**Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig**

Partial evaluation of the discrete solution of elliptic boundary value problems

by

Wolfgang Hackbusch and Florian Drechsler

Preprint no.: 59 2012

Partial evaluation of the discrete solution of elliptic boundary value problems

Wolfgang Hackbusch
Max-Planck-Institut *Mathematik in den Naturwissenschaften*
Inselstr. 22, D-04103 Leipzig

Florian Drechsler
Max-Planck-Institut *Mathematik in den Naturwissenschaften*
Inselstr. 22, D-04103 Leipzig

Abstract

The technique of hierarchical matrices is used to construct a solution operator for a discrete elliptic boundary value problem. The solution operator can be determined once for all from a recursive domain decomposition structure. Then, given boundary values and a source term, the solution can be evaluated by applying the solution operator. The complete procedure yields all components of the solution vector. The data size and computational cost is $O(n \log^* n)$, where n is the number of unknowns.

Once the data of the solution operator are constructed, components related to small subdomains can be truncated. This reduces the storage amount and still enables a partial evaluation of the solution (restricted to the skeletons of the remaining subdomains). The latter approach is in particular suited for problems with oscillatory coefficients, where one is not interested in all details of the solution.

AMS Subject Classifications: 65F05, 65N22, 65N99
Key words: elliptic boundary value problem, finite element discretisation, hierarchical matrices, homogenisation

1 Introduction

Let $Ax = b$ be a large system of equations ($x, b \in \mathbb{R}^I$, I index set). In Linear Algebra one is used to consider only the (complete) solution vector $x \in \mathbb{R}^I$ as the solution of the system. Similarly, only the (full) inverse matrix $A^{-1} \in \mathbb{R}^{I \times I}$ seems to be the answer for the inverse mapping.

However, if $Ax = b$ is the discretisation of an elliptic boundary value problem $Lu = f$ in Ω with suitable boundary data, one is often not interested in $u(\xi)$ at *all* $\xi \in \Omega$. Instead, certain functionals are of interest. Examples of functionals may be the boundary data $\partial u/\partial n$ at $\Gamma = \partial\Omega$ in the case of given Dirichlet data or only one integral $\int_{\Gamma_0} \partial u/\partial n \, d\Gamma$ along a part $\Gamma_0 \subset \Gamma$ describing the flux through Γ_0, or u at a fixed point $\xi_0 \in \Omega$ or at several points.

A particular situation originates from differential operators $L = \operatorname{div} a(\cdot) \operatorname{grad}$ with strongly oscillatory coefficients (or coefficients with other small-scale behaviour). Since also the solution is strongly oscillatory, one is usually not interested in the complicated solution with all its details, but only in local mean values $\bar{u}$ representing the macroscopic behaviour. In the case of periodic coefficients $a(\cdot)$ one can apply homogenisation techniques leading to an approximation of $\bar{u}$. If the presuppositions for this technique do not hold, a numerical homogenisation is of interest.

A model case for the previous situation might be as follows: $a(\cdot)$ is a given, strongly oscillatory function describing the coefficients of L. A rather small step width h is used to resolve these oscillations. In principle, one can solve the resulting system $A_h x_h = b_h$, but the interesting quantity is not x_h but a smoothed version $x_H = Rx_h$ where $R = R_{H \leftarrow h}$ denotes a (weighted) restriction onto a grid of the coarse step size H. In the case of a Galerkin discretisation, V_H could be a triangulation of size H and $V_h \supset V_H$ is a refinement of the triangulation up to step size $h \ll H$. Then $x_h \in V_h$ and $x_H \in V_H$ hold, and the mapping R in $x_H = Rx_h$ is the canonical restriction as used in multi-grid methods (cf. [8, §3.6]). Hence,

$$x_H = RA_h^{-1} b_h$$

is to be solved.

One may even go a step further. In general, there is no reason to represent the right-hand side f of the differential equation with the fine step size h, instead H may be sufficient, leading to a right-hand side $b_H \in V_H$. A prolongation $b_h = Pb_H$ inserted into the previous equation leads to the next problem:

$$x_H = RA_h^{-1}Pb_H.$$

The computation of A_h^{-1} is generally expensive. To reduce this cost, we develop a method where we divide the problem recursively in subdomains and solve for the nodal values on an internal boundary $\gamma(\omega) \subset \omega$:

$$x_H(\gamma(\omega)) = \Phi_\omega b_H, \qquad \omega \subset \Omega.$$

The right-hand side b_H contains the boundary data on $\partial\omega$ and the discrete source term in the subdomain ω. Solving these subproblems for all subdomains of the domain decomposition, we compute again the whole solution x_H.
The properties of this methods are the following:

- In a first phase of the computation, the matrices associated with the maps Φ_ω are determined. The related storage and computational cost are almost linear with respect to the total dimension of the problem. Hence, for very fine step sizes h the costs are high, but because of the used domain decomposition all tasks of the same level are completely independent and therefore easily parallelisable.
- In a second phase of the computation, the solution for a certain right-hand side can be computed. The evaluation of the solution might be only partial, i.e., it stops at a certain coarser step size $H \gg h$. This reduces the storage requirements for the Φ_ω-matrices and the computing time for the solution. Also here all tasks of the same level can be computed in parallel. The partial evaluation does not change the accuracy of the result, i.e., we do not compute a coarse-grid solution, but evaluate the fine-grid solution only at coarse-grid points.
- If several linear systems are to be solved with the same matrix but different right-hand sides, the first computing phase is to be performed only once.
- A family of local functionals of the solution is easily computable.

The basis scheme of the representation of the solution is described in §2. The mappings Φ_ω used there contain partial informations of the inverse A_h^{-1} and must be constructed suitably. The corresponding method is explained in §5.1.

2 Domain decomposition tree and associated trace mappings

First, we explain the basic idea for the *exact* boundary value problem, i.e., without discretisation. The situation at the start is given by the differential equation

$$Lu_\Omega = f_\Omega \qquad \text{in } \Omega \subset \mathbb{R}^d \tag{2.1a}$$

with Dirichlet boundary condition[1]

$$u_\Omega|_{\partial\Omega} = g_{\partial\Omega} \qquad \text{on } \partial\Omega. \tag{2.1b}$$

Let $\gamma(\Omega) \subset \Omega$ be an open $(d-1)$-dimensional manifold dividing Ω into two subdomains ω_1 and ω_2 with

$$\partial\omega_1 \cap \partial\omega_2 = \overline{\gamma(\Omega)}.$$

(first step in Figure 2.1). In the following, we call $\gamma(\Omega)$ the *internal boundary*. The restriction of the solution of (2.1a,b) onto $\gamma(\Omega)$ is the trace $u_\Omega|_{\gamma(\Omega)}$. Since u_Ω depends linearly on the right-hand side f_Ω and the boundary data $g_{\partial\Omega}$, this defines is a linear mapping

$$\Phi_\Omega : (f_\Omega, g_{\partial\Omega}) \mapsto u_\Omega|_{\gamma(\Omega)}.$$

[1] The type of boundary condition is not really essential for the method. For other types, the arising boundary conditions in the later subdomains ω will be of Dirichlet type for $\partial\omega \cap \Omega$ and of the given type on $\partial\omega \cap \Gamma$.

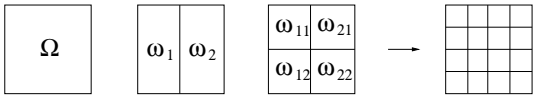

Figure 2.1: Sequence of domain decompositions

Ω forms the root of the *domain decomposition tree* G_Ω from Figure 2.2, while ω_1 and ω_2 are the sons of Ω. Before continuing the description of the method, we introduce some notations concerning the domain decomposition tree G_Ω.

Notation 2.1 *The nodes (subdomains) of the tree G_Ω are denoted by ω or ω_i. The sons of a node ω form the set $\mathcal{S}(\omega)$. If $\mathcal{S}(\omega) = \emptyset$, ω is called a leaf of G_Ω. The set of leaves forms $\mathcal{L}(G_\Omega) = \{\omega \in G_\Omega : \mathcal{S}(\omega) = \emptyset\}$. In the case of a binary tree, we have $\#\mathcal{S}(\omega) = 2$ for all $\omega \in G_\Omega \backslash \mathcal{L}(G_\Omega)$. ω is called the father of all $\omega' \in \mathcal{S}(\omega)$. The root is the unique node without a father.*

First we discuss the situation for the sons $\omega_i \in \mathcal{S}(\Omega)$ of the root Ω. For each ω_i the differential equation (2.1a) can be restricted onto ω_i:

$$Lu_{\omega_i} = f_{\omega_i} \qquad \text{in } \omega_i \ (i = 1, 2).$$

Note that the boundary $\partial\omega_i$ is the disjoint union of $\partial\omega_i \cap \partial\Omega$ and $\gamma(\omega)$. Hence, the boundary data

$$u_{\omega_i}|_{\partial\omega_i} = g_{\partial\omega_i} \qquad \text{on } \partial\omega_i$$

are directly given on the subset $\partial\omega_i \cap \partial\Omega$, where $g_{\partial\omega_i}|_{\partial\omega_i \cap \partial\Omega} = g_{\partial\Omega}|_{\partial\omega_i \cap \partial\Omega}$ with $g_{\partial\Omega}$ from (2.1b), while on $\gamma(\omega)$ the boundary data are defined via

$$g_{\omega_i}|_{\gamma(\omega)} = \Phi_\omega \left(f_\omega, g_{\partial\omega}\right).$$

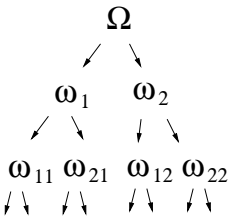

Figure 2.2: Domain decomposition tree G_Ω

Since now the boundary data on $\partial\omega_i$ are determined, the method can be continued recursively in the subdomains ω_i $(i = 1, 2)$: Boundary value problems like (2.1a,b) are defined in ω_i instead of Ω. Each ω_i can be divided by an internal boundary $\gamma(\omega_i)$ in two parts ω_{i1} and ω_{i2} (see middle part of Figure 2.1). Moreover, the mapping

$$\Phi_{\omega_i} : (f_{\omega_i}, g_{\partial\omega_i}) \mapsto u_{\omega_i}|_{\gamma(\omega_i)}$$

describes the trace of the solution on $\gamma(\omega_i)$.

After repeated application of this recursion one obtains a finer nested decomposition of Ω (as shown in the right part of Figure[2] 2.1) as well as the tree from Figure 2.2. This domain decomposition tree is denoted by G_Ω and has Ω as its root. The sons of a subdomain $\omega \in G_\Omega$ of Ω are again the parts ω_1, ω_2 of ω produced by the internal boundary $\gamma(\omega)$ (i.e., $\overline{\omega_1} \cup \overline{\omega_2} = \overline{\omega}$, $\overline{\omega_1} \cap \overline{\omega_2} = \gamma(\omega)$). For practical reasons only binary trees G_Ω are considered. In the continuous case, the decomposition can be repeated infinitely ($\mathcal{L}(G_\Omega) = \emptyset$) and because of $\overline{\bigcup_{\omega \in G_\Omega} \partial\omega} = \overline{\Omega}$ the traces $u|_{\partial\omega}$ define the complete solution u in Ω.

In the discrete case discussed below we reach the complete solution after finitely many steps.

[2]This illustration shows a regular decomposition. Of course, any kind of decomposition is possible.

3 Sketch of the discrete variant

In the following we assume that the discretisation method is the Galerkin method using finite elements of a small size h. Details of the discretisation are given in §4.1.

In this case the domain decomposition from §2 must be consistent to the finite element triangulation $T(\Omega)$, i.e., all subdomains $\omega \in G_\Omega$ are unions of triangles[3] from $T(\Omega)$. An equivalent property is that all internal boundaries $\gamma(\omega)$ coincide with edges of the triangles of $T(\Omega)$. The decomposition can be continued until the subdomains are triangles $\omega \in T(\Omega)$. Then, the leaves of the domain decomposition tree G_Ω are the triangles of $T(\Omega)$:

$$\mathcal{L}(G_\Omega) = T(\Omega). \tag{3.1}$$

For each subdomain $\omega \in G_\Omega$, which is not a leaf, a mapping $\Phi_\omega : (f_h(\overline{\omega}), g_h(\partial\omega)) \mapsto x_h|_{\gamma(\omega)}$ will be constructed. This mapping Φ_ω maps the nodal values $f_h(\overline{\omega})$ of the right-hand side f_Ω and the components $g_h(\partial\omega)$ of the boundary data $u_h|_{\partial\omega}$ to the coefficients $x_h|_{\gamma(\omega)}$ of the trace $u_h|_{\gamma(\omega)}$ of the discrete finite element solution u_h at $\gamma(\omega)$. Details concerning the underlying system and Φ_ω will follow in §4.1 and §4.5.

The discretisation of the boundary value problem in Ω produces the data $f_h(\overline{\Omega})$ and $g_h(\partial\Omega)$ for the root $\Omega \in G_\Omega$. After applying Φ_Ω, the data $g_h(\partial\omega)$ are available for the sons $\omega \in \mathcal{S}(\Omega)$ of Ω. Finally, further recursive applications of Φ_ω (see algorithm in §5.2) yield the nodal values for all triangles from $T(\Omega)$ and, therefore, all coefficients of the solution u_h. Hence, the algorithm produces the (complete) solution of the system of equations. Later, in §7.1, we will consider a variant in which the solution is not computed completely but only partially.

In a prephase, the mappings Φ_ω must be computed. The corresponding algorithm is described in §4.7. For this purpose we needs an auxiliary mapping Ψ_ω, which will be defined in §4.6.

4 Details

4.1 Finite element discretisation and matrix formulation

The finite element discretisation in Ω is based on a triangulation $T(\Omega)$. The additional *consistency property* of the domain decomposition tree G_Ω reads as follows: for all $\omega \in G_\Omega$ there is a subset $T(\omega) \subset T(\Omega)$ such that $\omega = \bigcup_{t\in T(\omega)} t$. The internal boundary $\gamma(\omega)$ necessarily consists of edges of the triangles. For practical reasons, ω should be divided by $\gamma(\omega)$ into subdomains ω_1 and ω_2 of similar size, while the length of $\gamma(\omega)$ (more precisely the number of involved nodal points) should be as small as possible. Let the leaves of the tree G_Ω be the triangles from $T(\Omega)$, i.e., the domain decomposition is continued as long as possible (see (3.1)).

Next we describe the finite element discretisation in a subdomain $\omega \in G_\Omega$. Note that the subset $T(\omega)$ described above is the triangulation of ω. Let $V_h \subset H_0^1(\omega)$ be the finite element space for the triangulation $T(\omega)$ with homogeneous boundary values on $\partial\omega$. Let the space $\bar{V}_h \supset V_h$ with $\bar{V}_h \subset H^1(\omega)$ additionally contain the triangles associated to boundary vertices. The desired finite element solution $u_h \in \bar{V}_h$ of (2.1a,b) has the variational formulation[4]

$$\begin{aligned} &a_\omega(u_h, v_h) = f_\omega(v_h) && \text{for all } v_h \in V_h, \\ &\textstyle\int_{\partial\omega} u_h w_h \mathrm{d}\Gamma = \int_{\partial\omega} g w_h \mathrm{d}\Gamma && \text{for all } w_h \in \bar{V}_h. \end{aligned} \tag{4.1}$$

The second equation[5] states that $u_h|_{\partial\omega} = g_h$ is the $L^2(\partial\omega)$-orthogonal projection of g onto $\bar{V}_h|_{\partial\omega}$: $\int_{\partial\omega} g_h w_h \mathrm{d}\Gamma = \int_{\partial\omega} g w_h \mathrm{d}\Gamma$. In the case of proper subdomains $\omega \neq \Omega$ the boundary values g will already belong to the space $\bar{V}_h|_{\partial\omega} = \{w|_{\partial\omega} : w \in \bar{V}_h\}$ so that the equality $u_h = g$ holds on $\partial\omega$. Only for $\omega = \Omega$ one has once to perform the projection of $g \in L^2(\partial\Omega)$ onto $u_h|_{\partial\Omega}$.

