Reparameterization and Adaptive Quadrature for the Isogeometric Discontinuous Galerkin Method

Agnes Seiler[1], Bert Jüttler[2]

[1] Doctoral Program "Computational Mathematics"
[2] Institute of Applied Geometry
Johannes Kepler University Linz, Altenberger Str. 69, 4040 Linz, Austria
agnes.seiler@dk-compmath.jku.at, bert.juettler@jku.at

Abstract. We use the Poisson problem with Dirichlet boundary conditions to illustrate the complications that arise from using non-matching interface parameterizations within the framework of Isogeometric Analysis on a multi-patch domain, using discontinuous Galerkin (dG) techniques to couple terms across the interfaces. The dG-based discretization of a partial differential equation is based on a modified variational form, where one introduces additional terms that measure the discontinuity of the values and normal derivatives across the interfaces between patches. Without matching interface parameterizations, firstly, one needs to identify pairs of associated points on the common interface of the two patches for correctly evaluating the additional terms. We will use reparameterizations to perform this task. Secondly, suitable techniques for numerical integration are needed to evaluate the quantities that occur in the discretization with the required level of accuracy. We explore two possible approaches, which are based on subdivision and adaptive refinement, respectively.

1 Introduction

Isogeometric Analysis (IgA) [5,6] uses the same spaces of spline functions for representing the geometry of a physical domain and for performing a discretization in the context of a PDE-based numerical simulation. This method is based on a parameterization of the physical domain, i.e., on a geometry map that relates the physical domain and the parameter domain.

Many approaches rely on tensor product parameterizations, where the domain is a unit square or a unit cube. Consequently, more complex domains have to be divided into several single patches, forming a multi-patch representation. There exist several methods for coupling the discrete discontinuous Galerkin IgA patch wise solution across the interfaces between single patches and enhancing global continuity of the solution. These include Nitsche's [16] method and mortar techniques [2] as well as the discontinuous Galerkin (dG) approach, which is the focus of the present paper.

DG methods discretize the variational form of a partial differential equation taking into account the discontinuity of the finite dimensional discretization spaces. The publications [4,17] provide a general description of dG techniques in the context of finite elements, which have been transferred to the isogeometric setting in [3,12,13,14].

So far, only matching interface parameterizations have been studied in the context of dG-IgA methods. More precisely, whenever two patches meet in an interface, then the parameterizations restricted to these interfaces are assumed to be identical (possibly after affine transformations of the parameter domains), see [12,13,14,18]. On the one hand, this limitation provides the advantage that the elements of the patches on both sides of the interface are perfectly matching, which significantly simplifies the implementation of such methods. On the other hand, it complicates substantially the creation of multi-patch parameterizations.

As notable exceptions we mention the recent publications [10,11], where the authors study gaps and overlaps at the interfaces. While the theory presented in these papers does not require any assumptions regarding matching interfaces, such conditions are assumed to be satisfied in all the computational examples. More precisely, the meshes of the considered domains fulfill restrictive correspondence conditions, which are quite similar to the matching case. This is due to the lack of an implementation for the non-matching case [9].

This recent work has motivated us to investigate the effect of non-matching interface parameterizations in the context of dG-IgA in the present paper. We aim to give a complete description of the necessary computational steps for applying the theoretical results of [10,11,12,13,14,18] to the case of non-matching parameterizations at the interfaces. In order to keep the presentation simple, we restrict ourselves to planar two-patch domains and we assume that the interfaces are geometrically matching, thus they have neither overlaps nor gaps. However, it is clear that the results from [10,11] apply to the non-matching case also, as the theory presented there is sufficiently general.

More precisely, the assembly of the local stiffness matrices derived from the dG bilinear form requires the computation of integrals of the type

$$\int_e D\, b_i^k(\boldsymbol{x}) D'\, b_j^\ell(\boldsymbol{x}) d\boldsymbol{x} \ , \tag{1}$$

where e is an interface between Ω^k and Ω^ℓ in the physical domain, $\boldsymbol{x} \in e$ is a point on the interface, b_i^k , b_j^ℓ are isogeometric basis functions defined on patches Ω^k , $\Omega^\ell \subset \Omega$ of the multi-patch domain $\Omega \subseteq \mathbb{R}$, and D, D' are differential operators. As we shall see, non-matching interface parameterizations give rise to two problems that need to be treated separately.

The first one concerns the evaluation of $b_i^k(\boldsymbol{x})$ and $b_j^\ell(\boldsymbol{x})$ at the same position $\boldsymbol{x}$ on the interface. Due to the use of non-matching parameterizations, a point $\boldsymbol{x}$ will have two possibly different preimages in the parameter domains of the two patches joined by the interface respectively. To identify pairs of corresponding preimages we use reparameterizations of the preimages of the interface. We also investigate the influence of the quality of the reparameterization on the accuracy of the overall result.

The second problem is related to the use of numerical integration methods. We need to find a quadrature method whose exactness does not depend on the smoothness of the integrands. We present different approaches, one resulting from dividing the element on which quadrature is performed and another one making use of automatized element splitting. The performance of both approaches is explored in numerical experiments.

The remainder of this paper is structured as follows: We establish the notation and describe the example problem we will focus on in the next section. We then state the two issues of evaluation and numerical integration, as described above. Section 3 treats the first problem of finding suitable reparameterizations, while Section 4 is devoted to the different quadrature techniques. Results of numerical experiments are presented in Section 5. Finally we conclude the paper.

2 Preliminaries

We recall the discontinuous Galerkin isogeometric (dG-IgA) discretization of a given model problem and discuss the computation of the stiffness matrix elements in the case of non-matching interface parameterizations. Hereby, we restrict ourselves to the two-patch case shown in Figure 1 due to better readability. All observations generalize directly to domains with more than two patches.

