Unterschrift des Betreuers

D I P L O M A R B E I T

Adaptive Isogeometrische BEM

geschrieben am
Institut für Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.Math. Dr.techn. Dirk Praetorius
Dipl.-Ing. Michael Feischl

durch

Gregor Gantner
Siedlung 36
2054 Haugsdorf

D I P L O M A T H E S I S

Adaptive Isogeometric BEM

written at the
Institute for Analysis and Scientific Computing
of the Vienna University of Technology

supervised by
Ao.Univ.Prof. Dipl.Math. Dr.techn. Dirk Praetorius
Dipl.-Ing. Michael Feischl

by

Gregor Gantner
Siedlung 36
2054 Haugsdorf

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Unterschrift:

Danksagung

Mein Dank gilt vor allem Professor Dirk Praetorius. Durch viele sehr gute Vorlesungen hat er mein Interesse an der Numerik geweckt. Das Thema, welches er mir für die Diplomarbeit vorgeschlagen hat, hat mich bis zum Schluss fasziniert.

Weiters möchte ich mich bei meinem Ko-Betreuer Michael Feischl herzlich bedanken. Traten während der Arbeit Unklarheiten auf, so nahm er sich stets für mich Zeit und half mir mit fruchtbaren Denkanstößen weiter.

Außerden möchte ich meinen Eltern Dank aussprechen, dafür dass sie mich stets in meinen Bestrebungen unterstüzt haben.

Nicht zuletzt will ich mich bei meinen Studienkollegen bedanken, mit denen ich eine wunderschöne Zeit verbringen durfte und hoffentlich auch zukünftig verbringen werde.

Finanziell wurde diese Arbeit durch das FWF Projekt *Adaptive Boundary Element Method* (P21732) unterstützt.

Contents

1. Introduction

Throughout this work, we consider an open two-dimensional set Ω whose boundary Γ can be parametrized by some piecewise smooth path γ, and a subset Γ_D of the boundary. For given right hand side $F \in H^{1/2}(\Gamma_D)$, we want to numerically approximate the solution $\phi \in \widetilde{H}^{-1/2}(\Gamma_D)$ of the boundary integral equation

$$V\phi = F. \tag{1.1}$$

Here $V : \widetilde{H}^{-1/2}(\Gamma_D) \to H^{1/2}(\Gamma_D)$ denotes a continuous and linear operator such that $(\chi, \psi) \mapsto \langle V\chi, \psi\rangle$ defines an elliptic bilinear form on $\widetilde{H}^{-1/2}(\Gamma_D)$. This problem is of practical interest, since the Dirichlet problem of elliptic partial differential equations can equivalently be written in such a form; see e.g. [McL00, Ste08].

As example serves the *Poisson problem with Dirichlet boundary conditions* for a two-dimensional bounded Lipschitz domain Ω: Find u such that

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma. \end{aligned} \tag{1.2}$$

Here f is a given volume force on Ω and g are given Dirichlet boundary conditions on Γ. A *classical solution* $u \in C^2(\overline{\Omega})$ satisfies (1.2) pointwise. In general, one can not expect that classical solutions exist. Instead one seeks for so-called *weak solutions* u in the Sobolev space $H^1(\Omega)$ with $u|_\Gamma = g$ and

$$\forall v \in H^1_0(\Omega) : \quad \langle \nabla u, \nabla v\rangle_{L^2(\Gamma)} = \langle f, v\rangle. \tag{1.3}$$

Here, we assume that $f \in \widetilde{H}^{-1}(\Omega)$ and $g \in H^{1/2}(\Gamma)$. There always exists a unique weak solution. The normal derivative $\phi := \partial u/\partial\nu \in H^{-1/2}(\Gamma)$ of this weak solution solves *Symm's integral equation*

$$V\phi = (K + 1/2)g - N_0 f =: F, \tag{1.4}$$

with the the *single-layer operator* $V : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma)$, the *double-layer operator* $K : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$, and the *Newton operator* $N_0 : \widetilde{H}^{-1}(\Omega) \to H^{1/2}(\Gamma)$. If we assume $\operatorname{diam}(\Omega) < 1$ for the two-dimensional case, the single-layer operator has exactly the properties as the integral operator V from above. This especially implies the unique solvability of Symm's integral equation. Therefore, the solution $\phi \in H^{-1/2}(\Gamma)$ of Symm's integral equation determines the weak solution $u \in H^1(\Omega)$ of the Dirichlet problem via the *representation formula*

$$u = \widetilde{V}\phi - \widetilde{K}g + \widetilde{N}_0 f, \tag{1.5}$$

with the *Newton potential* $\widetilde{N}_0 : \widetilde{H}^{-1}(\Omega) \to H^1(\Omega)$, the *single-layer potential* $\widetilde{V} : \widetilde{H}^{-1}(\Omega) \to H^1(\Omega)$ and the *double-layer potential* $\widetilde{K} : H^{1/2}(\Gamma) \to H^1(\Omega)$. For more details and proofs, see for example [Ste08, Chapter 4.1.1, 6.1, 6.2, 6.4 and page 136–143]. Steinbach denotes the double-layer potential with W instead of $\widetilde{K}$.

For the numerical solution of elliptic partial differential equations, one often assumes the considered domain to be a polygon. In [CHB09], Thomas Hughes and collaborators consider three dimensional geometries whose boundaries are so called *NURBS surfaces*, and propose to use transformed *NURBS functions*, corresponding to this geometry, as *finite element method* (FEM) approximation space. Indeed, *computer aided design* (CAD) is mainly based on NURBS, wherefore nearly all geometries of practical interest are of such a form. This new

concept is called *isogeometric analysis* (IGA). Since designers usually construct the surface of the geometry rather than the geometry itself, FEM requires to generate a volume mesh for the use of IGA. This drawback motivates the use of *boundary element method* (BEM). Here, one writes the partial differential equation (if possible) as boundary integral equation and then applies Galerkin method. Hence, for BEM only a description and discretization of the surface (boundary) of the computational domain is needed. In this sense, BEM is the natural method of IGA.

In this work, we consider BEM for the boundary integral equation (1.1). Following the idea of IGA, we use NURBS transformed to the boundary as approximation space. Notice that piecewise polynomials, which are usually considered for BEM, are just special NURBS. We introduce an adaptive algorithm which is based on the local Faermann error estimator developed in [Fae00]. Assuming that γ is an arc length parametrization and taking transformed piecewise polynomials as approximation space, Faermann proved efficiency and reliability of this estimator.

In Section 2 and 3, we generalize Faermann's results to abstract approximation spaces satisfying a certain assumption (Assumption 2.10). Moreover we prove reliability of a weighted residual error estimator in Section 3. In the next Section we define B-splines and NURBS, show that their transformed linear span fulfills Assumption 2.10, and introduce the adaptive algorithm, Algorithm 4.15. In contrast to Faermann, we allow non arc length parametrizations γ for the transformation to the boundary. Then, in Section 5 we develop methods for the implementation of Symm's integral equation (1.4). The stiffness matrix, the right-hand side vector and the Faermann estimator are approximated by tensor-Gauss quadrature. Finally, Section 6 provides some numerical experiments. Here we compare conventional and isogeometric approaches as well as uniform and adaptive mesh refinement. Yet we have not been able to prove convergence of the algorithm. However, the numerical examples even show certain convergence rates. The thesis is concluded with the Appendix, where we prove some assertions for piecewise smooth paths γ, needed throughout the work. Moreover the codes for the implementation of the numerical examples can be found there.

2. Localization of the Sobolev-Slobodeckij Norm

Througout this section, we assume that $\Omega \subset \mathbb{R}^2$ is an open set such that the boundary $\partial\Omega =: \Gamma$ can by parameterized by a fixed regular closed curve $\gamma : [a,b] \to \Gamma$ with $a < b$ as in Definition A.1. With γ we denote the $(b-a)$-periodic extension of γ to $\mathbb{R}$, and with γ'^{ℓ} and γ'^{r} its left resp. right derivative. The fact that γ is a regular closed curve now just means that it is closed, continuous, piecewise continuously differentiable, $\gamma|_{[a,b)}$ is bijective, $\gamma'^{\ell}(t) \neq 0$, $\gamma'^{r}(t) \neq 0$ for all $t \in \mathbb{R}$ and that $\gamma'^{\ell}(t) + c\gamma'^{r}(t) \neq 0$ for all $t \in \mathbb{R}$ and all $c > 0$.

Theorem A.3 implies that $\gamma|_{(a+t,b+t)} : (a+t, b+t) \to \Gamma \setminus \{\gamma(a+t)\}$ is a homeomorphism for all $t \in \mathbb{R}$. We consider a fixed subset $\Gamma_D \subseteq \Gamma$, where either $\Gamma_D = \Gamma$ or $\Gamma_D = \gamma([a_D, b_D])$ with $a_D < b_D \in \mathbb{R}$ and $b_D - a_D < b - a$.

With γ_L we denote the arc length parametrization of γ from Lemma A.7 as well as its L-periodic extension.

Lemma 2.1. *There exists $C_\Gamma > 0$ with*

$$\forall s,t \in \mathbb{R} \text{ with } 0 < |t-s| \le \frac{3}{4}L : \quad C_\Gamma^{-1} \le \frac{|\gamma_L(t) - \gamma_L(s)|}{|t-s|} \le C_\Gamma. \tag{2.1}$$

Proof. For $s,t \in \mathbb{R}$, with the substitution rule proves

$$|\gamma_L(t) - \gamma_L(s)| = \left| \int_s^t \gamma_L'(r)\,\mathrm{d}r \right| = \left| (t-s) \int_0^1 \gamma_L'\big(r(t-s)+s\big)\,\mathrm{d}r \right|.$$

Hence, the set

$$\left\{ \frac{\gamma_L(t) - \gamma_L(s)}{t-s} : s,t \in \mathbb{R}, s \neq t \right\}$$

is bounded above by some positive constant C_1. For each $r \in [0,L]$ there exists $\ell \in \{0,\dots,7\}$ with

$$\left[r, r + \frac{3}{4}L \right] \subseteq \left[\frac{\ell}{8}L, \frac{\ell+7}{8}L \right]. \tag{2.2}$$

Due to Corollary A.6, $\gamma_L|^{-1}_{\left[\frac{\ell}{8}L, \frac{\ell+7}{8}L\right]}$ is Lipschitz continuous, i.e. there exists a positive constant $C(\ell)$ with

$$\forall s,t \in \left[\frac{\ell}{8}L, \frac{\ell+7}{8}L \right] : \quad |t-s| \le C(\ell)|\gamma_L(t) - \gamma_L(s)|.$$

Defining $C_2 := \max_{\ell=0,\dots,7} C(\ell)$, we get with (2.2) for any $r \in [0,L]$

$$\forall s,t \in \left[r, r + \frac{3}{4}L \right] : \quad |t-s| \le C_2|\gamma_L(t) - \gamma_L(s)|.$$

This implies because of the periodicity of γ_L

$$\forall s,t \in \mathbb{R} \text{ with } |t-s| \le \frac{3}{4}L : \quad |t-s| \le C_2|\gamma_L(t) - \gamma_L(s)|.$$

With the definition $C_\Gamma := \max(C_1, C_2)$ we finally get (2.1). □

Definition 2.2. For any measurable subset Γ_0 of Γ, we define with the Lebesgue measure λ on $\mathbb{R}$

$$\mu_\Gamma(\Gamma_0) := \lambda\big(\gamma_L^{-1}(\Gamma_0)\big).$$

We call μ_Γ the *surface measure* on Γ. Obviously it holds $\mu_\Gamma(\Gamma) = L$.

Remark 2.3. For arbitrary parametrizations $\widetilde{\gamma}$ as in the beginning, one can define

$$\mu_\Gamma(\Gamma_0) := \int_{\widetilde{\gamma}^{-1}(\Gamma_0)} |\widetilde{\gamma}'(t)| \,\mathrm{d}\lambda(t). \tag{2.3}$$

This definition is not dependent on the considered path $\widetilde{\gamma}$, wherefore it coincides with the definition from above. To see this, let $\widetilde{a} = \xi_0 < \cdots < \xi_{n_{\widetilde{\gamma}}} = \widetilde{b}$ with $\widetilde{\gamma}|_{[\xi_{j-1},\xi_j]} \in C^1([\xi_{j-1},\xi_j])$ for $j = 1,\dots,n_{\widetilde{\gamma}}$. Then $\bigcup_{j=1}^{n_{\widetilde{\gamma}}} \widetilde{\gamma}((\xi_{j-1},\xi_j))$ is a manifold and the functions $\widetilde{\gamma}|_{(\xi_{j-1},\xi_j)}$ are embeddings (see [Kal11, Definition 15.2.1]). The surface measure on Γ introduced in (2.3), is just the extension of the surface measure on this manifold (see [Kal11, page 73–75]) with $\mu_\Gamma(\widetilde{\gamma}(\{\xi_j\})) = 0$ for $j = 0,\dots,n_{\widetilde{\gamma}}$. It does not depend on $\widetilde{\gamma}$, but only on Γ.

Let u be a measurable function on a measurable subset $\Gamma_0 \subseteq \Gamma$. Then the transformation theorem of measure theory shows

$$\int_{\Gamma_0} u(x) \,\mathrm{d}\mu_\Gamma(x) = \int_{\gamma_L^{-1}(\Gamma_0)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t), \tag{2.4}$$

where one side exists if and only if the other one exists. The following lemma is a generalization of this assertion.

Lemma 2.4. *Let u be a measurable function on a measurable subset $\Gamma_0 \subseteq \Gamma$. Then for all $s \in \mathbb{R}$*

$$\int_{\Gamma_0} u(x) \,\mathrm{d}\mu_\Gamma(x) = \int_{\gamma_L|_{(s,s+L)}^{-1}(\Gamma_0)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t), \tag{2.5}$$

where one side exists if and only if the other one exists.

Proof. We define $s_L := \lceil \frac{s}{L} \rceil L$. Because of the L-periodicity of γ_L, we then have

$$\begin{aligned}
&\int_{\gamma_L|_{(s,s+L)}^{-1}(\Gamma_0)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t) \\
&\quad= \int_{\gamma_L^{-1}(\Gamma_0)\cap(s,s_L)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t) + \int_{\gamma_L^{-1}(\Gamma_0)\cap(s_L,s+L)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t) \\
&\quad= \int_{\gamma_L^{-1}(\Gamma_0)\cap(s+L-s_L,L)} u(\gamma_L(t-L+s_L)) \,\mathrm{d}\lambda(t) + \int_{\gamma_L^{-1}(\Gamma_0)\cap(0,s+L-s_L)} u(\gamma_L(t+s_L)) \,\mathrm{d}\lambda(t) \\
&\quad= \int_{\gamma_L^{-1}(\Gamma_0)} u(\gamma_L(t)) \,\mathrm{d}\lambda(t).
\end{aligned}$$

Here the first term exists if and only if the last one exists. The assertion follows from(2.4). □

Remark 2.5. For arbitrary parametrizations $\widetilde{\gamma}$ as in the beginning of the section, there holds an analogous version of Lemma 2.4. Indeed we have because of Remark 2.3 and [Kal11, page 77] instead of (2.4)

$$\int_{\Gamma_0} u(x) \,\mathrm{d}\mu_\Gamma(x) = \int_{\widetilde{\gamma}^{-1}(\Gamma_0)} u(\widetilde{\gamma}(t))|\widetilde{\gamma}'(t)| \,\mathrm{d}\lambda(t). \tag{2.6}$$

Now one can repeat the proof of Lemma 2.4 to see

$$\int_{\Gamma_0} u(x) \,\mathrm{d}\mu_\Gamma(x) = \int_{\widetilde{\gamma}|_{(s,s+L)}^{-1}(\Gamma_0)} u(\widetilde{\gamma}(t))|\widetilde{\gamma}'(t)| \,\mathrm{d}\lambda(t), \tag{2.7}$$

where one side exists if and only if the other one exists.

Definition 2.6. Let $\Gamma_0 \subseteq \Gamma$ be measurable. For a real square-integrable $u \in L^2(\Gamma_0)$, we define the *Sobolev-Slobodeckij seminorm*[1]

$$|u|^2_{H^{1/2}(\Gamma_0)} := \int_{\Gamma_0}\int_{\Gamma_0} \frac{|u(x)-u(y)|^2}{|x-y|^2}\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) \in [0,\infty]$$

and the *Sobolev-Slobodeckij norm*

$$\|u\|^2_{H^{1/2}(\Gamma_0)} := \|u\|^2_{L^2(\Gamma_0)} + |u|^2_{H^{1/2}(\Gamma_0)} \in [0,\infty].$$

Moreover, we define the *Sobolev space* $H^{1/2}(\Gamma_0) := \big\{u \in L^2(\Gamma_0) : \|u\|_{H^{1/2}(\Gamma_0)} < \infty\big\}$ endowed with

$$\langle u, v\rangle_{H^{1/2}(\Gamma_0)} := \int_{\Gamma_0} u(x)v(x)\,\mathrm{d}\mu_\Gamma(x) + \int_{\Gamma_0}\int_{\Gamma_0} \frac{\big(u(x)-u(y)\big)\big(v(x)-v(y)\big)}{|x-y|^2}\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y).$$

For finite nonempty intervals $I \subseteq \mathbb{R}$, we define analogously $|\cdot|_{H^{1/2}(I)}$, $\|\cdot\|_{H^{1/2}(I)}$ and $H^{1/2}(I)$.

Theorem 2.7. *Let $\Gamma_0 \subseteq \Gamma$ be nonempty and measurable. Then, $H^{1/2}(\Gamma_0)$ is a real Hilbert space. An analogous result holds for finite nonempty intervals I.*

Proof. We define the measure $\nu(\cdot) := \int_{(\cdot)} \frac{1}{|x-y|^2}\,\mathrm{d}(\mu_\Gamma \times \mu_\Gamma)(x,y)$ on $\Gamma_0 \times \Gamma_0$ and $\Phi(u) := \big((x,y) \mapsto u(x) - u(x)\big)$ for $u \in L^2(\Gamma_0)$. Now we note that

$$\forall u \in L^2(\Gamma_0): \quad |u|_{H^{1/2}(\Gamma_0)} = \|\Phi(u)\|_{L^2(\nu)}$$

and

$$H^{1/2}(\Gamma_0) = \big\{u \in L^2(\Gamma_0) : \Phi(u) \in L^2(\nu)\big\}.$$

Therefore $H^{1/2}(\Gamma_0)$ is a vector space. Obviously $\langle\cdot,\cdot\rangle_{H^{1/2}(\Gamma_0)}$ is a scalar product on $H^{1/2}(\Gamma_0)$ which induces the norm $\|\cdot\|_{H^{1/2}(\Gamma_0)}$.

It remains to show completeness. Let $(u_n)_{n\in\mathbb{N}}$ be a Cauchy sequence in $H^{1/2}(\Gamma_0)$. Because of the completeness of $L^2(\Gamma_0)$ and $L^2(\nu)$, the sequence $(u_n)_{n\in\mathbb{N}}$ converges with respect to $\|\cdot\|_{L^2(\Gamma_0)}$ to some limit $u \in L^2(\Gamma_0)$ and the sequence $(\Phi(u_n))_{n\in\mathbb{N}}$ to some limit $U \in L^2(\nu)$ with respect to $\|\cdot\|_{L^2(\nu)}$. Hence there exists a subsequence $(u_{n_k})_{k\in\mathbb{N}}$ converging almost everywhere to u. Then $(\Phi(u_{n_k}))_{k\in\mathbb{N}}$ converges almost everywhere to $\Phi(u)$. Since $(\Phi(u_{n_k}(x)))_{k\in\mathbb{N}}$ also contains a subsequence which converges almost everywhere to $U(x,y)$, this implies $U(x,y) = \Phi(u)$ almost everywhere. Therefore $(u_n)_{n\in\mathbb{N}}$ converges with respect to $\|\cdot\|_{H^{1/2}(\Gamma_0)}$ to $u \in H^{1/2}(\Gamma_0)$. The proof for intervals I is analogous. □

Lemma 2.8. *Let $I \subseteq \mathbb{R}$ be a nonempty interval with length $\lambda(I) \le \frac{3}{4}L$ and u a measurable function on $\gamma_L(I)$. Then, there holds*

$$C_\Gamma^{-2}|u\circ\gamma_L|^2_{H^{1/2}(I)} \le |u|^2_{H^{1/2}(\gamma_L(I))} \le C_\Gamma^2|u\circ\gamma_L|^2_{H^{1/2}(I)}. \tag{2.8}$$

Proof. Due to Lemma 2.4, there holds

$$\int_{\gamma_L(I)}\int_{\gamma_L(I)} \frac{|u(x)-u(y)|^2}{|x-y|^2}\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) = \int_I\int_I \frac{|u(\gamma_L(s))-u(\gamma_L(t))|^2}{|\gamma_L(s)-\gamma_L(t)|^2}\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t).$$

The assertion of the lemma now follows immediately from Lemma (2.1). □

[1] The set $\big\{(x,y) \in \Gamma_0 \times \Gamma_0 : x = y\big\}$ has measure 0.

Definition 2.9. If $\Gamma = \Gamma_D$ let

$$\big\{\check{x}_j \,:\, j = 1, \dots, n\big\} \qquad \text{with} \qquad a < \check{x}_1 < \check{x}_2 < \cdots < \check{x}_n = b$$

be a set of nodes on $(a, b]$. We define the corresponding data:

- The nodes on $\mathbb{R}$ and the nodes on Γ

$$\check{x}_{j+n} = \check{x}_j + (b - a) \quad \text{for} \quad j \in \mathbb{Z} \qquad \text{and} \qquad x_j := \gamma(\check{x}_j) \quad \text{for} \quad j \in \mathbb{Z}.$$

- The elements on $\mathbb{R}$ and the elements on Γ

$$\check{T}_j := [\check{x}_{j-1}, \check{x}_j] \quad \text{for} \quad j \in \mathbb{Z} \qquad \text{and} \qquad T_j := \gamma(\check{T}_j) \quad \text{for} \quad j \in \mathbb{Z}.$$

- The length of the elements

$$h_{\check{T}_j} := \check{x}_j - \check{x}_{j-1} \quad \text{for} \quad j \in \mathbb{Z} \qquad \text{and} \qquad h_{T_j} := \mu_\Gamma(T_j) \quad \text{for} \quad j \in \mathbb{Z}.$$

  Due to Remark 2.5, it holds

$$\inf |\gamma'([a, b])| \cdot h_{\check{T}_j} \le h_{T_j} \le \sup |\gamma'([a, b])| \cdot h_{\check{T}_j}. \tag{2.9}$$

  If γ is even an arc length parametrization, it holds $h_{T_j} = h_{\check{T}_j}$.We also define the maximal length of the elements on Γ

$$h := \max_{j=1,\dots,n} h_{T_j}.$$

- The set of nodes

$$\check{\mathcal{N}}_h := \big\{\check{x}_j \,:\, j = 1, \dots, n\big\} \qquad \text{and} \qquad \mathcal{N}_h := \big\{x_j \,:\, j = 1, \dots, n\big\}.$$

- The mesh on $[a, b]$ and the mesh on Γ

$$\check{\mathcal{T}}_h := \big\{\check{T}_j \,:\, j = 1, \dots, n\big\} \qquad \text{and} \qquad \mathcal{T}_h := \big\{T_j \,:\, j = 1, \dots, n\big\}.$$

- The two nodes of an element T_j

$$x_{T_j,1} := x_{j-1} \qquad \text{and} \qquad x_{T_j,2} := x_j$$

- If $n \ge 2$, the two elements containing the node x_j

$$T_{x_j,1} := T_j \qquad \text{and} \qquad T_{x_j,2} := T_{j+1}$$

- If $n \ge 3$, the two neighbours of an element T_j

$$T_j^- := \gamma(\check{T}_{j-1}) \qquad \text{and} \qquad T_j^+ := \gamma(\check{T}_{j+1}).$$

  Indeed the elements $T_j^\pm$ are just the two elements in $\mathcal{T}_h$ with $T_j^\pm \ne T_j$ and $T_j^\pm \cap T_j \ne \emptyset$.
- The *shape regularity constant* of the mesh on $[a, b]$ and on Γ

$$\kappa(\check{\mathcal{T}}_h) := \max\Big(\big\{h_{\check{T}}/h_{\check{T}'} \,:\, \check{T}, \check{T}' \in \check{\mathcal{T}}_h, \gamma(\check{T}) \cap \gamma(\check{T}') \ne \emptyset\big\}\Big),$$
$$\kappa(\mathcal{T}_h) := \max\Big(\big\{h_T/h_{T'} \,:\, T, T' \in \mathcal{T}_h, T \cap T' \ne \emptyset\big\}\Big).$$

- The *patch function* of first order

$$\omega_h : \mathcal{P}(\Gamma) \to \mathcal{P}(\Gamma) : \Gamma_0 \mapsto \bigcup \{T \in \mathcal{T}_h \,:\, T \cap \Gamma_0 \neq \emptyset\}$$

and the patch function of arbitrary order

$$\omega_h^0 := id_{\mathcal{P}(\Gamma)\to\mathcal{P}(\Gamma)}$$
$$\omega_h^\ell := \omega_h \circ \omega_h^{\ell-1} \quad \text{for } \ell \in \mathbb{N}.$$

For $\ell \in \mathbb{N}_0$ and $x \in \Gamma$, we write $\omega_h(x) := \omega_h(\{x\})$ and $\omega_h^\ell(x) := \omega_h^\ell(\{x\})$.

If $\Gamma_D \neq \Gamma$ let

$$\{\check{x}_j \,:\, j = 0, \dots, n\} \qquad \text{with} \qquad a_D = \check{x}_0 < \check{x}_1 < \cdots < \check{x}_n = b_D$$

be a set of nodes on $[a_D, b_D]$. In this case we do not extend them to nodes on $\mathbb{R}$. The $n+1$ nodes on Γ_D, the n elements, their length, the sets of nodes consisting here of $n+1$ points and the meshes can be defined analogously. For inner nodes x_j with $j \in \{1, \dots, n-1\}$, we define the elements containing x_j analogously. We set $T_{x_0} := T_1$ and $T_{x_n} := T_n$. We also define the neighbours of an element T_j as before, where we set

$$T_0 := \emptyset \qquad \text{and} \qquad T_{n+1} := \emptyset.$$

Although it would be canonical to define $h_{T_1^-}$ and $h_{T_n^+}$ as 0, it will be more convenient for us to define them as $\mu_\Gamma(\Gamma \setminus \Gamma_D)$

$$h_{T_1^-} := h_{T_n^+} := \mu_\Gamma\big(\gamma([\check{x}_n, \check{x}_0 + b - a])\big).$$

If γ is even an arc length parametrization, Lemma 2.4 yields $h_{T_1^-} = h_{T_n^+} = \check{x}_0 + L - \check{x}_n$. Exchanging Γ with Γ_D, the definitions of the shape regularity constant and of the patch function look exactly the same.

For the rest of this section, we consider fixed mesh data with

$$h \leq \frac{\mu_\Gamma(\Gamma)}{4} = \frac{L}{4}. \tag{2.10}$$

This implies $|\mathcal{T}_h| \geq 4$ for $\Gamma = \Gamma_D$. For simplicity, we will also assume $|\mathcal{T}_h| \geq 4$ for $\Gamma \neq \Gamma_D$. Moreover we assume for the rest of the section that γ coincides with the arc length parametrization γ_L. This is no restriction, since the following results are all formulated on the boundary Γ itself and not on the parameter domain $[a, b]$. There holds $a = 0$ and $b = L$.

We suppose that we are given $m \in \mathbb{N}_0$ and a subspace $S^m(\mathcal{T}_h)$ of $L^2(\Gamma_D)$ which satisfies the following assumption.

Assumption 2.10. There holds:

(i) For all $T \in \mathcal{T}_h$ there exists a fixed function[2] $\phi_T \in S^m(\mathcal{T}_h)$ with connected support $\operatorname{supp}(\phi_T)$ and

$$T \subseteq \operatorname{supp}(\phi_T) \subseteq \omega_h^m(T). \tag{2.11}$$

[2]Since $S^m(\mathcal{T}_h)$ consists of equivalence classes, we actually we mean that ϕ_T is one particular representative of an element of $S^m(\mathcal{T}_h)$. Note that it is not necessarily possible to replace ϕ_T here by another element of the corresponding equivalence class.

(ii) There exists a constant $q_{space} \in (0,1]$ such that for all $T \in \mathcal{T}_h$

$$\|1-\phi_T\|^2_{L^2(\mathrm{supp}(\phi_T))} \leq (1-q_{space})\mu_\Gamma(\mathrm{supp}(\phi_T)). \tag{2.12}$$

Our aim is to prove the following localization result whose proof is mainly based on Proposition 2.13 and Proposition 2.16.

Theorem 2.11. *There exists a constant $C_{main} > 0$ such that for all $u \in H^{1/2}(\Gamma_D)$ satisfying*

$$\forall v \in S^m(\mathcal{T}_h): \quad \langle u, v\rangle_{L^2(\Gamma_D)} = 0, \tag{2.13}$$

there holds

$$\|u\|^2_{H^{1/2}(\Gamma_D)} \leq C_{main} \sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))}. \tag{2.14}$$

The constant is explicitly given by

$$C_{main} = 1 + C_{loc}C_{poinc}.$$

Here, C_{loc} is the constant in Proposition 2.13, and $C_{poinc} = C_{poinc}(h, \kappa(\mathcal{T}_h), m, q_{space})$ is the constant in Proposition 2.16.

This theorem, as well as of the following lemmata and propositions needed for its proof, are inspired by similar versions of [Fae00, Section 2]. The theorem corresponds to [Fae00, Theorem 2.2]. She considered $\Gamma_D = \Gamma$ and $\|\cdot\|_{H^s(\Gamma_D)}$ with $s > 0$ instead of $\|\cdot\|_{H^{1/2}(\Gamma_D)}$. Instead of the space $S^m(\mathcal{T}_h)$, she used transformed piecewise polynomials on $[0, L]$, see [Fae00, page 206]. It is also notable that Faermann did not prove any analogous version of Lemma 2.1, but just assumed that (2.1) holds, see [Fae00, Assumption 2.1].

The converse inequality even holds without orthogonality.

Theorem 2.12. *Let $u \in H^{1/2}(\Gamma_D)$. Then,*

$$\sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} \leq 2\|u\|^2_{H^{1/2}(\Gamma_D)}. \tag{2.15}$$

Proof of Theorem 2.12 for $\Gamma_D = \Gamma$. We define for $y \neq z \in \Gamma$

$$U(y,z) := \frac{|u(y)-u(z)|^2}{|y-z|^2}.$$

There holds

$$\begin{aligned}
\sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} &= \sum_{T\in\mathcal{T}_h} |u|^2_{H^{1/2}(T\cup T^+)} \\
&= \sum_{T\in\mathcal{T}_h} \left(\int_T \int_T U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) + \int_T \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) \right) \\
&\quad + \sum_{T\in\mathcal{T}_h} \left(\int_{T^+} \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) + \int_T \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) \right) \\
&\leq 2 \sum_{T\in\mathcal{T}_h} \int_T \int_\Gamma U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z).
\end{aligned}$$

The last term is just $2\|u\|^2_{H^{1/2}(\Gamma)}$, which concludes the proof. □

Proof of Theorem 2.12 for $\Gamma_D \neq \Gamma$. The proof reads nearly the same

$$\begin{aligned}
\sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} &= \sum_{T\in\mathcal{T}_h} |u|^2_{H^{1/2}(T\cup T^+)} + |u|^2_{H^{1/2}(T_1)} \\
&= \sum_{T\in\mathcal{T}_h} \left(\int_T \int_T U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) + \int_T \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) \right) \\
&\quad + \sum_{T\in\mathcal{T}_h} \left(\int_{T^+} \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) + \int_T \int_{T^+} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) \right) \\
&\quad + \int_{T_1} \int_{T_1} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z) \\
&\leq 2 \sum_{T\in\mathcal{T}_h} \int_T \int_{\Gamma_D} U(y,z)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(z).
\end{aligned}$$

The last term is just $2\|u\|^2_{H^{1/2}(\Gamma_D)}$, which concludes the proof. □

Proposition 2.13. *For* $u \in H^{1/2}(\Gamma_D)$*, we have*

$$\|u\|^2_{H^{1/2}(\Gamma_D)} \leq \sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} + C_{loc} \sum_{T\in\mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1}) \|u\|^2_{L^2(T)}, \tag{2.16}$$

with the constant

$$C_{loc} = \begin{cases} \frac{\mu_\Gamma(\Gamma)}{8} + 4C_\Gamma^2 & \text{if } \Gamma_D = \Gamma. \\ \frac{\mu_\Gamma(\Gamma)}{4} + 4C_\Gamma^2 & \text{if } \Gamma_D \neq \Gamma. \end{cases}$$

Proof of Proposition 2.13 for $\Gamma_D = \Gamma$. To simplify the notation, we define for $x, y \in \Gamma$

$$U(x,y) := \frac{|u(x) - u(y)|^2}{|x-y|^2}.$$

First, we estimate $|u|_{H^{1/2}(\Gamma_D)}$. Up to sets of measure zero, we have $\omega_h(T) = T^- \,\dot\cup\, T \,\dot\cup\, T^+$, which shows

$$\begin{aligned}
|u|^2_{H^{1/2}(\Gamma_D)} &= \int_{\Gamma_D} \int_{\Gamma_D} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) = \sum_{T\in\mathcal{T}_h} \int_T \int_{\Gamma_D} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) \\
&= \sum_{T\in\mathcal{T}_h} \Bigg(\int_T \int_T U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) + \int_T \int_{T^-} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) \\
&\quad + \int_T \int_{T^+} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) + \underbrace{\int_T \int_{\Gamma_D\setminus\omega_h(T)} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y)}_{=:E_T} \Bigg) \qquad (2.17) \\
&\leq \sum_{T\in\mathcal{T}_h} \left(\frac{1}{2}|u|^2_{H^{1/2}(T\cup T^-)} + \frac{1}{2}|u|^2_{H^{1/2}(T\cup T^+)} + E_T \right) \\
&= \sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} + \sum_{T\in\mathcal{T}_h} E_T.
\end{aligned}$$

We have

$$E_T = \int_T \int_{\Gamma_D \setminus \omega_h(T)} U(x,y)\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y) \le E_{T,1} + E_{T,2} \tag{2.18}$$

with

$$E_{T,1} := 2\int_T |u(y)|^2 \int_{\Gamma_D \setminus \omega_h(T)} |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(x)\,\mathrm{d}\mu_\Gamma(y)$$

and

$$E_{T,2} := 2\int_{\Gamma_D \setminus \omega_h(T)} |u(x)|^2 \int_T |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(x).$$

To estimate $E_{T,1}$, let $j \in \{1,\dots,n\}$ with $T = T_j$, $y \in T_j$ and $\check{y} \in \check{T}_j$ with $\gamma(\check{y}) = y$. For any $t \in M_j$ with

$$M_j := \gamma^{-1}\big(\Gamma_D \setminus \omega_h(T)\big) \cap \left(\check{x}_j - \frac{L}{2}, \check{x}_j + \frac{L}{2}\right) = \left(\check{x}_j - \frac{L}{2}, \check{x}_{j-2}\right) \cup \left(\check{x}_{j+1}, \check{x}_j + \frac{L}{2}\right), \tag{2.19}$$

it holds because of (2.10)

$$|t - \check{y}| \le |t - \check{x}_j| + |\check{y} - \check{x}_j| \le \frac{L}{2} + h_{\check{T}_j} \le \frac{3}{4}L. \tag{2.20}$$

Because of Lemma (2.1) and Lemma 2.4, we therefore get that

$$\begin{aligned}
\int_{\Gamma_D \setminus \omega_h(T)} |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(x) &= \int_{M_j} |\gamma(t) - \gamma(\check{y})|^{-2}\,\mathrm{d}\lambda(t) \\
&\le C_\Gamma^2 \int_{M_j} |t - \check{y}|^{-2}\,\mathrm{d}\lambda(t) \\
&\le C_\Gamma^2 \left(\int_{(-\infty, \check{x}_{j-2}]} (\check{y} - t)^{-2}\,\mathrm{d}\lambda(t) + \int_{[\check{x}_{j+1}, \infty)} (t - \check{y})^{-2}\,\mathrm{d}\lambda(t) \right) \\
&= C_\Gamma^2 \big((\check{y} - \check{x}_{j-2})^{-1} + (\check{x}_{j+1} - \check{y})^{-1} \big).
\end{aligned}$$

Since $\check{y} \in \check{T}_j$, we have for $y \in T$

$$\int_{\Gamma_D \setminus \omega_h(T)} |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(x) \le C_\Gamma^2 (h_{T_{j-1}}^{-1} + h_{T_{j+1}}^{-1}) = C_\Gamma^2 (h_{T^-}^{-1} + h_{T^+}^{-1}). \tag{2.21}$$

This implies

$$\sum_{T \in \mathcal{T}_h} E_{T,1} \le 2C_\Gamma^2 \sum_{T \in \mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1}) \|u\|_{L^2(\Gamma_D)}^2. \tag{2.22}$$

For $E_{T,2}$, we have

$$\begin{aligned}\sum_{T\in\mathcal{T}_h} E_{T,2} &= 2\sum_{T\in\mathcal{T}_h}\int_{\Gamma_D\setminus\omega_h(T)} |u(x)|^2 \int_T |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(x)\\ &= 2\sum_{T\in\mathcal{T}_h}\int_{\Gamma_D} (1-\chi_{\omega_h(T)}(x))|u(x)|^2 \int_T |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(x)\\ &= 2\int_{\Gamma_D} |u(x)|^2 \Bigg(\underbrace{\sum_{T\in\mathcal{T}_h}(1-\chi_{\omega_h(T)}(x))\int_T |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y)}_{=:g(x)}\Bigg)\,\mathrm{d}\mu_\Gamma(x)\\ &= 2\sum_{T'\in\mathcal{T}_h}\int_{T'} |u(x)|^2 g(x)\,\mathrm{d}\mu_\Gamma(x).\end{aligned}$$

Let $T, T' \in \mathcal{T}_h$ and $x \in T' \setminus \mathcal{N}_h$. Then, we have the equivalences

$$\begin{aligned}\chi_{\omega_h(T)}(x) = 1 \quad &\iff \quad x \in \omega_h(T)\\ &\iff \quad T \in \{T'^-, T', T'^+\}.\end{aligned}$$

Thus, (2.21) shows for almost every $x \in T'$ that

$$\begin{aligned}g(x) &= \sum_{\substack{T\in\mathcal{T}_h\\ T\notin\{T'^-,T',T'^+\}}} \int_T |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y)\\ &= \int_{\Gamma_D\setminus\omega_h(T')} |x-y|^{-2}\,\mathrm{d}\mu_\Gamma(y) \le C_\Gamma^2(h_{T'^-}^{-1}+h_{T'^+}^{-1}).\end{aligned}$$

Therefore, we get

$$\begin{aligned}\sum_{T\in\mathcal{T}_h} E_{T,2} &\le 2C_\Gamma^2 \sum_{T'\in\mathcal{T}_h}(h_{T'^-}^{-1}+h_{T'^+}^{-1})\|u\|_{L^2(T')}^2\\ &= 2C_\Gamma^2\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|_{L^2(T)}^2.\end{aligned} \tag{2.23}$$

Combining (2.17), (2.18)), (2.22) and (2.23) shows

$$|u|_{H^{1/2}(\Gamma_D)}^2 \le \sum_{x\in\mathcal{N}_h} |u|_{H^{1/2}(\omega_h(x))}^2 + 4C_\Gamma^2\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|_{L^2(T)}^2. \tag{2.24}$$

It remains to estimate $\|u\|_{L^2(\Gamma_D)}^2$. From (2.10), we get

$$\frac{8}{\mu_\Gamma(\Gamma)} \le (h_{T^-}^{-1}+h_{T^+}^{-1}). \tag{2.25}$$

Therefore

$$\|u\|_{L^2(\Gamma_D)}^2 = \sum_{T\in\mathcal{T}_h}\|u\|_{L^2(T)}^2 \le \frac{\mu_\Gamma(\Gamma)}{8}\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|_{L^2(T)}^2. \tag{2.26}$$

Adding (2.24) and (2.26) gives the assertion of the proposition. □

Proof of Proposition 2.13 for $\Gamma_D \neq \Gamma$. The proof follows essentially as for $\Gamma_D = \Gamma$. We therefore only consider the differences. In (2.17) the last "=" has to be replaced by "$\leq$". With $\check{x}_{-1} := \check{x}_n - L$ and $\check{x}_{n+1} := \check{x}_0 + L$, the last "=" of (2.19) has to be replaced by "$\subseteq$". To see this, we define $\omega_T := \omega_h(T) \cup \gamma([\check{x}_n, \check{x}_0 + L])$ for $T = T_1$ or $T = T_n$ and $\omega_T := \omega_h(T)$ else. Then we get (up to two points if $T = T_1$ or $T = T_n$)

$$\begin{aligned} M_j &:= \gamma^{-1}\big(\Gamma_D \setminus \omega_h(T)\big) \cap \left(\check{x}_j - \frac{L}{2}, \check{x}_j + \frac{L}{2}\right) \\ &\subseteq \gamma^{-1}\big(\Gamma \setminus \omega_T\big) \cap \left(\check{x}_j - \frac{L}{2}, \check{x}_j + \frac{L}{2}\right) \\ &= \left(\check{x}_j - \frac{L}{2}, \check{x}_{j-2}\right) \cup \left(\check{x}_{j+1}, \check{x}_j + \frac{L}{2}\right). \end{aligned}$$

Proceeding as before, we see again that for any $y \in T$

$$\int_{\Gamma_D \setminus \omega_h(T)} |x - y|^{-2} \,\mathrm{d}\mu_\Gamma(x) \leq C_\Gamma^2 (h_{T^-}^{-1} + h_{T^+}^{-1}).$$

Finally in (2.25) – (2.26), one must replace 8 by 4, since we do not necessarily have $\frac{4}{\mu_\Gamma(\Gamma)} \leq h_{T_1^-}^{-1}$ or $\frac{4}{\mu_\Gamma(\Gamma)} \leq h_{T_n^+}^{-1}$. □

To prove Theorem 2.11 it remains to estimate $\sum_{T \in \mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1}) \|u\|_{L^2(T)}^2$. Whereas we didn't use the orthogonality (2.13) for Proposition 2.13, it will be essential for this estimate given in Proposition 2.16.

Lemma 2.14. *Let $I \subseteq \mathbb{R}$ be a finite interval with length $\lambda(I) > 0$. Then, there holds for all $u \in L^2(I)$ the Poincaré-type inequality*

$$\|u\|_{L^2(I)}^2 \leq \frac{1}{2}\lambda(I)|u|_{H^{1/2}(I)}^2 + \frac{1}{\lambda(I)} \left| \int_I u(t) \,\mathrm{d}\lambda(t) \right|^2. \tag{2.27}$$

Proof. Assume $|u|_{H^{1/2}(I)} < \infty$, otherwise the assertion holds trivially. The definition $E := \int_I u(t) \,\mathrm{d}\lambda(t)$ leads to

$$\begin{aligned} &\int_I \int_I |u(s) - u(t)|^2 \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) \\ &\quad = \int_I \int_I u(s)^2 \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) + \int_I \int_I u(t)^2 \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) - 2 \int_I \int_I u(s)u(t) \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) \\ &\quad = 2\lambda(I) \int_I u(s)^2 \,\mathrm{d}\lambda(s) - 2E^2 = 2\lambda(I)\|u\|_{L^2(I)}^2 - 2E^2. \end{aligned}$$

This implies

$$\begin{aligned} 2\lambda(I)\|u\|_{L^2(I)}^2 - 2E^2 &= \int_I \int_I \frac{|u(s) - u(t)|^2}{|s - t|^2} |s - t|^2 \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) \\ &\leq \lambda(I)^2 \int_I \int_I \frac{|u(s) - u(t)|^2}{|s - t|^2} \,\mathrm{d}\lambda(s) \,\mathrm{d}\lambda(t) = \lambda(I)^2 |u|_{H^{1/2}(I)}^2, \end{aligned}$$

and concludes the proof. □

Lemma 2.15. *For all $u \in L^2(\Gamma_D)$ satisfying $\langle u, v\rangle_{L^2(\Gamma_D)} = 0$ for all $v \in S^m(\mathcal{T}_h)$, there holds for all $T \in \mathcal{T}_h$*

$$\begin{cases} \|u\|^2_{L^2(T)} \le C_{patch} h_T |u|^2_{H^{1/2}(T)} & \text{if } m = 0, \\ \|u\|^2_{L^2(\operatorname{supp}(\phi_T))} \le C_{patch} \mu_\Gamma(\operatorname{supp}(\phi_T)) \sum_{x \in \omega_h^{m-1}(T) \cap \mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} & \text{if } m > 0, \end{cases} \tag{2.28}$$

with the constant

$$C_{patch} = \begin{cases} \frac{C_\Gamma^2}{2q_{space}} & \text{if } m = 0. \\ \frac{C_\Gamma^2}{2q_{space}} (1 + 2\kappa(\mathcal{T}_h))^{2m-1} & \text{if } m > 0. \end{cases}$$

Here, the constant q_{space} is the one from Assumption 2.10.

