

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

J. Dölz, H. Harbrecht, and M. Peters*

University of Basel, Department of Mathematics and Computer Science, Spiegelgasse 1, 4051 Basel, Switzerland

SUMMARY

In this article, a black-box higher order fast multipole method for solving boundary integral equations on parametric surfaces in three spatial dimensions is proposed. 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 strategy 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, the cost complexity in the polynomial degree is reduced by one order. This gain is independent of the application of either \mathcal{H} - or \mathcal{H}^2 -matrices. In fact, several simplifications in the construction of \mathcal{H}^2 -matrices are pointed out, which are a by-product of the surface representation. Extensive numerical examples are provided in order to quantify and qualify the proposed method. Copyright © 0000 John Wiley & Sons, Ltd.

Received ...

KEY WORDS: Non-local operators; parametric surfaces; higher order ansatz functions; \mathcal{H}^2 -matrices; fast multipole method, isogeometric analysis.

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 operator 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 major 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 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]. Except for the wavelet Galerkin

*Correspondence to: E-mail: michael.peters@unibas.ch

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

scheme, all these methods can be cast in the \mathcal{H} -matrices structure, cf. [9], which separates compressible and non-compressible matrix blocks by the choice of a suitable partitioning of the underlying matrix. This compression might even be improved by means of \mathcal{H}^2 -matrices, cf. [10]. Then, the underlying tree structure of the \mathcal{H} -matrix is exploited to construct nested cluster bases for the compressible matrix blocks. In the end, all 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.

This article focuses on the fast multipole method (FMM) for the solution of boundary integral equations and casts 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. [11, 12]. 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 [13]. The representation with NURBS is intensively studied in the context of isogeometric analysis, see e.g. [14, 15, 16, 17]. Nevertheless, in contrast to the isogeometric analysis framework, the scope of this article is not restricted to geometries that can be represented by NURBS, but considers 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 tangent vector, 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. Particularly, there is no further approximation step required if the surface is given in this form. As a consequence, the rate of convergence in a corresponding boundary element method is not limited by the accuracy of the surface approximation.

This article shall provide 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, the presented fast multipole method interpolates the kernel function directly on the reference domain. This is in contrast to the interpolation of the kernel in space, as in e.g. [18, 19], and yields a remarkable speed-up of the FMM, since the dimension reduction due to the boundary integral formulation of the underlying problem can be fully exploited. 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, one can still profit from the \mathcal{H}^2 -matrix techniques presented in e.g. [18, 19]. Notably, since the considered realization of parametric surfaces is based on four-sided patches, one can exploit the tensor product structure of the reference domain to simplify the construction of \mathcal{H}^2 -matrices. More precisely, due to the special structure of the reference 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. Here, the Green's function is replaced by a more general integral kernel. In particular, no specific properties of the integral kernel are required except for its smoothness apart from the diagonal. For example, in [20], a lowest order black-box FMM on parametric surfaces has been employed to discretize correlation kernels.

The rest of this article is structured as follows. At first, in Section 2, the considered parametric surface representation is introduced. As a consequence from this representation, the mesh generation

is straightforward. Section 3 discusses 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, the algorithm which perfectly fits the framework of parametric surfaces is presented and extended to the \mathcal{H}^2 -matrix variant. Section 6 provides a straightforward extension of the FMM to higher order continuous ansatz functions. Finally, Section 7 presents numerical experiments to validate and quantify the presented numerical approach.

In the sequel, in order to avoid the repeated use of generic but not further specified constants, it is implied 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 we write $C \sim D$ if $C \lesssim D$ 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, a parametric representation of the surface Γ can be constructed as follows. Let $\square := [0, 1]^2$ denote the unit square, which serves as reference domain. Moreover, subdivide the given surface Γ into several smooth *patches*

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

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

$$\gamma_i : \square \rightarrow \Gamma_i \quad \text{with} \quad \Gamma_i = \gamma_i(\square) \quad \text{for } i = 1, 2, \dots, M, \quad (1)$$

as illustrated in Figure 1.

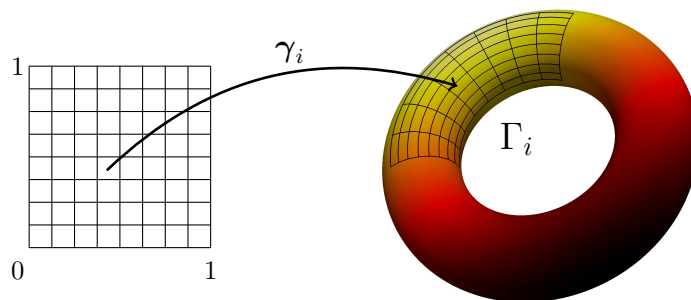


Figure 1. Surface representation and mesh generation.

For constructing regular surface meshes, one has to impose the following *matching condition*: For each $\mathbf{x} = \gamma_i(\mathbf{s})$ on a common edge of Γ_i and $\Gamma_{i'}$, there has to exist a bijective and affine mapping $\Xi: \square \rightarrow \square$ such that there holds $\gamma_i(\mathbf{s}) = (\gamma_{i'} \circ \Xi)(\mathbf{s})$. This means that the parameterizations γ_i and $\gamma_{i'}$ coincide on the common edge except for orientation.

In the sequel, in order to define transported kernel functions, the *surface measure* of the diffeomorphisms γ_i is required. On the patch Γ_i , it is given by

$$\kappa_i(\mathbf{s}) := \left\| \partial_{s_1} \gamma_i(\mathbf{s}) \times \partial_{s_2} \gamma_i(\mathbf{s}) \right\|_2. \quad (2)$$

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 [21]. This algorithm has been extended in [22, 23] to molecular surfaces. Figure 2 visualizes three

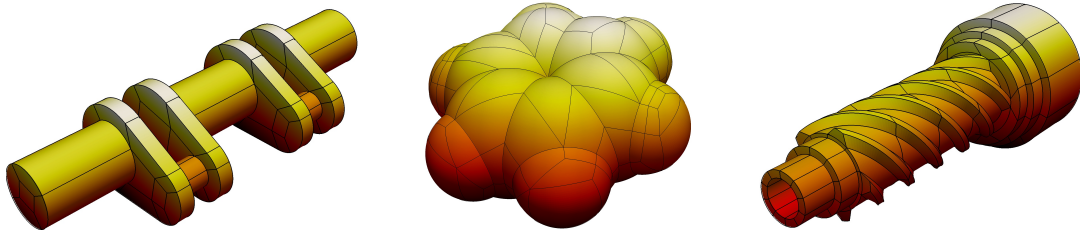


Figure 2. Different parametric surfaces with their patch boundaries.

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 γ_i , see Figure 1 for a visualization. This procedure leads to a nested and especially quad-tree structured sequence

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

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

The particular elements shall be referred to as $\Gamma_{i,j,k}$ where i is the index of the underlying parameterization γ_i , j denotes the level of the element and k is the index of the element in hierarchical order. For notational convenience the triple (i, j, k) shall be referenced by $\lambda := (i, j, k)$ with $|\lambda| := j$. In view of the fast multipole method, $\Gamma_{i,j,k}$ shall also be considered as a *cluster*. In this sense, $\Gamma_{i,j,k}$ can be thought of as the union $\{\Gamma_{i,J,k'} : \Gamma_{i,J,k'} \subset \Gamma_{i,j,k}\}$, i.e. the set of all tree leaves appended to either $\Gamma_{i,j,k}$ or one of its sons. Furthermore, the hierarchical ordered collection of all clusters, the *cluster tree*, is denoted by \mathcal{T} . A scheme for the subdivisions of the patch Γ_i up to level 2 is illustrated in Figure 3. Finally, with respect to the tree structure of \mathcal{T} , one defines $\text{dad}(\lambda) := (i, j - 1, \lfloor k/4 \rfloor)$ and $\text{sons}(\lambda) := \{(i, j + 1, 4k + \ell) : \ell = 0, \dots, 3\}$.

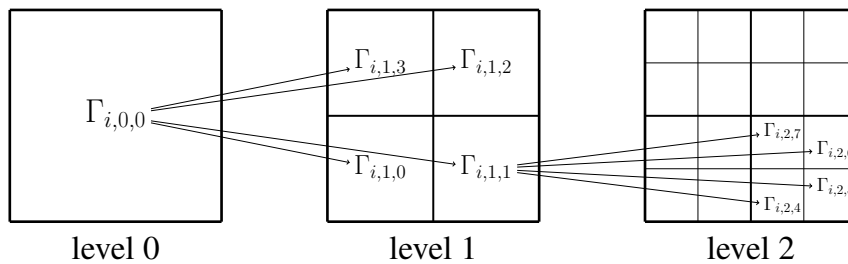


Figure 3. Visualization of the cluster tree.

