Fourteenth International Conference on Domain Decomposition Methods
Editors: Ismael Herrera , David E. Keyes, Olof B. Widlund, Robert Yates ©2003 DDM.org

4. Direct Domain Decomposition using the Hierarchical Matrix Technique

W. Hackbusch[1]

1. Introduction. In the time when the domain decomposition technique developed, direct solvers were quite common. We come back to direct methods; however, the term "direct" has another meaning. The usual understanding of a direct method is:

Given a matrix A and a vector b,
produce the solution x of $Ax = b$.

Here, we require a much more mighty procedure:

Given a matrix A and a vector b,
approximate the inverse A^{-1} and $x = \left(A^{-1}\right) * b$.

The technique of hierarchical matrices allows to perform this step with an almost optimal storage and operation cost $\tilde{O}(n)$ for $n \times n$ matrices related to elliptic operators. The symbol $\tilde{O}(n)$ means the order $O(n)$ up to a logarithmic factor, i.e., there is a (small) number α such that

$$\tilde{O}(n) = O\left(n \log^{\alpha} n\right).$$

In Section 2, we describe the underlying problem of a non-overlapping domain decomposition and the corresponding system of equations. It is interesting to remark that

- rough (exterior and) interior boundaries are allowed, i.e., no smoothness conditions on the subdomains or the interior boundary (skeleton) are necessary.
- inside of the subdomains, L^{∞}-coefficients are allowed (i.e., jumping coefficients, oscillatory coefficients, etc.). There is no need to place the skeleton along jump lines. A proof concerning robustness against rough boundaries and non-smooth coefficients is given in [1]. If, however, it happens that the coefficients are piecewise constant or analytic in the subdomains, a further improvement is possible using a new technique of Khoromskij and Melenk [7] (see Subsection 3.2).

The two items mentioned above allow to create the subdomains independent of smoothness considerations; instead we may use load balancing arguments.

Further advantages will be mentioned in the Section 3, where the direct solution is explained.

The basic of the solution method are the hierarchical matrices which are already described in several papers (cf. [4], [5]). An introduction is given in [2]. We give an outline of the method in Section 4.

[1]Max-Planck-Institut *Mathematik in den Naturwissenschaften*, wh@mis.mpg.de

Although the method of hierarchical matrices could be applied immediately to the global problem, the domain decomposition helps to achieve a parallelisation of the solution process. The details are discussed in Section 5.

We conclude this contribution in Section 6 with numerical example for the inversion of finite element stiffness matrices. We take an example with extremely non-smooth coefficients to support the remark above.

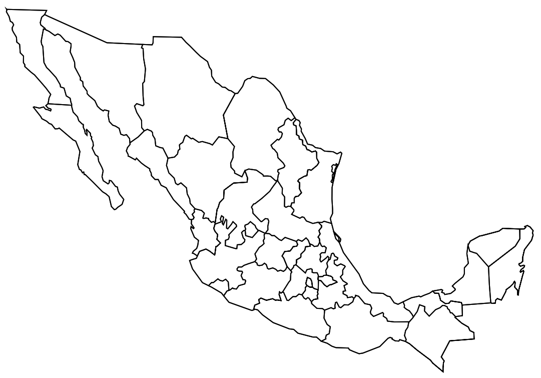


Figure 1.1: Domain decomposition with non-smooth interfaces

2. Non-Overlapping Domain Decomposition. Let the domain Ω be decomposed into p non-overlapping subdomains Ω_i, $i = 1, \ldots, p$ (cf. Figure 1.1). The skeleton Σ consists of the interior interfaces:

$$\Sigma := \left(\bigcup\nolimits_{i=1}^{p} \partial\Omega_i\right) \setminus \partial\Omega.$$

For a simpler finite element realisation we may assume (in 2D) that Ω_i are polygons. Then Σ is a union of straight lines. In 3D, Σ may consist of flat faces. As mentioned in the introduction, there is no need for Ω_i to form a regular macro element. Later, we will assume that all Ω_i contain a comparable number of degrees of freedom to achieve a load balance in the parallelisation process.

Let I_i be the index set of interior degrees of freedom in Ω_i (the precise definition of $j \in I_i$ is that the corresponding basis function b_j satisfies[2] $\operatorname{supp}(b_j) \subset \overline{\Omega_i}$). All remaining indices are associated with the skeleton and its set is denoted by I_Σ. Hence, we arrive at the decomposition of the global index set I into

$$I = I_1 \cup \cdots \cup I_p \cup I_\Sigma \qquad \text{(disjoint union).}$$

As usual, the total dimension is denoted by

$$n := \#I. \tag{2.1}$$

[2]Note that by definition the support $\operatorname{supp}(b_j)$ is always in $\overline{\Omega}$, also if the nodal point lies on $\partial\Omega$.

