An interpolation-based fast multipole method for higher order boundary elements on parametric surfaces

J. Dölz, H. Harbrecht, M. Peters

Departement Mathematik und Informatik
Fachbereich Mathematik
Universität Basel
CH-4051 Basel

Preprint No. 2015-20
June 2015

www.math.unibas.ch

An interpolation-based fast multipole method for higher order boundary elements on parametric surfaces[☆]

J. Dölz[a], H. Harbrecht[a], M. Peters[a]

[a] *University of Basel, Department of Mathematics and Computer Science, Spiegelgasse 1, 4051 Basel, Switzerland*

Abstract

In this article, we propose a black-box higher order fast multipole method for solving boundary integral equations on parametric surfaces in three dimensions. Such piecewise smooth surfaces are the topic of recent studies in isogeometric analysis. Due to the exact surface representation, the rate of convergence of higher order methods is not limited by approximation errors of the surface. An element-wise clustering yields a balanced cluster tree and an efficient numerical integration scheme for the underlying Galerkin method. By performing the interpolation for the fast multipole method directly on the reference domain, we reduce the cost complexity in the polynomial degree by one order. This gain is independent of the application of either $\mathcal{H}$- or $\mathcal{H}^2$-matrices. In fact, we point out several simplifications in the construction of $\mathcal{H}^2$-matrices, which are a by-product of the surface representation. Numerical examples are provided in order to quantify and qualify the proposed method.

Keywords: Non-local operators, parametric surfaces, higher order ansatz functions, $\mathcal{H}^2$-matrices, fast multipole method.

1. Introduction

In many situations, practical problems arising from science and engineering can be formulated in terms of differential equations for an unknown function. If a Green's function of the underlying differential equation is known, it may be reformulated by means of boundary integral equations. A Green's function is, for instance, known in case of the Laplace equation, the Helmholtz equation and the heat equation. The main advantage of considering boundary integral equations is the reduction of the problem's dimensionality.

Different approaches have been proposed to deal with the resulting, in general non-local, boundary integral operators. Beside collocation and Nyström

[☆] This research has been supported by the Swiss National Science Foundation (SNSF) through the project "H-matrix based first and second moment analysis".

Email addresses: Juergen.Doelz@unibas.ch (J. Dölz), Helmut.Harbrecht@unibas.ch (H. Harbrecht), Michael.Peters@unibas.ch (M. Peters)

methods, the boundary element method (BEM) is commonly used for the numerical discretization of such operators, see [1, 2, 3]. Due to their non-locality, one usually ends up with large and densely populated system matrices and, thus, the numerical solution of such problems is rather challenging.

Nevertheless, the system matrices exhibit very often a certain compressibility property. Therefore, in the last decades, several ideas for the efficient approximation of the discrete linear system of equations have been developed which exploit this compressibility. The most prominent examples of such methods are the fast multipole method [4], the panel clustering method [5], the wavelet Galerkin scheme [6, 7], and the adaptive cross approximation [8]. These discretization methods end up with linear or almost linear complexity, i.e. up to a poly-logarithmic factor, with respect to the number of boundary elements.

In this article, we focus on the fast multipole method (FMM) for the solution of boundary integral equations and cast it into the framework of *parametric surfaces*. Parametric surfaces can be described piecewise by the images of a certain reference domain under smooth diffeomorphisms. The images of each of these diffeomorphisms are referred to as *patches*.

Many parametric surfaces are nowadays directly accessible as technical surfaces generated by tools from Computer Aided Design (CAD). Very common surface representations in CAD are defined by the IGES (Initial Graphics Exchange Specification) and the STEP (Standard for the Exchange of Product Model Data) standards, cf. [9, 10]. In both standards, the initial CAD object is a solid, bounded by a closed surface that is given as a collection of parametric surfaces which can be trimmed or untrimmed. An untrimmed surface is already a four-sided patch, parameterized over a rectangle. Whereas, a trimmed surface is just a piece of a supporting untrimmed surface, described by boundary curves. There are several representations of the parameterizations including B-splines, NURBS (nonuniform rational B-Splines), surfaces of revolution, and tabulated cylinders, see [11]. The representation with NURBS is intensively studied in the context of isogeometric analysis, see e.g. [12, 13, 14, 15]. Nevertheless, in contrast to the isogeometric analysis framework, we do not restrict ourselves to geometries that can be represented by NURBS, but allow any surface which provides the requirements specified in the subsequent section.

One major advantage of parametric surfaces stems from the fact that more geometric information is available, which can therefore be exploited in the discretization. Especially, no difficulties arise if geometric entities occur in the kernel function of the integral operator under consideration, like the normal or tangents, as for example in the double layer operator or the adjoint double layer operator. Moreover, parametric surfaces provide an exact representation of the surface which is in contrast to the common approximation of surfaces by panels. There is no further approximation step required if the surface is given in this form. As a consequence, the rate of convergence in a boundary element method is not limited by the accuracy of the surface approximation.

We shall provide in this article a simple black-box version of the fast multipole method for higher order boundary elements in order to make use of the features of parametric surfaces. In particular, we interpolate the Green's func-

tion directly on the reference domain. This is in contrast to the interpolation of the kernel in space, as in e.g. [16, 17], and yields a remarkable speed-up of the FMM, since we can fully exploit the dimension reduction due to the boundary integral formulation of the underlying problem. In three spatial dimensions, the surface is a two-dimensional manifold and so the problem is inherently two-dimensional. This results in a dramatic reduction of the computational effort. Moreover, we can still profit from the $\mathcal{H}^2$-matrix techniques presented in e.g. [16, 17]. Notably, since our particular realization of parametric surfaces is based on four-sided patches, we can exploit the tensor product structure of the reference domain to considerably simplify the construction of $\mathcal{H}^2$-matrices. More precisely, we will see that, due to the special structure of our computational domain, the construction of $\mathcal{H}^2$-matrices only slightly differs from that of usual $\mathcal{H}$-matrices. A further specialty of the presented FMM is that it can also be regarded as black-box algorithm for the discretization of more general Hilbert-Schmidt operators, see e.g. [18]. Then, the Green's function function is replaced by a more general integral kernel. In particular, there is no explicit knowledge of the integral kernel presumed except for its smoothness apart from the diagonal.

The rest of this article is structured as follows. At first, in Section 2, we introduce the parametric surface representation under consideration. As a consequence from this representation, the mesh generation is straightforward. In Section 3, we discuss boundary integral equations together with their properties in general. The respective Galerkin discretization with piecewise polynomial (discontinuous) ansatz functions is performed in Section 4. Then, Section 5 is dedicated to the FMM for parametric surfaces. Here, we present the algorithm which perfectly fits the framework of parametric surfaces and extend it to the $\mathcal{H}^2$-matrix variant. In Section 6, we provide a straightforward extension of the FMM to higher order continuous ansatz functions. Finally, in Section 7, we perform numerical experiments to validate and quantify our numerical approach.

In the sequel, in order to avoid the repeated use of generic but not further specified constants, we imply by $C \lesssim D$ that C can be bounded by a multiple of D, independently of other parameters which C and D may depend on. Obviously, $C \gtrsim D$ is defined as $D \lesssim C$, and $C \sim D$ as $C \lesssim D$ and $C \gtrsim D$.

2. Surface Representation

Let $\Omega \subset \mathbb{R}^3$ denote a Lipschitz domain with piecewise smooth surface $\Gamma := \partial\Omega$. Then, we construct a parametric representation of the surface Γ as follows. Let $\square := [0,1]^2$ denote the unit square, which serves as reference domain. We subdivide the given surface Γ into several smooth *patches*

$$\Gamma = \bigcup_{i=1}^{M} \Gamma_i,$$

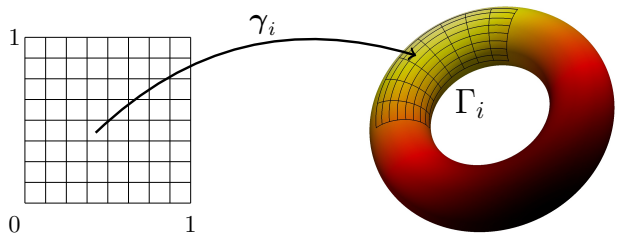


Figure 2.1: Surface representation and mesh generation.

where the intersection $\Gamma_i \cap \Gamma_{i'}$ consists at most of a common vertex or a common edge for $i \neq i'$. Then, for each patch, there exists a smooth diffeomorphism

$$\boldsymbol{\gamma}_i : \square \to \Gamma_i \quad \text{with} \quad \Gamma_i = \boldsymbol{\gamma}_i(\square) \quad \text{for } i = 1, 2, \ldots, M, \tag{2.1}$$

as illustrated in Figure 2.1. For constructing regular surface meshes, we impose the following *matching condition*: We demand the existence of a bijective and affine mapping $\boldsymbol{\Xi}\colon \square \to \square$ such that for each $\mathbf{x} = \boldsymbol{\gamma}_i(\mathbf{s})$ on a common edge of Γ_i and $\Gamma_{i,'}$ there holds holds $\boldsymbol{\gamma}_i(\mathbf{s}) = (\boldsymbol{\gamma}_{i'} \circ \boldsymbol{\Xi})(\mathbf{s})$. This means that the parameterizations $\boldsymbol{\gamma}_i$ and $\boldsymbol{\gamma}_{i'}$ coincide on the common edge except for orientation.

In the sequel, we shall also refer to the *surface measure* of the diffeomorphisms $\boldsymbol{\gamma}_i$. On the patch Γ_i, it is given by

$$\kappa_i(\mathbf{s}) := \left\| \partial_{s_1} \boldsymbol{\gamma}_i(\mathbf{s}) \times \partial_{s_2} \boldsymbol{\gamma}_i(\mathbf{s}) \right\|_2. \tag{2.2}$$

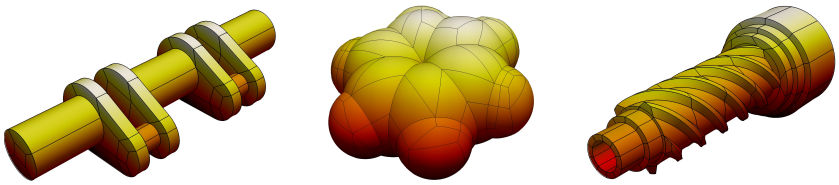


Figure 2.2: Different parametric surfaces with their patch boundaries.

An algorithm to decompose a technical surface, described in the IGES format, into a collection of parameterized four-sided patches, fulfilling all the above requirements, has been proposed in [19]. This algorithm has been extended in [20, 21] to molecular surfaces. Figure 2.2 visualizes three parameterizations which satisfy the present requirements.

Starting from this surface representation, it is straightforward to generate a nested sequence of meshes for Γ. The mesh $\mathcal{Q}_j$ on level j for Γ is induced by dyadic subdivisions of depth j of the unit square into 4^j congruent squares,

each of which is lifted to Γ by the associated parameterization $\boldsymbol{\gamma}_i$ (see Figure 2.1 for a visualization). This procedure leads to a nested and especially quad-tree structured sequence

$$\mathcal{Q}_0 \subset \mathcal{Q}_1 \subset \cdots \subset \mathcal{Q}_J$$

of meshes consisting of $N_j = 4^j M$ *elements* on level j.