Remark 2.1. *The proposed method of mesh generation particularly implies that all surface patches are represented by the same number of elements. In fact, this assumption is very similar to the “propagation of refinement” in isogeometric analysis, cf. [14, 15]. An approach to cope with locally refined meshes in the isogeometric context has been made in [24].*

3. PROBLEM FORMULATION

This article focuses on boundary integral equations defined on the closed, parametric surface $\Gamma := \partial\Omega$, i.e.

$$(\mathcal{A}u)(\mathbf{x}) = \int_{\Gamma} k(\mathbf{x}, \mathbf{y})u(\mathbf{y}) d\sigma_{\mathbf{y}} = f(\mathbf{x}). \quad (3)$$

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, it is supposed that the kernel is *asymptotically smooth*, i.e. for all $\mathbf{x}, \mathbf{y} \in \mathbb{R}^3$ holds

$$\left| \partial_{\mathbf{x}}^{\alpha} \partial_{\mathbf{y}}^{\beta} k(\mathbf{x}, \mathbf{y}) \right| \leq c_k \frac{(|\alpha| + |\beta|)!}{r_k^{|\alpha| + |\beta|}} \|\mathbf{x} - \mathbf{y}\|_2^{-2-2q-|\alpha|-|\beta|} \quad (4)$$

with some constants $c_k > 0$ and $r_k > 0$ which are independent of α and β .

The variational formulation of the boundary integral equation (3) reads now as follows:

$$\text{Find } u \in H^q(\Gamma) \text{ such that } (\mathcal{A}u, v)_{L^2(\Gamma)} = (f, v)_{L^2(\Gamma)} \text{ for all } v \in H^q(\Gamma). \quad (5)$$

Inserting the parametric representation (1) of Γ , the bilinear form becomes

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

and the linear form becomes

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

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

$$\left. \begin{aligned} k_{i, i'} : \square \times \square &\rightarrow \mathbb{R}, \\ k_{i, i'}(\mathbf{s}, \mathbf{t}) &:= k(\gamma_i(\mathbf{s}), \gamma_{i'}(\mathbf{t})) \kappa_i(\mathbf{s}) \kappa_{i'}(\mathbf{t}) \end{aligned} \right\} \quad i, i' = 1, 2, \dots, M. \quad (6)$$

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

$$\left| \partial_{\mathbf{s}}^{\alpha} \partial_{\mathbf{t}}^{\beta} k_{i, i'}(\mathbf{s}, \mathbf{t}) \right| \leq c_k \frac{(|\alpha| + |\beta|)!}{r_k^{|\alpha| + |\beta|}} \|\gamma_i(\mathbf{s}) - \gamma_{i'}(\mathbf{t})\|_2^{-(2+2q+|\alpha|+|\beta|)} \quad (7)$$

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

Note that, since the parametric representation is patch-wise smooth, all kernels which satisfy (4) are also analytically standard of order $2q$, see e.g. [25] for a proof of this statement.

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

$$\tau_{j, k} : \square \rightarrow \square_{j, k} \quad \text{for } j = 0, 1, \dots, J \text{ and } k = 0, 1, \dots, 4^j M - 1$$

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

$$k_{\lambda, \lambda'}(\mathbf{s}, \mathbf{t}) := k(\gamma_{\lambda}(\mathbf{s}), \gamma_{\lambda'}(\mathbf{t})) \kappa_{\lambda}(\mathbf{s}) \kappa_{\lambda'}(\mathbf{t}) \quad (8)$$

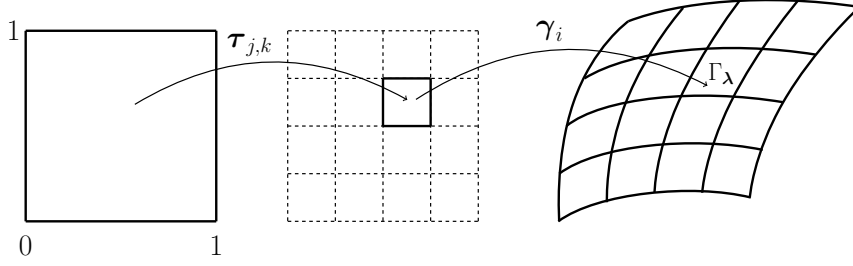


Figure 4. Localized parameterization.

with the *localized parameterizations* $\gamma_\lambda := \gamma_i \circ \tau_{j,k}$ and the corresponding surface measures $\kappa_\lambda := 2^{-2j} \kappa_i \circ \tau_{j,k}$ with κ_i as defined in (2). An illustration of the mappings γ_λ is given by Figure 4.

In the following, only localized kernel functions shall be considered. The next theorem is an immediate consequence of the definition (8) and the fact that $\partial_{\mathbf{s}}^\alpha \tau_{j,k}(\mathbf{s}) = 2^{-j}$ if $|\alpha| = 1$ and $\partial_{\mathbf{s}}^\alpha \tau_{j,k}(\mathbf{s}) = 0$ if $|\alpha| > 1$.

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*

$$|\partial_{\mathbf{s}}^\alpha \partial_{\mathbf{t}}^\beta k_{\lambda, \lambda'}(\mathbf{s}, \mathbf{t})| \leq c_k \frac{(|\alpha| + |\beta|)!}{r_k^{|\alpha| + |\beta|}} \frac{2^{-|\lambda|(|\alpha|+2)} 2^{-|\lambda'|(|\beta|+2)}}{\|\gamma_\lambda(\mathbf{s}) - \gamma_{\lambda'}(\mathbf{t})\|_2^{2+2q+|\alpha|+|\beta|}} \quad (9)$$

holds uniformly for all λ, λ' provided that $2 + 2q + |\alpha| + |\beta| > 0$.

4. GALERKIN DISCRETIZATION

In this section the Galerkin discretization of the variational formulation (5) is considered. To this end, fix a polynomial order $d \in \mathbb{N}$, a level of refinement $j \in \mathbb{N}$, and define the ansatz space

$$\hat{V}_j := \{\hat{\varphi}: \square \rightarrow \mathbb{R}: \hat{\varphi}|_{\square_{j,k}} \text{ is a polynomial of order } d\} \subset L^2(\square) \quad (10)$$

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

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

This construction of the ansatz spaces obviously yields a nested sequence

$$V_0 \subset V_1 \subset \dots \subset V_J \subset H^t(\Gamma), \quad (11)$$

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

By replacing the energy space $H^q(\Gamma)$ in the variational formulation (5) by the finite dimensional ansatz space $V_J \subset H^q(\Gamma)$, the Galerkin discretization for the boundary integral equation (3) is given by:

Find $u_J \in V_J$, such that

$$\int_{\Gamma} \int_{\Gamma} k(\mathbf{x}, \mathbf{y}) u_J(\mathbf{y}) v_J(\mathbf{x}) d\sigma_{\mathbf{y}} d\sigma_{\mathbf{x}} = \int_{\Gamma} f(\mathbf{x}) v_J(\mathbf{x}) d\sigma_{\mathbf{x}} \text{ for all } v_J \in V_J. \quad (12)$$

Setting $\hat{u}_\lambda := u_J \circ \gamma_\lambda$ and $\hat{v}_\lambda := v_J \circ \gamma_\lambda$, rewriting (12) yields

$$\sum_{|\lambda'|=J} \int_{\square} \int_{\square} k_{\lambda,\lambda'}(\mathbf{s}, \mathbf{t}) \hat{u}_{\lambda'}(\mathbf{t}) \hat{v}_\lambda(\mathbf{s}) \, dt \, ds = \int_{\square} f(\gamma_\lambda(\mathbf{s})) \hat{v}_\lambda(\mathbf{s}) \kappa_\lambda(\mathbf{s}) \, ds \quad (13)$$

for all λ with $|\lambda| = J$.

A basis for V_J is obtained by tensorizing polynomial shape functions on $[0, 1]$ and applying the localized parameterizations γ_λ . For $d = 1, 2, 3$, suitable shape functions are depicted in Table I.

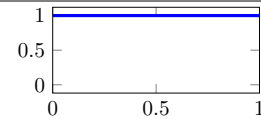
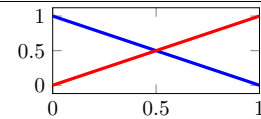
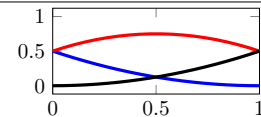
d	Shape Functions	Visualization
1	$\phi^{(i)}(x) = 1$	
2	$\phi^{(i)}(x) = \begin{cases} 1-x \\ x \end{cases}$	
3	$\phi^{(i)}(x) = \begin{cases} (1-x)^2/2 \\ -(x-1/2)^2 + 3/4 \\ x^2/2 \end{cases}$	

Table I. B-spline based shape functions on the interval.

By choosing such a basis, (12) immediately yields a system of linear equations:

$$\mathbf{A}_J \mathbf{u}_J = \mathbf{f}_J. \quad (14)$$

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

Having the Galerkin solution $u_J \in V_J$ at hand yields 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) and $u_J \in V_J$ the related Galerkin solution of (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 (14) is densely populated. This leads to a rather high computational effort for its assembly and for the matrix-vector multiplication, as well. Fortunately, the system matrix is block-wise of low rank, i.e. it is compressible in terms of an \mathcal{H} -matrix, cf. [26]. The computational complexity can thus be reduced drastically by a block-wise compression scheme.

