TU Clausthal

Mathematik-Bericht 2010/4

Numerical Solution of Convection-dominated Problems Using Isogeometric Analysis

F. Freiberger

April 2010

Institut für Mathematik, TU Clausthal, Erzstraße 1, D-38678 Clausthal-Zellerfeld, Germany

Numerical Solution of Convection-dominated Problems Using Isogeometric Analysis

Falk Freiberger

April 29, 2010

Contents

1 Introduction

In many fields of engineering science finding approximate solutions of partial differential equations is a major task. Numerical solutions can be found using finite element methods (FEM). A representative example is the stationary Stokes equation which can be derived from the well known Navier-Stokes equation which is fundamental in fluid mechanics [Che05].

Partitioning and refinement of a non-rectangular domain Ω can be a serious problem. In [HCB05] a finite element method called "isogeometric analysis" is presented. It solves partial differential equations numerically and provides an easier treatment of non-rectangular domains. Domains are described using a basis of non-uniform rational B-splines (NURBS). The same set of basis functions is used to give a representation of the numerical solution. With the help of NURBS it becomes much easier to refine a given domain Ω.

The paper is organized as follows. First, a review of the theory of Bézier curves and NURBS is given. As we apply isogeometric analysis to convection dominated diffusion-convection problems, a stabilization of the numerical solution is necessary. Therefore, the streamline diffusion method (sdFEM) is discussed in Section 3. Isogeometric analysis is tested in the one-dimensional (Section 4) and two-dimensional (Section 5) cases. A more difficult numerical problem with variable convection is considered in Section 6. We conclude with a summary in Section 7.

2 Bézier-Curves and NURBS

NURBS or rational B-spline curves are a generalization of B-spline curves. The use of these functions is standard in CAD and CAM industries [FHK02]. Rational B-spline curves are defined as the projection of a non-rational B-spline curve of $\mathbb{R}^4$ into a hyperplane [Far97].

2.1 Bézier Curves

A Bézier curve of degree n is a curve in $\mathbb{R}^3$ which is described by $n+1$ control points or Bézier points $b_0, \ldots, b_n \in \mathbb{R}^3$. We calculate a point at the curve by means of the following algorithm:

Let $t \in [0,1]$. Set $b_i^0(t) = b_i$.
For $r = 1, \ldots, n$
  For $i = 0, \ldots, n-r$
    $b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \in \mathbb{R}^3$.

This is the **de Casteljau Algorithm**, $b_0^n(t)$ is the point on the **Bézier curve** $b^n : [0,1] \to \mathbb{R}^3$ for the parameter t. It is obvious that in the general case no Bézier points besides b_0 and b_n are interpolated by the Bézier curve. Nevertheless every point $b^n(t)$ is situated inside the convex hull of the control polygon $b_0, \ldots, b_n$. A Bézier curve is invariant under affine parameter transformations. Furthermore such a curve is symmetric, i.e.

$$\sum_{j=0}^{n} b_j B_j^n(t) = \sum_{j=0}^{n} b_{n-j} B_j^n(1-t),$$

and invariant under barycentric combinations, i.e. for $\alpha + \beta = 1$ it holds

$$\sum_{j=0}^{n} (\alpha b_j + \beta c_j) B_j^n(t) = \alpha \sum_{j=0}^{n} b_j B_j^n(t) + \beta \sum_{j=0}^{n} c_j B_j^n(t).$$

2.2 Bernstein Polynomials

Bernstein polynomials are used to derive an explicit representation of a Bézier curve. The i-th **Bernstein polynomial** of degree n is defined by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \quad \text{with } \binom{n}{i} = 0, \text{ if } 0 \le i \le n \text{ is violated.}$$

Bernstein polynomials satisfy the recursion formula

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t)$$

with $B_0^0(t) \equiv 1$ and $B_j^n(t) \equiv 0$ for $j \neq \{0, \ldots, n\}$. For $t \in [0,1]$ these polynomials are nonnegative and they form a partition of unity, i.e.

$$\sum_{j=0}^{n} B_j^n(t) = 1.$$

We obtain two explicit representations of a Bézier curve b^n

$$b^n(t) = \sum_{i=0}^{n} b_j B_j^n(t) = \sum_{i=0}^{n-r} b_i^r(t) B_i^{n-r}(t) \tag{1}$$

with the control points $b_0, \ldots, b_n$ and the intermediate points b_i^r from the de Casteljau algorithm.