We will refer to the particular elements as $\Gamma_{i,j,k}$ where i is the index of the underlying parameterization $\boldsymbol{\gamma}_i$, j denotes the level of the element and k is the index of the element in hierarchical order. For notational convenience we shall also refer to the triple (i,j,k) by $\boldsymbol{\lambda} := (i,j,k)$ with $|\boldsymbol{\lambda}| := j$. In view of the fast multipole method, we will consider $\Gamma_{i,j,k}$ also as a *cluster*. In this sense, we think of $\Gamma_{i,j,k}$ as the union $\{\Gamma_{i,J,k'} \colon \Gamma_{i,J,k'} \subset \Gamma_{i,j,k}\}$, i.e. the set of all tree leafs appended to $\Gamma_{i,j,k}$ or one of its sons. Furthermore, we denote the hierarchical ordered collection of all clusters, the *cluster tree*, by $\mathcal{T}$. A scheme for the subdivisions of the patch Γ_i up to level 2 is illustrated in Figure 2.3. Finally, with respect to the tree structure of $\mathcal{T}$, we define $\operatorname{dad}(\boldsymbol{\lambda}) := (i, j-1, \lfloor k/4 \rfloor)$ and $\operatorname{sons}(\boldsymbol{\lambda}) := \{(i, j+1, 4k+\ell) \colon \ell = 0, \ldots, 3\}$.

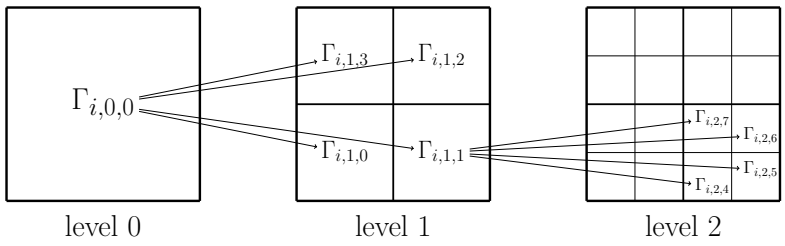


Figure 2.3: Visualization of the cluster tree.

3. Problem Formulation

In this article, we focus on boundary integral equations defined on the closed, parametric surface $\Gamma := \partial\Omega$, i.e.

$$(\mathcal{A}u)(\mathbf{x}) = \int_\Gamma k(\mathbf{x},\mathbf{y}) u(\mathbf{y}) \, \mathrm{d}\sigma_{\mathbf{y}} = f(\mathbf{x}). \tag{3.3}$$

Herein, the boundary integral operator $\mathcal{A}$ is supposed to be of order $2q$, which means that it maps $H^q(\Gamma)$ continuously and one-to-one onto $H^{-q}(\Gamma)$. The kernel functions under consideration have to be smooth as functions in the variables $\mathbf{x}$ and $\mathbf{y}$ apart from the diagonal $\{(\mathbf{x},\mathbf{y}) \in \Gamma \times \Gamma : \mathbf{x} = \mathbf{y}\}$ and may have a singularity on the diagonal. Such kernel functions arise, for instance, by applying a boundary integral formulation to a second order elliptic boundary value problem, see e.g. [2, 3]. In general, they decay like a negative power of the distance of the arguments which depends on the order $2q$ of the operator and the

spatial dimension. More precisely, we suppose that the kernel is *asymptotically smooth*, i.e. for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ holds

$$\left|\partial_{\mathbf{x}}^{\boldsymbol{\alpha}} \partial_{\mathbf{y}}^{\boldsymbol{\beta}} k(\mathbf{x}, \mathbf{y})\right| \leq c_k \frac{(|\boldsymbol{\alpha}| + |\boldsymbol{\beta}|)!}{r_k^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}} \|\mathbf{x} - \mathbf{y}\|_2^{-2-2q-|\boldsymbol{\alpha}|-|\boldsymbol{\beta}|} \tag{3.4}$$

with some constants $c_k > 0$ and $r_k > 0$ which are independent of $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$.

The variational formulation of the boundary integral equation (3.3) reads now as follows:

$$\text{Find } u \in H^q(\Gamma) \text{ such that } (\mathcal{A}u, v)_{L^2(\Gamma)} = (f, v)_{L^2(\Gamma)} \text{ for all } v \in H^q(\Gamma). \tag{3.5}$$

If we insert the parametric representation (2.1) of Γ, the bilinear form reads

$$\begin{aligned}(\mathcal{A}u, v)_{L^2(\Gamma)} &= \int_\Gamma \int_\Gamma k(\mathbf{x}, \mathbf{y}) u(\mathbf{y}) v(\mathbf{x}) \, \mathrm{d}\sigma_{\mathbf{y}} \, \mathrm{d}\sigma_{\mathbf{x}} \\ &= \sum_{i,i'=1}^{M} \int_\square \int_\square k_{i,i'}(\mathbf{s}, \mathbf{t}) u\big(\boldsymbol{\gamma}_{i'}(\mathbf{t})\big) v\big(\boldsymbol{\gamma}_i(\mathbf{s})\big) \, \mathrm{d}\mathbf{t} \, \mathrm{d}\mathbf{s}\end{aligned}$$

and the linear form reads

$$\begin{aligned}(f, v)_{L^2(\Gamma)} &= \int_\Gamma f(\mathbf{x}) v(\mathbf{x}) \, \mathrm{d}\sigma_{\mathbf{x}} \\ &= \sum_{i=1}^{M} \int_\square f\big(\boldsymbol{\gamma}_i(\mathbf{s})\big) v\big(\boldsymbol{\gamma}_i(\mathbf{s})\big) \kappa_i(\mathbf{s}) \, \mathrm{d}\mathbf{s}.\end{aligned}$$

Here, the kernels $k_{i,i'}$ denote the *transported kernel functions*

$$\left.\begin{aligned}&k_{i,i'} \colon \square \times \square \to \mathbb{R}, \\ &k_{i,i'}(\mathbf{s}, \mathbf{t}) := k\big(\boldsymbol{\gamma}_i(\mathbf{s}), \boldsymbol{\gamma}_{i'}(\mathbf{t})\big) \kappa_i(\mathbf{s}) \kappa_{i'}(\mathbf{t})\end{aligned}\right\} \quad i, i' = 1, 2, \ldots, M. \tag{3.6}$$

Definition 3.1. *A kernel function $k(\mathbf{x}, \mathbf{y})$ is called* analytically standard *of order $2q$ if constants $c_k > 0$ and $r_k > 0$ exist such that the partial derivatives of the transported kernel functions $k_{i,i'}(\mathbf{s}, \mathbf{t})$ are uniformly bounded by*

$$\left|\partial_{\mathbf{s}}^{\boldsymbol{\alpha}} \partial_{\mathbf{t}}^{\boldsymbol{\beta}} k_{i,i'}(\mathbf{s}, \mathbf{t})\right| \leq c_k \frac{(|\boldsymbol{\alpha}| + |\boldsymbol{\beta}|)!}{r_k^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}} \big\|\boldsymbol{\gamma}_i(\mathbf{s}) - \boldsymbol{\gamma}_{i'}(\mathbf{t})\big\|_2^{-(2+2q+|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)} \tag{3.7}$$

provided that $2 + 2q + |\boldsymbol{\alpha}| + |\boldsymbol{\beta}| > 0$.

Note that, since the parametric representation is patch-wise smooth, all kernels which satisfy (3.4) are also analytically standard of order $2q$, see e.g. [22] for a proof of this statement.

In the context of the Galerkin approximation, we will also refer to the *localized kernel functions*. To that end, let $\square_{j,k} := \boldsymbol{\gamma}_i^{-1}(\Gamma_{i,j,k})$ be the k-th element of the subdivided unit square on level j and define the affine mapping

$$\boldsymbol{\tau}_{j,k} \colon \square \to \square_{j,k} \quad \text{for } j = 0, 1, \ldots, J \text{ and } k = 0, 1, \ldots, 4^j M - 1$$

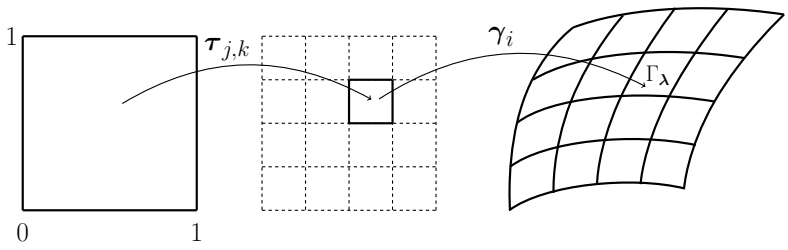


Figure 3.4: Localized parameterization.

via dilatation and translation. Then, the localized kernel functions are given by

$$k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{s},\mathbf{t}) \coloneqq k\big(\boldsymbol{\gamma}_{\boldsymbol{\lambda}}(\mathbf{s}),\boldsymbol{\gamma}_{\boldsymbol{\lambda}'}(\mathbf{t})\big)\kappa_{\boldsymbol{\lambda}}(\mathbf{s})\kappa_{\boldsymbol{\lambda}'}(\mathbf{t}) \tag{3.8}$$

with the *localized parameterizations* $\boldsymbol{\gamma}_{\boldsymbol{\lambda}} \coloneqq \boldsymbol{\gamma}_i \circ \boldsymbol{\tau}_{j,k}$ and the corresponding surface measures $\kappa_{\boldsymbol{\lambda}} \coloneqq 2^{-2j}\kappa_i \circ \boldsymbol{\tau}_{j,k}$ with κ_i defined in (2.2). An illustration of the mappings $\boldsymbol{\gamma}_{\boldsymbol{\lambda}}$ is given by Figure 3.4.

In the following, we will only consider the localized kernel functions. The next theorem is an immediate consequence of the definition (3.8) and the fact that $\partial_{\mathbf{s}}^{\boldsymbol{\alpha}}\boldsymbol{\tau}_{j,k}(\mathbf{s}) = 2^{-j}$ if $|\boldsymbol{\alpha}| = 1$ and $\partial_{\mathbf{s}}^{\boldsymbol{\alpha}}\boldsymbol{\tau}_{j,k}(\mathbf{s}) = 0$ if $|\boldsymbol{\alpha}| > 1$.

Theorem 3.2. *Let the kernel function $k(\mathbf{x},\mathbf{y})$ be analytically standard of order $2q$. Then, there exist constants $c_k > 0$ and $r_k > 0$ such that*

$$\big|\partial_{\mathbf{s}}^{\boldsymbol{\alpha}}\partial_{\mathbf{t}}^{\boldsymbol{\beta}}k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{s},\mathbf{t})\big| \leq c_k\frac{(|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|)!}{r_k^{|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}}\frac{2^{-|\boldsymbol{\lambda}|(|\boldsymbol{\alpha}|+2)}2^{-|\boldsymbol{\lambda}'|(|\boldsymbol{\beta}|+2)}}{\|\boldsymbol{\gamma}_{\boldsymbol{\lambda}}(\mathbf{s})-\boldsymbol{\gamma}_{\boldsymbol{\lambda}'}(\mathbf{t})\|_2^{2+2q+|\boldsymbol{\alpha}|+|\boldsymbol{\beta}|}} \tag{3.9}$$

holds uniformly for all $\boldsymbol{\lambda},\boldsymbol{\lambda}'$ provided that $2+2q+|\boldsymbol{\alpha}|+|\boldsymbol{\beta}| > 0$.