5.1. Block-Cluster Tree

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

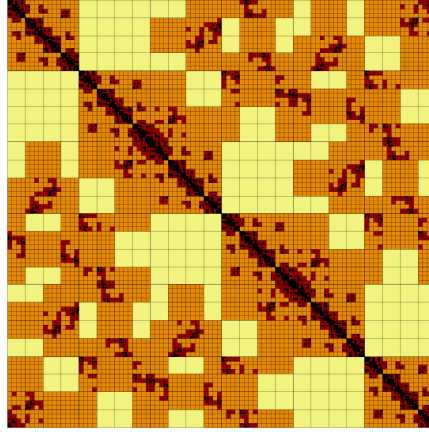


Figure 5. The special block partitioning of the \mathcal{H} -matrix.

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

$$\max \{ \text{diam}(\Gamma_\lambda), \text{diam}(\Gamma_{\lambda'}) \} \leq \eta \text{dist}(\Gamma_\lambda, \Gamma_{\lambda'}) \quad (15)$$

holds for a fixed $\eta \in (0, 1)$. The largest collection of admissible blocks $\Gamma_\lambda \times \Gamma_{\lambda'} \in \mathcal{T} \boxtimes \mathcal{T}$ such that $\Gamma_{\text{dad}(\lambda)} \times \Gamma_{\text{dad}(\lambda')}$ 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 conforms with 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 for a visualization of this special block partitioning of an \mathcal{H} -matrix. Such a special structure is not available in general, cf. [26], and will explicitly be exploited in the construction of the fast multipole method (FMM).

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

```

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

```

5.2. Kernel Interpolation

To compress the admissible matrix blocks, this subsection proposes a black-box version of the FMM based on the interpolation of the kernel $k(\mathbf{x}, \mathbf{y})$ as firstly proposed in [18]. Note that, later on, this idea was also followed in [19] to construct \mathcal{H}^2 -matrices. Nevertheless, in contrast to these works, the approach presented here interpolates the localized kernel (8) on the reference domain rather than the original kernel in space.

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

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

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

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

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

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

$$\mathbf{M}_{|\lambda|}^{\square} = \mathbf{M}_{|\lambda'|}^{\square}. \quad (16)$$

Thus, only a single moment matrix

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

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

$$\begin{aligned} \int_{\square} \mathbf{L}_m(\mathbf{s}) \hat{\phi}_\ell(\mathbf{s}) \, d\mathbf{s} &= \int_0^1 \int_0^1 L_{m_1}(s_1) \hat{\phi}_\ell^{(1)}(s_1) L_{m_2}(s_2) \hat{\phi}_\ell^{(2)}(s_2) \, ds_1 \, ds_2 \\ &= \int_0^1 L_{m_1}(s_1) \hat{\phi}_\ell^{(1)}(s_1) \, ds_1 \int_0^1 L_{m_2}(s_2) \hat{\phi}_\ell^{(2)}(s_2) \, ds_2 \\ &=: [\mathbf{M}_{|\lambda|} \otimes \mathbf{M}_{|\lambda|}]_{\ell, (p+1)m_1+m_2}. \end{aligned}$$

Since

$$\mathbf{M}_{|\lambda|} \in \mathbb{R}^{\sqrt{\dim(\hat{V}_{J-|\lambda|}) \times (p+1)}}, \quad (17)$$

this yields an improved storage complexity for the far-field.

5.3. Computational Complexity

In the sequel, complexity estimates for the FMM under consideration shall be derived. 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 $p > d$, matrix blocks of size $p^2 \times p^2$ are considered as near-field. Thus, these blocks will not be compressed by the FMM. The proof of the next theorem implies that this results in $\mathcal{O}(N_J(p/d)^{-2})$ near-field blocks with a storage cost of $\mathcal{O}(N_J(pd)^2)$, where N_J is the number of elements on level J . Moreover, the following result for the cost complexity of the far-field holds.

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

Proof

At first, it is shown 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 (15). For such clusters, consider the 4 son clusters on level j . Hence, there are $4\mathcal{O}(N_{j-1}) = \mathcal{O}(N_j)$ 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 $N_j = 4^j M$, one may estimate

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

leading to $\mathcal{O}(N_J(p/d)^{-2})$ far-field blocks and accordingly $\mathcal{O}(N_J(p/d)^{-2})$ near-field blocks.

For each far-field block, the localized kernel functions have to be evaluated and stored in $\mathcal{O}(p^4)$ points. The complexity for assembly and storage of the moment matrices is $\mathcal{O}(\sqrt{N_J}pd)$ in total, cf. (17). Consequently, the far-field complexity is

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

□

Remark 5.4. *Due to the parametric surface representation, an improved cost complexity is obtained. The standard interpolation-based FMM proposes to interpolate the kernel in space. Thus, the polynomial degree enters with $\mathcal{O}(p^3)$, cf. [18, 19]. Since the transported kernel is only interpolated on the reference domain, the cost can be reduced 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 $\mathcal{O}(N_J \log N_J p^3 d^2)$, see e.g. [9, 10], whereas for the here presented one one obtains the following result.

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

Proof

On level j , there are $\mathcal{O}(N_j)$ 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

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

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

5.4. Nested Cluster Bases

The cost complexity of the matrix-vector multiplication can be improved to $\mathcal{O}(N_J(pd)^2)$ by exploiting the fact that the explicit computation of the moment matrices $\mathbf{M}_{|\lambda|}$ for each particular level can be avoided by the concept of *nested cluster bases* which amounts to the \mathcal{H}^2 -matrix representation, cf. [19].

Since the polynomial degree for each cluster is p , the Lagrangian polynomials of the father cluster can obviously be represented 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. It holds $\{x_m^{(0)}\}_{m=0}^p \subset [0, 0.5]$ and $\{x_m^{(1)}\}_{m=0}^p \subset [0.5, 1]$. Denoting the corresponding Lagrangian polynomials by $L_m^{(0)}(x)$ and $L_m^{(1)}(x)$, respectively, one 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

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

see Figure 6 for an illustration.

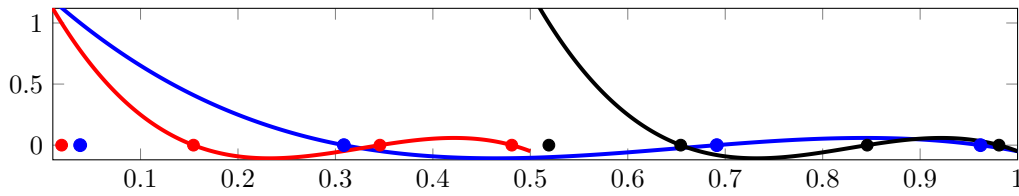


Figure 6. The Lagrange polynomial $L_0(x)$ in the father cluster and the Lagrange polynomials $L_0^{(0)}(x)$ and $L_0^{(1)}(x)$ in the son clusters for $p = 3$.

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

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

Now, by tensor product construction, one obtains the four transfer matrices

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

for the reference domain \square with the refinement relation

$$\mathbf{M}_{|\lambda|}^\square = \begin{bmatrix} \mathbf{M}_{|\lambda|+1}^\square(\mathbf{C}_0^\square)^\top \\ \mathbf{M}_{|\lambda|+1}^\square(\mathbf{C}_2^\square)^\top \\ \mathbf{M}_{|\lambda|+1}^\square(\mathbf{C}_3^\square)^\top \\ \mathbf{M}_{|\lambda|+1}^\square(\mathbf{C}_1^\square)^\top \end{bmatrix}. \quad (18)$$

Notice that the peculiar order of the transfer matrices results from the hierarchical, counter clock-wise ordering of the elements, cf. Figure 3. Fortunately, the transfer matrices $\mathbf{C}_0^\square, \mathbf{C}_1^\square, \mathbf{C}_2^\square, \mathbf{C}_3^\square$ are independent of the respective cluster and even independent of the level $|\lambda|$. 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. [10, 19], only \mathbf{M}_J^\square and $\mathbf{C}_0^\square, \mathbf{C}_1^\square, \mathbf{C}_2^\square, \mathbf{C}_3^\square$ have to be stored. 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  $\mathcal{H}^2$ -MATRIX-VECTOR( $\mathbf{H}, \mathbf{x}, \mathbf{y}$ )
   $\mathbf{u} = \text{FORWARDTRANSFORM}(\mathbf{x})$  ▷ Handle far-field
  for  $\lambda \times \lambda' \in \mathcal{F}$  do
     $\mathbf{v}_\lambda = \mathbf{v}_\lambda + \mathbf{H}|_{\lambda \times \lambda'} \cdot \mathbf{u}_{\lambda'}$ 
  end for
   $\mathbf{y} = \mathbf{y} + \text{BACKWARDTRANSFORM}(\mathbf{v})$ 
  for  $\lambda \times \lambda' \in \mathcal{N}$  do ▷ Handle near-field
     $\mathbf{y}|_\lambda = \mathbf{y}|_\lambda + \mathbf{H}|_{\lambda \times \lambda'} \cdot \mathbf{x}|_{\lambda'}$ 
  end for
end procedure

```

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