The FE system $Au = f$ has the structure

$$\begin{bmatrix} A_{11} & O & \cdots & O & A_{1,\Sigma} \\ O & A_{22} & \cdots & O & A_{2,\Sigma} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ O & O & \cdots & A_{pp} & A_{p,\Sigma} \\ A_{\Sigma,1} & A_{\Sigma,2} & \cdots & A_{\Sigma,p} & A_{\Sigma\Sigma} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_p \\ u_\Sigma \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \\ f_\Sigma \end{bmatrix} \tag{2.2}$$

when we order the unknowns in the sequence $I_1, \ldots, I_p, I_\Sigma$.

As usual in domain decomposition, we assume that besides A all submatrices A_{ii} are invertible, i.e., the subdomain problems are solvable.

In the case of a non-matching domain decomposition (mortar FEM), the elimination of all slave nodes by means of the mortar condition yields again the system (2.2), where I_Σ is the index set of all mortar nodes (cf. [3]).

3. Direct Solution Process.

3.1. Description of Single Steps. The system (2.2) can be reduced to the Schur complement equation

$$Su_\Sigma = g_\Sigma, \tag{3.1}$$

where

$$S := A_{\Sigma\Sigma} - \sum_{i=1}^{p} A_{\Sigma,i} A_{ii}^{-1} A_{i,\Sigma}, \tag{3.2}$$

$$g_\Sigma := f_\Sigma - \sum_{i=1}^{p} A_{\Sigma,i} A_{ii}^{-1} f_i. \tag{3.3}$$

The remaining variables u_i are the result of

$$u_i := A_{ii}^{-1} \left(f_i - A_{i,\Sigma} u_\Sigma \right) \qquad \text{for } i = 1, \ldots, p. \tag{3.4}$$

An obvious solution method which is usually not used because one is afraid of the bad complexity up to $O(n^3)$ of standard solvers is the following:

Step 1a	produce the inverse matrix A_{ii}^{-1},
Step 1b	form the products $A_{\Sigma,i} * \left(A_{ii}^{-1}\right)$ and $\left(A_{\Sigma,i}A_{ii}^{-1}\right) * A_{i,\Sigma}$,
Step 1c	compute the vectors $\left(A_{\Sigma,i}A_{ii}^{-1}\right) * f_i$,
Step 2a	form the sum $S = A_{\Sigma\Sigma} - \sum_{i=1}^{p} \left(A_{\Sigma,i}A_{ii}^{-1}A_{i,\Sigma}\right)$,
Step 2b	compute the vector $g_\Sigma = f_\Sigma - \sum_{i=1}^{p} \left(A_{\Sigma,i}A_{ii}^{-1}f_i\right)$,
Step 3a	produce the inverse matrix S^{-1},
Step 3b	compute the vector $u_\Sigma = \left(S^{-1}\right) * g_\Sigma$,
Step 4	compute the vectors $u_i = \left(A_{ii}^{-1}\right) * \left[f_i - A_{i,\Sigma} * u_\Sigma\right]$.

Terms in round brackets are already computed quantities. The necessary operations are indicated by $\circ^{-1}$, $*$, $-$, $\sum$.

In the sequel we follow the Steps 1-4 with the following modifications: Steps 1a,1b,2a,3a are performed only approximately up to an error ε. Usually[3], one wants ε to be similar to the discretisation error, i.e.,

$$\varepsilon = O(h^\kappa) = O(n^{-\beta}), \tag{3.5}$$

where h is the step size (if there is a quasi-uniform one) and κ is the consistency order. Then $\beta = \kappa/d$ holds, where d is the spatial dimension:

$$\Omega \subset \mathbb{R}^d. \tag{3.6}$$

In the non-uniform finite element case, one expects an discretisation error $O(n^{-\beta})$ for an appropriate triangulation. In that case ignore the middle term in (3.5).

Allowing approximation errors of order $O(\varepsilon)$, the technique of hierarchical matrices explained in the next section will be able to perform all Steps 1a-4 with storage and computer time of order $\tilde{O}(n)$. Hence, the costs are similar to usual iterative DD methods. One of the advantages of the direct method is its robustness and the relative easy implementation. To be precise: It is not so simple to implement the hierarchical matrix method for the first time, but as soon as one has programmed this method, it can be used without modification for different FE applications as well as for the Schur equation $Su_\Sigma = g_\Sigma$ with the (fully populated) matrix S.