4. Galerkin Discretization

In this section, we consider the Galerkin discretization of the variational formulation (3.5). To this end, we fix a polynomial order $d \in \mathbb{N}$, a level of refinement $j \in \mathbb{N}$, and define the ansatz space

$$\hat{V}_j \coloneqq \big\{\hat{\varphi}\colon \square \to \mathbb{R}\colon \hat{\varphi}|_{\square_{j,k}} \text{ is a polynomial of order } d\big\} \subset L^2(\square) \tag{4.10}$$

of discontinuous, element-wise polynomial ansatz functions on the reference domain. With the help of this space, we can introduce the ansatz space V_j in accordance with

$$V_j \coloneqq \big\{\hat{\varphi}\circ\boldsymbol{\gamma}_i^{-1} : \hat{\varphi} \in \hat{V}_j,\ i = 1,\dots,M\big\} \subset L^2(\Gamma).$$

This construction of the ansatz spaces obviously yields a nested sequence

$$V_0 \subset V_1 \subset \cdots \subset V_J \subset H^t(\Gamma), \tag{4.11}$$

where the Sobolev smoothness t depends on the global smoothness of the functions $\varphi \in V_j$. For arbitrary functions $\varphi \in V_j$, we have $t < 1/2$, and for the subset of globally continuous functions in V_j, we have $t < 3/2$.

By replacing the energy space $H^q(\Gamma)$ in the variational formulation (3.5) by the finite dimensional ansatz space $V_J \subset H^q(\Gamma)$, we arrive at the Galerkin discretization for the boundary integral equation (3.3):

Find $u_J \in V_J$, such that

$$\int_\Gamma \int_\Gamma k(\mathbf{x},\mathbf{y}) u_J(\mathbf{y}) v_J(\mathbf{x}) \, \mathrm{d}\sigma_{\mathbf{y}} \, \mathrm{d}\sigma_{\mathbf{x}} = \int_\Gamma f(\mathbf{x}) v_J(\mathbf{x}) \, \mathrm{d}\sigma_{\mathbf{x}} \text{ for all } v_J \in V_J. \tag{4.12}$$

By setting $\hat{u}_{\boldsymbol{\lambda}} := u_J \circ \boldsymbol{\gamma}_{\boldsymbol{\lambda}}$ and $\hat{v}_{\boldsymbol{\lambda}} := v_J \circ \boldsymbol{\gamma}_{\boldsymbol{\lambda}}$, we can rewrite (4.12) and arrive at the equation

$$\sum_{|\boldsymbol{\lambda}'|=J} \int_\square \int_\square k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{s},\mathbf{t}) \hat{u}_{\boldsymbol{\lambda}'}(\mathbf{t}) \hat{v}_{\boldsymbol{\lambda}}(\mathbf{s}) \, \mathrm{d}\mathbf{t} \, \mathrm{d}\mathbf{s} = \int_\square f\big(\boldsymbol{\gamma}_{\boldsymbol{\lambda}}(\mathbf{s})\big) \hat{v}_{\boldsymbol{\lambda}}(\mathbf{s}) \kappa_{\boldsymbol{\lambda}}(\mathbf{s}) \, \mathrm{d}\mathbf{s} \tag{4.13}$$

for all $\boldsymbol{\lambda}$ with $|\boldsymbol{\lambda}| = J$.

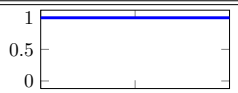
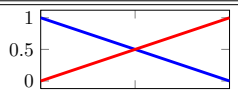
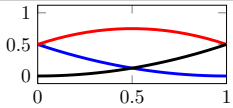
d	Shape Functions	Visualization
1	$\phi^{(i)}(x) = 1$	
2	$\phi^{(i)}(x) = \begin{cases} 1-x \\ x \end{cases}$	
3	$\phi^{(i)}(x) = \begin{cases} (1-x)^2/2 \\ -(x-1/2)^2 + 3/4 \\ x^2/2 \end{cases}$	

Table 4.1: B-spline based shape functions on the interval.

A basis for V_J is obtained by tensorizing polynomial shape functions on $[0,1]$ and applying the localized parameterizations $\boldsymbol{\gamma}_{\boldsymbol{\lambda}}$. For $d = 1, 2, 3$, suitable shape functions are depicted in Table 4.1. By choosing such a basis, (4.12) immediately yields a system of linear equations:

$$\mathbf{A}_J \mathbf{u}_J = \mathbf{f}_J. \tag{4.14}$$

To realize globally continuous B-splines as ansatz functions, enabling for example the discretization of the hypersingular integral operator, we shall apply suitable transformation matrices. The construction of these transformation matrices is the topic of Section 6.

Having the Galerkin solution $u_J \in V_J$ at hand, we obtain the following well known error estimate by the use of the standard approximation property for ansatz functions of polynomial order d. Note that the rate of convergence doubles due to the Aubin-Nitsche lemma.

Theorem 4.1. *Let $u \in H^q(\Gamma)$ be the solution of the boundary integral equation* (3.3) *and $u_J \in V_J$ the related Galerkin solution of* (4.12)*. Then, there holds the error estimate*

$$\|u - u_J\|_{H^{2q-d}(\Gamma)} \lesssim 2^{2J(q-d)} \|u\|_{H^d(\Gamma)}$$

provided that u and Γ are sufficiently regular.

5. Fast Multipole Method

In general, the system matrix $\mathbf{A}_J$ in (4.14) is densely populated. This yields a rather high computational effort for the assembly and for the matrix-vector multiplication. Fortunately, the system matrix is block-wise of low rank, i.e. it is compressible in terms of an $\mathcal{H}$*-matrix*, cf. [23]. The computational complexity can thus drastically be reduced by a block-wise compression scheme.

5.1. Block-Cluster Tree

For constructing the $\mathcal{H}$-matrix representation, we consider the level-wise Cartesian product $\mathcal{T} \boxtimes \mathcal{T} := \big\{\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'} \colon \Gamma_{\boldsymbol{\lambda}}, \Gamma_{\boldsymbol{\lambda}'} \in \mathcal{T}, |\boldsymbol{\lambda}| = |\boldsymbol{\lambda}'|\big\}$ of the cluster tree $\mathcal{T}$. Compressible matrix blocks are then identified by the following *admissibility condition*.

Definition 5.1. *The clusters $\Gamma_{\boldsymbol{\lambda}}$ and $\Gamma_{\boldsymbol{\lambda}'}$ with $|\boldsymbol{\lambda}| = |\boldsymbol{\lambda}'|$ are called* admissible *if*

$$\max\big\{\operatorname{diam}(\Gamma_{\boldsymbol{\lambda}}), \operatorname{diam}(\Gamma_{\boldsymbol{\lambda}'})\big\} \leq \eta \operatorname{dist}(\Gamma_{\boldsymbol{\lambda}}, \Gamma_{\boldsymbol{\lambda}'}) \tag{5.15}$$

holds for a fixed $\eta \in (0,1)$. The largest collection of admissible blocks $\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'} \in \mathcal{T} \boxtimes \mathcal{T}$ such that $\Gamma_{\operatorname{dad}(\boldsymbol{\lambda})} \times \Gamma_{\operatorname{dad}(\boldsymbol{\lambda}')}$ is not admissible forms the far-field *$\mathcal{F} \subset \mathcal{T} \boxtimes \mathcal{T}$ of the operator. The remaining non-admissible blocks correspond to the* near-field *$\mathcal{N} \subset \mathcal{T} \boxtimes \mathcal{T}$ of the operator.*

The far-field corresponds to the compressible matrix blocks, whereas the near-field is treated by the classical boundary element method.

The *block-cluster tree* $\mathcal{B} := \mathcal{F} \cup \mathcal{N}$ can be constructed by Algorithm 1. It induces a block partitioning of the system matrix in quadratic blocks since the cluster tree $\mathcal{T}$ is a balanced quad-tree. Hence, each block on a particular level contains exactly the same number of element-element interactions, see also Figure 5.5 for a visualization of this special block partitioning of an $\mathcal{H}$-matrix. Such a special structure is not available in general, cf. [23], and will explicitly be exploited in our construction of the fast multipole method (FMM).

Algorithm 1 Construction of the block-cluster tree $\mathcal{B}$

procedure BUILDBLOCKCLUSTERTREE(cluster $\Gamma_{\boldsymbol{\lambda}}, \Gamma_{\boldsymbol{\lambda}'}$)
  if $(\Gamma_{\boldsymbol{\lambda}}, \Gamma_{\boldsymbol{\lambda}'})$ is admissible **then**
    $\text{sons}(\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'}) := \emptyset$
  else
    $\text{sons}(\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'}) := \{\Gamma_{\boldsymbol{\mu}} \times \Gamma_{\boldsymbol{\mu}'} : \boldsymbol{\mu} \in \text{sons}(\boldsymbol{\lambda}), \boldsymbol{\mu}' \in \text{sons}(\boldsymbol{\lambda}')\}$
    for $\boldsymbol{\mu} \in \text{sons}(\boldsymbol{\lambda}), \boldsymbol{\mu}' \in \text{sons}(\boldsymbol{\lambda}')$ **do**
      BUILDBLOCKCLUSTERTREE($\Gamma_{\boldsymbol{\mu}}$,$\Gamma_{\boldsymbol{\mu}'}$)
    end for
  end if
end procedure

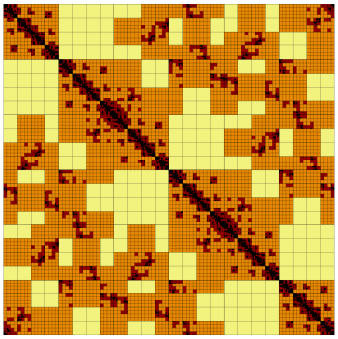


Figure 5.5: The special block partitioning of the $\mathcal{H}$-matrix.

5.2. Kernel Interpolation

To compress the admissible matrix blocks, we propose a black-box version of the FMM based on the interpolation of the kernel $k(\mathbf{x}, \mathbf{y})$ as firstly proposed in [16]. Note that, later on, this idea was also followed in [17] to construct $\mathcal{H}^2$-*matrices*. Nevertheless, in contrast to these works, our approach interpolates the localized kernel (3.8) on the reference domain rather than the original kernel in space.