```

procedure FORWARDTRANSFORM( $\mathbf{x}$ )
  for  $(i, j', k) \in \mathcal{T}, j' = J$  do
     $\mathbf{u}_{(i, j, k)} = (\mathbf{M}_J^\square)^\top \mathbf{x}|_{(i, j, k)}$ 
  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
      
$$\begin{bmatrix} \mathbf{v}_{(i, j+1, 4k)} \\ \mathbf{v}_{(i, j+1, 4k+1)} \\ \mathbf{v}_{(i, j+1, 4k+2)} \\ \mathbf{v}_{(i, j+1, 4k+3)} \end{bmatrix} = \begin{bmatrix} (\mathbf{C}_0^\square)^\top \\ (\mathbf{C}_2^\square)^\top \\ (\mathbf{C}_3^\square)^\top \\ (\mathbf{C}_1^\square)^\top \end{bmatrix} \mathbf{v}_{(i, j, k)}$$

    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

```

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

Proof

To estimate the complexity of Algorithm 3, note that applying $N_{J-j_{\min}}$ times the moment matrices with $\mathcal{O}(N_{j_{\min}}(pd)^2)$ entries takes at most $\mathcal{O}(N_J(pd)^2)$ 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 one concludes that the

overall complexity of Algorithm 3 is

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

In complete analogy, the complexity of Algorithm 4 is given by $\mathcal{O}(N_J(pd)^2)$. The complexity of the multiplication with the far-field coincides with the complexity of its memory consumption as it has been 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. This yields a total complexity of $\mathcal{O}(N_J(pd)^2)$ for the \mathcal{H}^2 -matrix-vector multiplication. \square

5.5. Error Estimates

In view of Definition 3.1, the following result for the convergence of the FMM holds. 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. [19, 25].

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

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

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

Remark 5.8. *The previous theorem indicates that the convergence behaviour of the FMM directly depends on the properties of the interpolated kernel function. Especially, one has to choose $\eta < r_k$ in (15) in order to obtain convergence. For example, for the fundamental solution of the Helmholtz equation, the constant r_k can become quite small, which in turn increases the near-field.*

From this theorem, one immediately derives an error estimate for the bilinear form which is associated with the variational formulation (5), cf. [18, 25].

Theorem 5.9. *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*

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

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

Hence, in order to maintain the approximation order of the Galerkin method, one has to choose $p \sim \log N_J$. This yields an over-all complexity of $\mathcal{O}(N_J(\log N_J)^2 d^2)$ 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. [27]. 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 $\mathcal{O}((\log N_J)^4)$ for each singular integral if tensor product quadrature is used. Thus, if the quadrature degree is properly decreased with the distance of the elements, one ends up with a complexity of $\mathcal{O}(N_J(\log N_J)^4 d^2)$ for the near-field. The reader is referred to [28] for more details concerning the numerical quadrature.

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. [9, 10]. In the context of higher order ansatz functions, this strategy has been applied to collocation matrices in [17] 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, it is proposed in this article to keep the element-wise cluster strategy introduced in Section 3. In the sequel, easy means to extend the FMM for discontinuous, element-wise polynomial ansatz functions from Section 5 to globally continuous ansatz functions shall be provided. This means that, from now on, ansatz spaces $V_j^c \subset V_j \cap C(\Gamma)$ are considered. Clearly, there exists then a transformation matrix \mathbf{T} such that

$$\mathbf{T}\mathbf{A}_J\mathbf{T}^\top\mathbf{u}_J^c = \mathbf{T}\mathbf{f}_J, \quad (19)$$

where \mathbf{A}_J is the system matrix and \mathbf{f}_J is the right hand side with respect to the discontinuous, element-wise polynomial ansatz functions from (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.

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, introduce the partition

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

Now, setting

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

and using the recursion formula

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

up to $k = d$ will result in the $n + d$ B-spline basis functions of order d on $[0, 1]$, cf. [29]. The B-spline bases up to order 3 are depicted in Figure 7.

It holds $\hat{V}_j^c \subset \hat{V}_j$, such that every function in \hat{V}_j^c can be expressed as a linear combination of functions in \hat{V}_j . Let $\hat{\mathbf{U}} = [\hat{\varphi}_1^c, \dots, \hat{\varphi}_{\hat{n}_j^c}]$ 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 $\hat{n}_j^c = \dim(\hat{V}_j^c)$ 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

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

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

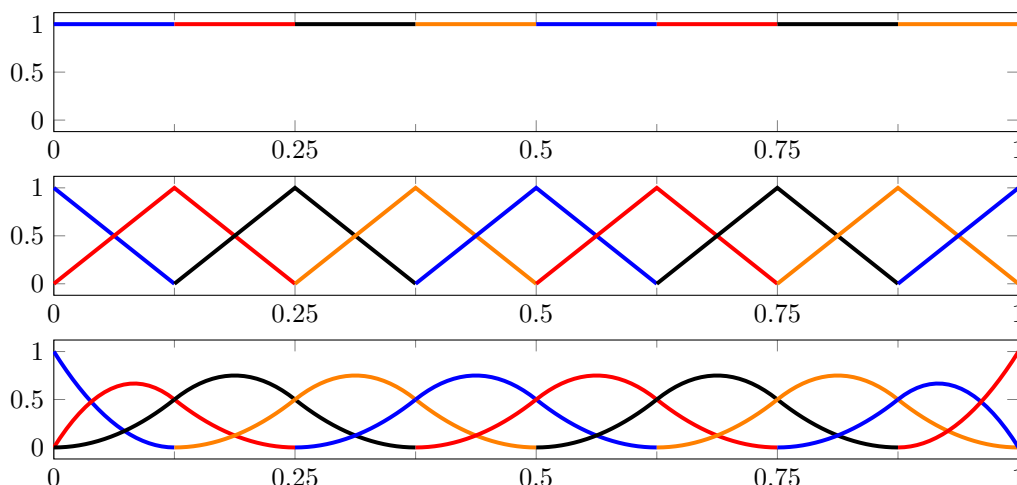


Figure 7. B-spline bases on the unit interval $[0, 1]$ of order 1 (top), order 2 (middle), and order 3 (bottom).

B-splines on level j is given by

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

Since $V_j^c \subset V_j$, every function in V_j^c can be expressed 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

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

Setting $\mathbf{b}_i = [b_{i,1}, \dots, b_{i,\hat{n}_j}]$ finally yields

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

The matrices $\hat{\mathbf{T}}$ and \mathbf{I} are sparse and so is their product \mathbf{T} . Thus, the transformation from V_j to V_j^c can be done in an efficient manner.

