Max-Planck-Institut
für Mathematik
in den Naturwissenschaften
Leipzig

Fast Evaluation of Boundary Integral Operators arising from an Eddy Current Problem

by

Steffen Börm and Jörg Ostrowski

Preprint no.: 33 2003

Fast Evaluation of Boundary Integral Operators Arising from an Eddy Current Problem

S. Börm [*] J. Ostrowski [†]

April 9, 2003

This paper deals with the $\mathcal{H}^2$-matrix approximation of matrices that arise from a Galerkin boundary element (BEM) discretization in the context of the **E**-based eddy current model. The BEM operators are dense, thus need to be compressed. They are of complicated structure, i.e., some kernels and basis functions are vector valued, and test and basis functions are not always identical. The $\mathcal{H}^2$-matrix approximation technique is applied to the kernels of the four different relevant boundary integral operators. Numerical experiments demonstrate the significant acceleration of an iterative solution procedure by means of matrix compression.

Keywords. Computational electromagnetism, $\mathcal{H}^2$-matrix approximation by interpolation, boundary integral operators, eddy current problems.

1 Introduction

We consider the simulation of the induction heating process. In this process, a slowly rotating conduction workpiece is exposed to an oscillating electromagnetic field generated by applying an alternating current to an *inductor*, usually a coil (cf. Figure 1). The field penetrates the workpiece and creates *eddy currents* due to Faraday's law. Ohmic losses of these currents heat the workpiece.

At the relevant frequency range of 10–40 kHz and in the presence of high conductivities, we can simplify Maxwell's equations governing the electromagnetic processes to get the *eddy current model*, which neglects the displacement current [3, 1].

[*] Max-Planck-Institut für Mathematik in den Naturwissenschaften, Inselstraße 22–26, 04103 Leipzig

[†] Sonderforschungsbereich 382, Transferbereich 26, Universität Tübingen. This work was supported by DFG.

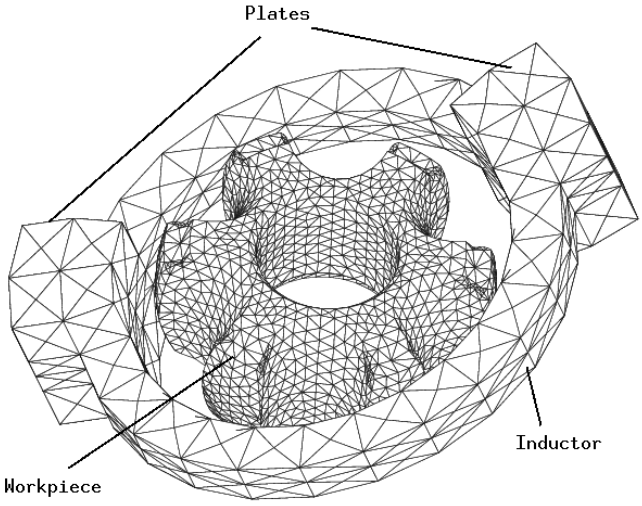

Figure 1: Typical setting for induction heating: Inductor, workpiece and two plates

In the frequency domain, i.e., for time-harmonic excitations with a constant angular frequency $\omega \in \mathbb{R}_{>0}$, it takes the form

$$\operatorname{div} \mathbf{E} = 0, \qquad \text{in } \Omega^+, \tag{1}$$

$$\mathbf{curl}\, \frac{1}{\mu}\, \mathbf{curl}\, \mathbf{E} = -i\omega(\sigma \mathbf{E} + \mathbf{j}_0) \qquad \text{in } \mathbb{R}^3, \tag{2}$$

$$[\mathbf{n} \times \mathbf{E}] = [\mathbf{n} \times \frac{1}{\mu}\, \mathbf{curl}\, \mathbf{E}] = 0 \qquad \text{in } \partial\Omega^-, \tag{3}$$

$$\mathbf{E}(\mathbf{x}) = \mathcal{O}(|\mathbf{x}|^{-2}), \quad \mathbf{curl}\, \mathbf{E}(\mathbf{x}) = \mathcal{O}(|\mathbf{x}|^{-2}) \qquad \text{for } |\mathbf{x}| \to \infty, \tag{4}$$

if a formulation based on the electric field $\mathbf{E}$ is used. Here μ is the magnetic permeability, and σ is the conductivity. The interior of the items, i.e. workpiece, inductor, and plates, is denoted by Ω^-, the exterior vacuum is denoted by $\Omega^+ = \mathbb{R}^3 \setminus \Omega^-$, and $\mathbf{j}_0$ is the exciting current density in the inductor. The jump conditions (3), which are chosen at the interface of the items and the vacuum, are the transmission conditions of normal and tangential component of the electric field.

The domain Ω_- is equipped with a triangulation Ω_h arising from CAD data files consisting of tetrahedra. This triangulation also induces a surface mesh Γ_h of the boundary $\Gamma := \partial\Omega^-$ consisting of flat open triangles.

Finding a viable numerical scheme for solving the eddy current model is not an easy task. One important reason is that one has to cope with the unbounded exterior of a rotating workpiece with general, genuinely three-dimensional geometry.

An appropriate approach is introduced in [8], where the author presents a FEM/BEM-coupled scheme based on edge elements. The FEM part is used in Ω^- and the BEM part, which is needed for the exterior vacuum Ω^+, consists of boundary integrals over

$\partial\Omega^-$. So this scheme uses only elements on $\partial\Omega^-$ and inside Ω^- and can easily be applied to a moving Lagrangian mesh.

For a simply connected domain Ω^-, the discretized version of the BEM part of the corresponding system of equations is given by

$$\begin{pmatrix} \mathsf{M}_\Re & -\mathsf{M}_\Im & -\mathsf{B}^\top & 0 \\ -\mathsf{M}_\Im & -\mathsf{M}_\Re & 0 & \mathsf{B}^\top \\ -\mathsf{B} & 0 & -\mathsf{Q} & 0 \\ 0 & \mathsf{B} & 0 & \mathsf{Q} \end{pmatrix} \begin{pmatrix} \mathbf{E}_\Re \\ \mathbf{E}_\Im \\ \varphi_\Re \\ \varphi_\Im \end{pmatrix} = \text{ right hand side,} \tag{5}$$

see also [9].

If the workpiece is not simply connected, i.e., if there are holes in it, one needs additional matrices that are small enough to remain uncompressed, thus are of minor interest here.

The right hand side of the equation arises from the exciting current in the inductor. The unknows $\mathbf{E}_\Re$ and $\mathbf{E}_\Im \in \mathbb{R}^{\mathcal{E}}$ are the real ($\Re$) and imaginary ($\Im$) parts of the electric field, discretized by surface edge elements (cf. Subsection 2.2). The unknows $\varphi_\Re$ and $\varphi_\Im \in \mathbb{R}^{\mathcal{N}}$ are the real and imaginary parts of a scaled scalar magnetic potential $\mathbf{grad}\,\phi = \frac{\mu_0}{\mu}\,\mathbf{curl}\,\mathbf{E}$, discretized by standard nodal basis functions on the surface.

Here, $\mathcal{N}$ denotes the set of surface nodes, $\mathcal{E}$ denotes that of surface edges, and the set of surface triangles is denoted by $\mathcal{T}$.

All these functions and the BEM matrices $\mathsf{M}_\Re \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$, $\mathsf{M}_\Im \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$, $\mathsf{Q} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ and $\mathsf{B} \in \mathbb{R}^{\mathcal{N}\times\mathcal{E}}$ will be defined more precisely in the next section.

A direct solver cannot be used for equation (5) because it needs too much storage and is too slow. Instead, a fast iterative solver is applied, where matrix-vector multiplications dominate the total complexity.

The triangulation of the boundary Γ must be fine enough to meet two different demands. First, the geometry of the items must be described in a satisfactory way, and second, the desired precision of the solution must be achieved. For our application, this means that a number $\#\mathcal{T} \geq 10000$ of surface triangles must be used for typical workpieces. The occuring BEM operators of equation (5) are dense. A matrix-vector multiplication for n unknowns needs $O(n^2)$ operations, and the amount of storage is of the same order. Parts with $\#\mathcal{T} = 10000$ surface triangles have approximately $\#\mathcal{E} = 15000$ edges and $\#\mathcal{N} = 5000$ nodes. For the storage requirements of the matrices in (5) this means:

- Storing $\mathsf{M}_\Re$ requires $15000^2 \times$ `sizeof(double)` $= 1.67$ GBytes,
- storing $\mathsf{M}_\Im$ requires $15000^2 \times$ `sizeof(double)` $= 1.67$ GBytes,
- storing B requires $15000 \times 5000 \times$ `sizeof(double)` $= 0.56$ GBytes and
- storing Q requires $5000^2 \times$ `sizeof(double)` $= 0.18$ GBytes.

We see that more than 4 GBytes are needed, an amount of memory beyond the capacity of current desktop computers. Therefore, a compression technique must be applied to

the four different boundary integral operators. This can be done by using the $\mathcal{H}^2$-*matrix approximation* [2].

We remark that there is a close relationship of $\mathcal{H}^2$-matrices to the panel clustering technique [7] and the fast multipole method for integral operators [10, 5]. The main advantages of our algorithm are that its implementation is relatively simple, that it leads to quasi-optimal complexity in memory and time and that it does not require specialized expansion systems but can deal with any type of asymptotically smooth (cf. (12)) kernel function.

