and remain zero in a subsequent LU-factorization, As a result, the constants in the (nearly optimal) storage and work complexities will be significantly smaller than for the standard $\mathcal{H}$-matrix setting. Furthermore, the $\mathcal{H}$-LU factorization is parallelizable. We will then construct preconditioners based on such an incomplete $\mathcal{H}$-LU-decomposition to accelerate the iterative solution of linear systems of equations. We will illustrate our new preconditioner with some numerical examples for convection-dominated partial differential equations.

The remainder of this paper is structured as follows: Section 2 is devoted to preliminaries and will provide a review of the nested dissection method as well as a brief introduction to $\mathcal{H}$-matrices. Section 3 introduces the new clustering algorithm. In Section 4, we will conclude with some numerical results for convection-dominated problems.

2 Preliminaries: Nested dissection and $\mathcal{H}$-matrices

2.1 A review of nested dissection

Most direct methods for sparse linear systems perform an LU factorization of the original matrix after some reordering of the indices in order to reduce fill-ins. One such popular reordering method is the so-called *nested dissection* which exploits the concept of separation. The idea of nested dissection has been introduced more than 30 years ago [Geo73] and since then attracted considerable attention (see, e.g., [BT02, HR98] and the references therein). The main idea is to separate a (matrix) graph into three parts, two of which have no coupling between each other. The third one, referred to as an interior boundary or separator, contains couplings with (possibly both of) the other two parts. The nodes of the separator are numbered last. This process is then repeated recursively in each subgraph. An illustration of the resulting sparsity pattern is shown in Figure 1 for the first decomposition step. In

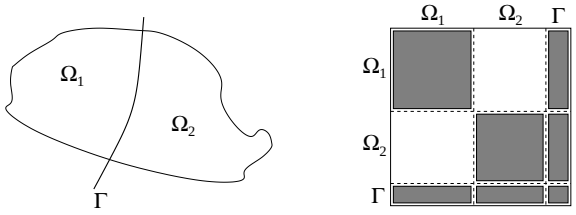

Fig. 1. Nested dissection and resulting matrix sparsity structure

domain-decomposition terminology, we recursively subdivide our domain into two disjoint subdomains and an interior boundary.

A favorable property of such an ordering is that a subsequent LU factorization maintains a major part of this sparsity structure, i.e., there occurs no fill-in in the large, off-diagonal zero matrix blocks. In fact, in the case of regular two-dimensional grids, the computational complexity amounts to $\mathcal{O}(n^{1.5})$ for a matrix $A \in R^{n \times n}$. In order to obtain a (nearly) optimal complexity, we approximate all nonzero, off-diagonal blocks in $\mathcal{H}$-matrix representation and compute them using $\mathcal{H}$-matrix arithmetic. The blocks along the diagonal and the corresponding LU factorizations will be stored as full matrices.

2.2 A brief introduction to $\mathcal{H}$-matrices

An $\mathcal{H}$-matrix approximation to a given (full) matrix is obtained by replacing certain blocks of the matrix by matrices of low rank, stored in so-called Rk-format defined in Definition 3. The formal definition of an $\mathcal{H}$-matrix depends on appropriate hierarchical partitionings of the index set which is organized in a cluster tree. Instead of a fixed partitioning, such a tree provides a hierarchy of partitions leading to a more flexible structure.

Definition 1 (Cluster tree). *Let I be a finite index set and let $T_I = (V, E)$ be a tree with vertex set V and edge set E. For a vertex $v \in V$, we define the set of sons of v as $S(v) := \{w \in V \mid (v, w) \in E\}$. Correspondingly, the father of a non-root vertex v is defined as the unique vertex $F(v)$ s.t. $(F(v), v) \in E$. The tree T_I is called a cluster tree of I if its vertices consist of subsets of I and satisfy the following conditions:*

1. *$I \in V$ is the root of T_I and $v \subset I$, $v \neq \emptyset$, for all $v \in V$.*
2. *For all $v \in V$ there either holds $S(v) = \emptyset$ or $v = \dot{\bigcup}_{w \in S(v)} w$.*

In the following we identify V and T_I, i.e., we write $v \in T_I$ instead of $v \in V$. The nodes $v \in V$ are called clusters. The nodes with no successors are called leaves and define the set $\mathcal{L}(T) = \{v \in T \mid S(v) = \emptyset\}$.

