Partial inversion of the elliptic operator to speed up computation of likelihood in Bayesian inference

Alexander Litvinenko

August 6, 2020

Contents

†E-mail: litvinenko@uq.rwth-aachen.de. RWTH Aachen, Aachen, Germany

Abstract

Often, when solving forward, inverse or data assimilation problems, only a part of the solution is needed. As a model, we consider the stationary diffusion problem. We demonstrate an algorithm that can compute only a part or a functional of the solution, without calculating the full inversion operator and the complete solution. It is a well-known fact about partial differential equations that the solution at each discretisation point depends on the solutions at all other discretisation points. Therefore, it is impossible to compute the solution only at one point, without calculating the solution at all other points. The standard numerical methods like a conjugate gradient or Gauss elimination compute the whole solution and/or the complete inverse operator. We suggest a method which can compute the solution of the given partial differential equation 1) at a point; 2) at few points; 3) on an interface; or a functional of the solution, without computing the solution at all points. The required storage cost and computational resources will be lower as in the standard approach.

With this new method, we can speed up, for instance, computation of the innovation in filtering or the likelihood distribution, which measures the data misfit (mismatch). Further, we can speed up the solution of the regression, Bayesian inversion, data assimilation, and Kalman filter update problems.

Applying additionally the hierarchical matrix approximation, we reduce the cubic computational cost to almost linear $\mathcal{O}(k^2 n \log^2 n)$, where $k \ll n$ and n is the number of degrees of freedom.

Up to the hierarchical matrix approximation error, the computed solution is exact. One of the disadvantages of this method is the need to modify the existing deterministic solver.

Keywords: mismatch, innovation, data misfit, likelihood, Bayesian inversion, Bayesian formula, partial inverse, FEM, domain decomposition, hierarchical matrices, $\mathcal{H}$-matrices, elliptic problem, data-sparse $\mathcal{H}$-matrix approximation, multiscale

AMS 65F10, 60H15, 60H35, 65C30

1 Introduction

We further develop the method, initially introduced in [31, 22, 7, 32] and in Chapter 12 of [21]. With this method, we will be able to compute the solution in a subdomain, in a point, the mean value over a subdomain and other functionals $F(u)$ without computing the full inverse operator and the complete solution. Similar ideas were considered in [1, 39]. This method can be very practical for speeding up the solution of the inverse and data assimilation problems, which appear in many science and engineering applications such as weather prediction, oil recovery, and subsurface flow. Under the inverse problem, we understand the estimation of unknown model parameters from (noisy) measurements. Under the data assimilation problem, we understand improving the existing mathematical model when the new measurement data become available.

The forward problem we consider is the diffusion problem with uncertain or unknown diffusion coefficient. A typical task is not only to solve the forward problem but also to identify this unknown coefficient. Initially, some prior probability density function for the diffusion coefficient is assumed. Then the Bayesian inference is applied to update this density.

Table 0.1: Notation

HDD	suggested here the hierarchical domain decomposition method
$u\|_\gamma$	restriction of the solution u onto the interface γ
h, H	grid step sizes on fine and coarse meshes
Ω, $\partial\Omega$	computational domain and its boundary
Z	random parameter vector $Z = (Z_1, ..., Z_{n_Z})$
Θ	space where parameter $Z = (Z_1, ..., Z_{n_z})$ is defined
ω, $\partial\omega$	local subdomain and its boundary
V_h, V_H	two finite element spaces, $V_H \subset V_h$
$\mathbf{f}$, $\mathbf{f}_h$, $\mathbf{f}_H$	the right hand side, discretized on fine (h) and coarse (H) meshes
$\boldsymbol{u}$, $\boldsymbol{u}_h$, $\boldsymbol{u}_H$	the solution, computed on fine (h) and coarse (H) meshes
$\kappa(x, Z) = e^{q(x,Z)}$	uncertain permeability coefficient, depends on parameter vector Z
$\mathcal{V}_N$	vector space spanned on the basis $\{\varphi_1(x), \ldots, \varphi_N(x)\}$
I, I_N	index sets
$\mathcal{T}_h$, $\mathcal{T}_H$	fine and coarse triangulations
$T_{\mathcal{T}_h}$	hierarchical domain decomposition tree
γ_ω	interface in the domain $\omega \subset \Omega$ (also call "internal" boundary)
$\Gamma_\omega = \partial\omega$	boundary (also call "external" boundary)
d_ω	$d_\omega := \left((f_i)_{i\in I(\omega)}, (g_i)_{i\in I(\partial_\omega)}\right) = (f_\omega, g_\omega)$ a composed vector consisting of the right-hand side restricted to ω and the Dirichlet boundary values $g_\omega = u_h\|_{\partial\omega}$
$\mathcal{F}_h$, $\mathcal{G}_h$	two operators, such that $u_h = \mathcal{F}_h f_h + \mathcal{G}_h g_h$,
$\hat{y}$	true observations
$y = \hat{y} + \varepsilon$	noisy observations
$\Phi^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$	maps the boundary data defined on $\partial\omega$ to the data defined on the interface γ_ω
$\Phi^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$	maps the right-hand side data defined on ω to the data defined on γ_ω.
$\Psi^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\partial\omega)}$	maps the whole subdomain to the external boundary
$\Psi^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}$	maps the external boundary to the external boundary
pdf	probability density function
PCG	preconditioned conjugate gradient

The Bayesian inference is a statistical inference method in which Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available.

Computing the likelihood function in the Bayesian formula requires multiple solutions of the forward problem and could be time-consuming. Depending on the available measurements, the complete solution of the (forward) diffusion problem could be unnecessary, rather only a part of the solution or a functional of the solution is needed. Computing only a part of the solution will make the whole computing process faster and less time-consuming.

Typically, the available measurement data is a functional $F(u)$ of the solution u. The misfit (or mismatch) function is the difference between the simulated data and the measurement values [58, 56, 45, 46, 53, 57]. Below we will show how to simulate these measurement data directly without computing the whole solution. Particularly, we will show that calculating the full inverse operator is unnecessary.

One possible application of our method is the data driven research, a very popular topic nowadays. In this research the available datasets are used either to improve (enrich) the existing mathematical model (often a system of PDEs), or to discover the governing system of PDEs. Another example when fast calculation of a part of the solution is required, is computing the mean square error when comparing the training and computed datasets. In [55], authors design data-driven algorithms for inferring solutions to various partial differential equations. They introduce neural networks that are trained to solve supervised learning tasks while respecting any given laws of physics described by general nonlinear PDEs. Their goal is to solve two classes of problems: data-driven solution and data-driven discovery of PDEs.

The structure of this paper is the following. In Section 1 we give our motivation by introducing the stochastic forward problem and the Bayesian updating procedure for computing posterior density function of the uncertain diffusion coefficient. The main ingredient and the main contribution — the hierarchical domain decomposition (HDD) method — is contained in Section 2. Details of the HDD method, including two algorithms "Leaves to Root" and "Root to Leaves", are shown in Section 3. The hierarchical (denoted by $\mathcal{H}$) - matrix technique to speed up the HDD method is explained in Section 4. Section 5 explains how to use the HDD method to compute a functional of the solution without computing the whole solution. Particularly, it explains how to compute the mean value in a small subdomain. The novelty here is that the whole solution is not available, only a small part of it. In the last section, we conclude the main achievements.

Example. This example shows how the solution (or measurements) in only a few points can reduce the uncertainty. Consider an elliptic PDE with uncertain coefficient and the right hand side as in Eq. 1.1, but in 1D, on the interval $[0, 1]$. We pose uncertain Dirichlet boundary conditions $g(0, \xi)$ and $g(1, \xi)$, where ξ is a Gaussian random variable. Assume three measurements at locations $x = \{0.3, 0.5, 0.8\}$ are given. The mean values $\overline{u}(0.3) = 22$, $\overline{u}(0.5) = 28$, $\overline{u}(0.8) = 18$ and the standard deviations are $\{0.2, 0.3, 0.3\}$ respectively. The following computations are done with the stochastic Galerkin library *sglib*, written by E. Zander at TU Braunschweig [1]. In Fig. 1.1 twenty realisations of the uncertain solution $u(x)$ before and after an update are shown. The mean value (dark bold line) and $\pm\{1, 2, 3\}$ standard deviations (red, orange and yellow lines) are shown. The left picture shows realisations, obtained with some prior assumption about distribution of random diffusion coefficient κ.

[1] `https://github.com/ezander/sglib`

In the following three pictures of the updated solutions are shown, after taking into account one, two and three measurements. To conclude this example, it is practical to have a numerical method, which can efficiently compute a part of the solution or a solution in a few points, without computing the complete solution.

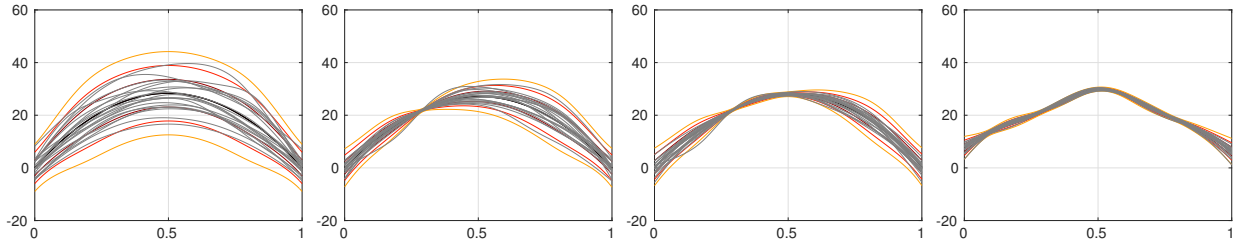

Figure 1.1: (left) 20 original realisations of the solution u (2nd, 3rd, 4th) the same realisations after update; the mean value (bold line) and $\pm 1, 2, 3$ standard deviations (red, orange and yellow lines).

1.1 Main idea

The main ingredients of the developed approach are the weak formulation, the hierarchical (or recursive) domain decomposition technique, the finite element method, and the Schur complement. Additionally, to speed up matrix operations and reduce the overall storage cost, we approximate the involved operators and the Schur complement in the hierarchical matrix format [20, 17, 21].

The **novelty** of this work is the application of the HDD method for faster computation of the innovation in filtering or the likelihood distribution, which measures the data misfit (mismatch).

The forward problem we consider is an elliptic boundary value problem with uncertain L^∞ coefficients and with Dirichlet boundary condition:

$$\begin{array}{ll} -\nabla\left(\kappa(x,Z)\nabla u(x,Z)\right) = f(x), & x \in \Omega \subset \mathbb{R}^2, \\ u = g(x), & x \in \partial\Omega, \end{array} \tag{1.1}$$

where $\kappa(x,Z)$ is a random field dependent on a random parameter $Z = (Z_1, ..., Z_{n_z}) \in \mathbb{R}^{n_z}$, $n_z \geq 1$, consisting of a set of independent continuous random variables characterizing the random coefficient of the governing equation. The solution $u(x,Z)$ is a stochastic quantity, given by

$$u(x,Z) : \overline{\Omega} \times \mathbb{R}^{n_z} \to \mathbb{R}^n, \tag{1.2}$$

where n is the number of finite element nodes in Ω.

For a fixed Z, the solution $u(x,Z)$ belongs to $H^1(\Omega)$, and for a fixed x to $L_2(\Theta)$. There is an established theory about the existence and uniqueness of the solution to Eq. 1.1 under various assumptions on κ and f; see, for example, [2, 11, 13, 43, 48]. In [11, 13] it is shown that under additional assumptions on the right-hand side f and special choices of the test space the problem Eq. 1.1 is well-posed. The case where the Lax-Milgram theorem is not applicable (e.g., upper and lower constants $\underline{\kappa}$, $\overline{\kappa}$ in $0 < \underline{\kappa} < \kappa < \overline{\kappa} < \infty$ do not exist) is also considered in [48]. In [11] the authors analyze assumptions on κ from [2] to guarantee

the uniqueness and the existence of the solution. Additionally, they offer a new method with weaker assumptions. If the expansion of κ is truncated, there is no guaranteed that the truncated series will stay strictly bounded from zero. As a result, the existence of the approximate solution to Eq. 1.1 is questionable, unless precautions are taken as in [43]. The settings where the ellipticity condition is preserved are considered in [11].

Further we assume that each continuous random variable Z_i has a prior distribution

$$F_i(z_i) = P(Z_i \leq z_i) \in [0, 1], \tag{1.3}$$

where P denotes probability and $\pi_i(z_i) = \frac{dF_i(z_i)}{dz_i}$ probability density function (pdf). The joint prior density function for Z is $\pi_Z(z) = \prod_{i=1}^{n_z} \pi_i(z_i)$. For the sake of simplicity, we will skip the subscript $_Z$ and will write $\pi(z)$ for denoting the probability density function of the random variable Z.

The elliptic boundary value problem in Eq. 1.1, can represent, for instance, an incompressible single-phase porous media flow or, another example, a steady state heat conduction through a composite material. In the single-phase flow, u is the flow potential, and κ the permeability of the porous medium. For heat conduction in composite materials, u is the temperature, $-\kappa\nabla u$ the heat flow density, and κ the thermal conductivity.

Iterative methods and preconditioners to solve the problem in Eq. 1.1 were developed in [27, 28, 44, 59, 64]. In [10] the authors assume that the solution has a low-rank *canonical* (CP) tensor format and develop methods for the CP-formatted postprocessing.

Tensor ranks of the stochastic operator were analysed in [47, 9]. The proper generalized decomposition was applied for solving high dimensional stochastic problems in [51, 52]. In [26] authors employed newer tensor formats for the approximation of coefficients and the solution of stochastic elliptic PDEs. Other classical techniques to cope with high-dimensional problems are sparse grids [18, 4, 50] and (quasi) Monte Carlo methods [15, 63, 29]. In [6, 5] authors approximate the polynomial chaos expansion (PCE) of the random input coefficient $\kappa(x, Z)$ in the tensor train (TT) data format, and then solve the problem in that format. A low-rank tensor approximation of random fields, covariance matrices and set of snapshots is done in [25, 37, 35].

1.2 Bayesian updating formula

The inverse problem and propagation of uncertainty through a computational (forward) model are strongly connected. Prior and posterior probabilities express our belief about possible values of the parameters $\kappa(x, Z)$ before and after observations.

Various ideas to speed up the Bayesian updating procedure were presented in [42, 40, 49, 3]. Surrogate based techniques were presented in [56, 47, 34]; reduction of the stochastic dimension by using KLE and PCE expansions in [58, 53, 56]; a non-linear Kalman filter extension in [46, 45, 36].

In [8], the authors develop an approach to Bayesian inference that entirely avoids the Markov chain simulation by constructing a map that pushes forward the prior measure to the posterior measure. The work [60] is devoted to optimal dimensionality reduction techniques for goal-oriented linear-Gaussian inverse problems, where the quantity of interest is a function of the inversion parameters. A multiscale strategy for Bayesian inference using transport maps was introduced in [54].

Further we assume that Θ is a measure space with σ-algebra $\mathcal{A}$ and with a probability measure $\mathbb{P}$, and that $q : \Theta \to \mathcal{Q}$ and $u : \Theta \to \mathcal{U}$ are random variables (RVs). Often, we are not able to observe the entity $q \in \mathcal{Q}$ directly, we can only see a 'shadow' of it, formally given by a 'measurement operator'

$$Y : \mathcal{Q} \times \mathcal{U} \ni (q, u) \mapsto Y(q; u) \in \mathcal{Y}, \tag{1.4}$$

where $q(x, Z) = \log(\kappa(x, Z))$. We assume that the space of possible measurements $\mathcal{Y}$ is a vector space, which frequently can be regarded as finite-dimensional, as one can only observe a finite number of quantities.