Finally we remark that A^{-1} can be computed in **Step 5**:

$$A^{-1} = \begin{bmatrix} \ddots & \vdots & \cdots & \vdots \\ \cdots & \delta_{ij}A_{ii}^{-1} + A_{ii}^{-1}A_{i,\Sigma}S^{-1}A_{\Sigma,j}A_{jj}^{-1} & \cdots & -A_{ii}^{-1}A_{i,\Sigma}S^{-1} \\ & \vdots & \ddots & \vdots \\ \cdots & -S^{-1}A_{\Sigma,j}A_{jj}^{-1} & \cdots & S^{-1} \end{bmatrix}.$$

However, we should make use of the representation by

$$A^{-1} = \begin{bmatrix} A_{11}^{-1} & O & O & O \\ O & \ddots & O & O \\ O & O & A_{pp}^{-1} & O \\ O & O & O & O \end{bmatrix} + \begin{bmatrix} A_{11}^{-1}A_{1,\Sigma} \\ \vdots \\ A_{pp}^{-1}A_{p,\Sigma} \\ -I \end{bmatrix} \begin{bmatrix} S^{-1}A_{\Sigma,1}A_{11}^{-1} & \cdots & S^{-1}A_{\Sigma,p}A_{pp}^{-1} & -S^{-1} \end{bmatrix}. \tag{3.7}$$

3.2. Improvement for Piecewise Smooth Coefficients. We mentioned that the approach from above works also efficiently if the subproblems corresponding to the FE matrices A_{ii} involve non-smooth coefficients of the elliptic differential equation. If, on the other hand, we know that the coefficients in one subdomain are constant (or

[3]If one performs only the Steps 1a,1b,2a and 3a to get a rough approximation of S^{-1} for the purpose of preconditioning, ε may be of fixed order $O(1)$, e.g., $\varepsilon = 1/10$.

analytic), one can exploit this fact by applying a more appropriate finite element discretisation. In [7] a so-called *boundary concentrated finite element method* is described which allows to solve the local problem with a number of unknowns proportional to area$(\partial\Omega_i)/h^{d-1}$. This number is usually smaller by a factor of h than the number of degrees of freedom in a classical FEM, although the same resolution is obtained at the boundary.

We do not discuss this modification in the subdomains in the following, i.e., we consider a traditional FEM.

4. Hierarchical Matrices.

It is to be remarked that the method of hierarchical matrices does not apply to any matrix but only to those related to elliptic (pseudo-) differential operators. In our application, A_{ii} as well as their inverse matrices are related to the local elliptic problem, while S is a nicely behaving pseudo-differential operator composed from local Steklov operators. Nevertheless, the method is of black-box character since its description does not depend on specific features of the involved matrices. The success of this kind of approximation depends only on the ellipticity properties.

In the following, we give an introduction into the definition and construction of $\mathcal{H}$-matrices. The interested reader will find more details in [4], [5] and [2].

4.1. The Main Ingredients.

We have to introduce

1. the index set I and the geometric properties of its indices;
2. the cluster tree $T(I)$;
3. the block-cluster tree $T(I \times I)$;
4. the admissibility criterion;
5. the (minimal admissible) partitioning of the matrix;
6. rank-k matrices;
7. the definition of an $\mathcal{H}$-matrix;
8. the (approximations of the) operations $A + B$, $A * B$, A^{-1};
9. the estimates for the storage and operation costs.

First, we give a preview of these topics. The *cluster tree* $T(I)$ describes how the whole index set can be partitioned into smaller pieces, which are needed, e.g., when we want to define a subblock of a vector. The *block-cluster tree* $T(I \times I)$ does the same for the matrix. Among the blocks contained in $T(I \times I)$ we can choose a collection of disjoint blocks covering $I \times I$. Then we get a *partitioning* of the matrix into various blocks. An example is given in Fig. 4.1.

The choice of this partitioning P is the essential part. It should contain as few blocks as possible to make the costs as low as possible. On the other hand, the approximation error must be sufficiently small. For this purpose, all blocks have to satisfy an *admissibility condition*. Then, filling all blocks (e.g., in Fig. 4.1) by matrices of rank smaller or equal some k, we obtain an $\mathcal{H}$-matrix from the class $\mathcal{H}(P, k)$. The results of $A + B$, $A * B$, A^{-1} for $A, B \in \mathcal{H}(P, k)$ will, in general, not be again in $\mathcal{H}(P, k)$, but they can be approximated in this class by a cost of $\mathcal{O}(n)$.