The following section sketches several aspects of the implementation of our method and demonstrates its properties by numerical experiments.

2 Boundary Element Formulation

2.1 Bilinear forms

For our discrete method, we replace the boundary Γ by a polygonal approximation Γ_h given by the triangulation $\mathcal{T}$:

$$\Gamma_h = \bigcup \left\{ \overline{t} \ : \ t \in \mathcal{T} \right\}.$$

The question of how to choose a suitable triangulation, and therefore approximation Γ_h of Γ, is not a subject of this paper.

The matrices $\mathsf{M}_{\Re}$, $\mathsf{M}_{\Im}$, Q and B occurring in (5) are Galerkin discretizations of boundary integral operators corresponding to the bilinear forms

$$\begin{aligned} m_{\Re}(\mathbf{U},\mathbf{E}) &:= \int_{\Gamma_h}\int_{\Gamma_h} C_1 \langle \gamma_D \mathbf{U}(\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x}) \rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \\ &\quad + \int_{\Gamma_h}\int_{\Gamma_h} \langle \operatorname{curl}_\Gamma \mathbf{U}(\mathbf{y}), \operatorname{curl}_\Gamma \mathbf{E}(\mathbf{x}) \rangle \Phi(\mathbf{x},\mathbf{y}) \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}, \end{aligned} \tag{6}$$

$$m_{\Im}(\mathbf{U},\mathbf{E}) := \int_{\Gamma_h}\int_{\Gamma_h} C_2 \langle \gamma_D \mathbf{U}(\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x}) \rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}, \tag{7}$$

$$q(\eta,\phi) := \int_{\Gamma_h}\int_{\Gamma_h} \langle \mathbf{curl}_\Gamma \,\eta(\mathbf{y}), \mathbf{curl}_\Gamma \,\phi(\mathbf{x}) \rangle \Phi(\mathbf{x},\mathbf{y}) \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}, \tag{8}$$

$$\begin{aligned} b(\eta,\mathbf{E}) &:= -\frac{1}{2}\int_{\Gamma_h} \langle \mathbf{curl}_\Gamma \,\eta(\mathbf{x}), \gamma_D \mathbf{E}(\mathbf{x}) \rangle \,\mathrm{d}\mathbf{x} \\ &\quad + \int_{\Gamma_h}\int_{\Gamma_h} \langle \mathbf{curl}_\Gamma \,\eta(\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x}) \rangle \langle \mathbf{grad}_x \,\Phi(\mathbf{x},\mathbf{y}), \mathbf{n}(\mathbf{x}) \rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}, \\ &\quad - \int_{\Gamma_h}\int_{\Gamma_h} \langle \mathbf{curl}_\Gamma \,\eta(\mathbf{y}), \mathbf{n}(\mathbf{x}) \rangle \langle \mathbf{grad}_x \,\Phi(\mathbf{x},\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x}) \rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}, \end{aligned} \tag{9}$$

where $C_1, C_2 \in \mathbb{R}_{>0}$ are constants, $\mathbf{n}$ is the outer normal of the surface Γ, $\mathbf{E}$ is an electric field and $\mathbf{U}$ a corresponding test function, while ϕ is the scalar magnetic potential mentioned above and η is the corresponding test function.

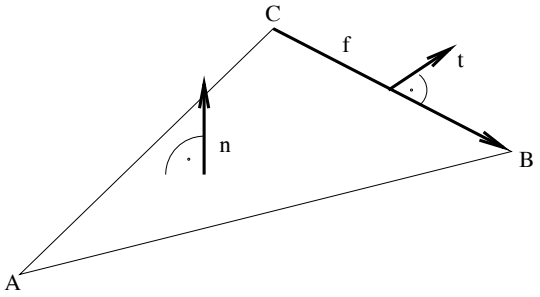

Figure 2: Setting for the definition of basis functions

Φ is the fundamental solution of the Laplace operator in three space dimensions given by

$$\Phi(\mathbf{x},\mathbf{y}) := \frac{1}{4\pi}\frac{1}{|\mathbf{x}-\mathbf{y}|}, \quad \mathbf{x},\mathbf{y} \in \mathbb{R}^3,\ \mathbf{x} \neq \mathbf{y}. \tag{10}$$

γ_D is the tangential trace operator on Γ_h, $\mathbf{curl}_\Gamma$ and curl_Γ are the vector-valued and scalar-valued surface curls. These three operators are given by

$$\begin{aligned} \gamma_D \mathbf{E} &:= \mathbf{n} \times (\lim_{\epsilon \to +0} \mathbf{E}(\mathbf{x} + \epsilon \mathbf{n}) \times \mathbf{n}), \\ \mathbf{curl}_\Gamma\, \phi &:= \gamma_D(\mathbf{grad}\, \phi) \times \mathbf{n}, \\ \mathrm{curl}_\Gamma\, \mathbf{E} &:= \langle \mathbf{n}, \mathbf{curl}\, \mathbf{E} \rangle. \end{aligned}$$

2.2 Discretization

We want to use a piecewise polynomial conforming Galerkin approach to discretize the problem, so we need appropriate discrete spaces. This implies that the discretized potentials η and ϕ have to be continuous at the edges of the triangulation, while the discretized vector fields $\mathbf{U}$ and $\mathbf{E}$ have to have a continuous tangential component.

Further details on the discretization and the properties of the boundary integral operators are given in [8].

Let us consider a triangle $T = \overline{\mathbf{ABC}} \in \mathcal{T}$. The surface measure of the triangle $\overline{\mathbf{ABC}}$ is given by $S := \|(\mathbf{B}-\mathbf{A}) \times (\mathbf{C}-\mathbf{A})\|/2$. We suppose that the vertices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are ordered counter-clockwise, so that the outer normal vector of T is given by

$$\mathbf{n} := \frac{(\mathbf{B}-\mathbf{A}) \times (\mathbf{C}-\mathbf{A})}{2S}.$$

We introduce the vectors

$$\mathbf{f} := \mathbf{B}-\mathbf{C} \quad \text{and} \quad \mathbf{t} := \frac{\mathbf{n} \times \mathbf{f}}{\|\mathbf{n} \times \mathbf{f}\|}$$

(cf. Figure 2).

The local edge element basis function corresponding to the edge $e := \overline{\mathbf{BC}}$ is given by

$$\mathbf{b}_{T,e}(\mathbf{x}) := \frac{(\mathbf{x}-\mathbf{A})\times\mathbf{n}}{2S}.$$

For $\mathbf{x} = \mathbf{C} + \alpha(\mathbf{B}-\mathbf{C})$, we have

$$\begin{aligned}\langle \mathbf{b}_{T,e}(\mathbf{x}), \mathbf{f}\rangle &= \frac{\langle (\mathbf{x}-\mathbf{A})\times\mathbf{n}, \mathbf{f}\rangle}{2S} = \frac{\det(\mathbf{x}-\mathbf{A},\mathbf{n},\mathbf{B}-\mathbf{C})}{2S} = \frac{\det(\mathbf{C}-\mathbf{A},\mathbf{n},\mathbf{B}-\mathbf{C})}{2S} \\ &= \frac{\det(\mathbf{B}-\mathbf{A},\mathbf{C}-\mathbf{A},\mathbf{n})}{2S} = \frac{\langle (\mathbf{B}-\mathbf{A})\times(\mathbf{C}-\mathbf{A}),\mathbf{n}\rangle}{2S} = \langle \mathbf{n},\mathbf{n}\rangle = 1,\end{aligned}$$

i.e., the tangential component of $\mathbf{b}_{T,e}$ on the edge e is constant and equal to 1. Similar computations reveal that the tangential component on $\mathbf{b}_{T,e}$ on the other edges $\overline{\mathbf{AB}}$ and $\overline{\mathbf{CA}}$ are constant and equal to zero. This implies that we can build a global edge element basis function $\mathbf{b}_e$ for each edge $e \in \mathcal{E}$ by combining the local edge element basis functions $\mathbf{b}_{T,e}$ corresponding to the triangles touching e.

The local nodal basis function corresponding to the vertex $v := \mathbf{A}$ is given by

$$\phi_{T,v}(\mathbf{x}) := -\frac{\|\mathbf{f}\|\langle \mathbf{x}-\mathbf{C},\mathbf{t}\rangle}{2S}.$$

We have

$$\begin{aligned}\phi_{T,v}(\mathbf{A}) &= -\frac{\|\mathbf{f}\|\langle \mathbf{A}-\mathbf{C},\mathbf{t}\rangle}{2S} = -\frac{\|\mathbf{f}\|\langle \mathbf{n}\times\mathbf{f},\mathbf{A}-\mathbf{C}\rangle}{2S\|\mathbf{n}\times\mathbf{f}\|} = -\frac{\det(\mathbf{n},\mathbf{B}-\mathbf{C},\mathbf{A}-\mathbf{C})}{2S} \\ &= \frac{\det(\mathbf{n},\mathbf{B}-\mathbf{A},\mathbf{C}-\mathbf{A})}{2S} = \frac{\det(\mathbf{B}-\mathbf{A},\mathbf{C}-\mathbf{A},\mathbf{n})}{2S} = \langle \mathbf{n},\mathbf{n}\rangle = 1.\end{aligned}$$

It is obvious that $\phi_{T,v}(\mathbf{B}) = \phi_{T,v}(\mathbf{C}) = 0$ holds. Since $\phi_{T,v}$ is affine, we can build a global nodal basis function ϕ_v for each vertex $v \in \mathcal{N}$ by combining the local nodal basis functions $\phi_{T,v}$ corresponding to the triangles touching v.