2.3 Derivatives of a Bézier Curve

Owing to the explicit representation (1) we can calculate all derivatives of a Bézier curve. We define the forward difference operator $\Delta b_j = b_{j+1} - b_j$ and find

$$\frac{d}{dt} b^n(t) = n \sum_{j=0}^{n-1} \Delta b_j B_j^{n-1}(t).$$

So we see that the first derivative is nothing but a Bézier curve with control points Δb_j. Higher derivatives can be found by iteration. We introduce the iterated forward difference operator $\Delta^r b_j = \Delta^{r-1} b_{j+1} - \Delta^{r-1} b_j$. We find

$$\Delta^r b_i = \sum_{j=0}^{r} \binom{r}{j} (-1)^{r-j} b_{i+j}$$

and the formula for the r-th derivative

$$\frac{d^r}{dt^r} b^n(t) = \frac{n!}{(n-r)!} \sum_{j=0}^{n-r} \Delta^r b_j B_j^{n-r}(t) = \frac{n!}{(n-r)!} \Delta^r b_0^{n-r}(t). \tag{2}$$

Considering the special case $t = 0$ this formula yields

$$\frac{d^r}{dt^r} b^n(0) = \frac{n!}{(n-r)!} \Delta^r b_0,$$

i.e. the r-th derivative only depends on the control points $b_0, \ldots, b_r$. Hence the first derivative in $t = 0$ is

$$\frac{d}{dt} b^n(0) = n(b_1 - b_0). \tag{3}$$

An analogous formula can be derived in the case $t = 1$. In view of (2) we have the following two possibilities of calculating the r-th derivative:

- Calculate the r-th forward differences and interpret the resulting points as control points of a Bézier curve. This curve can be evaluated by means of the de Casteljau algorithm.
- Use the intermediate points from the de Casteljau algorithm and calculate the r-th forward differences.

Because of

$$\frac{d}{dt} b^n(t) = n \left(b_1^{n-1}(t) - b_0^{n-1}(t) \right)$$

for the first derivative, i.e. $r = 1$, the tangent in the point $b^n(t)$ only depends on the two intermediate points $b_0^{n-1}(t)$ and $b_1^{n-1}(t)$.

2.4 Degree Elevation and Reduction

Our aim is to add control points to an existing Bézier curve without changing the form of the curve. This process is called degree elevation. Suppose that the control points $b_0, \ldots, b_n$ describe a Bézier curve. We want to find control points $b_0^{(1)}, \ldots, b_{n+1}^{(1)}$ so that we still get the same Bézier curve. These new points are given by

$$b_i^{(1)} = \frac{i}{n+1} b_{i-1} + \left(1 - \frac{i}{n+1}\right) b_i, \quad i = 0, \ldots, n+1.$$

This is a linear interpolation of the given control points. If we elevate the degree successively r times new control points $b_0^{(r)}, \ldots, b_{n+r}^{(r)}$ have to be found. These points can be described by

$$b_i^{(r)} = \sum_{j=0}^{n} b_j \binom{n}{j} \frac{\binom{r}{i-j}}{\binom{n+r}{i}}.$$

In the limit $r \to \infty$ the points $b_0^{(r)}, \ldots, b_{n+r}^{(r)}$ converge towards the Bézier curve.

In general, it is not possible to describe a Bézier curve with less than the initial number of control points. That means that there is no exact degree reduction. Therefore we search for the best approximation of the curve. To do so we inverse the process of degree elevation. We start again with control points $b_0, \ldots, b_n$ and try to find points $\hat{b}_0, \ldots, \hat{b}_{n-1}$. We assume that the points $b_0, \ldots, b_n$ have been calculated with the degree elevation formula from the points $\hat{b}_0, \ldots, \hat{b}_{n-1}$. This means

$$b_i = \frac{i}{n} \hat{b}_{i-1} + \frac{n-i}{n} \hat{b}_i, \quad i = 0, \ldots, n.$$

In the next step two recursion formulas can be obtained:

$$\overrightarrow{b}_i = \frac{n b_i - i \overrightarrow{b}_{i-1}}{n-i} \qquad \text{for } i = 0, \ldots, n-1,$$

$$\overleftarrow{b}_i = \frac{n b_i - (n-i) \overleftarrow{b}_{i-1}}{i} \qquad \text{for } i = n, n-1, \ldots, 1.$$

Both approximations are bad in most cases. To improve them one could choose a linear combination of both approximations which leads to

$$\hat{b}_i = (1 - \lambda_i) \overrightarrow{b}_i + \lambda_i \overleftarrow{b}_i, \quad i = 0, \ldots, n-1.$$

The parameter λ_i has to be chosen appropriately, for example by using Chebychev polynomials. We set

$$\lambda_i = \frac{1}{2^{2n-1}} \sum_{j=0}^{i} \binom{2n}{2j}.$$

The maximal deviation between the given Bézier curve and the new curve of lower degree is $||\Delta^n b_0|| 2^{-(2n-1)}$. Unfortunately, this choice of λ_i can destroy the end point interpolation property of a Bézier curve.