4.2. The Index Set and the Geometrical Data.

As input for the algorithm we only need the description of the index set I (e.g., $\{1, \ldots, n\}$ or list of nodal points, etc.) and a characteristic subset $X(i) \subset \mathbb{R}^d$ associated with $i \in I$. For a collocation

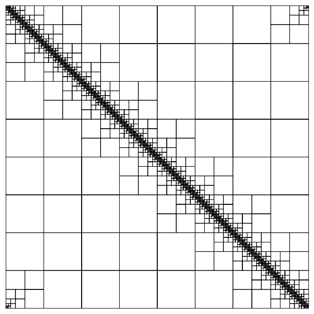


Figure 4.1: Block partitioning P for the unit circle

method, this may be the nodal point, i.e., $X(i) = \{x_i\}$. The appropriate choice for a Galerkin method is

$$X(i) = \operatorname{supp}(\phi_i)\,, \text{ where } \phi_i \text{ is the FE-basis function associated with } i \in I. \tag{4.1}$$

4.3. The Cluster Tree $T(I)$. Formally, the cluster tree $T(I)$ has to satisfy

1. $T(I) \subseteq \mathcal{P}(I)$ (i.e., each node of $T(I)$ is a subset of the index set I).
2. I is the root of $T(I)$.
3. If $\tau \in T(I)$ is a leaf, then $\#\tau = 1$ (i.e., the leaves consist only one index, $\tau = \{i\}$).
4. If $\tau \in T(I)$ is *not* a leaf, then it has exactly two sons and is their disjoint union.

All nodes of $T(I)$ are called “clusters”. For each $\tau \in T(I)$, we denote the set of its sons by $S(\tau) \subset T(I)$.

In practice, the condition $\#\tau = 1$ is replaced by $\#\tau \le C_T$, e.g., with $C_T = 32$. The condition for a binary tree (“exactly two sons”) in Part 4 can easily be generalised, although a binary tree is quite reasonable.

The sets $X(i)$ introduced above can immediately be generalised to all clusters by

$$X(\tau) = \bigcup_{i\in\tau} X(i) \subset \mathbb{R}^d \qquad \text{for all } \tau \in T(I). \tag{4.2}$$

Using the Euclidean metric in $\mathbb{R}^d$, we define the *diameter* of a cluster and the *distance* of a pair of clusters:

$$\begin{aligned} \operatorname{diam}(\tau) &= \sup\{|x-y| : x, y \in X(\tau)\} && \text{for } \tau \in T(I), \\ \operatorname{dist}(\tau,\sigma) &= \inf\{|x-y| : x \in X(\tau), y \in X(\sigma)\} && \text{for } \tau,\sigma \in T(I). \end{aligned}$$

The practical construction of $T(I)$ must take care that the clusters are as compact as possible, i.e., $\operatorname{diam}(\tau)$ should be as small as possible for a fixed number $\#\tau$ of indices. One possible construction is the recursive halving of bounding boxes as illustrated in Fig. 4.2. Note that this procedure applies to any irregular FE-triangulation in any spatial dimension.

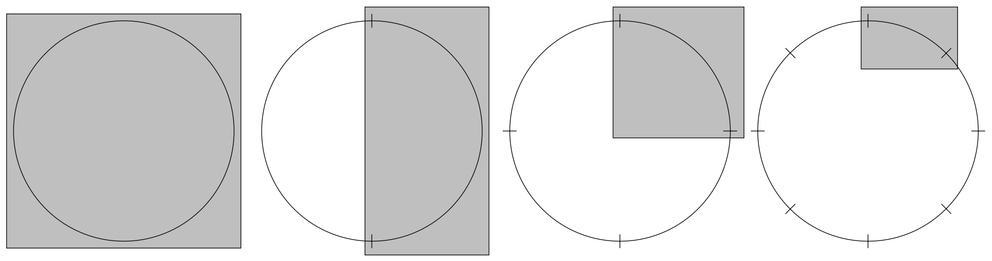


Figure 4.2: Dyadic clustering of the unit circle.

4.4. The Block-Cluster Tree $T(I\times I)$**.** The tree $T(I\times I)$ is completely defined by means of $T(I)$ when we use the following canonical choice. Let $I\times I$ belong to $T(I\times I)$. For all $\tau\times\sigma\in T(I\times I)$ with τ and σ not being leaves, assign the sons $\tau'\times\sigma'$ to $T(I\times I)$, where $\tau'\in S(\tau)$ and $\sigma'\in S(\sigma)$. Again, we write $S(\tau\times\sigma)$ for the set of sons of $\tau\times\sigma$.

Remark 4.1 *a) If $T(I)$ is a binary tree (as described by condition 4 from above), then $T(I\times I)$ is quad-tree.*

b) All "blocks" or "block-clusters" b from $T(I\times I)$ have the product form $b=\tau\times\sigma$ with $\tau,\sigma\in T(I)$. Indices $i\in\tau$ belong the rows of b, while $j\in\sigma$ are column indices.

The set $T(I\times I)$ provides a rich choice of larger and smaller blocks, which we can select to construct the partitioning of Subsection 4.6.