We denote the coefficients of the finite element function u_h by $x_{h,i}$, i.e., $x_h = (x_{h,i})_{i\in I}$ with

$$u_h = \sum_{i\in I} x_{h,i}\phi_i. \tag{4.2}$$

[3] The finite elements are called 'triangles', since this corresponds to the 2D figures. The method, however, can be used for any spatial dimension.

[4] The notation a_ω and f_ω with the index ω emphasises the integration domain which is variable.

[5] The variation over $w_h \in \bar{V}_h$ can be replaced by the variation over $w_h \in \bar{V}_h \backslash V_h$, since for $w_h \in V_h$ both sides of the equation vanish.

The involved index set $I(\overline{\omega})$ consists of all nodal points of the triangulation $T(\omega)$ including the boundary points. The disjoint decomposition into internal and boundary points yields

$$I(\overline{\omega}) = I(\omega) \cup I(\partial\omega). \tag{4.3a}$$

$I = I(\overline{\Omega})$ consist all nodal points of the triangulation $T(\Omega)$.

The system corresponding to the variational formulation (4.1) is denoted by

$$A_h(\overline{\omega})\, x_h(\overline{\omega}) = r_h(\overline{\omega}) \tag{4.3b}$$

using the subdomain $\omega \in G_\Omega$ as explicit argument. The right-hand side vector $r_h(\omega) = (r_{h,j}(\omega))_{j\in I(\omega)}$ contains the blocks $r_h(\omega)$ and $r_h(\partial\omega)$. The components of $r_h(\omega) = (r_{h,j}(\omega))_{j\in I(\omega)}$ discretise the data of f_ω (right-hand side of the differential equation):

$$r_{h,j}(\omega) = f_\omega(\phi_j) = \int_\omega f_\omega \phi_j \mathrm{d}x \qquad \text{for } j \in I(\omega), \tag{4.3c}$$

while the components of $r_h(\partial\omega) = (r_{h,j}(\partial\omega))_{j\in I(\partial\omega)}$ represent the boundary data:

$$u_h|_{\partial\omega} = \sum_{j\in I(\partial\omega)} r_{h,j}(\partial\omega)\, \phi_j \Big|_{\partial\omega}. \tag{4.3d}$$

Let $A_h = A_h(\overline{\omega})$ be the discretisation matrix corresponding to (4.1). According to the decomposition (4.3a), $A_h(\overline{\omega})$ has the form

$$A_h(\overline{\omega}) = \begin{bmatrix} A^{\omega,\omega} & A^{\omega,\partial\omega} \\ O & I \end{bmatrix}. \tag{4.3e}$$

The first equation in (4.1) yields the representation

$$\begin{aligned} A^{\omega,\omega}_{ij} &= a_\omega(\phi_j, \phi_i) \qquad \text{for } i,j \in I(\omega) \\ A^{\omega,\partial\omega}_{ij} &= a_\omega(\phi_j, \phi_i) \qquad \text{for } i \in I(\omega),\ j \in I(\partial\omega), \end{aligned} \tag{4.3f}$$

where $\{\phi_i : i \in I(\omega)\}$ is a basis of V_h and $\{\phi_i : i \in I(\overline{\omega})\}$ is a basis of $\bar{V}_h$.

Since $r_h(\partial\omega)$ already represents the coefficients of $u_h|_{\partial\omega}$ explicitly (cf. (4.3d)), the second block row $\begin{bmatrix} O & I \end{bmatrix}$ is obvious. Hence, the system is block-staggered: First one has to insert the boundary values $r_h(\partial\omega)$, then the equation of the first row can be solved for the internal nodal values $r_h(\omega)$.

It is beneficial for the multiscale version to use a projection of the right-hand side f_Ω instead of the vector $r_h(\Omega)$. In the following, $f_h \in \mathbb{R}^{I(\overline{\Omega})}$ fulfils

$$f = \sum_{i\in I} f_{h,i}\phi_i. \tag{4.4}$$

Subvectors of f_h are denoted by $f_h(\overline{\omega}) = f_h|_{I(\overline{\omega})}$. With $M_h(\overline{\omega}) \in \mathbb{R}^{I(\overline{\omega})\times I(\overline{\omega})}$ we denote the *mass matrix* with

$$(M_h(\overline{\omega}))_{ij} = \int_\omega \phi_j\phi_i dx \quad for\ i,j \in I(\overline{\omega}).$$

For later use we introduce the block structure

$$M_h(\overline{\omega}) := \begin{bmatrix} M_h^\omega \\ M_h^{\partial\omega} \end{bmatrix}$$

with $M_h^\omega \in \mathbb{R}^{I(\omega)\times I(\overline{\omega})}$ and $M_h^{\partial\omega} \in \mathbb{R}^{I(\partial\omega)\times I(\overline{\omega})}$. For these it holds $r_h(\omega) = M_h^\omega f_h(\overline{\omega})$.

4.2 Natural boundary conditions

Replacing the Dirichlet condition by the natural boundary condition (cf. [7, §7.4]), one obtains the matrix $A_h^{\mathrm{nat}}(\overline{\omega})$ with the entries

$$A^{\mathrm{nat}}_{i,j}(\overline{\omega}) = a(\phi_j, \phi_i) \qquad \text{for all } i,j \in I(\overline{\omega}). \tag{4.5}$$

The block partition of $A_h^{\mathrm{nat}}(\overline{\omega})$ into $\begin{bmatrix} A^{\mathrm{nat},\omega,\omega} & A^{\mathrm{nat},\omega,\partial\omega} \\ A^{\mathrm{nat},\partial\omega,\omega} & A^{\mathrm{nat},\partial\omega,\partial\omega} \end{bmatrix}$ yields the same blocks as (4.3e) in the first row: $A^{\omega,\omega} = A^{\mathrm{nat},\omega,\omega}$ and $A^{\omega,\partial\omega} = A^{\mathrm{nat},\omega,\partial\omega}$, but different entries in the second block row, i.e., for $(i,j) \in I(\partial\omega) \times I(\overline{\omega})$.

4.3 Interrelation of the matrices $A_h(\overline{\omega})$, $A_h(\overline{\omega_i})$, $A_h^{\rm nat}(\omega_i)$

Let the matrices A_h and $A_h^{\rm nat}$ for $\overline{\omega}$, $\overline{\omega_1}$, $\overline{\omega_2}$ be defined as above.

Remark 4.1 *The index range $\alpha \in I(\omega)$ corresponds to the first block row in (4.3e). For these indices we have*

$$(A_h(\overline{\omega}))_{\alpha,\beta} = \begin{cases} (A_h(\overline{\omega_i}))_{\alpha,\beta} & \text{for } \alpha \in I(\omega_i), \quad \beta \in I(\overline{\omega_i}), \quad i = 1,2, \\ (A_h^{\rm nat}(\omega_1))_{\alpha,\beta} + (A_h^{\rm nat}(\omega_2))_{\alpha,\beta} & \text{for } \alpha \in I(\gamma(\omega)), \quad \beta \in I(\partial\omega_1) \cap I(\partial\omega_2), \\ 0 & \text{for } \alpha \in I(\omega_i), \quad \beta \in I(\overline{\omega_j}), \quad i \neq j. \end{cases}$$

In the middle case, integration involves parts of ω_1 as well as of ω_2 (cf. Figure 4.1). The case $\alpha \in I(\partial\omega)$ (second block row in (4.3e)) is omitted because of the trivial structure.

Proof. 1) For $\alpha \in I(\omega_i)$ and $\beta \in I(\overline{\omega_i})$ the intersection of the supports of the basis functions ϕ_α, ϕ_β lies completely in $\overline{\omega_i} \subset \overline{\omega}$. Hence, $a_\omega(\phi_\beta, \phi_\alpha)$ and $a_{\omega_i}(\phi_\beta, \phi_\alpha)$ coincide.

2) For $\alpha, \beta \in I(\gamma(\omega)) \subset I(\partial\omega_1) \cap I(\partial\omega_2)$ the intersection of the supports of ϕ_α, ϕ_β lies partly in ω_1 and partly in ω_2, so that $a_\omega(\phi_\beta, \phi_\alpha) = a_{\omega_1}(\phi_\beta, \phi_\alpha) + a_{\omega_2}(\phi_\beta, \phi_\alpha)$.

3) For $\alpha \in I(\omega_i)$ and $\beta \in I(\overline{\omega_j})$ $(i \neq j)$ the supports of ϕ_α, ϕ_β are disjoint. ∎

4.4 Decomposition of the index set

Each subdomain $\omega \in G_\Omega$ is divided by the internal boundary $\gamma(\omega)$ into the subdomains ω_1 and ω_2, the sons of ω. The index sets of these three subdomains are $I(\overline{\omega})$, $I(\overline{\omega_1})$, $I(\overline{\omega_2})$. Each of these index sets decomposes into the disjoint subsets of inner and boundary points:

$$I(\overline{\omega}) = I(\omega) \cup I(\partial\omega), \qquad I(\overline{\omega_1}) = I(\omega_1) \cup I(\partial\omega_1), \qquad I(\overline{\omega_2}) = I(\omega_2) \cup I(\partial\omega_2).$$

The internal boundary $\gamma(\omega)$ consists of edges of the triangles of $T(\omega)$. Correspondingly, $I(\gamma(\omega))$ contains the nodal points in $\gamma(\omega)$. Since by definition $\gamma(\omega) \subset \omega$ is open, the sets $I(\gamma(\omega))$ and $I(\partial\omega)$ do not intersect. Hence, $I(\gamma(\omega))$ is a proper subset of $I(\overline{\omega_1}) \cap I(\overline{\omega_2})$:

$$I(\gamma(\omega)) := (I(\overline{\omega_1}) \cap I(\overline{\omega_2})) \setminus I(\partial\omega).$$

For an illustration let $\begin{matrix} a & a & a & s & b & b & b \\ a & 1 & 1 & \gamma & 2 & 2 & b \\ a & 1 & 1 & \gamma & 2 & 2 & b \\ a & a & a & s & b & b & b \end{matrix}$ be a grid with nodal points denoted by $a, b, \gamma, s, 1, 2$.

$I(\overline{\omega})$ is the set of all points. Points named γ or s indicate the separating internal boundary $\overline{\gamma(\omega)}$. The other index sets are characterised as follows:

index set	$I(\omega)$	$I(\partial\omega)$	$I(\gamma(\omega))$	$I(\omega_1)$	$I(\omega_2)$	$I(\overline{\omega_1})$	$I(\overline{\omega_2})$	$I(\partial\omega_1)$	$I(\partial\omega_2)$
notations	$1, \gamma, 2$	a, s, b	γ	1	2	$a, 1, s, \gamma$	$b, 2, s, \gamma$	$1, s, \gamma$	$2, s, \gamma$

4.5 The mapping Φ_ω

Let $\omega \in G_\Omega$ be no leaf. Then ω is associated with an internal boundary $\gamma(\omega)$, which determines the decomposition. In §2, Φ_ω has been defined as the mapping of the right-hand side and the boundary data into the trace $u|_{\gamma(\omega)}$. In the discrete case we use the same symbol Φ_ω, but we replace the functions by the representing coefficients. $f_h(\overline{\omega})$ is the discrete analogue of the right-hand side f_ω (cf. (4.4)), while $g_h(\partial\omega)$ replaces the boundary values $u_h|_{\partial\omega}$ (cf. (4.3d)). The trace $u_h|_{\gamma(\omega)}$ is represented by the coefficients $x_h|_{I(\gamma(\omega))} = (x_{h,i})_{i \in I(\gamma(\omega))}$ (cf. (4.2)). The system (4.3e) yields the solution $x_h = (A^{\omega,\omega})^{-1} \left(M_h^\omega f_h(\overline{\omega}) - A^{\omega,\partial\omega} g_h(\partial\omega) \right)$ for the solution vector $x_h \in \mathbb{R}^{I(\overline{\omega})}$. Partial evaluation on $I(\gamma(\omega))$ yields

$$\Phi_\omega \left(f_h(\overline{\omega}), g_h(\omega) \right)) := (A^{\omega,\omega})^{-1} \left(M_h^\omega f_h(\overline{\omega}) - A^{\omega,\partial\omega} g_h(\partial\omega) \right) \Big|_{I(\gamma(\omega))}. \tag{4.6}$$

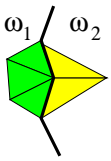

Figure 4.1: Triangles involved in the integration for the central nodal point x_i.

Although (4.6) defines the mapping Φ_ω, this equation will not be used for the practical construction of Φ_ω (see §4.7).

Assume that Φ_ω is available and let ω_i $(i = 1, 2)$ denote the subdomains obtained by one domain decomposition step (sons of ω in the tree G_Ω). It is essential that for each ω_i complete Dirichlet boundary data on $\partial\omega_i$ are available, which can be seen as follows. The index set $I(\partial\omega_i)$ is the disjoint union of $I(\partial\omega_i) \cap I(\partial\omega)$ and $I(\gamma(\omega))$. On $I(\partial\omega_i) \cap I(\partial\omega)$ the nodal values are available from the data on $I(\partial\omega)$, while on $I(\gamma(\omega))$ the values are given by $\Phi_\omega(f_h(\overline{\omega}), g_h(\omega))$.

The finite element discretisation in ω_i uses the triangulation $T(\omega_i)$ containing all triangles in ω_i. The discretisation matrix for ω_i $(i = 1, 2)$ is $A_h(\overline{\omega_i})$. It is also given by (4.3b-e), where now the bilinear form $a = a_{\omega_i}$ uses the integration over ω_i (instead of ω, see Figure 4.1) and the boundary integral on $\partial\omega_i = (\partial\omega \cap \overline{\omega_i}) \cup \gamma(\omega)$ instead of $\partial\omega$.

4.6 The mapping Ψ_ω

For the construction of the mapping Φ_ω (cf. (4.6)) we need a further mapping

$$\Psi_\omega : \mathbb{R}^{I(\overline{\omega})} \times \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}.$$

Ψ_ω is applied to $f_h(\overline{\omega}) \in \mathbb{R}^{I(\overline{\omega})}$ and $g_h(\partial\omega) \in \mathbb{R}^{I(\partial\omega)}$. We remind that $f_h(\overline{\omega})$ are the nodal values representing the right-hand side f_Ω, while $g_h(\partial\omega)$ are the boundary values of the discrete solution $x_h(\overline{\omega}) = \big(x_h|_{I(\omega)}, x_h|_{I(\partial\omega)}\big) \in \mathbb{R}^{I(\overline{\omega})}$, i.e.,

$$x_h|_{I(\partial\omega)} = g_h(\partial\omega).$$

According to the decomposition (4.3e), the internal coefficients $x_h|_{I(\omega)}$ are the solution of $A^{\omega,\omega} x_h|_{I(\omega)} + A^{\omega,\partial\omega} x_h|_{I(\partial\omega)} = r_h(\omega)$, so that

$$x_h|_{I(\omega)} = (A^{\omega,\omega})^{-1} \left(M_h^\omega f_h(\overline{\omega}) - A^{\omega,\partial\omega} g_h(\partial\omega) \right) \tag{4.7}$$

follows (compare (4.6) for the further restriction on $\gamma(\omega)$).