For a given polynomial degree $p \in \mathbb{N}$, let $\{x_0, x_1, \ldots, x_p\} \subset [0, 1]$ denote $p+1$ interpolation points. Furthermore, let $L_m(s)$ for $m = 0, \ldots, p$ be the Lagrangian basis polynomials with respect to these interpolation points. By a tensor product construction, we obtain the interpolation points $\mathbf{x}_{\mathbf{m}} := (x_{m_1}, x_{m_2})$ and the corresponding tensor product basis polynomials $\mathbf{L}_{\mathbf{m}}(\mathbf{s}) := L_{m_1}(s_1) \cdot L_{m_2}(s_2)$ for $m_1, m_2 = 0, \ldots, p$. In all admissible blocks $\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'} \in \mathcal{F}$, we approximate

$$k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{s}, \mathbf{t}) \approx \sum_{\|\mathbf{m}\|_\infty, \|\mathbf{m}'\|_\infty \leq p} k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{x}_{\mathbf{m}}, \mathbf{x}_{\mathbf{m}'}) \mathbf{L}_{\mathbf{m}}(\mathbf{s}) \mathbf{L}_{\mathbf{m}'}(\mathbf{t}).$$

Hence, for two basis functions $\hat{\phi}_\ell, \hat{\phi}_{\ell'} \in \hat{V}_{J-|\boldsymbol{\lambda}|}$ of the ansatz space on level

$J - |\boldsymbol{\lambda}|$, we derive the representation

$$\begin{aligned}[\mathbf{A}_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}]_{\ell,\ell'} &\approx \int_{\square}\int_{\square} \sum_{\|\mathbf{m}\|_\infty,\|\mathbf{m}'\|_\infty \le p} k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{x}_{\mathbf{m}},\mathbf{x}_{\mathbf{m}'})\mathbf{L}_{\mathbf{m}}(\mathbf{s})\mathbf{L}_{\mathbf{m}'}(\mathbf{t})\hat{\phi}_\ell(\mathbf{s})\hat{\phi}_{\ell'}(\mathbf{t})\,\mathrm{d}\mathbf{t}\,\mathrm{d}\mathbf{s} \\ &= \sum_{\|\mathbf{m}\|_\infty,\|\mathbf{m}'\|_\infty \le p} k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{x}_{\mathbf{m}},\mathbf{x}_{\mathbf{m}'}) \int_{\square}\mathbf{L}_{\mathbf{m}}(\mathbf{s})\hat{\phi}_\ell(\mathbf{s})\,\mathrm{d}\mathbf{s}\int_{\square}\mathbf{L}_{\mathbf{m}'}(\mathbf{t})\hat{\phi}_{\ell'}(\mathbf{t})\,\mathrm{d}\mathbf{t} \\ &=: \left[\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|}\mathbf{K}_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{M}^{\square}_{|\boldsymbol{\lambda}'|})^\intercal\right]_{\ell,\ell'}.\end{aligned}$$

By construction, each cluster on a particular level contains the same number of basis functions, namely $\dim(\hat{V}_{J-|\boldsymbol{\lambda}|})$. Additionally, the *moment matrices* $\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|}$ are independent of the particular parameterization. This yields the following

Theorem 5.2. *For $j = 1, 2, \ldots, J$ and all $|\boldsymbol{\lambda}| = |\boldsymbol{\lambda}'| = j$, it holds*

$$\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|} = \mathbf{M}^{\square}_{|\boldsymbol{\lambda}'|}. \tag{5.16}$$

As a consequence we have to compute and store only a single moment matrix

$$\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|} \in \mathbb{R}^{\dim(\hat{V}_{J-|\boldsymbol{\lambda}|})\times(p+1)^2}$$

for each particular level. This is in contrast to the classical fast multipole method, where one has to compute the moment matrices for each cluster separately.

Because of quadrangular meshes, we may exploit the tensor product structure of the ansatz functions. To that end, let $\hat{\phi}_\ell = \hat{\phi}^{(1)}_\ell \otimes \hat{\phi}^{(2)}_\ell$ and $\hat{\phi}_{\ell'} = \hat{\phi}^{(1)}_{\ell'} \otimes \hat{\phi}^{(2)}_{\ell'}$, respectively. Then, the moment matrices $\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|}$ can be decomposed even further:

$$\begin{aligned}\int_{\square}\mathbf{L}_{\mathbf{m}}(\mathbf{s})\hat{\phi}_\ell(\mathbf{s})\,\mathrm{d}\mathbf{s} &= \int_0^1\int_0^1 L_{m_1}(s_1)\hat{\phi}^{(1)}_\ell(s_1)L_{m_2}(s_2)\hat{\phi}^{(2)}_\ell(s_2)\,\mathrm{d}s_1\,\mathrm{d}s_2 \\ &= \int_0^1 L_{m_1}(s_1)\hat{\phi}^{(1)}_\ell(s_1)\,\mathrm{d}s_1\int_0^1 L_{m_2}(s_2)\hat{\phi}^{(2)}_\ell(s_2)\,\mathrm{d}s_2 \\ &=: \left[\mathbf{M}_{|\boldsymbol{\lambda}|}\otimes\mathbf{M}_{|\boldsymbol{\lambda}|}\right]_{\ell,(p+1)m_1+m_2}.\end{aligned}$$

Since

$$\mathbf{M}_{|\boldsymbol{\lambda}|} \in \mathbb{R}^{\sqrt{\dim(\hat{V}_{J-|\boldsymbol{\lambda}|})}\times(p+1)}, \tag{5.17}$$

we arrive at an improved storage complexity for the far-field.

5.3. Computational Complexity

In the sequel, we derive complexity estimates for the FMM under consideration. In practice, it is convenient to impose a lower threshold for the far-field in terms of the polynomial degree p and the polynomial order d of V_J. Assuming $p > d$, we consider matrix blocks of size $p^2 \times p^2$ as near-field. Thus, these

blocks will not be compressed by the FMM. The proof of the next theorem implies that this results in $\mathcal{O}\big(N_J(p/d)^{-2}\big)$ near-field blocks with a storage cost of $\mathcal{O}\big(N_J(pd)^2\big)$, where N_J is the number of elements on level J. Moreover, we have the following result for the cost complexity of the far-field.

Theorem 5.3. *The complexity for the computation and the storage of the far-field is* $\mathcal{O}\big(N_J(pd)^2\big)$.

Proof. At first, we show inductively that there are $\mathcal{O}(N_j)$ admissible and also $\mathcal{O}(N_j)$ non-admissible clusters on level j. On level 0, this is clearly true. Thus, let the assumption hold for level $j-1$.

On level $j-1$, for a fixed cluster, there exist $\mathcal{O}(1)$ neighbouring clusters which do not satisfy the admissibility condition (5.15). For such clusters, we have to consider the 4 son clusters on level j. Hence, we face $4\mathcal{O}(N_{j-1}) = \mathcal{O}(N_j)$ non-admissible and also $\mathcal{O}(N_j)$ admissible cluster-cluster interactions on level j. Furthermore, due to the lower threshold, the maximum level to be computed is

$$\lceil J - 2\log_4(p/d)\rceil = J - \lfloor 2\log_4(p/d)\rfloor =: J - j_{\min}.$$

Since $N_j = 4^j M$, we may estimate

$$\sum_{j=0}^{J-j_{\min}} \mathcal{O}(N_j) = \mathcal{O}\big(M4^{J-j_{\min}}\big) = \mathcal{O}\big(M4^J(p/d)^{-2}\big) = \mathcal{O}\big(N_J(p/d)^{-2}\big).$$

Thus, we end up with overall $\mathcal{O}\big(N_J(p/d)^{-2}\big)$ far-field blocks and accordingly $\mathcal{O}\big(N_J(p/d)^{-2}\big)$ near-field blocks.

For each far-field block, we have to evaluate and store the localized kernel function in $\mathcal{O}(p^4)$ points. The complexity for assembly and storage of the moment matrices is $\mathcal{O}\big(\sqrt{N_J}pd\big)$ in total, cf. (5.17). Consequently, the far-field complexity is

$$\mathcal{O}\big(N_J(p/d)^{-2}\big)\cdot\mathcal{O}(p^4) + \mathcal{O}\big(\sqrt{N_J}pd\big) = \mathcal{O}\big(N_J(pd)^2\big).$$

□

Remark 5.4. *Due to the parametric surface representation, we obtain an improved cost complexity. The standard interpolation-based FMM proposes to interpolate the kernel in space. Thus, the polynomial degree enters with* $\mathcal{O}(p^3)$*, cf. [16, 17]. Since we only interpolate the transported kernel on the reference domain, we can reduce the cost to* $\mathcal{O}(p^2)$.

The improved storage complexity also affects the cost of the $\mathcal{H}$-matrix-vector multiplication. The complexity of the conventional $\mathcal{H}$-matrix-vector multiplication is $\mathcal{O}\big(N_J \log N_J p^3 d^2\big)$, see e.g. [24, 25], whereas we obtain the following result.

Theorem 5.5. *The complexity of the matrix-vector multiplication for the FMM is* $\mathcal{O}\big(N_J \log N_J (pd)^2\big)$.

Proof. On level j, we have $\mathcal{O}(N_j)$ far-field blocks with a block size of $d^2 N_J/N_j$. The complexity of the matrix-vector multiplication for the far-field is therefore

$$\sum_{j=0}^{J-j_{\min}} \mathcal{O}(N_j) \cdot \mathcal{O}\Big(\frac{N_J}{N_j}(pd)^2\Big) = \sum_{j=0}^{J-j_{\min}} \mathcal{O}\big(N_J(pd)^2\big) = \mathcal{O}\big(N_J \log N_J (pd)^2\big),$$

where $j_{\min} = \lfloor 2\log_4(p/d) \rfloor$. Next, we look at the near-field blocks and recall that we find in the near-field $\mathcal{O}\big(N_J(p/d)^{-2}\big)$ blocks with with $\mathcal{O}\big(p^4\big)$ entries. Thus, the overall complexity of the matrix-vector multiplication is $\mathcal{O}\big(N_J \log N_J (pd)^2\big)$ as claimed. □

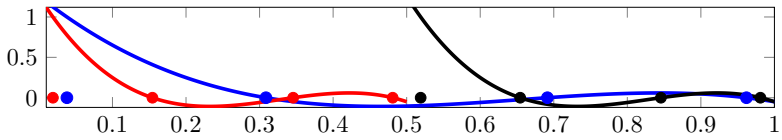


Figure 5.6: First Lagrange polynomials of son clusters and father cluster.

5.4. Nested Cluster Bases

We can improve the cost complexity of the matrix-vector multiplication to $\mathcal{O}\big(N_J(pd)^2\big)$ by exploiting the fact that the explicit computation of the moment matrices $\mathbf{M}_{|\boldsymbol{\lambda}|}$ for each particular level can be avoided by the concept of *nested cluster bases* which amounts to the $\mathcal{H}^2$-matrix representation, cf. [17].