Several approaches to construct a cluster tree have been suggested in previous papers [Hac99, HK00, GH03, GHL03]. All these constructions considered the cardinalities and/or the geometries of the resulting clusters. These constructions have in common that a cluster is either not subdivided (a *leaf*) or has exactly two sons. In Section 3, we will derive a new clustering algorithm in which clusters may have up to three sons, and thus obtain completely new cluster trees and subsequent partitions.

A hierarchy of block partitionings of the product index set $I \times I$ is based upon a cluster tree T_I and is organized in a block cluster tree $T_{I \times I}$:

Definition 2 (Block cluster tree). *Let T_I be a cluster tree of the index set I. A cluster tree $T_{I \times I}$ is called a block cluster tree (based upon T_I) if for all $v \in T_{I \times I}$ there exist $t, s \in T_I$ such that $v = t \times s$. The nodes $v \in T_{I \times I}$ are called block clusters.*

A block cluster tree may be constructed from a given cluster tree in the following canonical way. Here, the admissibility condition $Adm : T_{I\times I} \to \{True, False\}$ is a boolean function which we will specify in more detail later. Given a cluster tree T_I, we construct the block cluster tree $T_{I\times I}$ by $\mathrm{root}(T_{I\times I}) := I \times I$, and each vertex $s \times t \in T_{I\times I}$ has the set of successors

$$S(s\times t) := \begin{cases} \emptyset & \text{if } Adm(s\times t) = \text{True;} \\ \emptyset & \text{if } \min\{\#t, \#s\} \le n_{\min}; \\ \{s' \times t' \mid s' \in S(s), t' \in S(t);\} & \text{otherwise.} \end{cases} \quad (1)$$

The parameter n_{min} ensures that blocks do not become so small that the matrix arithmetic of a full matrix is more efficient. It is typically set to $n_{min} = 32$ or even $n_{min} = 64$. The leaves of a block cluster tree obtained through this construction yield a disjoint partition of the product index set $I \times I$. Matrix blocks which correspond to admissible block clusters will be approximated by low rank matrices in the following Rk-matrix representation:

Definition 3 (Rk-matrix representation). *Let $k, n, m \in N_0$. Let $M \in R^{n\times m}$ be a matrix of at most rank k. A representation of M in factorized form*

$$M = AB^T, \qquad A \in R^{n\times k}, B \in R^{m\times k}, \quad (2)$$

with A and B stored in full matrix representation, is called an Rk-matrix representation of M, or, in short, we call M an Rk-matrix.

If the rank k is small compared to the matrix size given by n and m, we obtain considerable savings in the storage and work complexities of an Rk-matrix compared to a full matrix [GH03].

A *standard* (or *strong*) admissibility condition has been employed in most previous papers [Hac99, HK00, GH03, GHL03, LeB04] and is given by

$$Adm_s(s\times t) = True \quad :\Leftrightarrow \quad min(diam(s), diam(t)) \le \eta\ dist(s,t) \quad (3)$$

for some $0 < \eta$. Here, "diam" and "dist" denote the Euclidean diameter/distance of the (union of the) supports of the basis functions with indices in s, t, resp. A weaker admissibility condition which yields smaller constants in (storage and work) complexities for $\mathcal{H}$-matrices has been introduced and analyzed in [HKK04]. It is given by

$$Adm_w(s\times t) = True \quad :\Leftrightarrow \quad s \ne t. \quad (4)$$

The block partition which is provided by the leaves of a block cluster tree is used to define an $\mathcal{H}$-matrix as follows.