Proof of Lemma 2.15 for $\Gamma_D = \Gamma$. Since $\operatorname{supp}(\phi_T)$ is connected, there is an interval I of length $\lambda(I) \le L$ with $\gamma(I) = \operatorname{supp}(\phi_T)$. We use Lemma 2.14 and get

$$\|u \circ \gamma\|^2_{L^2(I)} \le \frac{1}{2} \lambda(I) |u \circ \gamma|^2_{H^{1/2}(I)} + \frac{1}{\lambda(I)} \underbrace{\left| \int_I u \circ \gamma(t) \, \mathrm{d}\lambda(t) \right|^2}_{=:E^2}.$$

We use Lemma 2.4, the orthogonality (2.13) and Assumption 2.10, (ii), to get

$$\begin{aligned} E^2 &= \left| \int_{\operatorname{supp}(\phi_T)} u(y)(1 - \phi_T(y)) \, \mathrm{d}\lambda(y) \right|^2 \\ &= \left| \int_I \big(u \circ \gamma(t)\big)\big(1 - \phi_T \circ \gamma(t)\big) \, \mathrm{d}\lambda(t) \right|^2 \\ &\le \|1 - (\phi_T \circ \gamma)\|^2_{L^2(I)} \|u \circ \gamma\|^2_{L^2(I)} \le (1 - q_{space}) \lambda(I) \|u \circ \gamma\|^2_{L^2(I)}. \end{aligned}$$

Using the previous inequality, we therefore get

$$\|u \circ \gamma\|^2_{L^2(I)} \le \frac{1}{2} \lambda(I) |u \circ \gamma|^2_{H^{1/2}(I)} + (1 - q_{space}) \|u \circ \gamma\|^2_{L^2(I)},$$

which implies

$$\|u\|^2_{L^2(\operatorname{supp}(\phi_T))} \le \frac{\mu_\Gamma(\operatorname{supp}(\phi_T))}{2q_{space}} |u \circ \gamma|^2_{H^{1/2}(I)}. \tag{2.29}$$

If $m = 0$ we are done due to (2.10) and Lemma 2.8. To estimate $|u \circ \gamma|^2_{H^{1/2}(I)}$ for $p > 0$, we use induction on ℓ to prove the following assertion:

$$\forall \ell \in \mathbb{N} : \forall j \in \mathbb{Z} : \quad |u \circ \gamma|^2_{H^{1/2}([\check{x}_{j-1}, \check{x}_{j+\ell}])} \le (1 + 2\kappa(\mathcal{T}_h))^{\ell - 1} \sum_{q=j}^{j+\ell-1} |u \circ \gamma|^2_{H^{1/2}(\check{T}_q \cup \check{T}_{q+1})}. \tag{2.30}$$

For $\ell = 1$ it is obvious. The induction hypothesis for $\ell - 1 \ge 1$ is

$$\forall j \in \mathbb{Z} : \quad |u \circ \gamma|^2_{H^{1/2}([\check{x}_{j-1}, \check{x}_{j+\ell-1}])} \le (1 + 2\kappa(\mathcal{T}_h))^{\ell - 2} \sum_{q=j}^{j+\ell-2} |u \circ \gamma|^2_{H^{1/2}(\check{T}_q \cup \check{T}_{q+1})}. \tag{2.31}$$

For $r, s \in \mathbb{R}$ we introduce

$$\check{U}(r, s) := \frac{|u(\gamma(r)) - u(\gamma(s))|^2}{|r - s|^2}.$$

For $j \in \mathbb{Z}$, we have to estimate

$$\begin{aligned}
|u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell}])} &= \int_{[\check{x}_{j-1},\check{x}_{j+\ell-1}]}\int_{[\check{x}_{j-1},\check{x}_{j+\ell-1}]} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s)\\
&\quad + \int_{\check{T}_{j+\ell}}\int_{\check{T}_{j+\ell}} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s) + 2\int_{\check{T}_{j+\ell}}\int_{[\check{x}_{j-1},\check{x}_{j+\ell-1}]} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s)\\
&= |u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell-1}])} + |u\circ\gamma|^2_{H^{1/2}(\check{T}_{j+\ell})} + 2\int_{\check{T}_{j+\ell}}\int_{\check{T}_{j+\ell-1}} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s) \qquad (2.32)\\
&\quad + 2\underbrace{\int_{\check{T}_{j+\ell}}\int_{[\check{x}_{j-1},\check{x}_{j+\ell-1}]} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s)}_{:=E}\\
&\le |u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell-1}])} + |u\circ\gamma|^2_{H^{1/2}(\check{T}_{j+\ell-1}\cup\check{T}_{j+\ell})} + 2E.
\end{aligned}$$

For $r<t<s\in\mathbb{R}$, we have

$$\check{U}(r,s) \le 2\frac{|u(\gamma(r))-u(\gamma(t))|^2}{|r-s|^2} + 2\frac{|u(\gamma(t))-u(\gamma(s))|^2}{|r-s|^2} \le 2\check{U}(r,t) + 2\check{U}(t,s),$$

hence with $h_q := h_{\check{T}_q}$ for $q\in\mathbb{Z}$

$$\begin{aligned}
E &= \frac{1}{h_{j+\ell-1}}\int_{\check{T}_{j+\ell-1}}\int_{\check{T}_{j+\ell}}\int_{[\check{x}_{j-1},\check{x}_{j+\ell-2}]} \check{U}(r,s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\
&\le \frac{2}{h_{j+\ell-1}}\int_{\check{T}_{j+\ell-1}}\int_{[\check{x}_{j-1},\check{x}_{j+\ell-2}]} \check{U}(r,t)\int_{\check{T}_{j+\ell}} 1\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(t)\\
&\quad + \frac{2}{h_{j+\ell-1}}\int_{\check{T}_{j+\ell-1}}\int_{\check{T}_{j+\ell}} \check{U}(t,s)\int_{[\check{x}_{j-1},\check{x}_{j+\ell-2}]} 1\,\mathrm{d}\lambda(r)\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\
&\le \frac{h_{j+\ell}}{h_{j+\ell-1}}|u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell-1}])} + \frac{\check{x}_{j+\ell-2}-\check{x}_{j-1}}{h_{j+\ell-1}}|u\circ\gamma|^2_{H^{1/2}(\check{T}_{j+\ell-1}\cup\check{T}_{j+\ell})}.
\end{aligned}$$

There holds

$$\frac{\check{x}_{j+\ell-2}-\check{x}_{j-1}}{h_{j+\ell-1}} = \sum_{q=j}^{j+\ell-2}\frac{h_q}{h_{j+\ell-1}} \le \sum_{q=j}^{j+\ell-2}\kappa(\mathcal{T}_h)^{j+\ell-1-q} = \sum_{q=1}^{\ell-1}\kappa(\mathcal{T}_h)^q.$$

This implies

$$E \le \kappa(\mathcal{T}_h)|u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell-1}])} + |u\circ\gamma|^2_{H^{1/2}(\check{T}_{j+\ell-1}\cup\check{T}_{j+\ell})}\sum_{q=1}^{\ell-1}\kappa(\mathcal{T}_h)^q.$$

Inserting this into our estimate (2.32) and using

$$1 + 2\sum_{q=1}^{\ell-1}\kappa(\mathcal{T}_h)^q \le (1+2\kappa(\mathcal{T}_h))^{\ell-1}$$

as well as the induction hypothesis (2.31), we obtain

$$\begin{aligned}
&|u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell}])}\\
&\quad\le (1+2\kappa(\mathcal{T}_h))|u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell-1}])} + (1+2\kappa(\mathcal{T}_h))^{\ell-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_{j+\ell-1}\cup\check{T}_{j+\ell})}\\
&\quad\le (1+2\kappa(\mathcal{T}_h))^{\ell-1}\sum_{q=j}^{j+\ell-2}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})} + (1+2\kappa(\mathcal{T}_h))^{\ell-1}|u\circ\gamma|_{H^{1/2}(\check{T}_{j+\ell-1}\cup\check{T}_{j+\ell})}\\
&\quad= (1+2\kappa(\mathcal{T}_h))^{\ell-1}\sum_{q=j}^{j+\ell-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}.
\end{aligned}$$

This concludes the induction step and thus proves (2.30). There is a $j\in\mathbb{Z}$ with

$$\gamma([\check{x}_{j-1},\check{x}_{\min(j+2m,j-1+n)}]) = \omega_h^m(T).$$

Because of Assumption 2.10, (i), one can choose I such that $I\subseteq[\check{x}_{j-1},\check{x}_{\min(j+2m,j-1+n)}]$. We use (2.29) and (2.30) for $\ell=\min(2m,n-1)$ to see

$$\begin{aligned}
\|u\|^2_{L^2(\operatorname{supp}(\phi_T))} &\le \frac{\mu_\Gamma(\operatorname{supp}(\phi_T))}{2q_{space}}(1+2\kappa(\mathcal{T}_h))^{\min(2m,-1+n)-1}\sum_{q=j}^{\min(j+2m,j-1+n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}\\
&\le \frac{\mu_\Gamma(\operatorname{supp}(\phi_T))}{2q_{space}}(1+2\kappa(\mathcal{T}_h))^{2m-1}\sum_{q=j}^{\min(j+2m,j-1+n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}.
\end{aligned}$$

We use Lemma 2.8 and

$$\big\{x_q\,:\,q=j,\dots,\min(j+2m,j-1+n)-1\big\}\subseteq\omega_h^{m-1}(T)\cap\mathcal{N}_h,$$

to get

$$\begin{aligned}
\sum_{q=j}^{\min(j+2m,j-1+n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})} &\le C_\Gamma^2\sum_{q=j}^{\min(j+2m,j-1+n)-1}|u\circ\gamma|^2_{H^{1/2}(\omega_h(x_q))}\\
&\le C_\Gamma^2\sum_{x\in\omega_h^{m-1}(T)\cap\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))},
\end{aligned}$$

which concludes the proof. □

Proof of Lemma 2.15 for $\Gamma_D\neq\Gamma$. As before we see (2.29), where we choose $I\subseteq[a_D,b_D]$. For $m=0$ we are again done. There holds an analogous version of (2.30)

$$\begin{aligned}
&\forall\ell\in\mathbb{N}:\forall j\in\{1,\dots,n\}:\\
&\qquad j+\ell\le n \quad\Longrightarrow\quad |u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{j+\ell}])}\le(1+2\kappa(\mathcal{T}_h))^{\ell-1}\sum_{q=j}^{j+\ell-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}.
\end{aligned}\tag{2.33}$$

This can be proven just as before. Let $j\in\{1,\dots,n\}$ with $\gamma([\check{x}_{j-1},\check{x}_{\min(j+2m,n)}]) = \omega_h^m(T)$ and $I\subseteq[\check{x}_{j-1},\check{x}_{\min(j+2m,n)}]$. Note that $\ell:=\min(j+2m,n)-j\in\mathbb{N}$ and $\ell+j\le n$. Hence,

we may apply (2.33) to see

$$\begin{aligned}|u\circ\gamma|^2_{H^{1/2}(I)} &\le |u\circ\gamma|^2_{H^{1/2}([\check{x}_{j-1},\check{x}_{\min(j+2m,n)}])}\\ &\le (1+2\kappa(\mathcal{T}_h))^{\ell-1}\sum_{q=j}^{\min(j+2m,n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}\\ &\le (1+2\kappa(\mathcal{T}_h))^{2m-1}\sum_{q=j}^{\min(j+2m,n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}.\end{aligned}$$

Using Lemma 2.8, we see

$$\sum_{q=j}^{\min(j+2m,n)-1}|u\circ\gamma|^2_{H^{1/2}(\check{T}_q\cup\check{T}_{q+1})}\le C_\Gamma^2\sum_{x\in\omega_h^{m-1}(T)\cap\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))}$$

and conclude the proof. □

Proposition 2.16. *For all $u\in H^{1/2}(\Gamma_D)$ satisfying $\langle u,v\rangle_{L^2(\Gamma_D)}=0$ for all $v\in S^m(\mathcal{T}_h)$, there holds*

$$\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|^2_{L^2(T)}\le C_{poinc}\sum_{x\in\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))},\tag{2.34}$$

with the constant

$$C_{poinc}=\begin{cases}C_{patch}\kappa(\mathcal{T}_h) & \text{if } \Gamma_D=\Gamma \text{ and } m=0.\\ 4C_{patch}m\big(1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\big) & \text{if } \Gamma_D=\Gamma \text{ and } m>0.\\ C_{patch}\max\big(\frac{h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)},\kappa(\mathcal{T}_h)\big) & \text{if } \Gamma_D\ne\Gamma \text{ and } m=0.\\ 4C_{patch}m\max\big(\frac{(m+1)h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)},1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\big) & \text{if } \Gamma_D\ne\Gamma \text{ and } m>0.\end{cases}$$

Here, C_{patch} is the constant of Lemma 2.15.

Proof of Proposition 2.16 for $\Gamma_D=\Gamma$. For $m=0$, Lemma 2.15 implies

$$\begin{aligned}\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|^2_{L^2(T)} &\le \sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})C_{patch}h_T|u|^2_{H^{1/2}(T)}\\ &\le \sum_{T\in\mathcal{T}_h}C_{patch}\kappa(\mathcal{T}_h)\Big(|u|^2_{H^{1/2}(T)}+|u|^2_{H^{1/2}(T^+)}\Big)\\ &\le C_{patch}\kappa(\mathcal{T}_h)\sum_{x\in\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))}.\end{aligned}$$

For $m>0$, Assumption 2.10, (i), and Lemma 2.15 give

$$\|u\|^2_{L^2(T)}\le\|u\|^2_{L^2(\operatorname{supp}(\phi_T))}\le C_{patch}\mu_\Gamma(\omega_h^m(T))\sum_{x\in\omega_h^{m-1}(T)\cap\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))}.\tag{2.35}$$

Let $j\in\{1,\dots,n\}$ with $T=T_j$. Then, we have with $h_\ell:=h_{\check{T}_\ell}$ for $\ell\in\mathbb{Z}$

$$\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^-}}\le\frac{\check{x}_{j+m}-\check{x}_{j-1-m}}{h_{j-1}}=\sum_{\ell=-m+1}^{m+1}\frac{h_{j-1+\ell}}{h_{j-1}}\le\sum_{\ell=-m+1}^{m+1}\kappa(\mathcal{T}_h)^{|\ell|}\le 1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\tag{2.36}$$

and

$$\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^+}} \le \frac{\check{x}_{j+m} - \check{x}_{j-1-m}}{h_{j+1}} = \sum_{\ell=-m-1}^{m-1} \frac{h_{j+1+\ell}}{h_{j+1}} \le \sum_{\ell=-m-1}^{m-1} \kappa(\mathcal{T}_h)^{|\ell|} \le 1 + 2\sum_{\ell=1}^{m+1} \kappa(\mathcal{T}_h)^\ell. \quad (2.37)$$

Combining (2.35), (2.36) and (2.37), we obtain with $C := C_{patch}\big(1 + 2\sum_{\ell=1}^{m+1} \kappa(\mathcal{T}_h)^\ell\big)$

$$\begin{aligned}
\sum_{T\in\mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1})\|u\|_{L^2(T)}^2 &\le C \sum_{T\in\mathcal{T}_h} \left(\sum_{x\in\omega_h^{m-1}(T)\cap\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} + \sum_{x\in\omega_h^{m-1}(T)\cap\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} \right) \\
&= 2C \sum_{T\in\mathcal{T}_h} \sum_{\substack{x\in\mathcal{N}_h \\ x\in\omega_h^{m-1}(T)}} |u|^2_{H^{1/2}(\omega_h(x))} \\
&= 2C \sum_{x\in\mathcal{N}_h} \sum_{\substack{T\in\mathcal{T}_h \\ x\in\omega_h^{m-1}(T)}} |u|^2_{H^{1/2}(\omega_h(x))} \\
&= 4Cm \sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))}. \qquad (2.38)
\end{aligned}$$

This is just the assertion of the proposition. □

Proof of Proposition 2.16 for $\Gamma_D \neq \Gamma$. For $m = 0$, Lemma 2.15 implies

$$\sum_{T\in\mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1})\|u\|_{L^2(T)}^2 \le \sum_{T\in\mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1}) C_{patch} h_T |u|^2_{H^{1/2}(T)}.$$

Now, we split the sum into two sums, shift the indices in the second one, and see

$$\begin{aligned}
\sum_{T\in\mathcal{T}_h} (h_{T^-}^{-1} + h_{T^+}^{-1})\|u\|_{L^2(T)}^2 &\le C_{patch} \max\left(\frac{h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}, \kappa(\mathcal{T}_h) \right) \\
&\qquad \left(\sum_{T\in\mathcal{T}_h} \left(|u|^2_{H^{1/2}(T)} \right) + \left(\sum_{T\in\mathcal{T}_h\setminus\{T_n\}} \left(|u|^2_{H^{1/2}(T^+)} \right) + |u|^2_{H^{1/2}(T_1)} \right) \right) \\
&\le C_{patch} \max\left(\frac{h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}, \kappa(\mathcal{T}_h) \right) \sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))}.
\end{aligned}$$

For $m > 0$, we have again (2.35). Let $j \in \{1, \dots, n\}$ with $T = T_j$. If $j = 1$, we have

$$\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^-}} = \frac{\check{x}_{\min(1+m,n)} - \check{x}_0}{\mu_\Gamma(\Gamma\setminus\Gamma_D)} \le \frac{(m+1)h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}.$$

If $j \neq 1$, we have with $h_\ell := h_{\check{T}_\ell}$ for $\ell = 1, \dots, n$

$$\begin{aligned}
\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^-}} = \frac{\check{x}_{\min(j+m,n)} - \check{x}_{\max(j-1-m,0)}}{h_{j-1}} &\le \sum_{\ell=\max(-m+1,-j-2)}^{\min(m+1,n-j+1)} \frac{h_{j-1+\ell}}{h_{j-1}} \\
&\le \sum_{\ell=\max(-m+1,-j-2)}^{\min(m+1,n-j+1)} \kappa(\mathcal{T}_h)^{|\ell|} \le 1 + 2\sum_{\ell=1}^{m+1} \kappa(\mathcal{T}_h)^\ell.
\end{aligned}$$

This implies

$$\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^-}} \leq \max\left(\frac{(m+1)h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}, 1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\right).$$

Analogously one shows

$$\frac{\mu_\Gamma(\omega_h^m(T))}{h_{T^+}} \leq \max\left(\frac{(m+1)h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}, 1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\right).$$

With

$$C := C_{patch}\max\left(\frac{(m+1)h}{\mu_\Gamma(\Gamma\setminus\Gamma_D)}, 1+2\sum_{\ell=1}^{m+1}\kappa(\mathcal{T}_h)^\ell\right),$$

we get again (2.38), if we replace the last "=" with a "$\leq$". □

Now we can finally prove the main result of this section.

Proof of Theorem 2.11. The theorem follows directly from Proposition 2.13 and Proposition 2.16

$$\begin{aligned}\|u\|_{H^{1/2}(\Gamma)} &\leq \sum_{x\in\mathcal{N}_h}|u|^2_{H^{1/2}(\omega_h(x))} + C_{loc}\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|u\|^2_{L^2(T)}\\ &\leq (1+C_{loc}C_{poinc})\sum_{x\in\mathcal{N}_h}|u|_{H^{1/2}(\omega_h(x))}.\end{aligned}$$

This concludes the proof. □

3. Local Error Indicators

In this section let Ω be an open set and Γ_D a subset of the boundary Γ, which is parametrized by $\gamma : [a, b] \to \Gamma$, as in Section 2. We introduce $\widetilde{H}^{-1/2}(\Gamma_D) := H^{1/2}(\Gamma_D)'$ as the topological dual space of $H^{1/2}(\Gamma_D)$ endowed with the dual norm $\|\cdot\|_{\widetilde{H}^{-1/2}(\Gamma_D)}$. For $\Gamma_D = \Gamma$ we also write $H^{-1/2}(\Gamma_D) := \widetilde{H}^{-1/2}(\Gamma_D)$. We consider a bounded linear mapping

$$V : \widetilde{H}^{-1/2}(\Gamma_D) \to H^{1/2}(\Gamma_D)$$

such that

$$(\phi, \psi) \mapsto \langle V\phi, \psi\rangle$$

defines an elliptic bilinear form on $\widetilde{H}^{-1/2}(\Gamma_D)$. For a given right-hand side $F \in H^{1/2}(\Gamma_D)$, we want to find a solution $\phi \in \widetilde{H}^{-1/2}(\Gamma_D)$ of

$$V\phi = F.$$

It follows from the Hahn-Banach theorem that the topological dual space of a normed space separates points, see e.g. [Rud91, page 60]. Hence, we get

$$\forall \psi \in \widetilde{H}^{-1/2}(\Gamma_D): \quad \langle V\phi, \psi\rangle = \langle F, \psi\rangle$$

which has a unique solution due to the Lax-Milgram lemma, see e.g. [BS08, Theorem 2.7.7]. This also implies the bijectivity of the operator V.

The following definition and lemma are taken from [SS11, page 29–30].

Definition 3.1. If $X \leq Y$ are real Hilbert spaces such that the identity $I : X \to Y$ is a continuous and dense embedding, we call (X, Y, X') a *Gelfand triple*.

Lemma 3.2. *Let (X, Y, X') be a Gelfand triple. Then Y' is continuously, densely and injectively embedded in X' by the mapping $y' \mapsto y'|_X$.*

Proof. First we check that the mapping really maps to X'. Let $y' \in Y'$ and $x \in X$. Then we have

$$|y'(x)| \leq \|y'\|_{Y'}\|x\|_Y \leq \|y'\|_{Y'}\|x\|_X\|I : X \to Y\|.$$

The continuity of the mapping follows from

$$\|y'|_X\|_{X'} = \sup_{x\in X\setminus\{0\}} \frac{|y'(x)|}{\|x\|_X} \leq \|y'\|_{Y'}\|I : X \to Y\|.$$

We prove that $Y'|_X$ is dense in X' by showing that the orthogonal complement of $Y'|_{X'}$ in X' is $\{0\}$. Let x' be an element of the complement and $x := J_X^{-1}x'$, with the Riesz mapping

$$J_X : X \to X' : x \mapsto \langle \cdot, x\rangle_X.$$

Then, we have with the Riesz mapping $J_Y : Y \to Y'$

$$0 = \langle x', (J_Y x)|_X\rangle = \langle x, J_X^{-1}((J_Y x)|_X)\rangle_X = (J_Y x)(x) = \langle x, x\rangle_Y$$

and hence $x' = 0$. The injectivity directly follows from the fact that two continuous functions on Y coincide if they coincide on the dense subset X. □

Due to the Riesz representation theorem one can identify the Hilbert space Y with its dual space Y' by the Riesz mapping

$$J_Y : Y \to Y' : y \mapsto \langle \cdot, y \rangle_Y.$$

Because of Lemma 3.2, we can identify Y with $J_Y(Y)|_X$ which is a dense subspace of X'. Hence, we have

$$X \leq Y \leq X',$$

where every inclusion is dense.

Lemma 3.3. *The spaces $X := H^{1/2}(\Gamma_D), Y := L^2(\Gamma_D)$, and $\widetilde{H}^{-1/2}(\Gamma_D)$ define a Gelfand triple.*

Proof of Lemma 3.3 for $\Gamma_D = \Gamma$. Obviously the identity $I : H^{1/2}(\Gamma) \to L^2(\Gamma)$ is continuous. To see the density of $H^{1/2}(\Gamma)$ in $L^2(\Gamma)$ we proceed as follows.

First we note that $C_{00}\big((a,b)\big)$, the space of all continuous functions with compact support in (a,b), is dense in $L^2\big((a,b)\big)$. This is a direct consequence of the fundamental lemma of calculus of variations.

Because of Lemma 2.4 we have for all measurable u on (a,b)

$$\int_{\Gamma \setminus \{\gamma(a)\}} \big|u\big(\gamma|^{-1}_{(a,b)}(x)\big)\big|^2 \,\mathrm{d}\mu_\Gamma(x) = \int_{(a,b)} |u(t)|^2 \,\mathrm{d}\lambda(t).$$

This implies that the mapping $(\cdot) \circ \gamma|^{-1}_{(a,b)}$ is an isometric isomorphism from $L^2\big((a,b)\big)$ to $L^2(\Gamma \setminus \{\gamma(a)\})$. Therefore $C_{00}\big((a,b)\big) \circ \gamma|^{-1}_{(a,b)}$ is dense in $L^2(\Gamma \setminus \{\gamma(a)\})$. We recall that $\gamma|_{(a,b)}$ is a homeomorphism to $\Gamma \setminus \{\gamma(a)\}$. Moreover each element of $C_{00}\big((a,b)\big) \circ \gamma|^{-1}_{(a,b)}$ can be extended continuously at the point $\gamma(a)$ with the value 0. Hence we can identify $C_{00}\big((a,b)\big) \circ \gamma|^{-1}_{(a,b)}$ as subset of $C(\Gamma)$. Since Γ is compact and $L^2(\Gamma \setminus \{\gamma(a)\})$ can be identified with $L^2(\Gamma)$, we conclude that $C(\Gamma)$ is a dense subset of $L^2(\Gamma)$.

Obviously $\mathrm{Lip}(\Gamma)$, the space of all Lipschitz functions on Γ, is a subspace of $H^{1/2}(\Gamma)$. Due to [Geo67], $\mathrm{Lip}(\Gamma)$ is dense in $C(\Gamma)$ with respect to $\|\cdot\|_{L^\infty(\Gamma)}$ and therefore also with respect to $\|\cdot\|_{L^2(\Gamma)}$, which concludes the proof. □

Proof of Lemma 3.3 for $\Gamma_D \neq \Gamma$. Because of the fundamental lemma of calculus of variations, the space $C\big([a_D, b_D]\big)$ is dense in $L^2\big([a_D, b_D]\big)$. Applying Lemma 2.4 gives

$$\int_{\Gamma_D} \big|u\big(\gamma|^{-1}_{[a_D,b_D]}(x)\big)\big|^2 \,\mathrm{d}\mu_\Gamma(x) = \int_{[a_D,b_D]} |u(t)|^2 \,\mathrm{d}\lambda(t).$$

This shows that $(\cdot) \circ \gamma|^{-1}_{[a_D,b_D]}$ is an isometric isomorphism from $L^2\big([a_D, b_D]\big)$ to $L^2(\Gamma_D)$. Since $\gamma|_{[a_D,b_D]}$ is a homeomorphism to Γ_D, we conclude that $C(\Gamma_D)$ is dense in $L^2(\Gamma_D)$. Finally, we see as before that $H^{1/2}(\Gamma_D)$ is dense in $L^2(\Gamma_D)$. □

Because of this lemma, it makes sense to discretize the problem by introducing a finite dimensional Galerkin trial space $S \leq L^2(\Gamma_D) \leq \widetilde{H}^{-1/2}(\Gamma_D)$. Because of the Lax-Milgram lemma there exists a unique $\phi_S \in S$ such that

$$\forall \psi_S \in S : \quad \langle V\phi_S, \psi_S \rangle_{L^2(\Gamma_D)} = \langle F\psi_S \rangle_{L^2(\Gamma_D)}. \tag{3.1}$$

The Lemma of Céa, see for example [BS08, Theorem 2.8.1], states quasi-optimality

$$\|\phi - \phi_S\|_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{Cea} \inf_{\psi_S \in S} \|\phi - \psi_S\|_{\widetilde{H}^{-1/2}(\Gamma_D)}, \tag{3.2}$$

where C_{Cea} is the ratio of the continuity constant of the operator V and the ellipticity constant of the bilinear form. For adaptive mesh-refinement, we want to estimate the discretitization error

$$e_S := \phi - \phi_S \in \widetilde{H}^{-1/2}(\Gamma_D)$$

by computable local quantities. After the calculation of ϕ_S, one can compute the residual

$$r_S := Ve_S = F - V\phi_S \in H^{1/2}(\Gamma_D).$$

Due to the open mapping theorem, see for instance [Rud91, Chapter 2.11], there holds for $\psi \in \widetilde{H}^{-1/2}(\Gamma_D)$

$$C_{op,1}\|V\psi\|^2_{H^{1/2}(\Gamma_D)} \leq \|\psi\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{op,2}\|V\psi\|^2_{H^{1/2}(\Gamma_D)}$$

with the constants $C_{op,1} := \|V\|^{-2}$ and $C_{op,2} := \|V^{-1}\|^2$. Therefore, the error is equivalent to the residual, i.e.

$$C_{op,1}\|r_S\|^2_{H^{1/2}(\Gamma_D)} \leq \|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{op,2}\|r_S\|^2_{H^{1/2}(\Gamma_D)}. \tag{3.3}$$

Using this and the results of Section 2, we can construct three different local error estimators. We present them and show their efficiency resp. reliability in the following two subsections.

3.1. **Faermann-Estimators.** Analogous theorems as in this subsection have first been proven by Faermann, see [Fae00, Theorem 3.1 and Theorem 3.2]. She considered $\Gamma_D = \Gamma$ and instead of the space $S^m(\mathcal{T}_h)$, she used transformed piecewise polynomials on $[0, L]$, see [Fae00, page 206].

Theorem 3.4. *Let $m \in \mathbb{N}_0$, $\mathcal{T}_h$ be a mesh on Γ_D with $|\mathcal{T}_h| \geq 4$ and $h \leq \mu_\Gamma(\Gamma)/4$ and [3] $S = S^m(\mathcal{T}_h) \leq L^2(\Gamma_D)$ a finite dimensional trial space satisfying Assumption 2.10. Then, there holds*

$$C^{F_1}_{eff} \sum_{x \in \mathcal{N}_h} \eta^{F_1}_h(x)^2 \leq \|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C^{F_1}_{rel} \sum_{x \in \mathcal{N}_h} \eta^{F_1}_h(x)^2, \tag{3.4}$$

where for $x \in \mathcal{N}_h$

$$\eta^{F_1}_h(x)^2 := |r_S|^2_{H^{1/2}(\omega_h(x))} \tag{3.5}$$

and

$$C^{F_1}_{eff} = \frac{C_{op,1}}{2} \qquad \text{and} \qquad C^{F_1}_{rel} = C_{op,2}C_{main}.$$

Here, $C_{main} = C_{main}(h, \kappa(\mathcal{T}_h), m, q_{space})$ is the constant of Theorem 2.11. Note that C_{main} depends in a monotonously decreasing way on h, whereas the dependence on the other parameters is monotonously increasing.

[3] Indeed, the trial space only needs to be a superspace of $S^m(\mathcal{T}_h)$.

Proof. The lower estimate follows directly from Theorem 2.12 and (3.3)

$$\sum_{x\in\mathcal{N}_h} \eta_h^{F_1}(x)^2 = \sum_{x\in\mathcal{N}_h} |r_S|^2_{H^{1/2}(\omega_h(x))} \leq 2\|r_S\|^2_{H^{1/2}(\Gamma_D)} \leq \frac{2}{C_{op,1}}\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)}.$$

The residual r_S is orthogonal to $S^m(\mathcal{T}_h)$ due to (3.1). Hence the upper estimate is a consequence of (3.3) and Theorem 2.11

$$\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{op,2}\|r_S\|^2_{H^{1/2}(\Gamma_D)} \leq C_{op,2}C_{main}\sum_{x\in\mathcal{N}_h} |u|^2_{H^{1/2}(\omega_h(x))} = C_{rel}^{F_1}\sum_{x\in\mathcal{N}_h}\eta_h^{F_1}(x)^2.$$

This concludes the proof. □

Theorem 3.5. *Let $m \in \mathbb{N}_0$, $\mathcal{T}_h$ be a mesh on Γ_D with $|\mathcal{T}_h| \geq 4$ and $h \leq \mu_\Gamma(\Gamma)/4$, and [4] $S = S^m(\mathcal{T}_h) \leq L^2(\Gamma_D)$ a finite dimensional trial space satisfying Assumption 2.10. For $x \in \mathcal{N}_h$ we define*

$$\eta_h^{F_2}(x)^2 := |r_S|^2_{H^{1/2}(\omega_h(x))} + h^{-1}_{T_{x,2}}\|r_S\|^2_{L^2(T_{x,1})} + h^{-1}_{T_{x,1}}\|r_S\|^2_{L^2(T_{x,2})}, \tag{3.6}$$

if x is an interior node[5], else we set

$$\eta_h^{F_2}(x_0)^2 := |r_S|^2_{H^{1/2}(\omega_h(x))} + h^{-1}_{T_x^-}\|r_S\|^2_{L^2(T_x)}, \tag{3.7}$$

and

$$\eta_h^{F_2}(x_{|\mathcal{T}_h|})^2 := |r_S|^2_{H^{1/2}(\omega_h(x))} + h^{-1}_{T_x^+}\|r_S\|^2_{L^2(T_x)}, \tag{3.8}$$

Then there holds

$$C_{eff}^{F_2}\sum_{x\in\mathcal{N}_h}\eta_h^{F_2}(x)^2 \leq \|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{rel}^{F_2}\sum_{x\in\mathcal{N}_h}\eta_h^{F_2}(x)^2, \tag{3.9}$$

where

$$C_{eff}^{F_2} = \frac{C_{op,1}}{2(1+C_{poinc})} \qquad \text{and} \qquad C_{rel}^{F_2} = \max(1, C_{loc})C_{op,2}.$$

Here, $C_{poinc} = C_{poinc}(h, \kappa(\mathcal{T}_h), m, q_{space})$ is the constant of Proposition 2.16. Note that C_{main} depends in a monotonously decreasing way on h, whereas the dependence on the other parameters is monotonously increasing.

Proof. The upper estimate is an immediate consequence of (3.3) and Proposition 2.13

$$\begin{aligned}\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} &\leq C_{op,2}\|r_S\|^2_{H^{1/2}(\Gamma_D)}\\ &\leq C_{op,2}\sum_{x\in\mathcal{N}_h}|r_S|^2_{H^{1/2}(\omega_h(x))} + C_{op,2}C_{loc}\sum_{T\in\mathcal{T}_h}(h^{-1}_{T^-}+h^{-1}_{T^+})\|r_S\|^2_{L^2(T)}\\ &\leq C_{rel}^{F_2}\sum_{x\in\mathcal{N}_h}\eta_h^{F_2}(x)^2.\end{aligned}$$

[4] Indeed, the trial space only needs to be a superspace of $S^m(\mathcal{T}_h)$.

[5] This means that $\Gamma_D = \Gamma$, or, $\Gamma_D \neq \Gamma$ and $x \neq x_0$ and $x \neq x_{|\mathcal{T}_h|}$.

The residual r_S is orthogonal to S due to (3.1). Therefore the lower estimate holds because of Proposition 2.16, Theorem 2.12 and (3.3)

$$\begin{aligned}\sum_{x\in\mathcal{N}_h}\eta_h^{F_2}(x)^2 &= \sum_{x\in\mathcal{N}_h}|r_S|^2_{H^{1/2}(\omega_h(x))}+\sum_{T\in\mathcal{T}_h}(h_{T^-}^{-1}+h_{T^+}^{-1})\|r_S\|^2_{L^2(T)}\\ &\le (1+C_{poinc})\sum_{x\in\mathcal{N}_h}|r_S|^2_{H^{1/2}(\omega_h(x))}\\ &\le (1+C_{poinc})2\|r_S\|^2_{H^{1/2}(\Gamma_D)}\\ &\le (1+C_{poinc})\frac{2}{C_{op,1}}\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)}.\end{aligned}$$

This concludes the proof. □

3.2. **Weighted-residual error estimator.** Before we present the weighted-residual estimator η_h^R and show its reliability, we start with the following lemma and proposition. Analogous versions have been proved by Carstensen and Faermann in [CF01, Theorem 7.2, Lemma 7.4]. The proof of the following lemma was inspired by [NPV11, Proposition 2.2]

Lemma 3.6. *Let $I\subseteq\mathbb{R}$ be a finite nonempty open interval. Then, there holds for all $u\in H^1(I)$*

$$|u|^2_{H^{1/2}(I)}\le 2\lambda(I)|u|^2_{H^1(I)}. \tag{3.10}$$

Proof. We recall that $H^1(I)$ coincides with the space of all absolutely continuous functions on $\overline{I}$ with L^2 derivative, i.e.

$$H^1(I)=\big\{u\in AC(\overline{I})\,:\,u'\in L^2(I)\big\}; \tag{3.11}$$

for a proof, see e.g. [Kal13, Beispiel 19.1.6]. First we consider $I=(0,1)$. We use the transformation theorem, with $r=\rho(s-t)+t$ and $s-t=\sigma$, as well as the Cauchy Schwarz inequality to get

$$\begin{aligned}|u|^2_{H^{1/2}(I)} &= \int_I\int_I\left|\frac{u(s)-u(t)}{s-t}\right|^2\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\ &= \int_I\int_I\left|\frac{\int_{(0,s)}u'(r)\,\mathrm{d}\lambda(r)-\int_{(0,t)}u'(r)\,\mathrm{d}\lambda(r)}{s-t}\right|^2\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\ &= \int_I\int_I\left|\int_I u'\big(\rho(s-t)+t\big)\,\mathrm{d}\lambda(\rho)\right|^2\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\ &\le \int_I\int_I\int_I\big|u'\big(\rho(s-t)+t\big)\big|^2\,\mathrm{d}\lambda(\rho)\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\ &= \int_I\int_{(-t,1-t)}\int_I\big|u'\big(\rho\sigma+t\big)\big|^2\,\mathrm{d}\lambda(\rho)\,\mathrm{d}\lambda(\sigma)\,\mathrm{d}\lambda(t).\end{aligned}$$

We formally extend u' to $\mathbb{R}$, by defining it on $\mathbb{R} \setminus I$ as zero. This and the Fubini theorem lead to

$$\begin{aligned}
|u|^2_{H^{1/2}(I)} &\le \int_I \int_{(-1,1)} \int_I \big|u'\big(\rho\sigma + t\big)\big|^2 \,\mathrm{d}\lambda(\rho)\,\mathrm{d}\lambda(\sigma)\,\mathrm{d}\lambda(t) \\
&\le \int_I \int_{(-1,1)} \int_{\mathbb{R}} \big|u'\big(\rho\sigma + t\big)\big|^2 \,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(\sigma)\,\mathrm{d}\lambda(\rho) \\
&= \int_I \int_{(-1,1)} \|u'\|^2_{L^2(\mathbb{R})} \,\mathrm{d}\lambda(\sigma)\,\mathrm{d}\lambda(\rho) = 2|u|^2_{H^1(I)}.
\end{aligned}$$

Hence, we have

$$|u|^2_{H^{1/2}((0,1))} \le 2|u|^2_{H^1((0,1))}. \tag{3.12}$$

If $I = (a,b)$ is arbitrary, we define the function $u_{(0,1)} : (0,1) \to \mathbb{R} : \tau \mapsto u\big(\tau(b-a)+a\big)$. Obviously the function is in $H^1\big((0,1)\big)$, where $u'_{(0,1)}(\tau) = (b-a)u'\big(\tau(b-a)+a\big)$ for almost every $\tau \in (0,1)$. We use the transformation theorem, with $s = \sigma(b-a)+a$, $t = \tau(b-a)+a$ and $r = \rho(b-a)+a$, as well as (3.12) for $u_{(0,1)}$ to get

$$\begin{aligned}
|u|^2_{H^{1/2}(I)} &= \int_I \int_I \left|\frac{u(s)-u(t)}{s-t}\right|^2 \,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t) \\
&= \int_{(0,1)} \int_{(0,1)} \left|\frac{u\big(\sigma(b-a)+a\big) - u\big(\tau(b-a)+a\big)}{\sigma-\tau}\right|^2 \,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t) \\
&= |u_{(0,1)}|^2_{H^{1/2}((0,1))} \le 2|u_{(0,1)}|^2_{H^1((0,1))} \\
&= 2\int_{(0,1)} |u'_{(0,1)}(\rho)|^2 \,\mathrm{d}\lambda(\rho) = 2(b-a)\int_I |u'(r)|^2 \,\mathrm{d}\lambda(r) = 2\lambda(I)|u|_{H^1(I)}.
\end{aligned}$$

This is just the assertion of the lemma. □

Let I be a real interval with length $0 < \lambda(I) < L$. With the L-periodic extended arc length parametrization γ_L of γ as in Lemma A.7, we define[6]

$$H^1\big(\gamma_L(I)\big) := \big\{u \in L^2\big(\gamma_L(I)\big) \,:\, u \circ \gamma_L|_I \in H^1(I)\big\}.$$

For $u \in H^1\big(\gamma_L(I)\big)$, we set for almost every $t \in I$

$$\partial_\Gamma u\big(\gamma_L(t)\big) := (u \circ \gamma_L)'(t).$$

Moreover we define

$$H^1(\Gamma) := \big\{u \in L^2(\Gamma) \,:\, u \circ \gamma_L|_J \in H^1(J) \text{ for all finite intervals } J \subseteq \mathbb{R}\big\}.$$

For $u \in H^1(\Gamma)$, we set for almost every $t \in \mathbb{R}$

$$\partial_\Gamma u\big(\gamma_L(t)\big) := (u \circ \gamma_L)'(t).$$

[6]Here, we actually mean $H^1(I^\circ)$, where I° denotes the interior of I. Note that $\gamma(I_1) = \gamma(I_2)$ for intervals I_1, I_2 with length smaller than L, implies $I_1 = I_2 + qL$ with some $q \in \mathbb{Z}$.

Proposition 3.7. *Let $\mathcal{T}_h$ be a mesh on Γ_D with $h \leq \mu_\Gamma(\Gamma)/4$. Then, there holds for all $u \in H^1(\Gamma_D)$*

$$|u|^2_{H^{1/2}(\omega_h(x))} \leq 2C_\Gamma^2 \mu_\Gamma\big(\omega_h(x)\big) \|\partial_\Gamma u\|^2_{L^2(\omega_h(x))}. \tag{3.13}$$

Here, C_Γ is the constant of Lemma 2.1.

Proof. Let I be a real interval with $\gamma_L(I) = \omega_h(x)$. Lemma 2.8 and Lemma 3.6 show

$$|u|^2_{H^{1/2}(\gamma_L(I))} \leq C_\Gamma^2 |u \circ \gamma_L|^2_{H^{1/2}(I)} \leq 2C_\Gamma^2 \mu_\Gamma\big(\omega_h(x)\big) |u \circ \gamma_L|^2_{H^1(I)}.$$

Due to Lemma 2.4, $|u \circ \gamma_L|^2_{H^1(I)}$ just coincides with $\|\partial_\Gamma u\|^2_{L^2(\omega_h(x))}$. □

Theorem 3.8. *Let $m \in \mathbb{N}_0$, $\mathcal{T}_h$ be a mesh on Γ_D with $|\mathcal{T}_h| \geq 4$ and $h \leq \mu_\Gamma(\Gamma)/4$ and [7] $S = S^m(\mathcal{T}_h) \leq L^2(\Gamma_D)$ a finite dimensional trial space satisfying Assumption 2.10. For $r_S \in H^1(\Gamma_D)$, there holds*

$$\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C^R_{rel} \sum_{x \in \mathcal{N}_h} \eta^R_h(x)^2, \tag{3.14}$$

where for $x \in \mathcal{N}_h$

$$\eta^R_h(x)^2 := \mu_\Gamma(\omega_h(x)) \|\partial_\Gamma r_S\|^2_{L^2(\omega_h(x))}. \tag{3.15}$$

and

$$C^R_{rel} = 2C_\Gamma^2 C^{F_1}_{rel}.$$

Here, C_Γ is the constant of Lemma 2.1 and $C^{F_1}_{rel}$ is the constant of Theorem 3.4.

Proof. Due to Theorem 3.4 and Proposition 3.7 there holds

$$\|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C^{F_1}_{rel} \sum_{x \in \mathcal{N}_h} |r_S|^2_{H^{1/2}(\omega_h(x))} \leq C^R_{rel} \sum_{x \in \mathcal{N}_h} \eta^R_h(x),$$

concluding the proof. □

Remark 3.9. In Theorem 3.4, Theorem 3.5, and Theorem 3.8, the error estimators are node based. With these theorems it is easy to construct element based estimators. Indeed, if $\eta_h(x)$ is a node based error estimator, one can define for $T \in \mathcal{T}_h$

$$\eta_h(T)^2 := \eta_h(x_{T,1})^2 + \eta_h(x_{T,2})^2. \tag{3.16}$$

If the node based error estimator is efficient and reliable with constants C_{eff} and C_{rel}, the element based one satisfies

$$\frac{C_{eff}}{2} \sum_{T \in \mathcal{T}_h} \eta_h(T)^2 \leq \|e_S\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \leq C_{rel} \sum_{T \in \mathcal{T}_h} \eta_h(T)^2. \tag{3.17}$$

In Theorem 3.8, we can also define another element based version as

$$\eta^R_h(T)^2 := h_T \|\partial_\Gamma r_S\|^2_{L^2(T)} \tag{3.18}$$

for $T \in \mathcal{T}_h$. It is reliable with constant $C_{rel} = 2(\kappa(\mathcal{T}_h) + 1)C^R_{rel}$.