For boundary nodal indices $i \in I(\partial\omega)$ we define the i-th component of $\Psi_\omega(f_h(\overline{\omega}), g_h(\partial\omega))$ by means of the finite element solution $u_h = \sum_{j \in I(\overline{\omega})} x_{h,j} \phi_j$ as follows:

$$\big(\Psi_\omega(f_h(\overline{\omega}), g_h(\omega))\big)_i := a_\omega(u_h, \phi_i) - f_\omega(\phi_i) = \sum_{j \in I(\overline{\omega})} x_{h,j}\, a_\omega(\phi_j, \phi_i) - f_\omega(\phi_i) \qquad \text{for } i \in I(\partial\omega). \tag{4.8}$$

Note that $a_\omega(\cdot,\cdot)$ is the bilinear form from (4.1) with the integration restricted to ω.

The coefficients $a_\omega(\phi_j, \phi_i)$ of the latter equation represent the part

$$A^{\mathrm{nat},\partial\omega} = \big(A^{\mathrm{nat},\partial\omega,\partial\omega}, A^{\mathrm{nat},\partial\omega,\omega}\big) \qquad \text{with}$$
$$A^{\mathrm{nat},\partial\omega,\partial\omega} := \big(A^{\mathrm{nat}}_{i,j}\big)_{i,j \in I(\partial\omega)}, \quad A^{\mathrm{nat},\partial\omega,\omega} := \big(A^{\mathrm{nat}}_{i,j}\big)_{i \in I(\partial\omega), j \in I(\omega)}$$

of the matrix $A^{\mathrm{nat}} = A^{\mathrm{nat}}(\overline{\omega})$ from (4.5), so that $\Psi_\omega(f_h(\overline{\omega}, g_h(\partial\omega))) = A^{\mathrm{nat},\partial\omega} x_h$. Using the previous equations for x_h, we obtain the block representation

$$\Psi_\omega = \big(\Psi_\omega^{\overline{\omega}}, \Psi_\omega^{\partial\omega}\big) \qquad \text{with} \quad \begin{cases} \Psi_\omega^{\overline{\omega}} := -M_h^{\partial\omega} + A^{\mathrm{nat},\partial\omega,\omega} \left(A^{\omega,\omega}\right)^{-1} M_h(\omega), \\ \Psi_\omega^{\partial\omega} := A^{\mathrm{nat},\partial\omega,\partial\omega} - A^{\mathrm{nat},\partial\omega,\omega} \left(A^{\omega,\omega}\right)^{-1} A^{\mathrm{nat},\omega,\partial\omega}. \end{cases} \tag{4.9}$$

Remark 4.2 *Let ω be a triangle of the triangulation $T(\Omega)$ and hence a leaf of the domain decomposition tree G_Ω. For such an ω the mapping Ψ_ω is easily computable since $\#I(\omega) = 0$. In this case the formulae are reduced to*

$$\begin{aligned} \Psi_\omega^{\overline{\omega}} &= -M_h^{\partial\omega}, \\ \Psi_\omega^{\partial\omega} &= A^{\mathrm{nat},\partial\omega,\partial\omega}. \end{aligned} \tag{4.10}$$

Next, we perform the construction of Ψ_ω recursively from the leaves of the tree G_Ω to the root Ω.

4.7 Construction of Φ_ω from Ψ_{ω_1} and Ψ_{ω_2}

Let ω_1 and ω_2 be the sons of ω in the domain decomposition tree. The mappings Ψ_{ω_1} and Ψ_{ω_2} associated to ω_1 and ω_2 are assumed to be available. The mappings Φ_ω, Ψ_ω are to be determined. The arguments of the linear mapping Ψ_ω are $f_h(\overline{\omega})$ and $g_h(\omega)$. These data are used in the following Steps 1a-c to formulate a discrete boundary value problem in ω_1.

Step 1a) *Boundary data on $\partial\omega_1$:* $g_h(\partial\omega)$ coincides with the boundary data $x_h(\overline{\omega})|_{I(\partial\omega)}$. This defines in particular the boundary data $x_h(\overline{\omega_1})|_{I(\partial\omega)\cap I(\partial\omega_1)}$. The boundary index set $I(\partial\omega_1) \subset I(\overline{\omega_1})$ can be split disjointly into $I(\partial\omega) \cap I(\partial\omega_1)$ and $I(\gamma(\omega))$ (the latter nodal points are denoted by γ in the illustration in §4.4). The remaining boundary data $x_\gamma := x_h(\overline{\omega_1})|_{I(\gamma(\omega))}$ are still to be determined.

Step 1b) *Data for $\overline{\omega_1}$:* The restriction of $f_h(\overline{\omega})$ onto the index set $I(\overline{\omega_1})$ yields $f_h(\overline{\omega_1})$.

Step 1c) *Discrete boundary value problem:* The vector $x^{(1)} := x_h(\overline{\omega_1})$ is written as $\left(x_\omega^{(1)}, x_{\partial\omega}^{(1)}, x_\gamma^{(1)}\right)$ using the following block decomposition:

$$x_\omega^{(1)} = x_h(\overline{\omega_1})|_{I(\omega_1)}, \qquad x_{\partial\omega}^{(1)} = x_h(\overline{\omega_1})|_{I(\partial\omega)\cap I(\partial\omega_1)}, \qquad x_\gamma^{(1)} = x_h(\overline{\omega_1})|_{I(\gamma(\omega))}.$$

For given boundary data $\left(x_{\partial\omega}^{(1)}, x_\gamma^{(1)}\right)$ the system for the inner nodal values $x_\omega^{(1)}$ reads as follows:

$$A^{(1),\omega,\omega} x_\omega^{(1)} = M^{(1),\omega,\overline{\omega}} f_h(\overline{\omega_1}) - A^{(1),\omega,\partial\omega} x_{\partial\omega}^{(1)} - A^{(1),\omega,\gamma} x_\gamma^{(1)} \qquad \text{with} \tag{4.11a}$$

$$\begin{aligned} A^{(1),\omega,\omega} &:= \left((A_h(\overline{\omega_1}))_{ij} \right)_{i,j\in I(\omega_1)}, \\ A^{(1),\omega,\partial\omega} &:= \left((A_h(\overline{\omega_1}))_{ij} \right)_{i\in I(\omega_1),\, j\in I(\partial\omega)\cap I(\partial\omega_1)}, \\ A^{(1),\omega,\gamma} &:= \left((A_h(\overline{\omega_1}))_{ij} \right)_{i\in I(\omega_1), j\in I(\gamma(\omega))}, \\ M^{(1),\omega,\overline{\omega}} &:= \left((M_h(\overline{\omega_1}))_{ij} \right)_{i\in I(\omega_1), j\in I(\overline{\omega_1})}. \end{aligned} \tag{4.11b}$$

Step 2) Analogous arguments for the second subdomain ω_2 yield

$$A^{(2),\omega,\omega} x_\omega^{(2)} = M^{(2),\omega,\overline{\omega}} f_h(\overline{\omega_2}) - A^{(2),\omega,\partial\omega} x_{\partial\omega}^{(2)} - A^{(2),\omega,\gamma} x_\gamma^{(2)} \tag{4.11c}$$

with similar definitions of the block matrices.

Step 3) *Equation for x_γ:* The components of the vector $x^{(1)} := x_h(\overline{\omega_1})$ are defined for $j \in I(\overline{\omega_1})$, those of $x^{(2)} := x_h(\overline{\omega_2})$ for $j \in I(\overline{\omega_2})$. The domains of definition overlap in $I_s := I(\overline{\omega_1}) \cap I(\overline{\omega_2})$. For $j \in I_s \cap I(\partial\omega) = I_s \backslash I(\gamma(\omega))$ we have $x_j^{(1)} = x_{\partial\omega,j}^{(1)} = (g_h(\partial\omega))_j = x_{\partial\omega,j}^{(2)} = x_j^{(2)}$ (cf. Step 1a). For $j \in I(\gamma(\omega))$ we require the corresponding identity describing the coincidence of the boundary values at the internal boundary:

$$x_\gamma^{(1)} := x_\gamma^{(2)} := x_\gamma \in \mathbb{R}^{I(\gamma(\omega))}. \tag{4.11d}$$

Under this consistency condition, the vectors $x^{(1)}$ and $x^{(2)}$ define a unique vector $x_h(\overline{\omega}) \in \mathbb{R}^{I(\overline{\omega})}$ by means of

$$x_{h,j}(\overline{\omega}) := \begin{cases} x_j^{(1)} & \text{for } j \in I(\overline{\omega_1}), \\ x_j^{(2)} & \text{for } j \in I(\overline{\omega_2}), \end{cases} \tag{4.11e}$$

since $x_j^{(1)} = x_j^{(2)}$ for $j \in I(\overline{\omega_1}) \cap I(\overline{\omega_2})$. Vice versa, each $x_h(\overline{\omega}) \in \mathbb{R}^{I(\overline{\omega})}$ defines vectors $x^{(i)} := x_h(\overline{\omega})|_{I(\overline{\omega_i})}$ $(i = 1, 2)$ satisfying the consistency condition (4.11d).

For any $x_\gamma \in \mathbb{R}^{I(\gamma(\omega))}$, the equations (4.11a,b) determine $x^{(1)}$ and $x^{(2)}$. The latter vectors define $x_h(\overline{\omega})$ by means of (4.11e) and hence $u_h(\omega) := \sum_{j\in I(\overline{\omega})} x_{h,j}(\overline{\omega})\,\phi_j$. To determine x_γ, we use the equation

$$a_\omega(u_h,\phi_j) - f_\omega(\phi_j) = 0 \qquad \text{for all } j \in I(\gamma(\omega)). \tag{4.11f}$$

The bilinear form can be written as

$$a_\omega(u_h,\phi_j) = a_{\omega_1}(u_h,\phi_j) + a_{\omega_2}(u_h,\phi_j) \tag{4.11g}$$

and the vector $f_\omega(\phi_j)$ can be written as

$$f_\omega(\phi_j) = f_{\omega_1}(\phi_j) + f_{\omega_2}(\phi_j). \tag{4.11h}$$

Since $j \in I(\gamma(\omega))$ is an index of a nodal point on the internal boundary, the situation shown in Figure 4.1 holds. By definition we have

$$\begin{aligned} a_{\omega_1}(u_h,\phi_j) - f_{\omega_1}(\phi_j) &= \left(\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))\right)_j, \\ a_{\omega_2}(u_h,\phi_j) - f_{\omega_2}(\phi_j) &= \left(\Psi_{\omega_2}(f_h(\overline{\omega_2}), g_h(\partial\omega_2))\right)_j \qquad \text{for } j \in I(\gamma(\omega)). \end{aligned}$$

Restriction of the range $\mathbb{R}^{I(\partial\omega_1)}$ of $\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))$ onto $\mathbb{R}^{I(\gamma(\omega))}$ yields $\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))|_{I(\gamma(\omega))}$. The data $g_h(\partial\omega_1) \in \mathbb{R}^{I(\partial\omega_1)}$ can be written in the block form $(g_h(\Gamma_1), x_\gamma)$, $g_h(\Gamma_1) := g_h(\partial\omega_1)|_{I(\partial\omega_1)\setminus I(\gamma(\omega))}$ where we separate the fixed values $g_h(\partial\omega_1)|_{I(\gamma(\omega))} = x_\gamma$ at the internal boundary. Hence, the linear mapping $\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))|_{I(\gamma(\omega))}$ is of the form

$$\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))|_{I(\gamma(\omega))} = \Psi^\gamma_{1\gamma} x_\gamma + \Psi^\gamma_{1\partial\omega} g_h(\Gamma_1) + \Psi^\gamma_{1\overline{\omega}} f_h(\overline{\omega_1}) \tag{4.11i}$$

with suitable matrices

$$\Psi^\gamma_{1\gamma} := \Psi^{\partial\omega}_\omega|_{I(\gamma)\times I(\gamma)}, \quad \Psi^\gamma_{1\partial\omega} := \Psi^{\partial\omega}_\omega|_{I(\gamma)\times I(\partial\omega_1\setminus\gamma)}, \text{ and } \Psi^\gamma_{1\overline{\omega}} := \Psi^{\overline{\omega}}_\omega|_{I(\gamma)\times I(\overline{\omega_1})}.$$

Analogously, $\Psi_{\omega_2}(f_h(\overline{\omega_2}), g_h(\partial\omega_2))|_{I(\gamma(\omega))}$ can be written in the form

$$\Psi_{\omega_1}(f_h(\overline{\omega_2}), g_h(\partial\omega))|_{I(\gamma(\omega))} = \Psi^\gamma_{2\gamma} x_\gamma + \Psi^\gamma_{2\partial\omega} g_h(\Gamma_2) + \Psi^\gamma_{2\overline{\omega}} f_h(\overline{\omega_2}). \tag{4.11j}$$

Equations (4.11f,i,j) lead to

$$\left(\Psi^\gamma_{1\gamma} + \Psi^\gamma_{2\gamma}\right) x_\gamma = -\Psi^\gamma_{1\partial\omega} g_h(\Gamma_1) - \Psi^\gamma_{2\partial\omega} g_h(\Gamma_2) - \Psi^\gamma_{1\overline{\omega}} f_h(\overline{\omega_1}) - \Psi^\gamma_{2\overline{\omega}} f_h(\overline{\omega_2}). \tag{4.12}$$

Inversion yields the representation of the trace values on $I(\gamma(\omega))$:

$$x_\gamma = \left(\Psi^\gamma_{1\gamma} + \Psi^\gamma_{2\gamma}\right)^{-1} \left(- \Psi^\gamma_{1\partial\omega} g_h(\Gamma_1) - \Psi^\gamma_{2\partial\omega} g_h(\Gamma_2) - \Psi^\gamma_{1\overline{\omega}} f_h(\overline{\omega_1}) - \Psi^\gamma_{2\overline{\omega}} f_h(\overline{\omega_1})\right). \tag{4.13}$$

With

$$\Phi^{\partial\omega}_\omega := \left(\Psi^\gamma_{1\gamma} + \Psi^\gamma_{2\gamma}\right)^{-1} \left(-\Psi^\gamma_{1\partial\omega} - \Psi^\gamma_{2\partial\omega}\right) \tag{4.14}$$

and

$$\Phi^{\overline{\omega}}_\omega := \left(\Psi^\gamma_{1\gamma} + \Psi^\gamma_{2\gamma}\right)^{-1} \left(-\Psi^\gamma_{1\overline{\omega}} - \Psi^\gamma_{1\overline{\omega}}\right). \tag{4.15}$$

we have found the desired mapping $\Phi_\omega := (\Phi^{\partial\omega}_\omega, \Phi^{\partial\omega}_\omega)$.