The measurement operator Y with values in $\mathcal{Y}$ produces

$$y(Z) = Y(q(Z); u), \quad \text{where} \quad u = u(q(Z)).$$

Examples of measurements are a) $y(Z) = \int_\omega u(Z, x)dx$, with a subdomain $\omega \subset \Omega$, and b) u in a few points. For a given f, the measurement y is just a function of q. This function is usually not invertible since the measurement y does not contain enough information. In the Bayesian framework, the state of knowledge is modeled in a probabilistic way. The parameter q is uncertain and is modeled by a random variable. The Bayesian setting allows updating/sharpening of information about q when the measurement is performed.

Usually the observation of the "truth" $\hat{y} \in \mathbb{R}^{n_y}$ will deviate from what we expect to observe even if we know the right q due to some model error ϵ. The measurement can be also polluted by some measurement error ε. Hence we observe $y = \hat{y} + \epsilon + \varepsilon$, and would like to know what q is. Let $\mathcal{S} : \mathbb{R}^{n_z} \to \mathbb{R}^{n_y}$ be the solution operator (for instance, the set $\{\Phi^g, \Phi^f\}$ or the inverse) of Eq. 1.1. For the sake of simplicity we will only consider one error term

$$y = \hat{y} + \varepsilon = \mathcal{S}(Z) + \varepsilon, \quad \text{where} \quad \varepsilon = (\varepsilon_1, ..., \varepsilon_{n_y}) \in \mathbb{R}^{n_y} \text{ includes all the errors.} \tag{1.5}$$

Here $\varepsilon_1, ..., \varepsilon_{n_y}$ are mutually independent random variables with probability density function $\pi(\varepsilon) = \prod_{i=1}^{n_y} \pi(\varepsilon_i)$. We also assume here that ε and Z are independent.

The mapping in Eq. (1.4) is usually not invertible, and hence the problem is called ill-posed. By modeling our lack of knowledge about q in a Bayesian way [62] with a $\mathcal{Q}$-valued random variable, the problem becomes well-posed [61]. But of course one is looking now at the problem of finding a probability distribution that best fits the data; and one also obtains a probability distribution of q. Here we focus on the use of Bayesian approach [14].

Bayes's theorem is commonly accepted as a consistent way to incorporate new knowledge into a probabilistic description. It may be formulated as ([62] Ch. 1.5)

$$\pi(z|y) = \frac{\pi(y|z)}{\int_\Theta \pi(y|z)\pi_z(z)\mathrm{d}z}\pi_z(z), \tag{1.6}$$

where $\pi_z(z)$ is the pdf of Z, $\pi(y|z)$ is the likelihood as a function of y for fixed prior Z and $\pi(z|y)$ is the posterior pdf of Z conditioned on the data y. We follow the notation from [41]. Numerical approaches for computing a posterior pdf were developed in [40, 42, 61, 56]. Assuming independence on the measurement noise $\varepsilon = (\varepsilon_1, \ldots, \varepsilon_{n_y})$, the likelihood function becomes

$$L(z) := \pi(y|z) = \prod_{i=1}^{n_y} \pi_{\varepsilon_i}(y_i - \mathcal{S}_i(z)). \tag{1.7}$$

Again, we see a formula, where the noisy measurement y_i should be compared with the computed simulation $\mathcal{S}_i(z)$. And very often, the complete solution is not required.

2 Hierarchical domain decomposition (HDD) method

The hierarchical domain decomposition (HDD) method [31] combines the weak formulation, the finite element method (FEM), and the recursive domain decomposition method to obtain a fast and efficient algorithm for computing the partial inverse and a part of the solution (without computing the complete solution). This method was introduced by Hackbusch in 2002 and later on developed in [31, 32, 22, 7, 21].

HDD computes the solution operators $\mathcal{F}_h$ and $\mathcal{G}_h$ in Eq. 2.2, which after applying to the boundary condition and the right-hand side give us the solution.

Below in this section we define the main components of the HDD method - the hierarchical domain decomposition tree (see Fig. 2.2) in Section 2.1, the boundary-to-boundary mappings (Ψ^g) in Section 2.3, domain-to-boundary (Ψ^f) mappings in Section 2.2, boundary-to-interface (Φ^g) and domain-to-interface (Φ^f) mappings which are essential for the definition of the HDD method in Section 2.2.

For a fixed parameter Z, Eq. 1.1 can be written as follow:

$$\begin{aligned} -\nabla\left(\kappa(\mathbf{x})\nabla u(\mathbf{x})\right) &= f(\mathbf{x}), \quad \mathbf{x} \in \Omega \subset \mathbb{R}^2, \\ u &= g(\mathbf{x}), \quad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{2.1}$$

where $\mathbf{x} = (x_1, x_2) \in \Omega$.

The HDD method computes two discrete hierarchical solution operators $\mathcal{F}_h$ and $\mathcal{G}_h$ such that:

$$u_h = \mathcal{F}_h f_h + \mathcal{G}_h g_h, \tag{2.2}$$

where $u_h = u_h(f_h, g_h)$ is the FE solution of 2.1, f_h the discretized right-hand side, and g_h the Dirichlet boundary data. To decrease the computing time and the storage cost, both operators $\mathcal{F}_h$ and $\mathcal{G}_h$ are approximated by $\mathcal{H}$-matrices.

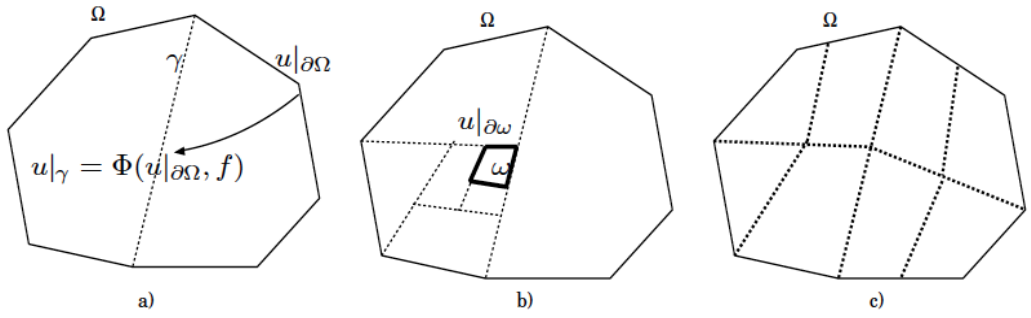

Figure 2.1: (a) The solution $u|_\gamma$ on the interface γ can be computed with the auxiliary operator Φ, by applying it to the right hand side f and to the boundary condition $u|_{\partial\Omega}$; (b) HDD can compute the solution in a subdomain $\omega \subset \Omega$; (c) HDD method can compute the solution on a coarse mesh (shown by dotted lines).

Three examples of possible problem setups, shown in Fig. 2.1, are the following:

1. Suppose the solution on the boundary $\partial\Omega$ (Fig. 2.1 (a)) is given. One is interested in the fast numerical approach which computes the solution $u|_\gamma$ on the interface γ. The solution $u|_\gamma$ depends on the right-hand side and $u|_{\partial\Omega} = g$, i.e. $u|_\gamma = \Phi(u|_{\partial\Omega}, f)$ with some mapping Φ;

2. Only the solution in a small subdomain $\omega \subset \Omega$ is of interest (Fig. 2.1 (b)). To solve the problem in a domain ω the boundary values on $\partial\omega$ are required. How to compute them efficiently from the global boundary data $\partial\Omega$ and the given right-hand side?

3. The third possible problem setup is as follows. The solution on the interface or on a very coarse mesh (see Fig. 2.1 (c)) is required. How can this solution be computed effectively without neglecting small scale features?

Other properties of the HDD method are the following. The HDD allows one to compute $u_h(f_h, g_h)$ for f_h given in a smaller space $V_H \subset V_h$. This could be useful, for instance, in multi-scale settings. The HDD provides the possibility to compute u_h restricted to a coarser grid with reduced computational effort. The HDD shows big advantages in complexity for problems with multiple right-hand sides and multiple Dirichlet data. In this case both operators $\mathcal{F}_h$ and $\mathcal{G}_h$ are computed only once and then applied multiple times to f_h and g_h. Due to the binary tree structure the HDD is an easily parallelizable method. If the problem contains repeated patterns (for instance, so-called *cells* in a multi-scale framework) then the computational resources can be reduced drastically.

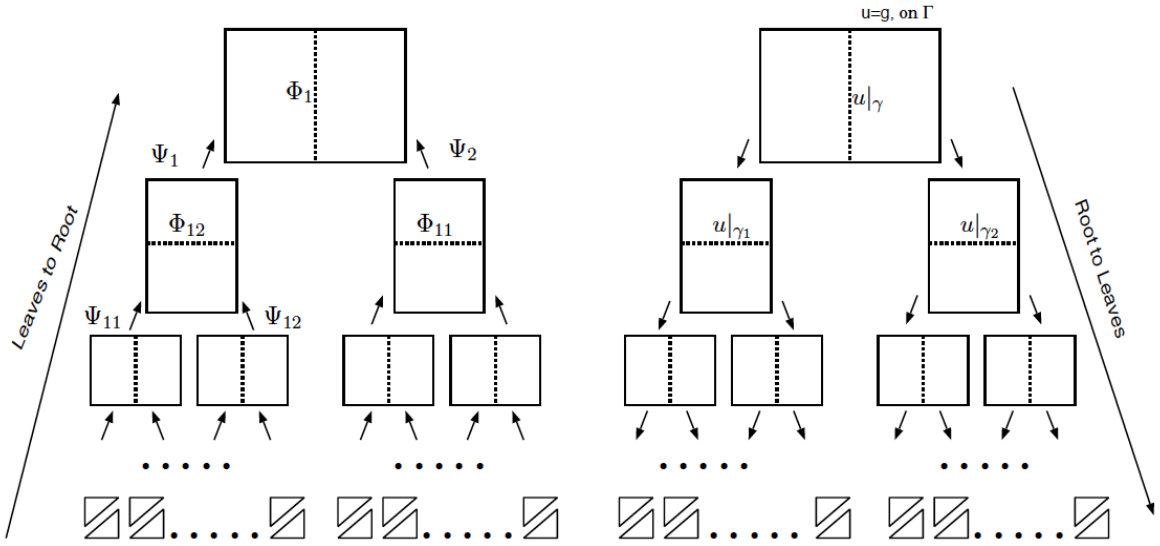

Figure 2.2: HDD contains two algorithms: "Leaves to Root" (shown on the left) which computes mappings $\{\Psi_1, \Psi_2, \Psi_{11}, \Psi_{12}, \ldots\}$ and $\{\Phi_1, \Phi_2, \Phi_{11}, \Phi_{12}, \ldots\}$ and "Root to Leaves" (on the right) which applies mappings $\{\Phi_{ij}\}$ to compute the solutions $u|_{\gamma_i}$ on the interfaces γ_i.

2.1 Notation

Let $\mathcal{T}_h$ be a triangulation of the spatial domain Ω. After hierarchical decomposition of Ω (cf. [12]), obtain the *hierarchical domain decomposition tree* $T_{\mathcal{T}_h}$ (see Fig. 2.2) with the following properties:

- Ω is the root of the tree,
- $T_{\mathcal{T}_h}$ is a binary tree,
- If $\omega \in T_{\mathcal{T}_h}$ has two sons $\omega_1, \omega_2 \in T_{\mathcal{T}_h}$, then $\omega = \omega_1 \cup \omega_2$ and ω_1, ω_2 have no interior point in common,
- $\omega \in T_{\mathcal{T}_h}$ is a leaf, if and only if $\omega \in \mathcal{T}_h$.

The construction of $T_{\mathcal{T}_h}$ is straight-forward by dividing Ω recursively into subdomains. For practical purposes, the subdomains ω_1, ω_2 must both be of size $\approx |\omega|/2$ and the internal boundary

$$\gamma_\omega := \partial\omega_1 \backslash \partial\omega = \partial\omega_2 \backslash \partial\omega \tag{2.3}$$

must not be too large (see Fig. 3.1 (left)).

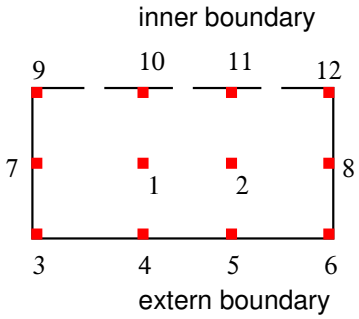

Figure 2.3: Domain $\omega_1 \in T_{\mathcal{T}_h}$ with $I(\omega_1) = \{1, ..., 12\}$, $I(\partial\omega_1) = \{3, 4, 5, 6, 7, 8, 9, 10, 11, 12\}$, $I(\gamma_{\omega_1}) = \{1, 2\}$ to be eliminated via the Schur complement, $I(\Gamma_{\omega_1}) = \{9, 7, 3, 4, 5, 6, 8, 12\}$. On the next level, when ω_1 will be coupled with ω_2, the points $I(\gamma_\omega) = \{10, 11\}$ will be eliminated.

Let $I := I(\overline{\Omega})$ and x_i, $i \in I$, be the set of all nodal points in $\overline{\Omega}$ (including nodal points on the boundary). We define $I(\omega)$ as a subset of I with $x_i \in \omega = \overline{\omega}$. Similarly, we define $I(\mathring{\omega})$, $I(\Gamma_\omega)$, $I(\gamma_\omega)$, where $\Gamma_\omega := \partial\omega$, $\mathring{\omega} = \omega \backslash \partial\omega$, for the interior, for the external boundary and for the interface.

Computing the discrete solution u_h, Eq. 1.1, in Ω is equivalent to the computation of u_h on all γ_ω, $\omega \in T_{\mathcal{T}_h}$, since $I(\Omega) = \cup_{\omega \in T_{\mathcal{T}_h}} I(\gamma_\omega)$. These computations are performed by using the linear mappings Φ^f_ω, Φ^g_ω defined for all nodes $\omega \in T_{\mathcal{T}_h}$.

Notation 2.1 *Let $g_\omega := u|_{I(\partial\omega)}$ be the local Dirichlet data and $f_\omega := f|_{I(\omega)}$ be the local right-hand side.*

Definition 2.1 *The mapping* $\Phi^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$ *maps the boundary data defined on* $\partial\omega$ *to the data defined on the interface* γ_ω. $\Phi^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$ *maps the right-hand side data defined on* ω *to the data defined on* γ_ω.

The final aim is to compute the solution u_h along γ_ω in the form $u_h|_{\gamma_\omega} = \Phi^f_\omega f_\omega + \Phi^g_\omega g_\omega$, $\omega \in T_{\mathcal{T}_h}$. For this purpose HDD builds the mappings $\Phi_\omega := (\Phi^g_\omega, \Phi^f_\omega)$, for all $\omega \in T_{\mathcal{T}_h}$. For computing the mapping Φ_ω, $\omega \in T_{\mathcal{T}_h}$, we first need to compute the auxiliary mapping $\Psi_\omega := (\Psi^g_\omega, \Psi^f_\omega)$ which will be defined later.

Thus, the HDD method consists of two steps: the first step is the construction of the mappings Φ^g_ω and Φ^f_ω for all $\omega \in T_{\mathcal{T}_h}$. The second step is the recursive computation of the solution u_h. In the second step HDD applies the mappings Φ^g_ω and Φ^f_ω to the local Dirichlet data g_ω and to the local right-hand side f_ω.