Since the polynomial degree for each cluster is p, we can obviously represent the Lagrangian polynomials of the father cluster by those of the son clusters. Let

$$\{x_m^{(0)}\}_{m=0}^p = \left\{\frac{x_m}{2}\right\}_{m=0}^p \quad \text{and} \quad \{x_m^{(1)}\}_{m=0}^p = \left\{\frac{x_m+1}{2}\right\}_{m=0}^p,$$

respectively, be the interpolation points in the son clusters, see Figure 5.6. It holds $\{x_m^{(0)}\}_{m=0}^p \subset [0, 0.5]$ and $\{x_m^{(1)}\}_{m=0}^p \subset [0.5, 1]$. If we denote the related Lagrangian polynomials by $L_m^{(0)}(x)$ and $L_m^{(1)}(x)$, respectively, we can now exactly represent the Lagrangian polynomials of the father cluster according to

$$L_m(x) = \sum_{i=0}^p L_m(x_i^{(0)}) L_i^{(0)}(x) \quad \text{for } x \in [0, 0.5]$$

and

$$L_m(x) = \sum_{i=0}^p L_m(x_i^{(1)}) L_i^{(1)}(x) \quad \text{for } x \in [0.5, 1].$$

This gives rise to the *transfer matrices*

$$\mathbf{C}^{(0)} := [L_i(x_j^{(0)})]_{i,j=0}^p \quad \text{and} \quad \mathbf{C}^{(1)} := [L_i(x_j^{(1)})]_{i,j=0}^p$$

and yields the representation

$$\mathbf{M}_{|\boldsymbol{\lambda}|} = \begin{bmatrix} \mathbf{M}_{|\lambda|+1}\big(\mathbf{C}^{(0)}\big)^{\intercal} \\ \mathbf{M}_{|\lambda|+1}\big(\mathbf{C}^{(1)}\big)^{\intercal} \end{bmatrix}.$$

By tensor product construction, we then obtain the four transfer matrices

$$\mathbf{C}^{\square}_{2i+j} := \mathbf{C}^{(i)} \otimes \mathbf{C}^{(j)}, \quad i,j = 0,1,$$

for the reference domain $\square$. Here, we have the refinement relation

$$\mathbf{M}^{\square}_{|\boldsymbol{\lambda}|} = \begin{bmatrix} \mathbf{M}^{\square}_{|\lambda|+1}\big(\mathbf{C}^{\square}_{0}\big)^{\intercal} \\ \mathbf{M}^{\square}_{|\lambda|+1}\big(\mathbf{C}^{\square}_{2}\big)^{\intercal} \\ \mathbf{M}^{\square}_{|\lambda|+1}\big(\mathbf{C}^{\square}_{3}\big)^{\intercal} \\ \mathbf{M}^{\square}_{|\lambda|+1}\big(\mathbf{C}^{\square}_{1}\big)^{\intercal} \end{bmatrix}.$$

Notice that the peculiar order of the transfer matrices results from our hierarchical, counter clock-wise ordering of the elements, cf. Figure 2.3. Fortunately, the transfer matrices $\mathbf{C}^{\square}_0, \mathbf{C}^{\square}_1, \mathbf{C}^{\square}_2, \mathbf{C}^{\square}_3$ are independent of the respective cluster and even independent of the level $|\boldsymbol{\lambda}|$. This is a major advantage compared to the classical construction of $\mathcal{H}^2$-matrices as in e.g. [17]. Moreover, the transfer matrices are independent of the ansatz functions chosen for the Galerkin discretization.

In order to make use of the efficient implementation of the $\mathcal{H}^2$-matrix-vector multiplication, cf. [17, 24], we have only to store $\mathbf{M}^{\square}_J$ and $\mathbf{C}^{\square}_0, \mathbf{C}^{\square}_1, \mathbf{C}^{\square}_2, \mathbf{C}^{\square}_3$. This leads together with the hierarchical ordering of the elements to some simplifications in the $\mathcal{H}^2$-matrix-vector multiplication. The algorithm, tailored to the framework of parametric surfaces, is split in three parts: Algorithms 2, 3 and 4.

Algorithm 2 $\mathcal{H}^2$-matrix-vector multiplication, $\mathbf{y} = \mathbf{y} + \mathbf{H} \cdot \mathbf{x}$

```
procedure H²-MATRIX-VECTOR(H, x, y)
    u = FORWARDTRANSFORM(x)                          ▷ Handle far-field
    for λ × λ' ∈ F do
        v_λ = v_λ + H|_{λ×λ'} · u_{λ'}
    end for
    y = y+BACKWARDTRANSFORM(v)
    for λ × λ' ∈ N do                                ▷ Handle near-field
        y|_λ = y|_λ + H|_{λ×λ'} · x|_{λ'}
    end for
end procedure
```

Theorem 5.6. *The $\mathcal{H}^2$-matrix-vector multiplication of the FMM as stated in Algorithm 2 has a complexity of $\mathcal{O}\big(N_J(pd)^2\big)$.*

Algorithm 3 Forward transformation of $\mathbf{x}$ to $\mathbf{u}$

procedure ForwardTransform($\mathbf{x}$)
  for $(i, j', k) \in \mathcal{T}$, $j' = J$ **do**
    $\mathbf{u}_{(i,J,k)} = \big(\mathbf{M}_J^{\square}\big)^{\intercal}\mathbf{x}|_{(i,J,k)}$
  end for
  for $j = J-1, \dots 1$ **do**
    for $(i, j', k) \in \mathcal{T}$, $j' = j$ **do**

$$\mathbf{u}_{(i,j,k)} = \begin{bmatrix} \mathbf{C}_0^{\square}\mathbf{u}_{(i,j+1,4k)} \\ \mathbf{C}_2^{\square}\mathbf{u}_{(i,j+1,4k+1)} \\ \mathbf{C}_3^{\square}\mathbf{u}_{(i,j+1,4k+2)} \\ \mathbf{C}_1^{\square}\mathbf{u}_{(i,j+1,4k+3)} \end{bmatrix}$$

    end for
  end for
end procedure

Algorithm 4 Backward transformation of $\mathbf{v}$ to $\mathbf{y}$

procedure BackwardTransform($\mathbf{v}$)
  for $j = 1, \dots J-1$ **do**
    for $(i, j', k) \in \mathcal{T}$, $j' = j$ **do**

$$\begin{bmatrix} \mathbf{v}_{(i,j+1,4k)} \\ \mathbf{v}_{(i,j+1,4k+1)} \\ \mathbf{v}_{(i,j+1,4k+2)} \\ \mathbf{v}_{(i,j+1,4k+3)} \end{bmatrix} = \begin{bmatrix} \big(\mathbf{C}_0^{\square}\big)^{\intercal} \\ \big(\mathbf{C}_2^{\square}\big)^{\intercal} \\ \big(\mathbf{C}_3^{\square}\big)^{\intercal} \\ \big(\mathbf{C}_1^{\square}\big)^{\intercal} \end{bmatrix} \mathbf{v}_{(i,j,k)}$$

    end for
  end for
  for $(i, j', k) \in \mathcal{T}$, $j' = J$ **do**
    $\mathbf{y}|_{(i,J,k)} = \mathbf{M}_J^{\square}\mathbf{v}_{(i,J,k)}$
  end for
end procedure

Proof. To estimate the complexity of Algorithm 3, we remark that applying $N_{J-j_{\min}}$ times the moment matrices with $\mathcal{O}\big(N_{j_{\min}}(pd)^2\big)$ entries takes at most $\mathcal{O}\big(N_J(pd)^2\big)$ operations, where $j_{\min} = \lfloor 2\log_4(p/d)\rfloor$. The application of the transfer matrices to level $j+1$ requires $4p^4$ operations for each of the N_j clusters on level j. Hence, in a similar way as in the proof of Theorem 5.3 we conclude that the overall complexity of Algorithm 3 is

$$\mathcal{O}\big(N_J(pd)^2\big) + 4p^4 \sum_{j=0}^{J-j_{\min}} N_j = \mathcal{O}\big(N_J(pd)^2\big) + 4p^4\mathcal{O}\big(N_J(p/d)^{-2}\big) = \mathcal{O}\big(N_J(pd)^2\big).$$

In complete analogy, the complexity of Algorithm 4 is given by $\mathcal{O}\big(N_J(pd)^2\big)$. The complexity of the multiplication with the far-field coincides with the complexity of its memory consumption as derived in Theorem 5.3. The complexity for the near-field is the same as in the classical $\mathcal{H}$-matrix-vector multiplication,

which has been estimated in Theorem 5.5. We therefore end up with a total complexity of $\mathcal{O}\big(N_J(pd)^2\big)$ for the $\mathcal{H}^2$-matrix-vector multiplication. $\square$

5.5. Error Estimates

In view of Definition 3.1, we have the following result for the convergence of our FMM. It refers to the situation, when Chebyshev nodes on $I := [0,1]$, i.e. the points

$$x_m := \frac{1}{2}\left[\cos\left(\frac{2m+1}{2(p+1)}\pi\right)+1\right], \quad m = 0,1,\dots,p,$$

are used for the interpolation, cf. [17, 22].

Theorem 5.7. *Let $k(\mathbf{x},\mathbf{y})$ be an analytically standard kernel of order $2q$. Then, in an admissible block $\Gamma_{\boldsymbol{\lambda}} \times \Gamma_{\boldsymbol{\lambda}'} \in \mathcal{F}$, it holds*

$$\begin{aligned}\Big\| k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{s},\mathbf{t}) - \sum_{\|\mathbf{m}\|_\infty,\|\mathbf{m}'\|_\infty \le p} k_{\boldsymbol{\lambda},\boldsymbol{\lambda}'}(\mathbf{x}_\mathbf{m},\mathbf{x}_{\mathbf{m}'})\mathbf{L}_\mathbf{m}(\mathbf{s})\mathbf{L}_{\mathbf{m}'}(\mathbf{t})\Big\|_{L^\infty(\square\times\square)} \\ \lesssim \left(\frac{\eta}{r_k}\right)^{p+1} 2^{-4|\boldsymbol{\lambda}|}\big\|\boldsymbol{\kappa}_{\boldsymbol{\lambda}}(\mathbf{s}) - \boldsymbol{\kappa}_{\boldsymbol{\lambda}'}(\mathbf{t})\big\|_{L^\infty(\square\times\square)}^{-2(1+q)}\end{aligned}$$

with $r_k > 0$ being the constant from Definition 3.1.

From this theorem, one immediately derives an error estimate for the bilinear form which is associated with the variational formulation (3.5), cf. [16, 22].

Theorem 5.8. *Let $\sigma > 0$ be arbitrary but fixed. Then, for the integral operator $\mathcal{A}_J$ which results from an interpolation of degree $p > 0$ of the kernel function in every admissible block and the exact representation of the kernel in all other blocks, there holds*

$$\big|(\mathcal{A}u,v)_{L^2(D)} - (\mathcal{A}_J u,v)_{L^2(D)}\big| \lesssim 2^{-J\sigma}\|u\|_{L^1(D)}\|v\|_{L^1(D)}$$

provided that $p \sim J(2+2q+\sigma)$.

Hence, in order to maintain the approximation order of the Galerkin method, we have to choose $p \sim \log N_J$. Therefore, we end up with an over-all complexity of $\mathcal{O}\big(N_J(\log N_J)^2 d^2\big)$ for the computation and the storage of the far-field. Nevertheless, in view of singular kernels, one has to deal with singular integrals, e.g. by the Duffy trick, cf. [2, 26]. To that end, one has to increase the degree of the quadrature for all singular integrals proportionally to $|\log h_J|$ where $h_J = 2^{-J}$ is the mesh size and the constant depends on the order of the ansatz functions. This yields an effort of $\mathcal{O}\big((\log N_J)^4\big)$ for each singular integral. Thus, if the quadrature degree is properly decreased with the distance of the elements, one ends up with a complexity of $\mathcal{O}\big(N_J(\log N_J)^4 d^2\big)$ for the near-field.