Remark 4.3 *The matrix representation of Φ_ω is given by the matrix blocks*

$$(-\Psi^\gamma_{1\partial\omega} - \Psi^\gamma_{2\partial\omega}) \in \mathbb{R}^{I(\gamma(\omega))\times I(\partial\omega)}, \quad (-\Psi^\gamma_{1\overline{\omega}} - \Psi^\gamma_{1\overline{\omega}}) \in \mathbb{R}^{I(\gamma(\omega))\times I(\overline{\omega_1})}$$

and $\left(\Psi^\gamma_{1\gamma} + \Psi^\gamma_{2\gamma}\right)^{-1} \in \mathbb{R}^{I(\gamma(\omega))\times I(\gamma(\omega))}$. However, note that the index sets $I(\overline{\omega_1})$ and $I(\overline{\omega_2})$ overlap (the common indices correspond to the notation s or γ of the nodal points in §4.4). Hence, the block contributions in the common index domain $(I(\overline{\omega_1}) \cap I(\overline{\omega_2}))$ must be added.

Lemma 4.4 *The right-hand side in (4.13) defines the desired trace mapping $\Phi_\omega : (f_h(\overline{\omega}), g_h(\partial\gamma)) \mapsto x_\gamma$.*

Proof. Let x_γ be defined by (4.13). In Step 3, the vectors $x^{(1)}$ and $x^{(2)}$ are determined by means of (4.11a,b). $u_h(\omega) = \sum_{j\in I(\overline{\omega})} x_{h,j}(\overline{\omega})\,\phi_j$ is defined via (4.11e). The equations (4.11a,b) are equivalent to $a_\omega(u_h,\phi_j) = f(\phi_j)$ for $j \in I(\omega_1)$ and $j \in I(\omega_2)$. The definition of x_γ ensures $a_\omega(u_h,\phi_j) = f(\phi_j)$ for $j \in I(\gamma(\omega))$. Since $I(\omega_1) \cup I(\omega_2) \cup I(\gamma(\omega)) = I(\omega)$, the function u_h satisfies the finite element equation $a_\omega(u_h, v_h) = f_\omega(v_h)$ $(v_h \in V_h)$ in ω. The boundary values of u_h are given by the nodal values $r_h(\partial\omega)$. Since x_γ are the nodal values of $u_h|_{\gamma(\omega)}$, the assertion is proved. ∎

4.8 Construction of Ψ_ω from Ψ_{ω_1} and Ψ_{ω_2}

Let $\omega \in G_\Omega$ be a domain with the sons (subdomains) ω_1 and ω_2. We have to construct the mapping Ψ_ω which should fulfil

$$(\Psi_\omega(f_h(\overline{\omega}), g_h(\partial\omega)))_i = a_\omega(u_h, \phi_i) - f_\omega(\phi_i) \quad \forall i \in I(\partial\omega).$$

It holds (cf. (4.11f,g,h))

$$a_\omega(u_h, \phi_i) - f_\omega(\phi_i) = (\Psi_{\omega_1}((f_h(\overline{\omega_1}), g_h(\partial\omega_1))) + \Psi_{\omega_2}((f_h(\overline{\omega_2}), g_h(\partial\omega_2))))_i \quad \forall i \in I(\partial\omega).$$

The mappings Ψ_{ω_1} and Ψ_{ω_2} depends on the boundary data x_γ, which we have to eliminate by means of the mapping Φ_ω. In this case we shrink the range of both mappings to $\mathbb{R}^{\partial\omega}$. Thus the linear mapping Ψ_{ω_1} is of the form

$$\Psi_{\omega_1}(f_h(\overline{\omega_1}), g_h(\partial\omega_1))|_{I(\partial\omega)} = \Psi_{1\gamma}^{\partial\omega_1\setminus\gamma} x_\gamma + \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma} g_h(\Gamma_1) + \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma} f_h(\overline{\omega_1}) \tag{4.16}$$

with suitable matrices

$$\Psi_{1\gamma}^{\partial\omega_1\setminus\gamma} := \Psi_\omega^{\partial\omega}|_{I(\partial\omega_1\setminus\gamma)\times I(\gamma)}, \quad \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma} := \Psi_\omega^{\partial\omega}|_{I(\partial\omega_1\setminus\gamma)\times I(\partial\omega_1\setminus\gamma)}, \text{ and } \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma} := \Psi_\omega^{\overline{\omega}}|_{I(\partial\omega_1\setminus\gamma)\times I(\overline{\omega_1})}.$$

Analogously, $\Psi_{\omega_2}(f_h(\overline{\omega_2}), g_h(\partial\omega_2))|_{I(\partial\omega)}$ can be written in the form

$$\Psi_{\omega_2}(f_h(\overline{\omega_2}), g_h(\partial\omega))|_{I(\partial\omega)} = \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)} x_\gamma + \Psi_{2\partial\omega}^{\partial\omega_2\setminus\gamma(\omega)} g_h(\Gamma_2) + \Psi_{2\overline{\omega}}^{\partial\omega_2\setminus\gamma(\omega)} f_h(\overline{\omega_2}). \tag{4.17}$$

Replacing x_γ with $\Phi_\omega(f_h(\overline{\omega}), g_h(\partial\omega))$ yields

$$\begin{aligned}
\Psi_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) =& \Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} x_\gamma + \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma(\omega)} g_h(\Gamma_1) + \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma(\omega)} f_h(\overline{\omega_1}) \\
&+ \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)} x_\gamma + \Psi_{2\partial\omega}^{\partial\omega_2\setminus\gamma(\omega)} g_h(\Gamma_2) + \Psi_{2\overline{\omega}}^{\partial\omega_2\setminus\gamma(\omega)} f_h(\overline{\omega_2}) \\
=& \Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} \Phi_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) + \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma(\omega)} g_h(\Gamma_1) + \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma(\omega)} f_h(\overline{\omega_1}) \\
&+ \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)} \Phi_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) + \Psi_{2\partial\omega}^{\partial\omega_2\setminus\gamma(\omega)} g_h(\Gamma_2) + \Psi_{2\overline{\omega}}^{\partial\omega_2\setminus\gamma(\omega)} f_h(\overline{\omega_2}).
\end{aligned}$$

This leads to $\Psi_\omega = (\Psi_\omega^{\overline{\omega}}, \Psi_\omega^{\partial\omega})$ with

$$\Psi_\omega^{\overline{\omega}} = \left(\Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)}\right) \Phi_\omega^{\overline{\omega}} + \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\overline{\omega}}^{\partial\omega_2\setminus\gamma(\omega)} \tag{4.18}$$

and

$$\Psi_\omega^{\partial\omega} = \left(\Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)}\right) \Phi_\omega^{\partial\omega} + \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\partial\omega}^{\partial\omega_2\setminus\gamma(\omega)}. \tag{4.19}$$

5 Basic algorithm

In the *definition phase*, we determine the mappings Φ_ω for all domains $\omega \in G_\Omega\setminus\mathcal{L}(G_\Omega)$ of the domain decomposition tree which are no leaves. The auxiliary mappings Ψ_ω for $\omega \in G_\Omega\setminus\{\Omega\}$ are determined only for intermediate use. Afterwards, the *evaluation phase* can be applied once or many times for different data (f_h, g_h).

5.1 Definition phase

The algorithm uses induction from the leaves of G_Ω to the root.

- The *start* of the algorithm is the determination of

  $$\Psi_\omega \qquad \text{for all } \omega \in \mathcal{L}(G_\Omega).$$

  Since $\mathcal{L}(G_\Omega) = T(\Omega)$ is assumed, Remark 4.2 holds: Ψ_ω is easily computable.

- *Induction* (from the sons ω_1, ω_2 to the father ω): Ψ_{ω_1} and Ψ_{ω_2} are assumed to be given. We construct Φ_ω according to §4.7. Using Φ_ω we can compute Ψ_ω as described in §4.8. As soon as Ψ_ω is determined, Ψ_{ω_1} and Ψ_{ω_2} are no more needed. In a concrete implementation the storage for Ψ_{ω_1} and Ψ_{ω_2} can be released.

For the algorithmic performance it is advantageous to split the tree G_Ω into the level sets $G_\Omega^{(\ell)}$, $0 \le \ell \le \mathit{depth}(G_\Omega)$, defined recursively by $G_\Omega^{(0)} := \{\Omega\}$ and $G_\Omega^{(\ell)} := \left\{\omega \in G_\Omega : \omega \text{ has a father in } G_\Omega^{(\ell-1)}\right\}$ for $\ell > 0$. The depth $\mathit{depth}(G_\Omega)$ of G_Ω is the largest number ℓ with $G_\Omega^{(\ell)} \neq \emptyset$.

```
for ℓ := depth(G_Ω) − 1 downto 0 do
begin for all ω ∈ G_Ω^(ℓ+1) ∩ L(G_Ω) do compute Ψ_ω explicitly;        {see Remark 4.2}
    for all ω ∈ G_Ω^(ℓ) \ L(G_Ω) do
    begin {ω_1, ω_2} := S_{G_Ω}(ω);                                  {S_{G_Ω} is the set of sons}
        determine the matrix corresponding to Φ_ω according to Remark 4.3;
        if ℓ > 0 then determine the matrix corresponding to Ψ_ω according to Chap. 4.8;
        delete the matrices corresponding to Ψ_{ω_1} and Ψ_{ω_2}
end end;
```
(5.1)

When the loop is finished, all Φ_ω for $\omega \in G_\Omega \backslash \mathcal{L}(G_\Omega)$ are determined, but no Ψ_ω is stored any more.

Remark 5.1 *Since all matrix operations are described in exact arithmetic, the method (5.1) are well-defined but very costly with respect to storage and computer time. The inverse matrices in (4.9) and (4.13) lead to dense matrices representing Φ_ω and Ψ_ω.*

These difficulties will be overcome in §6.

5.2 Evaluation phase

Starting from the boundary value formulation (2.1a,b), one has first to determine the values of $\big(f_h(\overline{\Omega}), g_h(\partial\Omega)\big)$ from f_Ω and $g_{\partial\Omega}$. Both parts are determined by the $L^2(\partial\Omega)$-orthogonal projection of f_Ω and g_Γ onto the ansatz space (second row in (4.1)).

The previous formulation uses the input data $(f_h(\omega), g_h(\partial\omega))$ and determines the output data $x_h(\overline{\omega})$. Because of the identity $g_h(\overline{\omega})|_{I(\partial\omega)} = x_h(\overline{\omega})|_{I(\partial\omega)}$ it is more reasonable to use only $f_h(\overline{\omega})$ (without $g_h(\partial\omega)$) as pure input data, while the vector $x_h(\overline{\omega})$ is used as *in- and output* vector carrying the required (input) boundary data on $\partial\omega$.

In the following procedure, the last two arguments f and x may be considered as vectors $f_h(\overline{\Omega}) \in \mathbb{R}^{I(\overline{\Omega})}$ and $x = x_h(\overline{\Omega}) \in \mathbb{R}^{I(\overline{\Omega})}$ in the complete domain Ω, where however only the parts $f_h(\overline{\omega})$ and $x_h(\overline{\omega}) = x|_{I(\overline{\omega})}$ are used (and changed). $f_h(\overline{\omega})$ is pure input. $x|_{I(\partial\omega)}$ is the input of the boundary value data, whereas $x|_{I(\gamma(\omega))}$ is the output. The first argument ω of the procedure *trace* must belong to $G_\Omega \backslash \mathcal{L}(G_\Omega)$.

```
procedure trace(ω, f, x);      {ω ∈ G_Ω \ L(G_Ω), f ∈ R^{I(Ω̄)}, x ∈ R^{I(Ω̄)}}
x|_{I(γ(ω))} := Φ_ω ( f|_{I(ω̄)}, x|_{I(∂ω)} );
```
(5.2)

In order to obtain the complete solution $x_h(\overline{\omega})$ in ω, one has to call *trace* recursively:

```
procedure complete_evaluation(ω, f, x);
{input: boundary values x|_{I(∂ω)} and right-hand side f|_{I(ω̄)}}
begin trace(ω, f, x);
    for ω' ∈ S_{G_Ω}(ω) do complete_evaluation(ω', f, x);
end;
```
(5.3)

The recursion in (5.3) terminates, when $\omega \in G_\Omega$ is a leaf and therefore the son set $\mathcal{S}_{G_\Omega}(\omega)$ is empty. The procedure (5.3) ensures a complete evaluation if the assumption $\mathcal{L}(G_\Omega) = T(\Omega)$ from above holds. In this case all nodal values[6] in $\overline{\omega}$ are determined.

The call *complete_evaluation*$(\Omega, f_h(\Omega), x_h(\overline{\Omega}))$ produces the solution $x_h(\overline{\Omega})$ of the discrete boundary value problem in Ω.

[6]This holds only for finite element nodes lying on the boundary of the element. If there are inner nodes (so-called bubble functions), one has still to solve for these degrees of freedom in all triangles. Since one can eliminate these degrees of freedom from the beginning, one may assume without loss of generality that only boundary nodes are present.

Remark 5.2 *???In Chap. 3 we have remarked that the leaves of G_Ω are the triangles of $T(\Omega)$. It is beneficially that for $\omega \in \mathcal{L}(G_\Omega)$ $I(\omega) > 0$ holds if we want to compute the whole solution $x_h(\overline{\Omega})$. In that case we have to solve the equation system (4.7). To compute the solution for many different data (f_h, g_h), we have to store the mappings of the equations system (4.7). These mappings are needed to compute the auxiliary matrices Ψ_ω for $\omega \in \mathcal{L}(G_\Omega)$. We will show this in Chap. 8.*

5.3 Homogeneous differential equation

The linear mapping Φ_ω is written in (4.6) with two arguments: $\Phi_\omega(f_h(\overline{\omega}), x_h(\partial\omega))$. Hence, there are two matrices $\Phi_\omega^\omega \in \mathbb{R}^{I(\gamma(\omega))\times I(\overline{\omega})}$, $\Phi_\omega^{\partial\omega} \in \mathbb{R}^{I(\gamma(\omega))\times I(\partial\omega)}$ with

$$\Phi_\omega(f_h(\overline{\omega}), x_h(\partial\omega)) = \Phi_\omega^\omega f_h(\overline{\omega}) + \Phi_\omega^{\partial\omega} x_h(\partial\omega),$$

which are to be determined in algorithm (5.1). Since, in general, $\#I(\omega) \gg \#I(\partial\omega)$, the matrix Φ_ω^ω has a much larger size than $\Phi_\omega^{\partial\omega}$.

A special case of the general boundary value problem, which is often of interest, is the homogeneous differential equation $Lu_\Omega = 0$ in Ω. Since $f_\Omega = 0$ implies also $f_h(\overline{\Omega}) = 0$ and $f_h(\overline{\omega}) = 0$ in all subdomains ω, the computation of Φ_ω^ω can be omitted. Analogously, the computation of Ψ_ω can be simplified: $\Psi_\omega : \mathbb{R}^{I(\overline{\omega})} \times \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}$ can be reduced to $\Psi_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}$.

6 Use of hierarchical matrices

As mentioned in Remark 5.1, the mappings Ψ_ω and Φ_ω correspond to dense matrices. Hence, the described method is not practicable unless the dimension is small. However, Ψ_ω and Φ_ω can be represented in the format of hierarchical matrices and all steps of the computation can be performed with the hierarchical matrix arithmetic.