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 (19) is n_J^2 , where $n_J = \dim(V_J)$, instead of $(n_J^c)^2$, i.e. the memory consumption will grow like $\mathcal{O}((n_J/n_J^c)^2)$, whereas the number of degrees of freedom only grows like $\mathcal{O}(n_J/n_J^c)$. 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 $\mathcal{O}((n_J/n_J^c)^2)$. 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 the 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

The numerical experiments focus on the numerical solution of boundary integral equations which amount from the reformulation of the Laplace equation in a three-dimensional Lipschitz domain $\Omega \in \mathbb{R}^3$ with either Dirichlet boundary conditions

$$\Delta U = 0 \text{ in } \Omega, \quad U = f \text{ on } \Gamma, \quad (20)$$

or Neumann boundary conditions

$$\Delta U = 0 \text{ in } \Omega, \quad \frac{\partial U}{\partial \mathbf{n}} = g \text{ on } \Gamma. \quad (21)$$

The first two boundary integral equations under consideration stem from the single layer potential ansatz, which reads

$$U(\mathbf{x}) = \int_{\Gamma} \frac{u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2} d\sigma_{\mathbf{y}} = \tilde{\mathcal{S}}u(\mathbf{x}), \quad \mathbf{x} \in \Omega. \quad (22)$$

This leads to a Fredholm integral equation of the first kind for the Dirichlet problem,

$$\mathcal{S}u(\mathbf{x}) = \int_{\Gamma} \frac{u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2} d\sigma_{\mathbf{y}} = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (23)$$

and to a Fredholm integral equation of the second kind for the Neumann problem,

$$\frac{1}{2}u(\mathbf{x}) + \mathcal{K}^*u(\mathbf{x}) = \frac{1}{2}u(\mathbf{x}) + \int_{\Gamma} \frac{\langle \mathbf{n}_{\mathbf{x}}, \mathbf{y} - \mathbf{x} \rangle u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2^3} d\sigma_{\mathbf{y}} = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (24)$$

for the unknown density u . The second two boundary integral equations are obtained from a double layer potential ansatz

$$U(\mathbf{x}) = \int_{\Gamma} \frac{\langle \mathbf{x} - \mathbf{y}, \mathbf{n}_{\mathbf{y}} \rangle u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2^3} d\sigma_{\mathbf{y}} = \tilde{\mathcal{K}}u(\mathbf{x}), \quad \mathbf{x} \in \Omega, \quad (25)$$

which leads to a Fredholm integral equation of the second kind for the Dirichlet problem,

$$\frac{1}{2}u(\mathbf{x}) - \mathcal{K}u(\mathbf{x}) = \frac{1}{2}u(\mathbf{x}) - \int_{\Gamma} \frac{\langle \mathbf{x} - \mathbf{y}, \mathbf{n}_{\mathbf{y}} \rangle u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2^3} d\sigma_{\mathbf{y}} = f(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (26)$$

and to a Fredholm integral equation of the first kind for the Neumann problem,

$$\mathcal{W}u(\mathbf{x}) = \frac{\partial}{\partial \mathbf{n}_{\mathbf{x}}} \int_{\Gamma} \frac{\langle \mathbf{x} - \mathbf{y}, \mathbf{n}_{\mathbf{y}} \rangle u(\mathbf{y})}{4\pi\|\mathbf{x} - \mathbf{y}\|_2^3} d\sigma_{\mathbf{y}} = g(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad (27)$$

for the unknown density u . The numerical treatment of (27) is the topic of the next subsection.

The ansatz functions shall be B-splines of order $d = 1, 2, 3$, 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 (19) of linear equations, a conjugate gradient method (CG) for (23) and (27) and the generalized minimal residual method (GMRES) with restart after 100 inner iterations for (24) and (26) are used, cf. [30]. The construction of appropriate preconditioners for the iterative solvers 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 (23), (24), (26) or (27). 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 (23), $2q = 0$ in case of (24) and (26), and $2q = 1$ in case of (27).

Proof

Together with Theorem 4.1, there holds

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

for all four cases under consideration. \square

7.2. Regularization of the Hypersingular Operator

The hypersingular operator \mathcal{W} from (27) is formally given by the integral operator

$$\mathcal{W}u(\mathbf{x}) = \frac{1}{4\pi} \int_{\Gamma} \left(\frac{\langle \mathbf{n}_{\mathbf{x}}, \mathbf{n}_{\mathbf{y}} \rangle}{\|\mathbf{x} - \mathbf{y}\|_2^3} - 3 \frac{\langle \mathbf{n}_{\mathbf{x}}, \mathbf{x} - \mathbf{y} \rangle \langle \mathbf{n}_{\mathbf{y}}, \mathbf{x} - \mathbf{y} \rangle}{\|\mathbf{x} - \mathbf{y}\|_2^5} \right) u(\mathbf{y}) \, d\sigma_{\mathbf{y}}, \quad (28)$$

which has a strong singularity in its kernel function. Therefore, a regularization is needed in order to apply quadrature rules for its discretization. The following representation, derived firstly in [31], reduces the singularity of the integral operator in the inner product to the same singularity as in (23):

$$(\mathcal{W}\rho, \eta)_{L^2(\Gamma)} = (\mathcal{S} \operatorname{curl}_{\Gamma} \rho, \operatorname{curl}_{\Gamma} \eta)_{L^2(\Gamma)} \quad \text{for all } \rho, \eta \in H^1(\Gamma). \quad (29)$$

It remains to find an easy means to compute the surface curl of a given function in $H^1(\Gamma)$.

For each patch $\Gamma_i = \gamma_i(\square)$, consider the first fundamental tensor of differential geometry, cf. [32],

$$\mathbf{K}_i(\mathbf{s}) = \begin{bmatrix} \langle \partial_{s_1} \gamma_i(\mathbf{s}), \partial_{s_1} \gamma_i(\mathbf{s}) \rangle & \langle \partial_{s_1} \gamma_i(\mathbf{s}), \partial_{s_2} \gamma_i(\mathbf{s}) \rangle \\ \langle \partial_{s_2} \gamma_i(\mathbf{s}), \partial_{s_1} \gamma_i(\mathbf{s}) \rangle & \langle \partial_{s_2} \gamma_i(\mathbf{s}), \partial_{s_2} \gamma_i(\mathbf{s}) \rangle \end{bmatrix}, \quad \mathbf{s} = [s_1, s_2]^{\top} \in \square.$$

Since γ_i is a diffeomorphism, \mathbf{K}_i is a symmetric and positive definite matrix and every function $\rho|_{\Gamma_i}$ has a unique representation $\tilde{\rho}_i = \rho \circ \gamma_i$ on \square . Thus, for $\mathbf{x} = \gamma_i(\mathbf{s}) \in \Gamma_i$, one can define the surface gradient of ρ by