Notation 2.2 *Let* $\omega \in T_{\mathcal{T}_h}$ *and*

$$d_\omega := \left((f_i)_{i\in I(\omega)}, (g_i)_{i\in I(\partial_\omega)} \right) = (f_\omega, g_\omega) \tag{2.4}$$

be a composed vector consisting of the right-hand side from Eq. 1.1 restricted to ω *and the Dirichlet boundary values* $g_\omega = u_h|_{\partial\omega}$ *(see also Notation 2.1).*

Note that g_ω coincides with the global Dirichlet data in Eq. 1.1 only when $\omega = \Omega$. For all other $\omega \in T_{\mathcal{T}_h}$ we compute g_ω in (Eq. 2.4) by the algorithm "Root to Leaves" (see Section 3.4).

Assuming that the elliptic boundary value problem, Eq. 2.1, restricted to ω is solvable, we can define the local FE solution by solving the following discrete problem in the variational form [19]:

$$\begin{cases} a_\omega(U_\omega, b_j) = (f_\omega, b_j)_{L^2(\omega)}, & \forall\, j \in I(\mathring{\omega}), \\ U_\omega(\mathbf{x}_j) = g_j, & \forall\, j \in I(\partial\omega). \end{cases} \tag{2.5}$$

Here, b_j is the P^1-Lagrange basis function at $\mathbf{x}_j$ and $a_\omega(\cdot,\cdot)$ is the bilinear form (see Eq. 1.1) with integration restricted to ω and $(f_\omega, b_j) = \int\limits_\omega f_\omega\, b_j\, d\mathbf{x}$.

Let $U_\omega \in V_h$ be the solution of (Eq. 2.5) in ω. The solution U_ω depends on the Dirichlet data on $\partial\omega$ and the right-hand side in ω. Dividing problem (Eq. 2.5) into two subproblems (Eq. 2.6) and (Eq. 2.7), we obtain $U_\omega = U^f_\omega + U^g_\omega$, where U^f_ω is the solution of

$$\begin{cases} a_\omega(U^f_\omega, b_j) = (f_\omega, b_j)_{L^2(\omega)}, & \forall\, j \in I(\mathring{\omega}), \\ U^f_\omega(\mathbf{x}_j) = 0, & \forall\, j \in I(\partial\omega) \end{cases} \tag{2.6}$$

and U^g_ω is the solution of

$$\begin{cases} a_\omega(U^g_\omega, b_j) = 0, & \forall\, j \in I(\mathring{\omega}), \\ U^g_\omega(\mathbf{x}_j) = g_j, & \forall\, j \in I(\partial\omega). \end{cases} \tag{2.7}$$

If $\omega = \Omega$ then (Eq. 2.5) is equivalent to the initial problem Eq. 2.1 in the weak formulation.

2.2 Mapping $\Phi_\omega = (\Phi^g_\omega, \Phi^f_\omega)$

In this section we define mappings Φ_ω, Φ^g_ω, Φ^f_ω. We consider $\omega \in T_{\mathcal{T}_h}$ with two sons ω_1, ω_2. Considering once more the data d_ω from (Eq. 2.4), U^f_ω from (Eq. 2.6) and U^g_ω from (Eq. 2.7), we define $\Phi^f_\omega(f_\omega)$ and $\Phi^g_\omega(g_\omega)$ by

$$\left(\Phi^f_\omega(f_\omega)\right)_i := U^f_\omega(\mathbf{x}_i) \quad \forall i \in I(\gamma_\omega) \tag{2.8}$$

and

$$\left(\Phi^g_\omega(g_\omega)\right)_i := U^g_\omega(\mathbf{x}_i) \quad \forall i \in I(\gamma_\omega). \tag{2.9}$$

Since $U_\omega = U^f_\omega + U^g_\omega$, we obtain

$$\left(\Phi_\omega(d_\omega)\right)_i := \Phi^g_\omega(g_\omega) + \Phi^f_\omega(f_\omega) = U^f_\omega(\mathbf{x}_i) + U^g_\omega(\mathbf{x}_i) = U_\omega(\mathbf{x}_i) \tag{2.10}$$

for all $i \in I(\gamma_\omega)$.
Hence, $\Phi_\omega(d_\omega)$ is the trace of U_ω on γ_ω. Definition in (Eq. 2.10) says that if the data d_ω are given then Φ_ω computes the solution of (Eq. 2.5). Indeed, $\Phi_\omega d_\omega = \Phi^g g_\omega + \Phi^f f_\omega$. Note that the solution u_h of the initial global problem coincide with U_ω in ω, i.e., $u_h|_\omega = U_\omega$.

2.3 Mapping $\Psi_\omega = (\Psi^g_\omega, \Psi^f_\omega)$

In this section we define mappings Ψ_ω, Ψ^g_ω, Ψ^f_ω.

First, we define the mapping Ψ^f_ω from (Eq. 2.6) as

$$\left(\Psi^f_\omega(d_\omega)\right)_{i\in I(\partial\omega)} := a_\omega(U^f_\omega, b_i) - (f_\omega, b_i)_{L^2(\omega)}\,, \tag{2.11}$$

where $U^f_\omega \in V_h$, $U^f_\omega|_{\partial\omega} = 0$ and

$$a(U^f_\omega, b_i) - (f, b_i) = 0, \quad \text{for } \forall i \in I(\mathring{\omega}).$$

Second, we define the mapping Ψ^g_ω from (Eq. 2.7) by setting

$$(\Psi^g_\omega(d_\omega))_{i\in I(\partial\omega)} := a_\omega(U^g_\omega, b_i) - (f_\omega, b_i)_{L^2(\omega)} = a_\omega(U^g_\omega, b_i) - 0 = a_\omega(U^g_\omega, b_i), \tag{2.12}$$

where $U^g_\omega \in V_h$ and $(\Psi^g_\omega(d_\omega))_i = 0$ for $\forall i \in I(\mathring{\omega})$.

The linear mapping Ψ_ω, which maps the data d_ω given by (Eq. 2.4) to the boundary data on $\partial\omega$, is given in the component form as

$$\Psi_\omega(d_\omega) = (\Psi_\omega(d_\omega))_{i\in I(\partial\omega)} := a_\omega(U_\omega, b_i) - (f_\omega, b_i)_{L^2(\omega)}\,. \tag{2.13}$$

By definition Ψ_ω is linear in (f_ω, g_ω) and can be written as $\Psi_\omega(d_\omega) = \Psi^f_\omega f_\omega + \Psi^g_\omega g_\omega$. Here U_ω is the solution of the local problem (Eq. 2.5) and it coincides with the global solution on $I(\omega)$.

2.4 Φ_ω and Ψ_ω in terms of the Schur complement matrix

Let the linear system $A\boldsymbol{u} = F\mathbf{c}$ for $\omega \in T_{\mathcal{T}_h}$ be given. In Sections 3.1 and 3.3 we explain how to obtain the matrices A and F. A is the stiffness matrix for the domain $\overline{\omega}$ after elimination of the unknowns corresponding to $I(\overset{\circ}{\omega} \setminus \gamma_\omega)$. The matrix F comes from the applied numerical integration rule [31].

We will write for simplicity γ instead of γ_ω. Thus, $A : \mathbb{R}^{I(\partial\omega\cup\gamma)} \to \mathbb{R}^{I(\partial\omega\cup\gamma)}$, $\boldsymbol{u} \in \mathbb{R}^{I(\partial\omega\cup\gamma)}$, $F : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\partial\omega\cup\gamma)}$ and $\mathbf{c} \in \mathbb{R}^{I(\omega)}$. Decomposing the unknown vector $\boldsymbol{u}$ into two components $\boldsymbol{u}_1 \in \mathbb{R}^{I(\partial\omega)}$ and $\boldsymbol{u}_2 \in \mathbb{R}^{I(\gamma)}$, obtain

$$\boldsymbol{u} = \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix}.$$

The component $\boldsymbol{u}_1$ corresponds to the boundary $\partial\omega$ and the component $\boldsymbol{u}_2$ to the interface γ. Then the equation $A\boldsymbol{u} = F\mathbf{c}$ becomes

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \mathbf{c}, \tag{2.14}$$

where

$$\begin{aligned} A_{11} &: \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}, \quad A_{12} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\partial\omega)}, \\ A_{21} &: \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\gamma)}, \quad A_{22} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\gamma)}, \\ F_1 &: \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\partial\omega)}, \quad F_2 : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\gamma)}. \end{aligned}$$

The elimination of the internal points is done as it is shown in (Eq. 2.15) below

$$\begin{pmatrix} A_{11} - A_{12}A_{22}^{-1}A_{21} & 0 \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \boldsymbol{u}_2 \end{pmatrix} = \begin{pmatrix} F_1 - A_{12}A_{22}^{-1}F_2 \\ F_2 \end{pmatrix} \mathbf{c}. \tag{2.15}$$

We rewrite the last system as two equations

$$\begin{gathered} \tilde{A}\boldsymbol{u}_1 := (A_{11} - A_{12}A_{22}^{-1}A_{21})\boldsymbol{u}_1 = (F_1 - A_{12}A_{22}^{-1}F_2)\mathbf{c}, \\ \boldsymbol{u}_2 = A_{22}^{-1}F_2\mathbf{c} - A_{22}^{-1}A_{21}\boldsymbol{u}_1. \end{gathered} \tag{2.16}$$

The explicit expressions for the mappings Ψ_ω and Φ_ω follow from (Eq. 2.16):

$$\Psi_\omega^g := A_{11} - A_{12}A_{22}^{-1}A_{21}, \quad \Psi_\omega^f := F_1 - A_{12}A_{22}^{-1}F_2, \tag{2.17}$$

$$\Phi_\omega^g := -A_{22}^{-1}A_{21}, \quad \Phi_\omega^f := A_{22}^{-1}F_2. \tag{2.18}$$

Thus, $\boldsymbol{u}_2 = \Phi_\omega^f(f_\omega) + \Phi_\omega^g(g_\omega)$, with the rhs $f_\omega = \mathbf{c}$, and local b.c. $g_\omega = \boldsymbol{u}_1$.

3 Construction Process

In this section we explain the recursive construction of mappings Ψ_ω^g, Ψ_ω^f, Φ_ω^g and Φ_ω^f.

3.1 Initialisation of the recursion

This section explains how to compute mapping Ψ_ω^f for the leaves of $T_{\mathcal{T}_h}$ and how it is connected with the quadrature rule.

Our purpose is to get for each triangle $\omega \in \mathcal{T}_h$, the system of linear equations

$$A \cdot \boldsymbol{u} = \tilde{\mathbf{c}} := F \cdot \mathbf{c}, \tag{3.1}$$

where A is the stiffness matrix, $\mathbf{c}$ the discrete values of the right-hand side in the nodes of ω and F will be defined later. The matrix coefficients A_{ij} are computed by the formula

$$A_{ij} = \int_\omega \kappa(\mathbf{x}) \langle \nabla b_i(\mathbf{x}) \cdot \nabla b_j(\mathbf{x}) \rangle d\mathbf{x}, \tag{3.2}$$

where $b_i(\mathbf{x})$ is a piecewise linear basis function [19]. For $\omega \in \mathcal{T}_h$, $F \in \mathbb{R}^{3\times 3}$ comes from the discrete integration and the matrix coefficients F_{ij} are computed using (Eq. 3.5). The components of $\tilde{\mathbf{c}}$ can be computed as follows:

$$\tilde{c}_i = \int_\omega f b_i d\mathbf{x} \approx \frac{f(\mathbf{x}_1)b_i(\mathbf{x}_1) + f(\mathbf{x}_2)b_i(\mathbf{x}_2) + f(\mathbf{x}_3)b_i(\mathbf{x}_3)}{3} \cdot |\omega|, \tag{3.3}$$

where $\mathbf{x}_i$, $i \in \{1, 2, 3\}$, are three vertices of the triangle $\omega \in T_{\mathcal{T}_h}$, $b_i(\mathbf{x}_j) = 1$ if $i = j$ and $b_i(\mathbf{x}_j) = 0$ otherwise. Rewrite (Eq. 3.3) in matrix form:

$$\tilde{\mathbf{c}} = \begin{pmatrix} \tilde{c}_1 \\ \tilde{c}_2 \\ \tilde{c}_3 \end{pmatrix} \approx \frac{1}{3} \begin{pmatrix} b_1(\mathbf{x}_1) & b_1(\mathbf{x}_2) & b_1(\mathbf{x}_3) \\ b_2(\mathbf{x}_1) & b_2(\mathbf{x}_2) & b_2(\mathbf{x}_3) \\ b_3(\mathbf{x}_1) & b_3(\mathbf{x}_2) & b_3(\mathbf{x}_3) \end{pmatrix} \begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ f(\mathbf{x}_3) \end{pmatrix}, \tag{3.4}$$

where $f(\mathbf{x}_i)$, $i = 1, 2, 3$, are the values of the right-hand side f in the vertices of ω. Then, for piecewise linear basis functions obtain

$$F := \frac{1}{3} \begin{pmatrix} b_1(\mathbf{x}_1) & b_1(\mathbf{x}_2) & b_1(\mathbf{x}_3) \\ b_2(\mathbf{x}_1) & b_2(\mathbf{x}_2) & b_2(\mathbf{x}_3) \\ b_3(\mathbf{x}_1) & b_3(\mathbf{x}_2) & b_3(\mathbf{x}_3) \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \text{ and } c := \begin{pmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ f(\mathbf{x}_3) \end{pmatrix}. \tag{3.5}$$

Thus, Ψ_ω^g corresponds to the matrix $A \in \mathbb{R}^{3\times 3}$ and Ψ_ω^f to $F \in \mathbb{R}^{3\times 3}$.

3.2 Recursion

This section explains how to build Ψ_ω from Ψ_{ω_1} and Ψ_{ω_2}, with $\omega \in T_{\mathcal{T}_h}$ and ω_1, ω_2 be two sons of ω. The coefficients of Ψ_ω can be computed by (Eq. 2.13). The external boundary Γ_ω of ω splits into (see Fig. 3.1 (left))

$$\Gamma_{\omega,1} := \partial\omega \cap \omega_1, \quad \Gamma_{\omega,2} := \partial\omega \cap \omega_2. \tag{3.6}$$

For simplicity of further notations, we will write γ instead of γ_ω.

Notation 3.1 *Recall that $I(\partial\omega_i) = I(\Gamma_{\omega,i}) \cup I(\gamma)$. We denote the restriction of $\Psi_{\omega_i} : \mathbb{R}^{I(\partial\omega_i)} \to \mathbb{R}^{I(\partial\omega_i)}$ to $I(\gamma)$ by ${}^\gamma\Psi_\omega := (\Psi_\omega)|_{i\in I(\gamma)}$.*

Suppose that by induction, the mappings Ψ_{ω_1}, Ψ_{ω_2} are known for the sons ω_1, ω_2. Now, we explain how to construct Ψ_ω and Φ_ω.