An $\mathcal{H}$-matrix provides a data-sparse approximation of a dense matrix or sparse matrix by replacing blocks by matrices of low rank which can be stored efficiently. The low rank blocks are selected from a hierarchy of partitions organised in a so-called cluster tree that provide hierarchies of partitionings.

Definition 6.1 (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 successors of v as $S(v) := \{w \in V \mid (v, w) \in E\}$. Each vertex v is associated with a subset of I via a mapping $m_I : V \to \mathcal{P}(I)$, where $\mathcal{P}(I)$ denotes the power set of I. The tree $T(I)$ is called a cluster tree of I if the following conditions are satisfied:*

1. *The root r of $T(I)$ satisfies $m_I(r) = I$, and $m(v) \subset I$, $m(v) \neq \emptyset$, for all $v \in V$.*
2. *For all $v \in V$, either $S(v) = \emptyset$ or $m(v) = \dot{\bigcup}_{w\in S(v)} m(w)$.*

In the following, we identify V and $T(I)$, i.e., we write $v \in T(I)$ instead of $v \in V$. We also identify the nodes $v \in$ with their associated subsets $m(v)$ and call them cluster.

In previous papers (cf. [5, 6, 12, 4]) a geometric and a black box approach have been described for the construction of a cluster tree. The geometric approach requires geometric information that are associated with the indices. For finite element methods we usually use the support of the basis functions $\phi_i \in V_h$ of a finite dimensional test space V_h as geometric information. The black box approach needs a matrix graph which represents the connectivity of the indices. For example, for a finite element problem, this matrix graph can be constructed from the stiffness matrix [5].

The definition of a cluster tree allows an arbitrary number of successors for a cluster. The standard cluster strategies create cluster trees with only up to three successors. We use in this paper the domain-decomposition clustering from [5], where a cluster has no, one, two or three successors. The concept originates from a domain decomposition approach where an index set is divided into three separated subsets, like in our domain decomposition tree. In Sect. 6.1 we will see that the use of the domain decomposition clustering has benefits for our basis algorithm performed with $\mathcal{H}$-matrices.

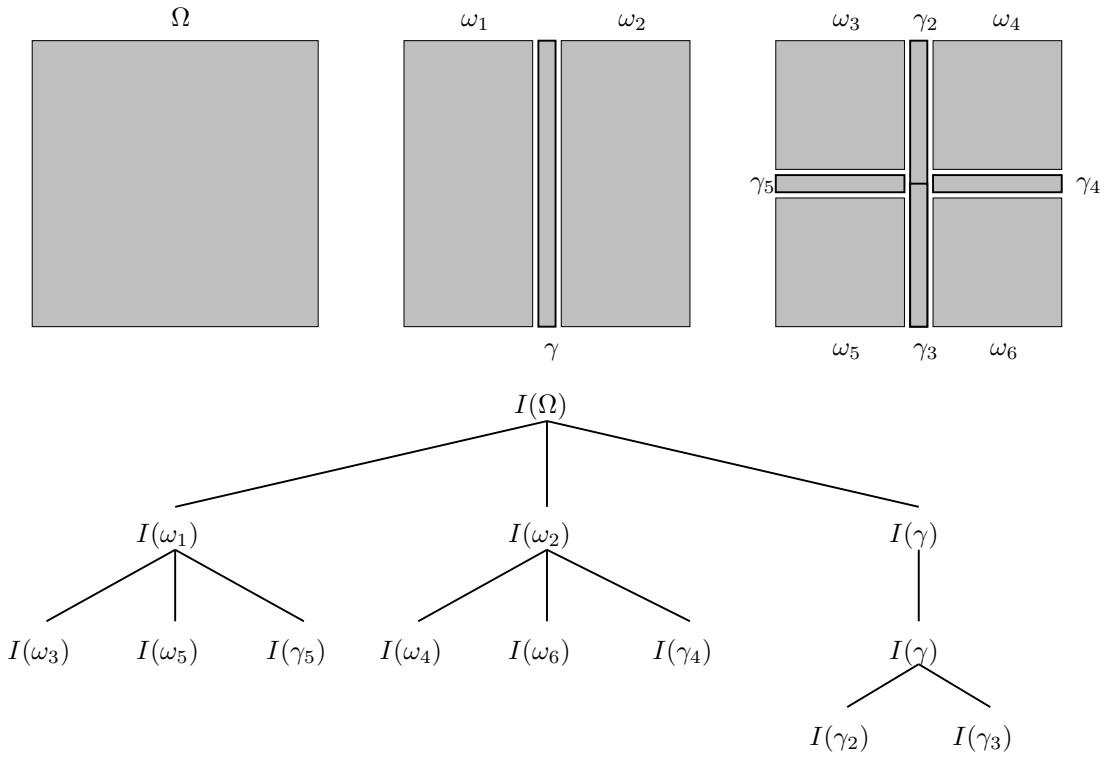

Figure 6.1: Example for domain decomposition clustering for the unit square $(0,1)^2$.

We give an overview about the domain decomposition clustering of [5] and we adapt this approach to our algorithm in Sect. 6.1.

The domain decomposition approach for the cluster tree construction is based on the decomposition of a subindex set $I(\omega)$, $\omega \subset \Omega$ open, into three parts. Thereby, the open subdomain ω is divided in the open subdomains ω_1, ω_2, and the separator γ such that $\omega = \omega_1 \dot\cup \omega_2 \dot\cup \gamma$ holds. With the separation of ω we obtain the decomposition of $I(\omega)$ in $I(\omega_1), I(\omega_2)$, and $I(\gamma)$. $I(\omega_1)$ and $I(\omega_2)$ are called domain clusters and are decomposed again in three parts. $I(\gamma)$ is decomposed via a bisection, such that both parts are of similar size. These interface clusters are not divided on every step. The clusters for open subdomains are usually bigger than the interface clusters. Hence, the subdivision is delayed every d-th step, where d is the dimension of the finite element problem, in order to avoid an imbalance of the cluster size on a level of the cluster tree ([5]). In Figure 6.1 we show an example for the domain decomposition clustering of the domain $\Omega = (0,1)^2$.

Following [6, 12], the construction of a cluster tree $T(I(\Omega))$ has the complexity (i.e., computational cost) of $\mathcal{O}\left(\#I \cdot depth(T(I(\Omega)))\right)$ and the depth of $T(I(\Omega))$ is bounded by $\mathcal{O}\left(\log \#I(\Omega)\right)$.

Cluster trees are used for the construction of block cluster trees which are defined next.

Definition 6.2 (block cluster tree) *Let $T(I)$, $T(J)$ be cluster trees for the index set I and J. A cluster tree $T(I \times J)$ is called a block cluster tree (based upon $T(I)$ and $T(J)$) if for all $v \in T(I \times J)$ exists $s \in T(I)$ and $t \in T(J)$ such that $v = s \times t$ and $level(s \times t) = level(s) = level(t)$. The nodes $v \in T(I \times J)$ are called block clusters.*

By the construction of a block cluster tree we stop the subdividing of a block if we can represent the matrix block with a low rank matrix. Thereby, we use an admissibility condition to decide which blocks can be replaced by a low rank matrix. The admissibility condition is a boolean function $Adm : \mathcal{P}(I) \times \mathcal{P}(J) \to \{true, false\}$ which is based on the underlying problem and the clustering strategy. Only clusters $s \times t$ with $Adm(s \times t) = false$ will be further subdivided.

So, block cluster trees $T(I \times J)$ fulfil the condition $Adm(father(v)) = false$ for all $v \in \mathcal{L}(T(I \times J))$. Furthermore, we demand $\#s \leq n_{min}$ or $\#t \leq n_{min}$ for $s \times t \in \mathcal{L}(T(I \times J))$, $s \times t$ not admissible, with

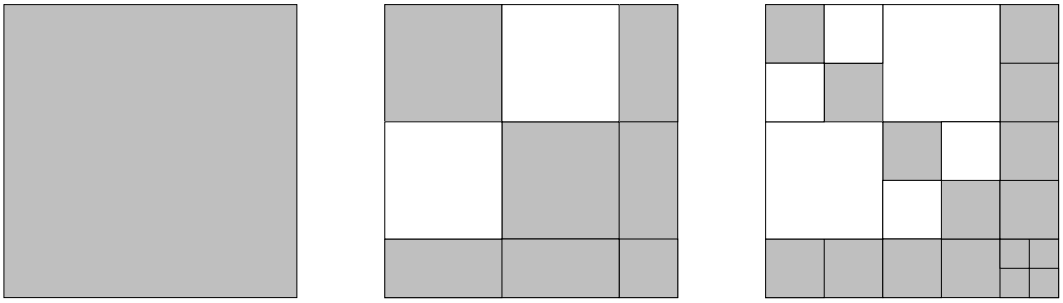

Figure 6.2: Block structure given by a block cluster tree constructed via the cluster tree from Figure 6.1

$n_{min} > 0$, to avoid very small blocks which we cannot handle efficiently. The leaves of a block cluster $T(I \times J)$ tree form a partition of the product index set $I \times J$.

In [6, 12, 4] the admissibility condition

$$Adm_{DD}(s \times t) = true \quad :\Leftrightarrow \begin{cases} (s \neq t, s, t \in \mathcal{C}_{dom}) \text{ or} \\ Adm_S(s \times t) = true \end{cases}$$

is used for finite element problems in combination with domain decomposition clustering to compute the LU-factorisation of a stiffness matrix. Thereby, $\mathcal{C}_{dom}$ is the set of domain clusters. $Adm_S(\cdot)$ is the standard admissibility condition. The standard admissibility condition uses a geometric distance and geometric diameter ([12]) or a graph distance and graph diameter for a black box version ([6]).

For example, the geometric standard admissibility condition for a block $s \times t \in T(I \times J)$ is

$$\min\{diameter(B_s), diameter(B_t)\} \leq \eta \cdot distance(B_s, B_t), \quad \eta > 0.$$

B_s, B_t are bounding boxes which contain the support of the corresponding basis functions of the clusters:

$$\bigcup_{i \in s} supp(\phi_i) \subset B_s \text{ and } \bigcup_{i \in t} supp(\phi_i) \subset B_t.$$

Figure 6.2 shows the block partition which we expect from the block cluster tree $T(I(\Omega) \times I(\Omega))$ constructed via Adm_{DD} and the domain decomposition cluster tree $T(I(\Omega))$ from Figure 6.1.

Remark 6.3 *The construction of a block cluster tree $T(I \times J)$ for a finite product index set $I \times J$, $\#I \geq \#J$, costs $\mathcal{O}(\#I \cdot depth(T(I \times J))$ ([6, 12, 4]). The depth of $T(I \times J)$ is limited by $\max\{depth(T(I)), depth(T(J))\}$.*

The admissible blocks are approximated by a low rank matrix in the following $\mathcal{R}_k$-matrix representation.

Definition 6.4 (Rk-Matrix representation) *Let $k \in \mathbb{N}$ and let I, J be two index sets. Let $M \in \mathbb{R}^{I \times J}$ be a matrix of at most rank k. A representation of M in factorised form*

$$M = AB^T, \quad A \in \mathbb{R}^{I \times \{1,\ldots,k\}}, \; B \in \mathbb{R}^{J \times \{1,\ldots,k\}}, \tag{6.1}$$

is called an $\mathcal{R}_k$-matrix representation of M, or, in short, we call M an $\mathcal{R}_k$-matrix.

Definition 6.5 ($\mathcal{H}$-matrix) *Let $k, n_{min} \in \mathbb{N}_0$. The set of $\mathcal{H}$-matrices induces by a block cluster tree $T(I \times J)$ with blockwise rank k and minimal block size n_{min} is defined as*

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

A matrix $M \in \mathcal{H}(T(I \times J), k)$ is said to be given in $\mathcal{H}$-matrix representation if for all leaves $s \times t$ with $\#s \leq n_{min}$ or $\#t \leq n_{min}$ the corresponding matrix block $M|_{s \times t}$ is given in full matrix representation and in $\mathcal{R}_k$-matrix representation for the other leaves.

Operation	Operands	Complexity
$y := y + Mx$	$y, x \in \mathbb{R}^I, M \in \mathcal{H}(T,k)$	$\mathcal{O}(k\#I \cdot depth(T))$
$M := M + M_1$	$M, M_1 \in \mathcal{H}(T,k)$	$\mathcal{O}(k^2\#I \cdot depth(T))$
$M := M + M_1 \cdot M_2$	$M, M_1, M_2 \in \mathcal{H}(T,k)$	$\mathcal{O}(k^2\#I \cdot depth(T)^2)$
M^{-1} (Inversion)	$M, M^{-1} \in \mathcal{H}(T,k)$	$\mathcal{O}(k^2\#I \cdot depth(T)^2)$
$M = LU$ (LU-decomposition)	$M, L, U \in \mathcal{H}(T,k)$	$\mathcal{O}(k^2\#I \cdot depth(T)^2)$
$M = LL^T$ (Cholesky-decomposition)	$M, L \in \mathcal{H}(T,k)$	$\mathcal{O}(k^2\#I \cdot depth(T)^2)$

Table 6.1: Complexity for the formated $\mathcal{H}$-matrix operations

Remark 6.6 (storage) *Let $T := T(I \times J)$ be a block cluster tree based on the cluster trees $T(I)$ and $T(J)$ and minimal block size n_{min}. Then the storage requirement for a matrix $M \in \mathcal{H}(T,k)$ is bounded by $\mathcal{O}(\max\{k, n_{min}\} depth(T) \cdot (\#I + \#J))$ ([4]).*

For the $\mathcal{H}$-matrices we can perform several operations in an almost linear complexity. These operations are the matrix-vector multiplication, matrix addition, matrix multiplication, matrix inversion, LU decomposition and Cholesky decomposition. All operations except the matrix-vector multiplication are performed approximately. This means the $\mathcal{R}_k$-matrices are truncated to the rank k or by the use of an adaptive arithmetic with respect to a given error bound ([12]). Therefore we call these operations formatted operations. The matrix-vector multiplication is performed exactly.

In [12, 6, 4] we find the definitions and estimates for the complexity. The complexity is shown for the special case where all matrices depend on the same block cluster tree $T(I \times I)$. In Table 6.1 we give an overview about the different complexity estimations. For Table 6.1 we assume that all $\mathcal{H}$-matrices depend on the block cluster tree $T := T(I \times I)$ with rank $k > 0$ for all $\mathcal{R}_k$-matrices and minimal leave size n_{min}.

6.1 Modifications

For our algorithm (cf. Chap. 5) we have to compute several matrices for every node of the domain decomposition tree G_Ω. For $\omega \in G_\Omega$, ω_1, ω_2 being the sons of ω, we compute (cf. Chap. 4)

$$\begin{aligned}
\Phi_\omega^{\overline{\omega}} &= \left(\Psi_{1\gamma}^{\gamma} + \Psi_{2\gamma}^{\gamma}\right)^{-1} \left(-\Psi_{1\overline{\omega}}^{\gamma} - \Psi_{1\overline{\omega}}^{\gamma}\right), \\
\Phi_\omega^{\partial\omega} &= \left(\Psi_{1\gamma}^{\gamma} + \Psi_{2\gamma}^{\gamma}\right)^{-1} \left(-\Psi_{1\partial\omega}^{\gamma} - \Psi_{2\partial\omega}^{\gamma}\right), \\
\Phi_\omega^{\overline{\omega}} &= \left(\Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)}\right) \Phi_\omega^{\overline{\omega}} + \Psi_{1\overline{\omega}}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\overline{\omega}}^{\partial\omega_2\setminus\gamma(\omega)}, \\
\Phi_\omega^{\partial\omega} &= \left(\Psi_{1\gamma}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\gamma}^{\partial\omega_2\setminus\gamma(\omega)}\right) \Phi_\omega^{\partial\omega} + \Psi_{1\partial\omega}^{\partial\omega_1\setminus\gamma(\omega)} + \Psi_{2\partial\omega}^{\partial\omega_2\setminus\gamma(\omega)}.
\end{aligned}$$

We can perform all the additions in these formulae as follows. Given are a target matrix $C \in \mathbb{R}^{I_3 \times J_3}$ and the two terms of the addition $A \in \mathbb{R}^{I_1 \times J_1}$ and $B \in \mathbb{R}^{I_2 \times J_2}$ with $I_1, I_2, I_3 \subset I$ and $J_1, J_2, J_3 \subset J$.