2.5 Spline Curves in Bézier Form

A spline curve s is a continuous map of a collection of intervals $u_0 < \ldots < u_L$ into $\mathbb{R}^3$, where each interval $[u_i, u_{i+1}]$ is mapped onto a polynomial curve segment. The set of the real numbers u_i is called a **knot sequence**. On the interval $[u_i, u_{i+1}]$ we define a local variable $t \in [0, 1]$. We introduce the notation $\Delta_i := u_{i+1} - u_i$ for the length of the interval $[u_i, u_{i+1}]$ and set

$$t = \frac{u - u_i}{u_{i+1} - u_i} = \frac{u - u_i}{\Delta_i} \quad \text{for } u \in [u_i, u_{i+1}].$$

The curve of the interval $[u_i, u_{i+1}]$ is called $s_i := s|_{[u_i,u_{i+1}]}$. We want the global curve to fulfil certain global smoothness conditions. Let $s_1 : [u_0, u_1] \to \mathbb{R}^3$ and $s_2 : [u_1, u_2] \to \mathbb{R}^3$ be two Bézier curves defined by their control polygons $b_0, \ldots, b_n$ and $b_n, \ldots, b_{2n}$. From (3) we conclude that the spline curve formed by s_1 and s_2 is a C^1-curve if and only if the points b_{n-1}, b_n and b_{n+1} are collinear and the equation

$$\frac{u_1 - u_0}{u_2 - u_1} = \frac{||b_1 - b_0||}{||b_2 - b_1||}$$

holds. Additionally, C^2 continuity can only be obtained in case the control points b_{n-2}, b_{n-1}, b_n and b_n, b_{n+1}, b_{n+2} describe the same quadratic polynomial. Hence, this polynomial can be specified by a polygon b_{n-2}, d, b_{n+2} with an unknown point d. From this polygon we can find the other two polygons employing a subdivision process [Far97]. As a consequence we see

$$b_{n-1} = (1 - t_1)b_{n-2} + t_1 d \quad \text{and} \quad b_{n+1} = (1 - t_1)d + t_1 b_{n+2}$$

with $t_1 = \frac{u_1 - u_0}{u_2 - u_0}$. These two equations can be used to calculate two points d^+ and d^-. For $d^+ = d^-$ the curve is C^2 continuous.

These differentiability conditions allow us to construct most of the well known quadratic and cubic C^1 and C^2 spline curves. For example, the cubic C^2 spline curve results from the following construction: Let $u_0, \ldots, u_L$ be a set of L intervals and d_i auxiliary points with $i = 0, \ldots, L - 1$. The conditions for C^1 and C^2 continuity yield the representation

$$b_{3i} = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} b_{3i-1} + \frac{\Delta_{i-1}}{\Delta_{i-1} + \Delta_i} b_{3i+1}.$$

for the connection point b_{3i}. The other points of the curve can be calculated by means of the formulas

$$b_{3i-2} = \frac{\Delta_{i-1} + \Delta_i}{\Delta} d_{i-1} + \frac{\Delta_{i-2}}{\Delta} d_i \quad \text{and} \quad b_{3i-1} = \frac{\Delta_i}{\Delta} d_{i-1} + \frac{\Delta_{i-2} + \Delta_{i-1}}{\Delta} d_i$$

for $i = 2, \ldots, L - 1$ with $\Delta = \Delta_{i-2} + \Delta_{i-1} + \Delta_i$. A modified formula has to be used to get the boundary points.

The advantage of the cubic spline curves consists in their flexibility. Whereas quadratic curves are piecewise planar the cubic ones are true space curves. Besides the convex hull property, affine invariance, symmetry and endpoint interpolation locality is one of the main features of the spline curves. That means that changing one Bézier point only has a local impact on the curve.