$$\operatorname{grad}_{\Gamma} \rho(\mathbf{x}) = \begin{bmatrix} \partial_{s_1} \gamma_i(\mathbf{s}) & \partial_{s_2} \gamma_i(\mathbf{s}) \end{bmatrix} \mathbf{K}_i^{-1}(\mathbf{s}) \begin{bmatrix} \partial_{s_1} \tilde{\rho}_i(\mathbf{s}) \\ \partial_{s_2} \tilde{\rho}_i(\mathbf{s}) \end{bmatrix}, \quad (30)$$

cf. [32]. Note that this definition is independent of the actual chosen parametrization γ_i .

Having the definition of the surface gradient at hand, the surface curl is defined by

$$\operatorname{curl}_{\Gamma} \rho(\mathbf{x}) = \mathbf{n}_{\mathbf{x}} \times \operatorname{grad}_{\Gamma} \rho(\mathbf{x}), \quad \mathbf{x} \in \Gamma. \quad (31)$$

Inserting (30) into (31) yields

$$\operatorname{curl}_{\Gamma} \rho(\mathbf{x}) = \frac{1}{\sqrt{\det \mathbf{K}_i(\mathbf{s})}} (\partial_{s_1} \tilde{\rho}_i(\mathbf{s}) \partial_{s_2} \gamma_i(\mathbf{s}) - \partial_{s_2} \tilde{\rho}_i(\mathbf{s}) \partial_{s_1} \gamma_i(\mathbf{s})), \quad (32)$$

where $\sqrt{\det \mathbf{K}_i(\mathbf{s})} = \kappa_i(\mathbf{s})$ is the surface measure of Γ_i from (2).

Finally, recall that an ansatz function φ_i on the surface Γ_i is given by $\varphi_i = \hat{\varphi} \circ \gamma_i^{-1}$, where $\hat{\varphi} \in \hat{V}_j$ is given as in (10). It therefore holds

$$\partial_{s_k} \tilde{\varphi}_i(\mathbf{s}) = \partial_{s_k} (\hat{\varphi} \circ \gamma_i^{-1} \circ \gamma_i)(\mathbf{s}) = \partial_{s_k} \hat{\varphi}(\mathbf{s}), \quad k = 1, 2.$$

Thus, in order to numerically compute (32) for the ansatz and the test space, one only has to provide the derivatives of the tensorized local shape functions on the unit square from Table I. Hence, the construction from Section 5.2, applied to (29) with $\operatorname{curl}_{\Gamma}$ computed as in (32), results in a slightly

modified approximation for admissible matrix blocks:

$$\mathbf{A}_{\lambda, \lambda'} \approx \begin{bmatrix} \mathbf{M}_{|\lambda|}^{1, \square} & \mathbf{M}_{|\lambda|}^{2, \square} \end{bmatrix} \begin{bmatrix} \mathbf{K}_{\lambda, \lambda'}^{(1,1)} & \mathbf{K}_{\lambda, \lambda'}^{(1,2)} \\ \mathbf{K}_{\lambda, \lambda'}^{(2,1)} & \mathbf{K}_{\lambda, \lambda'}^{(2,2)} \end{bmatrix} \begin{bmatrix} (\mathbf{M}_{|\lambda|}^{1, \square})^\top \\ (\mathbf{M}_{|\lambda|}^{2, \square})^\top \end{bmatrix}.$$

Herein, the modified moment matrices are given by

$$\begin{aligned} (\mathbf{M}_{|\lambda|}^\partial)_{m_1, \ell} &= \int_0^1 L_{m_1}(s_1) (\partial_{s_1} \hat{\phi}_\ell^{(1)})(s_1) ds_1, \\ \mathbf{M}_{|\lambda|}^{1, \square} &= \mathbf{M}_{|\lambda|}^\partial \otimes \mathbf{M}_{|\lambda|}, \quad \mathbf{M}_{|\lambda|}^{2, \square} = \mathbf{M}_{|\lambda|} \otimes \mathbf{M}_{|\lambda|}^\partial, \end{aligned}$$

and the modified kernel matrices by

$$\begin{aligned} [\mathbf{K}_{\lambda, \lambda'}^{(1,1)}]_{\mathbf{m}, \mathbf{m}'} &= k(\gamma_\lambda(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}(\mathbf{x}_{\mathbf{m}'})) \langle \gamma_\lambda^{\partial_2}(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}^{\partial_2}(\mathbf{x}_{\mathbf{m}'}) \rangle, \\ [\mathbf{K}_{\lambda, \lambda'}^{(1,2)}]_{\mathbf{m}, \mathbf{m}'} &= -k(\gamma_\lambda(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}(\mathbf{x}_{\mathbf{m}'})) \langle \gamma_\lambda^{\partial_2}(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}^{\partial_1}(\mathbf{x}_{\mathbf{m}'}) \rangle, \\ [\mathbf{K}_{\lambda, \lambda'}^{(2,1)}]_{\mathbf{m}, \mathbf{m}'} &= -k(\gamma_\lambda(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}(\mathbf{x}_{\mathbf{m}'})) \langle \gamma_\lambda^{\partial_1}(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}^{\partial_2}(\mathbf{x}_{\mathbf{m}'}) \rangle, \\ [\mathbf{K}_{\lambda, \lambda'}^{(2,2)}]_{\mathbf{m}, \mathbf{m}'} &= k(\gamma_\lambda(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}(\mathbf{x}_{\mathbf{m}'})) \langle \gamma_\lambda^{\partial_1}(\mathbf{x}_\mathbf{m}), \gamma_{\lambda'}^{\partial_1}(\mathbf{x}_{\mathbf{m}'}) \rangle, \end{aligned}$$

where the *localized parametrization derivatives* are defined by $\gamma_\lambda^{\partial_\ell} = \partial_{s_\ell} \gamma_i \circ \tau_{j,k}$, $\ell = 1, 2$.

Note that the very same refinement relation (18) holds between $\mathbf{M}_{|\lambda|}^{i, \square}$ and its predecessors $\mathbf{M}_{|\lambda|+1}^{i, \square}$, $i = 1, 2$. Thus, by a small but straightforward modification, the \mathcal{H}^2 -matrix-vector multiplication can also be applied in the case of the regularized hypersingular operator.

7.3. Convergence

The first example shall be concerned with the boundary value problems (20) and (21) in the unit ball with a prescribed analytical solution based on the spherical harmonic Y_0^2 ,

$$U(\mathbf{x}) = \|\mathbf{x}\|_2 Y_0^2\left(\frac{\mathbf{x}}{\|\mathbf{x}\|_2}\right), \quad \mathbf{x} \in \Omega, \quad Y_0^2(\mathbf{x}) = \sqrt{\frac{5}{16\pi}}(3x_3^2 - 1), \quad \mathbf{x} \in \Gamma,$$

see Figure 8 for a visualization. The Dirichlet and Neumann data for (20) and (21) are then given by

$$f(\mathbf{x}) = Y_0^2(\mathbf{x}), \quad \mathbf{x} \in \Gamma, \quad g(\mathbf{x}) = (\nabla U(\mathbf{x}), \mathbf{x}), \quad \mathbf{x} \in \Gamma.$$

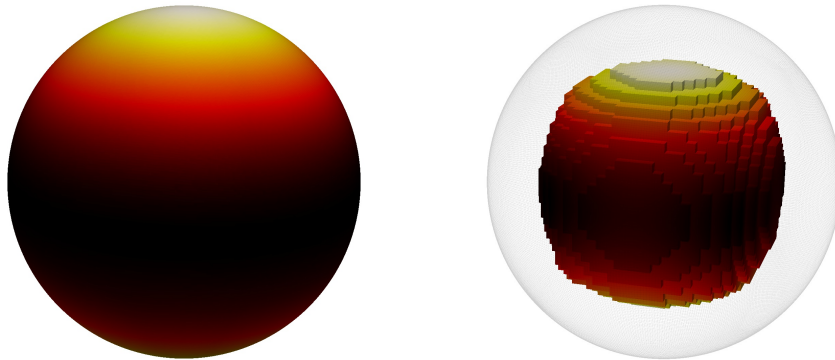


Figure 8. The spherical harmonic Y_0^2 (left) and the related potential (right) for the unit ball.

Furthermore, on the sphere, the spherical harmonic is an eigenfunction to all integral operators under consideration, i.e. there holds

$$\begin{aligned} \mathcal{V}Y_0^2 &= \frac{1}{5}Y_0^2, & \left(\frac{1}{2} - \mathcal{K}\right)Y_0^2 &= -\frac{3}{5}Y_0^2, \\ \left(\frac{1}{2} + \mathcal{K}^*\right)Y_0^2 &= \frac{2}{5}Y_0^2, & \mathcal{W}Y_0^2 &= \frac{6}{5}Y_0^2. \end{aligned}$$

Hence, the analytical solution of (23), (24), (26), and (27) is known and one can thus compute the $L^2(\Gamma)$ -error of the approximate density. For the following results, the boundary of the sphere is represented by six patches.

