$\mathcal{H}$-matrix preconditioners in convection-dominated problems

by

Sabine Le Borne and Lars Grasedyck

Preprint no.: 62 2004

$\mathcal{H}$-MATRIX PRECONDITIONERS IN CONVECTION-DOMINATED PROBLEMS

SABINE LE BORNE* AND LARS GRASEDYCK†

Abstract. Hierarchical matrices provide a data-sparse way to approximate fully populated matrices. In this paper we exploit $\mathcal{H}$-matrix techniques to approximate the LU-decompositions of stiffness matrices as they appear in (finite element or finite difference) discretizations of convection-dominated elliptic partial differential equations. These sparse $\mathcal{H}$-matrix approximations may then be used as preconditioners in iterative methods. Whereas the approximation of the matrix inverse by an $\mathcal{H}$-matrix requires some modification in the underlying index clustering when applied to convection-dominant problems [11], the $\mathcal{H}$-LU-decomposition works well in the standard $\mathcal{H}$-matrix setting even in the convection dominant case. We will complement our theoretical analysis with some numerical examples.

Key words. Hierarchical matrices, data-sparse approximation, preconditioning, convection-dominant problems

AMS subject classifications. 65F05, 65F30, 65F50

1. Introduction.

In a series of papers, the technique of hierarchical matrices, or $\mathcal{H}$-matrices in short, has been introduced [3, 4, 6, 7, 8]. $\mathcal{H}$-matrices provide an inexpensive but sufficiently accurate approximation to fully populated matrices as they appear in boundary element methods or finite element methods. In finite element methods, it is the inverse of the stiffness matrix as well as the matrices L and U in an LU-decomposition which are typically fully populated.

An $\mathcal{H}$-matrix approximation is based on a certain hierarchical block structure of the matrix in which some (preferably large) off-diagonal blocks are represented by low-rank approximations. Such a block structure and the resulting approximations for (inverse) matrices arising in the discretization of uniformly elliptic PDEs can be computed and stored in almost linear complexity, i.e., $\mathcal{O}(n \log_2^\alpha n)$ with moderate parameter α [1]. The constants in these complexity estimates depend on the structure of the involved $\mathcal{H}$-matrices and have been computed exactly in [4].

In a previous paper [11], the approximation quality for standard $\mathcal{H}$-matrices has been analyzed when applied to the inverse matrices that stem from convection-dominated problems with constant convection. Some modifications have been derived in which the direction and magnitude of the convection influence the block structure of the resulting $\mathcal{H}$-matrix, leading to significant improvements in the approximation by an $\mathcal{H}$-matrix compared to the standard setting.

In the case of a *non-constant* dominant convection, the standard $\mathcal{H}$-matrix turns out to provide a satisfactory approximation which is due to the numerical diffusion induced in a discretization.

In this paper, after a brief introduction to $\mathcal{H}$-matrices in Section 2, we will use $\mathcal{H}$-matrices to approximate the matrix factors in an LU-decomposition of the stiffness matrix. In [12], an analysis is provided for the Laplace equation in two spatial dimensions. In Section 3, we will describe the recursive construction of an LU-decomposition using $\mathcal{H}$-matrix arithmetic, and then we focus on $\mathcal{H}$-LU-decompositions for the stiffness matrix of the convection-dominant convection-diffusion equation, both for con-

*Department of Mathematics, Box 5054, Tennessee Technological University, Cookeville, TN 38505 (sleborne@tntech.edu).

†Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany (lgr@mis.mpg.de).

stant and non-constant convection directions. We will use the $\mathcal{H}$-matrix approximations of the LU-decomposition as preconditioners in Krylov-accelerated iterative methods. In Section 4, we will provide some numerical tests to illustrate the efficiency of the resulting method.

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 as will be further explained below. The formal definition of an $\mathcal{H}$-matrix depends on appropriate hierarchical partitionings of the index set and also of the product index set which are organized in (block) cluster trees as defined next. Instead of fixed partitionings, these trees will provide hierarchies of partitionings which gives a hierarchical matrix its name.