6. Higher Order Continuous Ansatz Functions

One of the issues to address for continuous higher order ansatz functions is the clustering strategy. In the classical $\mathcal{H}$-matrix framework, usually a per degree of freedom cluster strategy is employed, see e.g. [24, 25]. In the context of higher order ansatz functions, this strategy has been applied to collocation matrices in [15] to compress the system matrices by using adaptive cross approximation. However, a per degree of freedom cluster strategy requires to iterate over the degrees of freedom during the matrix assembly. For the Galerkin scheme, this means for every degree of freedom that all elements in the support of the associated ansatz function have to be taken into account. Thus, for continuous higher order ansatz functions, every element is visited several times during the matrix assembly and function evaluations for a numerical quadrature are possibly done multiple times for the same quadrature point.

In order to overcome this obstruction, one therefore often iterates over the elements for the matrix assembly. To maintain this element-wise strategy for the matrix assembly of a higher order FMM, we propose to keep the element-wise cluster strategy introduced in Section 3. In the sequel, we provide an easy means to extend the FMM for discontinuous, element-wise polynomial ansatz functions from Section 5 to globally continuous ansatz functions. This means, from now on, we consider ansatz spaces $V_j^c \subset V_j \cap C(\Gamma)$. Clearly, there exists then a transformation matrix $\mathbf{T}$ such that

$$\mathbf{T}\mathbf{A}_J\mathbf{T}^\intercal\mathbf{u}_J^c = \mathbf{T}\mathbf{f}_J, \tag{6.18}$$

where $\mathbf{A}_J$ is the system matrix and $\mathbf{f}_J$ is the right hand side with respect to discontinuous, element-wise polynomial ansatz functions from (4.14) and $\mathbf{u}_J^c$ are the coefficients of the globally continuous ansatz functions in V_J^c. The transformation matrix $\mathbf{T}$ is a sparse matrix if the supports of the ansatz functions in V_j^c only contain a few elements, as it is e.g. the case for B-splines, which are used in isogeometric analysis. This situation will be illustrated in the sequel.

We denote by $\hat{V}_j^c$ the space spanned by the tensor product B-splines of order d on the reference domain. The tensor product B-splines are obtained by tensorization of the B-spline basis of order d on the interval $[0, 1]$. To that end, we introduce the partition

$$0 = t_1 = \cdots = t_d < \cdots < t_{n+d+1} = \cdots = t_{n+2d} = 1.$$

Now, setting

$$B_{j,1}(x) = \begin{cases} 1, & \text{if } t_j \leq x < t_{j+1}, \\ 0, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n + 2d - 1,$$

and using the recursion formula

$$B_{j,k}(x) = \frac{x - t_j}{t_{j+k-1} - t_j} B_{j,k-1}(x) + \frac{t_{j+k} - x}{t_{j+k} - t_{j+1}} B_{j+1,k-1}(x), \qquad j = 1, \ldots, n + 2d - k,$$

up to $k = d$ will give us $n + d$ B-spline basis functions of order d on $[0, 1]$, cf. [27]. The B-splines bases up to order 3 are depicted in Figure 6.7.

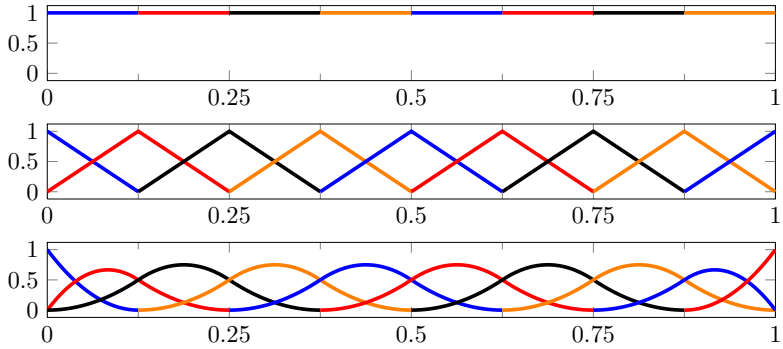


Figure 6.7: B-spline bases on the unit interval $[0, 1]$ of order 1 (top), order 2 (middle), and order 3 (bottom).

It holds $\hat{V}_j^c \subset \hat{V}_j$, such that we can express every function in $\hat{V}_j^c$ as a linear combination of functions in $\hat{V}_j$. Let $\hat{\mathbf{U}} = [\hat{\varphi}_1^c, \dots, \hat{\varphi}_{\hat{n}_j^c}^c]$ denote the tensor product B-spline basis of $\hat{V}_j^c$ and $\hat{\mathbf{V}} = [\hat{\varphi}_1, \dots, \hat{\varphi}_{\hat{n}_j}]$ the piecewise polynomial basis of $\hat{V}_j$, where we set $\hat{n}_j^c = \dim(\hat{V}_j^c)$ and $\hat{n}_j = \dim(\hat{V}_j)$. Then, the patchwise transformation matrix $\hat{\mathbf{T}}$ is uniquely determined by the relation $\hat{\mathbf{U}} = \hat{\mathbf{T}}\hat{\mathbf{V}}$. Unfortunately, the functions in the composed space

$$\tilde{V}_j = \left\{ \hat{\varphi} \circ \boldsymbol{\gamma}_i^{-1} : \hat{\varphi} \in \hat{V}_j^c,\ i = 1, \dots, M \right\} \subset L^2(\Gamma)$$

are, in general, discontinuous on the boundaries of the patches $\partial\Gamma_i$. Let therefore $\mathcal{I}$ be the operator which glues the ansatz functions that are nonzero at the patch boundaries $\partial\Gamma_i$ in a continuous way and let $\mathbf{I}$ be its discrete analogue. Then, the ansatz space V_j^c of globally continuous, tensorized B-splines on level j is given by

$$V_j^c := \mathcal{I}(\tilde{V}_j) \subset V_j \cap C(\Gamma).$$

Since $V_j^c \subset V_j$, we can express every function in V_j^c as a linear combination of functions in V_j. For that purpose, let $[\varphi_1^c, \dots, \varphi_{n_j^c}^c]$ with $n_j^c = \dim(V_j^c)$ denote the B-spline basis of V_j^c and

$$a = \sum_{i=1}^{n_j^c} a_i \varphi_i^c \in V_j^c, \quad b = \sum_{i=1}^{M} \sum_{k=1}^{\hat{n}_j} b_{i,k} (\hat{\varphi}_k \circ \boldsymbol{\gamma}_i^{-1}) \in V_j.$$

Setting $\mathbf{b}_i = [b_{i,1}, \dots b_{i,\hat{n}_j}]$, we obtain the relation

$$\begin{bmatrix} a_1 \\ \vdots \\ a_{n_j^c} \end{bmatrix} = \mathbf{I} \begin{bmatrix} \hat{\mathbf{T}} & & \\ & \ddots & \\ & & \hat{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_M \end{bmatrix} = \mathbf{T} \begin{bmatrix} \mathbf{b}_1 \\ \vdots \\ \mathbf{b}_M \end{bmatrix}.$$

Since $\hat{\mathbf{T}}$ and $\mathbf{I}$ are sparse matrices, $\mathbf{T}$ is also a sparse matrix and the transformation from V_j to V_j^c can be done in an efficient way.

At first glance, the simplicity of the presented method comes at a high price. The memory consumption of the uncompressed matrix $\mathbf{A}_J$ in (6.18) is n_J^2, where $n_J = \dim(V_J)$, instead of $(n_J^c)^2$, i.e. the memory consumption will grow like $\mathcal{O}\big((n_J/n_J^c)^2\big)$, whereas the number of degrees of freedom only grows like $\mathcal{O}\big(n_J/n_J^c\big)$. Although nowadays memory consumption can be considered as a minor problem, this also means that in case of uncompressed matrices the computational effort for the matrix-vector multiplication will grow like $\mathcal{O}\big((n_J/n_J^c)^2\big)$. Compared to this, the FMM compression presented in the previous section reduces the growth of the memory consumption and the operations for the $\mathcal{H}^2$-matrix-vector multiplication to $\mathcal{O}(n_J/n_J^c)$.

7. Numerical Results

Besides presenting numerical examples for the convergence of our fast multipole method, this section contains also a comparison of the computational cost versus accuracy. All computations of the following examples have been carried out on a single core of a computing server with two Intel(R) Xeon(R) E5-2670 CPUs with a clock rate of 2.60 GHz and a main memory of 256 GB.

7.1. Problem Setting

We focus on the numerical solution of boundary integral equations which amount from the reformulation of the Dirichlet boundary value problem for the Laplacian in a three-dimensional Lipschitz domain $\Omega \in \mathbb{R}^3$:

$$\Delta U = 0 \text{ in } \Omega, \quad U = f \text{ on } \Gamma. \tag{7.19}$$

The first boundary integral equation under consideration stems from the single layer potential ansatz, where one makes the ansatz

$$U(\mathbf{x}) = \int_\Gamma \frac{u(\mathbf{y})}{4\pi\|\mathbf{x}-\mathbf{y}\|_2} \, \mathrm{d}\sigma_{\mathbf{y}} = \tilde{\mathcal{S}}u(\mathbf{x}), \quad \mathbf{x} \in \Omega. \tag{7.20}$$

This leads to a Fredholm integral equation of the first kind

$$\mathcal{S}u(\mathbf{x}) = \int_\Gamma \frac{u(\mathbf{y})}{4\pi\|\mathbf{x}-\mathbf{y}\|_2} \, \mathrm{d}\sigma_{\mathbf{y}} = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \tag{7.21}$$

for the unknown density u. The second boundary integral equation is obtained from a double layer potential ansatz

$$U(\mathbf{x}) = \int_\Gamma \frac{\langle \mathbf{x}-\mathbf{y}, \mathbf{n}_\mathbf{y}\rangle u(\mathbf{y})}{4\pi\|\mathbf{x}-\mathbf{y}\|_2^3}\,\mathrm{d}\sigma_\mathbf{y} = \tilde{\mathcal{K}}u(\mathbf{x}), \quad \mathbf{x}\in\Omega, \tag{7.22}$$

which leads to a Fredholm integral equation of the second kind

$$\frac{1}{2}u(\mathbf{x}) - \mathcal{K}u(\mathbf{x}) = \frac{1}{2}u(\mathbf{x}) - \int_\Gamma \frac{\langle \mathbf{x}-\mathbf{y}, \mathbf{n}_\mathbf{y}\rangle u(\mathbf{y})}{4\pi\|\mathbf{x}-\mathbf{y}\|_2^3}\,\mathrm{d}\sigma_\mathbf{y} = f(\mathbf{x}), \quad \mathbf{x}\in\Gamma, \tag{7.23}$$

for the unknown density u.

We will employ B-splines of order $d = 1, 2, 3$ as ansatz functions, which are glued together at the patch interfaces to achieve global continuity for $d = 2, 3$. This means that the system of linear equations is computed with the help of the transformation matrices which have been introduced in Section 6. To solve the system (6.18) of linear equations, we use a conjugate gradient method (CG) for the single layer operator and the generalized minimal residual method (GMRES) with restart after 100 inner iterations for the double layer operator, cf. [28]. The construction of an appropriate preconditioner for the CG exceeds the scope of this article and is left as further work.

In view of Theorem 4.1, one obtains the following error estimate for the approximate potential U_J.