4.5. The Admissibility Condition. Let $b=\tau\times\sigma$ be a block from $T(I\times I)$. If τ or σ is a leaf in $T(I)$ (i.e., $\#\tau=1$ or $\#\sigma=1$), then also b is a leaf in $T(I\times I)$. In this case, b is accepted as "admissible". Otherwise, we recall diam and dist defined via (4.2) and require an admissibility condition like

$$\max\left(\operatorname{diam}\left(\tau\right),\operatorname{diam}\left(\sigma\right)\right)\geq 2\eta\operatorname{dist}\left(\tau,\sigma\right), \tag{4.3}$$

where, e.g., η may be chosen as $\frac{1}{2}$. Even the weaker requirement

$$\min\left(\operatorname{diam}\left(\tau\right),\operatorname{diam}\left(\sigma\right)\right)\geq 2\eta\operatorname{dist}\left(\tau,\sigma\right)$$

makes sense. Conditions of this form are known from panel clustering or from matrix compression in the case of wavelet bases.

It turns out that (4.3) is the appropriate condition to ensure that the rank-k matrices introduced below will lead to the desired accuracy.

4.6. The Partitioning. A partitioning of $I\times I$ is a set $P\subset T(I\times I)$, so that all elements (blocks) are disjoint and $I\times I=\cup_{b\in P}b$. The coarsest partitioning is $P=\{I\times I\}$, while the finest one consists of all leaves of $T(I\times I)$. In the first case we consider the matrix as one block, in the latter case each entry forms a one-by-one block.

We say that P is an admissible partitioning, if all $b\in P$ are admissible. The second of the trivial examples is such an admissible partitioning, since by definition one-by-one blocks are admissible. However, the second example leads to the standard (costly) representation.

To obtain a representation which is as data-sparse as possible but still ensures the desired accuracy, we choose the admissible partitioning with the minimal number of blocks. The construction of this optimal P is as follows. Start with $P := \{I \times I\}$. Since $I \times I$ is definitely not admissible, we divide it into its sons $s \in S(I \times I)$ and replace $I \times I$ by the sons: $P := (P \backslash \{I \times I\}) \cup S(I \times I)$. Similarly, we check for every new $b \in P$, whether it is admissible. If not, $P := (P \backslash \{b\}) \cup S(b)$.

Under mild conditions, one proves that the construction of $T(I)$ by means of bounding boxes explained above, leads to $\#P = \tilde{O}(n)$.

4.7. Rk-Matrices. Except when $\#\tau = 1$ or $\#\sigma = 1$, we represent all block matrices as so-called Rk-matrices represented by $2k$ vectors $a_\iota \in \mathbb{R}^\tau$, $b_\iota^\top \in \mathbb{R}^\sigma$,

$$M = \sum\nolimits_{\iota \neq 1}^{k} a_\iota b_\iota^\top$$

or in matrix form: $M = AB^\top$ with $A \in \mathbb{R}^{\tau,k}$, $B \in \mathbb{R}^{k,\sigma}$. Note that all matrices of rank $\leq k$ can be represented in this form. The storage equals $k * (\#\tau + \#\sigma)$.

4.8. $\mathcal{H}(P,k)$-Matrices. For any partition P and all $k \in \mathbb{N}$, we define

$$\mathcal{H}(P,k) := \left\{A \in \mathbb{R}^{I \times I} : \operatorname{rank}(A|_b) \leq k \text{ for all } b \in P\right\}$$

as the set of hierarchical matrices for the partitioning P of $I \times I$ and the maximal rank k. Here, $A|_b = (A_{ij})_{(i,j) \in b}$ is the block matrix corresponding to $b \in P$. $A|_b$ is represented as Rk-matrix.

There are generalisations i) where the integer k is replaced by a function $k : P \to \mathbb{N}$ (variable rank) and ii) where the condition $\operatorname{rank}(A|_b) \leq k$ is replaced by the stronger requirement that $A|_b$ belongs to a tensor space $V_\tau \otimes V_\sigma$ with $\min(\dim V_\tau, \dim V_\sigma) = k$ (see [6]).

4.9. $\mathcal{H}$-Matrix Operations. The simplest operation is the matrix-vector operation $(A, x) \mapsto A * x$. Obviously, subblocks of x must be multiplied by $A|_b$ and the partial results are summed up. Since $A|_b$ are Rk-matrices, $A|_b * x|_\sigma$ needs only simple scalar products. The overall cost is $\tilde{O}(n)$.

The addition of two $\mathcal{H}(P,k)$-matrices can be performed blockwise and yields a result in $\mathcal{H}(P,2k)$. Truncating all blocks to rank $\leq k$ (e.g., by means of SVD) gives the approximate result in $\mathcal{H}(P,k)$ with a cost of $\tilde{O}(n)$.

The approximative multiplication of two matrices can be performed recursively exploiting the hierarchical structure of the partitioning P (see [2]). The costs are again $\tilde{O}(n)$.