[7] Indeed, the trial space only needs to be a superspace of $S^m(\mathcal{T}_h)$.

4. Adaptive Algorithm with B-splines and NURBS

Let $\Omega \subset \mathbb{R}^2$ be an open set, Γ_D a subset of the boundary Γ, which is parametrized by $\gamma : [a,b] \to \Gamma$, as in Section 2. Additionally, we assume that γ is piecewise two times differentiable. We consider the problem $V\phi = F$, where $V : \widetilde{H}^{-1/2}(\Gamma_D) \to H^{1/2}(\Gamma_D)$ and $F \in H^{1/2}(\Gamma_D)$ are as in Section 3.

It is our goal to approximate the solution ϕ. To this end, we will use the linear span of B-splines and NURBS, transformed to the boundary by the parametrization γ, as approximation spaces. They will be presented in the first subsection. Here, we will also show that the transformed span satisfies Assumption 2.10 of Section 2. In [Fae00], Faermann only considered the span of B-splines, transformed to the boundary by an arc length parametrization of the boundary. In the second subsection, we introduce an adaptive algorithm based on the error estimators of Section 3.

4.1. B-splines and NURBS.

Throughout this subsection, we consider a sequence of *knots* $\check{\mathcal{K}} := (t_i)_{i\in\mathbb{Z}}$ on $\mathbb{R}$ with $t_{i-1} \le t_i$ for $i \in \mathbb{Z}$ and $\lim_{i\to\pm\infty} t_i = \pm\infty$. For the multiplicity of any knot t_i we write $\#t_i$. We denote the corresponding set of *nodes* $\check{\mathcal{N}} := \{t_i : i \in \mathbb{Z}\} = \{\check{x}_j : j \in \mathbb{Z}\}$ with $\check{x}_{j-1} < \check{x}_j$ for $j \in \mathbb{Z}$.

Definition 4.1. For $i \in \mathbb{Z}$, we define by inductivity the i-th *B-Spline* of degree p

$$\begin{aligned} B_{i,0} &:= \chi_{[t_{i-1},t_i)}, \\ B_{i,p} &:= \beta_{i-1,p} B_{i,p-1} + (1-\beta_{i,p}) B_{i+1,p-1} \quad \text{for } p \in \mathbb{N}, \end{aligned} \tag{4.1}$$

where, for $t \in \mathbb{R}$

$$\beta_{i,p}(t) := \begin{cases} \frac{t-t_i}{t_{i+p}-t_i} & \text{if } t_i \neq t_{i+p}, \\ 0 & \text{if } t_i = t_{i+p}. \end{cases}$$

We also use the notations $B^{\check{\mathcal{K}}}_{i,p} := B_{i,p}$ and $\beta^{\check{\mathcal{K}}}_{i,p} := \beta_{i,p}$ to stress the dependence on the knots $\check{\mathcal{K}}$.

The following assertions about B-splines are mainly taken from [dB86].

Lemma 4.2. *For $p \in \mathbb{N}_0$, the following assertions hold:*

(i) *For $i \in \mathbb{Z}$ and $\ell \in \mathbb{Z}$, $B_{i,p}|_{[t_{\ell-1},t_\ell)}$ is a polynomial of degree p.*

(ii) *For $i \in \mathbb{Z}$, $B_{i,p}$ vanishes outside the interval $[t_{i-1}, t_{i+p})$. It is positive on the open interval (t_{i-1}, t_{i+p}).*

(iii) *For $i \in \mathbb{Z}$, it holds $t_{i-1} = t_{i+p}$ if and only if $B_{i,p} = 0$.*

(iv) *For $i \in \mathbb{Z}$, $B_{i,p}$ is completely determined by the $p+2$ knots $t_{i-1}, \dots, t_{i+p}$. Therefore, we will also use the notation*

$$B(\cdot | t_{i-1}, \dots, t_{i+p}) := B_{i,p}. \tag{4.2}$$

(v) *For $i \in \mathbb{Z}$ and $s \in \mathbb{R}$, we have*

$$\forall t \in \mathbb{R} : \quad B^{s+\check{\mathcal{K}}}_{i,p}(t) = B^{\check{\mathcal{K}}}_{i,p}(t-s), \tag{4.3}$$

and for $c > 0$

$$\forall t \in \mathbb{R} : \quad B^{c\check{\mathcal{K}}}_{i,p}(t) = B^{\check{\mathcal{K}}}_{i,p}(t/c). \tag{4.4}$$

(vi) *For $\ell \in \mathbb{N}$, let $\check{\mathcal{K}}_\ell = (t_{i,\ell})_{i\in\mathbb{Z}}$ be a sequence of knots such that $\#t_{i,\ell} = \#t_i$ for all $i \in \mathbb{Z}$. If $(\check{\mathcal{K}}_\ell)_{\ell\in\mathbb{N}}$ converges pointwise to $\check{\mathcal{K}}$, then $\left(B_{i,p}^{\check{\mathcal{K}}_\ell}\right)_{\ell\in\mathbb{N}}$ converges almost everywhere to $B_{i,p}^{\check{\mathcal{K}}}$ for all $i \in \mathbb{N}$.*

(vii) *The B-splines of degree p form a partition of unity, i.e.*

$$\sum_{i\in\mathbb{Z}} B_{i,p} = 1 \quad \text{on } \mathbb{R}. \tag{4.5}$$

Proof. The proof for (i)–(iv) can be found in [dB86, Section 2]. To prove (v), we note that for all $\ell \in \mathbb{Z}$ and $t \in \mathbb{R}$ there holds

$$\chi_{[s+t_{\ell-1},s+t_\ell)}(t) = \chi_{[t_{\ell-1},t_\ell)}(t-s) \quad \text{and} \quad \chi_{[ct_{\ell-1},ct_\ell+s)}(t) = \chi_{[t_{\ell-1},t_\ell)}(t/c)$$

as well as

$$\frac{t-(s+t_\ell)}{(s+t_{\ell+p})-(s+t_\ell)} = \frac{(t-s)-t_\ell}{t_{\ell+p}-t_\ell} \quad \text{and} \quad \frac{t-ct_\ell}{ct_{\ell+p}-ct_\ell} = \frac{t/c-t_\ell}{t_{\ell+p}-t_\ell}.$$

Hence, the assertion is an immediate consequence of the definition of B-splines. (vi) is proved by induction, noting that for all $p' \in \mathbb{N}$ and $i \in \mathbb{Z}$, we have

$$\beta_{i,p'}^{\check{\mathcal{K}}_\ell} \xrightarrow{a.e.} \beta_{i,p'}^{\check{\mathcal{K}}} \qquad \text{and} \qquad B_{i,0}^{\check{\mathcal{K}}_\ell} \xrightarrow{a.e.} B_{i,0}^{\check{\mathcal{K}}}.$$

(vii) is proved in [dB86, page 9–10]. □

With Lemma 4.2, (ii), we can define for any $p \in \mathbb{N}_0$ the vector space

$$\mathscr{S}^p(\check{\mathcal{K}}) := \left\{ \sum_{i\in\mathbb{Z}} a_i B_{i,p} : a_i \in \mathbb{R} \right\}. \tag{4.6}$$

Note that the sum is locally finite.

The following two theorems are proved in [dB86, Theorem 5, Theorem 6].

Theorem 4.3. *The space $\mathscr{S}^p(\check{\mathcal{K}})$ coincides with the space of all right-continous piecewise polynomials of degree lower or equal p with break points $(t_i)_{i\in\mathbb{Z}}$ which are, for all $i \in \mathbb{Z}$, at least $p - \#t_i$ times continuously differentiable at t_i if $p - \#t_i \geq 0$.*

Theorem 4.4. *Let $I = [a, b)$ be a finite interval and $p \in \mathbb{N}_0$. Then*

$$\left\{ B_{i,p}|_I \,:\, i \in \mathbb{Z}, B_{i,p}|_I \neq 0 \right\} \tag{4.7}$$

is a basis for the space of all right-continuous piecewise polynomials of degree lower or equal p on I with breakpoints $\check{\mathcal{N}} \cap (a, b)$ and which are, at each break point t_i, $p - \#t_i$ times continuously differentiable if $p - \#t_i \geq 0$.

Remark 4.5. If we assume $\#t_i \leq p + 1$ for all $i \in \mathbb{Z}$, Lemma 4.2, (ii), implies that each element in $\mathscr{S}^p(\check{\mathcal{K}})$ has a unique representation of the form

$$\sum_{i\in\mathbb{Z}} a_i B_{i,p}.$$

For the proof of the following corollary and two lemmas, we refer to [dB86, Corollary 2, Algorithm 11, Section 10].

Corollary 4.6. *Let $p \in \mathbb{N}_0$. If $\check{\mathcal{K}}'$ is a refinement of $\check{\mathcal{K}}$, i.e. $\check{\mathcal{K}} = (t_i)_{i\in\mathbb{Z}}$ is a subsequence of $\check{\mathcal{K}}' = (t'_i)_{i\in\mathbb{Z}}$, then*

$$\mathscr{S}^p(\check{\mathcal{K}}) \subseteq \mathscr{S}^p(\check{\mathcal{K}}'). \tag{4.8}$$

Lemma 4.7. *Let $p \in \mathbb{N}_0$, $\ell \in \mathbb{Z}$ with $t_{\ell-1} < t_\ell$ and $\check{\mathcal{K}}'$ the refinement of $\check{\mathcal{K}}$, obtained by adding $t' := (t_{\ell-1} + t_\ell)/2$ such that $t'_\ell = t'$. If $a_i \in \mathbb{R}$, $i \in \mathbb{Z}$, are some coefficients there exist $a'_i \in \mathbb{R}$ with*

$$\sum_{i\in\mathbb{Z}} a_i B_{i,p}^{\check{\mathcal{K}}} = \sum_{i\in\mathbb{Z}} a'_i B_{i,p}^{\check{\mathcal{K}}'} \tag{4.9}$$

The new coefficients can be chosen as follows

$$a'_i = \begin{cases} a_i & \text{if } i \le \ell - p, \\ (1 - \beta_{i-1,p}^{\check{\mathcal{K}}}(t'))a_{i-1} + \beta_{i-1,p}^{\check{\mathcal{K}}}(t')a_i & \text{if } \ell + 1 - p \le i \le \ell, \\ a_{i-1} & \text{if } \ell + 1 \le i. \end{cases}$$

Lemma 4.8. *Let $p \in \mathbb{N}$. Then we have*

$$B'^r_{i,p} = \frac{p}{t_{i+p-1} - t_{i-1}} B_{i,p-1} - \frac{p}{t_{i+p} - t_i} B_{i+1,p-1}, \tag{4.10}$$

where we set $\frac{p}{0} := 0$.

In addition to the given knots $\check{\mathcal{K}} = (t_i)_{i\in\mathbb{Z}}$, let $\mathcal{W} := (w_i)_{i\in\mathbb{Z}}$ be a sequence of fixed positive weights $w_i > 0$. Then, we define the corresponding NURBS functions.

Definition 4.9. For $i \in \mathbb{Z}$ and $p \in \mathbb{N}_0$, we define the i-th *non-uniform rational B-Spline* of degree p or shortly *NURBS* as

$$R_{i,p} := \frac{w_i B_{i,p}}{\sum_{\ell\in\mathbb{Z}} w_\ell B_{\ell,p}}. \tag{4.11}$$

Note that the denominator is never zero because of Lemma 4.2 (ii) and (vii). We also use the notation $R_{i,p}^{\check{\mathcal{K}},\mathcal{W}} := R_{i,p}$.

For NURBS functions there hold analogous properties as for B-splines.

Lemma 4.10. *For $p \in \mathbb{N}_0$, the following assertions hold:*

(i) *For $i \in \mathbb{Z}$ and $\ell \in \mathbb{Z}$, $R_{i,p}|_{[t_{\ell-1},t_\ell)}$ is a rational function with nonzero denominator, which can be extended continuously at t_ℓ.*

(ii) *For $i \in \mathbb{Z}$, $R_{i,p}$ vanishes outside the interval $[t_{i-1}, t_{i+p})$. It is positive on the open interval (t_{i-1}, t_{i+p}).*

(iii) *For $i \in \mathbb{Z}$, it holds $t_{i-1} = t_{i+p}$ if and only if $R_{i,p} = 0$.*

(iv) *For $i \in \mathbb{Z}$, $R_{i,p}$ is completely determined by the $3p + 2$ knots $t_{i-p-1}, \dots, t_{i+2p}$ and the $2p + 1$ weights $w_{i-p}, \dots, w_{i+p}$. Therefore we will also use the notation*

$$R(\cdot|t_{i-p-1}, \dots, t_{i+2p}, w_{i-p}, \dots, w_{i+p}) := R_{i,p}.$$

(v) *For $i \in \mathbb{Z}$ and $s \in \mathbb{R}$ we have*

$$\forall t \in \mathbb{R}: \quad R_{i,p}^{s+\check{\mathcal{K}},\mathcal{W}}(t) = R_{i,p}^{\check{\mathcal{K}},\mathcal{W}}(t - s), \tag{4.12}$$

and for $c > 0$

$$\forall t \in \mathbb{R}: \quad R_{i,p}^{c\check{\mathcal{K}},\mathcal{W}}(t) = R_{i,p}^{\check{\mathcal{K}},\mathcal{W}}(t/c). \tag{4.13}$$

(vi) *For $\ell \in \mathbb{N}$, let $\check{\mathcal{K}}_\ell = (t_{i,\ell})_{i\in\mathbb{Z}}$ be a sequence of knots such that $\#t_{i,\ell} = \#t_i$ for all $i \in \mathbb{Z}$ and, $\mathcal{W}_\ell = (w_{i,\ell})_{i\in\mathbb{Z}}$ a sequence of positive weights. If $(\check{\mathcal{K}}_\ell)_{\ell\in\mathbb{N}}$ converges pointwise to $\check{\mathcal{K}}$ and $(\mathcal{W}_\ell)_{\ell\in\mathbb{N}}$ converges pointwise to $\mathcal{W}$, then $\left(R_{i,p}^{\check{\mathcal{K}}_\ell,\mathcal{W}_\ell}\right)_{\ell\in\mathbb{N}}$ converges almost everywhere to $R_{i,p}^{\check{\mathcal{K}},\mathcal{W}}$ for all $i \in \mathbb{N}$.*

(vii) *The NURBS functions of degree p form a partition of unity, i.e.*

$$\sum_{i\in\mathbb{Z}} R_{i,p} = 1 \quad \text{on } \mathbb{R}. \tag{4.14}$$

(viii) *For all $\ell \in \mathbb{Z}$ each NURBS function $R_{i,p}$ is at least $p - \#t_\ell$ times continuously differentiable at t_ℓ if $p - \#t_\ell \geq 0$.*

(ix) *If all weights are equal, then $R_{i,p} = B_{i,p}$ for all $i \in \mathbb{Z}$. Hence, B-splines are just special NURBS functions.*

Proof. The lemma is an easy consequence of Lemma 4.2 and Theorem 4.3. We only show (vi). Let

$$M := \left\{t \in \mathbb{R} : \lim_{\ell\to\infty} B_{i,p}^{\check{\mathcal{K}}_\ell}(t) = B_{i,p}^{\check{\mathcal{K}}}(t) \text{ for all } i \in \mathbb{Z}\right\}$$

Due to Lemma 4.10, (vi), M is a set of measure zero. For $i \in \mathbb{Z}$ and $t \in M$, there exists a finite index set I such that for all $\ell \in \mathbb{N}$

$$R_{i,p}^{\check{\mathcal{K}}_\ell,\mathcal{W}_\ell}(t) = \frac{w_{i,\ell}B_{i,p}^{\check{\mathcal{K}}_\ell}}{\sum_{q\in I} w_{q,\ell}B_{q,p}^{\check{\mathcal{K}}_\ell}}(t)$$

as well as

$$R_{i,p}^{\check{\mathcal{K}},\mathcal{W}}(t) = \frac{w_i B_{i,p}^{\check{\mathcal{K}}}}{\sum_{q\in I} w_q B_{q,p}^{\check{\mathcal{K}}}}(t).$$

This implies the convergence of $R_{i,p}^{\check{\mathcal{K}}_\ell,\mathcal{W}_\ell}(t)$. □

With Lemma 4.10, (ii), we can define for any $p \in \mathbb{N}_0$ the vector space

$$\mathscr{N}^p(\check{\mathcal{K}},\mathcal{W}) := \left\{\sum_{i\in\mathbb{Z}} a_i R_{i,p} : a_i \in \mathbb{R}\right\} = \frac{\mathscr{S}^p(\check{\mathcal{K}})}{\sum_{i\in\mathbb{Z}} w_i B_{i,p}^{\check{\mathcal{K}}}} \tag{4.15}$$

Note that the sums are locally finite.

Remark 4.11. As in Remark 4.5, $\#t_i \leq p + 1$ for all $i \in \mathbb{Z}$ implies the uniqueness of the representation

$$\sum_{i\in\mathbb{Z}} a_i R_{i,p}.$$

For the proof of her main result [Fae00, Theorem 2.2], Faermann needed a similar version of the following lemma. In the proof of [Fae00, Lemma 2.6], she only considered B-splines of degree $p = 0$, $p = 1$ and $p = 2$, induced by a strictly increasing sequence of knots. The following lemma generalizes her result to arbitrary NURBS functions. Note that the proofs are totally different.

For a finite sequence of knots $s_0 \le \cdots \le s_M$ and corresponding nodes $\check{y}_0, \dots, \check{y}_m$, we define the regularity constant

$$\kappa(s_0, \dots, s_M) := \max\left\{ \frac{\check{y}_{j+1} - \check{y}_j}{\check{y}_j - \check{y}_{j-1}}, \frac{\check{y}_j - \check{y}_{j-1}}{\check{y}_{j+1} - \check{y}_j} : j = 1, \dots, m-1 \right\}, \tag{4.16}$$

where we define the maximum as 1 if $m \le 1$.

Lemma 4.12. *Let I be a compact interval with nonempty interior, $\kappa_{\max} \ge 1$, $0 < w_{\min} \le w_{\max}$ real numbers, $p \in \mathbb{N}_0$, and $\theta : I \to \mathbb{R}^+$ a piecewise continuously differentiable function with positive infimum. Then there exists a constant*

$$q_{NURBS} = q_{NURBS}\big(\kappa_{\max}, w_{\min}, w_{\max}, p, \theta\big) \in (0, 1]$$

such that for arbitrary knots $s_0 \le \cdots \le s_{3p+1} \in I$ with $\kappa(s_0, \dots, s_{3p+1}) \le \kappa_{\max}$, weights $w_{\min} \le v_1, \dots, v_{2p+1} \le w_{\max}$ and all $\ell \in \{p+1, \dots, 2p+1\}$,

$$\big\| \big(1 - R(\cdot | s_0, \dots, s_{3p+1}, v_1, \dots, v_{2p+1})\big) \cdot \theta \big\|_{L^1([s_{\ell-1}, s_\ell])} \le (1 - q_{NURBS}) \|\theta\|_{L^1([s_{\ell-1}, s_\ell])}. \tag{4.17}$$

Note that there holds

$$\operatorname{supp}\big(R(\cdot | s_0, \dots, s_{3p+1}, v_1, \dots, v_{2p+1})\big) = [s_p, s_{2p+1}].$$

q_{NURBS} depends in a monotonously decreasing way of $\kappa_{\max}$ and $w_{\max}$, and in a monotonously increasing way of $w_{\min}$.

Proof. We prove the lemma in five steps.

(1) We give an abstract formulation of the problem. For $1 \le \nu \le 3p+1$, we define the bounded set

$$M_\nu := \Big\{ (\check{y}_0, \dots, \check{y}_\nu, v_1, \dots, v_{2p+1}) \in I^\nu \times [w_{\min}, w_{\max}]^{2p+1} : \check{y}_0 < \check{y}_1,$$
$$\forall q \in \{2, \dots, \nu\} : \frac{1}{\kappa_{\max}}\big(\check{y}_{q-1} - \check{y}_{q-2}\big) \le \check{y}_q - \check{y}_{q-1} \le \kappa_{\max}\big(\check{y}_{q-1} - \check{y}_{q-2}\big) \Big\}.$$

Note that $(\check{y}, v) \in M_\nu$ already implies $\check{y}_0 < \cdots < \check{y}_\nu$. For a vector of multiplicities $k \in \mathbb{N}^{\nu+1}$ with $\sum_{q=0}^{\nu} k_q = 3p+2$ we introduce the function

$$g_{k,\nu} : \mathbb{R}^\nu \to \mathbb{R}^{3p+2} : (\check{y}_0, \dots, \check{y}_\nu) \mapsto (\underbrace{\check{y}_0, \dots, \check{y}_0}_{k_0-\text{times}}, \dots, \underbrace{\check{y}_\nu, \dots, \check{y}_\nu}_{k_\nu-\text{times}}).$$

Moreover we define for $\ell \in \{p+1, \dots, 2p+1\}$ the function, setting $\frac{1}{0} := 0$,

$$\Theta_{k,\ell,\nu} : M_\nu \to \mathbb{R} : (\check{y}, v) \mapsto \frac{\big\| \big(1 - R(\cdot | g_{k,\nu}(\check{y}), v)\big) \cdot \theta \big\|_{L^1([g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell])}}{\|\theta\|_{L^1([g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell])}}.$$

Our aim is to show that for arbitrary k, ℓ, ν there holds $\sup(\Theta_{k,\ell,\nu}(M_\nu)) < 1$. Then we define the constant $(1 - q_{NURBS})$ as the maximum of all these suprema. Note that the maximum is taken over a finite set. Before we proceed, we show that $(1 - q_{NURBS})$ really has the desired properties. Without loss of generality, we can assume that not all considered knots $s_0, \dots, s_{3p+1}$ are equal. The corresponding nodes $\check{y}_0, \dots, \check{y}_\nu$ and weights $v_1, \dots, v_{2p+1}$ are in M_ν. If k is the corresponding multiplicity vector, (4.17) can indeed be equivalently written as

$$\Theta_{k,\ell,\nu}(\check{y}, v) \le (1 - q_{NURBS}).$$

The monotonicity of q_{NURBS} is an immediate consequence of the definition of M_ν.

(2) We fix k, ℓ, ν, where we assume without loss of generality that there exists $0 \leq \widetilde{\nu} \leq \nu$ such that $\ell - 1 = \sum_{q=0}^{\widetilde{\nu}} k_q$, this just means that the appearing integrals have nonempty integration domains $[g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell]$. Using Lemma 4.10, (ii) and (vii), we see that for $(\check{y}, v) \in M_\nu$, the function $R(\cdot|g_{k,\nu}(\check{y}), v)$ attains only values in $[0,1]$ and is positive on the interval $\big(g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell\big)$. This implies

$$\Theta_{k,\ell,\nu}(M_\nu) \subseteq [0,1). \tag{4.18}$$

Because of Lemma 4.10, (vi), we can apply Lebesgue's dominated convergence theorem to see that $\Theta_{k,\ell,\nu}$ is continuous. If M_ν was compact, we would be done. Unfortunately it is not.

(3) Now, we prove the lemma for $\theta = 1$. In the definition of M_ν we replace the interval I by $\mathbb{R}$ to define a superset of M_ν

$$\begin{aligned} M_{\nu,\mathbb{R}} := \Big\{ &(\check{y}, v) \in \mathbb{R}^\nu \times [w_{\min}, w_{\max}]^{2p+1} : \check{y}_0 < \check{y}_1, \\ &\forall q \in \{2, \dots, \nu\} : \frac{1}{\kappa_{\max}}\big(\check{y}_{q-1} - \check{y}_{q-2}\big) \leq \check{y}_q - \check{y}_{q-1} \leq \kappa_{\max}\big(\check{y}_{q-1} - \check{y}_{q-2}\big) \Big\}. \end{aligned}$$

We extend the function $\Theta_{k,\ell,\nu}$ to

$$\widetilde{\Theta}_{k,\ell,\nu} : M_{\nu,\mathbb{R}} \to \mathbb{R} : (\check{y}, v) \mapsto \frac{\big\|1 - R(\cdot|g_{k,\nu}(\check{y}), v)\big\|_{L^1([g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell])}}{g_{k,\nu}(\check{y})_\ell - g_{k,\nu}(\check{y})_{\ell-1}}.$$

We define a closed and bounded and hence compact subset of M_ν

$$M_{\nu,\mathbb{R}}^{0,1} := \big\{(\check{y}, v) \in M_{\nu,\mathbb{R}} \,:\, \check{y}_0 = 0, \check{y}_1 = 1\big\}.$$

If $(\check{y}, v) \in M_{\nu,\mathbb{R}}$, then $\big(\frac{\check{y} - \check{y}_0}{\check{y}_1 - \check{y}_0}, v\big) \in M_{\nu,\mathbb{R}}^{0,1}$ and due to the substitution rule and Lemma 4.10, (v), there holds with the notation $⨍_c^d(\cdot)(t)\,\mathrm{d}t = \int_c^d(\cdot)(t)\,\mathrm{d}t/(d-c)$

$$\begin{aligned} \widetilde{\Theta}_{k,\ell,\nu}(\check{y}, v) &= ⨍_{g_{k,\nu}(\check{y})_{\ell-1}}^{g_{k,\nu}(\check{y})_\ell} \big(1 - R(t|g_{k,\nu}(\check{y}), v)\big)\theta\,\mathrm{d}t \\ &= ⨍_{\frac{g_{k,\nu}(\check{y})_{\ell-1} - \check{y}_0}{\check{y}_1 - \check{y}_0}}^{\frac{g_{k,\nu}(\check{y})_\ell - \check{y}_0}{\check{y}_1 - \check{y}_0}} \big(1 - R\big(t(\check{y}_1 - \check{y}_0) + \check{y}_0|g_{k,\nu}(\check{y}), v\big)\big)\theta\,\mathrm{d}t \\ &= \widetilde{\Theta}_{k,\ell,\nu}\left(\frac{\check{y} - \check{y}_1}{\check{y}_2 - \check{y}_1}, v\right). \end{aligned}$$

Hence we have

$$\Theta_{k,\ell,\nu}(M_\nu) = \widetilde{\Theta}_{k,\ell,\nu}(M_{\nu,\mathbb{R}}^{0,1}).$$

As in (2) one sees that $\widetilde{\Theta}_{k,\ell,\nu}$ only attains values in $[0,1)$ and is continuous. Since $M_{\nu,\mathbb{R}}^{0,1}$ is compact we get

$$\sup\big(\Theta_{k,\ell,\nu}(M_\nu)\big) \leq \sup\big(\widetilde{\Theta}_{k,\ell,\nu}(M_{\nu,\mathbb{R}})\big) < 1.$$

This proves the lemma for $\theta = 1$.

(4) We prove the lemma for $\theta = c_1\chi_{(-\infty,T)}|_I + c_2\chi_{[T,\infty)}|_I$ with $c_1, c_2 > 0$ and $T \in I$. Again, we extend the function $\Theta_{k,\ell,\nu}$ to $M_{\nu,\mathbb{R}}$

$$\widetilde{\Theta}_{k,\ell,\nu} : M_{\nu,\mathbb{R}} \to \mathbb{R} :$$
$$(\check{y}, v) \mapsto \frac{\big\| \big(1 - R(\cdot|g_{k,\nu}(\check{y}), v)\big)\big(c_1\chi_{(-\infty,T)} + c_2\chi_{[T,\infty)}\big)\big\|_{L^1([g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell])}}{\|c_1\chi_{(-\infty,T)} + c_2\chi_{[T,\infty)}\|_{L^1([g_{k,\nu}(\check{y})_{\ell-1}, g_{k,\nu}(\check{y})_\ell])}}.$$

For the proof of the lemma, it is sufficient to show $\sup\big(\widetilde{\Theta}_{k,\ell,\nu}(M_{\nu,\mathbb{R}})\big) < 1$. Due to the substitution rule and Lemma 4.10, (v), we can assume without loss of generality that $T = 0$. Because of (3) it only remains to show that

$$\sup\big(\widetilde{\Theta}_{k,\ell,\nu}(\big\{(\check{y}, v) \in M_{\nu,\mathbb{R}} \,:\, \check{y}_0 \le 0 \le \check{y}_\nu\big\}\big) < 1.$$

As in (2), one verifies that $\widetilde{\Theta}_{k,\ell,\nu}$ only attains values in $[0, 1)$ and is continuous. Moreover, due to the substitution rule and Lemma 4.10 (v), we have for any $(\check{y}, v) \in \big\{(\check{y}, v) \in M_{\nu,\mathbb{R}} \,:\, \check{y}_0 \le 0 \le \check{y}_\nu\big\}$

$$\begin{aligned}
\widetilde{\Theta}_{k,\ell,\nu}(\check{y}, v) &= \frac{⨍_{g_{k,\nu}(\check{y})_{\ell-1}}^{g_{k,\nu}(\check{y})_\ell} \big(1 - R\big(t|g_{k,\nu}(\check{y}, v)\big)\big)\big(c_1\chi_{(-\infty,0)}(t) + c_2\chi_{[0,\infty)}(t)\big)\,\mathrm{d}t}{⨍_{g_{k,\nu}(\check{y})_{\ell-1}}^{g_{k,\nu}} c_1\chi_{(-\infty,0)}(t) + c_2\chi_{[0,\infty)}(t)\,\mathrm{d}t} \\
&= \frac{⨍_{\frac{g_{k,\nu}(\check{y})_{\ell-1}}{\check{y}_1 - \check{y}_0}}^{\frac{g_{k,\nu}(\check{y})_\ell}{\check{y}_1 - \check{y}_0}} \big(1 - R\big(t(\check{y}_1 - \check{y}_0)|g_{k,\nu}(\check{y}, v)\big)\big)\big(c_1\chi_{(-\infty,0)}(t) + c_2\chi_{[0,\infty)}(t)\big)\,\mathrm{d}t}{⨍_{\frac{g_{k,\nu}(\check{y})_{\ell-1}}{\check{y}_1 - \check{y}_0}}^{\frac{g_{k,\nu}(\check{y})_\ell}{\check{y}_1 - \check{y}_0}} c_1\chi_{(-\infty,0)}(t) + c_2\chi_{[0,\infty)}(t)\,\mathrm{d}t} \\
&= \widetilde{\Theta}_{k,\ell,\nu}\left(\frac{\check{y}}{\check{y}_1 - \check{y}_0}, v\right)
\end{aligned}$$

and hence

$$\begin{aligned}
&\widetilde{\Theta}_{k,\ell,\nu}(\big\{(\check{y}, v) \in M_{\nu,\mathbb{R}} \,:\, \check{y}_0 \le 0 \le \check{y}_\nu\big\}\big) \\
&\quad = \widetilde{\Theta}_{k,\ell,\nu}(\big\{(\check{y}, v) \in M_{\nu,\mathbb{R}} \,:\, \check{y}_1 - \check{y}_0 = 1, \check{y}_0 \le 0 \le \check{y}_\nu\big\}).
\end{aligned}$$

The second set is compact, since it is the image of a closed and bounded set under a continuous mapping. Therefore it attains a maximum smaller than one. This concludes the proof for $\theta = c_1\chi_{(-\infty,T)}|_I + c_2\chi_{[T,\infty)}|_I$.

(5) Finally, we are in the position to prove the assertion of the lemma for arbitrary functions θ with the desired properties. Let $\big((\check{y}^m, v^m)\big)_{m\in\mathbb{N}}$ be a sequence in M_ν such that the $\Theta_{k,\ell,\nu}$-values converge to $\sup(\Theta_{k,\ell,\nu}(M_\nu))$. Because of the boundedness of M_ν, we can assume convergence of the sequence, where the limit $(\check{y}^\infty, v^\infty)$ is in $\overline{M_\nu}$, i.e. $(\check{y}^\infty, v^\infty) \in M_\nu$ or $(\check{y}^\infty, v^\infty) \in I^\nu \times [w_{\min}, w_{\max}]^{2p+1}$ with $\check{y}_0^\infty = \cdots = \check{y}_\nu^\infty$. In the first case, we are done because of (4.18) and the continuity of $\Theta_{k,\ell,\nu}$. For the second case, we define

$$a_m := g_{k,\nu}(\check{y}^m, v^m)_{\ell-1}, b_m := g_{k,\nu}(\check{y}^m, v^m)_\ell \quad \text{and} \quad R_m := R(\cdot|\check{y}^m, v^m).$$

Note that $a_m < b_m$, and that the sequences $(a_m)_{m\in\mathbb{N}}$ and $(b_m)_{m\in\mathbb{N}}$ converge to the limit

$$T := \check{y}_0^\infty = \cdots = \check{y}_\nu^\infty \in I.$$

We consider two cases.

Case 1. If θ is continuous at the limit T, it is absolutely continuous on the interval $[a_m, b_m]$ for sufficiently big $m \in \mathbb{N}$. Hence we have for sufficiently big $m \in \mathbb{N}$

$$\begin{aligned}\Theta_{k,\ell,\nu}(\check{y}^m, v^m) &= \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\theta(t)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta(t)\,\mathrm{d}t} \\ &= \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\big(\theta(a_m) + \int_{a_m}^{t} \theta'(\tau)\,\mathrm{d}\tau\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \big(\theta(a_m) + \int_{a_m}^{t} \theta'(\tau)\,\mathrm{d}\tau\big)\,\mathrm{d}t} \\ &\leq \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\theta(a_m)\,\mathrm{d}t + (b_m - a_m)^2\|\theta'\|_{L^\infty(I)}}{(b_m - a_m)\theta(a_m) - (b_m - a_m)^2\|\theta'\|_{L^\infty(I)}}.\end{aligned}$$

The second summand converges to zero. We consider the first summand. For any $C > 1$, there holds for sufficiently big $m \in \mathbb{N}$

$$\begin{aligned}(b_m - a_m)\theta(a_m) - (b_m - a_m)^2\|\theta'\|_{L^\infty(I)} &\geq (b_m - a_m)\left(\theta(a_m) - \frac{1}{C}\inf_{t\in I}\theta(t)\right) \\ &\geq (b_m - a_m)\theta(a_m)\frac{C-1}{C}.\end{aligned}$$

Hence, (3) shows

$$\begin{aligned}\frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\theta(a_m)\,\mathrm{d}t}{(b_m - a_m)\theta(a_m) - (b_m - a_m)^2\|\theta'\|_{L^\infty(I)}} &\leq \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\theta(a_m)\,\mathrm{d}t}{(b_m - a_m)\theta(a_m)\frac{C-1}{C}} \\ &\leq \frac{C}{C-1}\Big(1 - q_{NURBS}\big(\kappa_{\max}, w_{\min}, w_{\max}, p, 1\big)\Big).\end{aligned}$$

Since C was arbitrary, this implies

$$\sup\big(\Theta_{k,\ell,\nu}(M_\nu)\big) \leq \Big(1 - q_{NURBS}\big(\kappa_{\max}, w_{\min}, w_{\max}, p, 1\big)\Big) < 1.$$

Case 2. If θ is not continuous at the limit T we proceed as follows. For sufficiently big $m \in \mathbb{N}$, θ is absolutely continuous on $[a_m, T]$ and on $[T, b_m]$. By considering suitable subsequences, we can assume that $a_m < b_m \leq T$, $T \leq a_m < b_m$ or $a_m \leq T \leq b_m$, each for all $m \in \mathbb{N}$. In the first two cases, we can proceed as in Case 1. In the third case, we argue similarly as in Case 1 to see, with the left-handed limit $\theta^\ell(T)$ and the right-handed

limit $\theta^r(T)$ for $m \in \mathbb{N}$ big enough

$$\begin{aligned}
\Theta_{k,\ell,\nu}(\check{y}^m, v^m) &= \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\theta(t)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta(t)\,\mathrm{d}t} \\
&= \frac{\int_{a_m}^{T} \big(1 - R_m(t)\big)\big(\theta^\ell(T) - \int_t^T \theta'(\tau)\,\mathrm{d}\tau\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta(t)\,\mathrm{d}t} \\
&\quad + \frac{\int_{t}^{b_m} \big(1 - R_m(t)\big)\big(\theta^r(T) + \int_T^t \theta'(\tau)\,\mathrm{d}\tau\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta(t)\,\mathrm{d}t} \\
&\leq \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\big(\theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t - 2(b_m - a_m)^2\|\theta'\|_{L^\infty(I)}} \\
&\quad + \frac{2(b_m - a_m)^2\|\theta'\|_{L^\infty(I)}}{\int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t - 2(b_m - a_m)^2\|\theta'\|_{L^\infty(I)}}.
\end{aligned}$$

Again, the second summand converges to zero, wherefore it remains to consider the first one. For any $C > 1$, there holds for sufficiently big $m \in \mathbb{N}$

$$\begin{aligned}
&\int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t - 2(b_m - a_m)^2\|\theta'\|_{L^\infty(I)} \\
&\quad \geq \int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t - (b_m - a_m)\frac{1}{C}\inf_{t\in I}\theta(t) \\
&\quad \geq \int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t\frac{C-1}{C}.
\end{aligned}$$

Hence, (4) shows

$$\begin{aligned}
&\frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\big(\theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t - 2(b_m - a_m)^2\|\theta'\|_{L^\infty(I)}} \\
&\quad \leq \frac{\int_{a_m}^{b_m} \big(1 - R_m(t)\big)\big(\theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\big)\,\mathrm{d}t}{\int_{a_m}^{b_m} \theta^\ell(T)\chi_{(-\infty,T)}(t) + \theta^r(T)\chi_{[T,\infty)}(t)\,\mathrm{d}t\frac{C-1}{C}} \\
&\quad \leq \frac{C}{C-1}\Big(1 - q_{NURBS}\big(\kappa_{\max}, w_{\min}, w_{\max}, p, \theta^\ell(T)\chi_{(-\infty,T)}|_I + \theta^r(T)\chi_{[T,\infty)}|_I\big)\Big)
\end{aligned}$$

Since C was arbitrary, this implies

$$\begin{aligned}
&\sup\big(\Theta_{k,\ell,\nu}(M_\nu)\big) \\
&\quad \leq \Big(1 - q_{NURBS}\big(\kappa_{\max}, w_{\min}, w_{\max}, p, \theta^\ell(T)\chi_{(-\infty,T)}|_I + \theta^r(T)\chi_{[T,\infty)}|_I\big)\Big) < 1,
\end{aligned}$$

which concludes the proof.

□

To create a link between NURBS and our problem $V\phi = F$, we have to extend Definition 2.9.

Definition 4.13. If $\Gamma_D = \Gamma$, let $\check{\mathcal{N}}_h = \{\check{x}_j : j = 1, \dots, n\}$ be a set of nodes on $(a, b]$ with $\check{x}_n = b$. To each node $\check{x} \in \check{\mathcal{N}}_h$, we assign a corresponding multiplicity $\#\check{x}$. This induces a sequence of non decreasing knots $\check{\mathcal{K}}_h = (t_i)_{i=1}^N$ on $(a, b]$. Let $\mathcal{W}_h = (w_i)_{i=1}^N$ be a sequence of weights on these knots. We extend the knot sequence $(b-a)$-periodically to $(t_i)_{i\in\mathbb{Z}}$ and the weight sequence to $(w_i)_{i\in\mathbb{Z}}$ by $w_{n+i} := w_i$ for $i \in \mathbb{Z}$. For the extended sequences we also write $\check{\mathcal{K}}_h$ and $\mathcal{W}_h$. We set

$$\hat{\mathscr{N}^p}(\check{\mathcal{K}}_h, \mathcal{W}_h) := \mathscr{N}^p(\check{\mathcal{K}}_h, \mathcal{W}_h)|_{[a,b)} \circ \gamma|^{-1}_{[a,b)}. \tag{4.19}$$

If $\Gamma_D \neq \Gamma$, let $\check{\mathcal{N}}_h = \{\check{x}_j : j = 0, \dots, n\}$ be a set of nodes on $[a_D, b_D]$ with $\check{x}_0 = a_D$ and $\check{x}_n = b_D$. To each node $\check{x} \in \check{\mathcal{N}}_h$ we assign a corresponding multiplicity $\#\check{x}$ such that $\#a_D = \#b_D = p + 1$. This induces a sequence of non decreasing knots $\check{\mathcal{K}}_h = (t_i)_{i=0}^N$ on $[a_D, b_D]$. Let $\mathcal{W}_h = (w_i)_{i=1}^{N-p}$ be a sequence of weights. We extend the sequences arbitrarily to $\check{\mathcal{K}}_h = (t_i)_{i\in\mathbb{Z}}$ with $a > t_i \to -\infty$ for $i < 0$ and $b < t_i \to \infty$ for $i > N$, and $\mathcal{W}_h = (w_i)_{i\in\mathbb{Z}}$ with $w_i > 0$. We define the space[8]

$$\hat{\mathscr{N}^p}(\check{\mathcal{K}}_h, \mathcal{W}_h) := \mathscr{N}^p(\check{\mathcal{K}}_h, \mathcal{W}_h)|_{[a_D,b_D]} \circ \gamma|^{-1}_{[a_D,b_D]}. \tag{4.20}$$

Due to Lemma 4.10 (ii) and (iv), this does not depend on how the sequences are extended.

With the following theorem we conclude that Theorem 2.11, Theorem 3.4, Theorem 3.5 and Theorem 3.8 hold for the span of transformed NURBS functions. To the best of our knowledge, this result is new. For her corresponding results of the first three mentioned theorems [Fae00, Therorem 2.2, Theorem 3.1, and Theorem 3.2], Faermann only considered transformed spline functions as trial space, see [Fae00, page 206]. She only allowed arc length parametrizations $\gamma : [0, L] \to \Gamma$ for the transformation of the splines onto the boundary, while our result is essentially independent of γ.

Theorem 4.14. *Let $p \in \mathbb{N}_0$ and $m := \lceil p/2 \rceil$. Then, the space $\hat{\mathscr{N}^p}(\check{\mathcal{K}}_h, \mathcal{W}_h)$ is a subspace of $L^2(\Gamma_D)$ which satisfies Assumption 2.10 with*

$$q_{space} = q_{NURBS}\big(\kappa(\check{\mathcal{T}}_h), \min(\mathcal{W}_h), \max(\mathcal{W}_h), p, \theta\big),$$

where $\theta = |\gamma|'_I|$ with $I = [a - (b-a)(m+p), b + (b-a)(2p-m)]$. Note that q_{space} depends in a monotonously decreasing way on $\kappa(\check{\mathcal{T}}_h)$and $\max(\mathcal{W}_h)$, and in a monotonously increasing way of $\min(\mathcal{W}_h)$.

Proof of Theorem 4.14 for $\Gamma_D = \Gamma$. Lemma 4.10, (i) and (ii), implies $\mathscr{N}^p(\check{\mathcal{K}}_h, \mathcal{W}_h) \leq L^2(\mathbb{R})$. Using Remark 2.5, we see

$$\mathscr{N}^p(\check{\mathcal{K}}_h, \mathcal{W}_h)|_{[a,b)} \circ \gamma|^{-1}_{[a,b)} \leq L^2(\Gamma_D).$$

Let T be an element of the mesh $\mathcal{T}_h$, $j \in \{1, \dots, n\}$ with $T = T_j$, and $i \in \{1, \dots, N\}$ with $\check{x}_{j-1} = t_{i-1}$ and $\check{x}_j = t_i$. We define $\check{\phi}_T(t) := R_{i-m,p}(t)$ for $t \in [a, b)$ and extend it continuously at b. We set $\phi_T := \check{\phi}_T|_{[a,b)} \circ \gamma|^{-1}_{[a,b)}$. Because of Lemma 4.10 (ii), there holds

$$\check{T}_j \subseteq [t_{i-m-1}, t_{i-m+p}] \cap [a, b] = \operatorname{supp}(\check{\phi}_T) \subseteq [\check{x}_{j-m-1}, \check{x}_{j-m+p}] \subseteq [\check{x}_{j-m-1}, \check{x}_{j+m}]\,. \tag{4.21}$$

[8]Lemma 4.10 (ii) and (vii) imply that the space always contains $\chi_{\{b_D\}}|_{[a_D,b_D]}$. Since we will consider it as a subset of $L^2(\Gamma_D)$, this is not problematic.