1. Calculate $\tilde{A} \in \mathbb{R}^{I_3 \times J_3}$ with $\tilde{A}_{ij} = \begin{cases} A_{ij} : & (i,j) \in I_1 \times J_1 \\ 0 : & otherwise. \end{cases}$
2. Calculate $\tilde{B} \in \mathbb{R}^{I_3 \times J_3}$ with $\tilde{B}_{ij} = \begin{cases} B_{ij} : & (i,j) \in I_2 \times J_2 \\ 0 : & otherwise. \end{cases}$
3. $C = \tilde{A} + \tilde{B}$.

???Because of the product form of an $\mathcal{R}_k$-matrix, the addition of an entry $(i,j) \in I_3 \times J_3$ is expensive because of additional conversions if it belongs in A to an $\mathcal{R}_k$-matrix and in B not to a $\mathcal{R}_k$-matrix. We have now the possibility to convert the hierarchical structure of A and B to the hierarchical structure of A or we choose a hierarchical structure of A and B always suitable for C.

We achieve this by constructing one cluster tree for the index set $I(\overline{\Omega})$ and restricting this cluster tree for all other occurring subindex sets. Due to the restriction we have to adjust the cluster tree $T(I(\overline{\Omega}))$ to

the structure of G_Ω to avoid too many extremely small blocks in the $\mathcal{H}$-matrices which occur because of the restriction to the cluster tree.

In the following, we first construct a cluster tree for the index set $I(\overline{\Omega})$. Secondly, we restrict this cluster tree to an index set $J \subset I(\overline{\Omega})$. Last, we explain the addition of $\mathcal{H}$-matrices with different product index sets. After these three extensions we can perform the basic algorithm via $\mathcal{H}$-matrices.

As mentioned above, we use the domain decomposition approach to construct a cluster tree for $I(\overline{\Omega})$. In [5, 6] a cluster tree is constructed for the index set $I(\Omega)$ and finite element problems. We adapt the algorithm from [5, 6] to our problem. In the following we assume that the domain decomposition tree $T(\Omega)$ is given.

First, we split $I(\overline{\Omega})$ into the subsets $I(\Omega)$ and $I(\partial\Omega)$. We associate $I(\Omega)$ and $I(\partial\Omega)$ to the root of $T(\Omega)$. $I(\omega)$ is a domain cluster and $I(\partial\Omega)$ is a boundary cluster. For the construction of the cluster tree we have to consider the following cases:

- Domain Cluster: If a domain cluster is associated to an $\omega \in T(\Omega)$ with sons ω_1, ω_2, we split the domain cluster into the three parts $I(\omega_1)$, $I(\omega_2)$, and $I(\gamma(\omega))$. Thereby, $I(\omega_1)$, $I(\omega_2)$ are new domain clusters and $I(\omega_1)$ is associated with ω_1 and $I(\omega_2)$ with ω_2. $I(\gamma(\omega))$ is associated with ω_1 and is a boundary cluster. If the domain cluster is not associated with a $\omega \in T(\Omega)$, we split the cluster via the domain decomposition approach again into three parts. In this case the interface cluster belongs to our boundary clusters.

- Boundary Cluster: We split a boundary cluster v only if $\#v > avg_{\text{dom}}/2$ holds. avg_{dom} is the average size of all domain clusters on the same level like v. If a boundary cluster v is associated to $\omega \in T(\Omega)$ with sons ω_1, ω_2, we split the domain cluster into the parts $v_1 = v \setminus I(\omega_2)$ and $v_2 \setminus v_1$. v_1 is associated with ω_1 and v_2 is associated with ω_2. If a boundary cluster is not associated to an $\omega \in T(\Omega)$, we use the bisection strategy of the domain decomposition cluster strategy.

Note that we have to subdivide clusters v only if $\#v \geq n_{min}$.

For our basis algorithm we have to construct many matrices. The block index sets of these matrices depend on index sets of the form $I(\overline{\omega}), I(\gamma), I(\partial\omega)$ for $\omega \in T(\Omega)$. With our cluster tree construction we can find one cluster $v \in T(I)$ with $v = I(\omega)$. Additionally, we need only few clusters to put together $I(\partial\omega)$ and γ. This is necessary to avoid to many small clusters in the restricted cluster trees.

After constructing the cluster tree $T(I(\overline{\Omega}))$, we have to restrict it to subsets of $I(\overline{\Omega})$.

Definition 6.7 (restricted cluster tree) *Let $T(I) := (V, E)$ be the cluster tree for the index set I with the mapping m. Let $K \subset I$. The restricted cluster tree $T(I \times J)|_{K\times L} = (V_K, E_K)$ is defined by*

$$V_K := \{v \in V \mid m(v) \cap K \neq \emptyset\} \quad E_K := \{(v, w) \in E \mid v, w \in V_K\} \tag{6.2}$$

with the mapping $m_k : V_K \to \mathcal{P}(K)$ satisfying

$$m_K(v) = m(v) \cap K \quad \text{for all } v \in V_K. \tag{6.3}$$

Remark 6.8 *$\mathcal{L}(T(I)|_K) \subset \mathcal{L}(T(I))$ and $depth(T(I)|_K) \leq depth(T(I))$ holds for all $K \subset I$.*

Remark 6.9 *We can construct the restricted cluster tree $T(I)|_K$ with a complexity of $\mathcal{O}(\#K \cdot depth(T(I)))$ ([3]). The algorithm uses that we sort the index sets in the same way and use the special storage structure ([12]) of the cluster tree.*

With $T(I)$ and the admissibility condition Adm_{DD} we construct the block cluster tree $T(I \times I)$. Now we restrict this block cluster tree.

Definition 6.10 (restricted block cluster tree) *Let $T(I \times J) = (V, E)$ be a minimal block cluster tree to admissibility condition Adm upon the cluster trees $T(I) = (V_I, E_I)$ and $T(J) = (V_J, E_J)$. Let $K \times L \subset I \times J$ and $T(I)|_K = (V_K, E_K)$, $T(J)|_L = (V_L, E_L)$ be restricted cluster trees. The restricted block cluster tree $T(I \times J)|_{K\times L} = (V_1, E_2)$ is defined by $V_1 := \{(v, w) \in V \mid v \in V_K, w \in V_L\}$ and $E_2 := \{(v, w) \in E \mid v, w \in V_1\}$.*

For a restricted block cluster tree $T(I \times J)|K \times L$ we define that all $v \in T(I \times J)|_{K\times L}$ are admissible if and only if $v \in T(I \times J)$ is admissible.

Remark 6.11 *The complexity of the restriction of a block cluster tree $T(I \times I)$ depends on the number of nodes of this block cluster tree. Following [12], the number of nodes is bounded by $\mathcal{O}(\#I \cdot depth(T(I \times I))$. In [3] it is shown that the complexity of the block cluster tree restriction $T(I \times I)|_{K \times L}$ is $\mathcal{O}(\max\{\#K, \#L\} depth(T(I \times I)))$.*

The additions to compute Φ_ω, $\omega \in G_\Omega$ have the form $C = A + B$ with $A \in \mathcal{H}(T(I \times I)|_{I_1 \times J_1}, k)$, $B \in \mathcal{H}(T(I \times I)|_{I_2 \times J_2}, k)$, $C \in \mathcal{H}(T(I \times I)|_{I_3 \times J_3}, k)$ with $I_1, I_2, I_3, J_1, J_2, J_3 \subset I$. In this case we can perform the addition blockwise and obtain the complexity $\mathcal{O}(k^2 \max\{\#I_3, \#J_3\} \cdot depth(T(I \times I))$ like for the formatted addition ([3]).

We assume for the following bounds a fixed arithmetic with rank k and $I(\overline{\Omega}) = I$.

Remark 6.12 *If $\omega \in \mathcal{L}(G_\omega)$ with $I(\omega) \neq \emptyset$, we have to solve the system (4.7). For that purpose we store the Cholesky decomposition of $A^{\omega,\omega}$ and the sparse matrices M_h^ω and $A^{\omega,\partial\omega}$. The computational cost of these matrices is bounded by $\mathcal{O}(k^2 \#I(\overline{\omega}) \cdot depth(T(I \times I))^2)$ and the storage complexity is bounded by $\mathcal{O}(k \#I(\overline{\omega}) \cdot depth(T(I \times I)))$.*

For $\omega \in G_\omega$, ω being not a leaf, we have to compute Φ_ω, Ψ_ω and to store Φ_ω. The computation complexity is dominated by multiplications (multiplications by matrices with the index set $I(\overline{\omega})$) such that it is bounded by $\mathcal{O}(k^2 \#I(\overline{\omega}) \cdot depth(T(I \times I))^2)$. The storage complexity is bounded by $\mathcal{O}(k \#I(\overline{\omega}) \cdot depth(T(I \times I)))$.

Remark 6.13 *The total storage requirement and the complexity to compute a solution is proportional to*

$$k \sum_{\omega \in G_\Omega} \#I(\overline{\omega}) \log(\#I(\overline{\omega}) \leq k \log(\#I(\overline{\Omega})) \sum_{\omega \in G_\Omega} \#I(\overline{\omega}) \leq k \#I(\overline{\Omega}) L_{G_\Omega} \log(\#I(\overline{\Omega})),$$

where L_{G_Ω} is the depth of the tree: $G_\Omega = \bigcup_{0 \leq \ell \leq L_{G_\Omega}} G_\Omega^{(\ell)}$. The total computation requirement is proportional to

$$k^2 \sum_{\omega \in G_\Omega} \#I(\overline{\omega}) \log^2(\#I(\overline{\omega}) \leq k^2 \log^2(\#I(\overline{\Omega})) \sum_{\omega \in G_\Omega} \#I(\overline{\omega}) \leq k^2 \#I(\overline{\Omega}) L_{G_\Omega} \log^2(\#I(\overline{\Omega})).$$

The (possibly large) factor $\#I(\overline{\Omega})$ holds for the matrix blocks $\Psi_\omega^\omega, \Phi_\omega^\omega : \mathbb{R}^{I(\overline{\omega}) \times I(\partial\omega)}$ (summed over all $\omega \in G_\Omega^{(\ell)}$). The matrices $\Psi_\omega^{\partial\omega}, \Phi_\omega^{\partial\omega} \in \mathbb{R}^{I(\gamma(\omega)) \times I(\partial\omega)}$ are much smaller. Since the manifolds $\gamma(\omega)$ and $\partial\omega$ have one dimension less, one expects $\#\gamma(\omega) \sim \#\partial\omega \sim (\#\omega)^{(d-1)/d}$. However, as the union of all $\partial\omega$ for $\omega \in G_\Omega^{\text{coarse}}$ covers the whole grid, the storage is at least proportional to $\#I(\overline{\Omega})$.

7 Homogenisational aspects

Now we assume that the differential equation contains coefficients with small-scale behaviour. Let for instance the bilinear form $a_\omega(u_h, v_h)$ be

$$a_\Omega(u_h, v_h) = \int_\Omega \langle A(x) \operatorname{grad} u_h, \operatorname{grad} v_h \rangle \, \mathrm{d}x \tag{7.1}$$

($A(x) \in \mathbb{R}^{d \times d}$, $\langle \cdot, \cdot \rangle$ $\mathbb{R}^d$-scalar product, d: spatial dimension, i.e., $\Omega \subset \mathbb{R}^d$), where the matrix-valued function $A(x)$ is highly oscillatory, jumping or non-smooth in another way. A further possibility is that (independently of the behaviour of $A(x)$) the domain Ω is complicated, e.g., it may contain many holes of different sizes. In order to discretise such problems by a finite element method, one must use a fine grid resolution. In the case of highly oscillating coefficients the grid size should small enough so that the variation in one element is sufficiently small. In the case of a complicated geometry one needs a fine triangulation to approximate the boundaries of the domain by, e.g., isoparametric[7] elements.

Even if the discretisation requires a fine grid, one is not necessarily interested to represent the finite element solution u_h by the same fine mesh. Oscillations or jumps of the coefficients lead to oscillations or kinks in the solution u_h, but often one is only interested in the mean behaviour, not in the details. In the case of periodically oscillating coefficients $A(x/\varepsilon)$ of frequency $1/\varepsilon$, so-called homogenisation methods are well-known which lead to a "homogenised" bilinear form. Since its coefficients are smoother, the homogenised

[7] For isoparametric finite elements compare [9, §8.5.3].

problem can be solved with essentially coarser step size than the original problem. The computation of the homogenised coefficients requires to solve the original problem in one periodicity cell. This can be achieved only by a numerical solution so that also here we have to assume that the periodic problem can be resolved by a sufficiently small grid size $h \ll 1/\varepsilon$.

7.1 Basic method

The standard homogenisation method is not applicable in irregular situations. Instead we will exploit the *partial* evaluation of the inverse. For this purpose we divide the domain decomposition tree G_Ω in a coarse part $G_\Omega^{\rm coarse}$ and a fine part $G_\Omega^{\rm fine}$:

$$G_\Omega = G_\Omega^{\rm coarse} \cup G_\Omega^{\rm fine} \qquad \text{(disjoint union)},$$

where $G_\Omega^{\rm coarse} \neq \emptyset$ is a subtree of $G_\Omega^{\rm fine}$ with the same root Ω, while $G_\Omega^{\rm fine} = G_\Omega \backslash G_\Omega^{\rm coarse}$ represents the remaining set ($G_\Omega^{\rm fine}$ is not a tree).

For a given step size $H \in (0, \operatorname{diam}(\Omega)]$, a possible criterion for determining the coarse part may look as follows:

$$G_\Omega^{\rm coarse} = G_{\Omega,H}^{\rm coarse} := \{\omega \in G_\Omega^{\rm coarse} : \operatorname{diam}(\omega) \geq H\}. \tag{7.2}$$

The basic method reads:

1. Definition phase as in §5.1: application of (5.1).
2. The evaluation phase is restricted to $G = G_\Omega^{\rm coarse}$. The procedure from (5.3) gets a further parameter G (subtree of G_Ω with same root[8]). The first parameter ω of the following procedure must belong to $G\backslash\mathcal{L}(G)$:

```
procedure partial_evaluation(ω, G, f, x);
{input: boundary values x|_{I(∂ω)} and right-hand side f|_{I(ω)}}
begin trace(ω, f, x);
    for ω' ∈ S_G(ω) do partial_evaluation(ω', f, x)
end;
```
(7.3)

   Since the son set $S_G(\omega)$ of G is used, the recursion terminates at the leaves of G.