The block Gauss elimination (of a 2×2 block matrix) allows to reduce the inversion of the whole matrix to the inversion of the first block and Schur complement together with additions and multiplications. This yields a recursive algorithm for computing the inverse matrix approximately with cost $\tilde{O}(n)$.

5. Parallelisation.

5.1. First Approach. We recall the disjoint splitting of I into the subsets $I_1, \ldots, I_p, I_\Sigma$. For the purpose of load balance we assume that p processors are available and that the cardinalities $\#I_i$ $(i = 1, \ldots, p)$ are of similar size, i.e.,

$$\#I_i \sim \frac{n}{p} \qquad (i = 1, \ldots, p). \tag{5.1}$$

The computations in Steps 1a-4 of Section 3 contain three different phases:

Phase I	Steps 1a-c
Phase II	Steps 2a-3b
Phase III	Step 4

Obviously, Phases I and III contain completely independent tasks for each $i = 1, \ldots, p$. Hence, assuming p processors, these phases are parallelisable without any communication. The work cost for each processor is $\tilde{O}(\#I_i) = \tilde{O}(\frac{n}{p})$ according to (5.1).

The summation $\sum_{i=1}^{p}$ in Steps 2a,b needs $\log_2(p)$ steps[4] to collect and add the terms. The computations of the Steps 3a,b are performed on one processor, i.e., no parallelisation is used in Phase II. The cost of Phase II amounts to $\tilde{O}(\#I_\Sigma)$.

In Phase III, u_Σ has to be copied to each processor. Then Step 4 can be performed with a cost of $\tilde{O}(\#I_i) = \tilde{O}(\frac{n}{p})$.

Similarly, the data can be distributed so that all p processors in Phase I,III need $\tilde{O}(\#I_i)$ storage, while the one processor of Phase II requires a storage of $\tilde{O}(\#I_\Sigma)$.

We may add a **Phase IV**, where Step 5 (computation of A^{-1}) is performed. For this purpose, the quantities $A_{ii}^{-1}A_{i,\Sigma}$, $S^{-1}\left(A_{\Sigma,i}A_{ii}^{-1}\right)$ from (3.7) are still to be computed, while $A_{\Sigma,i}A_{ii}^{-1}$ are already known from Step 1b.

In total, the whole computation of the phases I-III leads to a cost of $\tilde{O}(\frac{n}{p}) + \tilde{O}(\#I_\Sigma)$. We next assume that subdomains related to I_i are determined such there surface is of minimal order, i.e., the set $I_{\Sigma,i} = \{j \in I_\Sigma : j \text{ neighboured to some } k \in I_i\}$ has a cardinality of $O\left((\#I_i)^{(d-1)/d}\right) = O\left((\frac{n}{p})^{(d-1)/d}\right)$. Hence,

$$O(\#I_\Sigma) = O\left(p\left(\frac{n}{p}\right)^{(d-1)/d}\right) = O\left(p^{1/d}n^{(d-1)/d}\right), \tag{5.2}$$

where d is the spatial dimension. Under the assumption (5.2), the work equals $W = \tilde{O}\left(\frac{n}{p} + p^{1/d}n^{(d-1)/d}\right)$. If n is fixed, the optimal number of processors is $p = O\left(n^{1/(d+1)}\right)$ and leads to $W = \tilde{O}\left(n^{d/(d+1)}\right)$. If, alternatively, the number p of processors is given, the right scaling of n yields $n = O\left(p^{d+1}\right)$.

We summarise in

Remark 5.1 *A parallel treatment in the Phases I and III with $p = O(n^{1/(d+1)})$ processors leads to a work $W = \tilde{O}(n^{d/(d+1)})$. The distributed memory requirements are also $\tilde{O}(n^{d/(d+1)})$. A possible Phase IV requires a work and local storage of the same size.*

[4]We remark that the $\log_2(p)$ factor can be ignored because of our definition of $\tilde{O}(\cdot)$.

5.2. Multiple DD Levels in Phase I. In the previous subsection it was assumed that the Steps 1a-c are performed by means of the $\mathcal{H}$-matrix arithmetic. An alternative is to compute A_{ii}^{-1} in Step 1a again by a DD approach using a further subdivision of I_i into $I_{i,j}$ $(j = 1, \ldots, q_i)$ and $I_{i,\Sigma}$. Due to the representation (3.7), the matrix multiplication in Step 1b can be parallel in q_i processors. The vector operation in Step 1c needs $O(p_i)$ communications to add up all partial results. The work needed to perform Steps 1a-c for a particular i is given by Remark 5.1: Under the natural assumptions from above about the subdivision into $I_{i,1}, \ldots, I_{i,q_i}, I_{i,\Sigma}$ and assuming $q_i = O((\#I_i)^{1/(d+1)})$, the work for Phase I is reduced to $\tilde{O}((\#I_i)^{d/(d+1)})$ (instead of $\tilde{O}(\#I_i)$).