DEFINITION 2.1 (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 $S(v) := \{w \in V \mid (v, w) \in E\}$. 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.

For regular grids one can construct the cluster tree T_I in a cardinality balanced way, i.e., an index cluster is divided into a certain number of sons of approximately the same size with respect to the number of indices (see [6, 8]). In this paper we restrict our attention to the case of two (or no) sons per cluster which is the easiest with respect to analysis and implementation, and we are not aware of any disadvantages compared to an arbitrary number of sons. For locally refined grids, the results from [4] indicate that the cardinality balanced clustering is *not* optimal with respect to the complexity of adaptive $\mathcal{H}$-matrix updates. Instead, on can use a geometrically balanced approach as described in [5] which is used in the remainder of this paper, including the numerical tests. In the case of regular grids, however, both approaches are the same.

DEFINITION 2.2 (Leaf, father, level, depth). *Let T_I be a cluster tree. The* set of leaves *of the tree T_I is $\mathcal{L}(T_I) = \{v \in T_I \mid S(v) = \emptyset\}$. The uniquely determined predecessor (*father*) of a non-root vertex $v \in T_I$ is denoted by $\mathcal{F}(v)$. The* levels *of the tree T_I are defined by*

$$T_I^{(0)} := \{I\}, \quad T_I^{(\ell)} := \{v \in T_I \mid \mathcal{F}(v) \in T_I^{(\ell-1)}\} \qquad \text{for } \ell \in \mathbb{N},$$

and we write $\mathrm{level}(v) = \ell$ *if $v \in T_I^{(\ell)}$. The depth of T is defined as* $\mathrm{d}(T) := \max\{\ell \in \mathbb{N}_0 \mid T_I^{(\ell)} \neq \emptyset\}$. For any cluster tree T_I there holds $I = \dot{\bigcup}\{v \mid v \in \mathcal{L}(T_I)\}$.

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:

DEFINITION 2.3 (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 $v \in T_{I\times I}^{(l)}$ there exist $t, s \in T_I^{(l)}$ such that $v = t \times s$. The nodes $v \in T_{I\times I}$ are called block clusters.*

Analogously to the cluster tree, for any block cluster tree $T_{I\times I}$ there holds $I\times I = \dot{\bigcup}\{v \mid v\in\mathcal{L}(T_{I\times I})\}$, i.e., the leaves of the block cluster tree provide a disjoint block partition of the product index set $I\times I$.

The objective is to construct a block cluster tree from a given cluster tree such that the leaves (of the block cluster tree) correspond to (preferably large) matrix blocks with "smooth" data that can be approximated by low rank matrices in the following Rk-matrix representation:

DEFINITION 2.4 (Rk-matrix representation). *Let $k,n,m\in\mathbb{N}_0$. Let $M\in\mathbb{R}^{n\times m}$ be a matrix of at most rank k. A representation of M in factorized form*

$$M = AB^T, \qquad A\in\mathbb{R}^{n\times k}, B\in\mathbb{R}^{m\times k}, \tag{2.1}$$

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 [4].

In the following construction, we will build a block cluster tree iteratively starting from $I\times I$ and refining the block clusters if they do not satisfy a certain admissibility condition. The choice of the admissibility condition depends on the underlying continuous problem (i.e., elliptic partial differential equation, in particular its associated Green's function) and shall ensure that all admissible blocks allow a sufficiently accurate Rk-approximation. A typical admissibility condition for uniformly elliptic problems is as follows:

$$\text{block cluster } s\times t \text{ is admissible } \Leftrightarrow min(diam(s), diam(t)) \le \eta\ dist(s,t). \tag{2.2}$$

Here, "diam" and "dist" denote the Euclidean diameter/distance of the (union of the) supports of the basis functions (grid points in the case of finite differences) with indices in s,t, resp. A given cluster tree along with an admissibility condition allows the following canonical construction of a block cluster tree:

Let the cluster tree T_I be given. We define the block cluster tree $T_{I\times I}$ by root$(T) := I\times I$, and each vertex $s\times t\in T$ has the set of successors

$$S(s\times t) := \begin{cases} \emptyset & \text{if } s\times t \text{ admissible,}\\ \emptyset & \text{if } \min\{\#t,\#s\}\le n_{\min},\\ \{s'\times t' \mid s'\in S(s), t'\in S(t);\} & \text{otherwise.}\end{cases} \tag{2.3}$$

The parameter n_{min} ensures that blocks do not become too small where the matrix arithmetic of a full matrix is more efficient than any further subdivision. It is typically set to $n_{min} = 32$ or even $n_{min} = 64$. The leaves of a block cluster tree obtained through this construction will be used in the definition of an $\mathcal{H}$-matrix:

DEFINITION 2.5 ($\mathcal{H}$-matrix). *Let $k, n_{\min}\in\mathbb{N}_0$. 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*

$$\mathcal{H}(T,k) := \{M\in\mathbb{R}^{I\times I} \mid \forall t\times s\in\mathcal{L}(T) : \operatorname{rank}(M|_{t\times s})\le k \text{ or } \min\{\#t,\#s\}\le n_{\min}\}.$$

A matrix $M \in \mathcal{H}(T, k)$ is said to be given in $\mathcal{H}$-matrix representation if the blocks $M|_{t\times s}$ with $\operatorname{rank}(M|_{t\times s}) \leq k$ *are in Rk-matrix representation (and the remaining blocks with* $\min\{\#t, \#s\} \leq n_{\min}$ *are stored as full matrices). The set of indices $(i, j) \in I \times I$ that belong to Rk-matrix blocks is called the* farfield *while the complement is called the* nearfield.

Both the accuracy and (storage) complexity of an $\mathcal{H}$-matrix approximation to a given matrix depend on the construction of an appropriate cluster tree, i.e., a hierarchy of index set partitionings. Details regarding approximation errors for blocks that satisfy the admissibility condition, i.e., for blocks that have a large distance compared to their diameters as well as storage requirements for full, Rk- and $\mathcal{H}$-matrices are given in [4]. The intuitive objective in the construction of a cluster tree, given the standard admissibility condition (2.2), is to partition the index set into well separated clusters of vertices that are geometrically close to each other. As a result, relatively large blocks become admissible and we obtain an accurate $\mathcal{H}$-matrix approximation that is inexpensive to store.

3. $\mathcal{H}$-matrix approximations to the LU-decomposition. Typically, the factors L and U in an exact LU-decomposition of a sparse matrix A suffer from significant fill-in as illustrated in Figure 3.1. Here, the stiffness matrix A (left) and its LU-factors (middle) are presented which results from an upwind finite element discretization of the convection-diffusion equation $-\Delta u + u_x = f$ with Dirichlet boundary conditions on a regular triangulation of the square $\Omega = [-1, 1] \times [-1, 1]$. In this example, the problem size has been set to $n = 10000$. A count of the non-zero entries in the LU-decomposition for various problem sizes yields a non-optimal complexity of $\mathcal{O}(n^{\frac{3}{2}})$ for the number of non-zero entries. The ordering of the unknowns results from the clustering process in the $\mathcal{H}$-matrix construction. It is well-known that the ordering of the unknowns affects the sparsity pattern of the LU-decomposition as well as the approximation quality of an incomplete LU-decomposition (ILU).