Since $\gamma|_{[a,a+(b-a)/2]}$ and $\gamma|_{[a+(b-a)/2,b]}$ are homeomorphisms, there holds

$$\begin{aligned}\gamma(\operatorname{supp}(\check{\phi}_T)) &= \gamma\left(\overline{\left\{t \in [a,b) \,:\, \check{\phi}_T(t) \neq 0\right\}}\right) \\ &= \overline{\left\{x \in \gamma([a,a+(b-a)/2]) \,:\, \phi(x) \neq 0\right\}} \cup \overline{\left\{x \in \gamma([a+(b-a)/2,b)) \,:\, \phi(x) \neq 0\right\}} \\ &= \operatorname{supp}(\phi_T),\end{aligned} \tag{4.22}$$

wherefore $\operatorname{supp}(\phi_T)$ is connected. Moreover, this shows with (4.21)

$$T \subseteq \operatorname{supp}(\phi_T) \subseteq \omega_h^m(T).$$

This implies Assumption 2.10, (i). For Assumption 2.10, (ii), we want to apply Lemma 4.12. Note that $R_{i-m,p}$ is completely determined by the knots in I and their weights. This is due to $I \supseteq [t_{i-m-p-1}, t_{i+2p-m}]$ and Lemma 4.10, (iv). The regularity constant of these knots as in (4.16), is obviously smaller or equal than $\kappa(\check{\mathcal{K}}_h)$. Since γ is piecewise two times continuously differentiable and its left and right derivative vanishes nowhere, $|\gamma'|$ is piecewise continuously differentiable and is bounded from above by some positive constant. Hence, we get with Remark 2.5, Lemma 4.12 and (4.22)

$$\begin{aligned}\|1-\phi_T\|^2_{L^2(\operatorname{supp}(\phi_T))} &= \int_{\operatorname{supp}(\check{\phi}_T)} (1-\check{\phi}_T)^2 |\gamma'(t)| \,\mathrm{d}\lambda(t) \leq \int_{\operatorname{supp}(\check{\phi}_T)} (1-\check{\phi}_T)|\gamma'(t)| \,\mathrm{d}\lambda(t) \\ &= \|(1-\check{\phi}_T)\theta\|_{L^1([t_{i-m-1},t_{i-m+p}]\cap[a,b])} \leq (1-q_{NURBS})\|\theta\|_{L^1([t_{i-m-1},t_{i-m+p}]\cap[a,b])} \\ &= (1-q_{NURBS}) \int_{\operatorname{supp}(\check{\phi}_T)} |\gamma'(t)| \,\mathrm{d}\lambda(t).\end{aligned}$$

The last term is due to Remark 2.3 just

$$(1-q_{NURBS})\mu_\Gamma\big(\operatorname{supp}(\phi_T)\big).$$

Hence Assumption 2.10, (ii), is also fulfilled. This concludes the proof. □

Proof of Theorem 4.14 for $\Gamma_D \neq \Gamma$. We extend the knots by $t_{-i} := t_0 - i(t_{p+1} - t_0)$, $t_{N+i} := t_N + i(t_N - t_{N-p-1})$ for $i \in \mathbb{N}$, and the weights by $w_{1-i} := w_0$, $w_{N-p+i} := w_{N-p}$. Defining $\check{\phi}_T(t) := R_{i-m,p}|_{[a_D,b_D]}$ and $\phi_T := \check{\phi}_T \circ \gamma|^{-1}_{[a_D,b_D]}$, the proof follows as for $\Gamma_D = \Gamma$. Equality (4.22) becomes

$$\gamma(\operatorname{supp}(\check{\phi}_T)) = \gamma\left(\overline{\left\{t \in [a_D,b_D] \,:\, \check{\phi}_T(t) \neq 0\right\}}\right) = \overline{\left\{x \in \Gamma_D \,:\, \phi_T(x) \neq 0\right\}} = \operatorname{supp}(\phi_T),$$

because $\gamma|_{[a_D,b_D]}$ is a homeomorphism. □

4.2. **Adaptive Algorithm.** In this subsection, we will present an adaptive algorithm to solve the problem $V\phi = F$ of Section 3. Let $0 < \vartheta \leq 1$ be an adaptivity parameter, $p \in \mathbb{N}_0$ a polynomial order, and $\kappa_{\max}$ a bound for the shape regularity constant. We start with some nodes $\check{\mathcal{N}}_{h_0} = \check{\mathcal{N}}_0 = \left\{\check{x}_j^{[0]} \,:\, j = 1,\dots,n_0\right\}$ on $(a,b]$ with $\check{x}_{n_0} = b$ resp. $\check{\mathcal{N}}_{h_0} = \check{\mathcal{N}}_0 = \left\{\check{x}_j^{[0]} \,:\, j = 0,\dots,n_0\right\}$ on $[a_D,b_D]$ with $\check{x}_0 = a_D$ and $\check{x}_{n_0} = b_D$. Each node has a multiplicity lower or equal $p+1$, where for $\Gamma \neq \Gamma_D$ we assume $\#a_D = \#b_D = p+1$. This induces knots $\check{\mathcal{K}}_{h_0} = \check{\mathcal{K}}_0 = (t_i^{[0]})_{i=1}^{N_0}$ resp. $\check{\mathcal{K}}_{h_0} = \check{\mathcal{K}}_0 = (t_i^{[0]})_{i=0}^{N_0}$. Let $\mathcal{W}_{h_0} = \mathcal{W}_0 = (w_i^{[0]})_{i=1}^{N_0}$ resp. $\mathcal{W}_{h_0} = \mathcal{W}_0 = (w_i^{[0]})_{i=1}^{N_0-p}$ be some positive start weights. We extend the knots and weights as

in Definition 4.13 and use the same notation for the extended sequences. We denote $\check{\mathcal{T}}_{h_0} = \check{\mathcal{T}}_0$ and $\mathcal{T}_{h_0} = \mathcal{T}_0$ for the 0-th mesh on the parameter domain resp. on the boundary. For this initial mesh, we make the following assumptions

$$\begin{aligned} \kappa(\check{\mathcal{T}}_0) &\le \kappa_{\max}, \\ h_0 &\le \mu_\Gamma(\Gamma)/4, \\ n_0 &\ge 4, \\ p+1 &\le N_0. \end{aligned} \tag{4.23}$$

As the initial trial space, we consider

$$S(\mathcal{T}_0) := \hat{\mathcal{N}}^p(\check{\mathcal{K}}_0, \mathcal{W}_0) \le L^2(\Gamma_D) \le \widetilde{H}^{-1/2}(\Gamma_D). \tag{4.24}$$

The start approximation of ϕ is ϕ_0 with

$$\forall \psi_0 \in S(\mathcal{T}_0): \quad \langle V\phi_0, \psi_0\rangle_{L^2(\Gamma_D)} = \langle F, \psi_0\rangle_{L^2(\Gamma_D)}$$

To monitor the local error, we apply one of the local element based error estimators $\eta_{h_0} = \eta_0$ of Section 3, see Theorem 3.4, Theorem 3.5, Theorem 3.8 and Remark 3.9. In the k-th step of the algorithm, we use analogous notation. The adaptive algorithm with *Dörfler marking* reads as follows:

Algorithm 4.15. INPUT: *Initial knots $\check{\mathcal{K}}_0$ and weights $\mathcal{W}_0$, polynomial degree $p \in \mathbb{N}_0$, adaptivity parameter $0 < \vartheta \le 1$, bound $\kappa_{\max}$ for the shape regularity constants.*
Loop: *For $k = 0, 1, 2, \dots$ do* (i) – (iv)

(i) *Compute discrete approximation ϕ_k.*
(ii) *Compute refinement indicators $\eta_k(T)$ for all $T \in \mathcal{T}_k$.*
(iii) *For $0 < \vartheta < 1$, determine set (of minimal cardinaltiy) $\mathcal{M}_k \subseteq \mathcal{T}_k$ such that*

$$\vartheta \sum_{T\in\mathcal{T}_k} \eta_k(T)^2 \le \sum_{T\in\mathcal{M}_k} \eta_k(T)^2 \tag{4.25}$$

If $\vartheta = 1$, we choose $\mathcal{M}_k = \mathcal{T}_k$.
(iv) *Refine at least the marked elements $T \in \mathcal{M}_k$ by knot insertion (see below) to obtain $\check{\mathcal{K}}_{k+1}$ and $\mathcal{W}_{k+1}$ such that*

$$\sum_{i\in\mathbb{Z}} w_i^{[k]} B_{i,p}^{\check{\mathcal{K}}_k} = \sum_{i\in\mathbb{Z}} w_i^{[k+1]} B_{i,p}^{\check{\mathcal{K}}_{k+1}} \tag{4.26}$$

and $\kappa(\check{\mathcal{T}}_{k+1}) \le \kappa_{\max}$.

OUTPUT: *Approximate solutions ϕ_k and error estimator η_k for all $k \in \mathbb{N}$.*

We explain how knot insertion in (iv) works:

With $\mathcal{M}_k \subseteq \mathcal{T}_k$ the set of marked elements, let $\mathcal{M}_k \subseteq \mathcal{R}_k \subseteq \mathcal{T}_k$ be the set of all elements which are refined, i.e., $\mathcal{R}_k = \mathcal{T}_k \setminus \mathcal{T}_{k+1}$, and $\check{\mathcal{R}}_k$ be the corresponding elements in $\check{\mathcal{T}}_k$. We add $\{t : t \text{ midpoint of some } \check{T} \in \check{\mathcal{R}}_k\}$, each midpoint assigned with multiplicity one, to the knots $\check{\mathcal{K}}_k = (t_i^{[k]})_{i=1}^{N_k}$ resp. $\check{\mathcal{K}}_k = (t_i^{[k]})_{i=0}^{N_k}$ and obtain $\check{\mathcal{K}}_{k+1} = (t_i^{[k]})_{i=1}^{N_{k+1}}$ resp. $\check{\mathcal{K}}_{k+1} = (t_i^{[k]})_{i=0}^{N_{k+1}}$. We extend $\check{\mathcal{K}}_{k+1}$ as in Definition 4.13. Due to Remark 4.5 and Corollary 4.6, there exist unique weights $(w_i^{[k+1]})_{i\in\mathbb{Z}}$ with

$$\sum_{i\in\mathbb{Z}} w_i^{[k]} B_{i,p}^{\check{\mathcal{K}}_k} = \sum_{i\in\mathbb{Z}} w_i^{[k+1]} B_{i,p}^{\check{\mathcal{K}}_{k+1}}.$$

We have to show that the new weights are again of the form as in Definition 4.13. If $\Gamma = \Gamma_D$ they are indeed periodic thanks to Lemma 4.10 (v) and their uniqueness. For $\Gamma \neq \Gamma_D$, there is nothing to show. Corollary 4.6, and Equation (4.15) and (4.26) imply nestedness of the spaces, i.e., for $k \in \mathbb{N}_0$ holds

$$S(\mathcal{T}_k) \leq S(\mathcal{T}_{k+1}). \tag{4.27}$$

The new weights can even be calculated explicitly. Without loss of generality, we assume $\mathcal{R}_k = \{\gamma([t_{\ell-1}, t_\ell])\}$ for some $\ell \in \{1, \dots, N_k\}$ and denote $t' = (t_{\ell-1}^{[k]} + t_\ell^{[k]})/2$. For $|\mathcal{R}_k| > 1$, one just has to repetitively apply the following procedure.

Case $\boldsymbol{\Gamma \neq \Gamma_D}$. Now we even have $\ell \in \{p+1, \dots, N_k - p\}$. We add t' to $(t_i^{[k]})_{i\in\mathbb{Z}}$ to get $(t_i^{[k+1]})_{i\in\mathbb{Z}}$ with $t_\ell^{[k+1]} = t'$. Applying Lemma 4.7 shows for $i = 1, \dots, N_{k+1} - p$

$$w_i^{[k+1]} = \begin{cases} w_i^{[k]} & \text{if } i \leq \ell - p, \\ (1 - \beta_{i-1,p}^{\check{\mathcal{K}}_k}(t'))w_{i-1}^{[k]} + \beta_{i-1,p}^{\check{\mathcal{K}}_k} w_i^{[k]} & \text{if } \ell + 1 - p \leq i \leq \ell, \\ w_{i-1}^{[k]} & \text{if } \ell + 1 \leq i. \end{cases}$$

From this formula it is not hard to check that $(w_i^{[k+1]})_{i=1}^{N_{k+1}-p}$ depends only on $(t_i^{[k]})_{i=1}^{N_k}$ and on $(w_i^{[k]})_{i=1}^{N_k-p}$, but not on their (arbitrary) extensions. The new weights are even convex combinations of the old ones.

Case $\boldsymbol{\Gamma = \Gamma_D}$. We cannot directly apply Lemma 4.7, because theoretically we have to add infinitely many knots. First, we add $\{t' + (b-a)q : q \in \mathbb{Z} \setminus \{-1, 0, 1\}\}$ to the knots $\check{\mathcal{K}}_k$ and obtain $\check{\mathcal{K}}' = (t'_i)_{i\in\mathbb{Z}}$ with $t'_0 = t_0^{[k]}$, $t'_1 = t_1^{[k]}$. There exist unique weights $\mathcal{W}' = (w'_i)_{i\in\mathbb{Z}}$ with

$$\sum_{i\in\mathbb{Z}} w_i^{[k]} B_{i,p}^{\check{\mathcal{K}}_k} = \sum_{i\in\mathbb{Z}} w'_i B_{i,p}^{\check{\mathcal{K}}'}.$$

With $I := [t_{-1}^{[k]}, t_{N_k+1}^{[k]})$ there holds because of Lemma 4.2 (ii) and (iv), and our assumption $p + 1 \leq N_0 \leq N_k$

$$\sum_{i=-p}^{N_k+1} w_i^{[k]} B_{i,p}^{\check{\mathcal{K}}_k}|_I = \sum_{i=-p}^{N_k+1} w'_i B_{i,p}^{\check{\mathcal{K}}'}|_I = \sum_{i=-p}^{N_k+1} w'_i B_{i,p}^{\check{\mathcal{K}}_k}|_I.$$

With $t_{N_k}^{[k]} < t_{N_k+1}^{[k]}$, it is not hard to check that $B_{i,p}^{\check{\mathcal{K}}_k}|_I \neq 0$ for $i = 0, \dots, N_k$. Hence, Theorem 4.4 implies $w_i^{[k]} = w'_i$ for $i = 0, \dots, N_k$. It just remains to add the knots $t' - (b-a)$, t' and $t' + (b-a)$. To this end, we can apply Lemma 4.7. We start adding $t' - (b-a)$ to $\check{\mathcal{K}}'$ and obtain $\check{\mathcal{K}}'' = (t''_i)_{i\in\mathbb{Z}}$ with $t''_0 = t_0^{[k]}$, $t''_1 = t_1^{[k]}$. There exist unique weights $\mathcal{W}''$ with

$$\sum_{i\in\mathbb{Z}} w'_i B_{i,p}^{\check{\mathcal{K}}'} = \sum_{i\in\mathbb{Z}} w''_i B_{i,p}^{\check{\mathcal{K}}''}.$$

Because of Lemma 4.7, we have $w_i^{[k]} = w'_i = w''_i$ for $i = 0, \dots, N_k$. Now, we add the knot t' to $\check{\mathcal{K}}''$ and obtain $\check{\mathcal{K}}''' = (t'''_i)_{i\in\mathbb{Z}}$ with $t'''_\ell = t'$. There exist unique weights with

$$\sum_{i\in\mathbb{Z}} w''_i B_{i,p}^{\check{\mathcal{K}}''} = \sum_{i\in\mathbb{Z}} w'''_i B_{i,p}^{\check{\mathcal{K}}'''}.$$

Lemma 4.7 implies for $i = 1, \dots, N_k + 1$

$$w_i''' = \begin{cases} w_i^{[k]} & \text{if } i \le \ell - p, \\ (1 - \beta_{i-1,p}^{\check{\mathcal{K}}''}(t'))w_{i-1}^{[k]} + \beta_{i-1,p}^{\check{\mathcal{K}}''} w_i^{[k]} & \text{if } \ell + 1 - p \le i \le \ell, \\ w_{i-1}^{[k]} & \text{if } \ell + 1 \le i. \end{cases}$$

We even have $\beta_{i-1,p}^{\check{\mathcal{K}}''} = \beta_{i-1,p}^{\check{\mathcal{K}}_k}$. Finally, we add $t' + (b-a) = (t'''_{\ell+N_k} + t'''_{\ell+N_k+1})/2$ and end up with $\check{\mathcal{K}}_{k+1} = (t_i^{[k+1]})_{i\in\mathbb{Z}}$ with $t_\ell^{[k+1]} = t'$. The weights $w_i^{[k+1]}$ of

$$\sum_{i\in\mathbb{Z}} w_i''' B_{i,p}^{\check{\mathcal{K}}'''} = \sum_{i\in\mathbb{Z}} w_i^{[k+1]} B_{i,p}^{\check{\mathcal{K}}^{[k+1]}},$$

have for $i = 1, \dots, N_{k+1} = N_k + 1$ the following form

$$w_i^{[k+1]} = \begin{cases} w_i''' & \text{if } i \le \ell + N_{k+1} - p, \\ \big(1 - \beta_{i-1,p}^{\check{\mathcal{K}}'''}(t' + (b-a))\big)w_{i-1}''' + \beta_{i-1,p}^{\check{\mathcal{K}}'''} w_i''' & \text{if } \ell + N_{k+1} + 1 - p \le i. \end{cases}$$

Again the new weights are just convex combinations of the old ones.

In practice, one can determine a suitable set $\mathcal{R}_k \supseteq \mathcal{M}_k$, such that the shape regularity constant stays bounded by $\kappa_{\max}$, as follows. Whenever an element $T \in \mathcal{T}_k$ is refined, we check if $h_{\tilde{T}'}/(h_{\tilde{T}}/2) > \kappa_{\max}$ for those elements T' in the current mesh with $T \cap T' \neq \emptyset$. Due to $\kappa(\check{\mathcal{T}}_k) \le \kappa_{\max}$, this can only happen if $T' \in \mathcal{T}_k$. In this case, we refine T' as well, whether or not it is marked. Since $\mathcal{T}_k$ contains only finitely many elements, this procedure terminates and at most all elements in $\mathcal{T}_k$ are refined. Indeed, this procedure delivers the minimal set $\mathcal{R}_k$ with the desired properties.

The following theorem shows that convergence of the error estimator

$$\sum_{T\in\mathcal{T}_k} \eta_k(T)^2 \to 0$$

already implies convergence of our approximations ϕ_k to the exact solution ϕ. If the used error estimator is even efficient, both convergences are equivalent.

Theorem 4.16. *There holds reliability*

$$\|\phi - \phi_k\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)} \le C_{rel} \sum_{T\in\mathcal{T}_k} \eta_k(T)^2. \tag{4.28}$$

We abbreviate the constant of Theorem 4.14,

$$q_{space} = q_{NURBS}(\kappa_{\max}, \min(\mathcal{W}_0), \max(\mathcal{W}_0), p, |\gamma|'_I|),$$

where $I = [a - (b-a)(\lceil p/2\rceil + p), b + (b-a)(2p - \lceil p/2\rceil)]$, *and set*

$$K_{\max} := \kappa_{\max} \frac{\sup_{t\in[a,b]} |\gamma'(t)|}{\inf_{t\in[a,b]} |\gamma'(t)|}.$$

We have $C_{rel} = C_{rel}^{F_1}(h_0, K_{\max}, \lceil p/2\rceil, q_{space})$, $C_{rel} = C_{rel}^{F_2}$, *or* $C_{rel} = C_{rel}^{R}(h_0, K_{\max}, \lceil p/2\rceil, q_{space})$ *resp.* $C_{rel} = 2(K_{\max}+1)C_{rel}^{R}(h_0, K_{\max}, \lceil p/2\rceil, q_{space})$ *with the constants of Theorem 3.4, Theorem 3.5, and Theorem 3.8. If we use the element based estimator of Theorem 3.4 or Theorem 3.5, there even holds efficiency*

$$C_{eff} \sum_{T\in\mathcal{T}_k} \eta_k(T)^2 \le \|\phi - \phi_k\|^2_{\widetilde{H}^{-1/2}(\Gamma_D)}, \tag{4.29}$$

where $C_{eff} = C_{eff}^{F_1}/2$ *or* $C_{eff} = C_{eff}^{F_2}(h_0, K_{\max}, \lceil p/2 \rceil, q_{space})/2$ *with the constants of the corresponding theorems.*

Proof. Note that the weights $w \in \mathcal{W}_k$ of each step are just convex combinations of $\mathcal{W}_0$, wherefore there holds for any $k \in \mathbb{N}_0$

$$\min(\mathcal{W}_0) \leq \min(\mathcal{W}_k) \leq \max(\mathcal{W}_k) \leq \max(\mathcal{W}_0).$$

Equation (2.9) shows

$$\kappa(\mathcal{T}_k) \leq K_{\max}.$$

Due to Theorem 4.14, our approximation spaces $S(\mathcal{T}_k)$ satisfy Assumption 2.10. The theorem immediately follows from Theorem 3.4, Theorem 3.5, Theorem 3.8, and Remark 3.9. □

Because of (4.27) and the definition of the approximations ϕ_k of ϕ, there holds the orthogonality

$$\forall k \in \mathbb{N}_0 : \quad \langle V(\phi_{k+1} - \phi_k), \phi_k \rangle = 0$$

and hence

$$\forall k \in \mathbb{N}_0 : \quad \|\phi_{k+1} - \phi_k\|_V^2 = \|\phi_{k+1}\|_V^2 - \|\phi_k\|_V^2 \tag{4.30}$$

with the energy norm $\|\cdot\|_V^2 := \langle V(\cdot), \cdot \rangle$ on $\widetilde{H}^{-1/2}(\Gamma_D)$. Note that the properties of V imply equivalence of the energy norm and $\|\cdot\|_{\widetilde{H}^{-1/2}(\Gamma_D)}$. The same argumentation shows

$$\forall k \in \mathbb{N}_0 : \quad \|\phi - \phi_k\|_V^2 = \|\phi\|_V^2 - \|\phi_k\|_V^2. \tag{4.31}$$

Equation (4.30) and (4.31) show that the error in the energy norm is always monotonously decreasing. However, a mathematical convergence proof for the proposed adaptive strategy (Algorithm 4.15) remains open. Nevertheless, many numerical examples confirm the convergence conjecture. One can even observe certain convergence rates. Some of these examples are found in Section 6.

5. Numerical Solution of Symm's Integral Equation

Let $\Omega \subset \mathbb{R}^2$ be an open set as in Section 2. This time, we assume that the parametrization γ is even two times piecewise differentiable and consider $\Gamma_D = \Gamma$. Let $a = \check{x}_0^\gamma < \cdots < \check{x}_{n_\gamma}^\gamma = b$ such that $\gamma|_{[\check{x}_{j-1}^\gamma, \check{x}_j^\gamma]}$ is two times continuously differentiable for $j = 1, \ldots, n_\gamma$. We denote $\gamma(\check{x}_j^\gamma) = x_j^\gamma$. We additionally require that Ω is a Lipschitz domain with $\operatorname{diam}(\Omega) < 1$; see [Ste08, Definition 2.1]. We assume that γ is positively orientated, i.e. the outer normal ν at any point $x = \gamma(s) \in \Gamma \setminus \{x_1^\gamma, \ldots, x_n^\gamma\}$ satisfies

$$\nu(x) = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \frac{\gamma'(s)}{|\gamma'(s)|}.$$

We will use the notation $\gamma'(s)^\perp := \nu(\gamma(s))|\gamma'(s)|$. For $x \neq y \in \Gamma \setminus \{x_1^\gamma \ldots, x_{n_\gamma}^\gamma\}$ we define the *fundamental solution* on the boundary

$$G(x, y) := -\frac{1}{2\pi} \ln(|x - y|)$$

and its normal derivative with respect to y

$$\partial_\nu G(x, y) := \partial_{\nu(y)} G(x, y) = \frac{1}{2\pi} \frac{\langle x - y, \nu(y) \rangle}{|x - y|^2}.$$

In this section, we consider for given $g \in H^{1/2}(\Gamma)$ Symm's integral equation of Section 1 without volume force

$$V\phi = \left(K + \frac{1}{2} I \right) g, \tag{5.1}$$

with the continuous and linear *single-layer operator*[9]

$$V : H^{-1/2}(\Gamma) \to H^{1/2}(\Gamma) : \psi \mapsto \left(x \mapsto \int_\Gamma G(x, y) \psi(y) \, \mathrm{d}\mu_\Gamma(y) \right), \tag{5.2}$$

the continuous and linear *double-layer operator*

$$K : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma) : f \mapsto \left(x \mapsto \int_\Gamma \partial_\nu G(x, y) f(y) \, \mathrm{d}\mu_\Gamma(y) \right), \tag{5.3}$$

and the identity $I : H^{1/2}(\Gamma) \to H^{1/2}(\Gamma)$, see [Ste08, page 119 and 125].

Note that for $\psi \in L^2(\Gamma)$ and $f \in H^{1/2}(\Gamma)$ the integrals in (5.2) and in (5.3) exist for almost every $x \in \Gamma$ in the measure theoretical sense, i.e., the moduli of the integrands are integrable. To see this, fix $x = \gamma(s) \in \Gamma \setminus \{x_1^\gamma, \ldots, x_{n_\gamma}^\gamma\}$. For the single-layer operator, we

[9]The definition of V is a little bit sloppy. To be precise one should first define it on $L^2(\Gamma)$. Then it can be continuously extended to $H^{-1/2}(\Gamma)$.

have

$$\begin{aligned}
&\int_\Gamma |G(x,y)\psi(y)|\,\mathrm{d}\mu_\Gamma(y)\\
&\quad\le \int_\Gamma G(x,y)^2\,\mathrm{d}\mu_\Gamma(y)\int_\Gamma \psi(y)^2\,\mathrm{d}\mu_\Gamma(y)\\
&\quad= \int_{[a,b]} G(x,\gamma(t))^2|\gamma'(t)|\,\mathrm{d}\lambda(t)\cdot\|\psi\|^2_{L^2(\Gamma)}\\
&\quad= \frac{1}{4\pi^2}\int_{[a,b]}\left(\ln\left(\frac{|\gamma(s)-\gamma(t)|}{|s-t|}\right)+\ln(|s-t|)\right)^2|\gamma'(t)|\,\mathrm{d}\lambda(t)\cdot\|\psi\|^2_{L^2(\Gamma)}.
\end{aligned}$$

The term $\frac{|\gamma(s)-\gamma(t)|}{|s-t|}$ can be continuously extended by $|\gamma'(s)|$ if $s=t$. It is positive for $t\in[a,b]$, wherefore $\ln(|\gamma(s)-\gamma(t)|/|s-t|)$ is bounded. Elementary computations show that $t\mapsto\ln(|s-t|)^2$ is integrable. For the double-layer operator we have

$$\int_\Gamma |\partial_\nu G(x,y)f(y)|\,\mathrm{d}\mu_\Gamma(y)\le\frac{1}{4\pi^2}\int_{[a,b]}\frac{\langle\gamma(s)-\gamma(t),\gamma'(t)^\perp\rangle^2}{|\gamma(s)-\gamma(t)|^4}\frac{1}{|\gamma'(t)|}\,\mathrm{d}\lambda(t)\cdot\|f\|^2_{L^2(\Gamma)}.$$

At $s=t$ the integrand can be continuously extended by $\frac{\langle\gamma''(s),\gamma'(s)^\perp\rangle^2}{4|\gamma'(s)|^5}$ because for $t\neq s$ sufficiently close to s, the path γ is two times continuously differentiable on $[\min(s,t),\max(s,t)]$, which implies

$$\begin{aligned}
\frac{\langle\gamma(s)-\gamma(t),\gamma'(t)^\perp\rangle}{|\gamma(s)-\gamma(t)|^2}&=\frac{\left\langle\gamma'(t)(s-t)+\int_t^s\gamma''(\tau)(s-\tau)\,\mathrm{d}\tau,\gamma'(t)^\perp\right\rangle}{|\gamma(s)-\gamma(t)|^2}\\
&=\frac{\left\langle(s-t)\int_0^1\gamma''(t+(s-t)\tau)(s-t)(1-\tau)\,\mathrm{d}\tau,\gamma'(t)^\perp\right\rangle}{|\gamma(s)-\gamma(t)|^2}\\
&=\left\langle\int_0^1\gamma''(t+(s-t)\tau)(1-\tau)\,\mathrm{d}\tau,\gamma'(t)^\perp\right\rangle\frac{|s-t|^2}{|\gamma(s)-\gamma(t)|^2}.
\end{aligned}\tag{5.4}$$

In [Ste08, Theorem 6.23] it is shown that $(\phi,\psi)\mapsto\langle V\phi,\psi\rangle$ defines a symmetric elliptic bilinear form on $H^{-1/2}(\Gamma)$. Hence V is an operator as in Section 3. We set $F:=Kg+g/2$.

In this section we want to study step (i) and (ii) of Algorithm 4.15. Let $p\in\mathbb{N}_0$ and $\check{\mathcal{N}}_h=\{\check{x}_j:j=1,\dots,n\}\supseteq\{\check{x}_j^\gamma:j=1,\dots,n_\gamma\}$ a set of nodes[10] on $(a,b]$ with $\check{x}_n=b$. To each node $\check{x}\in\check{\mathcal{N}}_h$ we assign a corresponding multiplicity $\#\check{x}\le p+1$. This induces a sequence of non decreasing knots $\check{\mathcal{K}}_h=(t_i)_{i=1}^N$ on $(a,b]$. Let $\mathcal{W}_h=(w_i)_{i=1}^N$ be a sequence of positive weights on these knots. As in Definition 4.13, we periodically extend the knot sequence to $\check{\mathcal{K}}_h=(t_i)_{i\in\mathbb{Z}}$ and the weight sequence to $\mathcal{W}_h=(w_i)_{i\in\mathbb{Z}}$. We use $\hat{\mathcal{N}}^p(\check{\mathcal{K}}_h,\mathcal{W}_h)$ as approximation space $S(\mathcal{T}_h)$, and want to compute the unique solution of

$$\forall\psi_h\in S(\mathcal{T}_h):\quad\langle V\phi_h,\psi_h\rangle_{L^2(\Gamma)}=\langle F,\psi_h\rangle_{L^2(\Gamma)}.\tag{5.5}$$

Since we assumed that the multiplicity of each knot is lower equal $p+1$, we see with Lemma 4.2 (ii) and Corollary 4.4 that

$$\left\{R_{i,p}|_{[a,b)}:i=(1-p),\dots,N-\#b+1\right\}\circ\gamma|^{-1}_{[a,b)}\tag{5.6}$$

[10]Note the difference between the knots $\check{x}_j$ of the currently considered mesh and the break points $\hat{x}_j^\gamma$ of the fixed parametrization γ.

is a basis of $S(\mathcal{T}_h)$. We abbreviate $R_i := R_{i,p}|_{[a,b)}$ and $\hat{R}_{i,p} := R_i \circ \gamma|^{-1}_{[a,b)}$. There exist unique coefficients $c_h = \big(c_{h,1-p}, \dots, c_{h,N-\#b+1}\big)$ with $\phi_h = \sum_{i=1-p}^{N-\#b+1} c_{h,i}\hat{R}_i$. With the real symmetric positive definite matrix

$$V_h := \Big(\big\langle V\hat{R}_j, \hat{R}_i\big\rangle_{L^2(\Gamma)}\Big)_{j,i=1-p}^{N-\#b+1} \tag{5.7}$$

and the right-hand side vector

$$F_h := \Big(\big\langle F, \hat{R}_i\big\rangle_{L^2(\Gamma)}\Big)_{i=1-p}^{N-\#b+1} \tag{5.8}$$

there holds

$$V_h c_h = F_h. \tag{5.9}$$

To calculate ϕ_h, we just have to solve this system of linear equations. In the following subsections, we will simplify the entries of the matrix V_h and the vector F_h such that we can use *tensor-Gauss quadrature* to numerically approximate the appearing integrals. Moreover, we will approximate the Faermann estimator $\eta_h^{F_1}$ of Theorem 3.4. For this, we also have to numerically evaluate the residual $r_h := F - V\phi_h$.

For fixed positive weight functions $\theta_1, \theta_2 \in L^1([0,1])$ and $n_1, n_2 \in \mathbb{N}$, we define the tensor-Gauss quadrature as

$$Q_{n_1,n_2} : C\big([0,1]^2\big) \to \mathbb{R} : f \mapsto \sum_{q_1=1}^{n_1}\sum_{q_2=1}^{n_2} f(\xi_{1,q_1}, \xi_{2,q_2})\omega_{1,q_1}\omega_{2,q_2}, \tag{5.10}$$

where ξ_{1,q_1} resp. ξ_{2,q_2} and ω_{1,q_1} resp. ω_{2,q_2} are the nodes and weights for the Gaussian quadrature on $[0,1]$ with weight function θ_1 resp. θ_2, see e.g. [Pra06, Chapter 6.3]. We denote

$$Q : C\big([0,1]^2\big) \to \mathbb{R} : f \mapsto \int_{[0,1]}\int_{[0,1]} f(s,t)\theta_1(s)\theta_2(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \tag{5.11}$$

and

$$E_{n_1,n_2} := Q - Q_{n_1,n_2}. \tag{5.12}$$

For $\ell = 1, 2$ we denote the linear functionals on $C([0,1])$

$$\begin{aligned} Q^\ell_{n_\ell} &= \sum_{q_\ell=1}^{n_\ell} (\cdot)(\xi_{\ell,q_\ell})\omega_{\ell,q_\ell}, \\ Q^\ell &= \int_{[0,1]} (\cdot)(r)\theta_\ell(r)\,\mathrm{d}\lambda(r), \\ E^\ell_{n_\ell} &= Q^\ell - Q^\ell_{n_\ell}. \end{aligned}$$

We prove an estimate for the quadrature error and show convergence.

Theorem 5.1. *There holds the error estimate*

$$|E_{n_1,n_2}(f)| \le \|\theta_2\|_{L^1([0,1])}\| \max_{s\in[0,1]} |E^1_{n_1} f(s,\cdot)| + \|\theta_1\|_{L^1([0,1])}\| \max_{t\in[0,1]} |E^2_{n_2} f(\cdot,t)| \tag{5.13}$$

for arbitrary $f \in C\big([0,1]^2\big)$, where the right-hand side converges to zero for $n_1 \to \infty$ and $n_2 \to \infty$.

Proof. The proof for the estimate works analogously as in [SS11, Theorem 5.3.15]. We only prove convergence of the right-hand side. It suffices to consider the first summand, the argumentation for the second one is analogous. Let $s_{n_1} \in [0,1]$ such that

$$|E^1_{n_1} f(s_{n_1}, \cdot)| = \max_{s\in[0,1]} |E^1_{n_1} f(s, \cdot)|.$$

Let $(|E^1_{n_1(k)} f(s_{n_1(k)}, \cdot)|)_{k\in\mathbb{N}}$ be a subsequence. Due to the compactness of $[0,1]$ there exists a subsequence $\big(s_{n_1(k(j))}\big)_{j\in\mathbb{N}}$ of $\big(s_{n_1(k)}\big)_{k\in\mathbb{N}}$which converges to some limit $s_\infty \in [0,1]$. We have

$$|E^1_{n_1(k(j))} f(s_{n_1(k(j))}, \cdot)| \le |E^1_{n_1(k(j))} f(s_{n_1(k(j))}, \cdot) - E^1_{n_1(k(j))} f(s_\infty, \cdot)| + |E^1_{n_1(k(j))} f(s_\infty, \cdot)|.$$

The second summand is the quadrature error of a standard Gauss quadrature on $[0,1]$ with weight function. It is well known that this error converges to zero, see e.g. [Pra06, Kapitel 6.3]. It remains to estimate the first summand

$$\begin{aligned}
&|E^1_{n_1(k(j))} f(s_{n_1(k(j))}, \cdot) - E^1_{n_1(k(j))} f(s_\infty, \cdot)| \\
&\quad \le |Q^1 f(s_{n_1(k(j))}, \cdot) - Q^1 f(s_\infty, \cdot)| + |Q^1_{n_1(k(j))} f(s_{n_1(k(j))}, \cdot) - Q^1_{n_1(k(j))} f(s_\infty, \cdot)| \\
&\quad \le 2\|\theta_1\|_{L^1([0,1])} \|f(s_{n_1(k(j))}, \cdot) - f(s_\infty, \cdot)\|_{L^\infty([0,1])}.
\end{aligned}$$

The last term converges to zero, since the mapping $s \mapsto f(s, \cdot)$ from $[0,1]$ to $C([0,1])$ is continuous. We have thus shown that each subsequence of $(|E^1_{n_1} f(s_{n_1}, \cdot)|)_{n_1\in\mathbb{N}}$ has a subsequence converging to zero. This concludes the proof. □

5.1. **Numerical Approximation of V_h.** In this subsection we use the following abbreviations:

$$\begin{aligned}
&\check{G}(s,t) := G\big(\gamma(s), \gamma(t)\big) \\
&H_\ell := t_\ell - t_{\ell-1} \\
&\check{G}_{\ell_1,\ell_2}(s,t) := \check{G}(t_{\ell_1-1} + H_{\ell_1} s\,,\, t_{\ell_2-1} + H_{\ell_2} t) \\
&\widetilde{R}_q(s) := R_q(s)|\gamma'(s)| \\
&\widetilde{R}_{q,\ell}(s) := \widetilde{R}_q(t_{\ell-1} + H_\ell s)
\end{aligned}$$

There holds for $i, j = (1-p), \dots, N - \#b + 1$

$$\begin{aligned}
\big\langle V\hat{R}_i, \hat{R}_j\big\rangle_{L^2(\Gamma)} &= \int_\Gamma \int_\Gamma G(x,y)\hat{R}_i(x)\hat{R}_j(y)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(x) \\
&= \int_{[a,b]} \int_{[a,b]} G\big(\gamma(s),\gamma(t)\big) R_i(s) R_j(t) |\gamma'(s)||\gamma'(t)|\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \\
&= \sum_{\ell_1=\max(i,1)}^{\min(i+p,N)} \sum_{\ell_2=\max(j,1)}^{\min(j+p,N)} \int_{[t_{\ell_1-1}, t_{\ell_1}]} \int_{[t_{\ell_2-1}, t_{\ell_2}]} \check{G}(s,t)\widetilde{R}_i(s)\widetilde{R}_j(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \\
&= \sum_{\ell_1=\max(i,1)}^{\min(i+p,N)} \sum_{\ell_2=\max(j,1)}^{\min(j+p,N)} H_{\ell_1} H_{\ell_2} \int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_2}(s,t)\widetilde{R}_{i,\ell_1}(s)\widetilde{R}_{j,\ell_2}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s).
\end{aligned}$$

Now, we want to calculate each summand

$$\int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_2}(s,t)\widetilde{R}_{i,\ell_1}(s)\widetilde{R}_{j,\ell_2}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)$$

with $H_{\ell_1} H_{\ell_2} > 0$.

Case 1. $\gamma\big([t_{\ell_1-1}, t_{\ell_1}]\big) \cap \gamma\big([t_{\ell_2-1}, t_{\ell_2}]\big) = \emptyset$: Since the integrand has no singularities, no further simplification is necessary. We use tensor-Gauss quadrature with weight function 1. Theorem 5.1 can be applied.

Case 2. $\gamma\big([t_{\ell_1-1}, t_{\ell_1}]\big) = \gamma\big([t_{\ell_2-1}, t_{\ell_2}]\big)$: This implies $\ell_1 = \ell_2$. There holds

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_1}(s,t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}(t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = \int_{[0,1]} \int_{[0,s]} \check{G}_{\ell_1,\ell_1}(s, s-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}(s-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad + \int_{[0,1]} \int_{[s-1,0]} \check{G}_{\ell_1,\ell_1}(s, s-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}(s-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.14}$$

For the first summand in (5.14) we use the *Duffy transformation* $(s,t) \mapsto (s, st)$ on $[0,1] \times [0,1]$ with Jacobi determinant s. This transforms the singular points to $s = 0$ or $t = 0$. Then we apply the addition theorem for ln. Formally this reads as

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,s]} \check{G}_{\ell_1,\ell_1}(s, s-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}(s-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = \int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_1}(s, s-st) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}(s-st) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = -\frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln\left(\frac{|\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_1-1} + H_{\ell_1} s(1-t))|}{st} \right) \\
&\qquad \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}\big(s(1-t)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(s) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}\big(s(1-t)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_1}\big(s(1-t)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.15}$$

For the second summand of (5.14), we use again the Duffy transformation $(s,t) \mapsto (s, st)$ and the addition theorem for ln, to see

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,s]} \check{G}_{\ell_1,\ell_1}(1-s, 1-s+t) \widetilde{R}_{i,\ell_1}(1-s) \widetilde{R}_{j,\ell_1}(1-s+t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = \int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_1}(1-s, 1-s+st) \widetilde{R}_{i,\ell_1}(1-s) \widetilde{R}_{j,\ell_1}(1-s+st) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = -\frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln\left(\frac{\big|\gamma\big(t_{\ell_1-1} + H_{\ell_1}(1-s)\big) - \gamma\big(t_{\ell_1-1} + H_{\ell_1}(1+s(t-1))\big)\big|}{st} \right) \\
&\qquad \widetilde{R}_{i,\ell_1}(1-s) \widetilde{R}_{j,\ell_1}\big(1+s(t-1)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(s) \widetilde{R}_{i,\ell_1}(1-s) \widetilde{R}_{j,\ell_1}\big(1+s(t-1)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(t) \widetilde{R}_{i,\ell_1}(1-s) \widetilde{R}_{j,\ell_1}\big(1+s(t-1)\big) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.16}$$

To approximate the six final integrals, we use tensor-Gauss quadrature with weight function 1, $\ln(s)$ resp. $\ln(t)$. Due to the following lemma, we can apply Theorem 5.1.

Lemma 5.2. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}^\gamma_{m-1}, \check{x}^\gamma_m]$ for $m \in \{1, \dots, n_\gamma\}$, the integrands of the final terms in (5.15) and in (5.16) are, up to $\ln(s)$ resp. $\ln(t)$, $q-1$ times continuously partially differentiable on $[0,1] \times [0,1]$.*

Proof. We only prove the assertion for (5.15) and note that (5.16) can be treated analogously. For the second and third summand the assertion is obvious. For the first summand we consider for $s, t \in (0,1]$

$$\begin{aligned}
&\frac{\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_1-1} + H_{\ell_1} s(1-t))}{st} \\
&\quad = \frac{(t_{\ell_1-1} + H_{\ell_1} s - t_{\ell_1-1} - H_{\ell_1} s(1-t)) \int_{[0,1]} \gamma'(t_{\ell_1-1} + H_{\ell_1} s(1-t) + H_{\ell_1} st\tau)) \,\mathrm{d}\lambda(\tau)}{st} \\
&\quad = H_{\ell_1} \int_{[0,1]} \gamma'\big(t_{\ell_1-1} + H_{\ell_1}(s(1-t) + st\tau)\big) \,\mathrm{d}\lambda(\tau).
\end{aligned}$$

This term can be continuously extended for $s = 0$ or $t = 0$ by $H_{\ell_1} \gamma|'_{[t_{\ell-1}, t_\ell]}(t_{\ell_1-1} + H_{\ell_1} s)$. Due to the smoothness of the integrand, it is $q-1$ times continuously partially differentiable. Because of the injectivity of γ and the fact that γ' vanishes nowhere, its modulus is positive for all $s, t \in [0,1]$. Hence, applying $\ln(|\cdot|)$ to it, preserves the differentiability. The remaining factors in the integrand obviously have the same smoothness. □

Case 3. $\big|\gamma\big([t_{\ell_1-1}, t_{\ell_1}]\big) \cap \gamma\big([t_{\ell_2-1}, t_{\ell_2}]\big)\big| = 1$: We only consider the case, where the singularity in the integrand appears at $s = 0$ and $t = 1$. The other case can be treated analogously. We have $t_{\ell_1-1} = t_{\ell_2}$ or $t_{\ell_2} = b \wedge t_{\ell_1-1} = a$. There holds

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_2}(s,t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) = \\
&\quad = \int_{[0,1]} \int_{[0,s]} \check{G}_{\ell_1,\ell_2}(s, 1-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\qquad + \int_{[0,1]} \int_{[s,1]} \check{G}_{\ell_1,\ell_2}(s, 1-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.17}$$

We consider the first summand in (5.17) and use the Duffy transformation $(s,t) \mapsto (s, st)$, transforming the singularity to $s = 0$, and the addition theorem for ln

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,s]} \check{G}_{\ell_1,\ell_2}(s, 1-t) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-t) \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = \int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_2}(s, 1-st) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-st) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad = -\frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln\left(\frac{|\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_2-1} + H_{\ell_2}(1-st))|}{s} \right) \\
&\qquad \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-st) s \,\mathrm{d}\lambda(t) \mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(s) \widetilde{R}_{i,\ell_1}(s) \widetilde{R}_{j,\ell_2}(1-st) s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.18}$$

For the second summand in (5.17) we apply Fubini, the Duffy transformation $(s,t) \mapsto (st,t)$, transforming the singularity to $t = 0$, and the addition theorem for ln

$$\begin{aligned}
&\int_{[0,1]} \int_{[s,1]} \check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{R}_{j,\ell_2}(1-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \\
&\quad = \int_{[0,1]} \int_{[0,t]} \check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{R}_{j,\ell_2}(1-t)\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t) \\
&\quad = \int_{[0,1]} \int_{[0,1]} \check{G}_{\ell_1,\ell_2}(st,1-t)\widetilde{R}_{i,\ell_1}(st)\widetilde{R}_{j,\ell_2}(1-t)t\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \\
&\quad = -\frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln\left(\frac{|\gamma(t_{\ell_1-1}+H_{\ell_1}st)-\gamma(t_{\ell_2-1}+H_{\ell_2}(1-t))|}{t}\right) \\
&\qquad \widetilde{R}_{i,\ell_1}(st)\widetilde{R}_{j,\ell_2}(1-t)t\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s) \\
&\qquad - \frac{1}{2\pi} \int_{[0,1]} \int_{[0,1]} \ln(t)\widetilde{R}_{i,\ell_1}(st)\widetilde{R}_{j,\ell_2}(1-t)t\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.19}$$

To approximate the final four integrals, we use tensor-Gauss quadrature with weight function 1, $\ln(s)$ resp. $\ln(t)$. Due to the following lemma, we can apply Theorem 5.1.