Assume (5.1) and $q_i = q = O((\frac{n}{p})^{1/(d+1)})$ for all i, the total work is $W = \tilde{O}\left((\frac{n}{p})^{\frac{d}{d+1}} + p^{1/d} n^{\frac{d-1}{d}}\right)$, which is minimal for $p = n^{\frac{1}{d+1+d^2}}$, when $W = \tilde{O}(n^{\frac{d^2}{d+1+d^2}})$.

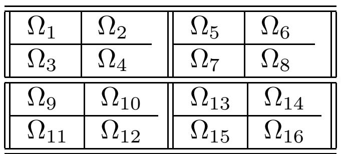
Remark 5.2 *a) The parallel two-level DD approach as described above reduces the work and storage to* $\tilde{O}(n^{\frac{d^2}{d+1+d^2}})$, *where* $P = pq = O(n^{\frac{d+1}{d+1+d^2}})$ *processors are used. These exponents are* $\frac{4}{7}$ *and* $\frac{3}{7}$ *in the case of* $d = 2$. *Note that three different kinds of parallelism appear: i) there are* $P = pq$ *problems to be solved in parallel for the index sets* $I_{i,j}$ $(i = 1, \ldots, p$ *and* $j = 1, \ldots, q)$, *ii)* p *tasks on* $I_{i,\Sigma}$, *iii) 1 task on* I_Σ.

b) There is an obvious generalisation to an L*-level DD approach. The exponents for the three-level case are* $W = \tilde{O}(n^{\frac{d^3}{(d+1)(1+d^2)}})$, $P = O(n^{\frac{d+1+d^2}{(d+1)(1+d^2)}})$. *The numbers for* $d = 2$ *and* $L = 3$ *are* $W = \tilde{O}(n^{\frac{8}{15}})$ *and* $P = O(n^{\frac{7}{15}})$. *For general* L, $W = \tilde{O}(n^{\omega_L})$ *and* $P = O(n^{1-\omega_L})$, *where the exponents* ω_L *converges to* $\lim \omega_L = \frac{d-1}{d}$, *i.e.,*

$$W \to \tilde{O}(n^{\frac{d-1}{d}}), \quad P \to O(n^{\frac{1}{d}}).$$

5.3. DD in Phase II. In the previous subsection, we have improved the performance in Phase I, while Phase II (Steps 2a-3b) remains unparallelised. The only side effect was that the number p of subdomains (of the first level) could be chosen smaller so that $\#I_\Sigma$ was decreasing.

Now we also parallelise Phase II, but it turns out that this approach is equivalent with the approach in Subsection 5.2. Consider a non-overlapping domain decomposition of Ω by Ω_i, $i = 1, \ldots, p$, which is organised in a hierarchical way, i.e., there is a coarser decomposition $\hat{\Omega}_k$, $k = 1, \ldots, K$, so that $\hat{\Omega}_k \supset \bigcup_{i \in J_k} \Omega_i$ for disjoint subsets satisfying $\bigcup_{k=1}^{K} J_k = \{1, \ldots, p\}$.



Coarse DD (double lines) and fine DD (single lines)

In the picture above, the first coarse subset is $\hat{\Omega}_1$ corresponding to the fine subsets Ω_i for $i \in J_1 = \{1, \ldots, 4\}$. The skeleton $\hat{\Sigma}$ of the coarse domain decomposition (double lines in the picture) is a subset of the skeleton Σ of the fine domain decomposition:

$\hat\Sigma \subset \Sigma$. The set $\Sigma\backslash\hat\Sigma$ consists of non-connected parts $\Sigma_k \subset \hat\Omega_k$, $k = 1, \dots, K$. The Schur complement system corresponding to Σ has again the structure of system (2.2), where now the sets $I_1, \dots, I_p, I_\Sigma$ correspond to $\Sigma_1, \dots, \Sigma_p, \hat\Sigma$. Hence, the methods from Subsection 5.1 apply again. The multiple application can be done as in Subsection 5.2.