Theorem 7.1. *Let $u \in H^q(\Gamma)$ be the solution of (7.21) or (7.23). Moreover, let U be the corresponding potential and U_J its numerical approximation. Then, there holds the error estimate*

$$|U(\mathbf{x}) - U_J(\mathbf{x})| \lesssim 2^{2J(q-d)}\|k(\mathbf{x},\cdot)\|_{H^{-2q+d}(\Gamma)}\|u\|_{H^d(\Gamma)}, \quad \mathbf{x}\in\Omega,$$

where $2q = -1$ in case of (7.21) and $2q = 0$ in case of (7.23).

Proof. Together with Theorem 4.1, there holds

$$\begin{aligned} |U(\mathbf{x}) - U_J(\mathbf{x})| &= \left|\int_\Gamma k(\mathbf{x},\mathbf{y})\big(u(\mathbf{y}) - u_J(\mathbf{y})\big)\,\mathrm{d}\sigma_\mathbf{y}\right| \\ &\le \|k(\mathbf{x},\cdot)\|_{H^{-2q+d}(\Gamma)}\|u - u_J\|_{H^{2q-d}(\Gamma)} \\ &\lesssim 2^{2J(q-d)}\|k(\mathbf{x},\cdot)\|_{H^{2q-d}(\Gamma)}\|u\|_{H^d(\Gamma)} \end{aligned}$$

for both, the single and the double layer potential. □

7.2. Convergence

In our first example, we solve the Laplace equation (7.19) in the unit ball with the spherical harmonic

$$Y_0^2(\mathbf{x}) = \sqrt{\frac{5}{16\pi}}\big(3x_3^2 - 1\big), \quad \mathbf{x}\in\Gamma,$$

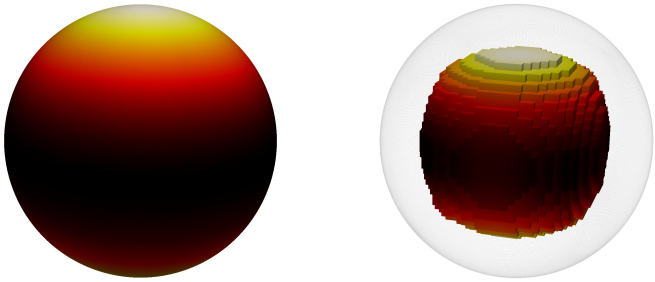


Figure 7.8: The spherical harmonic Y_0^2 (left) and the related potential (right) for the unit ball.

seen in Figure 7.8, as boundary condition f. The boundary of the unit ball is represented by six patches.

Since the spherical harmonic on the sphere is an eigenfunction to the eigenvalue $\lambda = 1/5$ in case of the single layer operator and $\lambda = -3/5$ in case of the double layer operator, we know the analytical solution of (7.21) and (7.23) and can thus compute the $L^2(\Gamma)$-error of the approximate density. Moreover, we find for the potential the analytical solution

$$U(\mathbf{x}) = \|\mathbf{x}\|_2 Y_0^2\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right).$$

Figure 7.9 validates that the proposed FMM provides the theoretical convergence rates on smooth domains in case of the single layer potential ansatz. Figure 7.10 validates this for the double layer potential ansatz. In both cases, the l^∞-error of the potential is measured in the 18'999 vertices of 16'616 cubes which lie in the interior of the ball, as depicted on the right of Figure 7.8. The polynomial degree p for FMM is chosen such that the overall accuracy is maintained. Hence, in accordance with Theorem 5.8, the number of interpolation points grows linearly with the discretization level J. The numbers of local and global degrees of freedom n_J and n_J^c, respectively, associated with the discretization level J, are tabulated in Table 7.2. Note that both numbers coincide in case of piecewise constant boundary elements, i.e. for $d = 1$.

In the second example, we want to deal with the more complex gear worm geometry depicted in Figure 7.11, which is represented by 290 patches. For our experiments, we prescribe the harmonic polynomial

$$U(\mathbf{x}) = 4x_1^2 - 3x_2^2 - x_3^2$$

as potential and restrict it to the boundary in order to get the boundary conditions for (7.19). Since the density u is unknown, we fitted a grid of 83'437 cubes inside the domain and measured the error of the potential inside the domain on the 115'241 vertices of these cubes. A visualization of the cubes together with the computed density for the single layer potential ansatz can be found in Figure 7.11.

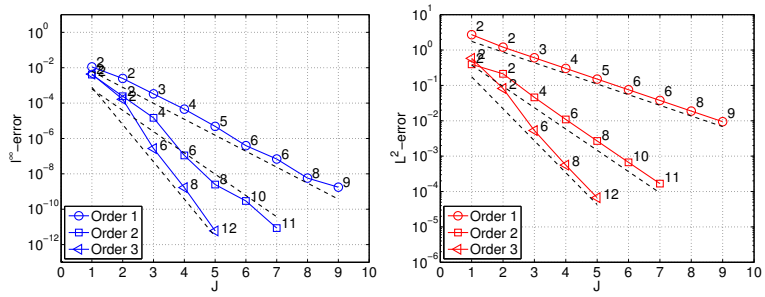


Figure 7.9: l^∞-error (left) and L^2-error (right) for the sphere for the single layer potential with the corresponding theoretical convergence rates h^3, h^5 and h^7 for the potential and h^1, h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

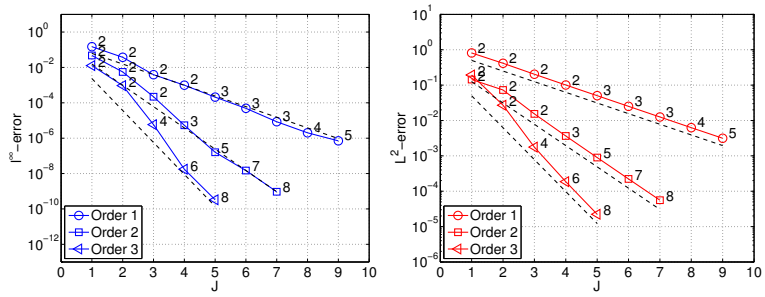


Figure 7.10: l^∞-error (left) and L^2-error (right) for the sphere for the double layer potential with the corresponding theoretical convergence rates h^2, h^4 and h^6 for the potential and h^1, h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

Since the gear worm only has a Lipschitz continuous boundary, the theoretical convergence rates are limited to at most h^3 for the single layer potential ansatz and to h^2 for the double layer potential ansatz. Figure 7.12 illustrates that we achieve these convergence rates for all ansatz functions under consideration. In fact, the higher order ansatz functions even seem to produce a convergence rate about h^5 for the single layer ansatz and up to h^4 for the double layer potential ansatz. Again, the numbers of local and global degrees of freedom n_J and n_J^c, respectively, associated with the discretization level J, are tabulated in Table 7.2.

7.3. Computational Cost and Accuracy

First of all, we want to demonstrate the benefit of the $\mathcal{H}^2$-matrix-vector multiplication compared to the $\mathcal{H}$-matrix-vector multiplication. To that end, we have measured the time for a matrix-vector multiplication of an $\mathcal{H}$-matrix

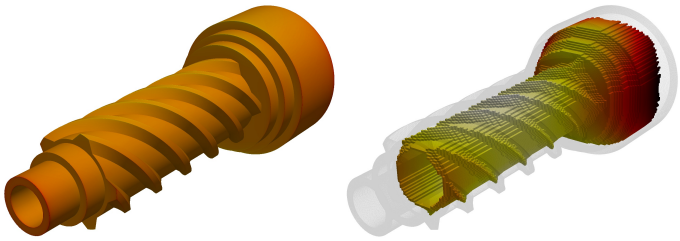


Figure 7.11: The approximate density of the single layer potential ansatz (left) and the related potential (right) for the gear worm.

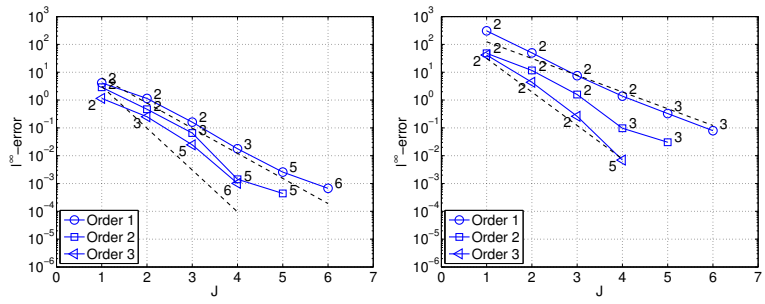


Figure 7.12: l^∞-error on the gear worm for the single layer potential (left) and the double layer potential (right) with the corresponding theoretical convergence rates h^3 and h^5 for the single layer potential and h^2 and h^4 for the double layer potential. The accompanying numbers are the polynomial degrees of the interpolation.

and of an $\mathcal{H}^2$-matrix, both stemming from the discretization of the double layer operator on the sphere or on the gear worm. The polynomial degree for the FMM is set to $p = 2$. Figure 7.13 illustrates that we reach an asymptotic complexity of $\mathcal{O}\big(N_J(pd)^2\big)$ for the $\mathcal{H}^2$-matrix-vector multiplication compared to the complexity of $\mathcal{O}\big(N_J \log N_J(pd)^2\big)$ for the $\mathcal{H}$-matrix-vector multiplication.

Second, we want to illustrate the effectiveness of higher order ansatz functions. To that end, we compare the l^∞-error of the potential with the computation time of the matrix and with the computation time of the matrix plus the solving time using the $\mathcal{H}^2$-matrix-vector multiplication. The results with respect to the sphere are depicted in Figure 7.14 and the results with respect to the gear worm are depicted in Figure 7.15. They indicate that the higher order ansatz functions achieve asymptotically a higher precision combined with a faster computation time. Note that the increased solving times for the gear worm geometry are due to a higher number of iterations in the solving process.

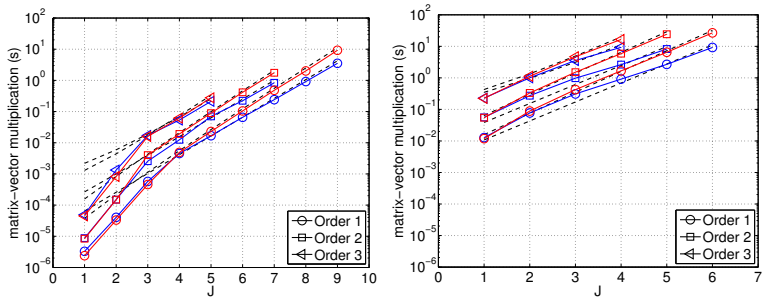


Figure 7.13: Computation times for the $\mathcal{H}^2$-matrix-vector multiplication (blue) and the $\mathcal{H}$-matrix-vector multiplication (red) on the sphere (left) and on the gear worm (right). The dashed lines illustrate the complexity rates $\mathcal{O}(N_J(pd)^2)$ and $\mathcal{O}(N_J \log N_J(pd)^2)$.