Figure 9 and 10 validate that the proposed FMM provides the theoretical convergence rates on smooth domains in case of the Dirichlet problem. Figure 11 and 12 validate this for the Neumann problem. Note that the hypersingular operator in Figure 12 requires globally continuous ansatz functions, thus the order of the ansatz functions for this particular example has to be at least $d = 2$. In both cases, the relative 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 8. The polynomial degree p for FMM is chosen such that the overall accuracy is sustained. Hence, in accordance with Theorem 5.9, 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 II. Note that both numbers coincide in case of piecewise constant boundary elements, i.e. for $d = 1$.

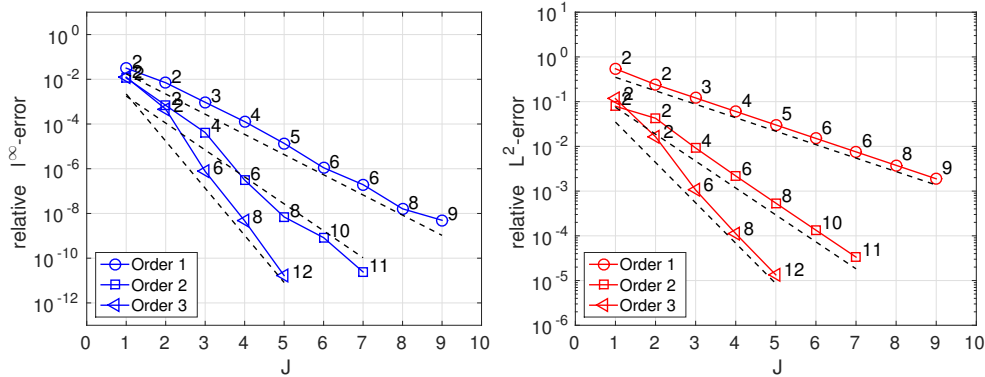


Figure 9. Relative l^∞ -error (left) and relative L^2 -error (right) for the Dirichlet problem on the sphere for the single layer ansatz. The corresponding theoretical convergence rates are h^3 , h^5 and h^7 for the potential and h^1 , h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

7.4. Convergence on a More Complex Geometry

The second example shall deal with the more complex gear worm geometry depicted in Figure 13, which is represented by 290 patches. The harmonic polynomial

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

is prescribed as potential and used to obtain the Dirichlet and Neumann data for (20) and (21). For sake of brevity, we restrict ourselves to one example for Dirichlet problems, namely (23), and one example for Neumann problems, namely (27). Since the density u is unknown, the error of the potential is measured on the 115'241 vertices of a grid of 83'437 cubes fitted inside the domain. A visualization of the cubes together with the computed density for the single layer ansatz for Dirichlet data can be found in Figure 13.

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 ansatz and to h^1 for the hypersingular ansatz. Figure 14

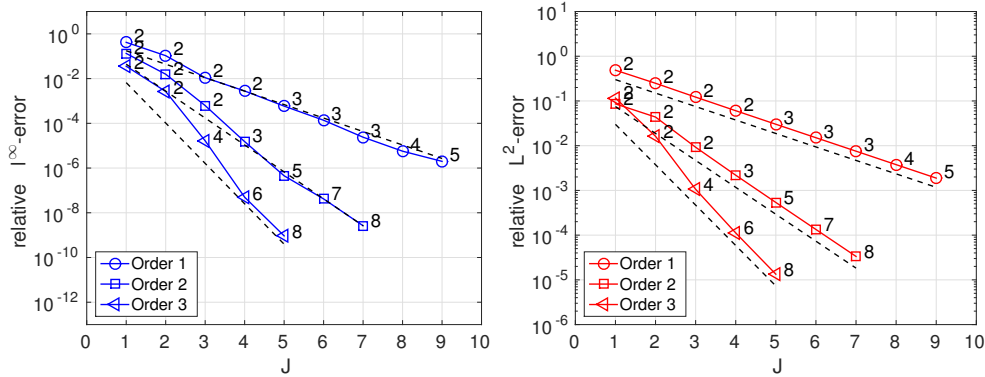


Figure 10. Relative l^∞ -error (left) and relative L^2 -error (right) for the Dirichlet problem on the sphere for the double layer ansatz. The corresponding theoretical convergence rates are h^2 , h^4 and h^6 for the potential and h^1 , h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

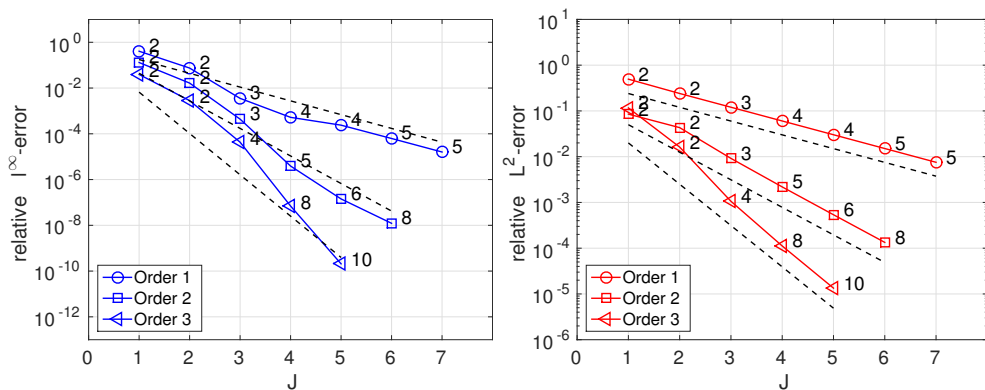


Figure 11. Relative l^∞ -error (left) and relative L^2 -error (right) on the sphere for the Neumann problem for the adjoint double layer ansatz. The corresponding theoretical convergence rates are h^2 , h^4 and h^6 for the potential and h^1 , h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

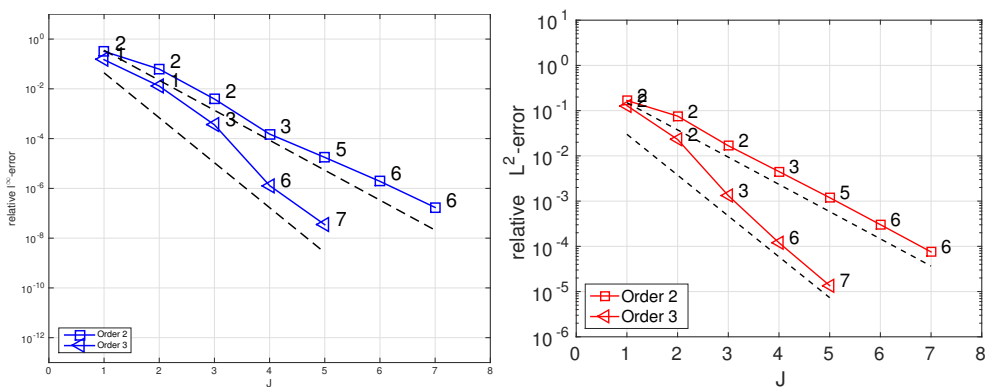


Figure 12. Relative l^∞ -error (left) and relative L^2 -error (right) on the sphere for the Neumann problem for the hypersingular ansatz. The corresponding theoretical convergence rates are h^3 and h^5 for the potential and h^2 and h^3 for the density. The accompanying numbers are the polynomial degrees of the interpolation.

illustrates that these convergence rates are achieved for all ansatz functions under consideration. In fact, the higher order ansatz functions even seem to produce a convergence rate of up to h^5 for the

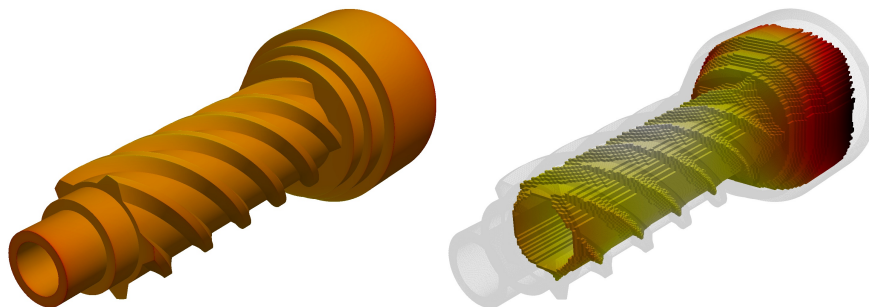


Figure 13. The approximate density of the single layer ansatz for Dirichlet data (left) and the related potential (right) for the gear worm.

single layer ansatz and the hypersingular 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 II.

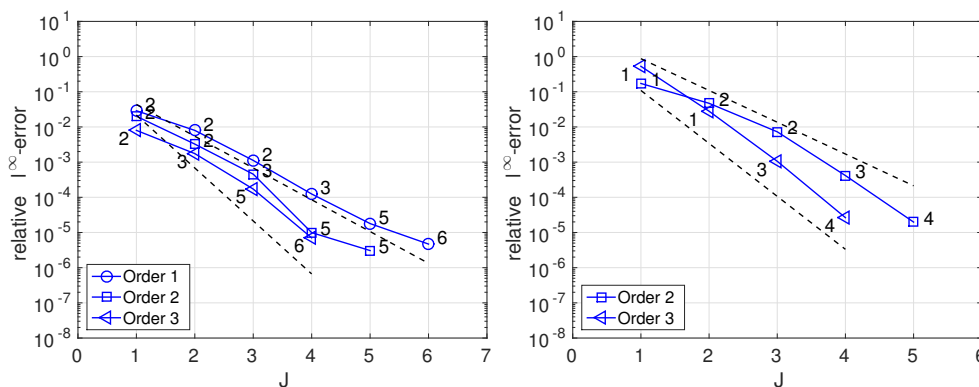


Figure 14. Relative l^∞ -error on the gear worm for the Dirichlet problem for the single layer ansatz (left) and Neumann problem for the hypersingular ansatz (right). The corresponding convergence rates are h^3 and h^5 . The accompanying numbers are the polynomial degrees of the interpolation.

7.5. Computational Cost and Accuracy

In a first example, the benefit of the \mathcal{H}^2 -matrix-vector multiplication compared to the \mathcal{H} -matrix-vector multiplication shall be demonstrated. To that end, the computation times for a non-symmetric matrix-vector multiplication of an \mathcal{H} -matrix and of an \mathcal{H}^2 -matrix are measured, 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 15 illustrates that an asymptotic complexity of $\mathcal{O}(N_J(pd)^2)$ for the \mathcal{H}^2 -matrix-vector multiplication compared to the complexity of $\mathcal{O}(N_J \log N_J(pd)^2)$ for the \mathcal{H} -matrix-vector multiplication is reached, leading to the conclusion that the \mathcal{H}^2 -matrix-vector multiplication is the method of choice. Thus, all of the following experiments are based on the \mathcal{H}^2 -matrix-vector multiplication.

In a second example the effectiveness of higher order ansatz functions shall be illustrated for the examples from Sections 7.3 and 7.4. To that end, the l^∞ -error of the potential is compared to the computation time of the matrix and to the computation time of the matrix plus the solving time. The results with respect to the sphere are depicted in Figure 16 for the Dirichlet problem and in Figure 17 for the Neumann problem. The results with respect to the gear worm are depicted in Figure 18. They indicate that the higher order ansatz functions achieve asymptotically a higher precision at a lower computation time. It therefore seems that the higher order FMM is favourable even on non-smooth

		$n_J = \dim V_J$			$n_J^c = \dim V_J^c$		
		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 II. 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.

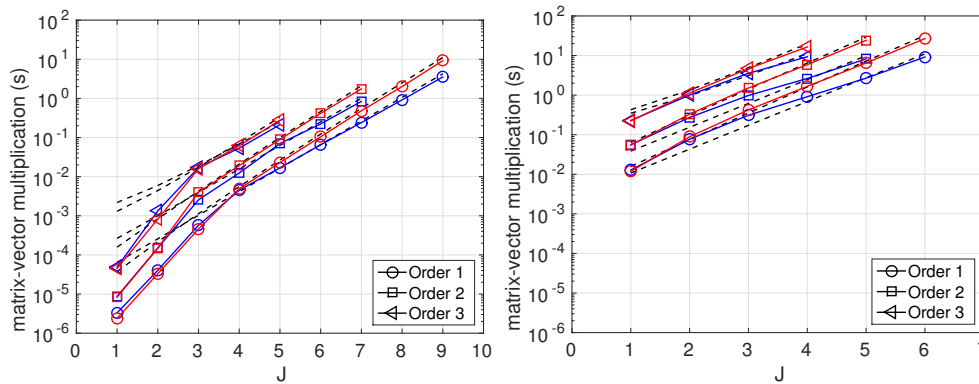


Figure 15. Computation times for the \mathcal{H}^2 -matrix-vector multiplication (blue) and the \mathcal{H} -matrix-vector multiplication (red) on the sphere (left) and on the gear worm (right). The dashed lines illustrate the complexity rates $\mathcal{O}(N_J(pd)^2)$ and $\mathcal{O}(N_J \log N_J(pd)^2)$.