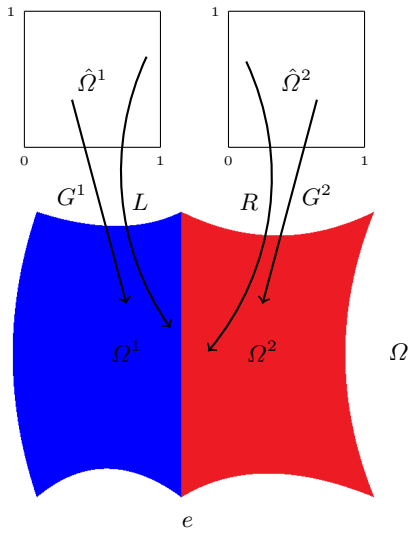

Fig. 1. Multi-patch domain with two patches Ω^1, Ω^2, one interface e and geometry mappings G^1, G^2. The mappings L and R will be introduced later.

2.1 The Model Problem and the Multi-patch Discretization

Given a domain $\Omega \subseteq \mathbb{R}^2$, we consider the Poisson problem

$$\text{Find } u: \quad \begin{cases} -\nabla \cdot (\alpha \nabla u) & = f \quad \text{on } \Omega \\ u & = 0 \quad \text{on } \partial\Omega \ , \end{cases} \tag{2}$$

where f is given and $\alpha > 0$ is the known diffusion coefficient. We allow α to be piecewise constant, i.e. α may take different values on every single patch (see below).

More precisely, we consider a multi-patch domain $\Omega \subseteq \mathbb{R}^2$ that consists of 2 non-overlapping single patches Ω^1, Ω^2 such that $\bar{\Omega}^1 \cup \bar{\Omega}^2 = \bar{\Omega}$. We use upper indices to refer to the number of the patch, and thus α^k denotes the value of the diffusion coefficient on the k-th patch, $k = 1, 2$.

An interface e between the two patches is the intersection $e = \bar{\Omega}^1 \cap \bar{\Omega}^2$. We consider interfaces that are curve segments only and ignore the remaining ones.

Each physical patch Ω^k is parameterized by an associated geometry mapping G^k with parameter domain $\hat{\Omega}^k = [0,1]^2$, $k = 1, 2$. These mappings are tensor product spline functions

$$G^k(\boldsymbol{\xi}) = \sum_{i \in \mathcal{R}^k} P_i^k \beta_i^k(\boldsymbol{\xi}) \, , \, \boldsymbol{\xi} \in \hat{\Omega}^k, \tag{3}$$

which are defined by control points $P_i^k \in \mathbb{R}^2$ and tensor product B-splines β_i^k, where $\mathcal{R}^k$ is the index set of the $k-$th patch. The lower index i identifies the i-th degree of freedom of the k-th patch.

We do not assume that the knot vectors of the patches are identical. These knot vectors split each parameter domain into elements. We will use open knot vectors which implies that the boundaries of the patches are B-spline curves.

An isogeometric basis function b_i^k on the physical patch Ω^k is the push-forward of a B-spline β^k defined on the parameter domain $\hat{\Omega}^k$,

$$b_i^k(\boldsymbol{x}) = \begin{cases} \left(\beta_i^k \circ \left(G^k\right)^{-1}\right)(\boldsymbol{x}) & \text{if } \boldsymbol{x} \in \Omega^k \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

These functions span the space which is used to derive the dG-IgA discretization.

For later reference we define the set of all edges

$$\mathit{\Gamma} = \bigcup_{k=1}^{2} \{G^k([0,1],0),\, G^k([0,1],1),\, G^k(0,[0,1]),\, G^k(1,[0,1])\} \tag{5}$$

of the multi-patch domain. It is the disjoint union of the set of the interface edges

$$\mathit{\Gamma}_C = \{e \in \mathit{\Gamma} : e \subseteq \bar{\Omega}^1 \cap \bar{\Omega}^2\} \tag{6}$$

and the set of boundary edges

$$\mathit{\Gamma}_D = \{e \in \mathit{\Gamma} : e \subseteq \bar{\Omega}^k \cap \partial\Omega \, , \, k = 1, 2\} \ . \tag{7}$$

2.2 DG-IgA Discretization

The discontinuous Galerkin isogeometric (dG-IgA) discretization space considers the subspace

$$\mathcal{V}_h = \operatorname{span}\{b_i^k \,:\, i \in \mathcal{R}^k,\, k = 1, \ldots, n\} \subseteq \prod_{k=1}^{2} \mathcal{H}^1\left(\Omega^k\right) \quad , \tag{8}$$

of the broken Sobolev space, see [17]. Functions in $\mathcal{V}_h$ are continuously differentiable on the interior of the single patches but not necessarily smooth across interface edges. This smoothness of the solution is achieved approximately by introducing a penalty term that considers the jump of the solution across the interface. Before stating the final variational formulation we need to define averages and jumps, see [4,17].

For each patch index k, any function $v \in \prod_{k=1}^{2} \mathcal{H}^1\left(\Omega^k\right)$ has a well-defined trace along any edge $e \subset \partial\Omega^k$. Thus, any such function v defines *two* traces on the interface $e \in \Gamma_C$, which we denote as $v|_{\Omega^{1_e}}$ and $v|_{\Omega^{2_e}}$, respectively. We use them to define the *average*

$$\{v\}^e = \frac{1}{2}\left(v|_{\Omega^{1_e}} + v|_{\Omega^{2_e}}\right) \tag{9}$$

and the *jump*

$$[v]^e = v|_{\Omega^{1_e}} - v|_{\Omega^{2_e}} \tag{10}$$

across the interface $e \in \Gamma_C$.