Lemma 3.1 *Let the data $d_1 = d_{\omega_1}$, $d_2 = d_{\omega_2}$ be given by (Eq. 2.4). Data d_1 and d_2 coincide along γ_ω, i.e.,*

- *(consistency conditions for the boundary)*

$$g_{1,i} = g_{2,i} \quad \forall i \in I(\omega_1) \cap I(\omega_2), \tag{3.7}$$

- *(consistency conditions for the right-hand side)*

$$f_{1,i} = f_{2,i} \quad \forall i \in I(\omega_1) \cap I(\omega_2). \tag{3.8}$$

If the local FE solutions $u_{h,1}$ and $u_{h,2}$ of the problem (2.5) for the data d_1, d_2 satisfy the additional equation

$$^{\gamma}\Psi_{\omega_1}(d_1) + {}^{\gamma}\Psi_{\omega_2}(d_2) = 0, \tag{3.9}$$

then the composed solution u_h defined by assembling

$$u_h(\mathbf{x}_i) = \begin{cases} u_{h,1}(\mathbf{x}_i) & for \quad i \in I(\omega_1), \\ u_{h,2}(\mathbf{x}_i) & for \quad i \in I(\omega_2) \end{cases} \tag{3.10}$$

satisfies (Eq. 2.5) for the data $d_\omega = (f, g)$ where

$$f_i = \begin{cases} f_{1,i} & for \quad i \in I(\omega_1), \\ f_{2,i} & for \quad i \in I(\omega_2), \end{cases} \tag{3.11}$$

$$g_i = \begin{cases} g_{1,i} & for \quad i \in I(\Gamma_{\omega,1}), \\ g_{2,i} & for \quad i \in I(\Gamma_{\omega,2}). \end{cases} \tag{3.12}$$

Proof: Note that the index sets in (Eq. 3.10)-(Eq. 3.12) overlap. Let $\omega_1 \in T_{\mathcal{T}_h}$, $f_{1,i} = f_i$, $i \in I(\omega_1)$, and $g_{1,i} = g_i$, $i \in I(\partial\omega_1)$. Then the existence of the unique solutions of (Eq. 2.5) gives $u_{h,1}(\mathbf{x}_i) = u_h(\mathbf{x}_i)$, $\forall i \in I(\overset{\circ}{\omega_1})$.
In a similar manner we get $u_{h,2}(\mathbf{x}_i) = u_h(\mathbf{x}_i)$, $\forall i \in I(\overset{\circ}{\omega_2})$. Equation (Eq. 2.13) gives

$$(\ {}^{\gamma}\Psi_{\omega_1}(d_1))_{i \in I(\gamma)} = a_{\omega_1}(u_h, b_i) - (f_{\omega_1}, b_i)_{L^2(\omega_1)} \tag{3.13}$$

and

$$(\ {}^{\gamma}\Psi_{\omega_2}(d_2))_{i \in I(\gamma)} = a_{\omega_2}(u_h, b_i) - (f_{\omega_2}, b_i)_{L^2(\omega_2)}\,. \tag{3.14}$$

The sum of the two last equations (see Figure 3.1 (right)) and (Eq. 3.9) give

$$0 = {}^{\gamma}\Psi_{\omega}(d_\omega)_{i \in I(\gamma)} = a_\omega(u_h, b_i) - (f_\omega, b_i)_{L^2(\omega)}. \tag{3.15}$$

We see that u_h satisfies (Eq. 2.5). ■
Note that

$$u_{h,1}(\mathbf{x}_i) = g_{1,i} = g_{2,i} = u_{h,2}(\mathbf{x}_i) \quad \text{holds for} \quad i \in I(\omega_1) \cap I(\omega_2).$$

Next, we use the decomposition of the data d_1 into the components

$$d_1 = (f_1, g_{1,\Gamma}, g_{1,\gamma}), \tag{3.16}$$

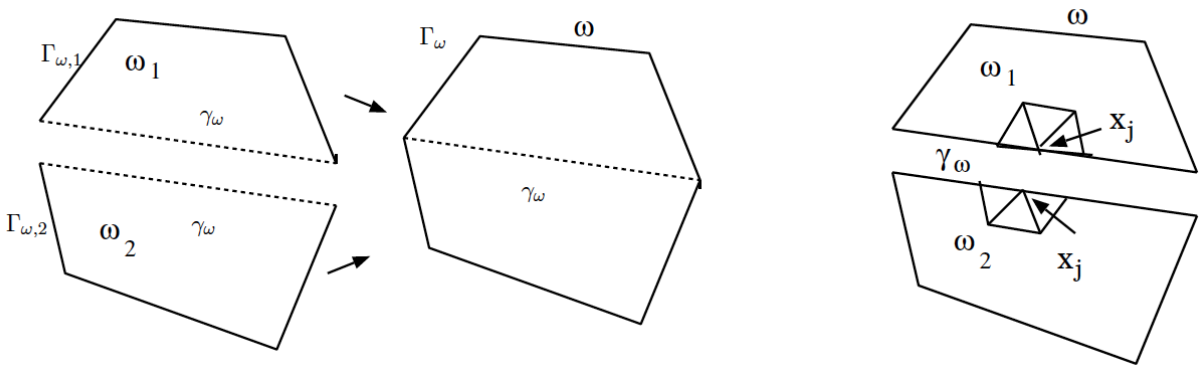

Figure 3.1: (**left**) Domain ω and its two sons ω_1 and ω_2. Here γ_ω is the internal boundary and $\Gamma_{\omega,i}$, $i = 1, 2$, parts of the external boundaries, see (Eq. 3.6). (**right**) The support of basis function b_j, $x_j \in \omega_1$ and $x_j \in \omega_2$.

where

$$g_{1,\Gamma} := (g_1)_{i \in I(\Gamma_{\omega,1})}, \quad g_{1,\gamma} := (g_1)_{i \in I(\gamma)} \tag{3.17}$$

and similarly for $d_2 = (f_2, g_{2,\Gamma}, g_{2,\gamma})$.
The decomposition $g \in \mathbb{R}^{I(\partial\omega_j)}$ into $g_{j,\Gamma} \in \mathbb{R}^{I(\Gamma_{\omega,j})}$ and $g_{j,\gamma} \in \mathbb{R}^{I(\gamma)}$ implies the decomposition of $\Psi^g_{\omega_j} : \mathbb{R}^{I(\partial\omega_j)} \to \mathbb{R}^{I(\partial\omega_j)}$ into $\Psi^\Gamma_{\omega_j} : \mathbb{R}^{I(\Gamma_{\omega,j})} \to \mathbb{R}^{I(\partial\omega_j)}$ and $\Psi^\gamma_{\omega_j} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\partial\omega_j)}$, $j = 1, 2$.
Thus, $\Psi^g_{\omega_1} g_{\omega_1} = \Psi^\Gamma_{\omega_1} g_{1,\Gamma} + \Psi^\gamma_{\omega_1} g_{1,\gamma}$ and $\Psi^g_{\omega_2} g_{\omega_2} = \Psi^\Gamma_{\omega_2} g_{2,\Gamma} + \Psi^\gamma_{\omega_2} g_{2,\gamma}$.
The maps Ψ_{ω_1}, Ψ_{ω_2} become

$$\Psi_{\omega_1} d_1 = \Psi^f_{\omega_1} f_1 + \Psi^\Gamma_{\omega_1} g_{1,\Gamma} + \Psi^\gamma_{\omega_1} g_{1,\gamma}, \tag{3.18}$$

$$\Psi_{\omega_2} d_2 = \Psi^f_{\omega_2} f_2 + \Psi^\Gamma_{\omega_2} g_{2,\Gamma} + \Psi^\gamma_{\omega_2} g_{2,\gamma}. \tag{3.19}$$

Definition 3.1 *We will denote the restriction of* $\Psi^\gamma_{\omega_j} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\partial\omega_j)}$ *to* $I(\gamma)$ *by*

$$^\gamma\Psi^\gamma_{\omega_j} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\gamma)},$$

where $j = 1, 2$ *and* $\partial\omega_j = \Gamma_{\omega,j} \cup \gamma$.

Restricting (Eq. 3.18), (Eq. 3.19) to $I(\gamma)$, we obtain from (Eq. 3.9) and $g_{1,\gamma} = g_{2,\gamma} =: g_\gamma$ that

$$\left({}^\gamma\Psi^\gamma_{\omega_1} + {}^\gamma\Psi^\gamma_{\omega_2} \right) g_\gamma = (-\Psi^f_{\omega_1} f_1 - \Psi^\Gamma_{\omega_1} g_{1,\Gamma} - \Psi^f_{\omega_2} f_2 - \Psi^\Gamma_{\omega_2} g_{2,\Gamma})|_{I(\gamma)}.$$

Next, we set $M := -({}^\gamma\Psi^\gamma_{\omega_1} + {}^\gamma\Psi^\gamma_{\omega_2})$. and after computing M^{-1}, we obtain:

$$g_\gamma = M^{-1}(\Psi^f_{\omega_1} f_1 + \Psi^\Gamma_{\omega_1} g_{1,\Gamma} + \Psi^f_{\omega_2} f_2 + \Psi^\Gamma_{\omega_2} g_{2,\Gamma})|_{I(\gamma)}. \tag{3.20}$$

Remark 3.1 *The inverse matrix* M^{-1} *exists since it is the sum of positive definite matrices corresponding to the mappings* ${}^\gamma\Psi^\gamma_{\omega_1}$, ${}^\gamma\Psi^\gamma_{\omega_2}$.

Remark 3.2 *Since* $g_{\gamma,i} = u_h(\mathbf{x}_i)$, $i \in I(\gamma)$, *we have determined the map* Φ_ω *(it acts on the data* d_ω *composed by* f_1, f_2, $g_{1,\Gamma}, g_{2,\Gamma}$*).*

Remark 3.3 *We have the formula* $\Psi_\omega(d_\omega) = \Psi_{\omega_1}(d_1) + \Psi_{\omega_2}(d_2)$, *where*

$$\begin{gathered} d_\omega = (f_\omega, g_\omega), \quad d_1 = (f_1, g_{1,\Gamma}, g_{1,\gamma}), \quad d_2 = (f_2, g_{2,\Gamma}, g_{2,\gamma}), \\ g_{1,\gamma} = g_{2,\gamma} = M^{-1}(\Psi^f_{\omega_1} f_1 + \Psi^\Gamma_{\omega_1} g_{1,\Gamma} + \Psi^f_{\omega_2} f_2 + \Psi^\Gamma_{\omega_2} g_{2,\Gamma})|_{I(\gamma)}. \end{gathered} \tag{3.21}$$

Here (f_ω, g_ω) *is build as in (Eq. 3.11)-(Eq. 3.12) and (Eq. 3.7),(Eq. 3.8) are satisfied.*

Conclusion:
Thus, using the given mappings Ψ_{ω_1}, Ψ_{ω_2}, defined on the sons $\omega_1, \omega_2 \in T_{\mathcal{T}_h}$, we can compute Φ_ω and Ψ_ω for the father $\omega \in T_{\mathcal{T}_h}$.

3.3 Building of Matrices Ψ_ω and Φ_ω from Ψ_{ω_1} and Ψ_{ω_2}

Let ω, ω_1 where $\omega_2 \in T_{\mathcal{T}_h}$ and ω_1, ω_2 are sons of ω. Recall that $\partial\omega_i = \Gamma_{\omega,i} \cup \gamma$. Suppose we have two linear systems of equations for ω_1 and ω_2 which can be written in the block-matrix form:

$$\begin{pmatrix} A^{(i)}_{11} & A^{(i)}_{12} \\ A^{(i)}_{21} & A^{(i)}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(i)}_1 \\ \boldsymbol{u}^{(i)}_2 \end{pmatrix} = \begin{pmatrix} F^{(i)}_{11} & F^{(i)}_{12} \\ F^{(i)}_{21} & F^{(i)}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{c}^{(i)}_1 \\ \mathbf{c}^{(i)}_2 \end{pmatrix}, \quad i = 1, 2, \tag{3.22}$$

where $\gamma := \gamma_\omega$,

$$\begin{gathered} A^{(i)}_{11} : \mathbb{R}^{I(\Gamma_{\omega,i})} \to \mathbb{R}^{I(\Gamma_{\omega,i})}, \quad A^{(i)}_{12} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\Gamma_{\omega,i})}, \\ A^{(i)}_{21} : \mathbb{R}^{I(\Gamma_{\omega,i})} \to \mathbb{R}^{I(\gamma)}, \quad A^{(i)}_{22} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\gamma)}, \\ F^{(i)}_{11} : \mathbb{R}^{I(\omega_i \setminus \gamma)} \to \mathbb{R}^{I(\partial\omega_i)}, \quad F^{(i)}_{12} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\partial\omega_i)}, \\ F^{(i)}_{21} : \mathbb{R}^{I(\omega_i \setminus \gamma)} \to \mathbb{R}^{I(\gamma)}, \quad F^{(i)}_{22} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}^{I(\gamma)}. \end{gathered}$$

Both the equations in (Eq. 3.22) are analogous to (Eq. 3.18) and (Eq. 3.19). Note that $\mathbf{c}^{(1)}_2 = \mathbf{c}^{(2)}_2$ and $\boldsymbol{u}^{(1)}_2 = \boldsymbol{u}^{(2)}_2$ because of the consistency conditions (see (Eq. 3.7),(Eq. 3.8)) on the interface γ. The system of linear equations for ω be

$$\begin{pmatrix} A^{(1)}_{11} & 0 & A^{(1)}_{12} \\ 0 & A^{(2)}_{11} & A^{(2)}_{12} \\ A^{(1)}_{21} & A^{(2)}_{21} & A^{(1)}_{22} + A^{(2)}_{22} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}^{(1)}_1 \\ \boldsymbol{u}^{(2)}_1 \\ \boldsymbol{u}^{(1)}_2 \end{pmatrix} = \begin{pmatrix} F^{(1)}_{11} & 0 & F^{(1)}_{12} \\ 0 & F^{(2)}_{11} & F^{(2)}_{12} \\ F^{(1)}_{21} & F^{(2)}_{21} & F^{(1)}_{22} + F^{(2)}_{22} \end{pmatrix} \begin{pmatrix} \mathbf{c}^{(1)}_1 \\ \mathbf{c}^{(2)}_1 \\ \mathbf{c}^{(1)}_2 \end{pmatrix}. \tag{3.23}$$

See the left matrix in Fig. A.1 in the Appendix. Using the notation

$$\tilde{A}_{11} := \begin{pmatrix} A^{(1)}_{11} & 0 \\ 0 & A^{(2)}_{11} \end{pmatrix}, \quad \tilde{A}_{12} := \begin{pmatrix} A^{(1)}_{12} \\ A^{(2)}_{12} \end{pmatrix},$$

$$\tilde{A}_{21} := (A^{(1)}_{21}, A^{(2)}_{21}), \quad \tilde{A}_{22} := A^{(1)}_{22} + A^{(2)}_{22},$$

$$\tilde{\boldsymbol{u}}_1 := \begin{pmatrix} \boldsymbol{u}^{(1)}_1 \\ \boldsymbol{u}^{(2)}_1 \end{pmatrix}, \quad \tilde{\boldsymbol{u}}_2 := \boldsymbol{u}^{(1)}_2 = \boldsymbol{u}^{(2)}_2,$$

$$\tilde{F}_1 := \begin{pmatrix} F^{(1)}_{11} & 0 & F^{(1)}_{12} \\ 0 & F^{(2)}_{11} & F^{(2)}_{12} \end{pmatrix}, \quad \tilde{F}_2 := \begin{pmatrix} F^{(1)}_{21}, & F^{(2)}_{21}, & F^{(1)}_{22} + F^{(2)}_{22} \end{pmatrix},$$

$$\tilde{\mathbf{c}}_1 := \begin{pmatrix} \mathbf{c}_1^{(1)} \\ \mathbf{c}_1^{(2)} \end{pmatrix}, \quad \tilde{\mathbf{c}}_2 := \mathbf{c}_2^{(1)} = \mathbf{c}_2^{(2)}, \quad \tilde{\mathbf{c}} := \begin{pmatrix} \tilde{\mathbf{c}}_1 \\ \tilde{\mathbf{c}}_2 \end{pmatrix},$$

the system (Eq. 3.23) can be rewritten as

$$\begin{pmatrix} \tilde{A}_{11}^{(i)} & \tilde{A}_{12}^{(i)} \\ \tilde{A}_{21}^{(i)} & \tilde{A}_{22}^{(i)} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{u}}_1 \\ \tilde{\boldsymbol{u}}_2 \end{pmatrix} = \begin{pmatrix} \tilde{F}_1 \\ \tilde{F}_2 \end{pmatrix} \tilde{\mathbf{c}}. \tag{3.24}$$

The system (Eq. 3.24), indeed, coincides with (Eq. 2.14). After elimination of variables $\boldsymbol{u}_1^{(2)}$ (on the interface), we obtain the matrices as it shown in Figures 4.1 and 4.2.