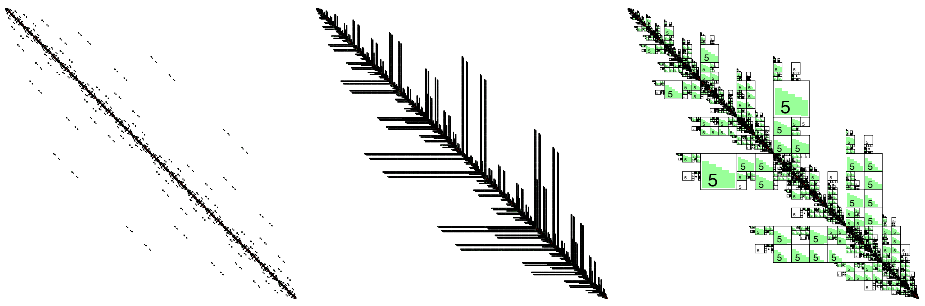

Fig. 3.1. *Sparsity pattern of a typical stiffness matrix A (left), its LU-factors (middle), and an $\mathcal{H}$-matrix approximation to its LU-factors (right).*

Figure 3.1 (right) shows an $\mathcal{H}$-matrix approximation to the LU-factors. Admissible off-diagonal blocks are represented by Rk-matrices with ranks at most five. For the larger off-diagonal blocks, one can observe the exponential decay of the first five singular values which are plotted in grey on a logarithmic scale.

3.1. Construction of the $\mathcal{H}$-matrix LU-decomposition. The computation of $\mathcal{H}$-matrix approximations to the LU-factors is done recursively (with respect to the block clustering) and is of complexity $\mathcal{O}\ (n \log^2 nk^2)$ where n denotes the problem size and k denotes the maximum rank of the Rk-blocks. It requires a triangular solve $TX = B$ for an unknown matrix X with a given (upper or lower) triangular matrix T and a given right hand side matrix B which we will describe first: For $n \leq n_{\min}$, we define $X := T^{-1}B$ and use the standard (dense) matrix arithmetic. We assume a 2×2 block structure of the matrix which is induced by the hierarchy of index partitionings, and we assume that the triangular solve is defined already on all coarser levels. We will solve for the unknown blocks X_{ij} in

$$\begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} X_{11} & X_{12} \\ X_{21} & X_{22} \end{pmatrix} = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

in the following four steps:

1. Solve $L_{11}X_{11} = B_{11}$ for X_{11} by a triangular solve (with L_{11}) on the next coarser level.
2. Solve $L_{11}X_{12} = B_{12}$ for X_{12} by a triangular solve (with L_{11}) on the next coarser level.
3. Solve $L_{22}X_{21} = B_{21} - L_{21}X_{11}$ for X_{21} by a triangular solve (with L_{22}) on the next coarser level.
4. Solve $L_{22}X_{22} = B_{22} - L_{21}X_{12}$ for X_{22} by a triangular solve (with L_{22}) on the next coarser level.

Any matrix arithmetic in the previous four steps will be replaced by (approximate) $\mathcal{H}$-matrix arithmetic which will ensure almost optimal complexity for the computational cost as well as storage for the resulting matrix blocks X_{ij}. The triangular solve involving an upper triangular matrix follows analogously.

To compute an $\mathcal{H}$-LU-decomposition $A = LU$, we use a standard (dense) LU-decomposition $A = LU$ on the coarsest level (where $n \leq n_{\min}$), and we assume that the $\mathcal{H}$-LU-decomposition is defined already on all coarser levels. We will solve for the unknown blocks L_{ij}, U_{ij} in

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} = \begin{pmatrix} L_{11} & \\ L_{21} & L_{22} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & U_{22} \end{pmatrix}$$

in the following four steps:

1. Solve $A_{11} = L_{11}U_{11}$ for L_{11}, U_{11} by an $\mathcal{H}$-LU-decomposition on the next coarser level.
2. Solve $A_{12} = L_{11}U_{12}$ for U_{12} by a triangular solve (with L_{11}).
3. Solve $A_{21} = L_{21}U_{11}$ for L_{21} by a triangular solve (with U_{11}).
4. Solve $L_{22}U_{22} = A_{22} - L_{21}U_{12}$ for L_{22}, U_{22} by an $\mathcal{H}$-LU-decomposition on the next coarser level.