These definitions are further extended to boundary edges $e \in \Gamma_D$,

$$\{v\}^e = v|_{\Omega^k} \text{ and } [v]^e = v|_{\Omega^k},\, k = 1, 2 \ . \tag{11}$$

The dG-IgA discretization

$$\text{Find } u \in \mathcal{V}_h : \quad a(u, v) = F(v) \quad \forall v \in \mathcal{V}_h \tag{12}$$

of the Poisson problem (2) uses the bilinear form

$$a(u, v) = \sum_{k=1}^{2} a_1^k(u, v) - \frac{1}{2} \sum_{e \in \Gamma_C \cup \Gamma_D} \left(a_{2,1}^e(u, v) + a_{2,2}^e(u, v)\right) + \sum_{e \in \Gamma_C \cup \Gamma_D} a_3^e(u, v) \tag{13}$$

with

$$a_1^k(u, v) = \int_{\Omega^k} \alpha^k \nabla u \cdot \nabla v \mathrm{d}\Omega \ , \tag{14}$$

$$a_{2,1}^e(u, v) = \int_e \{\nabla u \cdot n\}^e [v]^e de \ , \ \ a_{2,2}^e(u, v) = \int_e \{\nabla v \cdot n\}^e [u]^e \mathrm{d}e \ , \tag{15}$$

$$a_3^e(u, v) = \int_e \frac{\delta}{h} [u]^e [v]^e \mathrm{d}e \tag{16}$$

and the linear form

$$F(v) = \int_\Omega f v d\Omega \ . \tag{17}$$

The second group of terms $a^e_{2,1}$ and $a^e_{2,2}$ considers normal vectors $n = n_e$ of the interface e, which need to comply with the chosen orientation of the edges (determined by the patch numbering). The last terms a^e_3 in the bilinear form are the penalty terms mentioned before, which involve the sufficiently large parameter δ. They depend on the element size h, i.e. on the length of the knot spans[3].

A detailed derivation of the dG discretization is given in [17]. The adaptation to the isogeometric setting is discussed in the thesis [3], which also comments on the choice of the δ, and in the recent article [12].

The discretization (12) defines the associated *dG norm*

$$\|u\|^2_{dG} = \sum_{k=1}^{2} a^k_1(u,u) + \sum_{e \in \Gamma_C \cup \Gamma_D} a^e_3(u,u) \ , \tag{18}$$

where in $a_1(u,u)$ the gradient of u is restricted to Ω^k, see again [12].

The coefficients u^k_i of the approximate solution

$$u_h = \sum_{k=1}^{2} \sum_{i \in \mathcal{R}_k} u^k_i b^k_i \tag{19}$$

are found by solving the linear system $Su = b$ with

$$\begin{aligned} S &= \left(s_{(i,k),(j,\ell)}\right)_{(i,k),(j,\ell)} \ , \\ b &= \left(b_{(j,\ell)}\right)_{(j,\ell)} \ , \\ u &= \left(u^k_i\right)_{(i,k)} \ , \end{aligned}$$

where

$$\begin{aligned} s_{(i,k),(j,\ell)} &= a\left(b^k_i, b^\ell_j\right) \ , \ i \in \mathcal{R}^k \ , \ j \in \mathcal{R}^\ell \ , \ k,\ell = 1,2, \text{ and} \\ b_{(j,\ell)} &= F\left(b^\ell_j\right) \ , \ j \in \mathcal{R}^\ell \ , \ \ell = 1,2 \ . \end{aligned}$$

2.3 Integrals along Interfaces

Evaluating the forms in (13) involves integrals along interfaces, which pose considerable difficulties. We discuss the evaluation of these quantities in more detail, considering again the domain shown in Figure 1. As a representative example we shall focus on $a^e_{2,1}$. All observations generalize directly to other terms.

[3] For simplicity we consider uniform knots only. If this is not the case then one may consider quasi-uniform knots instead, choosing a parameter that controls the size of all knot spans.

In this situation we obtain

$$a^e_{2,1}(u,v) = \int_e \left(\nabla u|_{\Omega^1} \cdot n\right) v|_{\Omega^1} + \left(\nabla u|_{\Omega^2} \cdot n\right) v|_{\Omega^1} - \left(\nabla u|_{\Omega^1} \cdot n\right) v|_{\Omega^2} - \left(\nabla u|_{\Omega^2} \cdot n\right) v|_{\Omega^2} \mathrm{d}e \ .$$

The stiffness matrix is a combination of several matrices, each of which is contributed by one of the four forms in (13) defining it. In particular we focus on the contribution of $a^e_{2,1}$.

Taking into account that

$$b^2_i|_{\Omega^1} = 0 \ , \ \nabla b^2_i|_{\Omega^1} = 0 \quad \forall i \in \mathcal{R}^2 \ ,$$
$$b^1_i|_{\Omega^2} = 0 \ , \ \nabla b^1_i|_{\Omega^2} = 0 \quad \forall i \in \mathcal{R}^1 \ ,$$

we find that only the expressions

$$a^e_{2,1}\left(b^k_i, b^\ell_j\right) = (-1)^{\ell+1} \int_e \left(\nabla b^k_i|_{\Omega^k} \cdot n\right) b^\ell_j|_{\Omega^\ell} \mathrm{d}e \tag{20}$$

contribute to the element $s_{(i,k),(j,\ell)}$ of the stiffness matrix.

In order to compute these values we use an appropriate numerical quadrature rule, which means that we have to evaluate these products on the interface e. This is no major problem if $k = \ell$ since the integral involves only one trace in this case. However, the situation is more complicated if $k \neq \ell$ since the (possibly different) parameterizations of the interface need to be taken into account. In the remainder of this section we discuss the evaluation of $a^e_{2,1}\left(b^1_i, b^2_j\right)$ in more detail.

The interface

$$e = G^1([0,1]^2) \cap G^2([0,1]^2) = G^1(1,[0,1]) = G^2(0,[0,1]) \tag{21}$$

is parameterized by the restrictions

$$L = G^1|_{(G^1)^{-1}(e)} \text{ and } R = G^2|_{(G^2)^{-1}(e)} \ , \tag{22}$$

see Figure 1. These two different representations of the same interface are related by the reparameterizations

$$\lambda : [0,1] \to \{1\} \times [0,1] \tag{23}$$

and

$$\varrho : [0,1] \to \{0\} \times [0,1] \tag{24}$$

via

$$L \circ \lambda = R \circ \varrho. \tag{25}$$

The *construction of suitable reparameterizations* λ and ϱ is the first major problem related to the evaluation of this term. We will discuss it in the next section.