3.4 Algorithms "Leaves to Root" and "Root to Leaves"

The scheme of the recursive process of computing Ψ_ω and Φ_ω from Ψ_{ω_1} and Ψ_{ω_2} for all $\omega \in T_{\mathcal{T}_h}$ is shown in Fig. 2.2 (left). We call this process "**Leaves to Root**":

1. Compute $\Psi_\omega^f \in \mathbb{R}^{3\times 3}$ and $\Psi_\omega^g \in \mathbb{R}^{3\times 3}$ on all leaves of $T_{\mathcal{T}_h}$ (triangles of $\mathcal{T}_h$) by (Eq. 3.2) and (Eq. 3.5).
2. Compute recursive from leaves to root Φ_ω and Ψ_ω from $\Psi_{\omega_1}, \Psi_{\omega_2}$. Store Φ_ω and delete $\Psi_{\omega_1}, \Psi_{\omega_2}$.
3. Stop if $\omega = \Omega$.

Remark 3.4 *The result of this algorithm will be a collection of mappings* $\{\Phi_\omega : \omega \in T_{\mathcal{T}_h}\}$. *The mappings* Ψ_ω, $\omega \in T_{\mathcal{T}_h}$, *are only of auxiliary purpose and need not stored.*

The algorithm which applies the mappings $\Phi_\omega = (\Phi_\omega^g, \Phi_\omega^f)$ to compute the solution we call "**Root to Leaves**". This algorithm starts from the root and ends on the leaves. Figure 2.2 (right) presents the scheme of this algorithm.
Let the data $d_\omega = (f_\omega, g_\omega)$, $\omega = \Omega$, be given. We can then compute the solution u_h of the initial problem as follows.
The Algorithm "**Root to Leaves**":

1. Start with $\omega = \Omega$.
2. Given $d_\omega = (f_\omega, g_\omega)$, compute the solution u_h on the interior boundary γ_ω by $\Phi_\omega(d_\omega)$.
3. Build the data $d_{\omega_1} = (f_{\omega_1}, g_{\omega_1})$, $d_{\omega_2} = (f_{\omega_2}, g_{\omega_2})$ from $d_\omega = (f_\omega, g_\omega)$ and $g_{\gamma_\omega} := \Phi_\omega(d_\omega)$.
4. Repeat the same for the sons of ω_1 and ω_2.
5. End if ω does not contain internal nodes.

Since $u_h(\mathbf{x}_i) = g_{\gamma,i}$, the set of values (g_{γ_ω}), for all $\omega \in T_{\mathcal{T}_h}$, results the solution of the initial problem (Eq. 1.1) in the whole domain Ω.

3.5 Multiple scales

Let h and H be fine and coarse meshes, used for discretization of Eq. 2.1. The subscript $_h$ near the operator or function means that this operator or function was discretized on a mesh with the step size h. Let n_h and n_H be the numbers of degrees of freedom on a fine grid and on a coarse grid. For instance, if the right-hand side is smooth, we may use a coarser mesh for it (e.g., operator $\mathcal{F}_H$ and $\mathcal{G}_h$). So, the matrices $\Psi^f and \Phi^f$ will be much smaller. For discretising the diffusion coefficient and the Dirichlet b.c. we use a fine scale h (see more in [31]).

Lemma 3.2 *The complexities of the one-grid version and two-grid version of HDD are*

$$\mathcal{O}(n_h \log^3 n_h) \quad and \quad \mathcal{O}(\sqrt{n_h n_H} \log^3 \sqrt{n_h n_H}), \quad respectively.$$

The storage requirements of the one-grid version and two-grid version of HDD are

$$\mathcal{O}(n_h \log^2 n_h) \quad and \quad \mathcal{O}(\sqrt{n_h n_H} \log^2 \sqrt{n_h n_H}), \quad respectively.$$

Proof: see [31, 7] or Ch. 12 in [21].

4 Hierarchical matrix approximation

The mappings Ψ_ω and Φ_ω correspond to dense matrices, and, therefore, require quadratic storage and quadratic or cubic arithmetic cost. Both these mappings (matrices) Ψ_ω (see Fig. 4.1) and Φ_ω (see Fig. 4.2) can be approximated in the $\mathcal{H}$-matrix format. Additionally, all necessary computational steps can be performed within the hierarchical matrix format with a log-linear cost.

The matrices $\Phi^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$, $\Phi^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\gamma_\omega)}$, $\Psi^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}^{I(\partial\omega)}$, are rectangular. The matrix $\Psi^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}^{I(\partial\omega)}$ is quadratic.

The hierarchical matrices ($\mathcal{H}$-matrices) have been used in a wide range of applications since their introduction in 1999 by Hackbusch [20]. They provide a format for the data-sparse representation of fully-populated matrices. The complexity of the $\mathcal{H}$-matrix addition, multiplication, Schur complement and inversion is $\mathcal{O}(k^2 n \log^q n)$, $q = 1, 2$. See more details about $\mathcal{H}$-matrices in [21, 20, 23, 17, 16, 38, 33]. In [30] authors prove the existence of an $\mathcal{H}$-matrix approximation of the inverse (Assumption 2) and of the Schur complement (Theorem 1).

The following proposition follows from Theorem 1 ([30]) and [24].

Proposition 4.1 *The matrices $\Psi^g_\omega \in \mathbb{R}^{I(\partial\omega)\times I(\partial\omega)}$, $\Psi^f_\omega \in \mathbb{R}^{I(\partial\omega)\times I(\omega)}$, $\Phi^f_\omega \in \mathbb{R}^{I(\partial\omega)\times I(\omega)}$ and $\Phi^g_\omega \in \mathbb{R}^{I(\gamma_\omega)\times I(\partial\omega)}$ for all $\omega \in T_{\mathcal{T}_h}$ can be effectively approximated by $\mathcal{H}$-matrices.*

For more details and complexity estimates see [31].

5 Fast Evaluation of Functionals

In this section we describe how to use Φ^f_ω and Φ^g_ω for building different linear functionals of the solution. Indeed, the functional λ is determined in the same way as Ψ_ω.
Below we list some examples of problem settings which can be solved by using linear functionals.

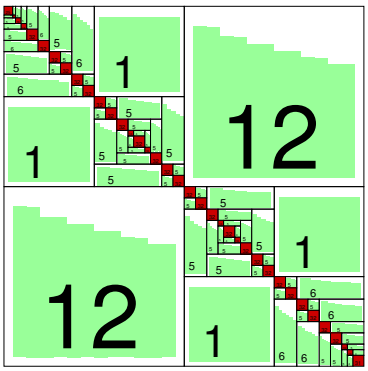

Figure 4.1: An $\mathcal{H}$-matrix approximation to $(\Psi_\omega^g)^{\mathcal{H}} \in \mathbb{R}^{I\times I}$, $I := I(\partial\omega)$. The dark (red) blocks are dense matrices and grey (green) blocks are low-rank matrices. The numbers inside the blocks indicate the ranks of these blocks.

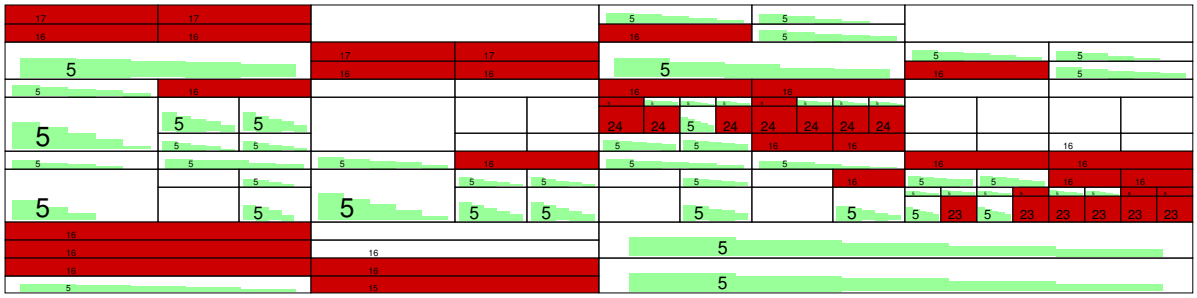

Figure 4.2: An $\mathcal{H}$-matrix approximation to $(\Psi_\omega^f)^{\mathcal{H}} \in \mathbb{R}^{I\times J}$, $I := I(\partial\omega)$, $J := J(\omega)$, $|I| = 256$, $|I| = 4225$. The dark (or red) blocks indicate the dense matrices and the grey (green) blocks indicate the rank-k matrices; the number inside each block is its rank. The steps inside the blocks show the decay of the singular values in log scale. The white blocks are empty.

Example 5.1 *If the solution u in a subdomain $\omega \in T_{\mathcal{T}_h}$ is known, the mean value $\mu(\omega)$ can be computed by the following formula*

$$\overline{u}_\omega := \mu(\omega) = \frac{\int_\omega u(\mathbf{x})d\mathbf{x}}{|\omega|} = \frac{\sum_{t\in\mathcal{T}_h(\omega)} \frac{|t|}{3}(u_1+u_2+u_3)}{|\omega|}, \tag{5.1}$$

where u is affine on each triangle t with values u_1, u_2, u_3 at the three corners and $\mathcal{T}_h(\omega)$ is the collection of all triangles in ω. If the solution u is unknown, we would like to have a linear functional $\lambda_\omega(f,g), \omega \in T_{\mathcal{T}_h}$, which computes the mean value μ_ω of the solution in ω.

Example 5.2 *Let us assume that Ω is decomposed into $p = 16$ subdomains $\Omega = \bigcup_{i=1}^{p} \Omega_i$. Sometimes these sub-domains Ω_i are called* `cells`*. The set of nodal points on the interface is denoted by I_Σ. HDD can compute the solution on the interface I_Σ and the mean value over each Ω_i, $i = 1, ..., p$, (see Fig. 5.1).*

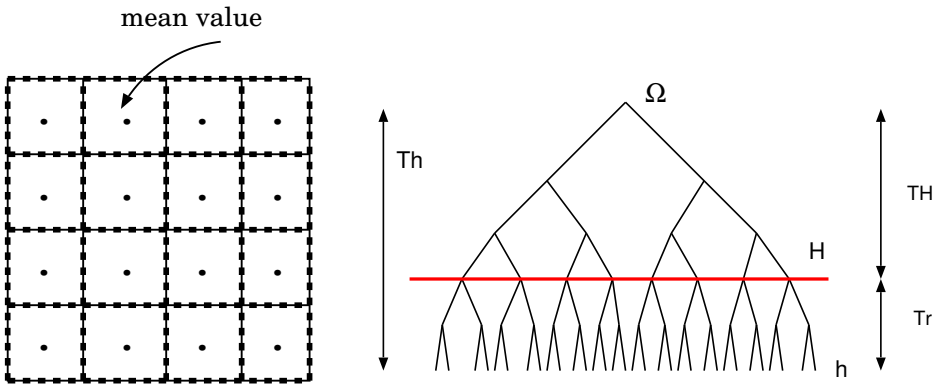

Figure 5.1: (left) HDD computes the solution on a coarse mesh (or the interface $I(\Sigma)$) and the mean value inside of each cell. (right) Algorithm "Leaves to Root" goes through the whole tree, but "Root to leaves" starts in the root, goes through a subtree $T_{\mathcal{T}_H}$ and terminates on a coarse level with the mesh size H (marked with the red horizontal line). After that the mean value inside of each coarse cell (of size $H \times H$) is computed.

Example 5.3 *To compute the FE solution $u_h(\mathbf{x}_i)$ in a fixed nodal point $\mathbf{x}_i \in \Omega$, i.e., to define how the solution $u_h(\mathbf{x}_i)$ depends on the given FE Dirichlet data $g_h \in \mathbb{R}^{I(\partial\Omega)}$ and the FE right-hand side $f_h \in \mathbb{R}^{I(\Omega)}$.*

5.1 Computing the mean value in all subdomains $\omega \in T_{\mathcal{T}_h}$

Let ω be the father node, ω_1 the left son, and ω_2 the right son. Then $\omega = \omega_1 \cup \omega_2$, $\omega_1 \cap \omega_2 \neq \varnothing$, with $\omega, \omega_1, \omega_2 \in T_{\mathcal{T}_h}$. To simplify the notation, we will write $d_i := d_{\omega_i}$ and (f_i, g_i) instead of $(f_{\omega_i}, g_{\omega_i})$, $i = 1, 2$ (see also Fig. 2.3). Recall the following notation (see (Eq. 3.16), (Eq. 3.17)):

$$\Gamma := \partial\omega, \quad \Gamma_{\omega,1} := \partial\omega \cap \omega_1, \quad \Gamma_{\omega,2} := \partial\omega \cap \omega_2, \quad \text{then}$$

$$d_1 = (f_1, g_1) = (f_1, g_{1,\Gamma}, g_{1,\gamma}), \quad d_2 = (f_2, g_2) = (f_2, g_{2,\Gamma}, g_{2,\gamma}), \text{ where} \tag{5.2}$$

$$\begin{aligned} g_{1,\Gamma} &:= (g_1)|_{\Gamma_{\omega,1}}, \quad g_{1,\gamma} := (g_1)|_{\gamma}, \\ g_{2,\Gamma} &:= (g_2)|_{\Gamma_{\omega,2}}, \quad g_{2,\gamma} := (g_2)|_{\gamma}. \end{aligned} \tag{5.3}$$

We consider a linear functional λ_ω with the following properties:

$$\lambda_\omega(d_\omega) = (\lambda^g_\omega, g_\omega) + (\lambda^f_\omega, f_\omega), \tag{5.4}$$

$$\lambda_\omega(d_\omega) = c_1\lambda_{\omega_1}(d_{\omega_1}) + c_2\lambda_{\omega_2}(d_{\omega_2}), \tag{5.5}$$

where $\lambda^g_\omega : \mathbb{R}^{I(\partial\omega)} \to \mathbb{R}$, $\lambda^f_\omega : \mathbb{R}^{I(\omega)} \to \mathbb{R}$, c_1, c_2 two constants, and $(\cdot,\cdot)$ the scalar product of two vectors.