Again, any matrix arithmetic will be replaced by (approximate) $\mathcal{H}$-matrix arithmetic which will ensure almost optimal complexity for the computational cost as well as storage for the resulting matrix blocks L_{ij}, U_{ij}.

Whereas the classical $\mathcal{H}$-matrix uses a fixed rank for the Rk-blocks, it is possible to replace it by a variable rank in order to enforce a desired accuracy within the individual blocks. We will demonstrate the performance of both approaches in the numerical tests in Section 4.

3.2. The case of constant convection. In this section, we consider the model convection-diffusion equation with constant convection

$$-\epsilon\Delta u + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix} \cdot \nabla u = f \text{ in } \Omega, \tag{3.1}$$
$$u = g \text{ on } \Gamma := \partial\Omega. \tag{3.2}$$

with $\epsilon \in (0,1)$ and constant $b_1, b_2 \in \mathbb{R}$.

In the case of dominant convection (i.e., $\epsilon \ll 1$), the non-zero entries in the stiffness matrix obtained through some upwind discretization typically differ significantly in their magnitudes, and it is helpful to distinguish between large and small matrix entries. One may neglect the small matrix entries by replacing them with zeros and then order the unknowns in a *downwind* manner such that one obtains an upper triangular matrix which can be used as a preconditioner [2, 10, 9]. Such an ordering is not determined uniquely, and the following Lemma guarantees that such an ordering can be obtained through the standard $\mathcal{H}$-matrix construction using a geometric partition by ordering the cluster sons according to the convection direction. This, in turn, will ensure that the quality of the $\mathcal{H}$-LU-decomposition will improve as $\epsilon \to 0$ since the stiffness matrix tends towards an upper triangular matrix.

LEMMA 3.1. *Let $(b_1, b_2)^T$ denote the constant convection direction in (3.3). We assume an upwind discretization. Let T_I be a geometrically balanced cluster tree, i.e., a cluster r is subdivided into two successors s, t by 'cutting' along either a vertical or horizontal line. Let (x_i, y_i) be the point associated with an index i. If s, t result from a vertical cut such that $x_i < x_j$ for all indices $i \in s$, $j \in t$, then we will order the indices in s before those in t if $b_1 \geq 0$. If $b_1 < 0$, we will order indices in t before those in s. Correspondingly, if s, t result from a horizontal cut such that $y_i < y_j$ for all indices $i \in s$, $j \in t$, then we will order the indices in s before those in t if $b_2 \geq 0$. If $b_2 < 0$, we will order indices in t before those in s. The resulting $\mathcal{H}$-matrix will have all "large" entries in the lower triangular part, i.e., $a_{ij} > a_{ji}$ if $j < i$. As $\epsilon \to 0$ in (3.1), the stiffness matrix tends to be a lower triangular matrix.*

We point out that the above Lemma establishes a significant difference between the computation of $\mathcal{H}$-LU-factors and an approximate $\mathcal{H}$-inverse matrix: In [11], it is shown that the computation of a convergent approximate $\mathcal{H}$-matrix inverse in the case of dominant, constant convection requires the adjustment of the clustering as well as the admissibility condition to the convection direction and its magnitude. The Lemma above states that such an adjustment is not required for the computation of $\mathcal{H}$-LU-factors.

3.3. The case of arbitrary convection. In this section, we consider the model convection-diffusion equation with arbitrary convection

$$-\epsilon\Delta u + \mathbf{b} \cdot \nabla u = f \text{ in } \Omega, \tag{3.3}$$
$$u = g \text{ on } \Gamma := \partial\Omega. \tag{3.4}$$

with $\epsilon \in (0,1)$ and arbitrary (possibly cyclic) $\mathbf{b} = (b_1(x), b_2(x))^T$. If the convection is acyclic one may still imitate downwind ordering strategies to obtain a mostly triangular matrix using the standard clustering algorithm. In the cyclic case, however, such an ordering does not exist, not even without the restriction of a standard clustering. Even though some convection-adapted clustering may seem reasonable, we found in our numerical results that the standard clustering still yields the best and very satisfactory results. One explanation may be the numerical viscosity of the underlying discretization.