2.6 Knot Sequences

If only data points $x_i \in \mathbb{R}^3$ are given there are several possibilities to find a parametrization. The easiest approch is the uniform parametrization, i.e. $u_i = i$. Other options include the chord length parametrization where u_i is chosen to fulfil

$$\frac{\Delta_i}{\Delta_{i+1}} = \frac{||\Delta x_i||}{||\Delta x_{i+1}||}$$

with $u_0 = 0$, $u_L = 1$ and the centripetal parametrization [Far97]

$$\frac{\Delta_i}{\Delta_{i+1}} = \left(\frac{||\Delta x_i||}{||\Delta x_{i+1}||} \right)^{\frac{1}{2}}.$$

2.7 B-spline Curves and B-splines

In this section we consider curves that can be obtained as the graph of a function $f : \mathbb{R} \to \mathbb{R}$. Let $n \in \mathbb{N}$ be the degree of the B-spline curve and $L \in \mathbb{N}$ the maximal number of polynomial segments. Let

$$u_0 \leq \ldots \leq u_{L+2n-2} \quad \text{with} \quad u_i \in \mathbb{R},$$

be a knot sequence. For $u_i = \ldots = u_{i+r_i-1}$ the knot u_i has **multiplicity** r_i. In case $r_i = 1$ the knot u_i is a **simple knot**, $u_{n-1}, \ldots, u_{L+n-1}$ are **domain knots**. Obviously $\sum_{i=n-1}^{L+n-1} r_i = L + 1$ holds. The **Greville abscissas** are defined by

$$\xi_i = \frac{1}{n}(u_i + \ldots u_{i+n-1}), \quad i = 0, \ldots, L + n - 1.$$

If we assign to every Greville abscissa a **de Boor ordinate** $d_i \in \mathbb{R}$ then the points (ξ_i, d_i) with $i = 0, \ldots, L + n - 1$ form a polygon P.

Now we want to insert a knot $u \in [u_{n-1}, u_{L+n-1}]$ into the existing knot sequence. As a consequence we have to calculate new Greville abscissas ξ_i^u and de Boor ordinates d_i^u. This results in a new polygon P^u. We evaluate P to find the new de Boor ordinates, i.e. $d_i^u = P(\xi_i^u)$. Hence, the formula

$$d_i^u = \frac{u_{i+n-1} - u}{u_{i+n-1} - u_{i-1}} d_{i-1} + \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} d_i, \quad i = I - n + 2, \ldots, I + 1$$

for $u \in [u_I, u_{I+1}]$ holds. Note that not all Greville abscissas have to be calculated, only a certain number in a neighbourhood of the knot u change their value. For the implementation of the algorithm it is expedient to store knot values and the multiplicities in separate vectors.

Once a knot u reaches multiplicity n further insertion of u into the knot sequence does not change the polygon. We start with a knot sequence and the de Boor ordinates which form a B-spline curve of degree n. Next we refine the knot sequence at the position u until u possesses multiplicity n. Then the polygon vertice associated to u is nothing but the function value $s(u)$. We call $B_n P$ the **B-spline curve of degree** n with control polygon P. Due to the location of the Greville abscissas this curve is only defined in the interval $[u_{n-1}, u_{l+n-1}]$. Let

$$u \in [u_I, u_{I+1}) \subset [u_{n-1}, u_{L+n-1}]$$

and

$$d_i^k(u) = \frac{u_{i+n-k} - u}{u_{i+n-k} - u_{i-1}} d_{i-1}^{k-1}(u) + \frac{u - u_{i-1}}{u_{i+n-k} - u_{i-1}} d_i^{k-1}(u) \tag{4}$$

for $k = 1, \ldots, n - r$ und $i = I - n + k + 1, \ldots, I - r + 1$. This implies

$$s(u) = [B_n P](u) = d_{I-r+1}^{n-r}(u).$$

Here r is the initial multiplicity of the knot u and $d_i^0(u) = d_i$. Repeated insertion of a knot u into the knot sequence and application of (4) to derive a new polygon is called the **de Boor algorithm**. In the special case

$$0 = u_0 = u_1 = \ldots = u_{n-1} < u_n = u_{n+1} = \ldots = u_{2n-1} = 1$$

both knots u_0 and u_n are of multiplicity n. The Greville abscissas are

$$\xi_i = \frac{1}{n} \sum_{j=i}^{i+n-1} u_j = \frac{i}{n}, \quad i = 0, \ldots, n.$$

For $0 \le u \le 1$ we conclude from the de Boor algorithm for $I = n - 1$

$$d_i^k(u) = (1 - u) d_{i-1}^{k-1} + u d_i^{k-1}, \quad k = 1, \ldots, n.$$

Due to $i \ge k$ and $n \ge i$ we see $u_{i+n-k} = 1$, $u_{i-1} = 0$ for all i, k. So we found again the de Casteljau algorithm. Now we summarize some consequences:

- Given two neighbouring knots of multiplicity n, the corresponding B spline curve is a Bézier curve between these two knots. The B spline control polygon is identical to the Bézier polygon, the Greville abscissas are equally spaced.