Discretizing the bilinear forms (6)-(9) leads to the matrices $\mathsf{M}_{\Re} \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$, $\mathsf{M}_{\Im} \in \mathbb{R}^{\mathcal{E}\times\mathcal{E}}$, $\mathsf{Q} \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ and $\mathsf{B} \in \mathbb{R}^{\mathcal{N}\times\mathcal{E}}$ defined by

$$\begin{gathered}\mathsf{M}_{\Re,ij} := m_{\Re}(\mathbf{b}_i,\mathbf{b}_j), \quad \mathsf{M}_{\Im,ij} := m_{\Im}(\mathbf{b}_i,\mathbf{b}_j), \\ \mathsf{Q}_{\iota\kappa} := q(\psi_\iota,\psi_\kappa) \quad \text{and} \quad \mathsf{B}_{\iota j} := b(\psi_\iota,\mathbf{b}_j)\end{gathered}$$

for $i,j \in \mathcal{E}$ and $\iota,\kappa \in \mathcal{N}$.

2.3 Properties of the discretized matrices

The amount of work involved in assembling the matrices can be significantly reduced by making use of the fact that

$$\begin{aligned}m_{\Re}(\mathbf{U},\mathbf{E}) &= (C_1/C_2)m_{\Im}(\mathbf{U},\mathbf{E}) + g(\operatorname{curl}_\Gamma \mathbf{U}, \operatorname{curl}_\Gamma \mathbf{E}) \quad \text{and} \\ q(\eta,\phi) &= \sum_{l=1}^{3} g((\mathbf{curl}_\Gamma\,\eta)_l, (\mathbf{curl}_\Gamma\,\phi)_l)\end{aligned}$$

hold for

$$g(\zeta,\theta) := \int_{\Gamma_h}\int_{\Gamma_h} \zeta(\mathbf{y})\theta(\mathbf{x})\Phi(\mathbf{x},\mathbf{y})\,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}. \tag{11}$$

An important property of the basis functions ϕ_v for $v \in \mathcal{N}$ and $\mathbf{b}_e$ for $e \in \mathcal{E}$ is that $(\mathbf{curl}_\Gamma\,\phi_v)|_t$ and $(\mathrm{curl}_\Gamma\,\mathbf{b}_e)|_t$ are constant for each triangle $t \in \mathcal{T}$. This implies that we can represent the non-sparse part of $m_{\mathfrak{R}}(\cdot,\cdot)$ and $q(\cdot,\cdot)$ by discretizing $g(\cdot,\cdot)$ by piecewise constant functions $(\chi_t)_{t\in\mathcal{T}}$. This does not hold for the discretization of $b(\cdot,\cdot)$, where we have to use the piecewise linear basis functions $\mathbf{b}_e$.

All bilinear forms are double integrals with differential operators (namely $\mathbf{curl}_\Gamma$ and curl_Γ), the trace operator γ_D and the fundamental solution $\Phi(\cdot,\cdot)$ and its derivatives as integrands. The differential operators and the trace operator do not increase the support of the basis function and can therefore be considered "harmless", which leaves us with the problem of discretizing the integral operators involving the non-local function $\Phi(\cdot,\cdot)$, namely those that correspond to the bilinear forms $g(\cdot,\cdot)$ and $b(\cdot,\cdot)$.

3 Matrix approximation

The integral kernels in (6)-(9) describe long range interactions between the boundary regions at $\mathbf{x}$ and $\mathbf{y}$. Their strength depends on the inverse of the distance $|\mathbf{x}-\mathbf{y}|$. A common strategy for compression of the matrices is to approximate the kernels in the so-called *far-field*, i.e., in regions that are far away from each other, whereas one sticks to exact kernels in the *near-field*.

Panel clustering methods are widely used [7, 11]. They are based on degenerate approximations of the kernel function in the far-field. We construct the approximation by using an interpolation instead of the more traditional Taylor expansion. This leads to fast algorithms, see [4], that can be stated in the context of $\mathcal{H}^2$-matrix techniques [2, 6].

We replace the singularity function Φ by its Chebyshev interpolation, so we need only pointwise evaluations of Φ instead of the derivatives required by Taylor-based approaches. Since the interpolation of Φ on the entire domain $\Gamma_h \times \Gamma_h$ would not lead to good results due to the singularity at $\mathbf{x} = \mathbf{y}$, we consider sub-domains of the form $\tau \times \sigma$ and apply the interpolation locally. If a constant order m of the interpolation is used, then the resulting approximation is an $\mathcal{H}^2$-matrix (cf. [2]). Using this structure, we can perform the matrix-vector multiplication and the discretization of the far-field in $\mathcal{O}(nm^3)$ operations, where n is the number of the degrees of freedom.

In the $\mathcal{H}^2$-matrix approximation method consists of two main parts: The *preparation phase* of the compressed matrix representation, which needs to be performed only once even for several matrix-vector multiplications, and the *matrix-vector multiplication* itself. The preparation phase consists of three parts: we have to find a suitable splitting of $\Gamma_h \times \Gamma_h$ into sub-domains, we have to compute the matrices corresponding to the $\mathcal{H}^2$-representation of the far-field blocks and we have to compute the coefficients of the near-field blocks.

3.1 Motivation

3.1.1 Approximation

The kernel function Φ is *asymptotically smooth*, i.e., there are constants $C_{\mathrm{as}}, c_0, d \in \mathbb{R}_{>0}$ such that

$$|\partial_x^\alpha \partial_y^\beta \Phi(\mathbf{x},\mathbf{y})| \leq C_{\mathrm{as}} c_0^{|\alpha|+|\beta|} (\alpha+\beta)! \|\mathbf{x}-\mathbf{y}\|^{-d-|\alpha|-|\beta|} \tag{12}$$

holds for all $\mathbf{x},\mathbf{y} \in \mathbb{R}^3$ with $\mathbf{x} \neq \mathbf{y}$ and all multi-indices $\alpha, \beta \in \mathbb{N}_0^3$. In the case of the Laplace kernel, we have $c_0 = 1$ and $d = 1$.

Let $\tau, \sigma \subseteq \mathbb{R}^d$ be sub-domains of Γ_h such that $\operatorname{dist}(\tau,\sigma) > 0$. We introduce the local bilinear form $g^{\tau,\sigma}$ given by

$$g^{\tau,\sigma}(\zeta,\theta) := \int_\tau \int_\sigma \zeta(\mathbf{y})\theta(\mathbf{x})\Phi(\mathbf{x},\mathbf{y}) \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x}.$$

The equation (12) implies that the function Φ is smooth on $\tau \times \sigma$, so we can approximate it on this sub-domain by polynomials and use the approximation of the kernel function to define an approximation of the bilinear form $g^{\tau,\sigma}$.

In order to keep our algorithm simple, we will not work with τ and σ directly, but use axis-parallel boxes: Let B^τ and B^σ be minimal d-dimensional axis-parallel boxes satisfying $\tau \subseteq B^\tau$ and $\sigma \subseteq B^\sigma$.

We apply m-th order tensor product interpolation operators $\mathcal{I}_m^\tau$ and $\mathcal{I}_m^\sigma$ given by

$$\mathcal{I}_m^\tau[u](\mathbf{x}) = \sum_{\nu \in M} u(\mathbf{x}_\nu^\tau)\mathcal{L}_\nu^\tau(\mathbf{x}) \quad \text{and} \quad \mathcal{I}_m^\sigma[v](\mathbf{y}) = \sum_{\mu \in M} v(\mathbf{x}_\mu^\sigma)\mathcal{L}_\mu^\sigma(\mathbf{y}),$$

where $(\mathbf{x}_\nu^\tau)_{\nu\in M}$ and $(\mathbf{x}_\mu^\sigma)_{\mu\in M}$ are interpolation points in B^τ and B^σ and $(\mathcal{L}_\nu^\tau)_{\nu\in M}$ and $(\mathcal{L}_\mu^\sigma)_{\mu\in M}$ are the corresponding Lagrange polynomials.