These parameterizations will be used to represent the edge as

$$e = (L \circ \lambda)([0,1]) = (G^1 \circ \lambda)([0,1]) = (G^2 \circ \varrho)([0,1]) = (R \circ \varrho)([0,1]) \ . \quad (26)$$

Finally we define $P = L \circ \lambda = R \circ \varrho$ and arrive at

$$\begin{aligned}
&- a^e_{2,1}\left(b^1_i, b^2_j\right) = \int_e \left(\nabla b^1_i(\boldsymbol{x})|_{\Omega^1} \cdot n(\boldsymbol{x})\right) b^2_j(\boldsymbol{x})|_{\Omega^2} \mathrm{d}\boldsymbol{x} \\
&= \int_e \left[\left(\nabla G^1(\boldsymbol{x})\right)^{-1} \nabla \beta^1_i \left((G^1)^{-1}(\boldsymbol{x})\right)|_{\Omega^1} \cdot n(\boldsymbol{x})\right] \beta^2_j \left((G^2)^{-1}(\boldsymbol{x})\right)|_{\Omega^2} \mathrm{d}\boldsymbol{x} \\
&= \int_0^1 \left[\left(\nabla G^1\left(P(t)\right)\right)^{-1} \nabla \beta^1_i \left(L^{-1}\left(P(t)\right)\right) \cdot n\left(P(t)\right)\right] \beta^2_j \left(R^{-1}\left(P(t)\right)\right) \|\dot{P}(t)\| \mathrm{d}t \\
&= \int_0^1 \left[\left(\nabla G^1\left(P(t)\right)\right)^{-1} \nabla \beta^1_i \left(\lambda(t)\right) \cdot n\left(P(t)\right)\right] \beta^2_j \left(\varrho(t)\right) \|\dot{P}(t)\| \mathrm{d}t \ .
\end{aligned}$$

The integral in the last line is evaluated by a quadrature rule. However, the *choice of the quadrature rule*, which is the second major problem related to the evaluation of this term, is nontrivial and will be discussed further in Section 4. In fact, the choice of the rules needs to take the different knots of the functions β^1_i, β^2_j, λ and ϱ into account. While one will generally choose the same knots for λ and ϱ, the knots of β^1_i and β^2_j are subject to a non-linear transformation and cannot be assumed to be identical.

3 Finding the Reparameterizations

It is quite common in the literature to assume matching parameterizations or almost matching ones, see [5, p. 4148], [6, p. 87], [12,13,14,18]. In this situation, the choice of the reparameterizations λ and ϱ is trivial, as they are simply linear parameterizations (possibly reversing the orientation) of the preimages of the interface in the parameter domains. However, the restriction to matching parameterizations poses constraints on the construction of multi-patch parameterizations, making it essentially impossible to parameterize the individual patches separately. This fact motivates us to study the *non-matching* case.

More precisely, we consider situations where the condition (25) cannot be satisfied by considering linear reparameterizations λ and ϱ. Clearly, the condition does not determine λ and ϱ uniquely. We fix one of the mappings, say λ, and compute the remaining one, ϱ. Figure 2 visualizes the relations between the mappings.

The unknown mapping ϱ satisfies $\varrho = R^{-1} \circ L \circ \lambda$. We compute it by least-squares approximation of point samples, as follows:

1. For a given number N of samples, we evaluate

$$\varrho_i = R^{-1} \circ L \circ \lambda \left(\frac{i}{N}\right) \quad (27)$$

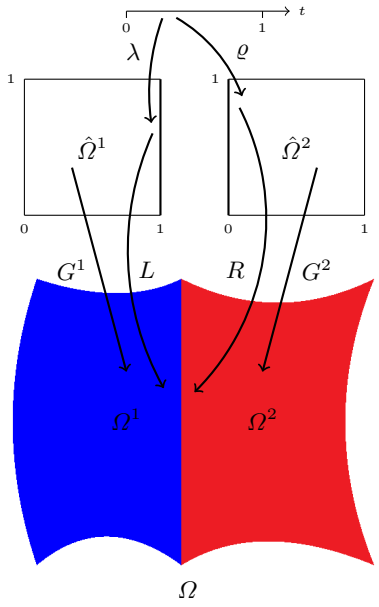

Fig. 2. Multi-patch domain with geometry maps G^1 and G^2, their restrictions L and R to the preimages of the interface and reparameterizations λ and ϱ

by performing the closest point computations

$$\varrho_i = \operatorname*{argmin}_{\boldsymbol{\xi} \in \{0\} \times [0,1]} \left\| L \circ \lambda \left(\frac{i}{N} \right) - R(\boldsymbol{\xi}) \right\|, \; i = 0, \ldots, N, \tag{28}$$

where $\| \cdot \|$ is the Euclidean norm. This formulation also applies to the case of geometrically inexact interfaces (cf. [10,11]).

2. We choose a suitable spline space (e.g. linear, quadratic or cubic splines with a few uniformly distributed inner knots) and find the control points $c_j \in \{0\} \times [0, 1]$ of the associated B-splines N_j, $j = 1, \ldots, m$, by solving the linear least-squares problem

$$\sum_{i=1}^{N} \left(\sum_{j=1}^{m} c_j N_j \left(\frac{i}{N} \right) - \varrho_i \right)^2 \longrightarrow \min, \tag{29}$$

cf. [7]. The influence of the choice of the spline space for ϱ will be discussed later in Section 5. The given reparameterization λ is chosen as a linear polynomial.