The computational cost of the definition phase is still the same, but since the evaluation phase can be called several times for different f, x, the reduction of the cost of *partial_evaluation* is important. In particular, the reduction of the storage requirement is essential.

Remark 7.1 *a) For the partial evaluation in the subtree G only the corresponding mappings*

$$\Phi_\omega \qquad \text{for } \omega \in G\backslash\mathcal{L}(G)$$

need to be stored.

b) In spite of the reduced data, one obtains unchanged results at all nodal points of

$$\partial G := \bigcup\nolimits_{\omega\in G} \partial\omega.$$

c) Using the definition (7.2) for $G_{\Omega,H}^{\rm coarse}$, one obtains a grid $\partial G_{\Omega,H}^{\rm coarse}$ of step size[9] $\leq H$.

The storage requirements and the complexity for the computation of a solution is reduced by the lower depth of the domain decomposition tree ($L_{G_\Omega}^{\rm coarse} := \mathit{depth}(G_\Omega^{\rm coarse})$):

$$k\#I(\overline{\Omega})L_{G_\Omega}^{\rm coarse}\log(\#I(\overline{\Omega}))$$

[8] *complete_evaluation*(ω, r, x) is equivalent to *partial_evaluation*(ω, G_Ω, r, x).

[9] For all $x \in \Omega$ the closed ball centred at x with radius $H/2$ contains at least one point of $\partial G_{\Omega,H}^{\rm coarse}$.

7.2 Coarsening of the ansatz space for the right-hand side

The fine step size h is caused by the small-scale behaviour of $A(x)$ in (7.1) or by the complicated geometry. In general, however, the right-hand side f_h is expected to be approximable by a coarser step size $H \gg h$. In the following we distinguish the grids Ω_h and Ω_H and we only create a domain decomposition tree $G_\Omega^{\mathrm{coarse}}$ for the coarse grid Ω_H.

By $J(\overline{\Omega})$ we denote the index set for the coarse grid Ω_H and by $I(\overline{\Omega})$ the index set for the fine grid Ω_h. We assume that the fine grid is a refined grid of the coarse grid and that $J(\overline{\Omega}) \subset I(\overline{\Omega})$. $f_H \in \mathbb{R}^{J(\overline{\Omega})}$ is the $L^2(\overline{\Omega})$-orthogonal projection of f.

Let $P^\omega_{h\leftarrow H} \in \mathbb{R}^a$ be a prolongation matrix for $\omega \in \mathcal{L}(G_\Omega^{\mathrm{coarse}})$. It holds $f_h(\overline{\omega}) = P^\omega_{h\leftarrow H} f_H(\overline{\omega})$ with $f_h(\overline{\omega}) \in \mathbb{R}^{I(\overline{\omega})}$ and $f_H(\overline{\omega}) \in \mathbb{R}^{J(\overline{\omega})}$.

For $\omega \in \mathcal{L}(G_\Omega^{\mathrm{coarse}})$ we have to solve the system

$$x_h|_{I(\omega)} = (A^{\omega,\omega})^{-1} \left(M_h^\omega P^\omega_{h\leftarrow H} f_H(\overline{\omega}) - A^{\omega,\partial\omega} g_h(\partial\omega) \right). \tag{7.4}$$

For $\omega \in \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, the auxiliary matrices are obtained by (compare Sect. 4.6)

$$\Psi_\omega = \left(\Psi_\omega^{\overline{\omega}}, \Psi_\omega^{\partial\omega}\right) \qquad \text{with} \quad \left\{ \begin{array}{l} \Psi_\omega^{\overline{\omega}} := -M_h^{\partial\omega} P^\omega_{h\leftarrow H} + A^{\mathrm{nat},\partial\omega,\omega} \left(A^{\omega,\omega}\right)^{-1} M_h(\omega) P^\omega_{h\leftarrow H}, \\ \Psi_\omega^{\partial\omega} := A^{\mathrm{nat},\partial\omega,\partial\omega} - A^{\mathrm{nat},\partial\omega,\omega} \left(A^{\omega,\omega}\right)^{-1} A^{\mathrm{nat},\omega,\partial\omega}. \end{array} \right. \tag{7.5}$$

For $\omega \in G_\Omega^{\mathrm{coarse}}$, $\omega \notin \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, the recursion formulae of $\Psi_\omega = (\Psi_\omega^{\overline{\omega}}, \Psi_\omega^{\partial\omega})$ and $\Phi_\omega = (\Phi_\omega^{\overline{\omega}}, \Phi_\omega^{\partial\omega})$ still hold. Only the size changes for $\Psi_\omega^{\overline{\omega}} \in \mathbb{R}^{I(\partial\omega)\times J(\overline{\omega})}$ and $\Phi_\omega^{\overline{\omega}} \in \mathbb{R}^{I(\gamma(\omega))\times J(\overline{\omega})}$.

For $\omega \in \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, the computational complexity is $\mathcal{O}(k^2 \#I(\overline{\omega}) \cdot \mathit{depth}(T(I\times I))^2)$ and the storage complexity is bounded by $\mathcal{O}(k\#I(\overline{\omega}) \cdot \mathit{depth}(T(I\times I)))$ (cf. Remark 6.12).

For $\omega \notin \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, we could reduce the complexity. We assume $c \cdot \#J(\overline{\omega}) > \#I(\partial\omega)$ for all $\omega \notin \mathcal{L}(G_\Omega^{\mathrm{coarse}}$. Like in Remark 6.12, the cost is $\mathcal{O}(k^2\#I(\overline{\omega}) \cdot \mathit{depth}(T(I\times I))^2)$, while the storage is bounded by $\mathcal{O}(k\#I(\overline{\omega}) \cdot \mathit{depth}(T(I\times I)))$.

The whole complexity for the computation is

$$k^2(\#I(\overline{\Omega}) + \#J(\overline{\Omega})(L_{G_\Omega}^{\mathrm{coarse}} - 1)) \log^2(\#I(\overline{\Omega})) \tag{7.6}$$

(cf. 6.13). The complexity for storage and computation of a solution is

$$k(\#I(\overline{\Omega}) + \#J(\overline{\Omega})(L_{G_\Omega}^{\mathrm{coarse}} - 1)) \log(\#I(\overline{\Omega})). \tag{7.7}$$

If we are only interested in the solution for the coarse grid, the complexity for storage and computation of a solution is reduced to

$$k\#J(\overline{\Omega})(L_{G_\Omega}^{\mathrm{coarse}}) \log(\#I(\overline{\Omega})). \tag{7.8}$$

7.3 Computation of functionals

According to §7.1 the partial evaluation yields the original values at the finite element nodes of $\partial G_\Omega^{\mathrm{coarse}} = \bigcup_{\omega\in G_\Omega^{\mathrm{coarse}}} \partial\omega$ (up to rounding errors due to the hierarchical matrix arithmetic). It may be more reasonable to ask for means values of a neighbourhood of the node. Such mean values are examples of a linear functional

$$J(u_h) = \sum_{\alpha\in I_J} J_\alpha x_{h,\alpha} \tag{7.9}$$

($x_{h,\alpha}$ are the coefficients of u_h, cf. (4.2)). The support of J is denoted by $I_J \subset I(\overline{\Omega})$ (i.e., $J_\alpha \neq 0$ for $\alpha \in I_J$).

Restricting the sum $\sum_{\alpha\in I_J}$ to the subsets $I(\overline{\omega})$ with $\omega \in G_\Omega^{\mathrm{coarse}}$, one obtains functionals $J_\omega(u_h)$. In order to ensure additivity,

$$J_\omega(u_h) = \sum_{\omega'\in S_{G_\Omega^{\mathrm{coarse}}}(\omega)} J_{\omega'}(u_h). \tag{7.10}$$

over the son-subdomains, one must take care for the correct definition at overlapping boundary nodes. A possible recursive definition of its coefficients is: $J_{\alpha,\Omega} := J_\alpha$ for the root $\Omega \in G_\Omega^{\mathrm{coarse}}$. For $\omega \in G_\Omega^{\mathrm{coarse}}$ let $\{\omega_1,\omega_2\} = S_{G_\Omega^{\mathrm{coarse}}}(\omega)$ be the son set. Then the definition

$$J_{\alpha,\omega_1} := \left\{ \begin{array}{ll} J_{\alpha,\omega} & \text{for } \alpha \in I(\overline{\omega_1}) \\ 0 & \text{otherwise} \end{array} \right\}, \qquad J_{\alpha,\omega_2} := \left\{ \begin{array}{ll} J_{\alpha,\omega} & \text{for } \alpha \in I(\overline{\omega_2})\backslash I(\overline{\omega_1}) \\ 0 & \text{otherwise} \end{array} \right\}$$

leads to property (7.10).

In the next step, we represent J_ω as a function of $f_h(\overline{\omega})$ and $g_h(\partial\omega)$:

$$\mathcal{J}_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) := J_\omega\left(u_h(f_h(\overline{\omega}), g_h(\partial\omega))\right).$$

The concrete problem is to determine the corresponding matrices $J_\omega^\omega, J_\omega^{\partial\omega}$ with

$$\mathcal{J}_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) = J_\omega^\omega f_h(\overline{\omega}) + J_\omega^{\partial\omega} g_h(\partial\omega).$$

This will be performed during the definition phase from §5.1 in recursion (5.1) from the leaves to the root. For $\omega \in \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, the vector $r_h(\overline{\omega})$ of the corresponding solution $x_h(\overline{\omega})$ and thus J_ω can be determined directly. Now let $\{\omega_1, \omega_2\} = S_{G_\Omega^{\mathrm{coarse}}}(\omega)$ be the son set of $\omega \in G_\Omega^{\mathrm{coarse}} \backslash \mathcal{L}(G_\Omega^{\mathrm{coarse}})$, and assume that $\mathcal{J}_{\omega_1}, \mathcal{J}_{\omega_2}$ are known. The arguments $f_h(\overline{\omega_i})$ and $g_h(\partial\omega_i)$ of $\mathcal{J}_{\omega_i}$ $(i = 1, 2)$ can be split into $(f_h(\overline{\omega_i}), g_h(\partial\omega_i \backslash \gamma(\omega)), g_h(\gamma(\omega)))$:

$$\mathcal{J}_{\omega_i}(f_h(\overline{\omega_i}), g_h(\partial\omega)) = \mathcal{J}_{\omega_i}\left(r_h(\overline{\omega_i}), g_h(\partial\omega_i \backslash \gamma(\omega)), g_h(\gamma(\omega))\right).$$

The data $g_h(\gamma(\omega))$ are the boundary values $x_h|_{\gamma(\omega)}$ which result from $(f_h(\overline{\omega}), g_h(\partial\omega))$ via Φ_ω. Together with the additivity (7.10) we obtain the equation

$$\mathcal{J}_\omega(f_h(\overline{\omega}), g_h(\partial\omega)) = \sum_{i=1,2} \mathcal{J}_{\omega_i}\left(f_h(\overline{\omega_i}), g_h(\partial\omega_i \backslash \gamma(\omega)), \Phi_\omega(g_h(\overline{\omega}))\right)$$

determining the matrices $J_\omega^\omega, J_\omega^{\partial\omega}$.

Remark 7.2 *a) In the practical realisation, $J_\omega^\omega, J_\omega^{\partial\omega}$ are to be represented in the hierarchical matrix format.*

b) As soon as $\mathcal{J}_\omega$ is computed, the data of $\mathcal{J}_{\omega_i}$, $\omega_i \in S_{G_\Omega^{\mathrm{coarse}}}(\omega)$, can be deleted.

c) As soon as $\omega \in G_\Omega^{\mathrm{coarse}}$ contains the support of J, i.e., $I_J \subset I(\overline{\omega})$, the recursion can be terminated[10]*. After the determination of $x_h(\partial\omega)$ the data $(f_h(\overline{\omega}), g_h(\partial\omega) = (f_h(\overline{\omega}), g_h(\partial\omega))$ are known because of $g_h(\partial\omega) = x_h(\partial\omega)$ and can be used for the evaluation of $\mathcal{J}_\omega(f_h(\overline{\omega}), g_h(\partial\omega))$.*

8 Numerics

In this section we show that our algorithm, mentioned as DD algorithm, in combination with the $\mathcal{H}$-matrix technique works like expected. First, we use the Laplace problem to show the correct behaviour of our algorithm. Second, we present an application of our algorithm for a multiscale problem. For all experiments we choose $n_{min} = 32$.

8.1 Laplace Problem

The Laplace problem with Dirichlet boundary conditions is defined by

$$L = -\Delta, \tag{8.1}$$
$$Lu = f \quad \text{in } \Omega, \tag{8.2}$$
$$u = g \quad \text{on } \partial\Omega. \tag{8.3}$$

We use the 2D and 3D Laplace problem to present the correct behaviour of the DD algorithm. As domain we use the unit square $(0,1)^d$, $d \in \{2,3\}$, and discretise this domain only with square and cube elements whereby h denotes the length of the edges of these elements. As error measurement we use the $L^2(\Omega)$ error $\|u - u_{mh}\|_{L^2} := \|u - u_{mh}\|_{L^2(\Omega)}$. Here, u is the exact solution of the Laplace problem and u_{mh}, $m \in \mathbb{N}$, the solution computed by our DD algorithm with the right-hand side vector $f_{mh}(\overline{\Omega})$. For example u_h is the solution of the DD algorithm which uses only one scale.

First, we show the effects of the fixed rank arithmetic with increasing rank and the effects of the adaptive rank arithmetic with decreasing error bound in Table 8.1. We use the 2D Laplace problem to show the effects of increasing k for the fixed rank arithmetic in the left part of the Table 8.1. For these calculation the

[10] When we proceed to larger subdomains or even to Ω, the storage requirement may increase.

k	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$	ε	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$
2	0.228	0.228	0.228258	10e-01	8.018e-2	8.142e-2	8.254e-2
4	4-236e-2	4.238e-2	4.24e-2	10e-02	3.707e-3	4.173e-3	5.342e-3
8	5.538e-5	5.618e-5	5.87403e-05	10e-03	3.662e-4	5.818e-4	1.646e-3
12	2.759e-6	2.571e-6	3.44661e-06	10e-04	3.305e-4	3.558e-4	1.318e-3
16	2.830e-6	2.527e-6	3.01929e-06	10e-05	3.381e-4	3.474e-4	1.299e-3
20	2.837e-6	2.530e-6	3.00364e-06	10e-06	3.388e-4	3.469e-4	1.297e-3

Table 8.1: The left part of the table shows the convergence behaviour for fixed rank arithmetic with increasing k and the 2D Laplace problem with $h = \frac{1}{512}$. The right part of the table shows the convergence behaviour for adaptive rank arithmetic for the 3D Laplace problem with $h = \frac{1}{64}$.

u	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$	$\|u-u_{8h}\|_{L^2}$	$\|u-u_{16h}\|_{L^2}$
x^2-y^2	1.608e-6	1.608e-6	1.608e-6	1.608e-6	1.608e-6
x^3-y^3	4.330e-6	4.330e-6	4.331e-6	4.330e-6	4.331e-6
x^4-y^4	8.034e-6	7.927e-6	9.364e-6	2.635e-5	1.061e-4
x^5-y^5	1.135e-5	1.012e-5	1.201e-5	5.172e-5	2.254e-4
$\sin(x)\cos(y)$	1.221e-6	1.435e-6	2.377e-6	6.419e-6	2.285e-5

Table 8.2: The $L^2(\Omega)$-Error for the two dimensional Laplace problem with step size $h = \frac{1}{256}$ and adaptive arithmetic with an error bound of $1e-16$.

solution is $u(x,y) = x^5 - y^5$. The results shows, that the errors are decreasing for increasing k. Also we have the effects that the error is bigger if we use coarse ansatz spaces for the right-hand side. This results from the additional error of the approximation of the right-hand side by the prolongation matrix. The right-hand side of Table 8.1 shows the effects of decreasing error bounds for the adaptive arithmetic of the $\mathcal{H}$-matrices. In this case we solve the 3D Laplace problem for $u(x,y,z) = x^5 - y^5$. Like for the fixed rank arithmetic we observe decreasing errors for decreasing error bounds. Also we have slightly growing errors for coarse ansatz spaces of the right-hand side. Additionally, we see convergence??? to the discretisation error for the fixed rank arithmetic and the adaptive arithmetic, because for the last rows of Table 8.1 the errors are not decreasing any more.