- After inserting u into the knot sequence as long as it reaches multiplicity n the initial Bézier polygon is split into two Bézier polygons. These polygons describe the same curve as the initial polygon. Hence, the de Casteljau algorithm subdivides Bézier curves.
- Now we refine a given knot sequence until every knot is of multiplicity n. The resulting B-spline polygon is the piecewise Bézier polygon of the curve. So B-spline curves are piecewise polynomial on the interval $[u_{n-1}, u_{L+n-1}]$.
- A B-spline curve is C^{n-r} in knots with multiplicity r.

Let $u_0, \ldots, u_k$ be a knot sequence. Given a set of piecewise defined polynomials of degree n. We assume that in every knot u_i all functions are C^{n-r_i}. The dimension of the resulting vector space is $(n+1) + \sum_{i=1}^{k-1} r_i$ which can be proven easily by counting the degrees of freedom.

We now consider B-spline curves that are piecewise polynomial on the interval $[u_{n-1}, u_{L+n-1}]$. This is a vector space of dimension $L+n$ which equals the number of the resulting Greville abscissas. The de Boor ordinates d_i can be used to describe a basis of this vector space. Its elements are called **B-splines** $N_i^n(u)$:

$$d_i = 1 \quad \text{and} \quad d_j = 0 \quad \text{for all } j \neq i.$$

These functions $N_i^n(u)$ possess several useful properties:

- They are piecewise polynomials on the interval $[u_{n-1}, u_{L+n-1}]$.
- They have a local support, i.e. $N_i^n(u) \neq 0$ only if $u \in [u_{i-1}, u_{i+n}]$.
- They even have a minimal support: If there exists a piecewise polynomial with the same differentiation properties like $N_i^n(u)$ but smaller support then this function is zero function.
- They are linearly independent.
- They form a partition of unity.

Therefore, every piecewise polynomial s in the interval $[u_{n-1}, u_{L+n-1}]$ can be written uniquely in the form

$$s(u) = \sum_{j=0}^{L+n-1} d_j N_j^n(u).$$

Useful recursion formulas for B-splines can be derived. Let $\hat{u}$ be a knot that is inserted into an existing knot sequence. Assume that N_i^n are the B-splines of the old knot sequence and $\hat{N}_i^n$ the new ones. We obtain the Boehm recursion

$$N_l^n(u) = \frac{\hat{u} - u_{l-1}}{u_{l+n-1} - u_{l-1}} \hat{N}_l^n(u) + \frac{u_{l+n} - \hat{u}}{u_{l+n} - u_l} \hat{N}_{l+1}^n(u)$$

and the Mansfield, de Boor or Cox recursion

$$N_l^n(u) = \frac{u - u_{l-1}}{u_{l+n-1} - u_{l-1}} N_l^{n-1}(u) + \frac{u_{l+n} - u}{u_{l+n} - u_l} N_{l+1}^{n-1}(u). \tag{5}$$

Thus, a B-spline of degree n can be interpreted as a strict convex combination of two B-splines of lower degree. Formula (5) is numerically stable. For the derivative we find

$$\frac{d}{du} N_l^n(u) = \frac{n}{u_{n+l-1} - u_{l-1}} N_l^{n-1}(u) - \frac{n}{u_{l+n} - u_l} N_{l+1}^{n-1}(u). \tag{6}$$

Given the polygon P, inserting r knots successively leads to the polygon P^r. Now we insert more knots until the set of the inserted knots is dense in the interval $[u_{n-1}, u_{L+n-1}]$. Then

$$\lim_{r \to \infty} P^r = [B_n P] \tag{7}$$

holds. For a proof we observe that the polygon P^r describes the same curve as the original polygon P and we use the convex hull property. The convergence result (7) is applied in rendering of B-spline curves. Knots are inserted until a certain precision level is reached.

Now we summarize some properties of B-spline curves:

- **Linear precision**: Let $l(u) = au + b$ be a straight line and ξ_i the Greville abscissas. Then

$$\sum l(\xi_i) N_i^n(u) = l(u).$$

- **Strong convex hull property**: Every point on the curve is located inside the convex hull of at most $n+1$ neighbouring control points.
- **Variation diminishing property**: The curve is not intersected by a straight line more often than the polygon.
- **Derivative**: An explicit formula for the derivative of a B-spline curve s is given by

$$\frac{d}{du} s(u) = n \sum_{i=1}^{L+n-1} \frac{\Delta d_{i-1}}{u_{n+i-1} - u_{i-1}} N_i^{n-1}(u).$$

- **Degree elevation**:

$$N_i^n(u) = \frac{1}{n+1} \sum_{j=i-1}^{n+i} N_i^{n+1}(u; u_j).$$

  Here the expression $N_i^{n+1}(u; u_j)$ means that we use the same knot sequence as before. Only the multiplicity of the knot u_j is augmented by 1.