4. Numerical results for $\mathcal{H}$-matrix preconditioners. In this section we will demonstrate some numerical results using the $\mathcal{H}$-LU-decomposition as a preconditioner. All tests were performed on a Dell Workstation 530 with a Xeon 2.4GHz processor using the standard $\mathcal{H}$-matrix library HLib (cf. `http://www.hlib.org`). Throughout all tests, we set the admissibility condition parameter $\eta = 4.0$ (2.2). Our test problem is the convection-diffusion equation (3.3), discretized on a regular triangulation on the square $\Omega = [-1,1] \times [-1,1]$ using Tabata's upwind triangle scheme [13, Chap. III, Sec. 3.3.1]. In all tests we choose a cyclic convection direction defined by

$$\mathbf{b}(x,y) = (b_1(x,y), b_2(x,y))^T \text{ with } b_1(x,y) = 0.5 - y, \quad b_2(x,y) = x - 0.5. \qquad (4.1)$$

In the first set of tests, the number of unknowns is $n = 160000$. We compare the required storage (Figure 4.1) and time (Figure 4.2) to obtain a certain approximation accuracy of the $\mathcal{H}$-LU-factors when using fixed ranks (diamonds) in the Rk-blocks compared to adaptive ranks (squares). In the case of adaptive ranks, we choose the local ranks k such that $\sigma_k \leq \delta\sigma_0$ where σ_i denotes the i'th largest singular value of the respective matrix block, and $\delta = 10^{-j}, j = 1,..,5$ for the results shown in Figures 4.1, 4.2.

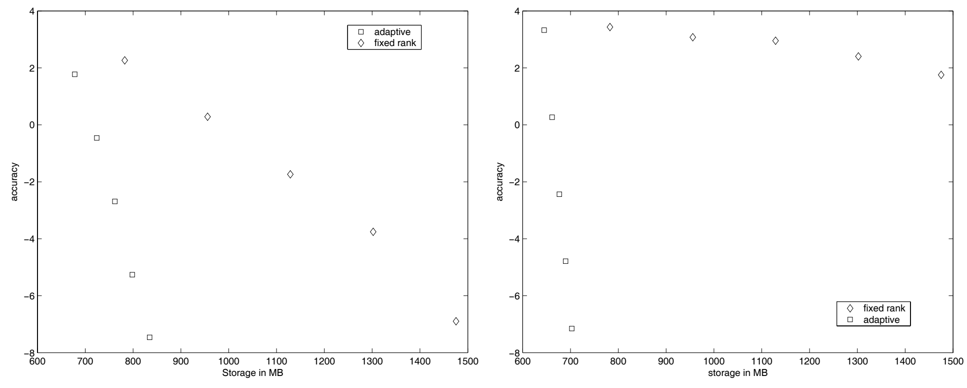

Fig. 4.1. *Approximation accuracy versus required storage of LU-factors for fixed rank (diamonds) or adaptive rank (squares); left for $\epsilon = 1.0$ and right for convection-dominant case with $\epsilon = 1e - 16$.*

The x-axis measures the required storage of the $\mathcal{H}$-LU-factors in megabytes, whereas the y-axis shows the accuracy $\|(L^{\mathcal{H}}U^{\mathcal{H}})^{-1}A - I\|_2$ of the $\mathcal{H}$-LU-factors on a logarithmic scale. The left plots in both Figures 4.1, 4.2 show the results where $\epsilon = 1.0$ in the convection-diffusion equation (3.3), i.e., when the convection is only moderate. We notice that for both choices - adaptive or fixed ranks - the approximation accuracy improves exponentially as the used storage (Figure 4.1) and the required time to compute the $\mathcal{H}$-LU-factors (Figure 4.2) increase (in view of the logarithmically scaled y-axis). It is apparent that a much faster convergence is obtained when adaptive ranks are used. The advantage of adaptive ranks over fixed ranks becomes even more apparent in the convection dominant case: The plots on the right of Figures 4.1, 4.2 show the results where $\epsilon = 1e - 16$ in the convection-diffusion equation