Definition 4 ($\mathcal{H}$-matrix). *Let $k, n_{\min} \in N_0$. The set of $\mathcal{H}$-matrices induced by a block cluster tree $T := T_{I\times I}$ with blockwise rank k and minimum block size $n_{\min}$ is defined by $\mathcal{H}(T,k) := \{M \in R^{I\times I} \mid \forall t\times s \in \mathcal{L}(T) : \mathrm{rank}(M|_{t\times s}) \le k$ or $\min\{\#t, \#s\} \le n_{\min}\}$. Blocks $M|_{t\times s}$ with $\mathrm{rank}(M|_{t\times s}) \le k$ are stored as Rk-matrices whereas all other blocks are stored as full matrices.*

It is our goal to approximate an LU-factorization of a (stiffness) matrix by $\mathcal{H}$-matrices $L^{\mathcal{H}}, U^{\mathcal{H}}$. The storage and computational complexities and also the accuracy of such an $\mathcal{H}$-LU factorization depend strongly on the construction of the cluster tree, i.e., the hierarchy of index set partitionings. In the following Section 3 we will derive a new index clustering algorithm which will permit a subsequent $\mathcal{H}$-LU factorization in which 1) large blocks remain zero, 2) non-zero off-diagonal blocks can be approximated in $\mathcal{H}$-matrix format, and 3) the factorization process can be parallelized. The actual $\mathcal{H}$-LU factorization is defined recursively in the block structure and has been derived in [Lin04, GL04] for matrices arising in finite element methods.

3 A new domain decomposition clustering algorithm

In [Hac03], a direct domain decomposition method is combined with the hierarchical matrix technique. In particular, a domain Ω is subdivided into p subdomains and an interior boundary Γ which separates the subdomains. Within each subdomain, standard $\mathcal{H}$-matrix techniques are used, i.e., $\mathcal{H}$-matrices are constructed by the standard index clustering with zero or two subsets. Here, we propose to use the canonical block cluster tree construction starting from a different cluster tree which will be derived below. The new idea is *not* to distinguish between the index clustering which the domain decomposition yields and the index clustering needed for the $\mathcal{H}$-matrix construction, but to unify these two clusterings.

In Figure 1, the two subdomains Ω_1, Ω_2 are not admissible w.r.t. (3), but since these blocks remain zero during the LU-factorization, we should admit them (rank zero). Thus, we distinguish between the sets of domain-clusters $\mathcal{C}_{\text{dom}}$ and interface clusters $\mathcal{C}_{\text{int}}$ in order to define the admissibility.

We assume some underlying domain decomposition algorithm (e.g., nested dissection) which divides an index set into three disjoint subsets of indices: $I(\Omega_1)$ consists of indices in the first subdomain Ω_1, $I(\Omega_2)$ consists of indices in the second subdomain Ω_2, and $I(\Gamma)$ consists of indices of an interior boundary and separates the index sets $I(\Omega_1)$ and $I(\Omega_2)$, i.e., matrix entries a_{ij} equal zero if $i \in I(\Omega_k)$ and $j \in I(\Omega_l)$ for $k \neq l$. Any interior boundary Γ is bisected into Γ_1, Γ_2 with corresponding index sets $I(\Gamma_1)$, $I(\Gamma_2)$. Based upon such a domain decomposition, we construct the cluster tree from the root to the leaves as follows: We initialize $root(T_I) := I$, $\mathcal{C}_{\text{dom}} := \{I\}$, $\mathcal{C}_{\text{int}} := \{\}$. Each cluster $v \in T_I \cap \mathcal{C}_{\text{dom}}$ with $\#v > n_{\min}$ is subdivided by the rule

$$S(v) := \{I(\Omega_1), I(\Omega_2), I(\Gamma)\} \tag{5}$$

and we add the sons to the corresponding sets of clusters:

$$\mathcal{C}_{\text{dom}} := \mathcal{C}_{\text{dom}} \cup \{I(\Omega_1), I(\Omega_2)\}, \quad \mathcal{C}_{\text{int}} := \mathcal{C}_{\text{int}} \cup \{I(\Gamma)\}.$$

Each node $v \in T_I \cap \mathcal{C}_{\text{int}}$ is subdivided by the rule

$$S(v) := \{w\},\ S(w) := \{I(\Gamma_1), I(\Gamma_2)\}, \qquad \mathcal{C}_{\text{int}} := \mathcal{C}_{\text{int}} \cup S(w), \tag{6}$$

where the cluster w is given by $w := v$, i.e., the boundary Γ corresponding to v is (eventually) split into two subsets and $v = I(\Gamma_1) \cup I(\Gamma_2)$.