Lemma 5.3. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}^\gamma_{m-1}, \check{x}^\gamma_m]$ for $m \in \{1,\dots,n_\gamma\}$, the integrands of the final terms in* (5.18) *and in* (5.19) *are, up to* $\ln(s)$ *resp.* $\ln(t)$*, $q-1$ times continuously partially differentiable on $[0,1]\times[0,1]$.*

Proof. We only prove the assertion for (5.18) and note that (5.19) can be treated analogously. For the second summand the assertion is obvious. For the first summand, let firstly $t_{\ell_1-1} = t_{\ell_2}$. We consider, for $s \in (0,1]$, $t \in [0,1]$,

$$\begin{aligned}
\frac{\gamma(t_{\ell_1-1}+H_{\ell_1}s)-\gamma(t_{\ell_2-1}+H_{\ell_2}(1-st))}{s} &= \frac{\int_{[t_{\ell_2}-H_{\ell_2}st,t_{\ell_2}+H_{\ell_1}s]}\gamma'(\tau)\,\mathrm{d}\lambda(\tau)}{s} \\
&= \int_{[-H_{\ell_2}t,H_{\ell_1}]} \gamma'(t_{\ell_2}+s\tau)\,\mathrm{d}\lambda(\tau).
\end{aligned}$$

This term can be continuously extended for $s = 0$ by $tH_{\ell_2}\gamma'^{\ell}(t_{\ell_2}) + H_{\ell_1}\gamma'^{r}(t_{\ell_2})$. Due to the smoothness of the integrand on $[H_{\ell_2}t, 0]$ and on $[0, H_{\ell_2}]$, it is $q-1$ times continuously partially differentiable. Because of the injectivity of γ and the fact that $\gamma'^{\ell}(t_{\ell_2})$ is no negative multiple of $\gamma'^{r}(t_{\ell_2})$, its modulus is positive for all $s,t \in [0,1]$. Hence, applying $\ln(|\cdot|)$ to it preserves the differentiability. The remaining factors in the integrand of the first summand in (5.18) obviously have the same smoothness. Now let $t_{\ell_2} = b \wedge t_{\ell_1-1} = a$. If one shifts t_{ℓ_1} to $t_{\ell_1} + (b-a) = b$, the proof works as before. □

5.2. **Numerical Approximation of F_h.** In this subsection we use the following abbreviations:

$$\begin{aligned}
&\partial_\nu \check{G}(s,t) := \partial_\nu G\big(\gamma(s),\gamma(t)\big)\\
&H_\ell := t_\ell - t_{\ell-1}\\
&\partial_\nu \check{G}_{\ell_1,\ell_2}(s,t) := \partial_\nu \check{G}(t_{\ell_1-1} + H_{\ell_1} s\,,\, t_{\ell_2-1} + H_{\ell_2} t)\\
&\widetilde{R}_i(s) := R_i(s)|\gamma'(s)|\\
&\widetilde{R}_{i,\ell}(s) := \widetilde{R}_i(t_{\ell-1} + H_\ell s)\\
&\check{g}(s) := g(\gamma(s))\\
&\widetilde{g}(s) := \check{g}(s)|\gamma'(s)|\\
&\widetilde{g}_\ell(s) := \widetilde{g}(t_{\ell-1} + H_\ell s)
\end{aligned}$$

We want to calculate $F_h = \langle g/2 + Kg, \hat{R}_i\rangle_{L^2(\Gamma)}$ for $i = (1-p),\dots,N-\#b+1$. There holds

$$\begin{aligned}
\langle g/2, \hat{R}_i\rangle_{L^2(\Gamma)} &= \frac{1}{2}\int_\Gamma g(x)\hat{R}_i(x)\,\mathrm{d}\mu_\Gamma(x)\\
&= \frac{1}{2}\int_{[a,b]} \check{g}(s)R_i(s)|\gamma'(s)|\,\mathrm{d}\lambda(s)\\
&= \sum_{\ell=\max(i,1)}^{\min(i+p,N)} \int_{[t_{\ell-1},t_\ell]} \widetilde{g}(s)\widetilde{R}_i(s)\,\mathrm{d}\lambda(s)\\
&= \frac{1}{2}\sum_{\ell=\max(i,1)}^{\min(i+p,N)} H_\ell \int_{[0,1]} \widetilde{g}_\ell(s)\widetilde{R}_{i,\ell}(s)\,\mathrm{d}\lambda(s).
\end{aligned}$$

We use classical Gauss quadrature to approximate these integrals. Now we consider

$$\begin{aligned}
\langle Kg, \hat{R}_i\rangle_{L^2(\Gamma)} &= \int_\Gamma\int_\Gamma \partial_\nu G(x,y)g(y)\hat{R}_i(x)\,\mathrm{d}\mu_\Gamma(y)\,\mathrm{d}\mu_\Gamma(x)\\
&= \int_{[a,b]}\int_{[a,b]} \partial_\nu \check{G}(s,t)\check{g}(t)R_i(s)|\gamma'(s)||\gamma'(t)|\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&= \sum_{\ell_1=\max(i,1)}^{\min(i+p,N)}\sum_{\ell_2=1}^{N} \int_{[t_{\ell_1-1},t_{\ell_1}]}\int_{[t_{\ell_2-1},t_{\ell_2}]} \partial_\nu \check{G}(s,t)\widetilde{R}_i(s)\widetilde{g}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&= \sum_{\ell_1=\max(i,1)}^{\min(i+p,N)}\sum_{\ell_2=1}^{N} H_{\ell_1}H_{\ell_2}\int_{[0,1]}\int_{[0,1]} \partial_\nu \check{G}_{\ell_1,\ell_2}(s,t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s).
\end{aligned}$$

For the calculation of each summand with $H_{\ell_1}, H_{\ell_2} > 0$, we have again three cases. They can be treated similarly to Subsection 5.1.

Case 1. $\gamma\big([t_{\ell_1-1},t_{\ell_1}]\big) \cap \gamma\big([t_{\ell_2-1},t_{\ell_2}]\big) = \emptyset$: Since the integrand has no singularities, no further simplification is necessary.

Case 2. $\gamma\big([t_{\ell_1-1},t_{\ell_1}]\big)=\gamma\big([t_{\ell_2-1},t_{\ell_2}]\big)$: This implies $\ell_1=\ell_2$. There holds

$$\begin{aligned}&\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_1}(s,t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_1}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\&\quad=\int_{[0,1]}\int_{[0,s]}\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_1}(s-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\&\qquad+\int_{[0,1]}\int_{[s-1,0]}\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_1}(s-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s).\end{aligned}\tag{5.20}$$

Using the Duffy transformation $(s,t)\mapsto(s,st)$ on $[0,1]\times[0,1]$ with Jacobi determinant s, which transforms the singularities to $s=0$ or $t=0$, one gets for the first summand in (5.20)

$$\begin{aligned}&\int_{[0,1]}\int_{[0,s]}\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_1}(s-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\&\quad=\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s-st)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_1}(s-st)s\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\end{aligned}\tag{5.21}$$

For the second summand of (5.20), we use again the Duffy transformation $(s,t)\mapsto(s,st)$ to see

$$\begin{aligned}&\int_{[0,1]}\int_{[0,s]}\partial_\nu\check{G}_{\ell_1,\ell_1}(1-s,1-s+t)\widetilde{R}_{i,\ell_1}(1-s)\widetilde{g}_{\ell_1}(1-s+t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\&\quad=\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_1}(1-s,1-s+st)\widetilde{R}_{i,\ell_1}(1-s)\widetilde{g}_{\ell_1}(1-s+st)s\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\end{aligned}\tag{5.22}$$

To approximate the final two integrals, we use tensor-Gauss quadrature with weight function 1. Due to the following lemma, we can apply Theorem 5.1.

Lemma 5.4. *If the parametrization γ is $q\geq 2$ times continuously differentiable on $[\check{x}_{j-1}^\gamma,\check{x}_j^\gamma]$ for $j\in\{1,\dots,n_\gamma\}$ and if $g\circ\gamma$ is $q-2$ times continuously differentiable on $[\check{x}_{j-1}^\gamma,\check{x}_j^\gamma]$ for $j\in\{1,\dots,n_\gamma\}$, the integrands of the final terms in* (5.21) *and in* (5.22) *are $q-2$ times continuously partially differentiable on $[0,1]\times[0,1]$.*

Proof. We only prove the assertion for (5.21), (5.22) can be treated analogously. For $s,t\in(0,1]$ we see as in (5.4) that

$$\begin{aligned}&\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s(1-t))\cdot 2\pi|\gamma'(t_{\ell_1-1}+H_{\ell_1}s(1-t))|\\&\quad=\Big\langle\int_{[0,1]}\gamma''\big(t_{\ell_1-1}+H_{\ell_1}s(1-t)+H_{\ell_1}(s-s(1-t))\tau\big)(1-\tau)\,\mathrm{d}\lambda(\tau),\\&\qquad\gamma'(t_{\ell_1-1}+H_{\ell_1}s(1-t))^\perp\Big\rangle\frac{H_{\ell_1}^2|s-s(1-t)|^2}{|\gamma(t_{\ell_1-1}+H_{\ell_1}s)-\gamma(t_{\ell_1-1}+H_{\ell_1}s(1-t))|^2}\end{aligned}$$

The first factor of this term can be continuously extended at $s=0$ or $t=0$ by

$$\frac{1}{2}\left\langle\gamma|''_{[t_{\ell_1-1},t_\ell]}(t_{\ell_1-1}+H_{\ell_1}s),\gamma|'_{[t_{\ell_1-1},t_\ell]}(t_{\ell_1-1}+H_{\ell_1}s)^\perp\right\rangle.$$

Due to the smoothness of the integrand, it is $q-2$ times continuously partially differentiable. For the second factor of the term we have

$$\begin{aligned}
&\frac{H_{\ell_1}^2|s-s(1-t)|^2}{|\gamma(t_{\ell_1-1}+H_{\ell_1}s)-\gamma(t_{\ell_1-1}+H_{\ell_1}s(1-t))|^2}\\
&\quad=\frac{H_{\ell_1}^2(st)^2}{H_{\ell_1}^2(st)^2|\int_{[0,1]}\gamma'(t_{\ell_1-1}+H_{\ell_1}s(1-t)+H_{\ell_1}st\tau)\,\mathrm{d}\lambda(\tau)|^2}\\
&\quad=\frac{1}{|\int_{[0,1]}\gamma'(t_{\ell_1-1}+H_{\ell_1}s(1-t)+H_{\ell_1}st\tau)\,\mathrm{d}\lambda(\tau)|^2},
\end{aligned}$$

which can be continuously extended at $s=0$ or $t=0$ by $1/\big|\gamma|'_{[t_{\ell_1-1},t_{\ell_1}]}(t_{\ell_1-1}+H_{\ell_1}s)\big|^2$. Due to the smoothness of the integrand it is $q-1$ times continuously partially differentiable. Altogether this shows that $(s,t)\mapsto\partial_\nu\check{G}_{\ell_1,\ell_1}(s,s(1-t))$ is $q-2$ times continuously partially differentiable on $[0,1]\times[0,1]$. The remaining factors in the integrand of (5.21) obviously have the same smoothness. □

Case 3. $\big|\gamma\big([t_{\ell_1-1},t_{\ell_1}]\big)\cap\gamma\big([t_{\ell_2-1},t_{\ell_2}]\big)\big|=1$: We only consider the case, where the singularity in the integrand appears at $s=0$ and $t=1$. The other case can be treated analogously. We have $t_{\ell_1-1}=t_{\ell_2}$ or $t_{\ell_2}=b\wedge t_{\ell_1-1}=a$. There holds

$$\begin{aligned}
&\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&\quad=\int_{[0,1]}\int_{[0,s]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&\qquad+\int_{[0,1]}\int_{[s,1]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s).
\end{aligned}\tag{5.23}$$

We consider the first summand in (5.23) and use the Duffy transformation $(s,t)\mapsto(s,st)$, transforming the singularity to $s=0$,

$$\begin{aligned}
&\int_{[0,1]}\int_{[0,s]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&\quad=\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-st)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-st)s\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)
\end{aligned}\tag{5.24}$$

For the second summand in (5.23), we apply Fubini and the Duffy transformation $(s,t)\mapsto(st,t)$, transforming the singularity to $t=0$,

$$\begin{aligned}
&\int_{[0,1]}\int_{[s,1]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-t)\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)\\
&\quad=\int_{[0,1]}\int_{[0,t]}\partial_\nu\check{G}_{\ell_1,\ell_2}(s,1-t)\widetilde{R}_{i,\ell_1}(s)\widetilde{g}_{\ell_2}(1-t)\,\mathrm{d}\lambda(s)\,\mathrm{d}\lambda(t)\\
&\quad=\int_{[0,1]}\int_{[0,1]}\partial_\nu\check{G}_{\ell_1,\ell_2}(st,1-t)\widetilde{R}_{i,\ell_1}(st)\widetilde{g}_{\ell_2}(1-t)t\,\mathrm{d}\lambda(t)\,\mathrm{d}\lambda(s)
\end{aligned}\tag{5.25}$$

To approximate the final two integrals, we use tensor-Gauss quadrature with weight function 1. Due to the following lemma, we can apply Theorem 5.1.

Lemma 5.5. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}_{j-1}^{\gamma}, \check{x}_j^{\gamma}]$ for $j \in \{1, \dots, n_\gamma\}$ and if $g \circ \gamma$ is $q-1$ times continuously differentiable on $[\check{x}_{j-1}^{\gamma}, \check{x}_j^{\gamma}]$ for $j \in \{1, \dots, n_\gamma\}$, the integrands of the final terms in (5.24) and in (5.25) are $q-1$ times continuously partially differentiable on $[0,1] \times [0,1]$.*

Proof. We only prove the assertion for (5.24), (5.25) can be treated analogously. Firstly we assume that $t_{\ell_1 - 1} = t_{\ell_2}$. For $s \in (0,1]$, $t \in [0,1]$, there holds

$$\begin{aligned}
&\frac{\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_2-1} + H_{\ell_2}(1-st))}{|\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_2-1} + H_{\ell_2}(1-st))|^2} \cdot s \\
&\quad = \frac{s^2}{|\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_2-1} + H_{\ell_2}(1-st))|^2} \cdot \frac{\gamma(t_{\ell_1-1} + H_{\ell_1} s) - \gamma(t_{\ell_2-1} + H_{\ell_2}(1-st))}{s} \\
&\quad = \frac{s^2}{|\int_{[t_{\ell_2} - H_{\ell_2} st, t_{\ell_2} + H_{\ell_1} s]} \gamma'(\tau) \, \mathrm{d}\lambda(\tau)|^2} \cdot \frac{\int_{[t_{\ell_2} - H_{\ell_2} st, t_{\ell_2} + H_{\ell_1} s]} \gamma'(\tau) \, \mathrm{d}\lambda(\tau)}{s} \\
&\quad = \frac{\int_{[-H_{\ell_2} t, H_{\ell_1}]} \gamma'(t_{\ell_2} + s\tau) \, \mathrm{d}\lambda(\tau)}{|\int_{[-H_{\ell_2} t, H_{\ell_1}]} \gamma'(t_{\ell_2} + s\tau) \, \mathrm{d}\lambda(\tau)|^2}.
\end{aligned}$$

At $s = 0$, this term is continuously extended by

$$\frac{H_{\ell_2} \gamma'^{\ell}(t_{\ell_2}) t + H_{\ell_1} \gamma'^{r}(t_{\ell_2})}{|H_{\ell_2} \gamma'^{\ell}(t_{\ell_2}) t + H_{\ell_1} \gamma'^{r}(t_{\ell_2})|^2},$$

since $\gamma'^{\ell}(t_{\ell_2})$ is no negative multiple of $\gamma'^{r}(t_{\ell_2})$. The continuous extension is even $q-2$ times continuously partially differentiable. This proves that $(s,t) \mapsto \partial_\nu \check{G}_{\ell_1,\ell_2}(s, 1-st)$ on $[0,1] \times [0,1]$ is $q-1$ times continuously differentiable. The remaining factors in the integrand of (5.24) obviously have the same smoothness. Now let $t_{\ell_2} = b \wedge t_{\ell_1 - 1} = a$. If one shifts t_{ℓ_1} to $t_{\ell_1} + (b-a) = b$, the proof works as before. □

5.3. **Numerical Evaluation of $V\psi_h$.** For given ψ_h in our approximation space $S(\mathcal{T}_h)$ and $s \in [a,b] \setminus \{\check{x}_0, \dots, \check{x}_n\}$, we want to evaluate $V\psi_h(\gamma(s))$. In this subsection we use the following abbreviations:

$$\begin{aligned}
\check{G}(s,t) &:= G\big(\gamma(s), \gamma(t)\big) \\
h_j &:= \check{x}_j - \check{x}_{j-1} \\
\check{G}_j(s,t) &:= \check{G}(s, \check{x}_{j-1} + h_j t) \\
\check{\psi}_h(t) &:= \psi_h(\gamma(t)) \\
\widetilde{\psi}_h(t) &:= \check{\psi}_h(t) |\gamma'(t)| \\
\widetilde{\psi}_{h,j}(t) &:= \widetilde{\psi}_h(\check{x}_{j-1} + h_j t)
\end{aligned}$$

Let $j_s \in \{1, \dots, n\}$ with $x \in [\check{x}_{j_s-1}, \check{x}_{j_s}]$. There holds

$$\begin{aligned}
\int_\Gamma G(x,y)\psi_h(y)\,\mathrm{d}\mu_\Gamma(y) &= \int_{[a,b]} \check{G}(s,t)\check{\psi}_h(t)|\gamma'(t)|\,\mathrm{d}\lambda(t) \\
&= \sum_{j=1}^{n} \int_{[\check{x}_{j-1},\check{x}_j]} \check{G}(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) \\
&= \sum_{\substack{j=1\\ j\neq j_s}}^{n} \int_{[\check{x}_{j-1},\check{x}_j]} \check{G}(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) + \int_{[\check{x}_{j_s-1},\check{x}_{j_s}]} \check{G}(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) \\
&= \sum_{\substack{j=1\\ j\neq j_s}}^{n} h_j \int_{[0,1]} \check{G}_j(s,t)\widetilde{\psi}_{h,j}(t)\,\mathrm{d}\lambda(t) + \int_{[\check{x}_{j_s-1},\check{x}_{j_s}]} \check{G}(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t).
\end{aligned}$$

For the integrals in the big sum, we use Gauss quadrature with weight function 1. To calculate the single summand we split the integral and use the addition theorem for ln

$$\begin{aligned}
&\int_{[\check{x}_{j_s-1},\check{x}_{j_s}]} G(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) = \int_{[\check{x}_{j_s-1},s]} G(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) + \int_{[s,\check{x}_{j_s}]} G(s,t)\widetilde{\psi}_h(t)\,\mathrm{d}\lambda(t) \\
&\quad = (s-\check{x}_{j_s-1}) \int_{[0,1]} -\frac{1}{2\pi} \ln\big(|\gamma(s)-\gamma(s-t(s-\check{x}_{j_s-1}))|\big)\widetilde{\psi}_h(s-t(s-\check{x}_{j_s-1}))\,\mathrm{d}\lambda(t) \\
&\qquad + (\check{x}_{j_s}-s) \int_{[0,1]} -\frac{1}{2\pi} \ln\big(|\gamma(s)-\gamma(s+t(\check{x}_{j_s}-s))|\big)\widetilde{\psi}_h(s+t(\check{x}_{j_s}-s))\,\mathrm{d}\lambda(t) \\
&\quad = (s-\check{x}_{j_s-1}) \int_{[0,1]} -\frac{1}{2\pi} \ln\left(\frac{|\gamma(s)-\gamma(s-t(s-\check{x}_{j_s-1}))|}{t}\right)\widetilde{\psi}_h(s-t(s-\check{x}_{j_s-1}))\,\mathrm{d}\lambda(t) \\
&\qquad + (s-\check{x}_{j_s-1}) \int_{[0,1]} -\frac{1}{2\pi} \ln(t)\widetilde{\psi}_h(s-t(s-\check{x}_{j_s-1}))\,\mathrm{d}\lambda(t) \\
&\qquad + (\check{x}_{j_s}-s) \int_{[0,1]} -\frac{1}{2\pi} \ln\left(\frac{|\gamma(s)-\gamma(s+t(\check{x}_{j_s}-s))|}{t}\right)\widetilde{\psi}_h(s+t(\check{x}_{j_s}-s))\,\mathrm{d}\lambda(t) \\
&\qquad + (\check{x}_{j_s}-s) \int_{[0,1]} -\frac{1}{2\pi} \ln(t)\widetilde{\psi}_h(s+t(\check{x}_{j_s}-s))\,\mathrm{d}\lambda(t).
\end{aligned} \tag{5.26}$$

To approximate the final four integrals, we use Gauss quadrature with weight function 1 resp. $\ln(t)$. The following lemma states, that for some piecewise smooth parametrization γ, the integrands are, up to $\ln(t)$, smooth.

Lemma 5.6. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}^\gamma_{j-1}, \check{x}^\gamma_j]$ for $j \in \{1, \dots, n_\gamma\}$, the integrands of the final terms in (5.26) are $q-1$ times continuously differentiable on $[0,1]$.*

Proof. We only consider the first summand. For $t \in (0,1]$ there holds

$$\frac{\gamma(s)-\gamma(s-t(s-\check{x}_{j_s-1}))}{t} = (s-\check{x}_{j_s-1}) \int_{[0,1]} \gamma'\big(s-t(s-\check{x}_{j_s-1})+t(s-\check{x}_{j_s-1})\tau\big)\,\mathrm{d}\lambda(\tau).$$

At $t = 0$, this term can be continuously extended by $(s - \check{x}_{j_s-1})\gamma'(s)$. It is $q-1$ times continuously differentiable and its modulus is positive for all $t \in [0,1]$. The application of $\ln(\cdot)$ preserves the differentiability. Moreover $t \mapsto \widetilde{\psi}_h(s - t(s - \check{x}_{j_s-1}))$ obviously has the same smoothness. □

5.4. **Numerical Evaluation of Kg.** Let $s \in [a,b] \setminus \{\check{x}_0^\gamma, \dots, \check{x}_{n_\gamma}^\gamma\}$ and $j_s \in \{1, \dots, n_\gamma\}$ with $s \in [\check{x}_{j_s-1}^\gamma, \check{x}_{j_s}^\gamma]$. In this subsection we use the following abbreviations:

$$\begin{aligned}
&\partial_\nu \check{G}(s,t) := \partial_\nu G\big(\gamma(s), \gamma(t)\big)\\
&h_j^\gamma := \check{x}_j^\gamma - \check{x}_{j-1}^\gamma\\
&\partial_\nu \check{G}_j(s,t) := \partial_\nu \check{G}(s, \check{x}_{j-1}^\gamma + h_j^\gamma t)\\
&\check{g}(t) = g(\gamma(t))\\
&\widetilde{g}(t) := \check{g}(t)|\gamma'(t)|\\
&\widetilde{g}_j(t) := \widetilde{g}(\check{x}_{j-1}^\gamma + h_j^\gamma t)
\end{aligned}$$

There holds

$$\begin{aligned}
&\int_\Gamma \partial_\nu G(x,y) g(y)\,\mathrm{d}\mu_\Gamma(y) = \int_\Gamma \partial_\nu \check{G}(x,y) g(y)\,\mathrm{d}\mu_\Gamma(y) = \sum_{j=1}^{n_\gamma} \int_{[\check{x}_{j-1}^\gamma, \check{x}_j^\gamma]} \partial_\nu \check{G}(s,t)\check{g}(t)|\gamma'(t)|\,\mathrm{d}\lambda(t)\\
&\quad = \sum_{\substack{j=1\\ j\neq j_s}}^{n_\gamma} \int_{[\check{x}_{j-1}^\gamma, \check{x}_j^\gamma]} \partial_\nu \check{G}(s,t)\widetilde{g}(t)\,\mathrm{d}\lambda(t) + \int_{[\check{x}_{j_s-1}^\gamma, s]} \partial_\nu \check{G}(s,t)\widetilde{g}(t)\,\mathrm{d}\lambda(t) + \int_{[s, \check{x}_{j_s}^\gamma]} \partial_\nu \check{G}(s,t)\widetilde{g}(t)\,\mathrm{d}\lambda(t)\\
&\quad = \sum_{\substack{j=1\\ j\neq j_s}}^{n_\gamma} h_j^\gamma \int_{[0,1]} \partial_\nu \check{G}_j(s,t)\widetilde{g}_j(t)\,\mathrm{d}\lambda(t) + \int_{[\check{x}_{j_s-1}^\gamma, s]} \partial_\nu \check{G}(s,t)\widetilde{g}(t)\,\mathrm{d}\lambda(t) + \int_{[s, \check{x}_{j_s}^\gamma]} \partial_\nu \check{G}(s,t)\widetilde{g}(t)\,\mathrm{d}\lambda(t)\\
&\quad = \sum_{\substack{j=1\\ j\neq j_s}}^{n_\gamma} h_j^\gamma \int_{[0,1]} \partial_\nu \check{G}_j(s,t)\widetilde{g}_j(t)\,\mathrm{d}\lambda(t)\\
&\qquad + (s - \check{x}_{j_s-1}^\gamma) \int_{[0,1]} \partial_\nu \check{G}\big(s, s - (s - \check{x}_{j_s-1}^\gamma)t\big)\widetilde{g}\big(s - (s - \check{x}_{j_s-1}^\gamma)t\big)\,\mathrm{d}\lambda(t)\\
&\qquad + (\check{x}_{j_s}^\gamma - s) \int_{[0,1]} \partial_\nu \check{G}\big(s, s + (\check{x}_{j_s}^\gamma - s)t\big)\widetilde{g}\big(s + (\check{x}_{j_s}^\gamma - s)t\big)\,\mathrm{d}\lambda(t).
\end{aligned} \tag{5.27}$$

To approximate the final three integrals, we use Gauss quadrature with weight function 1. This is justified by the following lemma.

Lemma 5.7. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}_{j-1}^\gamma, \check{x}_j^\gamma]$ for $j \in \{1, \dots, n_\gamma\}$ and if $g \circ \gamma$ is $q-2$ times continuously differentiable continuously differentiable on $[\check{x}_{j-1}^\gamma, \check{x}_j^\gamma]$ for $j \in \{1, \dots, n_\gamma\}$, the integrands of the final terms in (5.27) are $q-2$ times continuously differentiable on $[0,1]$.*

Proof. For the sum over j, the assertion is trivial. We only consider the second summand, the third one can be treated analogously. As in (5.4) we see for $t \in (0,1]$

$$\begin{aligned}
&\partial_\nu \check{G}\big(s, s-(s-\check{x}^\gamma_{j_s-1})t\big) \cdot 2\pi |\gamma'\big(s-(s-\check{x}^\gamma_{j_s-1})t\big)| \\
&\quad = \Big\langle \int_{[0,1]} \gamma''\big(s-(s-\check{x}^\gamma_{j_s-1})t + (s-\check{x}^\gamma_{j_s-1})t\tau\big)(1-\tau)\,\mathrm{d}\lambda(\tau), \\
&\qquad \gamma'\big(s-(s-\check{x}^\gamma_{j_s-1})t\big)^\perp \Big\rangle \frac{|(s-\check{x}^\gamma_{j_s-1})t|^2}{|\gamma(s)-\gamma(s-(s-\check{x}^\gamma_{j_s-1})t)|^2}.
\end{aligned}$$

For the second factor of this term, we have

$$\frac{|(s-\check{x}^\gamma_{j_s-1})t|^2}{|\gamma(s)-\gamma(s-(s-\check{x}^\gamma_{j_s-1})t)|^2} = \frac{1}{|\int_{[0,1]} \gamma'(s-(s-\check{x}^\gamma_{j_s-1})(-t+t\tau))\,\mathrm{d}\lambda(\tau)|^2},$$

wherefore the term can be continuously extended at $t=0$ by $\frac{\langle \gamma''(s), \gamma'(s)^\perp\rangle}{2|\gamma'(s)|^2}$. It is $q-2$ times continuously differentiable on $[0,1]$. This implies that the integrand in the second summand of (5.27) is $q-2$ times continuously differentiable as well. □

5.5. **Numerical Approximation of $|r_h|_{H^{1/2}(\omega_h(x_j))}$.** Let $j \in \{1,\dots,n\}$. In this subsection we use the following abbreviations:

$$\begin{aligned}
&\check{r}_h(s) = r_h(\gamma(s)) \\
&\check{P}(s,t) := \frac{|\check{r}_h(s)-\check{r}_h(t)|^2}{|\gamma(s)-\gamma(t)|^2} \\
&h_\ell := \check{x}_\ell - \check{x}_{\ell-1} \\
&\check{P}_{\ell,q}(s,t) := \check{P}(\check{x}_{\ell-1}+h_\ell s, \check{x}_{q-1}+h_q t) \\
&\gamma'_\ell(s) := \gamma'(\check{x}_{\ell-1}+h_\ell s)
\end{aligned}$$

There holds

$$\begin{aligned}
|r_h|_{H^{1/2}(\omega_h(x_j))} &= \int_{\omega_h(x_j)} \int_{\omega_h(x_j)} \frac{|r_h(x) - r_h(y)|^2}{|x-y|^2} \,\mathrm{d}\mu_\Gamma(y) \,\mathrm{d}\mu_\Gamma(x) \\
&= \int_{[\check{x}_{j-1},\check{x}_{j+1}]} \int_{[\check{x}_{j-1},\check{x}_{j+1}]} \frac{|\check{r}(s) - \check{r}(t)|^2}{|\gamma(s) - \gamma(t)|^2} |\gamma'(s)||\gamma'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&= \int_{[\check{x}_{j-1},\check{x}_j]} \int_{[\check{x}_{j-1},\check{x}_j]} \check{P}(s,t) |\gamma'(s)||\gamma'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad + 2 \int_{[\check{x}_j,\check{x}_{j+1}]} \int_{[\check{x}_{j-1},\check{x}_j]} \check{P}(s,t) |\gamma'(s)||\gamma'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad + \int_{[\check{x}_j,\check{x}_{j+1}]} \int_{[\check{x}_j,\check{x}_{j+1}]} \check{P}(s,t) |\gamma'(s)||\gamma'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&= h_j^2 \int_{[0,1]} \int_{[0,1]} \check{P}_{j,j}(s,t) |\gamma_j'(s)||\gamma_j'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad + 2 h_j h_{j+1} \int_{[0,1]} \int_{[0,1]} \check{P}_{j+1,j}(s,t) |\gamma_{j+1}'(s)||\gamma_j'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad + h_{j+1}^2 \int_{[0,1]} \int_{[0,1]} \check{P}_{j+1,j+1}(s,t) |\gamma_{j+1}'(s)||\gamma_{j+1}'(t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.28}$$

The first summand in (5.28) is due to the symmetry of the integrand just

$$2 \int_{[0,1]} \int_{[0,s]} \check{P}_{j,j}(s, s-t) |\gamma_j'(s)||\gamma_j'(s-t)| \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).$$

This term can be treated nearly exactly as in Case 2 of Subsection 5.2. It becomes

$$2 \int_{[0,1]} \int_{[0,1]} \check{P}_{j,j}(s, s(1-t)) |\gamma_j'(s)||\gamma_j'(s(1-t))| s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s). \tag{5.29}$$

For the third summand in (5.28) one proceeds analogously. The second one can be treated nearly exactly as in Case 3 of Subsection 5.2. It becomes

$$\begin{aligned}
&\int_{[0,1]} \int_{[0,1]} \check{P}_{j+1,j}(s, 1-st) |\gamma_{j+1}'(s)||\gamma_j'(1-st)| s \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s) \\
&\quad + \int_{[0,1]} \int_{[0,1]} \check{P}_{j+1,j}(st, 1-t) |\gamma_{j+1}'(st)||\gamma_j'(1-t)| t \,\mathrm{d}\lambda(t) \,\mathrm{d}\lambda(s).
\end{aligned} \tag{5.30}$$

To approximate the final three integrals, we use Gauss quadrature with weight function 1. Thanks to the following lemma, Theorem 5.1 can be applied.

Lemma 5.8. *If the parametrization γ is $q \geq 2$ times continuously differentiable on $[\check{x}_{j-1}, \check{x}_j]$ for $j \in \{1, \dots, n_\gamma\}$ and if $r_h \circ \gamma$ is $q-1$ times continuously differentiable on $[\check{x}_{j-1}, \check{x}_j]$ for $j \in \{1, \dots, n\}$, the integrands of the final terms in* (5.29) *and* (5.30) *are $q-1$ times continuously partially differentiable on $[0,1] \times [0,1]$.*

Proof. For $s,t \in (0,1]$ there holds

$$\begin{aligned}
\check{P}_{j,j}(s,s(1-t)) &= \frac{|\check{r}_h(\check{x}_{j-1}+h_j s)-\check{r}_h(\check{x}_{j-1}+h_j s(1-t))|^2}{|\gamma(\check{x}_{j-1}+h_j s)-\gamma(\check{x}_{j-1}+h_j s(1-t))|^2} \\
&= \frac{|\check{r}_h(\check{x}_{j-1}+h_j s)-\check{r}_h(\check{x}_{j-1}+h_j s(1-t))|^2}{(st)^2} \cdot \frac{(st)^2}{|\gamma(\check{x}_{j-1}+h_j s)-\gamma(\check{x}_{j-1}+h_j s(1-t))|^2} \\
&= \frac{\left|\int_{[0,1]} \check{r}_h'(\check{x}_{j-1}+h_j s(1-t)+h_j st\tau)\,\mathrm{d}\lambda(\tau)\right|^2}{\left|\int_{[0,1]} \gamma'(\check{x}_{j-1}+h_j s(1-t)+h_j st\tau)\,\mathrm{d}\lambda(\tau)\right|^2}.
\end{aligned}$$

At $s=0$ or $t=0$, this term can be continuously extended by

$$\frac{|\check{r}_h|'_{[\check{x}_{j-1},\check{x}_j]}(\check{x}_{j-1}+h_j s)|^2}{|\gamma|'_{[\check{x}_{j-1},\check{x}_j]}(\check{x}_{j-1}+h_j s)|^2}.$$

It is $q-1$ times continuously partially differentiable on $[0,1]\times[0,1]$. Therefore, the integrand in (5.29) has the same smoothness.

For $s \in (0,1], t \in [0,1]$ we have

$$\begin{aligned}
\check{P}_{j+1,j}(s,1-st) &= \frac{|\check{r}_h(\check{x}_j+h_{j+1}s)-\check{r}_h(\check{x}_{j-1}+h_j(1-st))|^2}{|\gamma(\check{x}_j+h_{j+1}s)-\gamma(\check{x}_{j-1}+h_j(1-st))|^2} \\
&= \frac{|\check{r}_h(\check{x}_j+h_{j+1}s)-\check{r}_h(\check{x}_{j-1}+h_j(1-st))|^2}{s^2} \cdot \frac{s^2}{|\gamma(\check{x}_j+h_{j+1}s)-\gamma(\check{x}_{j-1}+h_j(1-st))|^2} \\
&= \frac{|\int_{[\check{x}_j-h_j st,\check{x}_j+h_{j+1}s]} \check{r}_h'(\tau)\,\mathrm{d}\lambda(\tau)|^2}{s^2} \cdot \frac{s^2}{|\int_{[\check{x}_j-h_j st,\check{x}_j+h_{j+1}s]} \gamma'(\tau)\,\mathrm{d}\lambda(\tau)|^2} \\
&= \frac{|\int_{[-h_j t,h_{j+1}]} \check{r}_h'(\check{x}_j+s\tau)\,\mathrm{d}\lambda(\tau)|^2}{|\int_{[-h_j t,h_{j+1}]} \gamma'(\check{x}_j+s\tau)\,\mathrm{d}\lambda(\tau)|^2}
\end{aligned}$$

At $s=0$ this term can be continuously extended by

$$\frac{(h_j t\check{r}'^{\ell}(\check{x}_j)+h_{j+1}\check{r}'^{r}(\check{x}_j))^2}{|h_j t\gamma'^{\ell}(\check{x}_j)+h_{j+1}\gamma'^{r}(\check{x}_j)|^2}.$$

It is $q-1$ times continuously partially differentiable on $[0,1]\times[0,1]$. Therefore, the first integrand in (5.30) has the same smoothness. For the second one, one proceeds analogously. □

6. Numerical Examples

In this section, we apply Alogrithm 4.15 developed in Section 4 to Symm's integral equation $V\phi = Kg + g/2$ of Section 5. From Section 1, we see that the normal derivative of a weak solution of the Laplace problem with Dirichlet boundary conditions, i.e.

$$\begin{aligned} -\Delta u &= 0 \quad \text{in } \Omega, \\ u &= g \quad \text{on } \Gamma, \end{aligned} \tag{6.1}$$

solves Symm's integral equation. Conversely the weak solution of the Laplace problem can be calculated via the representation formula from the solution of Symm's integral equation.

We present three test examples for which the exact solution is known. In each example, the parametrization γ of the boundary Γ has the special form

$$\gamma(t) = \sum_{i\in\mathbb{Z}} C_i^\gamma R_{i,p_\gamma}^{\check{\mathcal{K}}^\gamma,\mathcal{W}^\gamma}(t) \tag{6.2}$$

for all $t \in [a, b]$. Here, $p_\gamma \in \mathbb{N}$ is the polynomial degree, $\check{\mathcal{K}}^\gamma$ and $\mathcal{W}^\gamma$ are periodic knots and weights of length N_γ as in Definition 4.13 and $(C_i)_{i\in\mathbb{Z}}$ are N_γ-periodic *control points* in $\mathbb{R}^2$. Curves of this type are called *NURBS curves*.

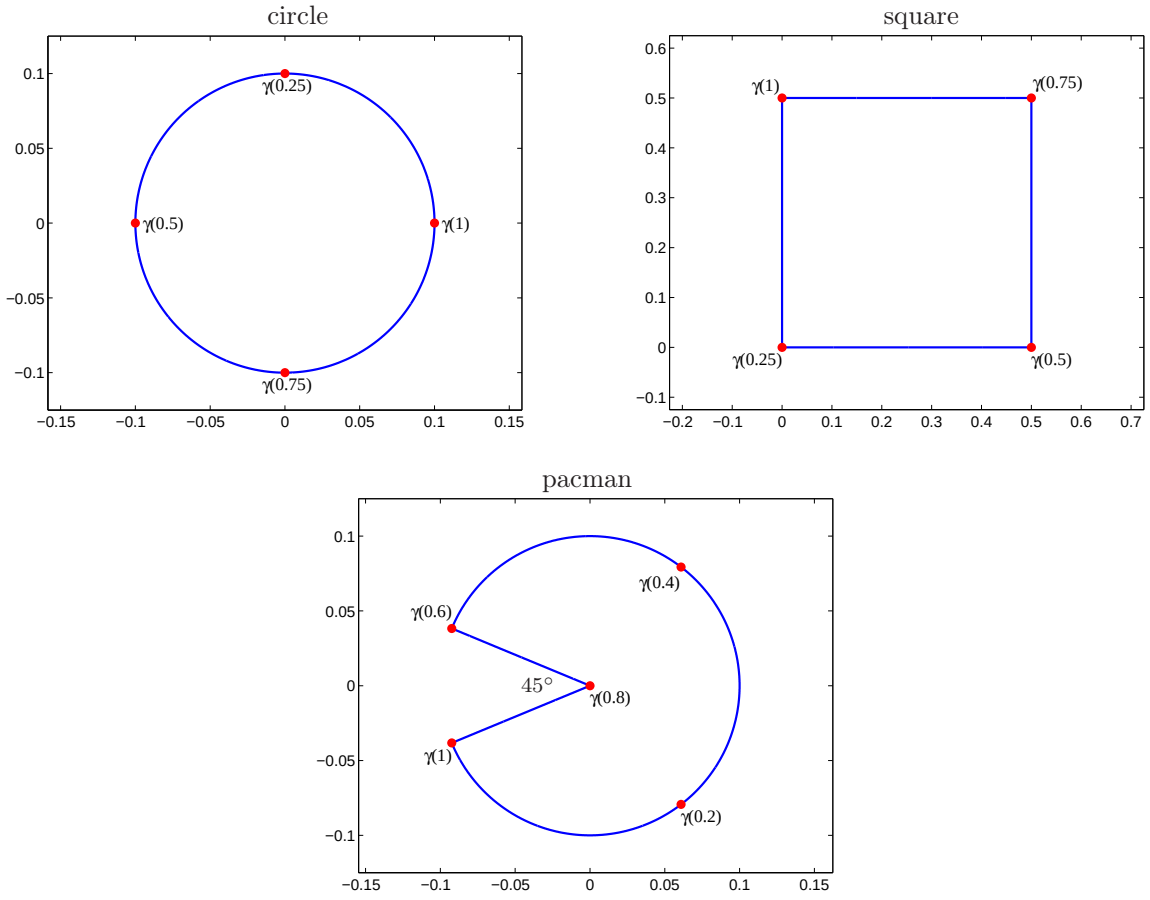

Figure 6.1. Geometries for the experiments.

We choose the same polynomial degree $p := p_\gamma$ for our approximation spaces $S(\mathcal{T}_k)$. As initial knots and weights for the algorithm, we take the same as for the geometry, i.e. $\check{\mathcal{K}}_0 := \check{\mathcal{K}}^\gamma$ and $\mathcal{W}_0 := \mathcal{W}^\gamma$. This approach just reflects the main idea of isogeometric analysis. For comparison, we shall also consider a rather conventional approach by choosing

$$\check{\mathcal{K}}_0 := (\underbrace{\check{x}_1^\gamma, \dots, \check{x}_1^\gamma}_{p+1 \text{ times}}, \underbrace{\check{x}_2^\gamma, \dots, \check{x}_2^\gamma}_{p+1 \text{ times}} \dots, \underbrace{\check{x}_{n_\gamma}^\gamma, \dots, \check{x}_{n_\gamma}^\gamma}_{p+1 \text{ times}}),$$

with the nodes $(\check{x}_j^\gamma)_{j=1}^{n_\gamma}$ of $\mathcal{K}^\gamma$, and all weights equal to one. According to Theorem 4.4 and Lemma 4.2, (viii), the start space coincides with the transformed space of piecewise polynomials on $[a,b)$ with break points $\{\check{x}_j^{[0]} : j = 1, \dots, n_0 - 1\}$. Recalling that the refined weights are just convex combinations of the old ones, we see that the space $S(\mathcal{T}_k)$ coincides with the transformed space of piecewise polynomials on $[a,b)$ with break points $\{\check{x}_j^{[k]} : j = 1, \dots, n_k - 1\}$ which are $p-1$ times differentiable at each point of $\{\check{x}_j^{[k]} : j = 1, \dots, n_k - 1\} \setminus \{\check{x}_j^{[0]} : j = 1, \dots, n_0 - 1\}$. In each case we choose $\kappa_{\max}$ as $2\kappa(\check{\mathcal{T}}_0)$. We consider uniform refinement, i.e. $\vartheta = 1$, and adaptive refinement with $\vartheta = 0.75$. For uniform refinement and smooth solution ϕ we expect the convergence rate $\mathcal{O}(h^{3/2+p}) = \mathcal{O}(n^{-3/2-p})$ in accordance to [SS11, Corollary 4.1.34].

To calculate the energy error, we use that the error is orthogonal to the approximation space

$$\|\phi - \phi_k\|_V^2 = \|\phi\|_V^2 - \|\phi_k\|_V^2.$$

The energy norm of the discrete solution ϕ_k reads $\|\phi_k\|_V^2 = c_{h_k} \cdot V_{h_k} c_{h_k}$ with the stiffness matrix V_{h_k} and the coefficient vector c_{h_k} as in Section 5. The unknown energy norm $\|\phi\|_V$ is either computed exactly or obtained by Aitkin's Δ^2 extrapolation of the sequence of values for discrete solutions of the conventional approach with adaptively refined meshes.