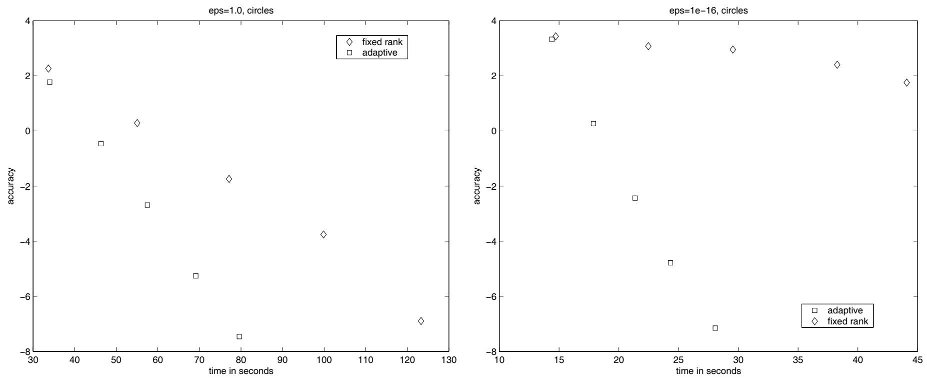

FIG. 4.2. *Approximation accuracy versus required time to compute $\mathcal{H}$-LU-factors for fixed rank (diamonds) or adaptive rank (squares); left for* $\epsilon = 1.0$ *and right for convection-dominant case with* $\epsilon = 1e-16$.

(3.3). The advantage of adaptive ranks can be explained by the non-symmetric nature of the stiffness matrix with greatly varying magnitudes of matrix entries: Whereas some Rk-blocks require a large rank to obtain a certain accuracy, other blocks are close to zero blocks, and imposing a fixed rank onto these blocks will yield hardly any improvement in accuracy but still be costly with respect to storage and time.

Consequently, we will choose adaptive ranks in the following numerical tests where we use the $\mathcal{H}$-LU-factors as preconditioners in a bicgstab iteration. As before, we choose a cyclic convection (4.1). Table 4.1 displays results for a moderate convection ($\epsilon = 1.0$ in (3.3)), and in Table 4.2 the same tests are repeated for a dominant convection ($\epsilon = 1e - 16$ in (3.3)). We record the times for the construction of the $\mathcal{H}$-LU-factors for various problem sizes ($n = 40000, 80089, 160000, 320356$) and accuracies δ responsible to determines the local ranks in the Rk-blocks. For a fixed accuracy, the time increases linearly as the problem size increases. Next, we record the time it takes to reduce the initial residual $r_0 := \|Ax_0 - b\|_2$ by a factor of 10^{-8} and the required number of iteration steps. We notice that the construction of the $\mathcal{H}$-LU-preconditioner dominates the solution time, and with respect to the combined solution time, the least accurate $\mathcal{H}$-LU-decomposition will yield the shortest solution time although the convergence rate is the worst. This may change, however, if a more accurate solution is desired, if we try to solve a worse conditioned test problem or if we further increase the problem size n. For the lowest accuracy 0.1, the number of iteration steps increases as the problem size increases, whereas for higher accuracies the preconditioner becomes optimal since the number of required steps remains constant as we increase the problem size.

The set-up time of the preconditioner has less impact when it is used for solving systems of equations such as the (Navier-) Stokes equations. Here, the problem size becomes $d \cdot n + p$ where d denotes the spatial dimension, n the number of velocity unknowns (per spatial dimension) and p the number of pressure unknowns. Here, we would only require an $\mathcal{H}$-LU-decomposition of the first diagonal block of the stiffness matrix which is of size $n \times n$ in order to construct a preconditioner. Therefore, the time for the construction of the $\mathcal{H}$-LU-factors will decrease relative to the time per iteration step.