We approximate Φ by

$$\tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}) := (\mathcal{I}_m^\tau \otimes \mathcal{I}_m^\sigma)[\Phi](\mathbf{x},\mathbf{y}) = \sum_{\nu\in M}\sum_{\mu\in M} \Phi(\mathbf{x}_\nu^\tau, \mathbf{x}_\mu^\sigma)\mathcal{L}_\nu^\tau(\mathbf{x})\mathcal{L}_\mu^\sigma(\mathbf{y}). \tag{13}$$

3.1.2 Low-rank representation

Replacing Γ_h by τ and σ and Φ by $\tilde{\Phi}^{\tau,\sigma}$ in (11), we get

$$\begin{aligned}
\tilde{g}^{\tau,\sigma}(\zeta,\theta) &:= \int_\tau \int_\sigma \zeta(\mathbf{y})\theta(\mathbf{x})\tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}) \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} && (14)\\
&= \sum_{\nu\in M}\sum_{\mu\in M} \underbrace{\Phi(\mathbf{x}_\nu^\tau,\mathbf{x}_\mu^\sigma)}_{=:\mathsf{S}_{\nu,\mu}^{\tau,\sigma}} \underbrace{\int_\tau \theta(\mathbf{x})\mathcal{L}_\nu^\tau(\mathbf{x})\,\mathrm{d}\mathbf{x}}_{=:V_\nu^\tau(\theta)} \underbrace{\int_\sigma \zeta(\mathbf{y})\mathcal{L}_\mu^\sigma(\mathbf{y})\,\mathrm{d}\mathbf{y}}_{=:V_\mu^\sigma(\zeta)} \\
&= \sum_{\nu\in M}\sum_{\mu\in M} \mathsf{S}_{\nu,\mu}^{\tau,\sigma} V_\nu^\tau(\theta) V_\mu^\sigma(\zeta), && (15)
\end{aligned}$$

i.e., the bilinear form can be expressed in terms of a, typically small, matrix $\mathsf{S}^{\tau,\sigma}$ and a small number of functionals $(V_\nu^\tau)_{\nu\in M}$ and $(V_\mu^\sigma)_{\mu\in M}$.

The advantage of the new representation becomes obvious if we discretize the new bilinear form $\tilde{g}^{\tau,\sigma}$ by introducing

$$\tilde{\mathsf{G}}^{\tau,\sigma}_{ts} := \tilde{g}^{\tau,\sigma}(\chi_t, \chi_s), \quad \mathsf{V}^{\tau}_{t\nu} := V^{\tau}_{\nu}(\chi_t) \quad \text{and} \quad \mathsf{V}^{\sigma}_{s\mu} := V^{\sigma}_{\mu}(\chi_s).$$

The equation (15) now takes the form

$$\tilde{\mathsf{G}}^{\tau,\sigma} = \mathsf{V}^{\tau}\mathsf{S}^{\tau,\sigma}(\mathsf{V}^{\sigma})^{\top}. \tag{16}$$

We set

$$n^{\tau} := \#\{t \in \mathcal{T} \ : \ \tau \cap t \neq \emptyset\}, \quad n^{\sigma} := \#\{s \in \mathcal{T} \ : \ \sigma \cap s \neq \emptyset\} \quad \text{and} \quad k := \#M$$

and find that storing $\tilde{\mathsf{G}}^{\tau,\sigma}$ as a dense matrix requires $n^{\tau}n^{\sigma}$ units of memory, while storing V^{τ}, V^{σ} and $\mathsf{S}^{\tau,\sigma}$ requires $n^{\tau}k + n^{\sigma}k + k^2$ units of memory. Typically k is much smaller than n^{τ} and n^{σ}, so the factorized representation is much more efficient.

3.1.3 Precision

We have seen that replacing the kernel function Φ by its interpolant $\tilde{\Phi}^{\tau,\sigma}$ leads to an efficient representation of the discretized matrix. In order to be able to use this representation, we have to ensure that the error introduced by the interpolation can be controlled, i.e., that an estimate of the form

$$|\tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}) - \Phi(\mathbf{x},\mathbf{y})| \leq \epsilon_m$$

holds for $\mathbf{x} \in \tau$, $\mathbf{y} \in \sigma$, where $\epsilon_m \in \mathbb{R}_{>0}$ depends favorably on the interpolation order m.

If we use Chebyshev interpolation for $\mathcal{I}^{\tau}_m$ and $\mathcal{I}^{\sigma}_m$, we have a constant $C_{\mathrm{in}} \in \mathbb{R}_{>0}$ such that

$$\begin{aligned} &\|\Phi - \mathcal{I}^{\tau}_m \otimes \mathcal{I}^{\sigma}_m[\Phi]\|_{\infty, B^{\tau}\times B^{\sigma}} \\ &\leq C_{\mathrm{in}} \frac{3^{-m}}{(m+1)!} \operatorname{diam}(B^{\tau} \times B^{\sigma})^{m+1} \|D^{m+1}_{xy}\Phi\|_{\infty, B^{\tau}\times B^{\sigma}} \end{aligned} \tag{17}$$

holds for all m and all functions $u \in C^{m+1}(B^{\tau} \times B^{\sigma})$ (cf. the Appendix of [2]), where $\operatorname{diam}(B^{\tau} \times B^{\sigma})$ denotes the Euclidean diameter of the axis-parallel box $B^{\tau} \times B^{\sigma}$ and $D^{m+1}\Phi$ is the total derivative of order $m+1$ of Φ.

Combining this estimate with (12), we find

$$\|\Phi - \tilde{\Phi}^{\tau,\sigma}\|_{\infty, B^{\tau}\times B^{\sigma}} \leq \frac{C_{\mathrm{in}}C_{\mathrm{as}}}{\operatorname{dist}(B^{\tau}, B^{\sigma})^d} \ 3^{-m} \ \left(\frac{c_0 \operatorname{diam}(B^{\tau} \times B^{\sigma})}{\operatorname{dist}(B^{\tau}, B^{\sigma})}\right)^{m+1}.$$

This implies that we have to be able to bound the diameter by the distance in order to reach a uniform bound for the approximation error, i.e., we require the *admissibility condition*

$$\operatorname{diam}(B^{\tau} \times B^{\sigma}) \leq \eta \operatorname{dist}(B^{\tau}, B^{\sigma}) \tag{18}$$

to hold for a parameter $\eta \in]0, 3/c_0[$ and find

$$\|\Phi - \tilde{\Phi}^{\tau,\sigma}\|_{\infty, B^{\tau}\times B^{\sigma}} \leq \epsilon_m := \frac{C_{\mathrm{in}}C_{\mathrm{as}}\eta^{d+1}}{\operatorname{diam}(B^{\tau} \times B^{\sigma})^d}(c_0\eta/3)^m, \tag{19}$$

so the interpolant converges exponentially on $B^{\tau} \times B^{\sigma} \supseteq \tau \times \sigma$ if the order m is increased.

3.2 Decomposition of $\Gamma_h \times \Gamma_h$

In the previous section, we have seen that we can efficiently approximate the local interpolants $\tilde{\Phi}^{\tau,\sigma}$ if the sub-domains τ, σ of Γ_h satisfy the admissibility condition (18).

3.2.1 Block partition

Obviously, the pair (Γ_h, Γ_h) does not satisfy this condition, so we have to split $\Gamma_h \times \Gamma_h$, the domain of integration of the bilinear form $g(\cdot,\cdot)$, into a collection P of sub-domains that either satisfy this condition or are so small that we can treat them directly without compromising the efficiency.

The family $P \subseteq \{(\tau,\sigma) \ : \ \tau,\sigma \subseteq \mathcal{T}\}$ has to satisfy the following conditions:

$$\bigcup \{\overline{\tau \times \sigma} \ : \ (\tau,\sigma) \in P\} = \Gamma_h \times \Gamma_h,$$

$$(\tau_1 \times \sigma_1) \cap (\tau_2 \times \sigma_2) \neq \emptyset \Rightarrow (\tau_1,\sigma_1) = (\tau_2,\sigma_2) \qquad \text{for all } (\tau_1,\sigma_1),(\tau_2,\sigma_2) \in P.$$

According to the admissibility condition, we split P into the set P_{far} of *far-field* blocks and the set P_{near} of *near-field* blocks:

$$P_{\mathrm{far}} := \{(\tau,\sigma) \in P \ : \ \operatorname{diam}(B^\tau \times B^\sigma) \leq \eta \operatorname{dist}(B^\tau, B^\sigma)\}, \quad P_{\mathrm{near}} := P \setminus P_{\mathrm{far}}.$$

For each $(\tau,\sigma) \in P_{\mathrm{far}}$, we can construct a local approximation $\tilde{g}^{\tau,\sigma}(\cdot,\cdot)$ of the form (11), (15). For the remaining blocks $(\tau,\sigma) \in P_{\mathrm{near}}$, we use the original local bilinear form $g^{\tau,\sigma}(\cdot,\cdot)$. The approximation of the *global* bilinear form $g(\cdot,\cdot)$ is then given as the sum of the local bilinear forms:

$$\tilde{g}(\zeta,\theta) = \sum_{(\tau,\sigma)\in P_{\mathrm{far}}} \tilde{g}^{\tau,\sigma}(\zeta,\theta) + \sum_{(\tau,\sigma)\in P_{\mathrm{near}}} g^{\tau,\sigma}(\zeta,\theta). \tag{20}$$

This corresponds to replacing the kernel function Φ by its piecewise m-th order interpolant given by

$$\tilde{\Phi}(\mathbf{x},\mathbf{y}) := \begin{cases} \tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}) & \text{if } (\mathbf{x},\mathbf{y}) \in \tau \times \sigma \text{ for } (\tau,\sigma) \in P_{\mathrm{far}}, \\ \Phi(\mathbf{x},\mathbf{y}) & \text{otherwise.} \end{cases}$$