The code for the following experiments can be found in Appendix B.

6.1. **Smooth solution on circle.** We consider problem (6.1) on the circle with midpoint $(0,0)$ and radius $1/10$ with exact solution $u(x,y) = x^2 + 2xy - y^2$ and solve the corresponding Symm's integral equation. The normal derivative $\phi := \partial u / \partial \nu$ is smooth on the whole boundary

$$\phi(x,y) = 20(x^2 + 2xy - y^2). \tag{6.3}$$

The geometry is parametrized on $[0,1]$ by the NURBS curve induced by[11]

$$\begin{aligned}
p_\gamma &= 2, \\
\check{\mathcal{K}}^\gamma &= \left(\frac{1}{4}, \frac{1}{4}, \frac{2}{4}, \frac{2}{4}, \frac{3}{4}, \frac{3}{4}, 1, 1, 1\right), \\
\mathcal{W}^\gamma &= \left(1, \frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}, 1, \frac{1}{\sqrt{2}}, 1, 1, \frac{1}{\sqrt{2}}\right), \\
(C_i)_{i=1}^{N_\gamma} &= \frac{1}{10} \cdot \left(\begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}-1\\1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}0\\-1\end{pmatrix}, \begin{pmatrix}1\\-1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\right).
\end{aligned}$$

In Figure 6.1 we can see the geometry and the γ-values of the initial nodes. As can be seen in Figure 6.2, $\phi \circ \gamma$ is smooth on the whole parameter domain.

[11] Notice that this parametrization does not coincide with $t \mapsto (\cos(t), \sin(t))$.

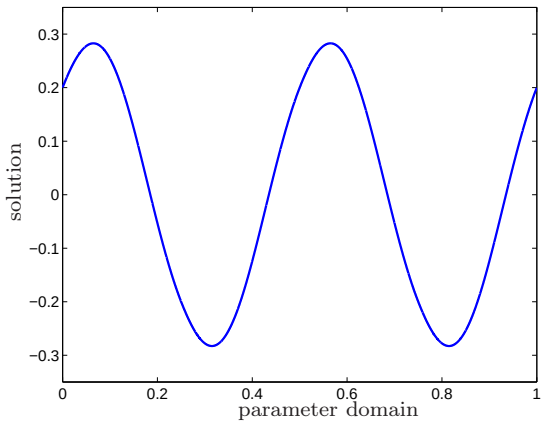

FIGURE 6.2. Experiment on circle geometry. The smooth solution $\phi \circ \gamma$ is plotted.

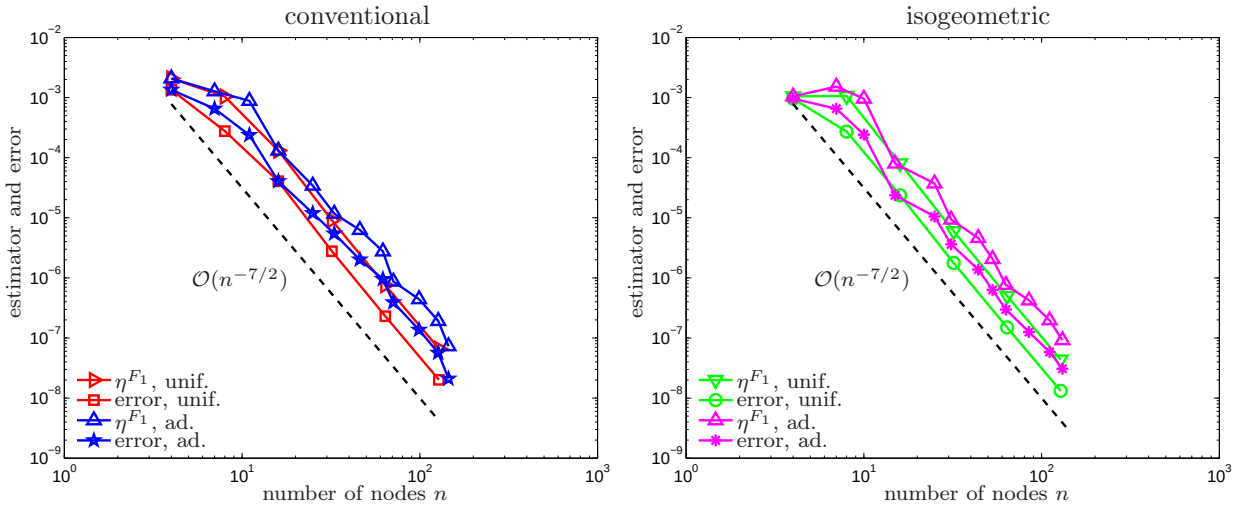

FIGURE 6.3. Experiment on circle geometry. Error and estimator are plotted versus the number of nodes.

Figure 6.3 and Figure 6.4 show the error and the error estimator for the conventional approach with piecewise polynomials and for the isogeometric one with NURBS corresponding to the geometry. All values are shown on a log-log scale so that the experimental convergence rates are visible as the slope of the corresponding curves. For the calculation of the error we used the exact energy norm of the solution $\|\phi\|_V^2 = \pi/5000$, calculated in Maple. As expected, the conventional approach with uniform refinement leads to the optimal rate $n^{-7/2}$, and adaptive refinement leads to the same rate. The isogeometric approach shows nearly the same convergence behavior. In each case, the curves for the error and its corresponding estimator are parallel up to a multiplicative constant, which appears in the log-log scale as additive constant, nearly identical. This empirically confirms the proven efficiency and reliability of the Faermann estimator η^{F_1}. In Figure 6.5 the error is plotted over the computational time. All curves show similar rates. However, since the solution does not

lack any regularity, we can observe (at least) a better multiplicative constant for uniform refinement.

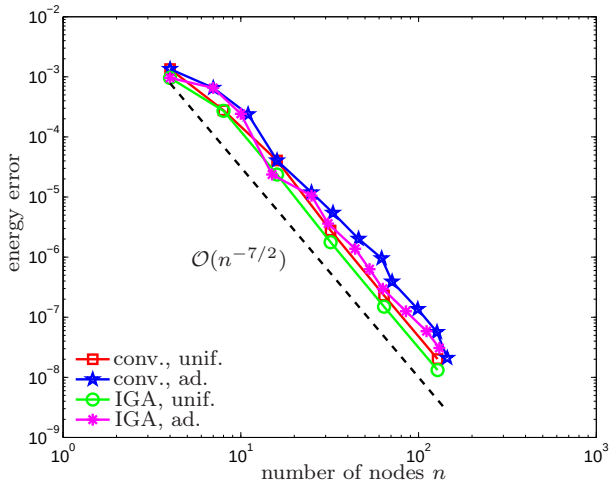

FIGURE 6.4. Experiment on circle geometry. The error is plotted versus the number of nodes.

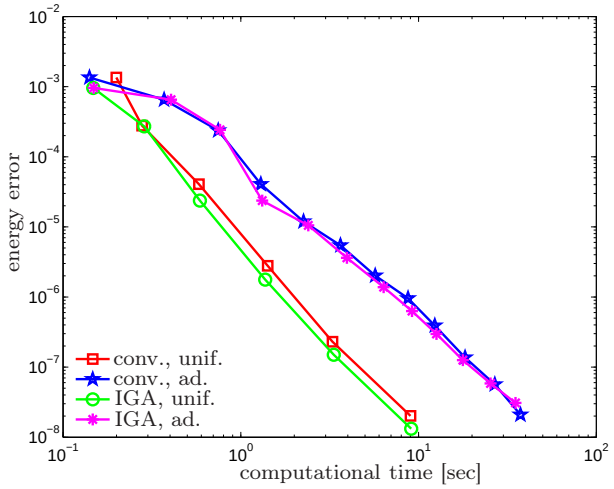

FIGURE 6.5. Experiment on circle geometry. The error is plotted versus the computational time.

6.2. **Piecewise smooth solution on square.** We consider problem (6.1) on the halved unit square $\Omega = [0, 1/2]^2$ with exact solution $u(x, y) = \exp(x)\cos(y)$ and solve the corresponding Symm's integral equation. The normal derivative $\phi := \partial u/\partial \nu$ is piecewise smooth on the boundary with discontinuities at the corners

$$\phi(x, y) = \begin{pmatrix} \exp(x)\cos(y) \\ -\exp(x)\sin(y) \end{pmatrix} \cdot \nu(x, y). \tag{6.4}$$

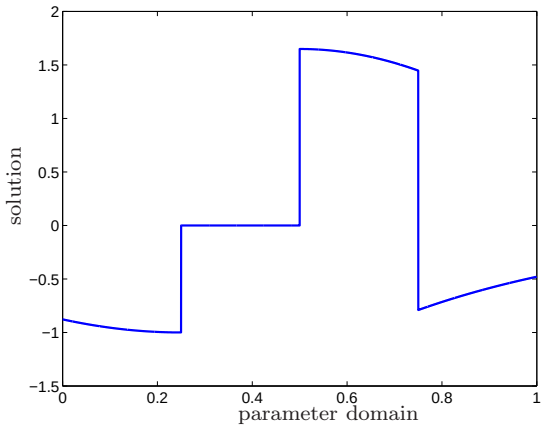

FIGURE 6.6. Experiment on square geometry. The piecewise smooth solution $\phi \circ \gamma$.

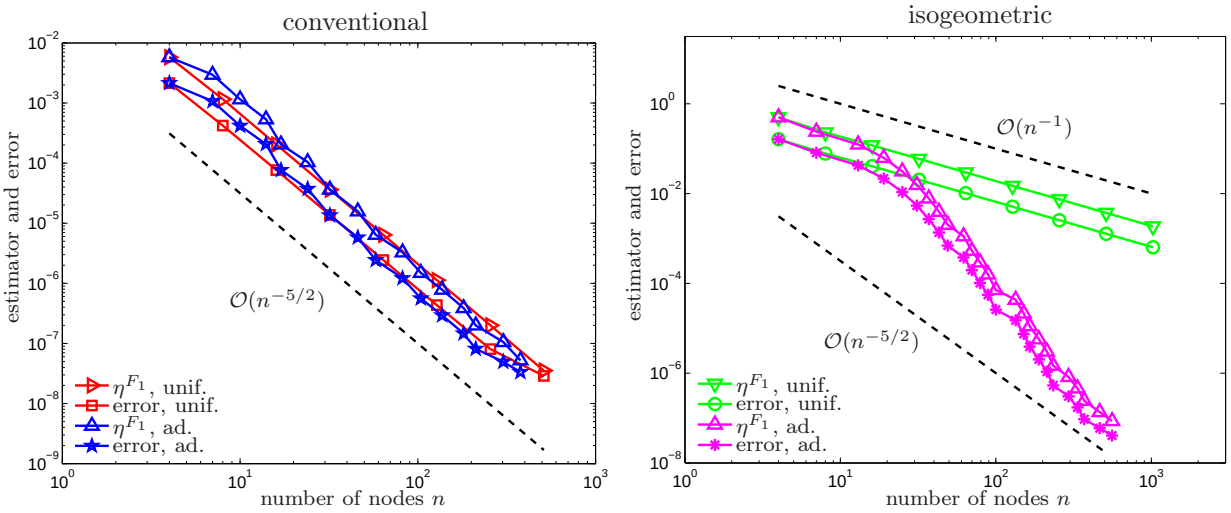

FIGURE 6.7. Experiment on square geometry. Error and estimator are plotted versus the number of nodes.

The geometry is parametrized on $[0,1]$ by the NURBS curve induced by

$$\begin{aligned} p_\gamma &= 1, \\ \check{\mathcal{K}}^\gamma &= \left(\frac{1}{4}, \frac{2}{4}, \frac{3}{4}, 1\right), \\ \mathcal{W}^\gamma &= (1,1,1,1), \\ (C_i)_{i=1}^{N_\gamma} &= \frac{1}{2} \cdot \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right). \end{aligned}$$

The geometry and the γ-values of the initial nodes are plotted in Figure 6.1. In Figure 6.6 we can observe that $\phi \circ \gamma$ is piecewise smooth on the parameter domain with jumps at the initial nodes. In this special example the approximation space $S(\mathcal{T}_k)$ for the isogeometric approach, transformed onto the parameter domain, consists of all piecewise linear continuous

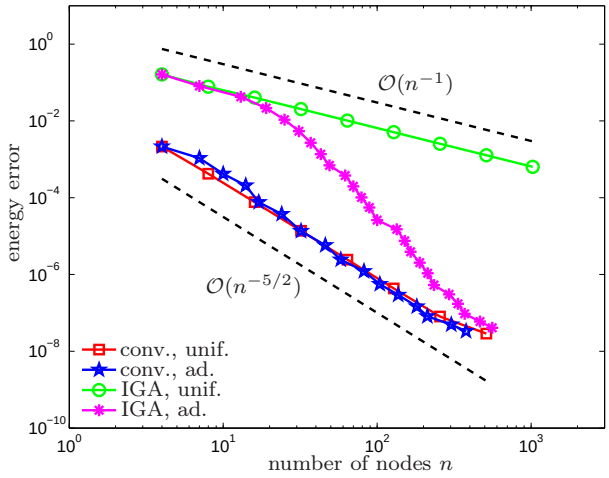

FIGURE 6.8. Experiment on square geometry. The error is plotted versus the number of nodes.

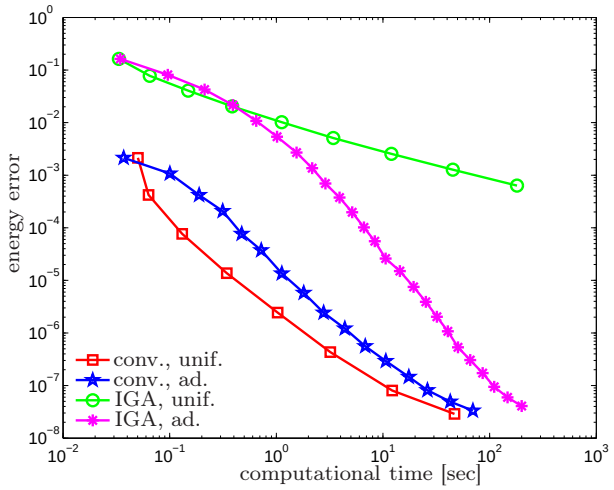

FIGURE 6.9. Experiment on square geometry. The error is plotted versus the computational time.

polynomials on the mesh. In contrast, the conventional approach does allow discontinuities at the initial nodes. Therefore, we expect the conventional approach to be superior in this case.

Indeed, in Figure 6.7 and in Figure 6.8 we can observe suboptimal convergence rate n^{-1} for IGA with uniform refinement. However this problem can be solved by adaptivity. Now, one obtains (at least asymptotically) a similar convergence rate as for the conventional approach, i.e. $\mathcal{O}(n^{-5/2})$. For the calculation of the error we use extrapolation and get $\|\phi\|_V^2 = 0.219715794638521$. The efficiency and reliability of the Faermann estimator η^{F_1} is again confirmed. In Figure 6.9, the error is plotted over the computational time. For adaptive refinement, we plot in Figure 6.10 the indices of the knots in $(a, b] = (0, 1]$ of the last refinement step over the knots. Big increase of the curve at any point t indicates high refinement at this point. The conventional approach refines nearly uniformly outside

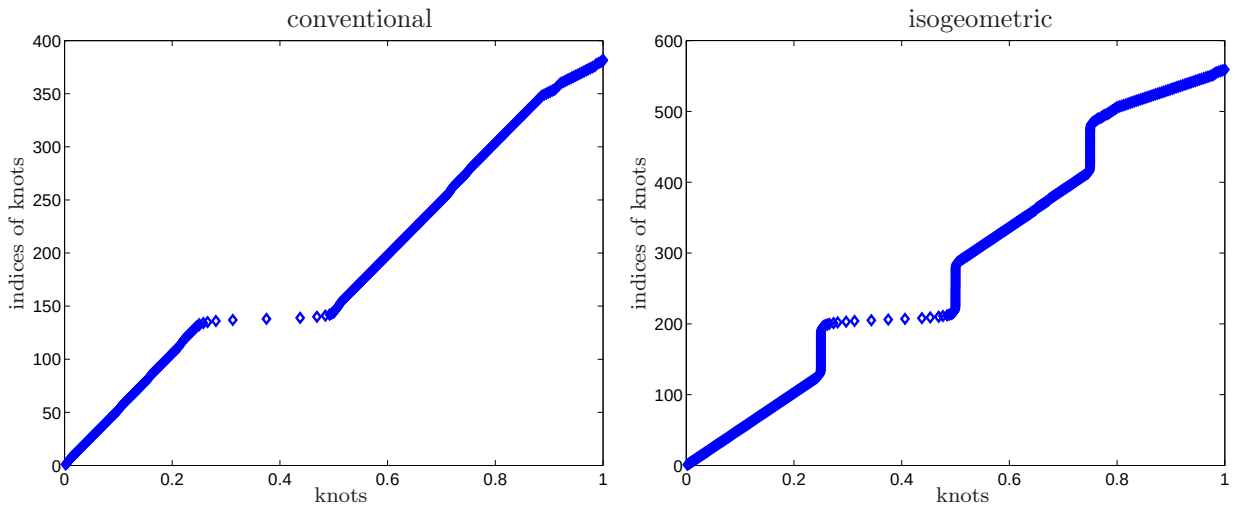

FIGURE 6.10. Experiment on square geometry. The indices of the last knot vector versus the knots are plotted.

of the interval $[0.25, 0.5]$, where the transformed solution $\phi \circ \gamma$ vanishes. As expected, the isogeometric approach mainly refines at the jump points of $\phi \circ \gamma$, $0.25, 0.5$ and 0.75, to deal with the discontinuity of the solution there. Note that by periodicity, 0 resp. 1 is as well a jump point. However, here no refinement is necessary, since the elements of the approximation space do not need to be periodic.

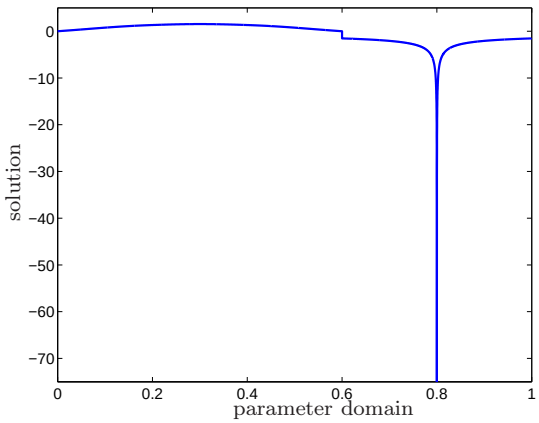

FIGURE 6.11. Experiment on pacman geometry. The singular solution $\phi \circ \gamma$ with jump is plotted.

6.3. **Singular solution on pacman.** As third example we consider, for the parameter $\tau = 4/7$, problem (6.1) on

$$\Omega := \left\{ r(\cos(\alpha), \sin(\alpha)) : 0 \leq r < \frac{1}{10}, \alpha \in \left(-\frac{\pi}{2\tau}, \frac{\pi}{2\tau} \right) \right\} \tag{6.5}$$

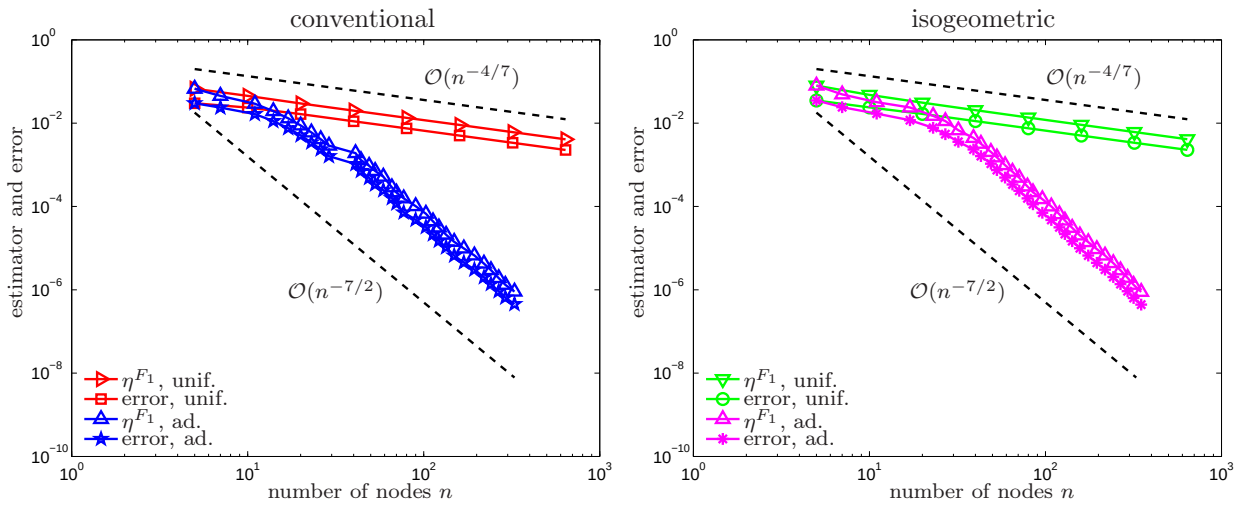

FIGURE 6.12. Experiment on pacman geometry. Error and estimator are plotted versus the number of nodes.

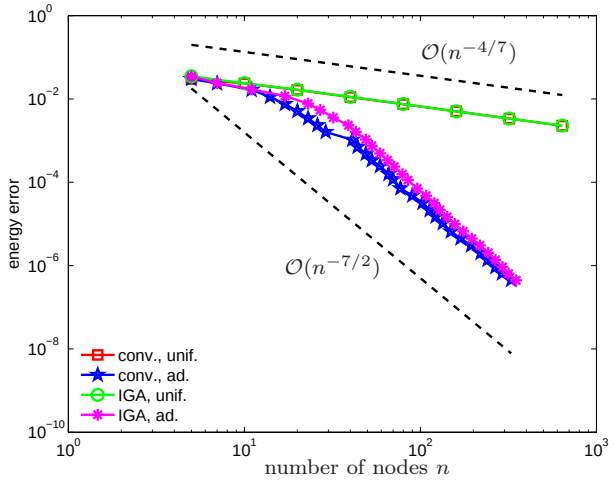

FIGURE 6.13. Experiment on pacman geometry. The error is plotted versus the number of nodes.

with exact solution

$$u(x,y) = r^\tau \cos(\tau\alpha) \quad \text{in polar coordinates} \quad (x,y) = r(\cos\alpha, \sin\alpha). \tag{6.6}$$

The normal derivative of u is

$$\phi(x,y) = \begin{pmatrix} \cos(\alpha)\cos(\tau\alpha) + \sin(\alpha)\sin(\tau\alpha) \\ \sin(\alpha)\cos(\tau\alpha) - \cos(\alpha)\sin(\tau\alpha) \end{pmatrix} \cdot \nu(x,y) \cdot \tau \cdot r^{\tau-1}. \tag{6.7}$$

It has a singularity at $(0,0)$. With $w = \cos(\pi/\tau)$, the geometry is parametrized on $[0,1]$ by the NURBS curve induced by

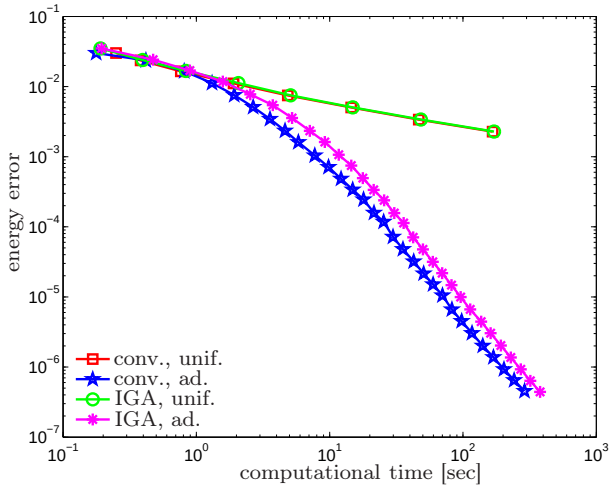

FIGURE 6.14. Experiment on pacman geometry. The error is plotted versus the computational time.

$$
\begin{aligned}
p_\gamma &= 2,\\
\check{\mathcal{K}}^\gamma &= \left(\frac{1}{5},\frac{1}{5},\frac{2}{5},\frac{2}{5},\frac{3}{5},\frac{3}{5},\frac{4}{5},\frac{4}{5},1,1,1\right),\\
\mathcal{W}^\gamma &= (1,w,1,w,1,1,1,1,1,1,w),\\
(C_i)_{i=1}^{N_\gamma} &= \frac{1}{10}\cdot\left(\begin{pmatrix}\cos(\pi/\tau\cdot(-1)/6)\\ \sin(\pi/\tau\cdot(-1)/6)\end{pmatrix},\frac{1}{w}\begin{pmatrix}\cos(\pi/\tau\cdot 0/6)\\ \sin(\pi/\tau\cdot 0/6)\end{pmatrix},\begin{pmatrix}\cos(\pi/\tau\cdot 1/6)\\ \sin(\pi/\tau\cdot 1/6)\end{pmatrix},\right.\\
&\quad \frac{1}{w}\begin{pmatrix}\cos(\pi/\tau\cdot 2/6)\\ \sin(\pi/\tau\cdot 2/6)\end{pmatrix},\begin{pmatrix}\cos(\pi/\tau\cdot 3/6)\\ \sin(\pi/\tau\cdot 3/6)\end{pmatrix},\frac{1}{2}\begin{pmatrix}\cos(\pi/\tau\cdot 3/6)\\ \sin(\pi/\tau\cdot 3/6)\end{pmatrix},\begin{pmatrix}0\\0\end{pmatrix},\\
&\quad \frac{1}{2}\begin{pmatrix}\cos(\pi/\tau\cdot(-3)/6)\\ \sin(\pi/\tau\cdot(-3)/6)\end{pmatrix},\begin{pmatrix}\cos(\pi/\tau\cdot(-3)/6)\\ \sin(\pi/\tau\cdot(-3)/6)\end{pmatrix},\begin{pmatrix}\cos(\pi/\tau\cdot(-3)/6)\\ \sin(\pi/\tau\cdot(-3)/6)\end{pmatrix},\\
&\quad \left.\frac{1}{w}\begin{pmatrix}\cos(\pi/\tau\cdot(-2)/6)\\ \sin(\pi/\tau\cdot(-2)/6)\end{pmatrix}\right).
\end{aligned}
$$

In Figure 6.11, the solution is plotted over the parameter domain. We can see that it has beside the singularity at $t = 0.8$ as well two jumps, one at 0 resp. 1 and one at 0.6.

In Figure 6.12 and in Figure 6.13, the error and the error estimator for the conventional approach with piecewise polynomials and for the isogeometric one with NURBS corresponding to the geometry are plotted. For the calculation of the error, we have extrapolated $\|\phi\|_V^2 = 0.083525924784109$. Since the solution lacks regularity, the conventional approach with uniform refinement leads to the suboptimal rate $n^{-4/7}$, whereas adaptive refinement leads to the optimal rate $n^{-7/2}$. The isogeometric approach shows nearly the same convergence behavior. In Figure 6.14, the error is plotted over the computational time. This time, adaptive refinement is by far superior to uniform refinement. The conventional and the isogeometric approach show the same rate. For adaptive refinement, we plot in Figure 6.15 the

indices of the knots in $(a, b] = (0, 1]$ of the last refinement step over the knots. Big increase of the curve at any point t indicates high refinement at t. For the conventional approach, we see that the algorithm mainly refines the mesh at $t = 0.8$, which is the point where the singularity occurs. The isogeometric algorithm refines additionally at the jump point $t = 0.6$ but not at the jump point $t = 0$ resp. $t = 1$. As in the previous example, this follows from the continuity of the functions in $S(\mathcal{T}_k) \circ \gamma$ at the point $t = 0.6$.

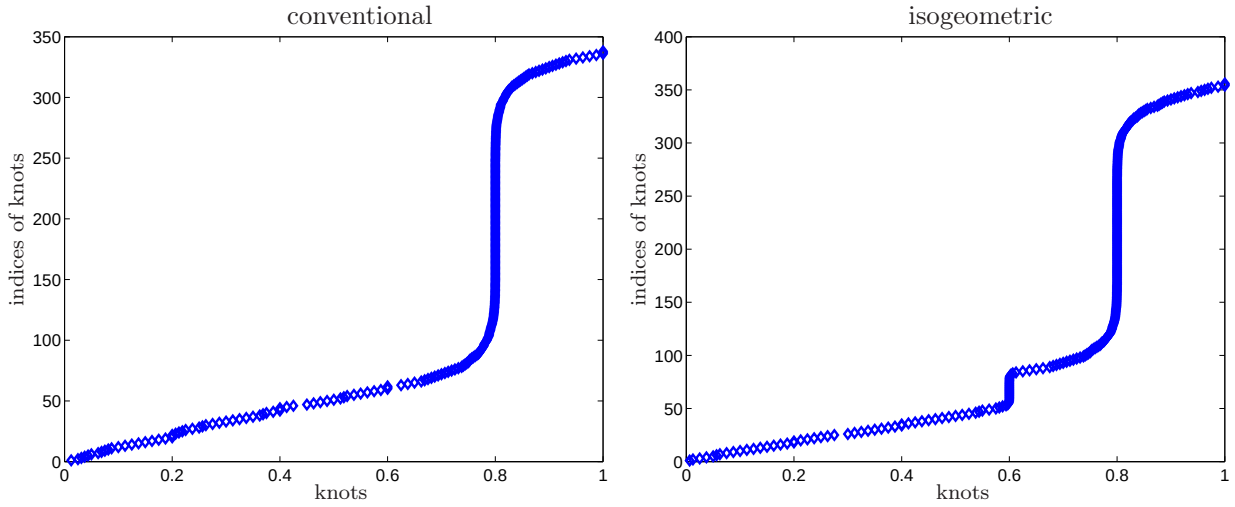

FIGURE 6.15. Experiment on pacman geometry. The indices of the last knot vector versus the knots are plotted.

Appendix A. Differential Geometry

Definition A.1. A *regular closed curve* is a closed path $\gamma : [a, b] \to \Gamma \subseteq \mathbb{R}^d$, where $d \geq 2$, with the following properties: It is continuous, piecewise continuously differentiable, and $\gamma|_{[a,b)}$ is bijective. The $(b-a)$-periodic extension of γ to $\mathbb{R}$ is denoted by γ as well. Furthermore, we demand $\gamma'^{\ell}(t) \neq 0, \gamma'^{r}(t) \neq 0$ for $t \in \mathbb{R}$ and that $\gamma'^{\ell}(t) + c\gamma'^{r}(t) \neq 0$ for all $c > 0$, where γ'^{ℓ} and γ'^{r} denote the left resp. right derivative of γ.

We call two regular closed curves $\gamma_1 : [a_1, b_1] \to \mathbb{R}^d$ and $\gamma_2 : [a_2, b_2] \to \mathbb{R}^d$ equivalent, if there exists a *parameter transformation* from $[a_1, b_1]$ to $[a_2, b_2]$, i.e. a bijective, continuous and piecewise continuously differentiable $\varphi : [a_1, b_1] \to [a_2, b_2]$ such that $\varphi'^{\ell}(t) > 0$ resp. $\varphi'^{r}(t) > 0$ for all $t \in [a_1, b_1]$, with $\gamma_1 = \gamma_2 \circ \varphi$.

Note that if $\gamma_2 : [a_2, b_2] \to \Gamma$ is a regular closed curve and $\varphi : [a_1, b_1] \to [a_2, b_2]$ is a parameter transformation, $\gamma_2 \circ \varphi$ is also a regular closed curve.

Remark A.2. The inverse of a parameter transformation is again a parameter transformation. Therefore, the defined relation between regular closed curves is symmetric. Since the composition of parameter transformations is again a parameter transformation, it is as well transitive. Moreover it is obviously reflexive and hence an equivalence relation.

In the following we consider some assertions about regular closed curves.

Theorem A.3. *Let $\gamma : [a, b] \to \Gamma$ be a regular closed curve. Then $\gamma|^{-1}_{(r+a,r+b)}$ is locally Lipschitz continuous for all $r \in \mathbb{R}$.*

Proof. Without loss of generality we assume that $r = 0$. First we show that $\gamma|^{-1}_{(a,b)}$ is continuous. Let $\big(\gamma(t_n)\big)_{n\in\mathbb{N}}$ with $t_n \in (a, b)$ for $n \in \mathbb{N}$ be a sequence converging to some limit $\gamma(t)$ with $t \in (a, b)$. If $\big(\gamma(t_{n_k})\big)_{k\in\mathbb{N}}$ is a subsequence, compactness of $[a, b]$ provides a convergent subsequence of $(t_{n_k})_{k\in\mathbb{N}}$ with limit $t' \in [a, b]$. Since γ is continuous, we have $\gamma(t) = \gamma(t')$ and hence $t = t'$. This proves the continuity. In particular, this implies that $\gamma|(a, b)$ is a homeomorphism from (a, b) to $\Gamma \setminus \{\gamma(a)\}$.

Now, let $x \in \Gamma \setminus \{\gamma(a)\}$, $\check{x} \in (a, b)$ with $\gamma(\check{x}) = x$, and $I \subseteq (a, b)$ a compact interval such that the interior of I contains $\check{x}$ and γ is differentiable on $I \setminus \{\check{x}\}$. Since $\gamma|_{(a,b)}$ is a homeomorphism, $\gamma(I)$ is a neigbourhood of x. It remains to show that the function

$$g : I \times I \setminus \underbrace{\big\{(s, t) \in \mathbb{R}^2 \,:\, s \neq t\big\}}_{=:D} \to \mathbb{R} : (s, t) \mapsto \left|\frac{\gamma(t) - \gamma(s)}{t - s}\right|$$

has a positive infimum. Because of the substitution rule, there holds for $s, t \in I \times I \setminus D$

$$g(s, t) = \left|\frac{1}{t - s}\int_s^t \gamma'(\rho)\,\mathrm{d}\rho\right| = \left|\int_0^1 \gamma'\big(\rho(t - s) + s\big)\,\mathrm{d}\rho\right|.$$

Let $\big((s_n, t_n)\big)_{n\in\mathbb{N}}$ be an infinizing sequence in $I \times I \setminus D$ which converges to some limit $(s_\infty, t_\infty) \in I \times I$. If the limit is in $I \times I \setminus D$ the infimum is a positive minimum. If the limit is in $D \setminus \{(\check{x}, \check{x})\}$, the Lebesgue dominated convergence theorem yields

$$g(s_n, t_n) \to \left|\int_0^1 \gamma'\big(\rho(t_\infty - t_\infty) + t_\infty\big)\,\mathrm{d}\rho\right| = |\gamma'(t_\infty)| \neq 0.$$

Finally, we consider the case $(s_\infty, t_\infty) = (\check{x}, \check{x})$. Without loss of generality we, assume that either $s_n, t_n \leq \check{x}$, or $s_n \leq \check{x} \leq t_n$, or $\check{x} \leq s_n, t_n$, each for all $n \in \mathbb{N}$. In the cases $s_n, t_n \leq \check{x}$ resp. $\check{x} \leq s_n, t_n$, we have

$$g(s_n, t_n) \to |\gamma'^{\ell}(\check{x})| \neq 0 \qquad \text{resp.} \qquad g(s_n, t_n) \to |\gamma'^{r}(\check{x})| \neq 0.$$

Finally for $s_n \leq \check{x} \leq t_n$, there holds

$$g(s_n, t_n) = \left| \int_0^{\frac{\check{x}-s_n}{t_n-s_n}} \gamma'\big(\underbrace{\rho(t-s)+s}_{\leq \check{x}}\big)\,\mathrm{d}\rho + \int_{\frac{\check{x}-s_n}{t_n-s_n}}^1 \gamma'\big(\underbrace{\rho(t-s)+s}_{\geq \check{x}}\big)\,\mathrm{d}\rho \right|. \tag{A.1}$$

Because of $\frac{\check{x}-s_n}{t_n-s_n} \in [0,1]$, we can assume convergence of $\big(\frac{\check{x}-s_n}{t_n-s_n}\big)_{n\in\mathbb{N}}$ to some limit $q \in [0,1]$. For any $\rho \in [0, q)$, it holds $\rho < \frac{\check{x}-s_n}{t_n-s_n}$ for sufficiently large $n \in \mathbb{N}$ and therefore

$$\gamma'\big(\rho(t_n - s_n) + s_n\big) \to \gamma'^{\ell}(\check{x}).$$

Hence we may use Lebesgue's dominated convergence theorem, to see that the first summand in (A.1) converges to $q\gamma'^{\ell}(\check{x})$. Analogous considerations for the second summand show that

$$g(s_n, t_n) \to |q\gamma'^{\ell}(\check{x}) + (1-q)\gamma'^{r}(\check{x})| \neq 0.$$

We conclude that in each case the limit of $\big(g(s_n, t_n)\big)_{n\in\mathbb{N}}$, which is just the infimum of g on $I \times I \setminus D$, is greater than zero. □

Lemma A.4. *Let (X, d_X) and (Y, d_Y) be metric spaces and $f : X \to Y$ locally Lipschitz continuous. Then $f|_K$ is Lipschitz continuous for each compact subset $K \subseteq X$.*

Proof. Without loss of generality we may assume that $X = K$. We prove the assertion by contradiction. If f was not Lipschitz continuous, for all $n \in \mathbb{N}$ there would exist $x_n, x_n' \in X$ with

$$d_Y\big(f(x_n), f(x_n')\big) > n d_Y\big(f(x_n), f(x_n')\big). \tag{A.2}$$

Because of the compactness, we can choose $(x_n)_{n\in\mathbb{N}}$ and $(x_n')_{n\in\mathbb{N}}$ such that they converge to some limit $x \in X$ resp. $x' \in X$. Moreover the range of the mapping $f(X)$ is compact, since f is continuous and X is compact. This implies the boundedness of $f(X)$ and hence

$$d_X(x_n, x_n') < \frac{1}{n} d_Y\big(f(x_n), f(x_n')\big) \to 0.$$

We therefore have $x = x'$. In a sufficiently small ϵ-neigbourhood of x, the function f is Lipschitz continuous. Thus, there exists a constant $C > 0$ with

$$\forall \widetilde{x}, \widetilde{x}' \in X: \quad \max\big(d_X(x, \widetilde{x}), d_X(x, \widetilde{x}')\big) < \epsilon \Rightarrow d_Y\big(f(\widetilde{x}), f(\widetilde{x}')\big) \leq C d_X(\widetilde{x}, \widetilde{x}').$$

For sufficiently large $n \in \mathbb{N}$, we have $\max\big(d_X(x, x_n), d_X(x, x_n')\big) < \epsilon$, and hence

$$d_Y\big(f(\widetilde{x}), f(\widetilde{x}')\big) \leq C d_X(x_n, x_n').$$

This contradicts (A.2) and concludes the proof. □

Remark A.5. If X is even locally compact, a function $f : X \to Y$ is locally Lipschitz continuous if and only if $f|_K$ is Lipschitz continuous for each compact subset $K \subseteq X$.

Corollary A.6. *Let $\gamma : [a, b] \to \Gamma$ be a regular closed curve. Then, for all $r \in \mathbb{R}$, $\gamma|^{-1}_{(r+a, r+b)}$ is Lipschitz continuous on each compact subset of $\Gamma \setminus \{\gamma(r+a)\}$.*

Proof. This is an immediate consequence of Theorem A.3 and Lemma A.4. □

Lemma A.7. *Each regular closed curve $\gamma : [a,b] \to \Gamma$ with length L is equivalent to a unique regular closed curve $\gamma_L : [0,L] \to \Gamma$ with $|\gamma_L'^{\ell}(t)| = 1$ for $t \in (0,L]$ and $|\gamma_L'^{r}(t)| = 1$ for $t \in [0,L)$.*

Proof. We define the parameter transformation

$$\psi : [a,b] \to [0,L] : t \mapsto \int_a^t |\gamma'(\tau)| \, \mathrm{d}\tau.$$

Then the inverse $\varphi := \psi$ is also a parameter transformation. It holds for $t \in [0,L]$

$$(\gamma \circ \varphi)'^{\ell,r}(t) = \gamma'^{\ell,r}\big(\varphi(t)\big)\varphi'^{\ell,r}(t) = \gamma'^{\ell,r}\big(\varphi(t)\big)\frac{1}{\psi'^{\ell,r}\big(\varphi(t)\big)} = \frac{\gamma'^{\ell,r}\big(\varphi(t)\big)}{\big|\gamma'^{\ell,r}\big(\varphi(t)\big)\big|}.$$

Hence $\gamma_L := \gamma \circ \varphi$ has the desired properties.

The uniqueness of γ_L resp. the corresponding parameter transformation φ follows from the uniqueness of the inverse ψ. The inverse is unique because for $t \in [a,b]$

$$|\gamma'^{\ell,r}(t)| = \big|\gamma_L'^{\ell,r}\big(\psi(t)\big)\big||\psi'^{\ell,r}(t)| = \psi'^{\ell,r}(t) \qquad \text{and} \qquad \psi(a) = 0.$$

This concludes the proof. □

Appendix B. Implementation

In this section, the generation of the Galerkin matrix for Symm's integral equation V_h and of the right-hand side vector F_h of Section 5 is implemented for the special case that the parametrization γ is a NURBS curve. This means

$$\gamma(t) = \sum_{i\in\mathbb{Z}} C_i^\gamma R_{i,p_\gamma}^{\check{\mathcal{K}}^\gamma,\mathcal{W}^\gamma}(t) \tag{B.1}$$

for all $t \in [a,b]$. Here, $p_\gamma \in \mathbb{N}$ is the polynomial degree, $\check{\mathcal{K}}^\gamma$ and $\mathcal{W}^\gamma$ are periodic knots and weights of length N_γ as in Definition 4.13, and $(C_i)_{i\in\mathbb{Z}}$ are N_γ-periodic control points in $\mathbb{R}^2$. Moreover, we implement the Faermann estimator $\eta_h^{F_1}$ for the corresponding solution, Dörfler marking, and a refinement procedure which refines all marked elements such that the shape regularity constant stays bounded.