We will refer to the case where at least one of the mappings λ and ϱ is different from the identity as non-matching parameterizations at the interface.

4 Numerical Integration

The evaluation of

$$a_{2,1}^e\left(b_i^1, b_j^2\right) = \int_0^1 \left[\left(\nabla G^1\left(P(t)\right)\right)^{-1} \nabla \beta_i^1\left(\lambda(t)\right) \cdot n\left(P(t)\right)\right] \beta_j^2\left(\varrho(t)\right) \|\dot{P}(t)\| \mathrm{d}t \ . \tag{30}$$

requires integration with respect to the parameter t, which varies in the parameter domain $[0,1]$. This is done by applying numerical quadrature and we present several strategies for doing so.

4.1 Gauss Quadrature with Exact Splitting

Gauss quadrature can be applied to segments of analytic functions. Consequently, we split the parameter domain $[0,1]$ into segments (separated by junctions) where the integrand satisfies this requirement. Three types of junctions arise:

- the inverse images $\lambda^{-1}(\kappa_i^1)$ of the knots κ_i^1 that were used to define the B-splines β_j^1,
- the inverse images $\varrho^{-1}(\kappa_i^2)$ that were used to define the B-splines β_j^2, and
- the knots τ_i that were used to define the B-splines N_j in (29).

These types are visualized in Figure 3.

Consequently we perform Gauss quadrature with exact splitting by applying the following algorithm:

- Compute all junction points (all three types) in $[0,1]$,
- sort the junction points, subdivide the domain into segments accordingly,
- subdivide the resulting segments if they are too long,
- apply a Gauss quadrature rule to each segment and sum up the contributions.

As a disadvantage, the inversion of λ and ϱ is costly and has to be done with high accuracy, as the sorting depends on it. Furthermore, the method may result in many segments of varying lengths.

We use Gauss quadrature with $p+1$ nodes per element (which exactly integrates polynomials of degree $2p+1$), where p is the degree used for defining the dG-IgA discretization, cf. [15].

4.2 Gauss Quadrature with Uniform Splitting

A computationally simpler approach is to use uniform subdivision, as follows:

- Split the domain $[0,1]$ uniformly into M segments, where M is a multiple of the number of knot spans used to define the B-splines N_j in (29),
- apply a Gauss quadrature rule to each segment and sum up the contributions.

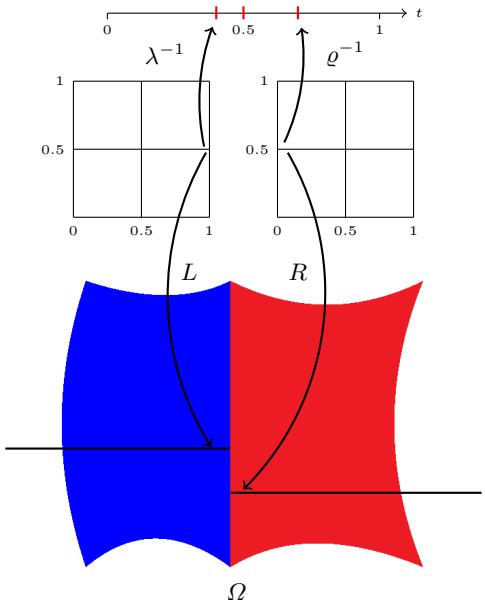

Fig. 3. Exact splitting of a knot span and application of a quadrature rule to each subsegment

As we shall see later, it is mandatory to use large values of M in order to reach the desired level of accuracy. This is due to the fact that the junctions of the first two types listed in the previous section may still be located within the segments obtained by uniform splitting. On the other hand, the use of uniform refinement also creates many small segments that could be merged into larger ones without compromising the accuracy. This can be exploited by using adaptive quadrature.

4.3 Adaptive Gauss Quadrature

We recall the main idea of adaptive quadrature, cf. [8]. In order to evaluate the integral

$$I = \int_a^b f(x)\mathrm{d}x \tag{31}$$

of an integrable function f over an interval $[a, b]$ adaptively one computes two different estimates I_1 and I_2 of I by using two different integration methods. One assumes that one of these estimates, say I_1, is more accurate than the other. Next, one computes the relative distance between I_1 and I_2 taking into account a given (or chosen) tolerance tol, e.g. machine precision. If the difference is small enough, one chooses I_1 as the value of the integral $\int_a^b f(x)\mathrm{d}x$. If this is not the case one splits the interval $[a, b]$ into two subintervals,

$$[a, b] = [a, m] \cup [m, b] \ , \quad \text{where} \quad m = \frac{a+b}{2},$$

and evalues I by summing up the two contributions. This means that one applies the procedure of computing two different estimates and checking their relative difference to both subintervals. Adaptive quadrature is therefore a recursive procedure, which is summarized in Algorithm 1.

Algorithm 1 Adaptive Quadrature: Basic routine.
adaptiveQuadrature(f, a, b, tol)

1: Input: f , a , b , tol where f is an integrable function, a and b are the interval boundaries and tol is a given tolerance
2: Choose knots u_i and weights w_i , $i = 1, \dots, n$.
3: Compute $I_1 = \sum_{i=1}^{n} w_i f(u_i)$.
4: Choose knots $\tilde{u}_i$ and weights $\tilde{w}_i$, $i = 1, \dots, m$.
5: Compute $I_2 = \sum_{i=1}^{m} \tilde{w}_i f(\tilde{u}_i)$.
6: **if** $|I_1 - I_2| \leq \text{tol} \cdot |I_1|$ **then**
7: Return I_1
8: **else**
9: Return

$$\text{adaptiveQuadrature}\left(f, a, \frac{a+b}{2}, \text{tol}\right) + \text{adaptiveQuadrature}\left(f, \frac{a+b}{2}, b, \text{tol}\right) \ .$$

10: **end if**

Note that the stopping criterion has to be chosen with care and in fact line 6 in the algorithm is a slight oversimplification of it. See [8] for further information.