surfaces. Note that the increased solving times for the gear worm geometry are due to a higher number of iterations in the solving process.

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. This article presents 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, one can easily deal with higher order ansatz functions. The complexity estimates as well as the numerical examples demonstrate the superior performance of the presented method.

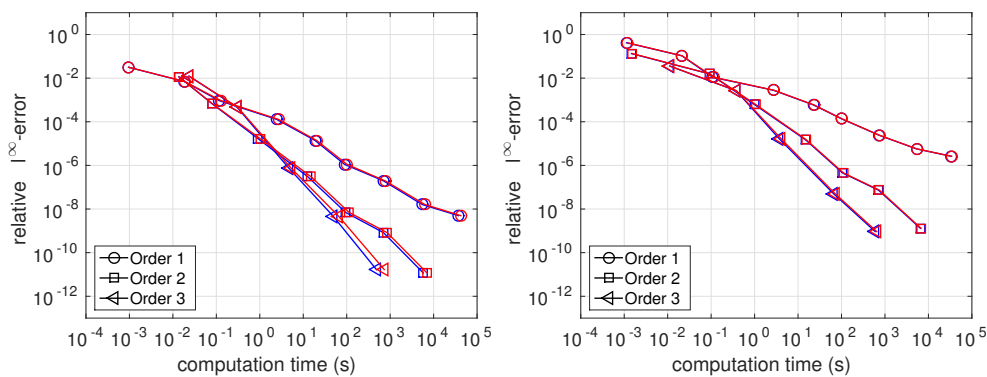


Figure 16. Relative l^∞ -error versus the computation time of the matrix (blue) and the computation time of the matrix plus the solving time (red) for the single layer ansatz (left) and the double layer ansatz (right) on the sphere for Dirichlet data.

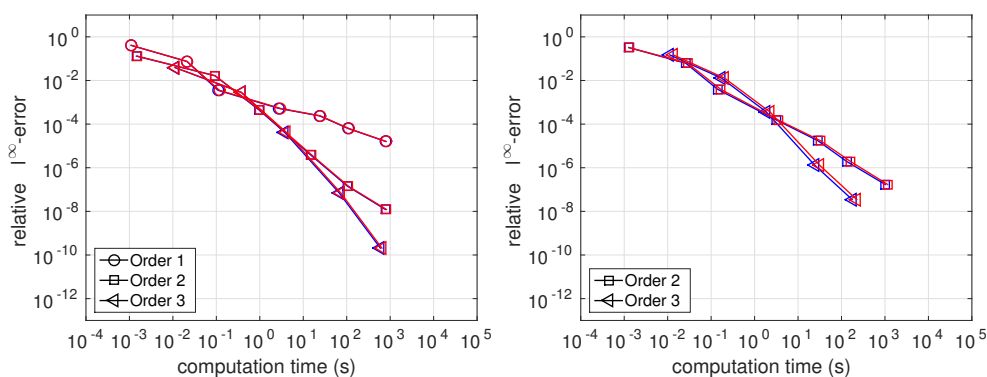


Figure 17. Relative l^∞ -error versus the computation time of the matrix (blue) and the computation time of the matrix plus the solving time (red) for the adjoint double layer ansatz (left) and the hypersingular ansatz (right) on the sphere for Neumann data.

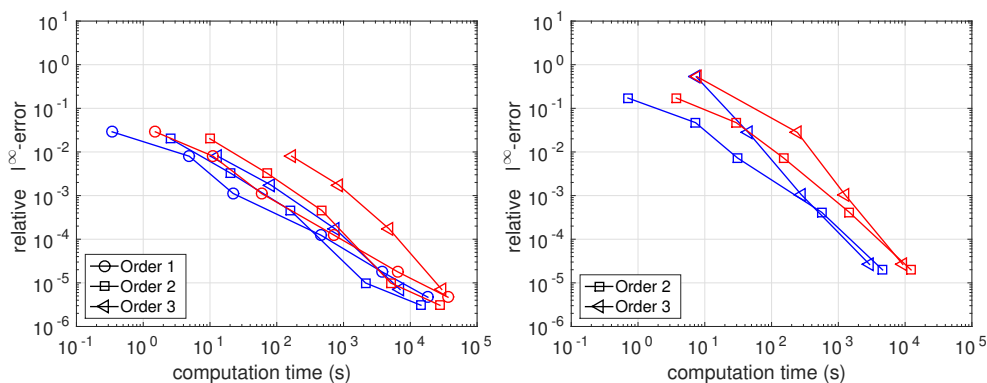


Figure 18. Relative l^∞ -error versus the computation time of the matrix (blue) and the computation time of the matrix plus the solving time (red) for the single layer ansatz and Dirichlet data (left) and the hypersingular ansatz and Neumann data (right) on the gear worm.

REFERENCES

1. Hackbusch W. *Integral Equations: Theory and Numerical Treatment*, vol. 4. Birkhäuser: Basel, 1995.
2. Sauter SA, Schwab C. *Boundary Element Methods*. Springer: Berlin-Heidelberg, 2011.
3. Steinbach O. *Numerical Approximation Methods for Elliptic Boundary Value Problems*. Springer Science + Business: New York, 2008.

4. Greengard L, Rokhlin V. A fast algorithm for particle simulations. *Journal of Computational Physics* 1987; **73**(2):325–348.
5. Hackbusch W, Nowak ZP. On the fast matrix multiplication in the boundary element method by panel clustering. *Numerische Mathematik* 1989; **54**(4):463–491.
6. Beylkin G, Coifman R, Rokhlin V. Fast wavelet transforms and numerical algorithms I. *Communications on Pure and Applied Mathematics* 1991; **44**(2):141–183.
7. Dahmen W, Harbrecht H, Schneider R. Compression techniques for boundary integral equations. Asymptotically optimal complexity estimates. *SIAM Journal on Numerical Analysis* 2006; **43**(6):2251–2271.
8. Bebendorf M. Approximation of boundary element matrices. *Numerische Mathematik* 2000; **86**(4):565–589.
9. Hackbusch W. *Hierarchische Matrizen: Algorithmen und Analysis*. Springer: Heidelberg, 2009.
10. Börm S. *Efficient Numerical Methods for Non-local Operators, EMS Tracts in Mathematics*, vol. 14. European Mathematical Society (EMS), Zürich, 2010.
11. *Initial Graphics Exchange Specification. IGES 5.3*. U.S. Product Data Association 1996.
12. *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.
13. Hoschek J, Lasser D. *Grundlagen der Geometrischen Datenverarbeitung*. Teubner: Stuttgart, 1989.
14. Cottrell JA, Hughes TJR, Bazilevs Y. *Isogeometric Analysis: Toward Integration of CAD and FEA*. 1st edn., Wiley Publishing, 2009.
15. Hughes TJR, Cottrell JA, Bazilevs Y. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering* 2005; **194**(39):4135–4195.
16. Marussig B, Zechner J, Beer G, Fries T. Fast isogeometric boundary element method based on independent field approximation. *Computer Methods in Applied Mechanics and Engineering* 2015; **284**(0):458–488. Isogeometric Analysis Special Issue.
17. Zechner J, Marussig B, Beer G, Fries T. Isogeometric boundary element method with hierarchical matrices. *CoRR* 2014; **abs/1406.2817**.
18. Giebermann K. Multilevel approximation of boundary integral operators. *Computing* 2001; **67**(3):183–207.
19. Hackbusch W, Börm S. \mathcal{H}^2 -matrix approximation of integral operators by interpolation. *Applied Numerical Mathematics* 2002; **43**(1-2):129–143.
20. Harbrecht H, Peters M, Siebenmorgen M. Efficient approximation of random fields for numerical applications. *Numerical Linear Algebra with Applications* 2015; **22**(4):596–617.
21. Harbrecht H, Randrianarivony M. From computer aided design to wavelet BEM. *Computing and Visualization in Science* 2010; **13**(2):69–82.
22. Harbrecht H, Randrianarivony M. Wavelet BEM on molecular surfaces: Parametrization and implementation. *Computing* 2009; **86**(1):1–22.
23. Harbrecht H, Randrianarivony M. Wavelet BEM on molecular surfaces: Solvent excluded surfaces. *Computing* 2011; **92**(4):335–364.
24. Cottrell J, Hughes T, Reali A. Studies of refinement and continuity in isogeometric structural analysis. *Computer Methods in Applied Mechanics and Engineering* 2007; **196**(41–44):4160 – 4183.
25. Harbrecht H, Peters M. Comparison of fast boundary element methods on parametric surfaces. *Computer Methods in Applied Mechanics and Engineering* 2013; **261–262**:39–55.
26. Hackbusch W. A sparse matrix arithmetic based on \mathcal{H} -matrices part I: Introduction to \mathcal{H} -matrices. *Computing* 1999; **62**(2):89–108.
27. Duffy MG. Quadrature over a pyramid or cube of integrands with a singularity at a vertex. *SIAM Journal on Numerical Analysis* 1982; **19**(6):1260–1262.
28. Sauter SA, Schwab C. Quadrature for hp -Galerkin BEM in \mathbb{R}^3 . *Numerische Mathematik* 1997; **78**(2):211–258.
29. De Boor C. *A Practical Guide to Splines*. Springer: New York, 1978.
30. Saad Y. *Iterative Methods for Sparse Linear Systems*. Second edn., Society for Industrial and Applied Mathematics, 2003.
31. Nedelec J. Integral equations with non integrable kernels. *Integral Equations and Operator Theory* 1982; **5**(1):562–572.
32. Colton D, Kress R. *Integral equation methods in scattering theory*, vol. 72. Society for Industrial and Applied Mathematics, 2013.