Example 1. Figure 2 gives an illustration of the new clustering applied to the 25 vertices (indices) of a regular triangulation of the unit square.

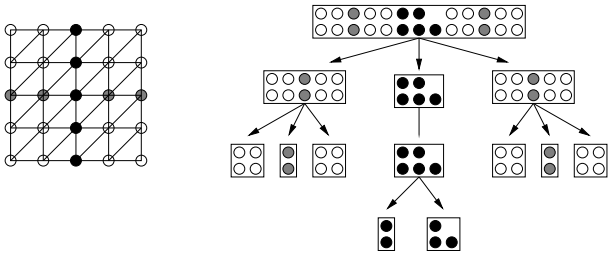

Fig. 2. Example for the new index clustering with $n_{min} = 4$.

Remark 1. In the new clustering, an index cluster v is either subdivided into the three clusters (5) corresponding to indices in the two subdomains and the interior boundary, resp., or it is "subdivided" only every second step by a simple bisection (6). The latter only happens for clusters corresponding to interior boundaries. This is motivated by the underlying geometry of two-dimensional subdomains versus one-dimensional interior boundaries. Roughly, two subdivision steps decrease the Euclidean diameters by a factor of two for both subdomains and the interior boundary (which is effectively only subdivided once). This has a favorable impact on the resulting $\mathcal{H}$-matrix structure in terms of its storage requirements and approximation accuracy.

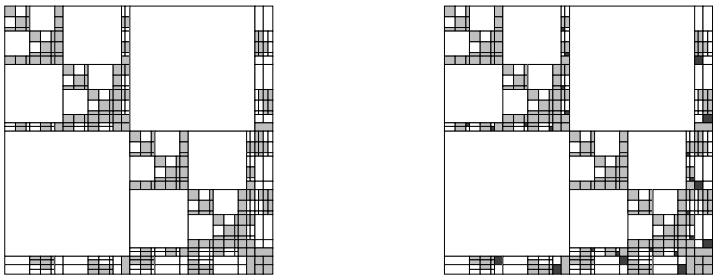

Fig. 3. Example for a domain decomposition $\mathcal{H}$-matrix (left) and a single precision $\mathcal{H}$-LU decomposition (right). Gray blocks correspond to full submatrices and black blocks represent non-zero Rk-matrices.

The block cluster tree $T_{I\times I}$ is build from the new cluster tree T_I by (1), where a pair (t, s) of clusters is admissible, if they are admissible with respect to (3) or if both are domain clusters: $t, s \in \mathcal{C}_{\text{dom}}$. A typical structure of the resulting $\mathcal{H}$-matrix and its $\mathcal{H}$-LU decomposition is plotted in Figure 3. The approximation of non-zero, off-diagonal blocks by Rk-matrices will yield the order reduction from $\mathcal{O}(n^{1.5})$ (exact LU based on nested dissection) down to $\mathcal{O}(n \log n)$ (approximate $\mathcal{H}$-LU). Savings for three-dimensional problems (to which the new clustering generalizes easily) are even more significant and will be illustrated in a forthcoming paper.

4 Numerical results

We will present numerical results of the domain-decomposition based $\mathcal{H}$-LU preconditioner applied to the convection-diffusion equation

$$-\epsilon \Delta u + b(x, y) \cdot \nabla u = f \qquad \text{in } \Omega = [-1, 1] \times [-1, 1]$$

with recirculating flow $b(x, y) = (4x(x-1)(1-2y), -4y(y-1)(1-2x))$ and $\epsilon = 10^{-8}$. In all the following numerical experiments, we set $\eta = 4.0$ in the admissibility condition (3), and we choose adaptive ranks to enforce certain accuracies of the local Rk-blocks. In particular, we choose the rank k of a given Rk-block such that $\sigma(k) \leq a \cdot \sigma(1)$ where $\sigma(j)$ denotes the j'th singular value, and we show numerical results for relative accuracies $a \in \{0.1, 0.25, 0.126, 0.0625\}$. The following examples have been computed on a DELL Precision 530 workstation (2.4 GHz, 4GB memory).