Definition 5.1 *Let $\omega_1 \subset \omega$, $\lambda^f_{\omega_1} : \mathbb{R}^{I(\omega_1)} \to \mathbb{R}$. a) We define the following extension of $\lambda^f_{\omega_1}$*

$$(\lambda^f_{\omega_1}|^\omega)_i := \begin{cases} (\lambda^f_{\omega_1})_i & \text{for } i \in I(\omega_1), \\ 0 & \text{for } i \in I(\omega \setminus \omega_1), \end{cases}$$

where $(\lambda^f_{\omega_1}|^\omega) : \mathbb{R}^{I(\omega)} \to \mathbb{R}$. b) The extension of the functional $\lambda^g_{1,\Gamma} : \mathbb{R}^{I(\Gamma_{\omega,1})} \to \mathbb{R}$ is defined as

$$(\lambda^g_{1,\Gamma}|^\Gamma)_i := \begin{cases} (\lambda^g_{1,\Gamma})_i & \text{for } i \in I(\Gamma_{\omega,1}), \\ 0 & \text{for } i \in I(\Gamma \setminus \Gamma_{\omega,1}), \end{cases}$$

where $(\lambda^g_{1,\Gamma}|^\Gamma) : \mathbb{R}^{I(\Gamma)} \to \mathbb{R}$.

Definition 5.2 *Using (Eq. 5.3), we obtain the following decompositions $\lambda^g_{\omega_1} = (\lambda^g_{1,\Gamma}, \lambda^g_{1,\gamma})$ and $\lambda^g_{\omega_2} = (\lambda^g_{2,\Gamma}, \lambda^g_{2,\gamma})$, where $\lambda^g_{1,\Gamma} : \mathbb{R}^{I(\Gamma_{\omega,1})} \to \mathbb{R}$, $\lambda^g_{1,\gamma} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}$, $\lambda^g_{2,\Gamma} : \mathbb{R}^{I(\Gamma_{\omega,2})} \to \mathbb{R}$, $\lambda^g_{2,\gamma} : \mathbb{R}^{I(\gamma)} \to \mathbb{R}$.*

Lemma 5.4 *Let $\lambda_\omega(d_\omega)$ satisfy (Eq. 5.4) and (Eq. 5.5) with $\omega = \omega_1 \cup \omega_2$. Let $\lambda^g_{\omega_1}$, $\lambda^g_{\omega_2}$, $\lambda^f_{\omega_1}$ and $\lambda^f_{\omega_2}$ be the vectors for the representation of the functionals $\lambda_{\omega_1}(d_{\omega_1})$ and $\lambda_{\omega_2}(d_{\omega_2})$. Then the vectors λ^f_ω, λ^g_ω for the representation*

$$\lambda_\omega(d_\omega) = (\lambda^f_\omega, f_\omega) + (\lambda^g_\omega, g_\omega), \text{ where } f_\omega \in \mathbb{R}^{I(\omega)}, g_\omega \in \mathbb{R}^{I(\partial\omega)}, \tag{5.6}$$

are given by

$$\lambda^f_\omega = \tilde{\lambda}^f_\omega + (\Phi^f_\omega)^T\lambda^g_\gamma,$$

$$\lambda^g_\omega = \tilde{\lambda}^g_\omega + (\Phi^g_\omega)^T\lambda^g_\gamma,$$

$$\tilde{\lambda}^f_\omega := c_1\lambda^f_{\omega_1}|^\omega + c_2\lambda^f_{\omega_2}|^\omega, \tag{5.7}$$

$$\tilde{\lambda}^g_\Gamma := c_1\lambda^g_{1,\Gamma}|^\Gamma + c_2\lambda^g_{2,\Gamma}|^\Gamma, \tag{5.8}$$

$$\lambda^g_\gamma = c_1\lambda^g_{1,\gamma} + c_2\lambda^g_{2,\gamma}. \tag{5.9}$$

Proof: Let d_{ω_1}, d_{ω_2} be the given data and λ_{ω_1} and λ_{ω_2} be the given functionals. Then the functional λ_ω satisfies

$$\begin{aligned}\lambda_\omega(d_\omega) &\overset{(Eq.\ 5.5)}{=} c_1\lambda_{\omega_1}(d_{\omega_1}) + c_2\lambda_{\omega_2}(d_{\omega_2})\\ &\overset{(Eq.\ 5.4)}{=} c_1((\lambda^f_{\omega_1}, f_1) + (\lambda^g_{\omega_1}, g_1)) + c_2((\lambda^f_{\omega_2}, f_2) + (\lambda^g_{\omega_2}, g_2)).\end{aligned}$$

Using the decomposition (Eq. 5.2), we obtain

$$\lambda_\omega(d_\omega) = c_1(\lambda^f_{\omega_1}, f_1) + c_2(\lambda^f_{\omega_2}, f_2) + c_1((\lambda^g_{1,\Gamma}, g_{1,\Gamma}) + (\lambda^g_{1,\gamma}, g_{1,\gamma})) \tag{5.10}$$

$$+ c_2((\lambda^g_{2,\Gamma}, g_{2,\Gamma}) + (\lambda^g_{2,\gamma}, g_{2,\gamma})). \tag{5.11}$$

The consistency of the solution implies $g_{1,\gamma} = g_{2,\gamma} =: g_\gamma$. From the Definition 5.1 follows

$$(\lambda^f_{\omega_1}, f_1) = (\lambda^f_{\omega_1}|^\omega, f_\omega), \quad (\lambda^f_{\omega_2}, f_2) = (\lambda^f_{\omega_2}|^\omega, f_\omega),$$

$$(\lambda^g_{1,\Gamma}, g_{1,\Gamma}) = (\lambda^g_{1,\Gamma}|^\Gamma, g_\omega), \quad (\lambda^g_{2,\Gamma}, g_{2,\Gamma}) = (\lambda^g_{2,\Gamma}|^\Gamma, g_\omega).$$

Then, we substitute the last expressions in (Eq. 5.10) to obtain

$$\lambda_\omega(d_\omega) = (c_1\lambda^f_{\omega_1}|^\omega + c_2\lambda^f_{\omega_2}|^\omega, f_\omega) + (c_1\lambda^g_{1,\Gamma}|^\Gamma + c_2\lambda^g_{2,\Gamma}|^\Gamma, g_\omega) \tag{5.12}$$

$$+(c_1\lambda^g_{1,\gamma} + c_2\lambda^g_{2,\gamma}, g_\gamma).$$

Set $\tilde\lambda^f_\omega := c_1\lambda^f_{\omega_1}|^\omega + c_2\lambda^f_{\omega_2}|^\omega$, $\tilde\lambda^g_\Gamma := c_1\lambda^g_{1,\Gamma}|^\Gamma + c_2\lambda^g_{2,\Gamma}|^\Gamma$and $\lambda^g_\gamma := c_1\lambda^g_{1,\gamma} + c_2\lambda^g_{2,\gamma}$.
From the algorithm "Root to Leaves" we know that

$$g_\gamma = \Phi_\omega(d_\omega) = \Phi^g_\omega \cdot g_\omega + \Phi^f_\omega \cdot f_\omega. \tag{5.13}$$

Substituting g_γ from (Eq. 5.13) in (Eq. 5.12), we obtain

$$\begin{aligned}\lambda_\omega(d_\omega) &= (\tilde\lambda^f_\omega, f_\omega) + (\tilde\lambda^g_\omega, g_\omega) + (\lambda^g_\gamma, \Phi^g_\omega g_\omega + \Phi^f_\omega f_\omega)\\ &= (\tilde\lambda^f_\omega + (\Phi^f_\omega)^T\lambda^g_\gamma, f_\omega) + (\tilde\lambda^g_\omega + (\Phi^g_\omega)^T\lambda^g_\gamma, g_\omega).\end{aligned}$$

We define $\lambda^f_\omega := \tilde\lambda^f_\omega + (\Phi^f_\omega)^T\lambda^g_\gamma$ and $\lambda^g_\omega := \tilde\lambda^g_\omega + (\Phi^g_\omega)^T\lambda^g_\gamma$ and obtain

$$\lambda_\omega(d_\omega) = (\lambda^f_\omega, f_\omega) + (\lambda^g_\omega, g_\omega). \tag{5.14}$$

■

Example 5.5 *Lemma 5.4 with* $c_1 = \frac{|\omega_1|}{|\omega|}$, $c_2 = \frac{|\omega_2|}{|\omega|}$ *can be used to compute the mean values in all subdomains* $\omega \in T_{\mathcal{T}_h}$.

5.2 Algorithms for computing the mean values

Below we describe two algorithms which are required for computing the mean value in all subdomains $\omega \in T_{\mathcal{T}_h}$. These algorithms compute vectors λ^g_ω and λ^f_ω respectively.

The initialisation is $\lambda^g_\omega := (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $\lambda^f_\omega := (0, 0, 0)$ for all leaves of $T_{\mathcal{T}_h}$. Let us denote $\lambda^g_1 := \lambda^g_{\omega_1}$, $\lambda^g_2 := \lambda^g_{\omega_2}$. The algorithms for building λ^g_ω and λ^f_ω for all $\omega \in T_{\mathcal{T}_h}$, which have internal nodes, are the following:

Algorithm 5.1 *(Building of λ^g_ω)*
build_functional_g$(\lambda^g_1,\ \lambda^g_2,\ \Phi^g_\omega,\ldots)$
begin
allocate memory for λ^g_ω;
for all $i \in I(\Gamma_{\omega,1})$ ***do***
$\lambda^g_\omega[i] += c_1\lambda^g_1[i]$;
for all $i \in I(\Gamma_{\omega,2})$ ***do***
$\lambda^g_\omega[i] += c_2\lambda^g_2[i]$;
for all $i \in I(\gamma)$ ***do***
$z[i] = c_1\lambda^g_1[i] + c_2\lambda^g_2[i]$;
$v := (\Phi^g_\omega)^T \cdot z$;
for all $i \in I(\partial\omega)$ ***do***
$\lambda^g_\omega[i] := \lambda^g_\omega[i] + v[i]$;
return λ^g_ω;
end;

Let $\lambda^f_1 := \lambda^f_{\omega_1}$, $\lambda^f_2 := \lambda^f_{\omega_2}$.

Algorithm 5.2 *(Building of λ^f_ω)*
build_functional_f$(\lambda^f_1,\ \lambda^f_2,\ \Phi^f_\omega,\ldots)$
begin
for all $i \in I(\omega_1\backslash\gamma)$ ***do***
$\lambda^f_\omega[i] += c_1\lambda^f_1[i]$;
for all $i \in I(\omega_2\backslash\gamma)$ ***do***
$\lambda^f_\omega[i] += c_2\lambda^f_2[i]$;
for all $i \in I(\gamma)$ ***do***
$z[i] = c_1\lambda^f_1[i] + c_2\lambda^f_2[i]$;
$v := (\Phi^f_\omega)^T \cdot z$;
for all $i \in I(\omega)$ ***do***
$\lambda^f_\omega[i] := \lambda^f_\omega[i] + v[i]$;
return λ^f_ω;
end;

Remark 5.1 *a) If only the functionals λ_ω, $\omega \in T_{\mathcal{T}_h}$, are of interest, the maps Φ_ω do not need to be stored.*
b) For functionals with local support in some $\omega_0 \in T_{\mathcal{T}_h}$, it suffices that Φ_ω is given for all $\omega \in T_{\mathcal{T}_h}$ with $\omega \supset \omega_0$, while $\lambda_{\omega_0}(d_{\omega_0})$ is associated with $\omega_0 \in T_{\mathcal{T}_h}$. The computation of $\Lambda(u_h) = \lambda(d)$ starts with the recursive evaluation of Φ_ω for all $\omega \supset \omega_0$. Then the data d_{ω_0} are available and λ_{ω_0} can be applied.

5.3 Solution in a subdomain

Suppose that the solution is only required in a small subdomain $\omega \in T_{\mathcal{T}_h}$ (Fig. 5.2, left). For this purpose, the HDD method requires less computational resources as usual. The algorithm "Leaves to Root" is performed completely, but the algorithm "Root to Leaves" computes the solution only on the internal boundaries (dotted lines) which are necessary for

computing the solution in ω. The storage requirements are also significantly reduced. We only store the mappings Φ^f_ω and Φ^g_ω for all $\omega \in T_{\mathcal{T}_h}$ that belong to the path from the root of $T_{\mathcal{T}_h}$ to ω. The storage requirement is $\mathcal{O}(n_h \log n_h)$, where n_h is the number of degrees of freedom in Ω. The computational cost of the "Root to Leaves" is $\mathcal{O}(n_h \log^2 n_h)$. If the right-hand side is smooth, it can be discretized (defined) only on a coarse mesh $\mathcal{T}_H$ (see Fig. 5.2, right). About the interpolation and restriction operators read in [31, 22, 7, 21].

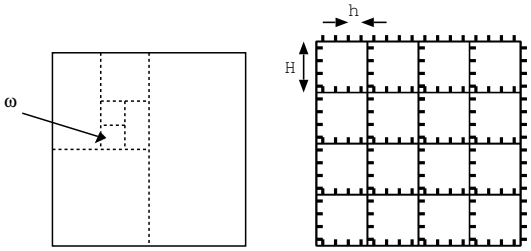

Figure 5.2: (left) The solution in a subdomain $\omega \in T_{\mathcal{T}_h}$ is required. HDD computes subsequently the solution only on the dotted lines and then only in ω; (right) The coarse H and the fine h scales.

6 Numerics

In [31, 7, 22], the HDD method was compared with the preconditioned conjugate gradient (PCG) method, with the hierarchical ($\mathcal{H}$)- Cholesky method and the direct full $\mathcal{H}$-matrix inverse. A cheap $\mathcal{H}$-matrix approximation of the inverse, computed from the $\mathcal{H}$-Cholesky factors, was used as a preconditioner.

Some experiments were performed with two meshes - a coarse for the right-hand side and a fine for the diffusion coefficient. Technical details and implementation of the HDD method can be found in [32, 7, 31]. The data misfit and the likelihood function in the Bayesian-like approach were computed in [58, 56, 45, 46, 53, 57]. In the following numerical experiments we compare the computational time and memory requirement of HDD with the times and memory requirements of the $\mathcal{H}$-matrix inverse and the $\mathcal{H}$-Cholesky factorisation.

We consider the problem as in Eq. 1.1 with fixed Z. The computational domain is a unit square $\Omega = [0,1]^2$, the diffusion coefficient is $\kappa(x,y) = 1 + 0.5 \cdot \sin(50x)\sin(50y)$. Note, that HDD does not require an axes parallel triangulation. All numerical experiments were performed on a usual notebook. Figure 6.1 demonstrates the dependence of computing times (left) and memory requirements (right) on the $\mathcal{H}$-matrix accuracy (see the adaptive rank arithmetic in [21]). One can see that the computational time and storage requirement of the $\mathcal{H}$-Cholesky factorisation are the best. The HDD method shows a slightly larger time and storage than the $\mathcal{H}$-Cholesky factorisation (due to some overhead) and is better than the direct $\mathcal{H}$-matrix inverse. Note that HDD computes more details about the operator and the solution than the $\mathcal{H}$-Cholesky factorisation.

If the right-hand side is smooth, we can discretise it on a mesh with the mesh size, for instance, $H = 2h$. As result, the matrices Φ^g and Ψ^g will stay the same, but Φ^f and Ψ^f will be smaller. See more for the restriction and prolongation operators in [31].