2.8 Rational Bézier and B-Spline Curves

A **conic section** in $\mathbb{R}^2$ is the projection of a parabola of $\mathbb{R}^3$ into a plane. We identify a point $\begin{pmatrix} x & y \end{pmatrix}^T \in \mathbb{R}^2$ with $\begin{pmatrix} x & y & 1 \end{pmatrix}^T \in \mathbb{R}^3$. Hence, the projection map is given by

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix}.$$

Let $c(t) \in \mathbb{R}^2$ be a point of the conic section. Then real numbers w_0, w_1, $w_2 \in \mathbb{R}$ and points b_0, b_1, $b_2 \in \mathbb{R}^2$ exist such that the equation

$$c(t) = \frac{w_0 b_0 B_0^2(t) + w_1 b_1 B_1^2(t) + w_2 b_2 B_2^2(t)}{w_0 B_0^2(t) + w_1 B_1^2(t) + w_2 B_2^2(t)}$$

holds. The polygon with the vertices $b_i \in \mathbb{R}^2$ is the control polygon of the conic section c. The real numbers w_i are the weights of the corresponding vertices of the control polygon. In the case that all weights are equal, c is a nonrational quadratic polynomial, a parabola.

This idea can be generalized. A **rational Bézier curve** $x(t)$ of degree n in $\mathbb{R}^3$ is the projection of a n-th degree Bézier curve in $\mathbb{R}^4$ into the hyperplane $w = 1$, let $\begin{pmatrix} x & y & z & w \end{pmatrix}^T \in \mathbb{R}^4$. In complete analogy to the conic section case we find

$$x(t) = \frac{w_0 b_0 B_0^n(t) + \ldots + w_n b_n B_n^n(t)}{w_0 B_0^n(t) + \ldots + w_n B_n^n(t)}.$$

Again, $w_i \in \mathbb{R}$ are weights, the points $b_i \in \mathbb{R}^3$ form the control polygon. For $w_0 = \ldots = w_n$ the curve x becomes a nonrational Bézier curve.

To evaluate a rational Bézier curve we could apply the de Casteljau algorithm first to the numerator, second to the denominator and then calculate the quotient. If the absolute values of the weights are of varying size the method is not numerically stable. Another possibility is to apply the projection into the hyperplane $w = 1$ in every step of the de Casteljau algorithm. This leads to the **rational de Casteljau algorithm**:

$$b_i^r(t) = (1-t)\frac{w_i^{r-1}}{w_i^r}b_i^{r-1} + t\frac{w_{i+1}^{r-1}}{w_i^r}b_{i+1}^{r-1} \quad \text{with} \quad w_i^r(t) = (1-t)w_i^{r-1}(t) + tw_{i+1}^{r-1}(t).$$

An explcit formula is

$$b_i^r(t) = \frac{\sum_{j=0}^r w_{i+j}b_{i+j}B_j^r(t)}{\sum_{j=0}^r w_{i+j}B_j^r(t)}.$$

If all weights w_i are positive the intermediate points b_i^r are all located inside the convex hull of the original points b_i. Therefore the algorithm is numerically stable.

Even in the rational case it is possible to derive differentiability criteria. Given two rational Bézier curves, one with control polygon $b_0, \ldots, b_n$, weights $w_0, \ldots, w_n$ on the interval $[u_0, u_1]$, the other with control polygon $b_n, \ldots, b_{2n}$, weights $w_n, \ldots, w_{2n}$ on the interval $[u_0, u_1]$. Both curves form a C^1-curve if

$$\frac{w_{n-1}}{\Delta_0}\Delta b_{n-1} = \frac{w_{n+1}}{\Delta_1}\Delta b_n$$

holds. Obviously the weight w_n has no impact on the differentiability. We obtain a C^r-curve if

$$\frac{\Delta^r(w_{n-r}b_{n-r})}{(\Delta_0)^r} = \frac{\Delta^r(w_nb_n)}{(\Delta_1)^r}$$

holds.