We apply this procedure to the knot spans that were used to define the B-splines N_j in (29). Therefore we choose I_1 as a Gauss quadrature rule with $p + 1$ quadrature knots, where again p is the degree of the basis functions in the dG-IgA discretization space. For the computation of I_2 we split the interval manually into two halves, apply a a Gauss quadrature rule of the same order on both halves, and sum up. The tolerance tol is set to machine precision.
As an advantage, adaptive quadrature can be performed without inverting the reparameterizations. Moreover, it avoids the oversegmentation problem that was present for the previous approach. We observed experimentally that the adaptive procedure accurately detects the junction points and subdivides the domain accordingly. Clearly, the implementation is more costly and requires a recursive algorithm.

5 Numerical Results

We examine the performance of the quadrature methods presented in Section 4 as well as the influence of the accuracy of the reparameterization. All experiments were performed using G+Smo[4], an object-oriented C++ IgA library named "Geometry + Simulation Modules".

[4] G+Smo: gs.jku.at

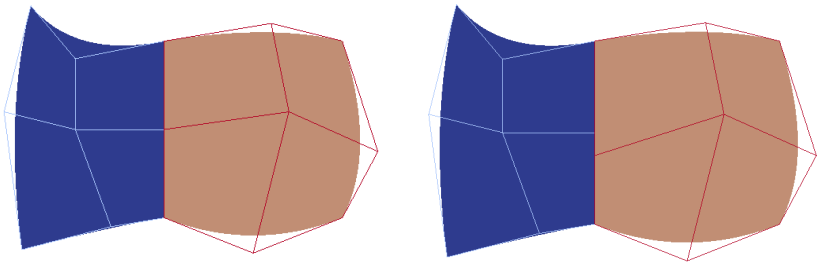

Fig. 4. Patch and its control net. Left: matching parameterizations at the interface. Right: non-matching parameterizations at the interface.

5.1 Reference Results

As a reference we will first show the convergence plot of the solution of the Poisson equation in the case of matching parameterizations, i.e. for $\lambda = \varrho = \mathrm{id}$. In this case we can restrict ourselves to a simple quadrature rule. There is no need for using more elaborate versions of numerical integration. Furthermore, since $\lambda = \varrho = \mathrm{id}$, we do not need to consider the influence of the quality of the reparameterization. More precisely, we consider the two-patch domain with biquadratic matching interface parameterizations shown in Figure 4, left.

Figure 5 demonstrates the convergence behaviour of the numerical solutions that were obtained for various values of the element size h that was used to define the dG-IgA discretization. We consider a problem with a known solution and measure the error as the difference to it. The quadrature method we used is Gauss quadrature with three quadrature knots. A convergence rate of three for the L2 error and of two for the dG error is clearly visible. This is in accordance with the theoretical predictions, see [1,6].

5.2 Influence of the Quadrature Rule

We now consider a different parameterization of the same computational domain, with non-matching parameterizations of the interface, see Figure 4, right. Again we use biquadratic patches. Now we need to use a more complicated integration technique, and we consider the three approaches that were described in Section 4.

Figure 6, left and right, visualizes the convergence behaviour measured in the L2 and dG norms respectively. Each plot contains four curves, corresponding to four different numerical quadrature techniques. More precisely, we consider Gauss quadrature with exact splitting (yellow), Gauss quadrature with uniform splitting into 10 (blue) and into 30 (red) segments, and adaptive Gauss quadrature (purple). We observe that the first and the last method perform better than the results based on uniform splitting and they achieve the optimal convergence rates (compare with Figure 5). In particular we note that using uniform quadrature leads to a reduced order of convergence for smaller mesh sizes h.

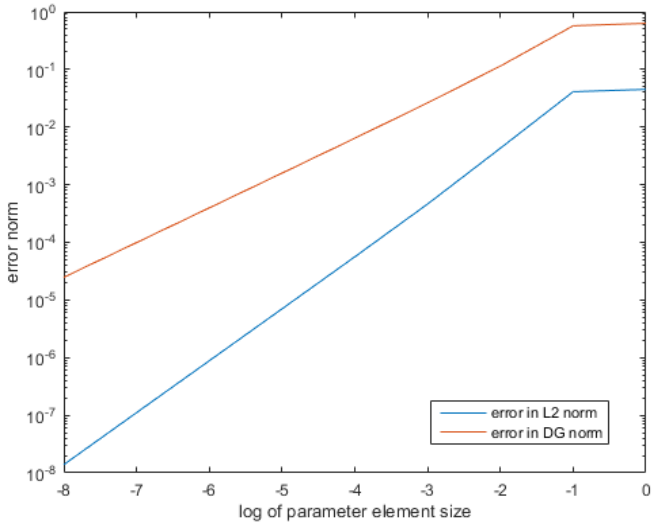

Fig. 5. Matching parameterizations at the interface, convergence behaviour of error in different norms: L2 norm (blue curve), dG norm (red curve).

Even the use of a very fine but uniform segmentation (30 (red) instead of 10 (blue) segments) does not improve this significantly.

Based on these observations we decided to use solely adaptive and exact Gauss quadrature in the remaining example.

5.3 Influence of the Reparameterization

Next we analyse the influence of the quality of the representation of the reparameterization. Consider again the parameterization of the domain in Figure 4, right, with non-matching parameterizations of the interface. We compare three different choices of the reparameterizations λ and ϱ.

For the first reparameterization, which generates the results represented by the blue curve in Figure 7, we choose polynomials λ and ϱ such that the equation $L \circ \lambda = R \circ \varrho$ is exactly satisfied. In this case it was possible to find such polynomials, due to the particular construction of the example. However, this would be impossible in general and it is used here to generate a reference result.