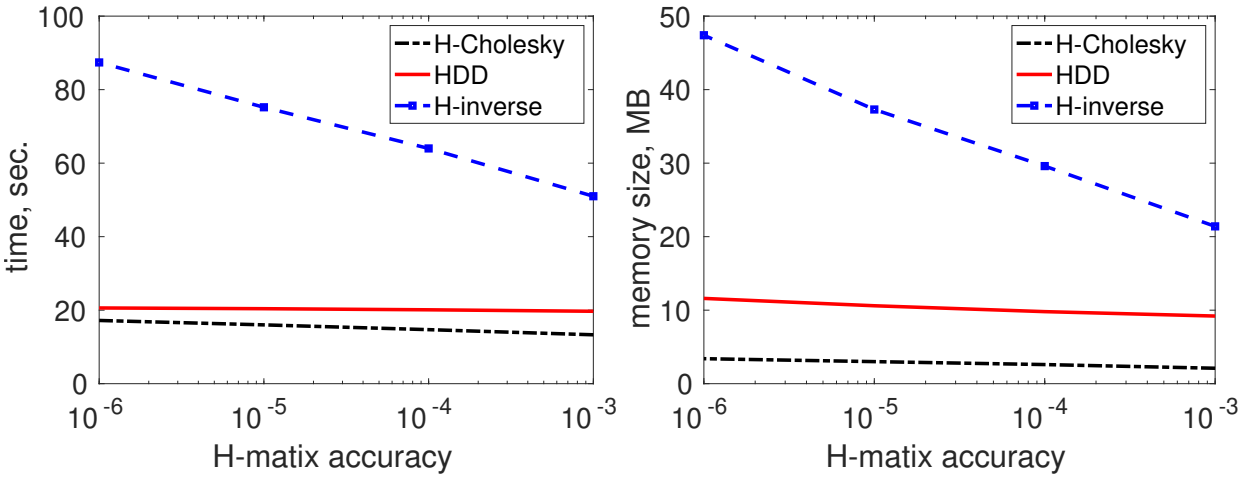

Figure 6.1: Comparison of the HDD, $\mathcal{H}$-Cholesky factorisation, and $\mathcal{H}$-matrix inverse. (left) Dependence of the computing time (in sec.) and (right) memory requirements (in MB) on the $\mathcal{H}$-matrix accuracy, $n = 129^2$ dofs.

In the next example we consider again the problem as in Eq. 1.1. The parameter Z is fixed and κ is a jumping coefficient as in Fig. 6.2 with $\alpha = 10^{-5}$ and $\beta = 1$. Such problems appear in the material sciences and in medicine (the, so-called, skin problem).

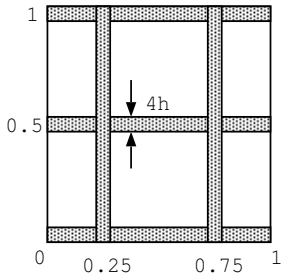

Figure 6.2: Model domain $\Omega = [0, 1]^2$. The diffusion coefficient is very small ($\alpha = 10^{-5}$) inside the grey areas and large in white subdomains ($\beta = 1$).

Figure 6.3 compares the computational times of HDD and PCG methods. The PCG time includes the time needed for: (a) computing the stiffness matrix A in the $\mathcal{H}$-matrix format; (b) computing the $\mathcal{H}$-Cholesky decomposition of A (used as a preconditioner); (c) PCG iterations.
For example, for $n = 66049$, the PCG time is $53 = 38.2 + 11.4 + 3.4$ (sec.). Note that for $n \approx 263000$ dofs there is not enough memory to compute the stiffness matrix A and perform its $\mathcal{H}$-Cholesky factorization. The advantage of the HDD method is that it does not require an agglomeration of the whole stiffness matrix. The memory is dynamically allocated and deallocated.

In the next example we take a coarse mesh for the right-hand side with the grid step size $H = 2h$. n Figure 6.4(left) we visualise the difference $\|\tilde{\boldsymbol{u}}_{cg} - \tilde{\boldsymbol{u}}\|$ in the Frobenius and infinity norms. Here $\tilde{\boldsymbol{u}}$ is the HDD solution and $\tilde{\boldsymbol{u}}_{cg}$ the solution obtained by the PCG method with the $\mathcal{H}$-Cholesky preconditioner. The corresponding HDD and PCG times are compared in

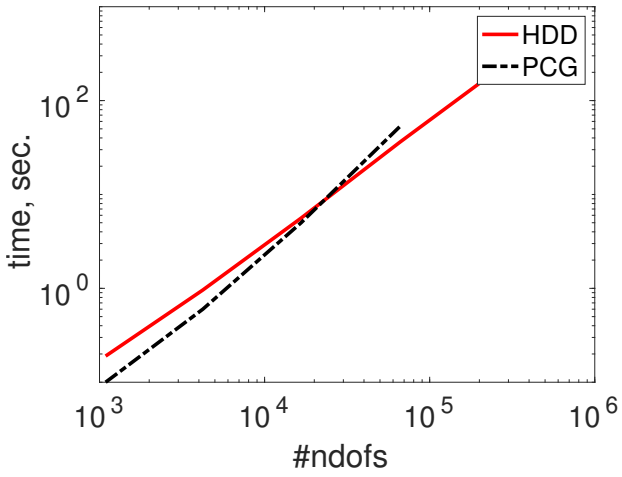

Figure 6.3: HDD and PCG computing times vs. n. The accuracy in each $\mathcal{H}$-matrix subblock is 10^{-8}, the PCG stopping criteria $\varepsilon_{cg} = 10^{-8}$, $\frac{H}{h} = 2$ as in Sec. 3.5.

Fig. 6.4(right). The accuracy inside of each $\mathcal{H}$-matrix subblock is 10^{-5}. Note, that PCG requires too much memory for $n = 513^2$ dofs and we were not able to compute $\tilde{\boldsymbol{u}}_{cg}$.

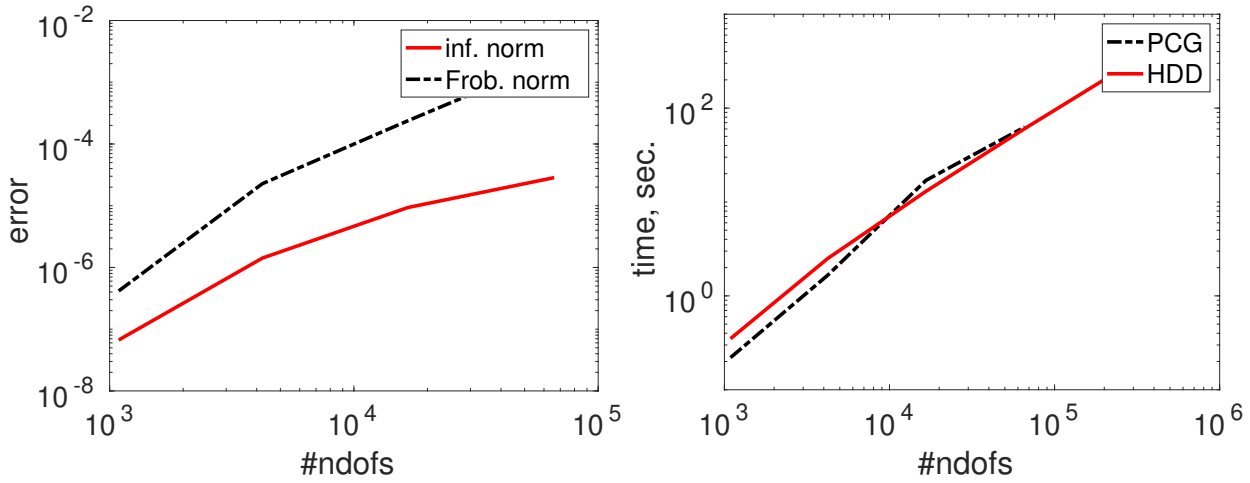

Figure 6.4: Dependence of the absolute errors on the number of dofs, $f = 1$, $\alpha(x, y) = 1/(1.0001 + \sin(500x)\sin(500y))$. $\mathcal{H}$-matrix accuracy $\varepsilon = 10^{-5}$, $\frac{H}{h} = 2$. (left) errors (left) dependence of the errors $\|\tilde{\boldsymbol{u}}_{cg} - \tilde{\boldsymbol{u}}\|_2$ and $\|\tilde{\boldsymbol{u}}_{cg} - \tilde{\boldsymbol{u}}\|_\infty$ on n; (right) computing times PCG and HDD vs. n.

Figure 6.5 shows the total storage requirement for all matrices Φ^g_ω and Φ^f_ω, $\omega \in T_{\mathcal{T}_h}$. We see an almost linear dependence on n.

7 Conclusion and discussion

We suggested another useful application of the already known HDD method. Namely, HDD speeds up computations of the data misfit (mismatch) in the likelihood function in the

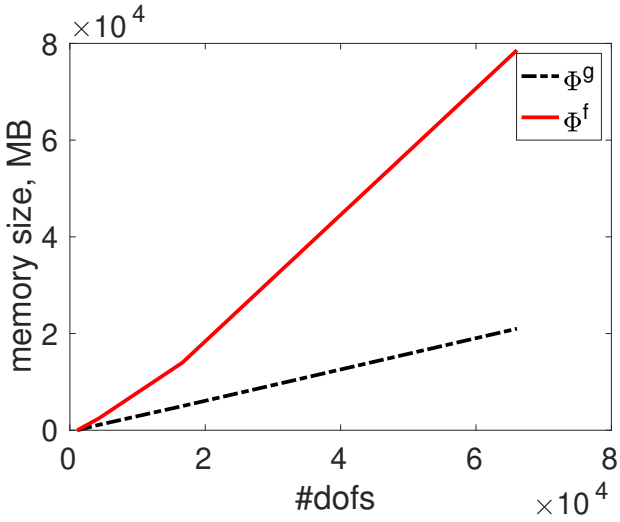

Figure 6.5: Dependence of the total memory requirement for all Φ^g_ω and Φ^f_ω on n, the maximal $\mathcal{H}$-matrix rank is $k = 7$.

Bayesian approach. HDD can also be used when the simulated data and measurement data are compared (e.g., in regression, parameter inference, data assimilation, Kalman filter, and Bayesian update problems). HDD uses the fact that often only a functional of the solution or a small part of it is observed or measured. Therefore, HDD computes only a part of the inverse operator and only a part of the solution. Optimally, HDD computes only what is needed, i.e., what is measured.

As such the computational accuracy is as usual (for instance, as in the standard FEM method), but the computational recourses (FLOPS and storage) are smaller. The HDD method is based on the hierarchical (recursive) domain decomposition, FEM, and the Schur complement methods. If the forward operator and the right-hand side can be discretized on different meshes, which is often the case in multiscale problems, the HDD method can get significant advantages. The computational resources will be reduced even more.

Additionally, to speed up the Schur complement computations, we approximate all intermediate and auxiliary matrices in the $\mathcal{H}$-matrix format. We then achieve the computational cost $\mathcal{O}(n \log^3 n)$ and the storage $\mathcal{O}(n \log^2 n)$. There is some overhead due to the construction of the hierarchical decomposition tree $T_{\mathcal{T}_h}$ and permutation of indices.

To apply the HDD method, the user should have a possibility to 1) modify the assembling procedure of the stiffness matrix; 2) build the hierarchical domain decomposition tree.

We note that the interface size in a d-dimensional problem is $\mathcal{O}(n^{d-1})$. So in a 2D case, the interface is $\mathcal{O}(n)$, whereas, in 3D problems, the interface is $\mathcal{O}(n^2)$. This fact results in increasing matrix sizes. The structure and the cost of the $\mathcal{H}$-matrix arithmetics become more expensive too.

Numerical tests showed that HDD requires more computational resources than PCG with the $\mathcal{H}$-Cholesky preconditioner, and less resources than the direct $\mathcal{H}$-matrix inverse. But the HDD method computes more details than PCG. It computes the solution operators Φ^g and Φ^f on each level of the hierarchical domain decomposition tree. These can be used later to compute a functional of the solution, or solution on different scales, in a sub-domain, in a point or on an interface.

The HDD method can be coupled with more uncertainty quantification and parameter inference techniques. Potentially interesting could be the coupling with the Multi-Level Monte Carlo method.

Acknowledgments

This work was supported by funding from the Alexander von Humboldt foundation (chair of Mathematics for Uncertainty Quantification at RWTH Aachen).

References

[1] Tracy Babb, Adrianna Gillman, Sijia Hao, and Per-Gunnar Martinsson. An accelerated poisson solver based on multidomain spectral discretization. *BIT Numerical Mathematics*, 58(4):851–879, 2018.

[2] I. Babuka, R. Tempone, and G.E. Zouraris. Galerkin finite element approximations of stochastic elliptic partial differential equations. *SIAM Journal on Numerical Analysis*, 42(2):800–825, 2004.

[3] R.D. Berry, H. N. Najm, B.J. Debusschere, H. Adalsteinsson, and Y.M. Marzouk. Data-free inference of the joint distribution of uncertain model parameters. *Journal of Computational Physics*, 231:2180–2198, 2012.

[4] H.-J. Bungartz and M. Griebel. Sparse grids. *Acta Numer.*, 13:147–269, 2004.

[5] S. Dolgov, B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Computation of the response surface in the tensor train data format. *arXiv preprint arXiv:1406.2816*, 2014.

[6] S. Dolgov, B. N. Khoromskij, A. Litvinenko, and H. G. Matthies. Polynomial chaos expansion of random coefficients and the solution of stochastic partial differential equations in the tensor train format. *IAM/ASA J. Uncertainty Quantification*, 3(1):1109–1135, 2015.

[7] F. Drechsler. *Über die Lösung von elliptischen Randwertproblemen mittels Gebietszerlegungstechniken, Hierarchischer Matrizen und der Methode der finiten Elemente*. PhD thesis, PhD thesis, Universitaet Leipzig, Germany, 2016.

[8] T. A. El Moselhy and Y. Marzouk. Bayesian inference with optimal maps. *Journal of Computational Physics*, 231(23):7815–7850, 2012.

[9] M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies, and Ph. Wähnert. Efficient low-rank approximation of the stochastic galerkin matrix in tensor formats. *Computers and Mathematics with Applications*, 67(4):818–829, 2014.

[10] M. Espig, W. Hackbusch, A. Litvinenko, H. G. Matthies, and E. Zander. Efficient analysis of high dimensional data in tensor formats. In *Sparse Grids and Applications*, pages 31–56. Springer, 2013.

[11] J. Galvis and M. Sarkis. Approximating infinity-dimensional stochastic darcy's equations without uniform ellipticity. *SIAM Journal on Numerical Analysis*, 47(5):3624–3651, 2009.

[12] A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numer. Anal.*, 10:345–363, 1973. Collection of articles dedicated to the memory of George E. Forsythe.

[13] C. J. Gittelson. Stochastic Galerkin discretization of the log-normal isotropic diffusion problem. *Mathematical Models and Methods in Applied Sciences*, 20(02):237–263, 2010.

[14] M. Goldstein and D. Wooff. *Bayes Linear Statistics*, volume 160 of *Wiley Series in Probability and Statistics*. Wiley, Chichester, UK, 2007.

[15] I. G. Graham, F. Y. Kuo, D. Nuyens, R. Scheichl, and I.H. Sloan. Quasi-Monte Carlo methods for elliptic PDEs with random coefficients and applications. *J. Comput. Phys.*, 230(10):3668–3694, 2011.

[16] L. Grasedyck. Theorie und anwendungen hierarchischer matrizen. *Ph.D. Thesis, University of Kiel, Germany*, 2001.

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

[18] M. Griebel. Sparse grids and related approximation schemes for higher dimensional problems. In *Foundations of computational mathematics, Santander 2005*, volume 331 of *London Math. Soc. Lecture Note Ser.*, pages 106–161. Cambridge Univ. Press, Cambridge, 2006.

[19] W. Hackbusch. *Elliptic differential equations*, volume 18 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1992. Theory and numerical treatment, Translated from the author's revision of the 1986 German original by Regine Fadiman and Patrick D. F. Ion.