TABLE 4.1
$\epsilon = 1.0$, *cyclic convection, adaptive rank*

	time LU-decompose (in sec.)				time solver/#steps			
n\accuracy	0.1	1e-2	1e-3	1e-4	0.1	1e-2	1e-3	1e-4
40000	6.55	8.63	10.50	12.51	0.70/3	0.51/2	0.54/2	0.29/1
80089	12.78	17.61	22.15	26.93	1.63/4	1.42/3	1.00/2	1.07/2
160000	34.18	46.45	57.28	68.84	4.06/4	1.42/3	2.39/2	2.61/2
320356	68.55	98.96	126.25	154.03	8.88/5	8.11/4	4.54/2	4.85/2

TABLE 4.2
$\epsilon = 1e - 16$, *cyclic convection, adaptive rank*

	time LU-decompose (in sec.)				time solver/#steps			
n\accuracy	0.1	1e-2	1e-3	1e-4	0.1	1e-2	1e-3	1e-4
40000	2.89	3.46	3.98	4.48	0.72/4	0.38/2	0.39/2	0.20/1
80089	5.39	6.74	7.97	9.51	1.59/5	1.01/3	0.71/2	0.73/2
160000	14.09	17.50	20.79	24.23	3.81/5	2.41/3	1.68/2	1.74/2
320356	27.46	35.65	43.03	51.77	9.38/7	4.31/3	3.03/2	3.15/2

REFERENCES

[1] M. BEBENDORF AND W. HACKBUSCH, *Existence of $\mathcal{H}$-matrix Approximants to the inverse FE-Matrix of Elliptic Operators with L^∞-Coefficients*, Numerische Mathematik, 95 (2003), pp. 1–28.

[2] J. BEY AND G. WITTUM, *Downwind numbering: Robust multigrid for convection diffusion problems*, Applied Numerical Mathematics, 23 (1997), pp. 177–192.

[3] S. BÖRM, L. GRASEDYCK, AND W. HACKBUSCH, *Hierarchical matrices*, 2003. Lecture Notes No. 21, Max Planck Institute for Mathematics in the Sciences, Leipzig, Germany.

[4] L. GRASEDYCK AND W. HACKBUSCH, *Construction and arithmetics of $\mathcal{H}$-matrices*, Computing, 70 (2002), pp. 295–334.

[5] L. GRASEDYCK, W. HACKBUSCH, AND S. LE BORNE, *Adaptive geometrically balanced clustering of $\mathcal{H}$-matrices*, Computing, 73 (2003), pp. 1–23.

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

[7] W. HACKBUSCH, L. GRASEDYCK, AND S. BÖRM, *An introduction to hierarchical matrices*, Math. Bohem., 127 (2002), pp. 229–241.

[8] W. HACKBUSCH AND B. KHOROMSKIJ, *A sparse $\mathcal{H}$-matrix arithmetic. Part II: Application to multi-dimensional problems*, Computing, 64 (2000), pp. 21–47.

[9] W. HACKBUSCH AND T. PROBST, *Downwind Gauß-Seidel Smoothing for Convection Dominated Problems*, Numerical Linear Algebra with Applications, 4 (1997), pp. 85–102.

[10] S. LE BORNE, *Ordering techniques for two- and three-dimensional convection-dominated elliptic boundary value problems*, Computing, 64 (2000), pp. 123–155.

[11] ———, *$\mathcal{H}$-matrices for convection-diffusion problems with constant convection*, Computing, 70 (2003), pp. 261–274.

[12] M. LINTNER, *The eigenvalue problem for the 2D Laplacian in H-matrix arithmetic and application to the heat and wave equation*, Computing, 72 (2004), pp. 293–323.

[13] H. ROOS, M. STYNES, AND L. TOBISKA, *Numerical methods for singularly perturbed differential equations: convection diffusion and flow problems*, vol. 24 of Computational Mathematics, Springer, Berlin, 1996.