All following tests are performed with the adaptive arithmetic. If we want to show properties of the DD algorithm without the additional errors of the non-exact arithmetic of the $\mathcal{H}$ -matrices we use the error bound 10^{-16}. With this error bound the arithmetic operations are performed almost exact. Otherwise, we use the error bound 10^{-4}.

We show the influence for using different coarse ansatz spaces for the right-hand side and different functions u. We use functions u such that f is a polynomial of degree 0 to 3. Additionally, we compute the solution for a function $u(x,y) = \sin(x)\cos(y)$. In Table 8.2 we present the $L^2(\Omega)$ errors, where we use an almost exact arithmetic. For the results in Table 8.2 we use the step size $h = \frac{1}{256}$. u_h have the smallest errors because of the not coarsened ansatz space for the right-hand side. For constant and linear f we also have the same error sizes for coarse ansatz spaces for the right-hand side. In this cases we avoid errors for the approximation of f by the prolongation: $f_h(\Omega) = P^{\Omega}_{h \leftarrow mh} f_{mh}(\Omega)$. If f is not constant or linear, the errors increase for coarser ansatz spaces for the right-hand side.

Next, we show convergence of our method for decreasing h. In this case we also use an almost exact error bound. Table 8.3 shows convergence for decreasing h for the 2D Laplace problem with $u = x^5 - y^5$. Again we see the convergence behaviour of the DD algorithm in presence of small increasing errors for coarser ansatz spaces for the representation of f.

Table 8.4 shows convergence for decreasing h for the 3D Laplace problem. The setup of these experiments is the same like for Table 8.3 and the interpretation of the results is identical. The exact calculation of u_h on the two finest meshes is omitted since it is too expensive.

???Up to now we present the convergence behaviour of the DD algorithm if we use almost exact accuracy

h	$\#I(\overline{\Omega})$	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$	$\|u-u_{8h}\|_{L^2}$
$\frac{1}{64}$	4225	1.815e-04	1.618e-04	6.213e-04	8.357e-04
$\frac{1}{128}$	16641	4.539e-05	4.047e-05	4.805e-05	2.072e-04
$\frac{1}{256}$	66049	1.134e-05	1.012e-05	1.201e-05	5.172e-05
$\frac{1}{512}$	263169	2.837e-06	2.530e-06	3.003e-06	1.292e-05
$\frac{1}{1024}$	1050625	7.093e-07	6.325e-07	7.509e-07	3.230e-06

Table 8.3: Convergence for decreasing step size of the 2D Laplace problem with an adaptive arithmetic accuracy of $\varepsilon = 10^{-16}$ and analytic solution $u(x, y) = x^5 - y^5$.

h	$\#I(\overline{\Omega})$	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$
$\frac{1}{20}$	9261	3.468e-3	3.55e-3	1.334e-2
$\frac{1}{40}$	68921	8.674e-4	8.879e-4	3.324e-3
$\frac{1}{60}$	226981	3.855e-4	3.946e-4	1.476e-3
$\frac{1}{80}$	531441	not calc.	2.22e-4	8.304e-4
$\frac{1}{100}$	1030301	not calc.	1.421e-4	5.314e-4

Table 8.4: Convergence for decreasing step size of the 3D Laplace problem with an adaptive arithmetic accuracy of $\varepsilon = 10^{-16}$ and analytic solution $u(x, y, z) = x^5 - y^5$.

and show the influence of the coarse ansatz space for f. Therefore we observe convergence for decreasing h. In Table 8.5 we calculate the same problems like in Table 8.2, but with the error bound $\varepsilon = 10^{-4}$ for the adaptive arithmetic. As result we obtain errors of the order ε.

The last experiments for the Laplace problem show the error, run time and storage behaviour with an adaptive error bound 10^{-4}. We expect a logarithmic behaviour for the run time and storage consumption. Table 8.6 shows the results for the 2D Laplace problem and Table 8.7 for the 3D Laplace problem. In both tables we present results for our DD algorithm using one or two scales. Therefore we specify the error, the time for the setup, which includes the construction of the DD tree and the calculation of the solution matrices, the needed memory to save all necessary data and the average time to solve a boundary problem. One advantage of our solver is the application to different boundary value problems with identical differential operator. Also almost exact arithmetic is used in Tables 8.3 and 8.4.Comparing the results of these tables, we see that the errors are now bounded by the error bound of the adaptive arithmetic. The time for the setup, the memory consumption and the solving time shows a logarithmic behaviour. For the 3D case the run time and storage consumption is much greater because of the greater boundary compared with the inner nodes of the grid. Additionally, the admissible condition for 3D problems creates partitions with less and smaller admissible blocks ([5, 6]).

u	$\|u-u_h\|_{L^2}$	$\|u-u_{2h}\|_{L^2}$	$\|u-u_{4h}\|_{L^2}$	$\|u-u_{8h}\|_{L^2}$	$\|u-u_{16h}\|_{L^2}$
$x^2 - y^2$	8.807e-5	8.807e-5	8.807e-5	4.745e-5	9.892e-4
$x^3 - y^3$	6.161e-5	6.164e-5	6.170e-5	3.226e-5	7.077e-4
$x^4 - y^4$	3.980e-5	4.072e-5	4.443e-5	4.268e-5	5.538e-4
$x^5 - y^5$	2.053e-5	2.296e-5	3.312e-5	6.588e-5	5.087e-4
$sin(x)cos(y)$	1.221e-4	1.224e-4	1.234e-4	7.219e-5	1.37e-3

Table 8.5: The $L^2(\Omega)$-error for the 2D Laplace problem with step size $h = \frac{1}{256}$, different ansatz spaces for f, and adaptive arithmetic with an error bound of 10^{-4}.

h	$\|u-u_h\|_{L^2}$	Setup	Memory	Solve	$\|u-u_{4h}\|_{L^2}$	Setup	Memory	Solve
$\frac{1}{64}$	1.814e-4	1s	8MB	<5ms	1.922e-4	1s	5MB	<5ms
$\frac{1}{128}$	4.423e-5	6s	38MB	0.02s	4.896e-5	4s	25MB	0.02s
$\frac{1}{256}$	1.225e-5	32s	174MB	0.12s	2.361e-5	20s	104MB	0.1s
$\frac{1}{512}$	5.356e-5	150s	772MB	0.52s	5.691e-5	99s	454MB	0.45s
$\frac{1}{1024}$	2.325e-4	745s	3443MB	2.47s	2.336e-4	532s	1984MB	1.85s

Table 8.6: Errors, run time and storage behaviour for 2D Laplace with adaptive arithmetic error bound 10^{-4} and $u(x,y) = x^5 - y^5$.

h	$\|u-u_h\|_{L^2}$	Setup	Memory	Solve	$\|u-u_{4h}\|_{L^2}$	Setup	Memory	Solve
$\frac{1}{20}$	3.468e-3	52s	75MB	0.03s	1.334e-2	21s	37MB	0.02s
$\frac{1}{40}$	8.644e-4	810s	867MB	0.39s	3.333e-3	279s	392MB	0.19s
$\frac{1}{60}$	3.751e-4	3849s	3716MB	1.59s	1.502e-3	1339s	1668MB	0.80s
$\frac{1}{80}$	2.013e-4	9812s	8946MB	3.95s	8.713e-4	3388s	3882MB	2s
$\frac{1}{100}$	8.691e-5	23006s	19.5GB	8.85s	6.014e-4	8212s	8589MB	4s

Table 8.7: Errors, run time and storage behaviour for 3D Laplace with adaptive arithmetic error bound 10^{-4} and $u(x,y,z) = x^5 - y^5$.

8.2 Multiscale Problems

Our DD algorithm is also applicable and useful for multiscale problems and now we introduce one possible application of our DD algorithm for the following multiscale problem. The following problem is solved by a multiscale finite element method in [16]:

$$L = \nabla a_\epsilon(x)\nabla, \tag{8.4}$$

$$Lu = g, \tag{8.5}$$

$$u(\mathbf{x}) = \sqrt{\frac{2^d - 1.8^d}{2}} \left(\sum_{i=1}^{d} (x_i)^d \right) \quad \text{on } \partial\Omega, \tag{8.6}$$

with

$$a_\epsilon(\mathbf{x}) = \frac{1}{2^d + 1,8 \sum_{i=1}^{d} \sin(2\pi x_i/\epsilon)}, \quad \mathbf{x} = (x_1, \ldots, x_d)$$

and a small parameter ϵ. For example, such equations arises in composite materials and flows in porous media. The multiscale method used in [15, 16] incorporates the local micro-structures of the differential operator into the finite element base functions.

For this problem we compute the values on the nodal points of the grid Ω_H. For the error calculation we compute a reference solution u_D with the software package deal.II ([18]). The reference solution is computed for a grid with step size $h = \frac{1}{4096}$. As error measurement we use $\|u_D(J(\overline{\Omega})) - u_{mh}(J(\overline{\Omega}))\|_2$. $u_D^J := u_D(J(\overline{\Omega})) \in \mathbb{R}^{J(\overline{\Omega})}$ is the solution restricted to the index set $J(\overline{\Omega})$. $u_{mh}^J := u_{mh}((\overline{\Omega})) \in \mathbb{R}^{J(\overline{\Omega})}$ is the solution for the indices $J(\overline{\Omega})$ computed by the DD algorithm with the right-hand side vector $f_{mh}(\overline{\Omega}) \in \mathbb{R}^{J(\overline{\Omega})}$. The results of Table 8.8 show also a convergence??? and the solution time for one boundary problem. It is necessary that $\varepsilon > h$ holds to discretise the small scale behaviour of the differential operator. The DD algorithm was performed with an error bound of 10^{-6} such that the solving time is bigger than in Table 8.6.

References

[1] M. Bebendorf and W. Hackbusch: *Existence of $\mathcal{H}$-matrix approximants to the inverse FE-matrix of elliptic operators with L^∞-coefficients.* Numer. Math. **95** (2003) 1-28.

ε	h	$\|u_D^J - u_{2h}^J\|_2$	Solve	$\|u_D^J - u_{4h}^J\|_2$	Solve
$\frac{1}{10}$	$\frac{1}{32}$	2.658e-2	>5ms	2.658e-2	>5ms
	$\frac{1}{64}$	6.068e-3	0.01s	6.068e-3	0.01s
	$\frac{1}{128}$	1.511e-3	0.04s	1.511e-3	0.03s
	$\frac{1}{256}$	3.721e-4	0.19s	3.721e-4	0.21s
	$\frac{1}{512}$	8.777e-5	1.14s	8.777e-5	0.77s
	$\frac{1}{1024}$	7.168e-6	4.52s	7.165e-6	3.72s
$\frac{1}{100}$	$\frac{1}{128}$	6.047e-2	0.04s	6.047e-2	0.03s
	$\frac{1}{256}$	2.757e-2	0.25s	2.757e-2	0.17s
	$\frac{1}{512}$	7.986e-3	1.05	7.986e-3	0.86s
	$\frac{1}{1024}$	1.978e-3	4.70s	1.978e-3	3.21s
$\frac{1}{200}$	$\frac{1}{256}$	5.913e-2	0.17s	5.913e-2	0.22s
	$\frac{1}{512}$	2.688e-2	1.01s	2.688e-2	0.76s
	$\frac{1}{1024}$	7.514e-3	4.41s	7.514e-3	3.32s

Table 8.8: 2D multiscale results

[2] S. Börm, L. Grasedyck, and W. Hackbusch: *Introduction to hierarchical matrices with applications.* Eng. Anal. Boundary Elements, **27** (2003) 405-422.

[3] F. Drechsler: *Über die Lösung von elliptischen Randwertproblemen mittels Gebietszerlegungstechniken, Hierarchischer Matrizen und der Mehtode der finiten Elemente.* Doct. thesis, Universität Leipzig (2011).

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

[5] L. Grasedyck, R. Kriemann, and Sabine Le Borne: *Domain decomposition based $\mathcal{H}$-LU preconditioners.* In: O.B. Widlund and D.E. Keyes (eds.), Domain Decomposition Methods in Science and Engineering XVI. Lect. Notes in Computational Science and Engineering **55**, Springer-Verlag, Berlin, 2006; pages 661-668.

[6] L. Grasedyck, R. Kriemann, and Sabine Le Borne: *Parallel black box $\mathcal{H}$-LU preconditioning for elliptic boundary value problems.* Comput. Vis. Sci. **11** (2008) 273-291.

[7] W. Hackbusch: *Elliptic differential equations. Theory and numerical treatment*, Springer, Berlin, 2nd edition, 2003. (Third Germany edition: Lecture Notes 28/2005, Max-Planck-Institut für Mathematik, Leipzig, 2005).

[8] W. Hackbusch: *Multi-grid methods and applications.* Springer, Berlin, 2nd printing, 2003.

[9] W. Hackbusch: *Elliptic differential equations. Theory and numerical treatment*, Vol. 18 of *SCM.* Springer, Berlin, 2nd edition, 2003.

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

[11] W. Hackbusch: *Direct domain decomposition using the hierarchical matrix technique.* In: I. Herrera, D.E. Keyes, O.B. Widlund, and R.Yates (eds.), Domain decomposition methods in science and engineering. Fourteenth international conference on domain decomposition methods, pages 39-50. National Autonomous University of Mexico, Mexico City, 2003.

[12] W. Hackbusch: *Hierarchische Matrizen – Algorithmen und Analysis.* Spinger-Verlag, Heidelberg, 2009.

[13] W. Hackbusch, B. Khoromskij, and R. Kriemann: *Hierarchical matrices based on a weak admissibility criterion.* Computing **73** (2004) 207-243.

[14] W. Hackbusch, B.N. Khoromskij, and R. Kriemann: *Direct Schur complement method by domain decomposition based on $\mathcal{H}$-matrix approximation.* Comput. Vis. Sci. **8** (2005) 179-188.

[15] T. Y. Hou, X.-H. Wu: *A multiscale finite element method for elliptic problems in composite materials and porous media.* Journal of Computational Physics, **134** (1997) 169-189

[16] T. Y. Hou, X.-H. Wu, Z. Cai: *Convergence of a multiscale finite element method for elliptic problems with rapidly oscillating coefficients.* Math. Comp., **68** (1999) 913-943

[17] A.G. Litvinenko: *Application of hierarchical matrices for solving multiscale problems.* Doct. thesis, Universität Leipzig (2007).

[18] www.dealii.org: *A Finite Element Differential Equations Analysis Library*