The estimate (19) implies

$$|\Phi(\mathbf{x},\mathbf{y}) - \tilde{\Phi}(\mathbf{x},\mathbf{y})| \leq \epsilon_m$$

for all $\mathbf{x},\mathbf{y} \in \Gamma_h$, so we find

$$\begin{aligned} |g(\zeta,\theta) - \tilde{g}(\zeta,\theta)| &= \left| \int_{\Gamma_h} \int_{\Gamma_h} \zeta(\mathbf{y})\theta(\mathbf{x}) \left(\Phi(\mathbf{x},\mathbf{y}) - \tilde{\Phi}(\mathbf{x},\mathbf{y})\right) \mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \right| \\ &\leq \|\Phi - \tilde{\Phi}\|_{\infty,\Gamma_h\times\Gamma_h} \int_{\Gamma_h} |\zeta(\mathbf{y})|\,\mathrm{d}\mathbf{y} \int_{\Gamma_h} |\theta(\mathbf{x})|\,\mathrm{d}\mathbf{x} \\ &\leq \epsilon_m \|\zeta\|_{L^1} \|\theta\|_{L^1} \leq \epsilon_m \ |\Gamma_h| \ \|\zeta\|_{L^2(\Gamma_h)} \|\theta\|_{L^2(\Gamma_h)}. \end{aligned}$$

The construction of a good partition, i.e., a partition where the number of non-admissible blocks is small, is not trivial. Therefore, we will start by constructing a hierarchy of partitions of Γ_h, i.e., of the set of triangles, and then use this hierarchy to create a partition of $\Gamma_h \times \Gamma_h$.

3.2.2 Cluster tree

We will construct the hierarchy of partitions of Γ_h by successively splitting domains. Since a pair (τ, σ) of sub-domains of Γ_h is admissible if the diameter of $B^\tau \times B^\sigma$ is smaller than the distance of these boxes (recall that B^τ and B^σ are the minimal axis-parallel boxes containing τ and σ), a good strategy is to split domains in such a way that the diameters of the newly created sub-domains are decreased as much as possible.

In order to keep the implementation simple, we consider only sub-domains τ that are the union of a set $\hat{\tau} \subseteq \mathcal{T}$ of triangles, and we identify the sub-domain τ with the set $\hat{\tau}$. Then successive splitting of domains, starting with the set $\mathcal{T}$ of all triangles, leads to a tree structure, the *cluster tree*:

Definition 1 *A tree $\mathcal{C}$ is called a* cluster tree *for a set $\mathcal{T}$ of triangles if*

- *the set of all triangles is the root of $\mathcal{C}$, i.e.,* $\operatorname{root}(\mathcal{C}) = \mathcal{T}$, *and*
- *if a node $\tau \in \mathcal{C}$ is not a leaf, then it is the disjoint union of its sons, i.e.,*

$$\tau = \dot{\bigcup} \left\{ \tau' \ : \ \tau' \in \operatorname{sons}(\tau) \right\}.$$

Each node $\tau \in \mathcal{C}$ is called a cluster.

A cluster tree can be constructed from an arbitrary set of triangles by *binary space partitioning*: We start with the root cluster containing all the triangles, split it into two son clusters and repeat the procedure recursively until the clusters contain less than a fixed number $C_{\mathrm{lf}} \geq 1$ of triangles.

The splitting strategy is based on the geometry: We denote the center of each triangle $t \in \mathcal{T}$ by $\mu_t \in \mathbb{R}^3$, choose a suitable coordinate axis and split the set along this axis. This leads to the algorithm in Figure 3.

Remark 2 (Geometric balancing) *If the surface triangulation is not quasi-uniform, e.g., if it is the result of an adaptive refinement strategy, then a different splitting technique than that given in Figure 3 is to be applied: Instead of splitting the cluster τ into two clusters τ_1, τ_2 that are of similar cardinality, we use the middle of the coordinate interval $[i_l, s_l]$ in order to determine which son of τ has to contain which triangles*

$$\tau_1 := \{t \in \tau \ : \ \mu_{t,l} \leq (s_l + i_l)/2\}, \quad \tau_2 := \{t \in \tau \ : \ \mu_{t,l} > (s_l + i_l)/2\},$$

with l still denoting the longest edge.

3.2.3 Construction of a block partition

The definition of the axis-parallel boxes B^τ carries over to clusters:

Definition 3 (Bounding boxes) *Let $\tau \in \mathcal{C}$. The minimal axis-parallel box $B \subseteq \mathbb{R}^3$ satisfying $t \subseteq B$ for all $t \in \tau$ is called the* bounding box *of the cluster τ and denoted by B^τ.*

```
procedure GeometricBisection(τ);
begin
  if #τ ≥ C_lf then begin
    for j := 1 to 3 do begin                    { find splitting coordinate }
      s_j := max{μ_{t,j} : t ∈ T}; i_j := min{μ_{t,j} : t ∈ T}; δ_j := s_j − i_j
    end;
    Choose l ∈ {1,2,3} such that δ_l = max{δ_j : j ∈ {1,...,3}};
    Split τ = τ_1 ∪ τ_2 with |#τ_1 − #τ_2| ≤ 1 and μ_{t_1,l} ≤ μ_{t_2,l} for t_1 ∈ τ_1, t_2 ∈ τ_2;
    GeometricBisection(τ_1); GeometricBisection(τ_2);
    sons(τ) := {τ_1, τ_2}
  end else sons(τ) := ∅
end
```

Figure 3: Cardinality (#) balanced geometric bisection

We will use the following simplified admissibility condition:

Remark 4 (Simplified admissibility condition) *In some applications, the admissibility condition* (18) *is replaced by the condition*

$$\max\{\operatorname{diam}(B^\tau), \operatorname{diam}(B^\sigma)\} \leq 2\eta' \operatorname{dist}(B^\tau, B^\sigma). \tag{21}$$

By setting $\eta := 2\sqrt{2}\eta'$*, we find*

$$\operatorname{diam}(B^\tau \times B^\sigma) \leq \sqrt{2}\max\{\operatorname{diam}(B^\tau), \operatorname{diam}(B^\sigma)\} \leq 2\sqrt{2}\eta' \operatorname{dist}(B^\tau, B^\sigma),$$

so the simplified admissibility condition implies the original condition. If $\eta' < 3/(2\sqrt{2}c_0)$*, then we get* $\eta < 3/c_0$ *and therefore exponential convergence.*

Based on the criteria (18) or (21) and a cluster tree, we can find a partition of $\Gamma_h \times \Gamma_h$ by calling the algorithm BlockPartition(Γ_h, Γ_h, $\emptyset$) of Figure 4.

3.3 Matrix-vector multiplication

The matrix-vector multiplication $\mathbf{v} = \tilde{\mathsf{G}}\mathbf{u}$ with a vector $\mathbf{u} \in \mathbb{R}^{\mathcal{T}}$ corresponds to the evaluation of equation (20). We split this evaluation into four parts:

- **Forward Transformation:** We transform $\mathbf{u}$ into cluster coefficients $\mathbf{u}^\sigma := \mathsf{V}^{\sigma\top}\mathbf{u} \in \mathbb{R}^M$.
- **Transformed Multiplication:** We evaluate the sum $\mathbf{v}^\tau := \sum_{\sigma\in\operatorname{row}(\tau)} \mathsf{S}^{\tau,\sigma}\mathbf{u}^\sigma \in \mathbb{R}^M$ for $\operatorname{row}(\tau) := \{\sigma \in \mathcal{C} \ : \ (\tau,\sigma) \in P_{\text{far}}\}$.

```
procedure BlockPartition(τ, σ, var P );
begin
  if (τ, σ) is admissible then P := P ∪ {(τ, σ)}
  else if sons(τ) = ∅ or sons(σ) = ∅ then P := P ∪ {(τ, σ)}
  else for τ′ ∈ sons(τ) and σ′ ∈ sons(σ) do BlockPartition(τ′, σ′, P)
end
```

Figure 4: Construction of a partition of $\Gamma_h \times \Gamma_h$

- **Backward Transformation:** We transform the coefficients $\mathbf{v}^\tau$ back into the standard base in order to find $\mathbf{v}_{\text{far}} := \sum_{\tau\in\mathcal{C}} \mathsf{V}^\tau \mathbf{v}^\tau$.
- **Near-field Computation:** We conclude the computation by adding the near-field part $\mathbf{v} := \mathbf{v}_{\text{far}} + \sum_{(\tau,\sigma)\in P_{\text{near}}} \mathsf{G}^{\tau,\sigma}\mathbf{u}$.

Due to

$$\begin{aligned}\tilde{\mathsf{G}}\mathbf{u} &= \sum_{(\tau,\sigma)\in P_{\text{near}}} \mathsf{G}^{\tau,\sigma}\mathbf{u} + \sum_{(\tau,\sigma)\in P_{\text{far}}} \mathsf{V}^\tau \mathsf{S}^{\tau,\sigma}\mathsf{V}^{\sigma\top}\mathbf{u}\\ &= \sum_{(\tau,\sigma)\in P_{\text{near}}} \mathsf{G}^{\tau,\sigma}\mathbf{u} + \sum_{(\tau,\sigma)\in P_{\text{far}}} \mathsf{V}^\tau \mathsf{S}^{\tau,\sigma}\mathbf{u}^\sigma\\ &= \sum_{(\tau,\sigma)\in P_{\text{near}}} \mathsf{G}^{\tau,\sigma}\mathbf{u} + \sum_{\tau\in\mathcal{C}} \mathsf{V}^\tau\mathbf{v}^\tau = \mathbf{v},\end{aligned}$$

this four-step procedure indeed computes the matrix-vector product.