The second and third reparameterizations (red and yellow curves) were obtained using the Algorithm from Section 3 to find ϱ, while λ was chosen as a linear polynomial. The second reparameterization uses a linear spline with 8 segments and has an L2 error of $1.3 \cdot 10^{-2}$, and the third reparameterization uses a cubic spline with 4 segments and has an L2 error of $3.1 \cdot 10^{-15}$.

Figure 7 compares the errors in the L2 (left) and dG norms (right) for the three reparameterizations. We observe that using a high quality reparameterization is essential for the convergence of the method. Depending on the accuracy of

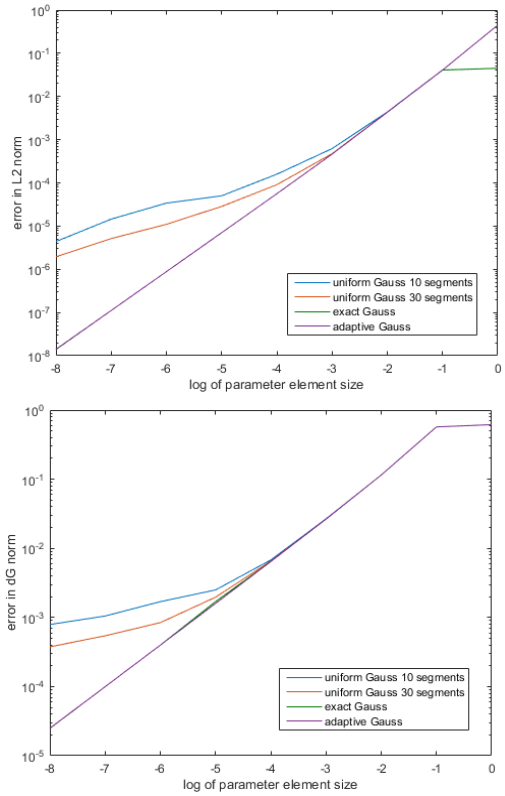

Fig. 6. Influence of the quadrature rule. Left: Convergence behavior of the error in L2 norm. Right: Convergence behavior of the error in dG norm. Blue and red curves: 10 and 30 uniform segments per t-knot span. Yellow curves: exact splitting of the knot spans. Purple curves: adaptive quadrature. Note that the yellow curve coincides with the purple one for smaller values of h. Exact representation of the reparameterizations λ and ϱ.

the reparameterization, h-refinement only works until it reaches a critical mesh size, where further refinement does not have any effect.

The plots show the results obtained by using adaptive Gauss quadrature. The exact method gives virtually identical results.

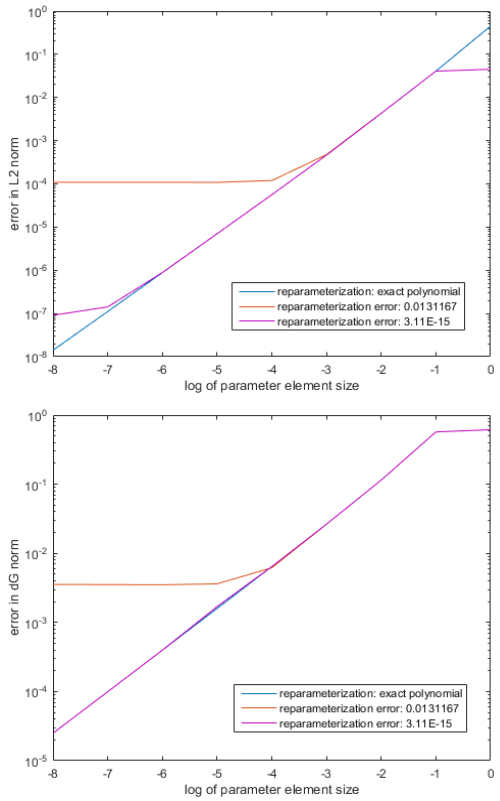

Fig. 7. Influence of the reparameterization. Adaptive quadrature on interface integrals. Left: Convergence behaviour of the error in L2 norm. Right: Convergence behaviour of the error in dG norm. Blue curves: Exact representation of λ and ϱ. Red curves: Approximation error of $\varrho \approx 0.0131167$. Yellow curves: Approximation error of $\varrho \approx 3.10616 \cdot 10^{-15}$

5.4 Comparison of Exact and Adaptive Quadrature

We perform an experimental comparison of the computational complexity of exact and adaptive quadrature for the domain in Figure 4, right.

First we demonstrate the effect of using adaptive quadrature, by showing the automatically created splitting points in Figure 8. We used an accuracy of

10^{-6} instead of machine precision for this picture to obtain a clearer image. Both patches were uniformly refined into 4×4 elements by knot insertion. The mappings λ and ϱ are cubic splines on $[0, 1]$ with four knot spans of equal length. Their knots τ_i coincide with the inverse images $\lambda^{-1}(\kappa_i^1)$, as the first mapping is simply the identity. The adaptive quadrature, which is applied to the knot spans $[\tau_i, \tau_{i+1}]$, thus creates additional splitting points around the inverse images $\varrho^{-1}(\kappa_i^2)$, as shown in the Figure. In this particular case, only one splitting point (at 0.5625) is created near $\varrho^{-1}(\kappa_2^2) = 0.5615$ since this suffices to reach the desired accuracy.

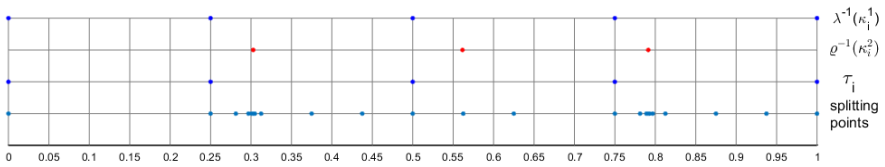

Fig. 8. Splitting points created by adaptive quadrature - see text for details.

These results indicate that, unlike uniform Gauss quadrature, adaptive quadrature avoids over-segmentation of the integration domains. Still, it splits the knot spans more often than exact Gauss quadrature, which also results in a higher number of quadrature knots and thus evaluations.