A **cubic rational B-spline curve** in $\mathbb{R}^3$ is the projection of a nonrational cubic B-spline curve in $\mathbb{R}^4$ into the hyperplane $w = 1$. The control polygon of the rational B-spline curve is given by the points $d_{-1}, \ldots, d_{L+1}$, to every point $d_i \in \mathbb{R}^3$ a weight $w_i \in \mathbb{R}$ is associated. A rational B-spline curve possesses a piecewise rational cubic Bézier representation. This one can be obtained by projecting the appropriate Bézier points in $\mathbb{R}^4$ into the hyperplane $w = 1$. This leads to

$$b_{3i-2} = \frac{w_{i-1}(1-\alpha_i)d_{i-1} + w_i\alpha_i d_i}{v_{3i-2}} \quad \text{and} \quad b_{3i-1} = \frac{w_{i-1}\beta_i d_{i-1} + w_i(1-\beta_i)d_i}{v_{3i-1}}$$

with points $b_j, d_k \in \mathbb{R}^3$, $\alpha_i = \frac{1}{\Delta}\Delta_{i-2}$ and $\beta_i = \frac{1}{\Delta}\Delta_i$. The weights of these Bézier points are given by

$$v_{3i-2} = w_{i-1}(1-\alpha_i) + w_i\alpha_i \quad \text{and} \quad v_{3i-1} = w_{i-1}\beta_i + w_i(1-\beta_i).$$

The connection points are

$$b_{3i} = \frac{\gamma_i v_{3i-1}b_{3i-1} + (1-\gamma_i)v_{3i+1}b_{3i+1}}{v_{3i}}$$

with

$$\gamma_i = \frac{\Delta_i}{\Delta_{i-1} + \Delta_i} \quad \text{and} \quad v_{3i} = \gamma_i v_{3i-1} + (1-\gamma_i)v_{3i+1}.$$

Here v_{3i} is the weight of the point b_{3i}.

Alternatively, the piecewise rational Bézier polygon is the result of the following procedure: Start with the control polygon $\begin{pmatrix} w_i d_i & w_i \end{pmatrix}^T \in \mathbb{R}^4$, derive its Bézier representation and divide it by the Bézier weights.

3 Streamline Diffusion Method

Standard finite element methods are not suitable to solve convection dominated convection diffusion problems because the numerical solution exhibits unphysical oscillations. This shows the need for a stabilization of the finite element method which can be done by modifying the method appropriately [KA03].

Let $\Omega \subset \mathbb{R}^d$ be a bounded Lipschitz domain. Consider the stationary problem

$$\begin{aligned} -\epsilon\Delta u + c \cdot \nabla u + ru &= f \qquad \text{in } \Omega \\ u &= 0 \qquad \text{on } \partial\Omega \end{aligned} \tag{8}$$

with $\epsilon > 0$, $f : \Omega \to \mathbb{R}$, $c : \Omega \to \mathbb{R}^d$ and $r : \Omega \to \mathbb{R}$ sufficiently smooth. We assume that for some $r_0 > 0$ the inequality $r - \frac{1}{2}\nabla \cdot c \geq r_0$ holds. We define the forms

$$\begin{aligned} a(u, v) &:= \epsilon(\nabla u, \nabla v) + (c \cdot \nabla u, v) + (ru, v) \\ \langle f, v\rangle_0 &:= (f, v) \end{aligned}$$

with $(u, v) := \int_\Omega u \cdot v \, dx$ to transform (8) into the weak form. Let V_h be the discrete finite element space and $K \in \mathcal{T}_h$ an element of the triangulation $\mathcal{T}_h$ of the domain Ω. Now we define for $v_h \in V_h$ the modified forms

$$\begin{aligned} a_h(u, v_h) &:= a(u, v_h) + \sum_{K\in\mathcal{T}_h} \delta_K(-\epsilon\Delta u + c \cdot \nabla u + ru, \tau(v_h))_K \\ \langle f, v_h\rangle_h &:= \langle f, v_h\rangle_0 + \sum_{K\in\mathcal{T}_h} \delta_K(f, \tau(v_h))_K \end{aligned}$$

with $(u, v)_K = \int_K u \cdot v \, dx$, a map $\tau : V_h \to L^2(\Omega)$ and local stabilizing parameters $\delta_K \in \mathbb{R}$. We consider the problem: Find $u_h \in V_h$ such that

$$a_h(u_h, v_h) = \langle f, v_h\rangle_h \quad \forall v_h \in V_h \, . \tag{9}$$

For the choice $\tau(v_h) = c \cdot \nabla v_h$ problem (9) is called **streamline diffusion method** (sdFEM). The Galerkin/least squares finite element method uses $\tau(v_h) := -\epsilon\Delta v_h + c \cdot \nabla v_h + rv_h$. Another option is the "bubble stabilized" sdFEM which is described in [MS] for one spatial dimension. In this method further weights that are bubble functions are added.