In order to find a fast algorithm for the matrix-vector multiplication, we will now introduce an alternative representation of the matrices V^τ: Let $\tau \in \mathcal{C}$ be a cluster with $\text{sons}(\tau) \neq \emptyset$. Since we use the same order of interpolation for all clusters, we have

$$\mathcal{L}^\tau_\nu = \mathcal{I}^{\tau'}[\mathcal{L}^\tau_\nu] = \sum_{\nu'\in M} \mathcal{L}^\tau_\nu(\mathbf{x}^{\tau'}_{\nu'})\mathcal{L}^{\tau'}_{\nu'} = \sum_{\nu'\in M} \mathsf{T}^{\tau',\tau}_{\nu'\nu}\mathcal{L}^{\tau'}_{\nu'}$$

with *transfer matrices* $\mathsf{T}^{\tau',\tau} \in \mathbb{R}^{M\times M}$ defined by

$$\mathsf{T}^{\tau',\tau}_{\nu'\nu} := \mathcal{L}^\tau_\nu(\mathbf{x}^{\tau'}_{\nu'}). \tag{22}$$

This alternative representation implies

$$\begin{aligned}\mathsf{V}^\tau_{t\nu} &= \int_\tau \chi_t(\mathbf{x})\mathcal{L}^\tau_\nu(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{\tau'\in\text{sons}(\tau)} \int_{\tau'} \chi_t(\mathbf{x})\mathcal{L}^\tau_\nu(\mathbf{x})\,\mathrm{d}\mathbf{x}\\ &= \sum_{\tau'\in\text{sons}(\tau)} \sum_{\nu'\in M} \mathcal{L}^\tau_\nu(\mathbf{x}^{\tau'}_{\nu'}) \int_{\tau'} \chi_t(\mathbf{x})\mathcal{L}^{\tau'}_{\nu'}(\mathbf{x})\,\mathrm{d}\mathbf{x}\end{aligned}$$

```
procedure FastForward(σ, u, var (u^σ)_{σ∈C});
begin
  if sons(σ) = ∅ then u^σ := V^σ^T u
  else begin
    for σ' ∈ sons(σ) do FastForward(σ', u, (u^σ));
    u^σ := Σ_{σ'∈sons(σ)} T^{σ',σ}^T u^σ'
  end
end;

procedure FastBackward(τ, (v^τ)_{τ∈C}, var v);
begin
  if sons(τ) = ∅ then v := v + V^τ v^τ
  else
    for τ' ∈ sons(τ) do begin
      v^τ' := v^τ' + T^{τ',τ} v^τ; FastBackward(τ', (v^τ), v)
    end
end
```

Figure 5: Fast forward and backward transformations

$$= \sum_{\tau' \in \operatorname{sons}(\tau)} \sum_{\nu' \in M} \mathcal{L}_\nu^\tau(\mathbf{x}_{\nu'}^{\tau'}) \mathsf{V}_{t\nu'}^{\tau'} = \sum_{\tau' \in \operatorname{sons}(\tau)} \sum_{\nu' \in M} \mathsf{T}_{\nu'\nu}^{\tau',\tau} \mathsf{V}_{t\nu'}^{\tau'} = \left(\sum_{\tau' \in \operatorname{sons}(\tau)} \mathsf{V}^{\tau'} \mathsf{T}^{\tau',\tau} \right)_{t\nu}.$$

We recall that χ_t is the characteristic function of the triangle t. Using these equations, we find the recursive procedures for the computation of $\mathbf{u}^\tau$ and $\mathbf{v}_{\mathrm{far}}$ given in Figure 5. Combining these procedures, we can derive the fast matrix-vector multiplication algorithm that is given in Figure 6.

Remark 5 (Storage) *The introduction of the transfer matrices $\mathsf{T}^{\tau',\tau}$ leads to a significant reduction in the amount of memory needed to store the $\mathcal{H}^2$-approximation of the matrix G: Since we are able to construct V^τ for all non-leaf clusters τ by using the transfer matrices, we need to store V^τ only for leaf clusters.*

This reduces the amount of storage required to store the $\mathcal{H}^2$-matrix approximation to $\mathcal{O}(nm^3)$ (cf. [2]).

3.4 Treatment of $b(\cdot,\cdot)$

Since the bilinear forms $m_{\Re}(\cdot,\cdot)$, $m_{\Im}(\cdot,\cdot)$ and $q(\cdot,\cdot)$ can be expressed by $g(\cdot,\cdot)$, we only have to treat $b(\cdot,\cdot)$ (cf. (9)) in order to be able to compress all matrices occurring in our boundary element formulation.

```
procedure MatrixVectorMultiplication(u, var v);
begin
  FastForward(T, u, (u^σ));
  for τ ∈ C do v^τ := Σ_{σ∈row(τ)} S^{τ,σ} u^σ;
  FastBackward(T, (v^τ), v_far);
  v := v_far + Σ_{(τ,σ)∈P_near} G^{τ,σ} u
end
```

Figure 6: Matrix-vector multiplication

Since this bilinear form is based on $\mathbf{grad}\,\Phi$ instead of Φ, we have to find a degenerate approximation of the derivatives of the kernel function. In order to keep our algorithm simple, we use the derivatives of the approximation $\tilde{\Phi}$ of Φ, i.e., replace $\mathbf{grad}\,\Phi$ by $\mathbf{grad}\,\tilde{\Phi}$. Please note that $\mathbf{grad}\,\tilde{\Phi}$ exists almost everywhere, since the local interpolants $\tilde{\Phi}^{\tau,\sigma}$ are polynomials and therefore differentiable.

We ignore the sparse parts of the bilinear form and use the same approach as before on the remainder, i.e., we replace the singularity function Φ by its local approximations $\tilde{\Phi}^{\tau,\sigma}$ for $(\tau,\sigma) \in P_{\mathrm{far}}$. This leads to the following local bilinear forms:

$$\begin{aligned}
\tilde{b}^{\tau,\sigma}(\eta,\mathbf{E}) &= \int_\tau \int_\sigma \langle \mathbf{curl}_\Gamma\, \eta(\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x})\rangle \langle \mathbf{grad}_x\, \tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}), \mathbf{n}(\mathbf{x})\rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \\
&\quad - \int_\tau \int_\sigma \langle \mathbf{curl}_\Gamma\, \eta(\mathbf{y}), \mathbf{n}(\mathbf{x})\rangle \langle \mathbf{grad}_x\, \tilde{\Phi}^{\tau,\sigma}(\mathbf{x},\mathbf{y}), \gamma_D \mathbf{E}(\mathbf{x})\rangle \,\mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \\
&= \sum_{\nu\in M} \sum_{\mu\in M} \mathsf{S}^{\tau,\sigma}_{\nu\mu} \int_{\Gamma_\tau} \int_{\Gamma_\sigma} \left(\sum_{l=1}^{3} (\mathbf{curl}_\Gamma\, \eta)_l(\mathbf{y}) (\gamma_D \mathbf{E})_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle \mathcal{L}^\sigma_\mu(\mathbf{y}) \right. \\
&\qquad \left. - \sum_{l=1}^{3} (\mathbf{curl}_\Gamma\, \eta)_l(\mathbf{y}) \mathbf{n}_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \gamma_D \mathbf{E}(\mathbf{x})\rangle \mathcal{L}^\sigma_\mu(\mathbf{y}) \right) \mathrm{d}\mathbf{y}\,\mathrm{d}\mathbf{x} \\
&= \sum_{l=1}^{3} \sum_{\nu\in M} \sum_{\mu\in M} \mathsf{S}^{\tau,\sigma}_{\nu\mu} \int_\sigma (\mathbf{curl}_\Gamma\, \eta)_l(\mathbf{y}) \mathcal{L}^\sigma_\mu(\mathbf{y})\,\mathrm{d}\mathbf{y} \\
&\qquad \cdot \int_\tau (\gamma_D \mathbf{E})_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle - \mathbf{n}_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \gamma_D \mathbf{E}(\mathbf{x})\rangle \,\mathrm{d}\mathbf{x}.
\end{aligned}$$

Apart from the summation over l, this representation is similar to that in (16), so we introduce matrices $\mathsf{V}^{\tau,l} \in \mathbb{R}^{\mathcal{E}\times M}$ and $\mathsf{W}^{\sigma,l} \in \mathbb{R}^{\mathcal{N}\times M}$ by setting

$$\mathsf{V}^{\tau,l}_{e\nu} := \int_\tau (\gamma_D \mathbf{b}_e)_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \mathbf{n}(\mathbf{x})\rangle - \mathbf{n}_l(\mathbf{x}) \langle \mathbf{grad}\, \mathcal{L}^\tau_\nu(\mathbf{x}), \gamma_D \mathbf{b}_e\rangle \,\mathrm{d}\mathbf{x},$$

$$\mathsf{W}_{v\mu}^{\sigma,l} := \int_\sigma (\mathbf{curl}_\Gamma \phi_v)_l(\mathbf{y}) \mathcal{L}_\mu^\sigma(\mathbf{y}) \,\mathrm{d}\mathbf{y}$$

and find

$$\tilde{\mathsf{B}}^{\tau,\sigma} = \sum_{l=1}^{3} \mathsf{V}^{\tau,l} \mathsf{S}^{\tau,\sigma} \mathsf{W}^{\sigma,l^\top},$$

where $\tilde{\mathsf{B}}^{\tau,\sigma}$ is the Galerkin discretization of the local bilinear form $\tilde{b}^{\tau,\sigma}(\cdot,\cdot)$. Using this representation, we can treat the approximation of $b(\cdot,\cdot)$ by exactly the same techniques as that of $g(\cdot,\cdot)$.