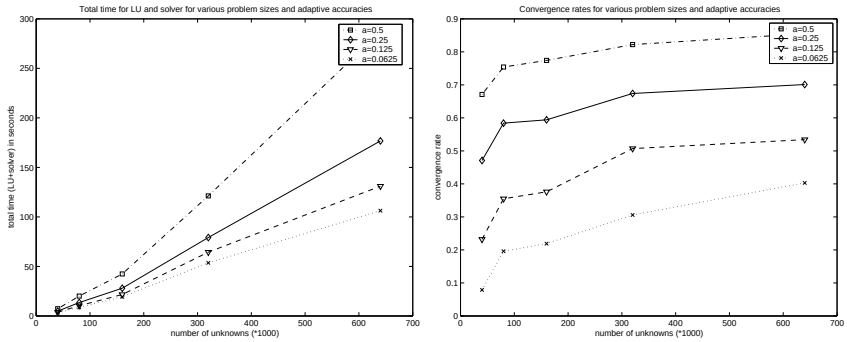

Fig. 4. Total time for the $\mathcal{H}$-LU decomposition and iterative solver (left) and corresponding convergence rates (right)

In Figure 4, on the left we show the time (in seconds) to compute the $\mathcal{H}$-LU decomposition and the subsequent iterative solution depending on the

problem size n (starting from $n = 40000$ up to $n = 640000$) for various adaptive accuracies a. Here, we have used the $\mathcal{H}$-LU preconditioner in a bicg-stab iteration, and we stopped the iteration when the residual had been reduced by 10^{-6}. We note that the $\mathcal{H}$-LU preconditioner with higher accuracy a leads to significantly faster convergence, especially for larger problem sizes. The highest accuracy $a = 0.0625$ yields the overall fastest method for the larger problem sizes. The convergence rates improve significantly with higher accuracy a, indicating that for a given problem size, we are able to construct very efficient $\mathcal{H}$-LU preconditioners by increasing the relative accuracy a.

References

[BT02] I. Brainman and S. Toledo. Nested-dissection orderings for sparse LU with partial pivoting. *SIAM J. Mat. Anal. Appl.*, (23):998–1012, 2002.

[Geo73] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, (10):345–363, 1973.

[GH03] L. Grasedyck and W. Hackbusch. Construction and arithmetics of $\mathcal{H}$-matrices. *Computing*, 70:295–334, 2003.

[GHL03] L. Grasedyck, W. Hackbusch, and S. Le Borne. Adaptive geometrically balanced clustering of $\mathcal{H}$-matrices. *Computing*, 73:1–23, 2003.

[GL04] L. Grasedyck and S. LeBorne. H-matrix preconditioners in convection-dominated problems. *SIAM J. Mat. Anal.*, 2004. submitted.

[Hac99] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. Part I: Introduction to $\mathcal{H}$-matrices. *Computing*, 62:89–108, 1999.

[Hac03] W. Hackbusch. Direct domain decomposition using the hierarchical matrix technique. In I. Herrera, D. Keyes, O. Widlund, and R. Yates, editors, *Domain Decomposition Methods in Science and Engineering*, pages 39–50. UNAM, 2003. available electronically at www.ddm.org/DD14/.

[HK00] W. Hackbusch and B.N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems. *Computing*, 64:21–47, 2000.

[HKK04] W. Hackbusch, B.N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73:207–243, 2004.

[HR98] B. Hendrickson and E. Rothberg. Improving the run time and quality of nested dissection ordering. *SIAM J. Sci. Comp.*, (20):468–489, 1998.

[Kri05] R. Kriemann. Parallel H-matrix arithmetics on shared memory systems. *Computing*, 2005. to appear.

[LeB04] S. LeBorne. Hierarchical matrices for convection-dominated problems. In *Domain Decomposition Methods in Science and Engineering*, volume 40 of *LNCSE*, pages 631–638, 2004. online available at ddm.org.

[Lin04] M. Lintner. The eigenvalue problem for the 2D Laplacian in H-matrix arithmetic and application to the heat and wave equation. *Computing*, 72:293–323, 2004.