B.1. `Functions.h`. In this file we define some help functions and the Dirichlet boundary conditions g.

```
#ifndef _Functions_h
#define _Functions_h

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define EPS 1e-15


inline int mod(int x, int y){
    // returns x mod y for integer x and positive integer y
    int z=x%y;
    if (z<0) {z=z+y;}
    return z;
}


inline int floordiv(int x, int y) {
    // returns floor(((double)x)/((double)y)) for int x and pos. int y
    if (x >= 0) {return x/y;}else{return (x-y+1)/y;}
}


inline double min(double x,double y){
    if (x<=y){return x;}else{return y;}
}


inline double max(double x, double y){
    if (x<=y){return y;}else{return x;};
}


inline int nearly_equal(double x,double y){
```

```
    if (fabs(x-y)<EPS*max(fabs(x),fabs(y))){
        return 1;
    }else{
        return 0;
    }
}


inline double norm(double* vector){
    return sqrt(vector[0]*vector[0]+vector[1]*vector[1]);
}


inline double g(double* x){
    // returns g(x) for x in Gamma

    // quadratic:
    return x[0]*x[0]-x[1]*x[1]+2*x[0]*x[1];

    // exponential:
    // return exp(x[0])*cos(x[1]);

    // singular:
    // double tau=4.0/7.0;
    // return pow(norm(x),tau)*cos(tau*atan2(x[1],x[0]));
}


#endif
```

B.2. `Structures.h`. In this file we define the structures `NURBSData` and `QuadData`.

```
#ifndef _Structures_h_
#define _Structures_h_
#include <stdio.h>
#include <stdlib.h>


// NURBSData
typedef struct _NURBSData_{
    double a; // left point of real interval [a,b]
    double* knots; // non-decreasing points in (a,b],
    //t_1=knots[0],...,t_N=knots[N-1]=b with #t_i<=p+1
    int N; // number of t_i in (a,b
    double* weights; // positive weights correspoding to knots
    int p; // polynomial degree
    double* nodes; // unique(knots)
    int n; // number of nodes
    int is_rational; // 0 if all weights are one, 1 else
}NURBSData;


```

```
21 NURBSData* new_NURBSData(double a,double* knots,int N,double* weights,int p,
22                          double* nodes,int n,int is_rational) {
23     NURBSData* Data = malloc(sizeof(NURBSData));
24 
25     Data->a = a;
26     Data->knots = knots;
27     Data->N = N;
28     Data->weights = weights;
29     Data->p = p;
30     Data->nodes=nodes;
31     Data->n=n;
32     Data->is_rational=is_rational;
33 
34     return Data;
35 }
36 
37 
38 NURBSData* del_NURBSData(NURBSData* Data) {
39     if (Data != NULL) {
40         free(Data);
41     }
42     return NULL;
43 }
44 
45 
46 // get NURBSData
47 inline double get_NURBSData_a(NURBSData* Data){return Data->a;}
48 inline double* get_NURBSData_knots(NURBSData* Data){return Data->knots;}
49 inline int get_NURBSData_N(NURBSData* Data){return Data->N;}
50 inline double* get_NURBSData_weights(NURBSData* Data){return Data->weights;}
51 inline int get_NURBSData_p(NURBSData* Data){return Data->p;}
52 inline double* get_NURBSData_nodes(NURBSData* Data){return Data->nodes;}
53 inline int get_NURBSData_n(NURBSData* Data){return Data->n;}
54 inline int get_NURBSData_is_rational(NURBSData* Data)
55 {return Data->is_rational;}
56 
57 
58 
59 
60 // QuadData
61 typedef struct _QuadData_{
62     double* nodes; // nodes of Gauss quadrature in [0,1]
63     double* weights; // weights of Gauss quadrature in [0,1]
64     int n; // number of nodes resp. weights
65 }QuadData;
66 
67 
68 QuadData* new_QuadData(double* nodes,double* weights,int n) {
69     QuadData* Data = malloc(sizeof(QuadData));
70 
71     Data->nodes = nodes;
72     Data->weights = weights;
73     Data->n = n;
```

```
74
75      return Data;
76  }
77
78
79  QuadData* del_QuadData(QuadData* Data) {
80      if (Data != NULL) {
81          free(Data);
82      }
83      return NULL;
84  }
85
86
87  // get QuadData
88  inline double* get_QuadData_nodes(QuadData* Data){return Data->nodes;}
89  inline double* get_QuadData_weights(QuadData* Data){return Data->weights;}
90  inline int get_QuadData_n(QuadData* Data){return Data->n;}
91
92  #endif
```

B.3. `Spline.h` and `Spline.c`. In `Spline.c` the evaluation of NURBS and its first derivative is implemented. It is mainly based on [dB86, Algorithm 9 and Algorithm 10].

```
1   #ifndef _Spline_h
2   #define _Spline_h
3
4   #include "Structures.h"
5   #include "Functions.h"
6
7   /* parameters:
8    Data...NURBSData*, see Structures.h
9    i...arbitrary unfixed integer
10   t...arbitrary unfixed evaluation point in [a,b)
11   wcpoints1...first component of weighted control points w_i*C_i
12   for geometry corresponding to knots
13   wcpoints2...second component of weighted control points w_i*C_i
14   for geometry corresponding to knots
15   */
16
17
18  inline double knotseq(NURBSData* Data,int i);
19  // returns t_i, (knots (b-a)-periodic extended)
20
21  int find_Span(NURBSData* Data, double t);
22  // returns integer index between 0 and N-1 with t in [t_index,t_{index+1})
23  // using binary search
24
25  double deBoor_help(NURBSData* Data,int i, double* coeffdata,int coefftype,
26                     int l,double t);
27  // help function for function deBoor which returns
28  // a_i^[l](t)=(1-\beta_{i,p-l+1}(t))*a_{i-1}^[l-1](t)
```

```
// +\beta_{i,p-l+1}(t)*a_i^[l-1](t), where a_i^[0]=c_{i+1}
// (see B-Spline Basics: Algorithm 9 with k=p+1 and a_i=c_{i+1}),
// for parameter description see function deBoor


double deBoorDeriv_help(NURBSData* Data, int i, double* coeffdata,
                        int coefftype,int l,double t);
// help function for function deBoorDeriv which returns
// (not derivative of function a_i^[l])
// a_i'^[l](t)=(1-\beta_{i,p-l})*a_{i-1}'^[l-1]+\beta_{i,p-l}*a_i'^[l-1] with
// a_i'^[0]=a_i'=p*(a_i-a_{i-1})/(t_{i+p}-t_i)
// (see B-Spline Basics: Algorithm 9 and Algorithm 10 with k=p+1 and
// a_i=c_{i+1}), for parameter description see function deBoorDeriv


double deBoor(NURBSData* Data, double* coeffdata, int coefftype,double t);
// returns sum_{i in Z}c_i*B_{i,p}(t) using deBoor algorithm, where:
// if coefftype=0: coeffdata[i+p-1]=c_i for i=(1-p)...(N-#b+1)
// if coefftype=1: coeffdata[i-1]=c_i for i=1...N, c_i N-periodic extended
// if coefftype=2: coeffdata[0] is unique integer i where c_i!=0, it is c_i=1

double deBoorDeriv(NURBSData* Data,double* coeffdata,int coefftype,double t);
// returns sum_{i in Z}c_i*D^r(B_{i,p})(t) using deBoor algorithm and formula
// for derivative coefficients c_i', where:
// if coefftype=0: coeffdata[i+p-1]=c_i for i=(1-p)...(N-#b+1)
// if coefftype=1: coeffdata[i-1]=c_i for i=1...N, c_i N-periodic extended
// if coefftype=2: coeffdata[0] is unique integer i where c_i!=0, it is c_i=1

double eval_NURBS(NURBSData* Data, int i, double t);
// returns R_{i,p}(t)


double eval_NURBSDeriv(NURBSData* Data,int i, double t);
// returns D^r(R_{i,p})(t) by using quotient rule


void eval_NURBSCurve(double* output, NURBSData* Data, double* wcpoints1,
                     double* wcpoints2, double t);
// turns output[0] resp. output[1] into first resp. second coordinate of
// sum_{l in Z} C_l*R_{l,p}(t)
// =(sum_{l in Z} w_l*C_l*B_{l,p}(t))/(sum_{l in Z} w_l*B_{l,p}(t))

void eval_NURBSCurveDeriv(double* output, NURBSData* Data, double* wcpoints1,
                          double* wcpoints2, double t);
// turns output[0] resp. output[1] into first resp. second coordinate of
// right derivative of
// sum_{l in Z} C_l*R_{l,p}(t)
// =(sum_{l in Z} w_l*C_l*B_{l,p}(t))/(sum_{l in Z} w_l*B_{l,p}(t))
// using quotient rule

double eval_NURBSComb(NURBSData* Data,double* wcoeffs, double t);
// returns sum_{l in Z} c_l*R_{l,p}(t), where wcoeffs[l+p-1]=w_l*c_l
// for l=(1-p),...,(N-#b+1)
```

```
82
83
84 #endif
```

```
1  #include "Spline.h"
2
3
4  inline double knotseq(NURBSData* Data,int i){
5
6      double a= get_NURBSData_a(Data);
7      double* knots=get_NURBSData_knots(Data);
8      int N=get_NURBSData_N(Data);
9      double b=knots[N-1];
10
11     // t_i=t_{i mod N}+(b-a)*floor(i/N)
12     if (mod(i,N)!=0) {
13         return knots[mod(i,N)-1]+(b-a)*floordiv(i,N);
14     } else {
15         return a+(b-a)*floordiv(i,N); // a=t_0
16     }
17 }
18
19
20 int find_Span(NURBSData* Data, double t){
21
22     int N=get_NURBSData_N(Data);
23     int low=0, high=N+1; // t in [t_low,t_high)
24     int index=(low+high)/2;
25     double t_index=knotseq(Data,index); // t_index
26     double t_indexp1=knotseq(Data,index+1); // t_{index+1}
27
28     while (t<t_index || t>=t_indexp1) {
29         if (t<t_index) {
30             high=index;
31         } else {
32             low=index;
33         }
34         index=(low+high)/2;
35         t_index=knotseq(Data,index);
36         t_indexp1=knotseq(Data,index+1);
37     }
38     return index;
39 }
40
41
42 double deBoor_help(NURBSData* Data,int i, double* coeffdata,int coefftype,
43                    int l,double t){
44
45     int N=get_NURBSData_N(Data);
46     int p=get_NURBSData_p(Data);
47     int k,m;
```

```
48      double beta; // beta_{k,p-m+1}
49      double t_k,t_kppmmp1; // t_k, t_{k+p-m+1}
50      double A[l+1]; // help vector
51
52      // calculation of a_k^[0]=a_k=c_{k+1}=A[k+l-i]
53      for (k=(i-l);k<=i;k=k+1){
54          switch(coefftype){
55              case 0:
56                  A[k+l-i]=coeffdata[k+p]; break;
57              case 1:
58                  if (mod(k+1,N)!=0) {
59                      A[k+l-i]=coeffdata[mod(k+1,N)-1];
60                  } else {
61                      A[k+l-i]=coeffdata[N-1];
62                  } break;
63              case 2:
64                  // c_{k+1}=Δ_{k+1,coeffdata[0]}
65                  if ( ((int) coeffdata[0]) == (k+1) ){
66                      A[k+l-i]=1;
67                  } else {
68                      A[k+l-i]=0;
69                  }break;
70          }
71      }
72
73      // calculation of a_k^[m]=A[k+l-i-m]
74      for (m=1;m<=l;m=m+1){
75          for (k=(i-l+m);k<=i;k=k+1){
76              t_k=knotseq(Data,k);
77              t_kppmmp1=knotseq(Data,k+p-m+1);
78              // \beta_{k,p-m+1}
79              if (!(nearly_equal(t_k,t_kppmmp1))) {
80                  beta=(t-t_k)/(t_kppmmp1-t_k);
81              } else {
82                  beta=0;
83              }
84              // a_k^[m]=(1-\beta_{k,p-m+1})*a_{k-1}^[m-1]
85              // + \beta_{k,p-m+1}*a_k^[m-1]
86              A[k+l-i-m]=(1-beta)*A[k+l-i-m] + beta*A[k+l-i-m+1];
87          }
88      }
89
90      return A[0];
91  }
92
93
94  double deBoorDeriv_help(NURBSData* Data, int i, double* coeffdata,
95                          int coefftype,int l,double t){
96
97      int N=get_NURBSData_N(Data);
98      int p=get_NURBSData_p(Data);
99      int k,m;
100     double beta; // beta_{k,p-m}
```

```
101     double t_k,t_kppmm,t_kpp; // t_k, t_{k+p-m}, t_{k+p}
102     double tmp1,tmp2;
103     double A_prime[l+1]; // help vector A'
104
105
106     // calculation of a_k'^[0]=a_k'=c_{k+1}'=A'[k+l-i]
107     for (k=(i-l);k<=i;k=k+1){
108         t_k=knotseq(Data,k);
109         t_kpp=knotseq(Data,k+p);
110         switch(coefftype){
111             case 0:
112                 if (nearly_equal(t_kpp,t_k)){
113                     A_prime[k+l-i]= 0;
114                 }else{
115                     A_prime[k+l-i]=p*(coeffdata[k+p]-coeffdata[k+p-1])
116                     /(t_kpp-t_k);
117                 } break;
118             case 1:
119                 if (mod(k,N)!=0) {
120                     tmp1=coeffdata[mod(k,N)-1]; // c_k
121                 } else {
122                     tmp1=coeffdata[N-1]; // c_k
123                 }
124                 if (mod(k+1,N)!=0) {
125                     tmp2=coeffdata[mod(k+1,N)-1]; // c_{k+1}
126                 } else {
127                     tmp2=coeffdata[N-1]; // c_{k+1}
128                 }
129                 if (nearly_equal(t_kpp,t_k)){
130                     A_prime[k+l-i]= 0;
131                 } else {
132                     A_prime[k+l-i]=p*(tmp2-tmp1)/(t_kpp-t_k);
133                 } break;
134             case 2:
135                 // c_{k+1}=Δ_{k+1,coeffdata[0]}
136                 if (((int) coeffdata[0]) == (k+1)){
137                     if (nearly_equal(t_kpp,t_k)){
138                         A_prime[k+l-i]=0;
139                     }else{
140                         A_prime[k+l-i]=p/(t_kpp-t_k);
141                     }
142                 } else {
143                     if (((int) coeffdata[0]) == k){
144                         if (nearly_equal(t_kpp,t_k)){
145                             A_prime[k+l-i]= 0;
146                         }else{
147                             A_prime[k+l-i]= -p/(t_kpp-t_k);
148                         }
149                     } else {
150                         A_prime[k+l-i]=0;
151                     }
152                 } break;
153         }
```

```
154     }
155
156     // calculation of a_k'^[m]=A'[k+l-i-m]
157     for (m=1;m<=l;m=m+1){
158         for (k=(i-l+m);k<=i;k=k+1){
159             t_k=knotseq(Data,k);
160             t_kppmm=knotseq(Data,k+p-m);
161             // \beta_{k,p-m}
162             if (!(nearly_equal(t_k,t_kppmm))) {
163                 beta=(t-t_k)/(t_kppmm-t_k);
164             } else {
165                 beta=0;
166             }
167             // a_k'^[m]=(1-\beta_{k,p-m})*a_{k-1}'^[m-1]
168             // +\beta_{k,p-m}*a_k'^[m-1]
169             A_prime[k+l-i-m]=(1-beta)*A_prime[k+l-i-m]
170             + beta*A_prime[k+l-i-m+1];
171         }
172     }
173
174     return A_prime[0];
175 }
176
177
178 double deBoor(NURBSData* Data, double* coeffdata, int coefftype,double t){
179
180     int p=get_NURBSData_p(Data);
181     double index=find_Span(Data,t);
182
183     return deBoor_help(Data,index,coeffdata,coefftype,p,t);
184 }
185
186
187 double deBoorDeriv(NURBSData* Data,double* coeffdata,int coefftype,double t){
188
189     int p=get_NURBSData_p(Data);
190     double index=find_Span(Data,t);
191
192     return deBoorDeriv_help(Data,index,coeffdata,coefftype,p-1,t);
193 }
194
195
196 double eval_NURBS(NURBSData* Data, int i, double t){
197
198     int N=get_NURBSData_N(Data);
199     double* weights=get_NURBSData_weights(Data);
200     int is_rational=get_NURBSData_is_rational(Data);
201     double weight, nominator,denominator;
202     double coeffdata[1]={(double)i};
203
204     if (is_rational==0){
205         return deBoor(Data,coeffdata,2,t);
206     }else{
```

```
207         if (mod(i,N)!=0) {
208             weight=weights[mod(i,N)-1]; // w_i
209         } else {
210             weight=weights[N-1]; // w_i
211         }
212         nominator=weight*deBoor(Data,coeffdata,2,t);
213         denominator=deBoor(Data,weights,1,t);
214         return nominator/denominator;
215     }
216 }
217
218
219 double eval_NURBSDeriv(NURBSData* Data,int i, double t){
220
221     int N=get_NURBSData_N(Data);
222     double* weights=get_NURBSData_weights(Data);
223     int is_rational=get_NURBSData_is_rational(Data);
224     double weight, nominator, nominator_prime,denominator, denominator_prime;
225     double coeffdata[1]={(double)i};
226
227     if (is_rational==0){
228         return deBoorDeriv(Data,coeffdata,2,t);
229     }else{
230         if (mod(i,N)!=0) {
231             weight=weights[mod(i,N)-1]; // w_i
232         } else {
233             weight=weights[N-1]; // w_i
234         }
235         nominator=weight*deBoor(Data,coeffdata,2,t);
236         nominator_prime=weight*deBoorDeriv(Data,coeffdata,2,t);
237         denominator=deBoor(Data,weights,1,t);
238         denominator_prime=deBoorDeriv(Data,weights,1,t);
239         return (nominator_prime*denominator-nominator*denominator_prime)
240         /(denominator*denominator);
241     }
242 }
243
244
245 void eval_NURBSCurve(double* output, NURBSData* Data, double* wcpoints1,
246                      double* wcpoints2, double t){
247
248     int N=get_NURBSData_N(Data);
249     double* weights=get_NURBSData_weights(Data);
250     int is_rational=get_NURBSData_is_rational(Data);
251     double nominator1, nominator2, denominator;
252
253     if (is_rational==0){
254         output[0]=deBoor(Data,wcpoints1,1,t);
255         output[1]=deBoor(Data,wcpoints2,1,t);
256     }else{
257         nominator1=deBoor(Data,wcpoints1,1,t);
258         nominator2=deBoor(Data,wcpoints2,1,t);
259         denominator=deBoor(Data,weights,1,t);
```

```
260
261         output[0]=nominator1/denominator;
262         output[1]=nominator2/denominator;
263     }
264 }
265
266
267 void eval_NURBSCurveDeriv(double* output, NURBSData* Data, double* wcpoints1,
268                           double* wcpoints2, double t){
269
270     int N=get_NURBSData_N(Data);
271     double* weights=get_NURBSData_weights(Data);
272     int is_rational=get_NURBSData_is_rational(Data);
273     double nominator1,nominator2,denominator;
274     double nominator1_prime,nominator2_prime,denominator_prime;
275
276     if (is_rational==0){
277         output[0]=deBoorDeriv(Data,wcpoints1,1,t);
278         output[1]=deBoorDeriv(Data,wcpoints2,1,t);
279     }else{
280         nominator1=deBoor(Data,wcpoints1,1,t);
281         nominator2=deBoor(Data,wcpoints2,1,t);
282         denominator=deBoor(Data,weights,1,t);
283         nominator1_prime=deBoorDeriv(Data,wcpoints1,1,t);
284         nominator2_prime=deBoorDeriv(Data,wcpoints2,1,t);
285         denominator_prime=deBoorDeriv(Data,weights,1,t);
286
287         output[0]=(nominator1_prime*denominator-nominator1*denominator_prime)
288         /(denominator*denominator);
289         output[1]=(nominator2_prime*denominator-nominator2*denominator_prime)
290         /(denominator*denominator);
291     }
292 }
293
294
295 double eval_NURBSComb(NURBSData* Data,double* wcoeffs, double t) {
296
297     double* knots=get_NURBSData_knots(Data);
298     int N=get_NURBSData_N(Data);
299     double* weights=get_NURBSData_weights(Data);
300     int p=get_NURBSData_p(Data);
301     int is_rational=get_NURBSData_is_rational(Data);
302     double b=knots[N-1];
303     int multb=0; // #b
304     while (nearly_equal(knots[N-multb-1],b)) {multb=multb+1;}
305     double weight,nominator,denominator;
306
307     if (is_rational==0){
308         return deBoor(Data,wcoeffs,0,t);
309     }else{
310         nominator=deBoor(Data,wcoeffs,0,t);
311         denominator=deBoor(Data,weights,1,t);
312         return nominator/denominator;
```

```
    }
}
```

B.4. `Vmatrix.h` **and** `Vmatrix.c`**.** The computation of the Galerkin matrix is based on Subsection 5.1.

```
#ifndef _Vmatrix_h
#define _Vmatrix_h

#include <math.h>
#include <stdio.h>
#include "Spline.h"

/* parameters:
 Data_Gamma...NURBSData* for geometry Gamma, see Structures.h
 wcpoints1_gam...first component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 wcpoints2_gam...second component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 Data_Basis...NURBSData* for Basis of approximation space
 Data_Gauss...QuadData* for quadrature with weight function 1 om [0,1],
 see Structures.h
 Data_LogGauss...QuadData* for quadrature with weight function -log(x)
 on [0,1], see Structures.h

 comments:
 we assume that:
 -)path gamma induced by Data_Gammais and wcpoints_gam is positively
 orientated regular closed curve,
 which parametizes boundary Gamma of Lipschitz domain Omega with diam(Omega)<1
 -)#t_i^gamma<=p_gam+1
 -)number of different entries in knots of Data_Gamma >= 4
 -){t_i^gamma:i=1...N_gam}<={t_i:i=1...N}
 -)#t_i<=p+1
 */


double SquareIntegrand_V_smooth(NURBSData* Data_Gamma,double* wcpoints1_gam,
                                double* wcpoints2_gam,NURBSData* Data_Basis,
                                double s,double t,int i,int k,
                                double denominator,double Jdet);
// returns -1/2pi*log(|\gamma(s)-\gamma(t)|/denominator)
// *\tilde{R}_i(s)*\tilde{R}_k(t)*Jdet,
// where s!=t in [a,b), i,k in {1-p,...,N-#b+1}, denominator>0 and Jdet in \R


double SquareIntegrand_V_log(NURBSData* Data_Gamma,double* wcpoints1_gam,
                             double* wcpoints2_gam,NURBSData* Data_Basis,
                             double s,double t,int i,int k,double Jdet);
// returns 1/2pi*\tilde{R}_i(s)*\tilde{R}_k(t)*Jdet,
// where s!=t in [a,b), i,k in {1-p,...,N-#b+1} and Jdet in \R
```

```
46
47
48 double SquareIntegral_V_Identical(NURBSData* Data_Gamma,double* wcpoints1_gam,
49                                    double* wcpoints2_gam,NURBSData* Data_Basis,
50                                    QuadData* Data_Gauss,QuadData* Data_LogGauss,
51                                    int i,int k,int l);
52 // returns \int_0^1 \int_0^1 \check{G}_{l,l}(s,t) \tilde{R}_{i,l}(s)
53 // \tilde{R}_{k,l}(t) dt ds,
54 // where i,k in {1-p,...,N-#b+1} and
55 // l in {max(i,1),...,min(i+p,N)} \cap {max(k,1)...,min(k+p,N)} with H_l>0
56
57
58 double SquareIntegral_V_Adjacent(NURBSData* Data_Gamma,double* wcpoints1_gam,
59                                  double* wcpoints2_gam,NURBSData* Data_Basis,
60                                  QuadData* Data_Gauss,QuadData* Data_LogGauss,
61                                  int i,int k,int l1,int l2);
62 // returns \int_0^1 \int_0^1 \check{G}_{l1,l2}(s,t)
63 // \tilde{R}_{i,l1}(s) \tilde{R}_{k,l2}(t) dt ds for adjacent elements,
64 // i.e. \gamma([t_{l_1-1},t_{l_1} \cap \gamma([t_{l_2-1},t_{l_2}]) consists
65 // of one point, singularity at s=0 and t=1,
66 // i,k in {1-p,...,N-#b+1}, l1 in {max(i,1),...,min(i+p,N)} and
67 // l2 in {max(k,1),...,min(k+p,N)} with min(H_l1,H_l2)>0
68
69
70 void build_Vmatrix(double* output,NURBSData* Data_Gamma,double* wcpoints1_gam,
71                    double* wcpoints2_gam,NURBSData* Data_Basis,
72                    QuadData* Data_Gauss,QuadData* Data_LogGauss);
73 // turns output[(i+p-1)+(k+p-1)*(N-#b+1+p)] into
74 // <V \hat{R}_i,\hat{R}_k>_{L^2(Gamma)} for i,k=1-p...N-#b+1,
75 // \hat{R}_i=R_{i,p} circ gamma^(-1) are the transformed basis functions
76
77 #endif
```

```
1  #include "Vmatrix.h"
2
3
4  double SquareIntegrand_V_smooth(NURBSData* Data_Gamma,double* wcpoints1_gam,
5                                  double* wcpoints2_gam,NURBSData* Data_Basis,
6                                  double s,double t,int i,int k,
7                                  double denominator,double Jdet){
8
9      double tmp1[2];
10     double tmp2[2];
11     double diff_gam[2];
12     double R_til_i,R_til_k; // \tilde{R}_i(s), \tilde{R}_k(t)
13
14     eval_NURBSCurve(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
15     // gamma(s)
16     eval_NURBSCurve(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
17     // gamma(t)
18     diff_gam[0] = tmp1[0] - tmp2[0];
```

```
19      diff_gam[1] = tmp1[1] - tmp2[1];
20      eval_NURBSCurveDeriv(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
21      // gamma'(s)
22      eval_NURBSCurveDeriv(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
23      // gamma'(t)
24      R_til_i = eval_NURBS(Data_Basis, i, s) * norm(tmp1);
25      R_til_k = eval_NURBS(Data_Basis, k, t) * norm(tmp2);
26      return -log(norm(diff_gam)/denominator) / (2*M_PI)*R_til_i *R_til_k*Jdet;
27  }
28
29
30  double SquareIntegrand_V_log(NURBSData* Data_Gamma,double* wcpoints1_gam,
31                               double* wcpoints2_gam,NURBSData* Data_Basis,
32                               double s,double t,int i,int k,double Jdet){
33
34
35      double tmp1[2];
36      double tmp2[2];
37      double R_til_i,R_til_k; // \tilde{R}_i(s), \tilde{R}_k(t)
38
39      eval_NURBSCurveDeriv(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
40      // gamma'(s)
41      eval_NURBSCurveDeriv(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
42      // gamma'(t)
43      R_til_i = eval_NURBS(Data_Basis, i, s) * norm(tmp1);
44      R_til_k = eval_NURBS(Data_Basis, k, t) * norm(tmp2);
45      return R_til_i *R_til_k*Jdet/(2*M_PI);
46  }
47
48
49  double SquareIntegral_V_Identical(NURBSData* Data_Gamma,double* wcpoints1_gam,
50                                    double* wcpoints2_gam,NURBSData* Data_Basis,
51                                    QuadData* Data_Gauss,QuadData* Data_LogGauss,
52                                    int i,int k,int l){
53
54      double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
55      double* weights_gauss=get_QuadData_weights(Data_Gauss);
56      int n_gauss=get_QuadData_n(Data_Gauss);
57      double* nodes_loggauss=get_QuadData_nodes(Data_LogGauss);
58      double* weights_loggauss=get_QuadData_weights(Data_LogGauss);
59      int n_loggauss=get_QuadData_n(Data_LogGauss);
60      int q1,q2;
61      double t_lm1=knotseq(Data_Basis,l-1); // t_{l-1}
62      double t_l=knotseq(Data_Basis,l); // t_{l}
63      double H_l=t_l-t_lm1; // H_l
64      double squareint=0; // integral over square
65      double intpoint1, intpoint2; // first and second integration point
66      double denominator;
67      double Jdet; // Jacobi determinant for Duffy transformation
68
69      // smooth integrals
70      for (q1=0;q1<n_gauss;q1=q1+1) {
71          for (q2=0;q2<n_gauss;q2=q2+1) {
```

```
72              // first double integral
73              intpoint1=t_lm1+H_l*nodes_gauss[q1];
74              intpoint2 = t_lm1 + H_l*(nodes_gauss[q1] * (1-nodes_gauss[q2]));
75              denominator=nodes_gauss[q1]*nodes_gauss[q2];
76              Jdet=nodes_gauss[q1];
77              squareint += weights_gauss[q1] * weights_gauss[q2]
78              *SquareIntegrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
79                                        Data_Basis,intpoint1,intpoint2,i,k,
80                                        denominator,Jdet);
81              // second double integral
82              intpoint1=t_lm1+H_l*(1-nodes_gauss[q1]);
83              intpoint2 = t_lm1 + H_l * (1+nodes_gauss[q1]*(nodes_gauss[q2]-1));
84              denominator=nodes_gauss[q1]*nodes_gauss[q2];
85              Jdet=nodes_gauss[q1];
86              squareint += weights_gauss[q1]*weights_gauss[q2]
87              *SquareIntegrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
88                                        Data_Basis,intpoint1,intpoint2,i,k,
89                                        denominator,Jdet);
90          }
91      }
92      // integrals with s-logarithmic singularity
93      for (q1=0;q1<n_loggauss;q1=q1+1) {
94          for (q2=0;q2<n_gauss;q2=q2+1) {
95              // first double integral
96              intpoint1=t_lm1+H_l*nodes_loggauss[q1];
97              intpoint2 = t_lm1 + H_l*(nodes_loggauss[q1]*(1-nodes_gauss[q2]));
98              Jdet=nodes_loggauss[q1];
99              squareint += weights_loggauss[q1]*weights_gauss[q2]
100             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
101                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
102             // second double integral
103             intpoint1=t_lm1+H_l*(1-nodes_loggauss[q1]);
104             intpoint2 = t_lm1+H_l*(1+nodes_loggauss[q1]*(nodes_gauss[q2]-1));
105             Jdet=nodes_loggauss[q1];
106             squareint += weights_loggauss[q1]*weights_gauss[q2]
107             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
108                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
109         }
110     }
111     // integrals with t-logarithmic singularity
112     for (q1=0;q1<n_gauss;q1=q1+1) {
113         for (q2=0;q2<n_loggauss;q2=q2+1) {
114             // first double integral
115             intpoint1=t_lm1+H_l*nodes_gauss[q1];
116             intpoint2 = t_lm1+H_l*(nodes_gauss[q1]*(1-nodes_loggauss[q2]));
117             Jdet=nodes_gauss[q1];
118             squareint += weights_gauss[q1]*weights_loggauss[q2]
119             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
120                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
121             // second double integral
122             intpoint1=t_lm1+H_l*(1-nodes_gauss[q1]);
123             intpoint2 = t_lm1+H_l*(1+nodes_gauss[q1]*(nodes_loggauss[q2]-1));
124             Jdet=nodes_gauss[q1];
```

```
125             squareint += weights_gauss[q1]*weights_loggauss[q2]
126             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
127                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
128         }
129     }
130     return squareint;
131 }
132
133
134 double SquareIntegral_V_Adjacent(NURBSData* Data_Gamma,double* wcpoints1_gam,
135                                  double* wcpoints2_gam,NURBSData* Data_Basis,
136                                  QuadData* Data_Gauss,QuadData* Data_LogGauss,
137                                  int i,int k,int l1,int l2){
138
139
140     double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
141     double* weights_gauss=get_QuadData_weights(Data_Gauss);
142     int n_gauss=get_QuadData_n(Data_Gauss);
143     double* nodes_loggauss=get_QuadData_nodes(Data_LogGauss);
144     double* weights_loggauss=get_QuadData_weights(Data_LogGauss);
145     int n_loggauss=get_QuadData_n(Data_LogGauss);
146     int q1,q2;
147     double t_l1m1=knotseq(Data_Basis,l1-1); // t_{l_1-1}
148     double t_l1=knotseq(Data_Basis,l1); // t_{l_1}
149     double H_l1=t_l1-t_l1m1; // H_{l_1}
150     double t_l2m1=knotseq(Data_Basis,l2-1); // t_{l_2-1}
151     double t_l2=knotseq(Data_Basis,l2); // t_{l_2}
152     double H_l2=t_l2-t_l2m1; // H_{l_1}
153     double squareint=0; // integral over square
154     double intpoint1, intpoint2; // first and second integration point
155     double denominator;
156     double Jdet; // Jacobi determinant of Duffy transformation
157
158     // smooth integrals
159     for (q1=0;q1<n_gauss;q1=q1+1) {
160         for (q2=0;q2<n_gauss;q2=q2+1) {
161             // first double integral
162             intpoint1=t_l1m1+H_l1*nodes_gauss[q1];
163             intpoint2 = t_l2m1 + H_l2 *  (1-nodes_gauss[q1]*nodes_gauss[q2]);
164             denominator=nodes_gauss[q1];
165             Jdet=nodes_gauss[q1];
166             squareint += weights_gauss[q1]*weights_gauss[q2]
167             *SquareIntegrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
168                                       Data_Basis,intpoint1,intpoint2,i,k,
169                                       denominator,Jdet);
170             // second double integral
171             intpoint1=t_l1m1+H_l1*nodes_gauss[q1]*nodes_gauss[q2];
172             intpoint2 = t_l2m1 + H_l2 * (1-nodes_gauss[q2]);
173             denominator=nodes_gauss[q2];
174             Jdet=nodes_gauss[q2];
175             squareint += weights_gauss[q1]*weights_gauss[q2]
176             *SquareIntegrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
177                                       Data_Basis,intpoint1,intpoint2,i,k,
```

```
178                                         denominator,Jdet);
179         }
180     }
181     // integral with s-logarithmic singularity
182     for (q1=0;q1<n_loggauss;q1=q1+1) {
183         for (q2=0;q2<n_gauss;q2=q2+1) {
184             intpoint1=t_l1m1+H_l1*nodes_loggauss[q1];
185             intpoint2 = t_l2m1+H_l2* (1-nodes_loggauss[q1]*nodes_gauss[q2]);
186             Jdet=nodes_loggauss[q1];
187             squareint += weights_loggauss[q1]*weights_gauss[q2]
188             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
189                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
190         }
191     }
192     // integral with t-logarithmic singularity
193     for (q1=0;q1<n_gauss;q1=q1+1) {
194         for (q2=0;q2<n_loggauss;q2=q2+1) {
195             intpoint1=t_l1m1+H_l1*nodes_gauss[q1]*nodes_loggauss[q2];
196             intpoint2 = t_l2m1 + H_l2 * (1-nodes_loggauss[q2]);
197             Jdet=nodes_loggauss[q2];
198             squareint += weights_gauss[q1]*weights_loggauss[q2]
199             *SquareIntegrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
200                                    Data_Basis,intpoint1,intpoint2,i,k,Jdet);
201         }
202     }
203     return squareint;
204 }
205
206
207 void build_Vmatrix(double* output,NURBSData* Data_Gamma,double* wcpoints1_gam,
208                    double* wcpoints2_gam,NURBSData* Data_Basis,
209                    QuadData* Data_Gauss,QuadData* Data_LogGauss){
210
211     double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
212     double* weights_gauss=get_QuadData_weights(Data_Gauss);
213     int n_gauss=get_QuadData_n(Data_Gauss);
214     double* nodes_loggauss=get_QuadData_nodes(Data_LogGauss);
215     double* weights_loggauss=get_QuadData_weights(Data_LogGauss);
216     int n_loggauss=get_QuadData_n(Data_LogGauss);
217     double* knots=get_NURBSData_knots(Data_Basis);
218     int N=get_NURBSData_N(Data_Basis);
219     int p=get_NURBSData_p(Data_Basis);
220     double tmp[2];
221     int i,k,l1,l2,q1,q2;
222     double b=knots[N-1];
223     int multb=0; // #b
224     while (nearly_equal(knotseq(Data_Basis,N-multb),b)) {multb=multb+1;}
225     double R_til[N-multb+1+p][p+1][n_gauss];
226     // R_til[i-1+p][l1-i][q1]=\tilde{R}_{i,l1}(nodes_gauss[q1])
227     double gamma1[N][n_gauss];
228     // gamma1[l1-1][q1] is first component of
229     // gamma(t_{l1-1}+H_l1*nodes_gauss[q1])
230     double gamma2[N][n_gauss];
```

```
231     // gamma2[l1-1][q1] is second component of
232     // gamma(t_{l1-1}+H_l1*nodes_gauss[q1])
233     double squareint; // integral over square
234     double t_l1m1,t_l1,H_l1,t_l2m1,t_l2,H_l2;
235     // t_{l1-1},t_l1,H_l1,t_{l2-1},t_l2,H_l2
236     double intpoint; // integration point
237
238
239     // calculation of R_til
240     // R_i
241     for (i=1-p;i<=(N-multb+1);i=i+1) {
242         // elements with nonemty intersection with support of R_i
243         for (l1=max(i,1);l1<=min(i+p,N);l1=l1+1) {
244             // quadrature points
245             for (q1=0;q1<n_gauss;q1=q1+1) {
246                 t_l1m1=knotseq(Data_Basis,l1-1);
247                 t_l1=knotseq(Data_Basis,l1);
248                 H_l1=t_l1-t_l1m1;
249                 intpoint=t_l1m1+H_l1*nodes_gauss[q1];
250                 eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,
251                                      wcpoints2_gam,intpoint);
252                 R_til[i-1+p][l1-i][q1]=eval_NURBS(Data_Basis, i,intpoint)
253                 *norm(tmp);
254             }
255         }
256     }
257     // calculation of gamma1, gamma2
258     for (l1=1;l1<=N;l1=l1+1){
259         for (q1=0;q1<n_gauss;q1=q1+1){
260             t_l1m1=knotseq(Data_Basis,l1-1);
261             t_l1=knotseq(Data_Basis,l1);
262             H_l1=t_l1-t_l1m1;
263             intpoint=t_l1m1+H_l1*nodes_gauss[q1];
264             eval_NURBSCurve(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,
265                             intpoint);
266             gamma1[l1-1][q1]=tmp[0];
267             gamma2[l1-1][q1]=tmp[1];
268         }
269     }
270
271
272     // calculation of Vmatrix
273     // R_i
274     for (i=1-p;i<=(N-multb+1);i=i+1) {
275         // R_k
276         for (k=1-p;k<=i;k=k+1) {
277             output[i+p-1+(k+p-1)*(N-multb+1+p)]=0;
278             // elements with nonemty intersection with support of R_i
279             for (l1=max(i,1);l1<=min(i+p,N);l1=l1+1) {
280                 // elements with nonemty intersection with support of R_k
281                 for (l2=max(k,1);l2<=min(k+p,N);l2=l2+1) {
282                     t_l1m1=knotseq(Data_Basis,l1-1);
283                     t_l1=knotseq(Data_Basis,l1);
```

```
284                     t_l2m1=knotseq(Data_Basis,l2-1);
285                     t_l2=knotseq(Data_Basis,l2);
286                     H_l1=t_l1-t_l1m1;
287                     H_l2=t_l2-t_l2m1;
288                     // quadrature
289                     if (0<min(H_l1,H_l2)){
290                         // elements with no intersection
291                         squareint=0;
292                         if ((!nearly_equal(t_l1m1,t_l2m1))
293                             && (!nearly_equal(t_l1m1,t_l2))
294                             && (!nearly_equal(t_l1,t_l2m1))
295                             && (!nearly_equal(t_l1,t_l2))
296                             && ((l1!=(N-multb+1)) || (l2!=1))
297                             && ((l2!=(N-multb+1)) || (l1!=1))) {
298                             for (q1=0;q1<n_gauss;q1=q1+1) {
299                                 for (q2=0;q2<n_gauss;q2=q2+1) {
300                                     tmp[0]=gamma1[l1-1][q1]
301                                     - gamma1[l2-1][q2];
302                                     tmp[1]=gamma2[l1-1][q1]
303                                     - gamma2[l2-1][q2];
304                                     squareint+=
305                                     -weights_gauss[q1]*weights_gauss[q2]
306                                     *log(norm(tmp))*R_til[i-1+p][l1-i][q1]
307                                     * R_til[k-1+p][l2-k][q2]/(2*M_PI);
308                                 }
309                             }
310                         }
311                         // elements with intersection
312                         else {
313                             // identical elements
314                             if (l1==l2){
315                                 squareint=SquareIntegral_V_Identical(
316                                 Data_Gamma,wcpoints1_gam,wcpoints2_gam,
317                                 Data_Basis,Data_Gauss,
318                                 Data_LogGauss,i,k,l1);
319                             }
320                             // adjacent elements
321                             else{
322                                 // singularity at s=0,t=1
323                                 if (nearly_equal(t_l1m1,t_l2)
324                                     || ((l2==(N-multb+1)) && (l1==1))){
325                                     squareint=SquareIntegral_V_Adjacent(
326                                     Data_Gamma,wcpoints1_gam,wcpoints2_gam,
327                                     Data_Basis,Data_Gauss,Data_LogGauss,
328                                     i,k,l1,l2);
329                                 }
330                                 // singularity at s=1,t=0
331                                 else{
332                                     squareint=SquareIntegral_V_Adjacent(
333                                     Data_Gamma,wcpoints1_gam,wcpoints2_gam,
334                                     Data_Basis,Data_Gauss,Data_LogGauss,
335                                     k,i,l2,l1);
336                                 }
```

```
                                }
                            }
                            output[i+p-1+(k+p-1)*(N-multb+1+p)]+=
                            H_l1*H_l2*squareint;
                        }
                    }
                }
                if (i!=k){
                    // V symmetric
                    output[k+p-1+(i+p-1)*(N-multb+1+p)] =
                    output[i+p-1+(k+p-1)*(N-multb+1+p)];
                }
            }
        }
}
```

B.5. `Fvector.h` **and** `Fvector.c`**.** The computation of the right-hand side vector is based on Subsection 5.2.

```
#ifndef _Fvector_h
#define _Fvector_h

#include <math.h>
#include <stdio.h>
#include "Spline.h"

/* parameters:
 Data_Gamma...NURBSData* for geometry Gamma, see Structures.h
 wcpoints1_gam...first component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 wcpoints2_gam...second component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 Data_Basis...NURBSData* for Basis of approximation space
 Data_Gauss...QuadData* for quadrature with weight function 1 on [0,1],
 see Structures.h
 Data_Gauss_small...QuadData* (see Structures.h) for quadrature with weight
 function 1 on [0,1] (with smaller number of nodes as Data_Gauss),
 used for quadrature for identical elements or adjacent elements
 in function build_Fvector

 comments:
 we assume that:
 -)path gamma induced by Data_Gammais and wcpoints_gam is positively
 orientated regular closed curve, which parametizes boundary Gamma
 of Lipschitz domain Omega with diam(Omega)<1
 -)#t_i^gamma<=p_gam+1
 -)number of different entries in knots of Data_Gamma >= 4
 -){t_i^gamma:i=1...N_gam}<={t_i:i=1...N}
 -)#t_i<=p+1
 */

```

```
double SquareIntegrand_K(NURBSData* Data_Gamma,double* wcpoints1_gam,
                         double* wcpoints2_gam,NURBSData* Data_Basis,
                         int i,double s,double t,double Jdet);
// returns \partial_\nu\check{G}(s,t)*\tilde{g}(t)*\tilde{R}_i(s)*Jdet,
// where s!=t in [a,b), i,k in {1-p,...,N-#b+1} and Jdet in \R


double SquareIntegral_K_Identical(NURBSData* Data_Gamma,double* wcpoints1_gam,
                                  double* wcpoints2_gam,NURBSData* Data_Basis,
                                  QuadData* Data_Gauss,
                                  int i,int l);
// returns \int_0^1 \int_0^1 \partial_\nu\check{G}_{l,l}(s,t)
// \tilde{R}_{i,l}(s) \tilde{g}(t) dt ds,
// where i in {1-p,...,N-#b+1} and l in {max(i,1),...,min(i+p,N)} with H_l>0


double SquareIntegral_K_Adjacent(NURBSData* Data_Gamma,double* wcpoints1_gam,
                                 double* wcpoints2_gam,NURBSData* Data_Basis,
                                 QuadData* Data_Gauss,
                                 int i,int l1,int l2,int singtype);
// returns \int_0^1 \int_0^1 \partial_nu\check{G}_{l1,l2}(s,t)
// \tilde{R}_{i,l1}(s) \tilde{g}(t) dt ds for adjacent elements,
// i.e. \gamma([t_{l_1-1},t_{l_1} \cap \gamma([t_{l_2-1},t_{l_2}]) consists
// of one point, if singtype=0: singularity at s=0 and t=1
// else singularity at s=1 and t=0,
// i in {1-p,...,N-#b+1}, l1 in {max(i,1),...,min(i+p,N)} and l2 in {1,...,N}
// with min(H_l1,H_l2)>0


void build_Fvector(double* output,NURBSData* Data_Gamma,double* wcpoints1_gam,
                   double* wcpoints2_gam,NURBSData* Data_Basis,
                   QuadData* Data_Gauss,QuadData* Data_Gauss_small);
// turns output[i+p-1] into <Kg+g/2,\hat{R}_i>_{L_2(Gamma)}
// for i=1-p...N-#b+1, \hat{R}_i=R_{i,p} \circ gamma^(-1) are the transformed
// basis functions


#endif
```

```
#include "Fvector.h"

double SquareIntegrand_K(NURBSData* Data_Gamma,double* wcpoints1_gam,
                         double* wcpoints2_gam,NURBSData* Data_Basis,
                         int i,double s,double t,double Jdet){

    double tmp1[2];
    double tmp2[2];
    double diff_gam[2]; // gamma(s)-gamma(t)
    double R_til_i,g_nu1,g_nu2; // \tilde{R}_{i,p}(s), first resp. second
    // coordinate of \check{g}(t)|\gamma'(t)|\nu(\gamma(t))