Remark 5.3 *The Schur complement system for the skeleton $\hat\Sigma$ from above can (identically) obtained in three different ways: a) eliminate directly all interior nodes in $\hat\Omega_k$, $k = 1, \dots, K$; b) follow Subsection 5.2 and eliminate the interior nodes in $\hat\Omega_k$ by means of the secondary domain decomposition by Ω_i, $i \in J_k$; c) compute first the Schur complement system for the finer skeleton Σ and eliminate the nodes from Σ_k, $k = 1, \dots, K$. The approaches b) and c) differ only in the ordering of the unknowns.*

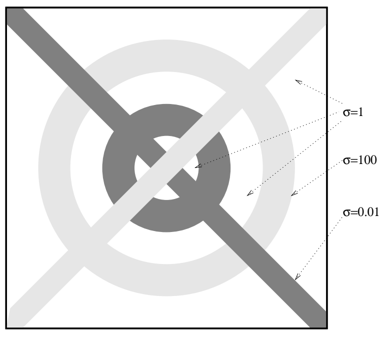
6. Numerical Example.

Since the critical question is the ability to compute the approximate inverse of a FE matrix, we give numerical results for this step. Furthermore, we choose an example with jumping coefficients.

Consider the differential equation

$$\begin{aligned} -\operatorname{div}(\sigma(x)\nabla u(x)) &= f(x) \qquad &&\text{in } \Omega = [0,1]^2, \\ u &= 0 &&\text{on } \Gamma = \partial\Omega, \end{aligned}$$

where the function $\sigma : \mathbb{R}^2 \to \mathbb{R}_{>0}$ defined on Ω has values depicted in the following figure:



$$\sigma(x,y) = \begin{cases} 0.01 & |x+y-1| < 0.05 \text{ or} \\ & (0.1 \le \|(x,y)\| < 0.2) \\ & \text{ and } (|x-y| \ge 0.05) \\ 100 & |x-y| < 0.05 \text{ or} \\ & (0.3 \le \|(x,y)\| < 0.4) \\ & \text{ and } (|x+y-1| \ge 0.05) \\ 1 & \text{otherwise} \end{cases}$$

We introduce a regular finite element discretisation which leads to the sparse $n \times n$ matrix, where $n \in \{32^2, 64^2, 128^2, 256^2\}$. The inversion algorithm applied to A yields the approximation $A_{\mathcal{H}}^{-1}$. The relative error $\|A^{-1} - A_{\mathcal{H}}^{-1}\|_2 / \|A^{-1}\|_2$ is $\le \|I - A_{\mathcal{H}}^{-1}A\|_2$. The later values are given in

	$n =$ degree of freedom			
k	32^2	64^2	128^2	256^2
1	3.5+1	1.1+2	3.1+2	9.5+2
2	2.4-0	1.7+1	1.3+2	4.3+2
3	6.0-1	3.9-0	1.3+1	5.4+1
4	9.4-2	1.0-0	3.4-0	1.0+1
5	2.6-2	2.8-1	7.6-1	6.6-0
6	1.1-3	7.7-2	2.8-1	1.3-0
7	3.9-5	2.1-2	4.8-2	2.3-1
8	9.6-6	1.3-3	1.6-2	4.2-2
9	7.8-6	4.5-4	3.4-3	6.2-3
10	7.0-7	2.9-4	9.7-4	2.5-3
15	5.1-12	7.9-9	8.3-7	1.6-6
20	5.9-12	2.5-11	4.5-9	6.3-9

Due to the multiplication by A, these values increase with n like $\|I - A_{\mathcal{H}}^{-1}A\|_2 \approx \frac{n}{10} * 0.26^k$, confirming the exponential convergence with respect to the rank k. Note that equal approximation errors are obtained when k is chosen proportional to $\log n$.

Quite similar numbers as above are obtained in the case of a differential equation with smooth coefficient σ. This underlines that the smoothness or regularity of the boundary value problem does not deteriorate the approximation by $\mathcal{H}$-matrices. Tests with irregular triangulations in more complicated domains give again similar approximations.

Further examples can be seen in [1].

Acknowledgment. The numerical tests from the previous section are produced by Dr. L. Grasedyck (Leipzig).

REFERENCES

[1] M. Bebendorf and W. Hackbusch. Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients. Technical Report 21, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2002.

[2] S. Börm, L. Grasedyck, and W. Hackbusch. Introduction to hierarchical matrices with applications. Technical Report 18, Max-Planck-Institut für Mathematik in den Naturwissenschaften, Leipzig, 2002.

[3] D. Braess, M. Dryja, and W. Hackbusch. Grid transfer for nonconforming FE-discretisations with application to non-matching grids. *Computing*, 63:1–25, 1999.

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

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

[6] W. Hackbusch, B. Khoromskij, and S. A. Sauter. On $\mathcal{H}^2$-matrices. In H.-J. Bungartz, R. H. W. Hoppe, and C. Zenger, editors, *Lectures on Applied Mathematics*, pages 9–29. Springer-Verlag Berlin, 2000.

[7] B. Khoromskij and J. M. Melenk. Boundary concentrated finite element methods. Technical Report 45, Max-Planck-Institut für Mathematik in den Naturwissenschaften, 2001.