4 Implementation

4.1 Interpolation

We use tensor product Chebyshev interpolation, i.e., the interpolation points $(\mathbf{x}_\nu^\tau)_{\nu\in M}$ and $(\mathbf{y}_\mu^\sigma)_{\mu\in M}$ in equation (13) are the Chebyshev points for the axis-parallel boxes B^τ and B^σ.

The computation of these points is straightforward: The m-th order Chebyshev points $(x_i)_{i=0}^m$ for the interval $[-1,1]$ are given by

$$x_i := \cos\left(\pi \frac{2i+1}{2m+2}\right).$$

For a given interval $[a,b] \subseteq \mathbb{R}$, the transformed Chebyshev points $(x_i^{[a,b]})_{i=0}^m$ are

$$x_i^{[a,b]} := \frac{b+a}{2} + \frac{b-a}{2} x_i,$$

and the corresponding one-dimensional Lagrange polynomials have the form

$$\mathcal{L}_i^{[a,b]}(x) := \prod_{j\neq i} \frac{x - x_j^{[a,b]}}{x_i^{[a,b]} - x_j^{[a,b]}}.$$

The axis-parallel box B^τ can be written as $B^\tau = [a_1,b_1]\times[a_2,b_2]\times[a_3,b_3]$, so the tensor product Chebyshev points are given by

$$\mathbf{x}_\nu^\tau := (x_{\nu_1}^{[a_1,b_1]}, x_{\nu_2}^{[a_2,b_2]}, x_{\nu_3}^{[a_3,b_3]})$$

for $\nu = (\nu_1,\nu_2,\nu_3) \in M := \{\nu \in \mathbb{N}_0^3 \;:\; \|\nu\|_\infty \leq m\}$. The corresponding Lagrange polynomials are

$$\mathcal{L}_\nu^\tau := \mathcal{L}_{\nu_1}^{[a_1,b_1]} \otimes \mathcal{L}_{\nu_2}^{[a_2,b_2]} \otimes \mathcal{L}_{\nu_3}^{[a_3,b_3]},$$

so they can be evaluated efficiently for $\mathbf{x} = (x_1,x_2,x_3) \in \mathbb{R}^3$ due to

$$\mathcal{L}_\nu^\tau(\mathbf{x}) = \mathcal{L}_{\nu_1}^{[a_1,b_1]}(x_1) \mathcal{L}_{\nu_2}^{[a_2,b_2]}(x_2) \mathcal{L}_{\nu_3}^{[a_3,b_3]}(x_3).$$

4.2 Setup of the $\mathcal{H}^2$-matrices

In order to set up the $\mathcal{H}^2$-matrix approximation, we have to create the coefficient matrices $\mathsf{S}^{\tau,\sigma}$ for $(\tau,\sigma) \in P_{\mathrm{far}}$, the basis matrices V^τ for $\tau \in \mathcal{C}$ with $\mathrm{sons}(\tau) = \emptyset$, the transfer matrices $\mathsf{T}^{\tau',\tau}$ for $\tau \in \mathcal{C}$ with $\mathrm{sons}(\tau) \neq \emptyset$, and the near-field matrices $\mathsf{G}^{\tau,\sigma}$ for $(\tau,\sigma) \in P_{\mathrm{near}}$.

The computation of $\mathsf{S}^{\tau,\sigma}$ is straightforward: Using the interpolation points $(\mathbf{x}_\nu^\tau)_{\nu\in M}$ and $(\mathbf{x}_\mu^\sigma)_{\mu\in M}$ defined in Subsection 4.1, we have to evaluate the kernel function Φ:

$$\mathsf{S}_{\nu\mu}^{\tau,\sigma} = \Phi(\mathbf{x}_\nu^\tau, \mathbf{x}_\mu^\sigma).$$

The matrices V^τ satisfy

$$\mathsf{V}_{t\nu}^\tau = \int_\tau \chi_t(\mathbf{x}) \mathcal{L}_\nu^\tau(\mathbf{x})\,\mathrm{d}\mathbf{x} = \int_t \mathcal{L}_\nu^\tau(\mathbf{x})\,\mathrm{d}\mathbf{x},$$

since χ_t is the characteristic function of the triangle t. This integral can be computed by standard quadrature techniques, since the integrand $\mathcal{L}_\nu^\tau$ is a polynomial.

Let $B^\tau = [a_1,b_1] \times [a_2,b_2] \times [a_3,b_3]$ and $B^{\tau'} = [c_1,d_1] \times [c_2,d_2] \times [c_3,d_3]$. Then the transfer matrix $\mathsf{T}^{\tau',\tau}$ is given by

$$\mathsf{T}_{\nu'\nu}^{\tau',\tau} = \mathcal{L}_\nu^\tau(\mathbf{x}_{\nu'}^{\tau'}) = \mathcal{L}_{\nu_1}^{[a_1,b_1]}(x_{\nu'_1}^{[c_1,d_1]})\ \mathcal{L}_{\nu_2}^{[a_2,b_2]}(x_{\nu'_2}^{[c_2,d_2]})\ \mathcal{L}_{\nu_3}^{[a_3,b_3]}(x_{\nu'_3}^{[c_3,d_3]}),$$

i.e., is the Kronecker product of three matrices computed by evaluating one-dimensional Lagrange polynomials. This allows us to compute the matrices $\mathsf{T}^{\tau',\tau}$ efficiently. Alternatively, we can reduce the memory requirements by storing the Kronecker factors instead of the full matrix.

The near-field matrices $\mathsf{G}^{\tau,\sigma}$ are computed by a semi-analytical approach[1] for the double integrals with singular kernels, where the interior integral is calculated analytically and the exterior integral is evaluated by a Gaussian quadrature scheme. Some of the analytical integrations can be looked up in [9].

Remark 6 (Compact storage) *Let $t \in \mathcal{T}$ and $\tau \in \mathcal{C}$. If $\operatorname{supp}\chi_t \cap \Gamma_\tau = \emptyset$ holds, we have $\mathsf{V}_{t\nu}^\tau = 0$ for all $\nu \in K$.*

Let $t,s \in \mathcal{T}$ and $(\tau,\sigma) \in P_{\mathrm{near}}$. If $\operatorname{supp}\chi_t \cap \Gamma_\tau = \emptyset$ or $\operatorname{supp}\chi_s \cap \Gamma_\sigma = \emptyset$ hold, we have $\mathsf{G}_{ts}^{\tau,\sigma} = 0$.

This means that only the small number of non-zero entries of $\mathsf{G}^{\tau,\sigma}$ and V^τ has to be computed and stored.

4.3 Preconditioning

A *conjugate residual method* (CR) is used for solving equation (5), which needs to be preconditioned. We use the block diagonal preconditioner P given by

$$\mathsf{P} := \begin{pmatrix} \mathsf{M}_{\Re} + \mathsf{M}_{\Im} & 0 & 0 & 0 \\ 0 & \mathsf{M}_{\Re} + \mathsf{M}_{\Im} & 0 & 0 \\ 0 & 0 & \mathsf{Q} & 0 \\ 0 & 0 & 0 & \mathsf{Q} \end{pmatrix}, \tag{23}$$

[1] Private communication with Dr. Olaf Steinbach, University of Stuttgart.

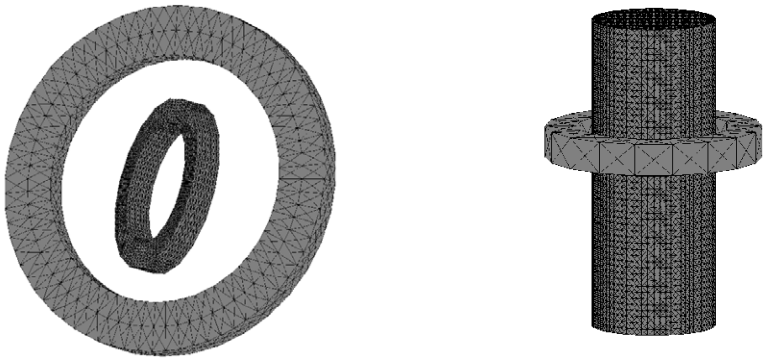

Figure 7: Test geometries A and B

which is approximately inverted by applying a *conjugate gradient method*. In the standard method, P is preconditioned by a *Jacobi preconditioner*.

All the operators of P, namely $\mathsf{M}_{\Re}$, $\mathsf{M}_{\Im}$ and Q, are also part of equation (5), thus are already compressed if the $\mathcal{H}^2$-matrix approximation is applied. So the $\mathcal{H}^2$-matrix approximation also pays for the preconditioner.