```

```
13      eval_NURBSCurve(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
14      // gamma(s)
15      eval_NURBSCurve(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
16      // gamma(t)
17      diff_gam[0] = tmp1[0] - tmp2[0];
18      diff_gam[1] = tmp1[1] - tmp2[1];
19      g_nu1 = g(tmp2);
20      g_nu2 = g(tmp2);
21      eval_NURBSCurveDeriv(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
22      // gamma'(s)
23      eval_NURBSCurveDeriv(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
24      // gamma'(t)
25      R_til_i = eval_NURBS(Data_Basis,i, s) * norm(tmp1);
26      g_nu1 *= tmp2[1];
27      g_nu2 *= -tmp2[0];
28      return ((diff_gam[0]*g_nu1 + diff_gam[1]*g_nu2) / (2*M_PI*norm(diff_gam)))
29      *(Jdet/norm(diff_gam)) * R_til_i;
30  }
31
32
33  double SquareIntegral_K_Identical(NURBSData* Data_Gamma,double* wcpoints1_gam,
34                                    double* wcpoints2_gam,NURBSData* Data_Basis,
35                                    QuadData* Data_Gauss,int i,int l){
36
37      double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
38      double* weights_gauss=get_QuadData_weights(Data_Gauss);
39      int n_gauss=get_QuadData_n(Data_Gauss);
40      int q1,q2;
41      double t_lm1=knotseq(Data_Basis,l-1); // t_{l-1}
42      double t_l=knotseq(Data_Basis,l); // t_{l}
43      double H_l=t_l-t_lm1; // H_l
44      double squareint=0; // integral over square
45      double intpoint1, intpoint2; // first and second integration point
46      double Jdet; // Jacobi determinant for Duffy transformation
47
48      for (q1=0;q1<n_gauss;q1=q1+1) {
49          for (q2=0;q2<n_gauss;q2=q2+1) {
50              // first double integral
51              intpoint1=t_lm1+H_l*nodes_gauss[q1];
52              intpoint2 = t_lm1 + H_l * nodes_gauss[q1] * (1 - nodes_gauss[q2]);
53              Jdet=nodes_gauss[q1];
54              squareint+=weights_gauss[q1]*weights_gauss[q2]
55              *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
56                                 Data_Basis,i,intpoint1,intpoint2,Jdet);
57              // second double integral
58              intpoint1=t_lm1+H_l*nodes_gauss[q1];
59              intpoint2 = t_lm1
60              +H_l*(nodes_gauss[q1]+nodes_gauss[q2]*(1-nodes_gauss[q1]));
61              Jdet=(1-nodes_gauss[q1]);
62              squareint+=weights_gauss[q1]*weights_gauss[q2]
63              *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
64                                 Data_Basis,i,intpoint1,intpoint2,Jdet);
65          }
```

```
66      }
67      return squareint;
68  }
69
70
71  double SquareIntegral_K_Adjacent(NURBSData* Data_Gamma,double* wcpoints1_gam,
72                                   double* wcpoints2_gam,NURBSData* Data_Basis,
73                                   QuadData* Data_Gauss,
74                                   int i,int l1,int l2,int singtype){
75
76      double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
77      double* weights_gauss=get_QuadData_weights(Data_Gauss);
78      int n_gauss=get_QuadData_n(Data_Gauss);
79      int q1,q2;
80      double t_l1m1=knotseq(Data_Basis,l1-1); // t_{l_1-1}
81      double t_l1=knotseq(Data_Basis,l1); // t_{l_1}
82      double H_l1=t_l1-t_l1m1; // H_{l_1}
83      double t_l2m1=knotseq(Data_Basis,l2-1); // t_{l_2-1}
84      double t_l2=knotseq(Data_Basis,l2); // t_{l_2}
85      double H_l2=t_l2-t_l2m1; // H_{l_1}
86      double squareint=0; // integral over square
87      double intpoint1, intpoint2; // first and second integration point
88      double Jdet; // Jacobi determinant of Duffy transformation
89
90      if (singtype==0){
91          for (q1=0;q1<n_gauss;q1=q1+1) {
92              for (q2=0;q2<n_gauss;q2=q2+1) {
93                  // first double integral
94                  intpoint1=t_l1m1+H_l1*nodes_gauss[q1];
95                  intpoint2 = t_l2m1+H_l2*(1-nodes_gauss[q1]*nodes_gauss[q2]);
96                  Jdet=nodes_gauss[q1];
97                  squareint+=weights_gauss[q1]*weights_gauss[q2]
98                  *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
99                                     Data_Basis,i,intpoint1,intpoint2,Jdet);
100                 // second double integral
101                 intpoint1=t_l1m1+H_l1*nodes_gauss[q1]*(1-nodes_gauss[q2]);
102                 intpoint2 = t_l2m1 + H_l2 * nodes_gauss[q2];
103                 Jdet=1-nodes_gauss[q2];
104                 squareint+=weights_gauss[q1]*weights_gauss[q2]
105                 *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
106                                    Data_Basis,i,intpoint1,intpoint2,Jdet);
107             }
108         }
109     } else {
110         for (q1=0;q1<n_gauss;q1=q1+1) {
111             for (q2=0;q2<n_gauss;q2=q2+1) {
112                 // first double integral
113                 intpoint1=t_l1m1+H_l1*(1-nodes_gauss[q1]*nodes_gauss[q2]);
114                 intpoint2 = t_l2m1 + H_l2 *nodes_gauss[q2];
115                 Jdet=nodes_gauss[q2];
116                 squareint+=weights_gauss[q1]*weights_gauss[q2]
117                 *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
118                                    Data_Basis,i,intpoint1,intpoint2,Jdet);
```

```
119                 // second double integral
120                 intpoint1=t_l1m1+H_l1*nodes_gauss[q1];
121                 intpoint2 = t_l2m1
122                 + H_l2 * nodes_gauss[q2] * (1 - nodes_gauss[q1]);
123                 Jdet=1-nodes_gauss[q1];
124                 squareint+=weights_gauss[q1]*weights_gauss[q2]
125                 *SquareIntegrand_K(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
126                                    Data_Basis,i,intpoint1,intpoint2,Jdet);
127             }
128         }
129     }
130     return squareint;
131 }
132
133
134 void build_Fvector(double* output,NURBSData* Data_Gamma,double* wcpoints1_gam,
135                    double* wcpoints2_gam,NURBSData* Data_Basis,
136                    QuadData* Data_Gauss,QuadData* Data_Gauss_small){
137
138     double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
139     double* weights_gauss=get_QuadData_weights(Data_Gauss);
140     int n_gauss=get_QuadData_n(Data_Gauss);
141     double* nodes_gauss_small=get_QuadData_nodes(Data_Gauss_small);
142     double* weights_gauss_small=get_QuadData_weights(Data_Gauss_small);
143     int n_gauss_small=get_QuadData_n(Data_Gauss_small);
144     double* knots_gam=get_NURBSData_knots(Data_Gamma);
145     int N_gam=get_NURBSData_N(Data_Gamma);
146     double* knots=get_NURBSData_knots(Data_Basis);
147     int N=get_NURBSData_N(Data_Basis);
148     int p=get_NURBSData_p(Data_Basis);
149     double tmp[2];
150     int i,k,l1,l2,q1,q2;
151     double b=knots[N-1];
152     int multb=0; // #b
153     while (nearly_equal(knotseq(Data_Basis,N-multb),b)) {multb=multb+1;}
154     double R_til[N-multb+1+p][p+1][n_gauss];
155     // R_til[i-1+p][l1-i][q1]=\tilde{R}_{i,l1}(nodes_gauss[q1])
156     double gamma1[N][n_gauss];
157     // gamma1[l1-1][q1] is first component of
158     // gamma(t_{l1-1}+H_l1*nodes_gauss[q1])
159     double gamma2[N][n_gauss];
160     // gamma2[l1-1][q1] is second component of
161     // gamma(t_{l1-1}+H_l1*nodes_gauss[q1])
162     double g_nu1[N][n_gauss];
163     // g_nu1[l2-1][q2] is first component of
164     // {{0,1},{-1,0}}*gamma'(t_{l2-1}+H_l2*nodes_gauss[q2])
165     // *\check{g}(t_{l2-1}+H_l2*nodes_gauss[q2])
166     double g_nu2[N][n_gauss];
167     // g_nu1[l2-1][q2] is second component of
168     // {{0,1},{-1,0}}*gamma'(t_{l2-1}+H_l2*nodes_gauss[q2])
169     // *\check{g}(t_{l2-1}+H_l2*nodes_gauss[q2])
170     double squareint; // integral over square
171     double Jdet; // Jacobi determinant of Duffy transformation
```

```
    double diff_gam[2];
    double t_l1m1,t_l1,H_l1,t_l2m1,t_l2,H_l2;
    // t_{l1-1},t_l1,H_l1,t_{l2-1},t_l2,H_l2
    double intpoint1, intpoint2; // first and second integration point
    int index; // help index

    // calculation of R_til
    // R_i
    for (i=1-p;i<=(N-multb+1);i=i+1) {
        // elements with nonemty intersection with support of R_i
        for (l1=max(i,1);l1<=min(i+p,N);l1=l1+1) {
            // quadrature points
            for (q1=0;q1<n_gauss;q1=q1+1) {
                t_l1m1=knotseq(Data_Basis,l1-1);
                t_l1=knotseq(Data_Basis,l1);
                H_l1=t_l1-t_l1m1;
                intpoint1=t_l1m1+H_l1*nodes_gauss[q1];
                eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,
                                     wcpoints2_gam,intpoint1);
                R_til[i-1+p][l1-i][q1] = eval_NURBS(Data_Basis,i,intpoint1)
                * norm(tmp);
            }
        }
    }


    // calculation of gamma1, gamma2, g_nu1, g_nu2
    for (l2=1;l2<=N;l2=l2+1){
        for (q2=0;q2<n_gauss;q2=q2+1){
            t_l2m1=knotseq(Data_Basis,l2-1);
            t_l2=knotseq(Data_Basis,l2);
            H_l2=t_l2-t_l2m1;
            intpoint2=t_l2m1+H_l2*nodes_gauss[q2];
            eval_NURBSCurve(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                            intpoint2);
            gamma1[l2-1][q2]=tmp[0];
            gamma2[l2-1][q2]=tmp[1];
            g_nu1[l2-1][q2] = g(tmp);
            g_nu2[l2-1][q2] = g(tmp);
            eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                 intpoint2);
            g_nu1[l2-1][q2] *= tmp[1];
            g_nu2[l2-1][q2] *= -tmp[0];
        }
    }


    // calculation of <Kg,\hat{R}_i>_{L_2(Gamma)}
    // R_i
    for (i=1-p;i<=(N-multb+1);i=i+1) {
        output[i+p-1]=0;
        // elements with nonemty intersection with support of R_i
        for (l1=max(i,1);l1<=min(i+p,N);l1=l1+1) {
```

```
225             // all elements
226             for (l2=1;l2<=N;l2=l2+1) {
227                 t_l1m1=knotseq(Data_Basis,l1-1);
228                 t_l1=knotseq(Data_Basis,l1);
229                 t_l2m1=knotseq(Data_Basis,l2-1);
230                 t_l2=knotseq(Data_Basis,l2);
231                 H_l1=t_l1-t_l1m1;
232                 H_l2=t_l2-t_l2m1;
233                 // quadrature
234                 if (0<min(H_l1,H_l2)){
235                     squareint=0;
236                     // elements with no intersection
237                     if ((!nearly_equal(t_l1m1,t_l2m1))
238                         && (!nearly_equal(t_l1m1,t_l2))
239                         && (!nearly_equal(t_l1,t_l2m1))
240                         && (!nearly_equal(t_l1,t_l2))
241                         && ((l1!=(N-multb+1)) || (l2!=1))
242                         && ((l2!=(N-multb+1)) || (l1!=1))) {
243                         for (q1=0;q1<n_gauss;q1=q1+1) {
244                             for (q2=0;q2<n_gauss;q2=q2+1) {
245                                 tmp[0] = gamma1[l1-1][q1]
246                                 - gamma1[l2-1][q2];
247                                 tmp[1] = gamma2[l1-1][q1]
248                                 - gamma2[l2-1][q2];
249                                 squareint +=
250                                 (weights_gauss[q1]*weights_gauss[q2]*
251                                 (tmp[0]*g_nu1[l2-1][q2]+tmp[1]*g_nu2[l2-1][q2])
252                                 /norm(tmp))/norm(tmp)*R_til[i-1+p][l1-i][q1]
253                                 /(2*M_PI);
254                             }
255                         }
256                     }
257                     // elements with intersection
258                     else {
259                         // identical elements
260                         if (l1==l2){
261                             squareint=SquareIntegral_K_Identical(
262                             Data_Gamma,wcpoints1_gam,wcpoints2_gam,
263                             Data_Basis,Data_Gauss_small,i,l1);
264                         }
265                         // elements with point intersection
266                         else{
267                             // singularity at s=0,t=1
268                             if (nearly_equal(t_l1m1,t_l2)
269                                 || ((l2==(N-multb+1)) && (l1==1))){
270                                 index=1;
271                                 while(!nearly_equal(t_l2,knots_gam[index-1])){
272                                     index=index+1;
273                                     if (index==(N_gam+1)){break;}
274                                 }
275                                 // t_l2 no knot of Gamma
276                                 if (index==(N_gam+1)){
277                                     squareint=SquareIntegral_K_Adjacent(
```

```
                                    Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                    Data_Basis,Data_Gauss_small,i,l1,l2,0);
                                }
                                // t_l2 knot of Gamma
                                else {
                                    squareint=SquareIntegral_K_Adjacent(
                                    Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                    Data_Basis,Data_Gauss,i,l1,l2,0);
                                }
                            }
                            // singularity at s=1,t=0
                            else{
                                index=1;
                                while(!nearly_equal(t_l1,knots_gam[index-1])){
                                    index=index+1;
                                    if (index==(N_gam+1)){break;}
                                }
                                // t_l1 no knot of Gamma
                                if (index==(N_gam+1)){
                                    squareint=SquareIntegral_K_Adjacent(
                                    Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                    Data_Basis,Data_Gauss_small,i,l1,l2,1);
                                }
                                // t_l1 knot of Gamma
                                else {
                                    squareint=SquareIntegral_K_Adjacent(
                                    Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                    Data_Basis,Data_Gauss,i,l1,l2,1);
                                }
                            }
                        }
                    }
                    output[i+p-1] += H_l1 * H_l2 * squareint;
                }
            }
        }
    }


    // calculation of <Kg+g/2,R_hat_i>_{L_2(Gamma)}
    // R_i
    for (i=1-p;i<=(N-multb+1);i=i+1) {
        // elements with nonemty intersection with support of R_i
        for (l1=max(i,1);l1<=min(i+p,N);l1=l1+1) {
            t_l1m1=knotseq(Data_Basis,l1-1); // t_{l1-1}
            t_l1=knotseq(Data_Basis,l1); // t_l1
            H_l1=t_l1-t_l1m1;
            // quadrature
            if (0<H_l1){
                for (q1=0;q1<n_gauss;q1=q1+1) {
                    eval_NURBSCurve(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                    t_l1m1+H_l1*nodes_gauss[q1]);
                    output[i+p-1] += H_l1  * weights_gauss[q1]
```

```
                        * g(tmp)/2 * R_til[i-1+p][l1-i][q1];
                }
            }
        }
    }
}
```

B.6. `Faer1Estimator.h` **and** `Faer1Estimator.c`**.** The computation of the Faermann estimator $\eta_h^{F_1}$ is based on Subsection 5.3, 5.4 and 5.5.

```
#ifndef _Faer1Estimator_h
#define _Faer1Estimator_h

#include <math.h>
#include <stdio.h>
#include "Spline.h"


/* parameters:
 Data_Gamma...NURBSData* for geometry Gamma, see Structures.h
 wcpoints1_gam...first component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 wcpoints2_gam...second component of weighted control points
 w_l^\gamma*C_l^\gamma for geometry corresponding to knots
 Data_Basis...NURBSData* (see Structures.h) for basis of approximation space
 weighted_c...vector with weighted coefficients
 w_{1-p}*c_{1-p},...,w_{N-#b+1}*c_{N-#b+1}
 of approximative solution corresponding to Data_Basis
 Data_Gauss...QuadData* for quadrature with weight function 1 on [0,1],
 see Structures.h
 Data_Gauss_V...QuadData* for quadrature with weight function 1 on [0,1],
 see Structures.h, used to evaluate V(\phi_h)
 Data_LogGauss_V...QuadData* for quadrature with weight function -log(x)
 on [0,1], see Structures.h, used to evaluate V(\phi_h)
 Data_Gauss_F...QuadData* for quadrature with weight function 1 om [0,1],
 see Structures.h, used to evaluate right-hand side F

 comments:
 we assume that:
 -)path gamma induced by Data_Gamma is and wcpoints_gam is positively
 orientated regular closed curve,
 which parametizes boundary Gamma of Lipschitz domain Omega with diam(Omega)<1
 -)#t_i^gamma<=p_gam+1
 -)number of different entries in knots of Data_Gamma >= 4
 -){t_i^gamma:i=1...N_gam}<={t_i:i=1...N}
 -)#t_i<=p+1
 */


double Integrand_V_smooth(NURBSData* Data_Gamma,double* wcpoints1_gam,
                          double* wcpoints2_gam,NURBSData* Data_Basis,
```

```
                            double* wcoeffs,double s,double t,
                            double denominator);
// returns -1/2pi * log(|\gamma(s)-\gamma(t)| / denominator)
// * \psi_h(\gamma(t))*|\gamma'(t)| for s!=t in [a,b) and denominator>0,
// where wcoeffs is a vector with weighted coefficients
// w_{1-p}*coeffs_{1-p},...,w_{N-#b+1}*coeffs_{N-#b+1} of
// function \psi_h in the approximation space


double Integrand_V_log(NURBSData* Data_Gamma,double* wcpoints1_gam,
                       double* wcpoints2_gam,NURBSData* Data_Basis,
                       double* wcoeffs,double t);
// returns 1/2pi \psi_h(\gamma(t))*|\gamma'(t)| for t in [a,b),
// where wcoeffs is a vector with weighted coefficients
// w_{1-p}*coeffs_{1-p},...,w_{N-#b+1}*coeffs_{N-#b+1} of function \psi_h in
// the approximation space


double eval_Vpsi(NURBSData* Data_Gamma,double* wcpoints1_gam,
                 double* wcpoints2_gam,NURBSData* Data_Basis,double* wcoeffs,
                 QuadData* Data_Gauss,QuadData* Data_LogGauss,double s);
// returns V(\psi_h)(\gamma(s)) for s in [a,b)-{check{x}_0,...,check{x}_n},
// where wcoeffs is a vector with weighted coefficients
// w_{1-p}*coeffs_{1-p},...,w_{N-#b+1}*coeffs_{N-#b+1} of function \psi_h in
// the approximation space


double Integrand_K(NURBSData* Data_Gamma,double* wcpoints1_gam,
                   double* wcpoints2_gam,double s,double t);
// returns \partial_\nu\check{G}(s,t)*\check{g}(t)*|\gamma'(t)| for
// s!=t in [a,b)


double eval_F(NURBSData* Data_Gamma,double* wcpoints1_gam,
              double* wcpoints2_gam,NURBSData* Data_Basis,
              QuadData* Data_Gauss,double s);
// returns F(\gamma(s)) for s in [a,b)-{check{x}_0,...,check{x}_n}


double SquareIntegrand_P(NURBSData* Data_Gamma,double* wcpoints1_gam,
                         double* wcpoints2_gam,NURBSData* Data_Basis,
                         double* weighted_c,QuadData* Data_Gauss_V,
                         QuadData* Data_LogGauss_V,QuadData* Data_Gauss_F,
                         double s,double t);
// returns \check{P}(s,t)*|gamma'(s)|*|gamma'(t)|


void build_Faer1Estimator(double* output,NURBSData* Data_Gamma,
                          double* wcpoints1_gam,double* wcpoints2_gam,
                          NURBSData* Data_Basis,double* weighted_c,
                          QuadData* Data_Gauss,QuadData* Data_Gauss_V,
                          QuadData* Data_LogGauss_V,QuadData* Data_Gauss_F);
// turns output[j] into j-th component of first element based Faermann
```

```
// estimator for j=1,...,n

#endif
```

```
#include "Faer1Estimator.h"


double Integrand_V_smooth(NURBSData* Data_Gamma,double* wcpoints1_gam,
                          double* wcpoints2_gam,NURBSData* Data_Basis,
                          double* wcoeffs,double s,double t,
                          double denominator){

    double s_hat[2]; // \gamma(s)
    double t_hat[2]; // \gamma(t)
    double tmp[2];
    double psi_til; // \tilde{\psi_h}(t)
    eval_NURBSCurve(s_hat,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
    eval_NURBSCurve(t_hat,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);

    eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
    // \gamma'(t)
    psi_til = eval_NURBSComb(Data_Basis,wcoeffs,t) * norm(tmp);
    tmp[0]=s_hat[0]-t_hat[0]; // \gamma(s)-\gamma(t)
    tmp[1]=s_hat[1]-t_hat[1];
    return -0.5/M_PI * log(norm(tmp)/denominator) * psi_til;
}


double Integrand_V_log(NURBSData* Data_Gamma,double* wcpoints1_gam,
                       double* wcpoints2_gam,NURBSData* Data_Basis,
                       double* wcoeffs,double t){

    double tmp[2];

    eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
    // \gamma'(t)
    return 0.5/M_PI * eval_NURBSComb(Data_Basis,wcoeffs,t) * norm(tmp);
}


double eval_Vpsi(NURBSData* Data_Gamma,double* wcpoints1_gam,
                 double* wcpoints2_gam,NURBSData* Data_Basis,double* wcoeffs,
                 QuadData* Data_Gauss,QuadData* Data_LogGauss,double s){

    double a=get_NURBSData_a(Data_Gamma);
    double* nodes=get_NURBSData_nodes(Data_Basis);
    int n=get_NURBSData_n(Data_Basis);
    double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
    double* weights_gauss=get_QuadData_weights(Data_Gauss);
    int n_gauss=get_QuadData_n(Data_Gauss);
    double* nodes_loggauss=get_QuadData_nodes(Data_LogGauss);
```

```
48      double* weights_loggauss=get_QuadData_weights(Data_LogGauss);
49      int n_loggauss=get_QuadData_n(Data_LogGauss);
50      int j,q;
51      double output=0;
52      double x_ch_jm1,x_ch_j,h_j; // \check{x}_{j-1},\check{x}_j,h_j
53      double intpoint;
54
55      for (j=1;j<=n;j=j+1){
56          if (j==1) {x_ch_jm1=a;} else {x_ch_jm1=nodes[j-2];}
57          x_ch_j=nodes[j-1];
58          h_j=x_ch_j-x_ch_jm1;
59          if ((x_ch_jm1>s) || (x_ch_j<s)){
60              for (q=0;q<n_gauss;q=q+1){
61                  intpoint = x_ch_jm1 + h_j * nodes_gauss[q];
62                  output += h_j * weights_gauss[q]
63                  * Integrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
64                                       Data_Basis,wcoeffs,s,intpoint,1);
65              }
66          }else{
67              // smooth integrals
68              for (q=0;q<n_gauss;q=q+1){
69                  // first integral
70                  intpoint = s - (s - x_ch_jm1) * nodes_gauss[q];
71                  output += (s - x_ch_jm1) * weights_gauss[q]
72                  * Integrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
73                                       Data_Basis,wcoeffs,s,intpoint,
74                                       nodes_gauss[q]);
75
76                  //second integral
77                  intpoint = s + (x_ch_j - s) * nodes_gauss[q];
78                  output += (x_ch_j - s) * weights_gauss[q]
79                  * Integrand_V_smooth(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
80                                       Data_Basis,wcoeffs,s,intpoint,
81                                       nodes_gauss[q]);
82              }
83              // log integrals
84              for (q=0;q<n_loggauss;q=q+1){
85                  // first integral
86                  intpoint = s - (s - x_ch_jm1) * nodes_loggauss[q];
87                  output += (s - x_ch_jm1) * weights_loggauss[q]
88                  * Integrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
89                                    Data_Basis,wcoeffs,intpoint);
90
91                  //second integral
92                  intpoint = s + (x_ch_j - s) * nodes_loggauss[q];
93                  output += (x_ch_j - s) * weights_loggauss[q]
94                  *Integrand_V_log(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
95                                   Data_Basis,wcoeffs,intpoint);
96              }
97          }
98      }
99      return output;
100 }
```

```
101
102
103 double Integrand_K(NURBSData* Data_Gamma,double* wcpoints1_gam,
104                     double* wcpoints2_gam,double s,double t){
105
106     double g_nu[2]; // \check{g}(t)|\gamma'(t)|\nu(\gamma(t))
107     double s_hat[2]; // \gamma(s)
108     double t_hat[2]; // \gamma(t)
109     double tmp[2];
110     eval_NURBSCurve(s_hat,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
111     eval_NURBSCurve(t_hat,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
112
113     eval_NURBSCurveDeriv(tmp,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
114     // \gamma'(t)
115     g_nu[0] = g(t_hat) * tmp[1];
116     g_nu[1] = -g(t_hat) * tmp[0];
117     tmp[0] = s_hat[0] - t_hat[0]; // \gamma(s)-\gamma(t)
118     tmp[1] = s_hat[1] - t_hat[1];
119     return 0.5 * (tmp[0]*g_nu[0] + tmp[1]*g_nu[1]) / (M_PI*norm(tmp))
120     /norm(tmp);
121 }
122
123
124 double eval_F(NURBSData* Data_Gamma,double* wcpoints1_gam,
125               double* wcpoints2_gam,NURBSData* Data_Basis,
126               QuadData* Data_Gauss,double s){
127
128     double a=get_NURBSData_a(Data_Gamma);
129     double* nodes=get_NURBSData_nodes(Data_Basis);
130     int n=get_NURBSData_n(Data_Basis);
131     double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
132     double* weights_gauss=get_QuadData_weights(Data_Gauss);
133     int n_gauss=get_QuadData_n(Data_Gauss);
134     int j,q;
135     double output=0;
136     double x_ch_jm1,x_ch_j,h_j;
137     // \check{x}_{j-1},\check{x}_j,h_j
138     double intpoint; // integration point
139     double s_hat[2]; // \gamma(s)
140     eval_NURBSCurve(s_hat,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
141
142
143     // calculation of Kg(\gamma(s))
144     for (j=1;j<=n;j=j+1){
145         if (j==1) {x_ch_jm1=a;} else {x_ch_jm1=nodes[j-2];}
146         x_ch_j=nodes[j-1];
147         h_j=x_ch_j-x_ch_jm1;
148
149         if ((x_ch_jm1>s) || (x_ch_j<s)){
150             for (q=0;q<n_gauss;q=q+1){
151                 intpoint = x_ch_jm1 + h_j * nodes_gauss[q];
152                 output += h_j * weights_gauss[q] * Integrand_K(
153                 Data_Gamma,wcpoints1_gam,wcpoints2_gam,s,intpoint);
```

```
154             }
155         }else{
156             for (q=0;q<n_gauss;q=q+1){
157                 // first integral
158                 intpoint = s - (s - x_ch_jm1) * nodes_gauss[q];
159                 output += (s - x_ch_jm1) * weights_gauss[q] * Integrand_K(
160                 Data_Gamma,wcpoints1_gam,wcpoints2_gam,s,intpoint);
161                 //second integral
162                 intpoint = s + (x_ch_j - s) * nodes_gauss[q];
163                 output += (x_ch_j - s) * weights_gauss[q] * Integrand_K(
164                 Data_Gamma,wcpoints1_gam,wcpoints2_gam,s,intpoint);
165             }
166         }
167     }
168     return output + g(s_hat)*0.5;
169 }
170
171
172 double SquareIntegrand_P(NURBSData* Data_Gamma,double* wcpoints1_gam,
173                          double* wcpoints2_gam,NURBSData* Data_Basis,
174                          double* weighted_c,QuadData* Data_Gauss_V,
175                          QuadData* Data_LogGauss_V,QuadData* Data_Gauss_F,
176                          double s,double t){
177
178     double tmp1[2];
179     double tmp2[2];
180     double diff_gam[2]; // gamma(s)-gamma(t)
181     double diff_Res; // \check{r}_h(s)-\check{r}_h(t)
182
183     eval_NURBSCurve(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
184     // gamma(s)
185     eval_NURBSCurve(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
186     // gamma(t)
187     diff_gam[0] = tmp1[0] - tmp2[0];
188     diff_gam[1] = tmp1[1] - tmp2[1];
189     diff_Res = eval_Vpsi(Data_Gamma,wcpoints1_gam,wcpoints2_gam,Data_Basis,
190                          weighted_c,Data_Gauss_V,Data_LogGauss_V,s)
191     -eval_F(Data_Gamma,wcpoints1_gam,wcpoints2_gam,Data_Basis,Data_Gauss_F,s)
192     -eval_Vpsi(Data_Gamma,wcpoints1_gam,wcpoints2_gam,Data_Basis,weighted_c,
193                 Data_Gauss_V,Data_LogGauss_V,t)
194     +eval_F(Data_Gamma,wcpoints1_gam,wcpoints2_gam,Data_Basis,Data_Gauss_F,t);
195     eval_NURBSCurveDeriv(tmp1,Data_Gamma,wcpoints1_gam,wcpoints2_gam,s);
196     // gamma'(s)
197     eval_NURBSCurveDeriv(tmp2,Data_Gamma,wcpoints1_gam,wcpoints2_gam,t);
198     // gamma'(t)
199     return pow(diff_Res/norm(diff_gam),2)* norm(tmp1)*norm(tmp2);
200 }
201
202
203 void build_Faer1Estimator(double* output,NURBSData* Data_Gamma,
204                           double* wcpoints1_gam,double* wcpoints2_gam,
205                           NURBSData* Data_Basis,double* weighted_c,
206                           QuadData* Data_Gauss,QuadData* Data_Gauss_V,
```

```
207                                 QuadData* Data_LogGauss_V,QuadData* Data_Gauss_F){
208
209     double a=get_NURBSData_a(Data_Gamma);
210     double* nodes_gam=get_NURBSData_nodes(Data_Gamma);
211     int n_gam=get_NURBSData_n(Data_Gamma);
212     double* nodes=get_NURBSData_nodes(Data_Basis);
213     int n=get_NURBSData_n(Data_Basis);
214     double* nodes_gauss=get_QuadData_nodes(Data_Gauss);
215     double* weights_gauss=get_QuadData_weights(Data_Gauss);
216     int n_gauss=get_QuadData_n(Data_Gauss);
217     int j,q1,q2;
218     double x_ch_jm1,x_ch_j,x_ch_jp1;
219     // check{x}_{j-1}, check{x}_{j}, check{x}_{j+1} corresponding to nodes
220     // exceptionally we set check{x}_{n+1}=check{x}_1
221     double h_j,h_jp1; // h_j, h_{j+1}
222     double squareint1;
223     // 0.5 * \int\check{x}_{j-1}..\check{x}_j,\check{x}_{j-1}..\check{x}_j
224     double squareint2;
225     // \check{x}_{j-1}..\check{x}_j,\check{x}_j..\check{x}_{j+1}
226     double squareint3;
227     // 0.5 * \int\check{x}_j..\check{x}_{j+1},\check{x}_j..\check{x}_{j+1}
228     double intpoint1,intpoint2,Jdet;
229     double r_patch[n]; // |r_h|_{H^{1/2}(omega_h(x_j))}^2
230     int index; // help index
231
232     // calculation of |r_h|_{H^{1/2}(omega_h(x_j))}^2
233     // elements
234     for (j=1;j<=n;j=j+1) {
235         // nodes
236         if (j==1) {x_ch_jm1=a;} else {x_ch_jm1=nodes[j-2];}
237         x_ch_j=nodes[j-1];
238         if (j==n){x_ch_jp1=nodes[0];}else{x_ch_jp1=nodes[j];};
239         h_j=x_ch_j-x_ch_jm1;
240         if (j==n){h_jp1=nodes[0]-a;}else{h_jp1=x_ch_jp1-x_ch_j;}
241
242
243         // definition of squareints
244         if (j==1){
245             squareint1=0;
246         }else{
247             squareint1=squareint3;
248         }
249         squareint2=0; squareint3=0;
250
251         // quadrature
252         for (q1=0;q1<n_gauss;q1=q1+1) {
253             for (q2=0;q2<n_gauss;q2=q2+1) {
254                 if (j==1){
255                     // squareint1
256                     intpoint1=x_ch_jm1+h_j*nodes_gauss[q1];
257                     intpoint2 = x_ch_jm1
258                     + h_j * nodes_gauss[q1] * (1 - nodes_gauss[q2]);
259                     Jdet=nodes_gauss[q1];
```

```
                squareint1+=
                h_j*h_j*weights_gauss[q1]*weights_gauss[q2]*Jdet
                *SquareIntegrand_P(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                                   Data_Basis,weighted_c,Data_Gauss_V,
                                   Data_LogGauss_V,Data_Gauss_F,
                                   intpoint1,intpoint2);
            }
            // squareint2: first double integral
            intpoint1=(x_ch_jp1-h_jp1)+h_jp1*nodes_gauss[q1];
            intpoint2 = x_ch_jm1
            +h_j*(1-nodes_gauss[q1]*nodes_gauss[q2]);
            Jdet=nodes_gauss[q1];
            squareint2+=
            h_j*h_jp1*weights_gauss[q1]*weights_gauss[q2]*Jdet
            *SquareIntegrand_P(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                               Data_Basis,weighted_c,Data_Gauss_V,
                               Data_LogGauss_V,Data_Gauss_F,
                               intpoint1,intpoint2);
            // squareint2: second double integral
            intpoint1=+(x_ch_jp1-h_jp1)
            +h_jp1*nodes_gauss[q1]*nodes_gauss[q2];
            intpoint2 = x_ch_jm1+h_j*(1-nodes_gauss[q2]);
            Jdet=nodes_gauss[q2];
            squareint2+=
            h_j*h_jp1*weights_gauss[q1]*weights_gauss[q2]*Jdet
            *SquareIntegrand_P(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                               Data_Basis,weighted_c,Data_Gauss_V,
                               Data_LogGauss_V,Data_Gauss_F,
                               intpoint1,intpoint2);

            // squareint3
            intpoint1=(x_ch_jp1-h_jp1)+h_jp1*nodes_gauss[q1];
            intpoint2 = (x_ch_jp1-h_jp1)
            +h_jp1 * nodes_gauss[q1] * (1 - nodes_gauss[q2]);
            Jdet=nodes_gauss[q1];
            squareint3+=
            h_jp1*h_jp1*weights_gauss[q1]*weights_gauss[q2]*Jdet
            *SquareIntegrand_P(Data_Gamma,wcpoints1_gam,wcpoints2_gam,
                               Data_Basis,weighted_c,Data_Gauss_V,
                               Data_LogGauss_V,Data_Gauss_F,
                               intpoint1,intpoint2);
        }
    }
    r_patch[j-1] = 2*(squareint1+squareint2+squareint3);
}

// element based estimator
for (j=1;j<=n;j=j+1){
    if (j==1){
        output[j-1]=sqrt(r_patch[n-1]+r_patch[j-1]);
    }else{
        output[j-1]=sqrt(r_patch[j-2]+r_patch[j-1]);
    }
```

```
313     }
314 }
```

B.7. `MEX` **files.** All the codes of the previous subsections can be used in MATLAB with the help of MEX. Here one example.

```
1  #include <mex.h>
2  #include "Spline.c"
3  #include "Vmatrix.c"
4
5
6  void mexFunction(int nlhs, mxArray* plhs[], int nrhs, const mxArray* prhs[]) {
7      double a=*mxGetPr(prhs[0]);
8      double* knots_gam=mxGetPr(prhs[1]);
9      double* weights_gam=mxGetPr(prhs[2]);
10     int p_gam=(int)*mxGetPr(prhs[3]);
11     double* nodes_gam=mxGetPr(prhs[4]);
12     int is_rational_gam=(int)*mxGetPr(prhs[5]);
13     double* cpoints_gam=mxGetPr(prhs[6]);
14     double* knots=mxGetPr(prhs[7]);
15     double* weights=mxGetPr(prhs[8]);
16     int p=(int)*mxGetPr(prhs[9]);
17     double* nodes=mxGetPr(prhs[10]);
18     int is_rational=(int)*mxGetPr(prhs[11]);
19     double* nodes_gauss=mxGetPr(prhs[12]);
20     double* weights_gauss=mxGetPr(prhs[13]);
21     double* nodes_loggauss=mxGetPr(prhs[14]);
22     double* weights_loggauss=mxGetPr(prhs[15]);
23     double* output;
24     int l;
25
26     int N_gam=mxGetM(prhs[1]);
27     int n_gam=mxGetM(prhs[4]);
28     int N=mxGetM(prhs[7]);
29     int n=mxGetM(prhs[10]);
30     int n_gauss=mxGetM(prhs[12]);
31     int n_loggauss=mxGetM(prhs[14]);
32     double b;
33     int multb=0; // #b
34     NURBSData* Data_Basis;
35     NURBSData* Data_Gamma;
36     QuadData* Data_Gauss;
37     QuadData* Data_LogGauss;
38     double wcpoints1_gam[N_gam];
39     double wcpoints2_gam[N_gam];
40
41     b=knots[N-1];
42     while (nearly_equal(knots[N-multb-1],b)) {
43         multb=multb+1;
44         if (multb==N){break;}
45     }
```

```
    plhs[0]=mxCreateDoubleMatrix(N_multb+1+p,N_multb+1+p,mxREAL);
    output=mxGetPr(plhs[0]);

    if (is_rational_gam==0){
        for (l=0;l<N_gam;l=l+1){
            wcpoints1_gam[l]=cpoints_gam[0+2*l];
            wcpoints2_gam[l]=cpoints_gam[1+2*l];
        }
    }else{
        for (l=0;l<N_gam;l=l+1){
            wcpoints1_gam[l]=weights_gam[l]*cpoints_gam[0+2*l];
            wcpoints2_gam[l]=weights_gam[l]*cpoints_gam[1+2*l];
        }
    }

    Data_Basis=new_NURBSData(a,knots,N,weights,p,nodes,n,is_rational);
    Data_Gamma=new_NURBSData(a,knots_gam,N_gam,weights_gam,p_gam,nodes_gam,
                             n_gam,is_rational_gam);
    Data_Gauss=new_QuadData(nodes_gauss,weights_gauss,n_gauss);
    Data_LogGauss=new_QuadData(nodes_loggauss,weights_loggauss,n_loggauss);

    build_Vmatrix(output,Data_Gamma,wcpoints1_gam,wcpoints2_gam,Data_Basis,
                  Data_Gauss,Data_LogGauss);

    Data_Basis=del_NURBSData(Data_Basis);
    Data_Gamma=del_NURBSData(Data_Gamma);
    Data_Gauss=del_QuadData(Data_Gauss);
    Data_LogGauss=del_QuadData(Data_LogGauss);
}
```

B.8. `markElements.m`. This file implements Dörfler marking.

```
function marked = mark_Elements(indicators,theta)
% input:
%   indicators...column vector of error indicators
%   theta...constant for Doerfler marking
% output:
%   marked...column vector of indices of marked elements


if theta<1
    [indicators_tmp,tmp] = sort(indicators,'descend');
    sum_indicatorssq = cumsum(indicators_tmp.^2);
    index = find(sum_indicatorssq >= (sum_indicatorssq(end)*theta),1);
    marked = sort(tmp(1:index));
else
    marked=(1:length(indicators))';
end
```

B.8.1. `refine_BoundaryMesh.m`. Here, refinement of the marked elements is implemented.

```
function [knots_fine,weights_fine]=refine_BoundaryMesh(a,knots,weights,...
    p,marked,kappa_max)
% input:
%   a...left interval boundary of [a,b]
%   knots...column vector of knots in (a,b],
%       where kappa(\check{\mathcal{T}})<=kappa_max
%   weights...column vector of positive weights corresponding to knots
%   marked...column vector with indices of marked elements
%   kappa_max...maximal allowed mesh constant
% output:
%   knots_fine...column vector of refined knots (via knot insertion),
%       where kappa(\check{\mathcal{T}}_fine)<=kappa_max, at least all
%       marked elements are
%       refined, no element is refined more than one time
%   weights_fine...column vector of new weights corresponding to knots_fine
% comments:
% we assume that
%   -) number of knots N<=p+1
%   -) number of nodes n>=4

b=knots(end);

% start data
weights_current=weights;
knots_current=knots;
N_current=length(knots_current);
nodes_current=unique(knots_current);
n_current=length(nodes_current);
nodes0_current=[a;nodes_current];
marked_current=marked; % marked elements of current mesh

% loop, in each step one refinement
while ~isempty(marked_current)
    refine=marked_current(1); % element to be refined
    l=find(knots_current==nodes_current(refine),1);
    % marked element = [t_{l-1},t_l]
    t_lm1=knotseq(a,knots_current,l-1);
    t_l=knotseq(a,knots_current,l);

    % length of first marked element and its neighbours
    if (refine ~= 1)
        h_ch_left=nodes0_current(refine)-nodes0_current(refine-1);
    else
        h_ch_left=b-nodes0_current(n_current);
    end
    h_ch=nodes0_current(refine+1)-nodes0_current(refine);
    if (refine ~= n_current)
        h_ch_right=nodes0_current(refine+2)-nodes0_current(refine+1);
    else
        h_ch_right=nodes0_current(2)-a;
    end

```

```
53      % mark neighbours of first marked element if necessary
54      % to guarantee kappa<=kappa_max (+1e-15)
55      if max((h_ch/2)/h_ch_left,h_ch_left/(h_ch/2))>(kappa_max+1e-15)
56          marked_current=unique([refine-1+n_current*(refine==1);...
57              marked_current]);
58      end
59      if max((h_ch/2)/h_ch_right,h_ch_right/(h_ch/2))>(kappa_max+1e-15)
60          marked_current=unique([marked_current;...
61              refine+1-n_current*(refine==n_current)]);
62      end
63
64      % refine element
65      t_pr=(t_lm1+t_l)/2; % t'
66      weights_tmp=zeros(N_current+1,1); % w'''
67      weights_tmp(1:(l-p))=weights_current(1:(l-p));
68      for i=max(1,(l+1-p)):l
69          t_im1=knotseq(a,knots_current,i-1);
70          t_im1pp=knotseq(a,knots_current,i-1+p);
71          if t_im1<t_im1pp
72              beta=(t_pr-t_im1)/(t_im1pp-t_im1);
73          else
74              beta=0;
75          end
76          weights_tmp(i)=(1-beta)*weights_current(i-1+N_current*(i==1))...
77              +beta*weights_current(i);
78      end
79      weights_tmp((l+1):(N_current+1))=weights_current(l:N_current);
80      weights_current(1:min(N_current+1,l+N_current+1-p))=...
81          weights_tmp(1:min(N_current+1,l+N_current+1-p));
82      for i=(l+N_current+2-p):(N_current+1)
83          t_im2=knotseq(a,knots_current,i-2);
84          t_im2pp=knotseq(a,knots_current,i-2+p);
85          if t_im2<t_im2pp
86              beta=(t_pr+b-a-t_im2)/(t_im2pp-t_im2);
87          else
88              beta=0;
89          end
90          weights_current(i)=(1-beta)*weights_tmp(i-1)...
91              +beta*weights_tmp(i);
92      end
93
94      % update data
95      knots_current=[knots_current(1:(l-1));t_pr;knots_current(l:end)];
96      N_current=N_current+1;
97      nodes_current=unique(knots_current);
98      n_current=length(nodes_current);
99      nodes0_current=[a;nodes_current];
100     marked_current=[marked_current(marked_current<refine);...
101         1+marked_current(marked_current>refine)];
102 end
103 knots_fine=knots_current;
104 weights_fine=weights_current;
105 end
```

```
106
107 function output=knotseq(a,knots,i)
108 % returns t_i
109 b=knots(end);
110 N=length(knots);
111 if (mod(i,N) ~= 0)
112     output=knots(mod(i,N))+(b-a)*floor(i/N);
113 else
114     output=a+(b-a)*floor(i/N);
115 end
116 end
```

References

[BS08] Susanne C. Brenner and L.Ridgway Scott. *The mathematical theory of finite element methods. 3rd ed.* New York, NY: Springer, 3rd ed. edition, 2008.

[CF01] C. Carstensen and B. Faermann. Mathematical foundation of a posteriori error estimates and adaptive mesh-refining algorithms for boundary integral equations of the first kind. *Eng. Anal. Bound. Elem.*, 25(7):497–509, 2001.

[CHB09] J Austin Cottrell, Thomas JR Hughes, and Yuri Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[dB86] Carl de Boor. *B (asic)-spline basics*. Mathematics Research Center, University of Wisconsin-Madison, 1986.

[Fae00] Birgit Faermann. Localization of the Aronszajn-Slobodeckij norm and application to adaptive boundary element methods. I: The two-dimensional case. *IMA J. Numer. Anal.*, 20(2):203–234, 2000.

[Geo67] G. Georganopoulos. Sur l'approximation des fonctions continues par des fonctions lipschitziennes. *C. R. Acad. Sci., Paris, Sér. A*, 264:319–321, 1967.

[Kal11] Michael Kaltenbäck. *Analysis 3*. Lecture Notes of TU Vienna, August 2011.

[Kal13] Michael Kaltenbäck. *Analysis 3*. Lecture Notes of TU Vienna, September 2013.

[McL00] William Charles Hector McLean. *Strongly elliptic systems and boundary integral equations*. Cambridge university press, 2000.

[NPV11] Eleonora Di Nezza, Giampiero Palatucci, and Enrico Valdinoci. Hitchhiker's guide to the fractional sobolev spaces. 04 2011.

[Pra06] Dirk Praetorius. *Numerische Mathematik*. Lecture Notes of TU Vienna, 2006.

[Rud91] Walter Rudin. *Functional analysis*. International Series in Pure and Applied Mathematics. McGraw-Hill, Inc., New York, second edition, 1991.

[SS11] Stefan A. Sauter and Christoph Schwab. *Boundary element methods*. Berlin: Springer, 2011.

[Ste08] Olaf Steinbach. Numerical approximation methods for elliptic boundary value problems. *Finite and Boundary Elements*, 2008.