In order to analyze this effect, Figure 9 compares the number of evaluations (i.e., quadrature knots) used by exact and adaptive Gauss quadrature for increasing numbers of elements. In addition, we also show the number of root finding operations (which are more expensive than evaluations) needed to compute the splitting points of exact Gauss quadrature. Clearly, adaptive quadrature requires more evaluations than exact splitting. However, for sufficiently fine discretizations, the number of evaluations in the interior of the patches dominates the total effort.

6 Conclusion

We used a simple model problem to investigate the complications that arise from using non-matching interface parameterizations within the framework of Isogeometric Analysis on a multi-patch domain, using discontinuous Galerkin techniques to couple terms across the interfaces. More precisely, we studied two particular problems. Firstly, we explored the use of reparameterizations to identify pairs of associated points on the common interface. This was found to be useful for correctly evaluating certain terms in the dG discretization. Secondly, we addressed the construction of a suitable procedure for numerical integration. As demonstrated in our numerical experiments, both problems are important for ensuring the optimal rate of convergence for the numerical simulation based on the isogeometric dG discretization.

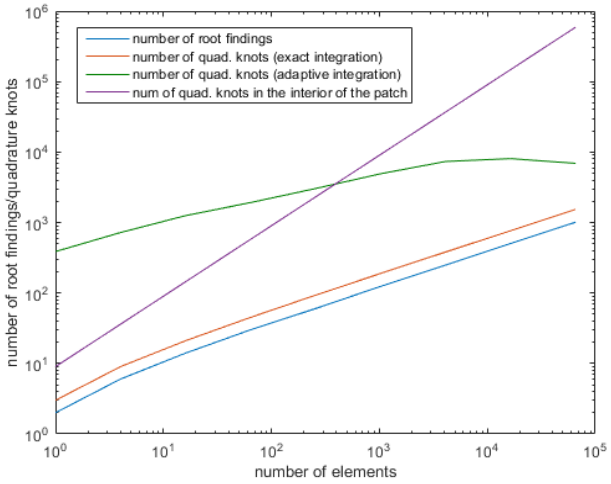

Fig. 9. Number of quadrature knots and root finding operations needed by exact and adaptive quadrature for increasingly finer discretizations.

Future work may be devoted to the extension of the adaptive quadrature-based approach to the three-dimensional case. Moreover, we are currently studying dG-type techniques for performing multi-patch spline surface fitting with approximate geometric smoothness across patch interfaces.

References

1. Bazilevs, Y., de Veiga, L.B., Cottrell, J.A., Hughes, T.J., Sangalli, G.: Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. Mathematical Models and Methods in Applied Sciences 16, 1031 – 1090 (2006)
2. Brivadis, E., Buffa, A., Wohlmuth, B., Wunderlich, L.: Isogeometric mortar methods. Computer Methods in Applied Mechanics and Engineering 284, 292–319 (2015)
3. Brunero, F.: Discontinuous Galerkin Methods for Isogeometric Analysis. Master's thesis, Università degli Studi di Milano (2012)
4. Cockburn, B.: Discontinuous Galerkin methods. Journal of Applied Mathematics and Mechanics 83, 731–754 (2003)
5. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. Computer Methods in Applied Mechanics and Engineering 194, 4135–4195 (2005)
6. Cottrell, J.A., Hughes, T.J.R., Bazilevs, Y.: Isogeometric Analysis. Toward Integration of CAD and FEA. John Wiley and Sons, Chichester, England (2009)
7. Dierckx, P.: Curve and Surface Fitting with Splines. Monographs on Numerical Analysis, Oxford Science Publications (1995)
8. Gander, W., Gautschi, W.: Adaptive quadrature - revisited. BIT Numerical Mathematics 40(1), 84 – 101 (2000)
9. Hofer, C.: personal communication

10. Hofer, C., Langer, U., Toulopoulos, I.: Discontinuous Galerkin isogeometric analysis of elliptic diffusion problems on segmentations with gaps. SIAM Journal on Scientific Computing (2016), accepted manuscript, also available at arXiv:1511.05715
11. Hofer, C., Toulopoulos, I.: Discontinuous Galerkin isogeometric analysis of elliptic problems on segmentations with non-matching interfaces. Computers and Mathematics with Applications 72, 1811 – 1827 (2016)
12. Langer, U., Mantzaflaris, A., Moore, S.E., Toulopoulos, I.: Multipatch discontinuous Galerkin isogeometric analysis. In: Jüttler, B., Simeon, B. (eds.) Isogeometric Analysis and Applications. pp. 1–32. Springer, Heidelberg (2014), also available as NFN Technical Report No. 18 at www.gs.jku.at
13. Langer, U., Moore, S.E.: Discontinuous Galerkin isogeometric analysis of elliptic PDEs on surfaces (2014)
14. Langer, U., Toulopoulos, I.: Analysis of multipatch discontinuous Galerkin IgA approximations to elliptic boundary value problems. Computing and Visualization in Science 17(5), 217–233 (2016)
15. Mantzaflaris, A., Jüttler, B.: Integration by interpolation and look-up for Galerkin-based isogeometric analysis. Computer Methods in Applied Mechanics and Engineering 284, 373 – 400 (2015)
16. Nguyen, V.P., Kerfriden, P., Brino, M., Bordas, S.P., Bonisoli, E.: Nitsche's method for two and three dimensional NURBS patch coupling. Compuational Mechanics 53(6), 1163–1182 (2014)
17. Rivière, B.: Discontinuous Galerkin Methods for Solving Elliptic and Parabolic Equations: Theory and Implementation. SIAM (2008)
18. Zhang, F., Xu, Y., Chen, F.: Discontinuous Galerkin methods for isogeometric analysis for elliptic equations on surfaces. Communications in Mathematics and Statistics 2(3), 431–461 (2014)