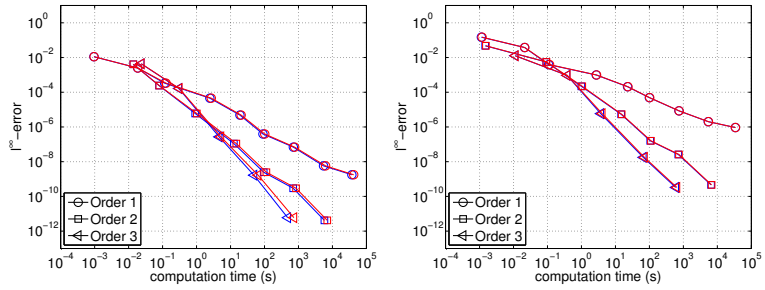


Figure 7.14: l^∞-error versus the computation time of the matrix (blue) and the computation time of the matrix plus the solving time (red) for the single layer potential (left) and the double layer potential (right) on the sphere.

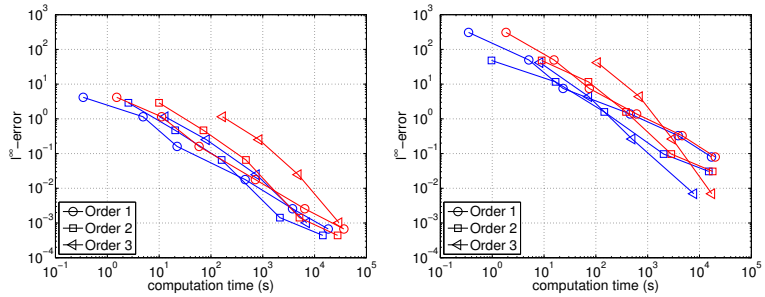


Figure 7.15: l^∞-error versus the computation time of the matrix (blue) and the computation time of the matrix plus the solving time (red) for the single layer potential (left) and the double layer potential (right) on the gear worm.

		$n_J = \dim V_J$			$n_J^c = \dim V_J^c$			
	J	$d=1$	$d=2$	$d=3$	$d=2$		$d=3$	
sphere	1	24	96	216	26	(3.7)	56	(3.9)
	2	96	384	864	98	(3.9)	152	(5.7)
	3	384	1'536	3'456	386	(4.0)	488	(7.1)
	4	1'536	6'144	13'824	1'538	(4.0)	1'736	(8.0)
	5	6'144	24'576	55'296	6'146	(4.0)	6'536	(8.5)
	6	24'576	98'304		24'578	(4.0)		
	7	98'304	393'216		98'306	(4.0)		
	8	393'216						
	9	1'572'864						
gear worm	1	1'160	4'640	10'440	1'160	(4.0)	2'610	(4.0)
	2	4'640	18'560	41'760	4'640	(4.0)	7'250	(5.8)
	3	18'560	74'240	167'040	18'560	(4.0)	23'490	(7.1)
	4	74'240	296'960	668'160	74'240	(4.0)	83'810	(8.0)
	5	296'960	1'187'840		296'960	(4.0)		
	6	1'187'840						

Table 7.2: Dimensions n_J and n_J^c of the ansatz spaces V_J and V_J^c, respectively, for the sphere and the gear worm for different polynomial orders. The associated ratios n_J/n_J^c are given in the parentheses.

8. Conclusion

Parametric surfaces are easily accessible from computer aided design. They are recently of interest in isogeometric analysis, the goal of which is the direct integration of the finite element or even the boundary element analysis into the design process. In this article, we have presented a fast boundary element method, namely an $\mathcal{H}^2$-matrix fast multipole method based on the interpolation of the related integral kernel on the reference domain. This approach perfectly exploits the features of the parametric surface representation. By a tensor product construction and appropriate transformation matrices, we can easily deal with higher order ansatz functions. Our complexity estimates as well as our numerical examples demonstrate the superior performance of the presented method.

References

[1] W. Hackbusch, Integral Equations: Theory and Numerical Treatment, Vol. 4, Birkhäuser, Basel, 1995.

[2] S. A. Sauter, C. Schwab, Boundary Element Methods, Springer, Berlin-Heidelberg, 2011.

[3] O. Steinbach, Numerical Approximation Methods for Elliptic Boundary Value Problems, Springer Science + Business, New York, 2008.

[4] L. Greengard, V. Rokhlin, A fast algorithm for particle simulations, Journal of Computational Physics 73 (2) (1987) 325–348.

[5] W. Hackbusch, Z. P. Nowak, On the fast matrix multiplication in the boundary element method by panel clustering, Numerische Mathematik 54 (4) (1989) 463–491.

[6] G. Beylkin, R. Coifman, V. Rokhlin, Fast wavelet transforms and numerical algorithms I, Communications on Pure and Applied Mathematics 44 (2) (1991) 141–183.

[7] W. Dahmen, H. Harbrecht, R. Schneider, Compression techniques for boundary integral equations. Asymptotically optimal complexity estimates, SIAM Journal on Numerical Analysis 43 (6) (2006) 2251–2271.

[8] M. Bebendorf, Approximation of boundary element matrices, Numerische Mathematik 86 (4) (2000) 565–589.

[9] Initial Graphics Exchange Specification. IGES 5.3, U.S. Product Data Association (1996).

[10] ISO 10303-242:2014, Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 242: Application Protocol: Managed Model-based 3D Engineering, International Organization for Standardization (2014).

[11] J. Hoschek, D. Lasser, Grundlagen der Geometrischen Datenverarbeitung, Teubner, Stuttgart, 1989.

[12] J. A. Cottrell, T. J. R. Hughes, Y. Bazilevs, Isogeometric Analysis: Toward Integration of CAD and FEA, 1st Edition, Wiley Publishing, 2009.

[13] T. J. R. Hughes, J. A. Cottrell, Y. Bazilevs, Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement, Computer Methods in Applied Mechanics and Engineering 194 (39) (2005) 4135–4195.

[14] B. Marussig, J. Zechner, G. Beer, T. Fries, Fast isogeometric boundary element method based on independent field approximation, Computer Methods in Applied Mechanics and Engineering 284 (0) (2015) 458–488, isogeometric Analysis Special Issue.

[15] J. Zechner, B. Marussig, G. Beer, T. Fries, Isogeometric boundary element method with hierarchical matrices, CoRR abs/1406.2817.

[16] K. Giebermann, Multilevel approximation of boundary integral operators, Computing 67 (3) (2001) 183–207.

[17] W. Hackbusch, S. Börm, $\mathcal{H}^2$-matrix approximation of integral operators by interpolation, Applied Numerical Mathematics 43 (1-2) (2002) 129–143.

[18] H. Harbrecht, M. Peters, M. Siebenmorgen, Efficient approximation of random fields for numerical applications, Preprint 2014-01, Mathematisches Institut, Universität Basel (to appear in Numerical Linear Algebra with Applications).

[19] H. Harbrecht, M. Randrianarivony, From computer aided design to wavelet BEM, Computing and Visualization in Science 13 (2) (2010) 69–82.

[20] H. Harbrecht, M. Randrianarivony, Wavelet BEM on molecular surfaces: Parametrization and implementation, Computing 86 (1) (2009) 1–22.

[21] H. Harbrecht, M. Randrianarivony, Wavelet BEM on molecular surfaces: Solvent excluded surfaces, Computing 92 (4) (2011) 335–364.

[22] H. Harbrecht, M. Peters, Comparison of fast boundary element methods on parametric surfaces, Computer Methods in Applied Mechanics and Engineering 261–262 (2013) 39–55.

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

[24] S. Börm, Efficient Numerical Methods for Non-local Operators, Vol. 14 of EMS Tracts in Mathematics, European Mathematical Society (EMS), Zürich, 2010.

[25] W. Hackbusch, Hierarchische Matrizen: Algorithmen und Analysis, Springer, Heidelberg, 2009.

[26] S. A. Sauter, C. Schwab, Quadrature for *hp*-Galerkin BEM in $\mathbb{R}^3$, Numerische Mathematik 78 (2) (1997) 211–258.

[27] C. De Boor, A Practical Guide to Splines, Springer, New York, 1978.

[28] Y. Saad, Iterative Methods for Sparse Linear Systems, 2nd Edition, Society for Industrial and Applied Mathematics, 2003.

LATEST PREPRINTS

No.	Author: *Title*
2015-01	**Ali Hyder** *Existence of Entire Solutions to a Fractional Liouville Equation in R^n*
2015-02	**H. Harbrecht, M. Peters** *Solution of Free Boundary Problems in the Presence of Geometric Uncertainties*
2015-03	**M. Dambrine, C. Dapogny, H. Harbrecht** *Shape Optimization for Quadratic Functionals and States with Random Right-Hand Sides*
2015-04	**A. Bohun, F. Bouchut, G. Crippa** *Lagrangian Flows for Vector Fields with Anisotropic Regularity*
2015-05	**A. Bohun, F. Bouchut, G. Crippa** *Lagrangian Solution to the Vlasov-Poisson System with L^1 Density*
2015-06	**S. Iula, A. Maalaoui, L. Martinazzi** *A fractional Moser-Trudinger type inequality in one dimension and its critical points*
2015-07	**A. Hyder** *Structure of conformal metrics on R^n with constant Q-curvature*
2015-08	**M. Colombo, G. Crippa, S. Spirito** *Logarithmic Estimates for Continuity Equations*
2015-09	**M. Colombo, G. Crippa, S. Spirito** *Renormalized Solutions to the Continuity Equation with an Integrable Damping Term*
2015-10	**G. Crippa, N. Gusev, S. Spirito, E. Wiedemann** *Failure of the Chain Rule for the Divergence of Bounded Vector Fields*
2015-11	**G. Crippa, N. Gusev, S. Spirito, E. Wiedemann** *Non-Uniqueness and Prescribed Energy for the Continuity Equation*
2015-12	**G. Crippa, S. Spirito** *Renormalized Solutions of the 2D Euler Equations*
2015-13	**G. Crippa, E. Semenova, S. Spirito** *Strong Continuity for the 2D Euler Equations*

Preprints are available under *https://math.unibas.ch/research/publications*

LATEST PREPRINTS

No.	Author: *Title*
2015-14	**J. Diaz, M. J. Grote** *Multi-Level Explicit Local Time-Stepping Methods for Second-Order Wave Equations*
2015-15	**F. Da Lio, L. Martinazzi, T. Riviere** *Blow-up analysis of a nonlocal Liouville-type equation*
2015-16	**A. Maalaoui, L. Martinazzi, A. Schikorra** *Blow-up behaviour of a fractional Adams-Moser-Trudinger type inequality in odd dimension*
2015-17	**T. Boulenger, E. Lenzmann** *Blowup for Biharmonic NLS*
2015-18	**D. Masser, U. Zannier (with Appendix by V. Flynn)** *Torsion point on families of simple abelian surfaces and Pell's equation over polynomial rings*
2015-19	**M. Bugeanu, R. Di Remigio, K. Mozgawa, S. S. Reine, H. Harbrecht, L. Frediani** *Wavelet Formulation of the Polarizable Continuum Model. II. Use of Piecewise Bilinear Boundary Elements*
2015-20	**J. Dölz, H. Harbrecht, M. Peters** *An interpolation-based fast multipole method for higher order boundary elements on parametric surfaces*

Preprints are available under https://math.unibas.ch/research/publications