[20] W. Hackbusch. A sparse matrix arithmetic based on $\mathcal{H}$-matrices. I. Introduction to $\mathcal{H}$-matrices. *Computing*, 62(2):89–108, 1999.

[21] W. Hackbusch. *Hierarchical Matrices: Algorithms and Analysis*. Springer Series in Computational Mathematics, Volume 49. Springer, 2015.

[22] W. Hackbusch and F. Drechsler. Partial evaluation of the discrete solution of elliptic boundary value problems. *Computing and Visualization in Science*, 15(5):227–245, Oct 2012.

[23] W. Hackbusch and B. N. Khoromskij. A sparse $\mathcal{H}$-matrix arithmetic. II. Application to multi-dimensional problems. *Computing*, 64(1):21–47, 2000.

[24] W. Hackbusch, B. N. Khoromskij, and R. Kriemann. Hierarchical matrices based on a weak admissibility criterion. *Computing*, 73(3):207–243, 2004.

[25] B. N. Khoromskij and A. Litvinenko. Data sparse computation of the Karhunen-Loève expansion. In *AIP Conference Proceedings*, volume 1048(1), pages 311–314. AIP, 2008.

[26] B. N. Khoromskij and Ch. Schwab. Tensor-structured Galerkin approximation of parametric and stochastic elliptic PDEs. *SIAM J. of Sci. Comp.*, 33(1):1–25, 2011.

[27] D. Kressner and Ch. Tobler. Low-rank tensor Krylov subspace methods for parametrized linear systems. *SIAM J. Matrix Anal. Appl.*, 32(4):1288–1316, 2011.

[28] D. Kressner and Ch. Tobler. Preconditioned low-rank methods for high-dimensional elliptic PDE eigenvalue problems. *Comput. Methods Appl. Math.*, 11(3):363–381, 2011.

[29] F. Y. Kuo, Ch. Schwab, and I. H. Sloan. Multi-level quasi-Monte Carlo finite element methods for a class of elliptic pdes with random coefficients. *Foundations of Computational Mathematics*, pages 1–39, 2015.

[30] S. Le Borne, L. Grasedyck, and R. Kriemann. Parallel black box domain decomposition based $\mathcal{H}-lu$ preconditioning. *Max-Planck-Institut MIS, Leipzig, www.mis.mpg.de*, Preprint 115:(electronic), 2005.

[31] A. Litvinenko. Application of hierarchical matrices for solving multiscale problems. *PhD Dissertation, Leipzig University, Germany, https://publications.rwth-aachen.de/record/754296*, 2006.

[32] A. Litvinenko. Documentation for the HDD method. *Technical report in Max-Planck-Institut MIS, Leipzig, Germany, www.mis.mpg.de/preprints/tr/index.html*, 5, 2006.

[33] A. Litvinenko, R. Kriemann, M. G. Genton, Y. Sun, and D. E. Keyes. Hlibcov: Parallel hierarchical matrix approximation of large covariance matrices and likelihoods with applications in parameter identification. *MethodsX*, 7:100600, 2020.

[34] A. Litvinenko and H. G. Matthies. Inverse problems and uncertainty quantification. *arXiv preprint arXiv:1312.5048*, 2013.

[35] A. Litvinenko and H. G. Matthies. Numerical methods for uncertainty quantification and bayesian update in aerodynamics. In *Management and Minimisation of Uncertainties and Errors in Numerical Aerodynamics*, pages 265–282. Springer Berlin Heidelberg, 2013.

[36] A. Litvinenko and H. G. Matthies. Uncertainty quantification and non-linear bayesian update of pce coefficients. *PAMM*, 13(1):379–380, 2013.

[37] A. Litvinenko, H.G. Matthies, and T. A. El-Moselhy. Sampling and low-rank tensor approximation of the response surface. In Josef Dick, Frances Y. Kuo, Gareth W. Peters, and Ian H. Sloan, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2012*, volume 65 of *Springer Proceedings in Mathematics & Statistics*, pages 535–551. Springer Berlin Heidelberg, 2013.

[38] A. Litvinenko, Y. Sun, M. G. Genton, and D. E. Keyes. Likelihood approximation with hierarchical matrices for large spatial datasets. *Computational Statistics & Data Analysis*, 137:115 – 132, 2019.

[39] P.-G. Martinsson. The hierarchical poincaré-steklov (hps) solver for elliptic pdes: A tutorial. *arXiv preprint arXiv:1506.01308*, 2015.

[40] Y. Marzouk, H. Najm, and L. Rahn. Stochastic spectral methods for efficient Bayesian solution of inverse problems. *Journal of Computational Physics*, 224(2):560–586, June 2007.

[41] Y. Marzouk and D. Xiu. A stochastic collocation approach to bayesian inference in inverse problems. *Communications in Computational Physics*, 6(4):826–847, 10 2009.

[42] Y. M. Marzouk and H. N. Najm. Dimensionality reduction and polynomial chaos acceleration of Bayesian inference in inverse problems. *Journal of Computational Physics*, 228(6):1862–1902, 2009.

[43] H. G. Matthies and A. Keese. Galerkin methods for linear and nonlinear elliptic stochastic partial differential equations. *Computer Methods in Applied Mechanics and Engineering*, 194(12-16):1295–1331, 2005.

[44] H. G. Matthies and E. Zander. Solving stochastic systems with low-rank tensor compression. *Linear Algebra and its Applications*, 436(10):3819–3838, 2012.

[45] H. G. Matthies, E. Zander, O. Pajonk, B. V. Rosić, and A. Litvinenko. Inverse problems in a Bayesian setting. In *Computational Methods for Solids and Fluids Multiscale Analysis, Probability Aspects and Model Reduction Editors: Ibrahimbegovic, Adnan (Ed.), ISSN: 1871-3033*, pages 245–286. Springer, 2016.

[46] H. G. Matthies, E. Zander, B. V. Rosić, and A. Litvinenko. Parameter estimation via conditional expectation: a Bayesian inversion. *Advanced Modeling and Simulation in Engineering Sciences*, 3(1):24, 2016.

[47] H.G. Matthies, A. Litvinenko, O. Pajonk, B. V. Rosić, and E. Zander. Parametric and uncertainty computations with tensor product representations. In Andrew M. Dienstfrey and Ronald F. Boisvert, editors, *Uncertainty Quantification in Scientific Computing*, volume 377 of *IFIP Advances in Information and Communication Technology*, pages 139–150. Springer Berlin Heidelberg, 2012.

[48] A. Mugler and H.-J. Starkloff. On elliptic partial differential equations with random coefficients. *Stud. Univ. Babes-Bolyai Math*, 56(2):473–487, 2011.

[49] H. N. Najm, B.J. Debusschere, Y.M. Marzouk, S. Widmer, and O. P. Le Maître. Uncertainty Quantification in Chemical Systems. *Int. J. Num. Meth. Eng.*, 80:789–814, 2009.

[50] F. Nobile, R. Tempone, and C. G. Webster. A sparse grid stochastic collocation method for partial differential equations with random input data. *SIAM Journal on Numerical Analysis*, 46(5):2309–2345, 2008.

[51] A. Nouy. A generalized spectral decomposition technique to solve a class of linear stochastic partial differential equations. *Comput. Methods Appl. Mech. Engrg.*, 196(45-48):4521–4537, 2007.

[52] A. Nouy. Proper generalized decompositions and separated representations for the numerical solution of high dimensional stochastic problems. *Archives of Computational Methods in Engineering*, 17(4):403–434, 2010.

[53] O. Pajonk, B. V. Rosić, A. Litvinenko, and H. G. Matthies. A deterministic filter for non-Gaussian Bayesian estimation — applications to dynamical system estimation with noisy measurements. *Physica D: Nonlinear Phenomena*, 241(7):775–788, 2012.

[54] M. Parno, T. Moselhy, and Y. Marzouk. A multiscale strategy for bayesian inference using transport maps. *SIAM/ASA Journal on Uncertainty Quantification*, 4(1):1160–1190, 2016.

[55] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[56] B. V. Rosić, A. Kucerová, J. Sỳkora, O. Pajonk, A. Litvinenko, and H. G. Matthies. Parameter identification in a probabilistic setting. *Engineering Structures*, 50:179–196, 2013.

[57] B. V. Rosić, A. Litvinenko, O. Pajonk, and H. G. Matthies. Direct Bayesian update of polynomial chaos representations. *Journal of Computational Physics*, 2011.

[58] B. V. Rosić, A. Litvinenko, O. Pajonk, and H. G. Matthies. Sampling-free linear Bayesian update of polynomial chaos representations. *Journal of Computational Physics*, 231(17):5761–5787, 2012.

[59] B. Sousedík and R. Ghanem. Truncated hierarchical preconditioning for the stochastic Galerkin FEM. *International Journal for Uncertainty Quantification*, 4(4):333–348, 2014.

[60] A. Spantini, T. Cui, K. Willcox, L. Tenorio, and Y. Marzouk. Goal-oriented optimal approximations of bayesian linear inverse problems. *arXiv preprint arXiv:1607.01881*, 2016.

[61] A. M. Stuart. Inverse problems: a Bayesian perspective. *Acta Numerica*, 19:451–559, 2010.

[62] A. Tarantola. *Inverse Problem Theory and Methods for Model Parameter Estimation*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005.

[63] A. L. Teckentrup, R. Scheichl, M.B. Giles, and E. Ullmann. Further analysis of multilevel Monte Carlo methods for elliptic PDEs with random coefficients. *Numer. Math.*, 125(3):569–600, 2013.

[64] E. K. Zander. *Tensor Approximation Methods for Stochastic Problems*. PhD thesis, Dissertation, Technische Universität Braunschweig, ISBN: 978-3-8440-1871-4, 2013.

A Appendix A

Example A.1 *Figure A.1 shows an example of building* $(\Psi^g_\omega)^{\mathcal{H}} \in \mathbb{R}^{512\times512}$ *from* $(\Psi^g_{\omega_1})^{\mathcal{H}} \in \mathbb{R}^{384\times384}$ *and* $(\Psi^g_{\omega_2})^{\mathcal{H}} \in \mathbb{R}^{384\times384}$. *Let* $I := I(\partial\omega \cup \gamma)$. *The construction is performed in three steps:*
1) embed matrix $(\Psi^g_{\omega_1})^{\mathcal{H}}$ *into a larger matrix* $H' := (\Psi^g_{\omega_1})^{\mathcal{H}}|^{I\times I}$ *and* $(\Psi^g_{\omega_2})^{\mathcal{H}}$ *into* $H'' := (\Psi^g_{\omega_2})^{\mathcal{H}}|^{I\times I}$,
2) since H' *and* H'' *have the same* $\mathcal{H}$*-matrix format, compute the sum* $\tilde{H} = H' \oplus H''$,
3) compute the Schur complement and eliminate the block (2,2) of size $I(\gamma) \times I(\gamma)$.
Note that H', H'', $\tilde{H}$ *have the same block structures. The symmetries of* $(\Psi^g_{\omega_1})^{\mathcal{H}}$, $(\Psi^g_{\omega_2})^{\mathcal{H}}$ *and* $(\Psi^g_\omega)^{\mathcal{H}}$ *are used. See more details and a similar construction of* $(\Psi^f_\omega)^{\mathcal{H}}$ *in [31].*

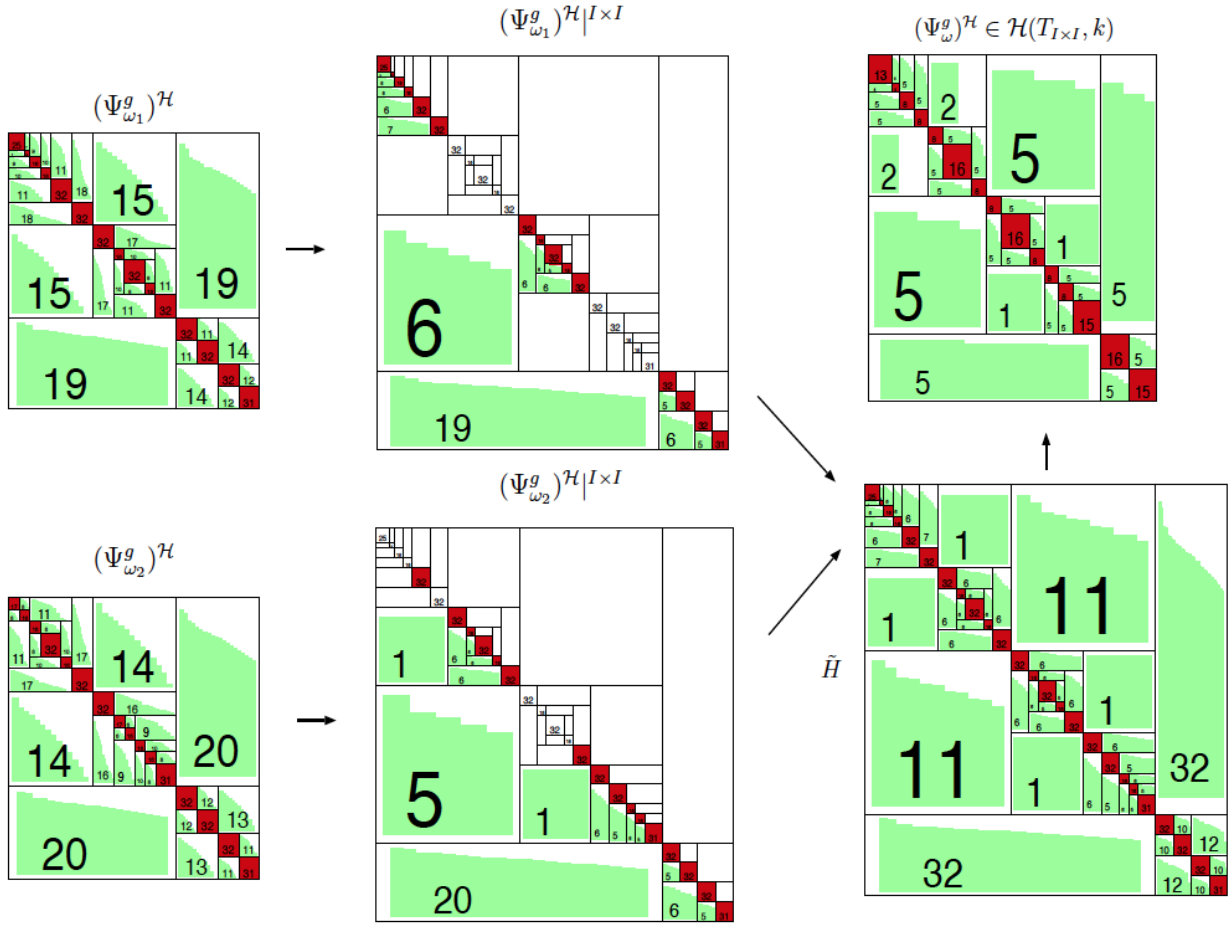

Figure A.1: Building $(\Psi^g_\omega)^{\mathcal{H}} \in \mathbb{R}^{512\times512}$ from $(\Psi^g_{\omega_1})^{\mathcal{H}}$ and $(\Psi^g_{\omega_2})^{\mathcal{H}} \in \mathbb{R}^{384\times384}$. The intermediate matrix $\tilde{H} \in \mathbb{R}^{639\times639}$ is an auxiliary matrix. The maximal size of the diagonal blocks is 32×32. The red (dark) blocks indicate dense matrices. The green (grey) blocks indicate low-rank matrices. The steps inside these blocks show an exponential decay of the corresponding singular values. The white blocks indicate zero blocks. For the acceleration of building the symmetry of Ψ^g_ω is used.