The near-field part Q_{near} of the operator Q can be inverted by a *sparse Gaussian elimination scheme* as long as the dimensions are moderate, giving us a preconditioner for the third and forth row of P.

5 Numerical experiments

The predicted behavior of the $\mathcal{H}^2$*-matrix approximation method* was tested for the three operators $\mathsf{M}_{\Re}$, Q, and B of equation (5) by using the typical geometry of the induction heating setting, as shown in Figure 7. The storage and cpu-time requirements for solving equation (5) with the uncompressed standard method and the interpolation-based $\mathcal{H}^2$-method were compared. The order of interpolation was set to be 2 and the simplified admissibility condition (21) with $\eta' = 0.99$ was used. This turned out to lead to sufficiently small approximation errors.

Differently from standard $\mathcal{H}^2$-techniques, degrees of freedom located in the edges and nodes can appear multiple times during the course of the matrix-vector multiplication due to the fact that our clustering technique is based on the triangles of the grid, not on the degrees of freedom. The compression rates in Figure 8 show that the performance of our method is still good.

The diagonal part of $\mathsf{M}_{\Re}$ consists of scalar products of linear edge functions, and it does not need to be compressed, because the functions have local support. The non-diagonal part is the scalar single layer potential of equation (11) with constant basis functions.

The functions appearing in the single layer potential of Q are vector-valued and con-

stant, but an adaptation of the matrices V to this fact is straightforward. The transfer matrices of equation (22) are identical for all occurring operators and therefore have to be calculated and stored only once.

The operator B is the most complicated one. It consists of a diagonal part, a double layer potential, and a slightly modified double layer potential. The basis functions are vector-valued and mixed, so we have to adapt the corresponding V-matrices once more.

All in all, the elaborate operators can be compressed without major difficulties by the $\mathcal{H}^2$-matrix approximation technique. A large number of matrices must be calculated only once and can be reused for different operators. This automatically saves a lot of storage and enhances the efficiency of the algorithm.

Table 8 shows the performance of the $\mathcal{H}^2$-matrix approximation compared with the use of dense matrices. The solver was stopped when the residual had decreased to a value below 0.0001 times the residual in the first step. The documented time is the time needed for filling the matrices and solving the system. The number of unknowns is defined as

$$n := 2 \times (\#\mathcal{E} + \#\mathcal{N} + \text{number of holes in the workpiece}),$$

and for the storage requirements one finds

$$\begin{aligned} \text{Standard storage} &= (2 \times (\#\mathcal{E})^2 + (\#\mathcal{E} \times \#\mathcal{N}) + (\#\mathcal{N})^2) \times \texttt{sizeof(double)} \text{ Bytes}, \\ \mathcal{H}^2 \text{ storage} &= \text{Storage needed for the } \mathcal{H}^2 \text{ approximation}. \end{aligned}$$

The relative error is defined as the difference between the surface current $\gamma_D \mathbf{j} := \sigma \cdot \gamma_D \mathbf{E}$ of the $\mathcal{H}^2$-solution and that of the solution obtained without compression

$$\text{relative error} := \int_\Gamma \frac{\|\gamma_D \mathbf{j}_{\mathcal{H}^2}(\mathbf{x}) - \gamma_D \mathbf{j}_{st.}(\mathbf{x})\|}{\|\gamma_D \mathbf{j}_{st.}(\mathbf{x})\|} \, dS_{\mathbf{x}} \,,$$

because the current is the most important entity for the calculation of the inductive heating.

The first four rows were produced by interpolation on the geometry A of Figure 7 and the last on geometry B. Storage requirements and calculation times are strongly reduced, and geometries consisting of 20000 surface faces can be calculated. In each case the time for filling the matrices amounts to 90% of the total time. The experiments were made on a Sun Ultra 450 computer with a 300 MHz Ultrasparc II processor.

A special feature is implemented due to the fact that the workpiece rotates. This means that the current has to be calculated for several positions of the workpiece per rotation. Therefore, it is desirable to find a clever way for reusing entities, which have already been calculated:

The BEM operators change only for items that are moving relative to each other. Parts of the operators which describe workpiece/workpiece interactions (w/w) or inductor/inductor interactions (i/i) need to be calculated only once. But how to reach this goal in the $\mathcal{H}^2$ context where everything is linked together in the tree? The option chosen here is to apply the geometric bisection algorithm (cf. Figure 3) separately to workpiece and inductor. Then each cluster consists exclusively of workpiece triangles or inductor

triangles. In this case, the matrices V, W and T do not change during the rotation and have to be calculated only once (except for the useless T of the root). After merging the two resulting trees under one big root, one finds the situation of Figure 9. If the block partition algorithm (cf. Figure 4) is now applied to this new tree, each pair of admissible clusters belongs either to w/w or i/i or to the interaction w/i between workpiece and inductor. The near-field and the matrices S have to be refreshed only for the w/i pairs. This is a big advantage because the interesting part with the biggest number of triangles is the workpiece, and w/w does not have to be refreshed.

References

[1] H. Ammari, A. Buffa, and J.-C. Nédélec, *A justification of eddy currents model for the Maxwell equations*, SIAM J. Appl. Math., 60 (2000), pp. 1805–1823.

[2] S. Börm and W. Hackbusch, *$\mathcal{H}^2$-matrix approximation of integral operators by interpolation*, Applied Numerical Mathematics, 43 (2002), pp. 129–143.

[3] H. Dirks, *Quasi–stationary fields for microelectronic applications*, Electrical Engineering, 79 (1996), pp. 145–155.

[4] K. Giebermann, *Multilevel representation of boundary integral operators*, Tech. Rep. 637, SFB 256, Universität Bonn, Bonn, Germany, 2000.

[5] L. Greengard and V. Rokhlin, *A fast algorithm for particle simulations*, Journal of Computational Physics, 73 (1987), pp. 325–348.

[6] W. Hackbusch, B. Khoromskij, and S. Sauter, *On $\mathcal{H}^2$-matrices*, in Lectures on Applied Mathematics, H. Bungartz, R. Hoppe, and C. Zenger, eds., 2000, pp. 9–29.

[7] W. Hackbusch and Z. Nowak, *On the fast matrix multiplication in the boundary element method by panel clustering*, Numer. Math., 54 (1989), pp. 463–491.

[8] R. Hiptmair, *Symmetric coupling for eddy current problems*, SIAM J. Numer. Anal., 40 (2002), pp. 41–65.

[9] J. Ostrowski, *Boundary Element Methods for Inductive Hardening*, PhD thesis, University of Tübingen, Germany, 2003. available under `http://w210.ub.uni-tuebingen.de/dbt/volltexte/2003/672`.

[10] V. Rokhlin, *Rapid solution of integral equations of classical potential theory*, Journal of Computational Physics, 60 (1985), pp. 187–207.

[11] S. Sauter, *Variable order panel clustering (extended version)*, Tech. Rep. 52, Max-Planck-Institut für Mathematik, Leipzig, Germany, 1999.

n	*Standard storage*	$\mathcal{H}^2$ *storage*	*Standard time*	$\mathcal{H}^2$ *time*	*Relative error*
2948	22.7 MB	15.5 MB	19.75 min	15.0 min	0.00139
6916	125.3 MB	35.0 MB	1.9 h	43.3 min	0.00250
11420	342.0 MB	71.0 MB	6.7 h	1.6 h	0.00146
23840	954.1 MB	93.4 MB	14.6 h	2.4 h	0.00218
46724	5725.0 MB	333.0 MB	-	9.0 h	-

Figure 8: Time, storage, and errors for the impedance model

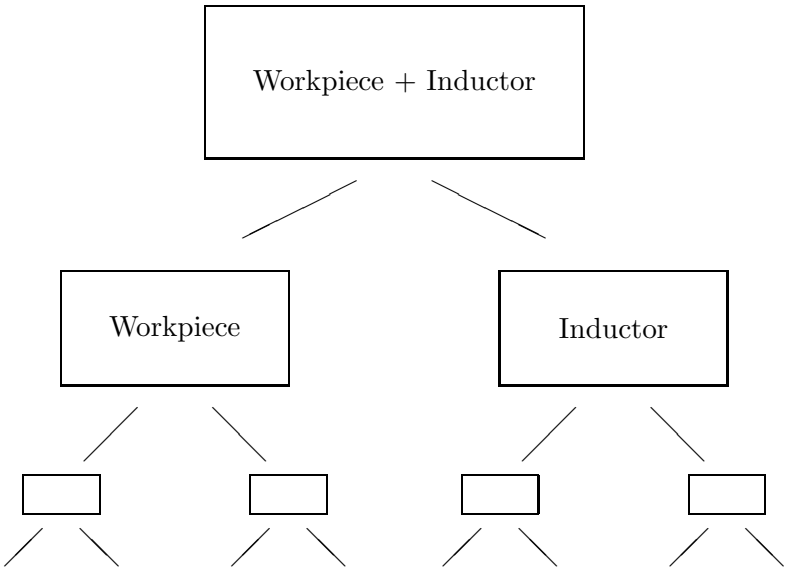

Figure 9: Cluster tree for the inductor/workpiece domain