Let

$$||v||_\epsilon := \left(\epsilon|v|_1^2 + ||v||_0^2\right)^{\frac{1}{2}}$$

be the ϵ-weighted H^1-norm. For a standard finite element method the error estimate

$$||u - u_h||_\epsilon \leq Ch|u|_2$$

holds [KA03]. For the sdFEM we define the sd norm

$$||v||_{\text{sd}} := \left(\epsilon|v|_1^2 + r_0||v||_0^2 + \sum_{k\in\mathcal{T}_h} \delta_k||c \cdot \nabla v||_{0,K}^2\right)^{\frac{1}{2}} .$$

This leads in the case of linear continuous elements to the error estimate

$$||u - u_h||_{\text{sd}} \leq Ch^{\frac{3}{2}}|u|_2.$$

Hence, the asymptotic behaviour of the sdFEM is better. Note that the sd norm is stronger than the ϵ-norm because the estimate

$$\min\{1, \sqrt{r_0}\}||v||_\epsilon \leq ||v||_{\text{sd}} \quad \forall v \in H_0^1(\Omega)$$

holds. The sdFEM is easy to implement. If a standard finite element code is already available only a few lines have to be added.

One major disadvantage is the need to choose the stabilizing parameter empirically. As a consequence it is difficult to automate the sdFEM. Furthermore no inverse monotonicity can be proven.

It is also possible to include sdFEM in a *hp*-method [MS] and [SS96]. Here the solution is decomposed like in a multi scale approach. More details on sdFEM can be found in [Wah91].

4 One-dimensional Model Problem

In this section the ideas of isogeometric analysis [HCB05] are applied to the one-dimensional model problem

$$\begin{aligned} -\epsilon(x)u'' + c(x)u' + r(x)u &= f \qquad \text{in } \Omega \\ u &= g \qquad \text{on } \partial\Omega \end{aligned} \tag{10}$$

with $\Omega = (0,1)$ and $g \in L^2(\partial\Omega)$. For $0 < \epsilon \ll 1$ and $|c| \approx 1$ the problem (10) is a convection dominated convection-diffusion problem. Therefore the resulting finite element method has to be stabilized. We apply the streamline diffusion method discussed in Section 3.

To find weak solutions $u \in C^1(\Omega)$ problem (10) is transformed into a variational problem. Given $w \in C^1(\Omega)$ with $w(0) = g(0)$ and $w(1) = g(1)$, i.e. $w|_{\partial\Omega} = g$. We introduce the spaces

$$\begin{aligned} V &:= \{v \in C^1(\Omega) \ : \quad v|_{\partial\Omega} \equiv 0\} \\ \tilde{V} &:= \{v \in C^1(\Omega) \ : \quad v|_{\partial\Omega} \equiv g\} = \{v \in C^1(\Omega) \ : \quad v - w \in V\}. \end{aligned}$$

For $u \in C^2(\Omega)$ und $v \in C^1(\Omega)$ the equations (10) yield

$$-\int_0^1 \epsilon(x)u''v \, dx + \int_0^1 c(x)u'v \, dx + \int_0^1 r(x)uv \, dx = \int_0^1 fv \, dx.$$

Partial integration eliminates the second derivative. For $u \in \tilde{V}$ and $v \in V$ it follows

$$\int_0^1 \epsilon(x)u'v' \, dx + \int_0^1 c(x)u'v \, dx + \int_0^1 r(x)uv \, dx = \int_0^1 fv \, dx.$$

With

$$\begin{aligned} a(u,v) &:= \int_0^1 u' \left(\epsilon(x)v' + c(x)v\right) \, dx + \int_0^1 r(x)uv \, dx \\ b(v) &:= \int_0^1 fv \, dx \end{aligned}$$

we derive from (10): Find $\tilde{u} \in \tilde{V}$

$$a(\tilde{u}, v) = b(v) \quad \forall v \in V.$$

Unfortunately, $\tilde{V}$ is not a vector space. With the setting $\tilde{u} := u + w$ the equation

$$a(u,v) = a(\tilde{u} - w, v) = a(\tilde{u}, v) - a(w, v) = b(v) - a(w, v) \quad \forall v \in V$$

is satisfied. Thus we arrive at the following variational formulation of (10): Find $u \in V$ such that

$$a(u,v) = \tilde{b}(v) \quad \forall v \in V \tag{11}$$

with $\tilde{b}(v) := b(v) - a(w,v)$. Formulation (11) serves as the basis for the finite element method we use.

For the choice

$$\epsilon(x) = 10^{-4}, \quad c(x) = 1, \quad r(x) = 0, \quad f(x) = 0, \quad g(0) = 0, \quad g(1) = 1, \tag{12}$$

the exact solution u of (10) is given by

$$u(x) = \frac{1 - \exp(x/\epsilon)}{1 - \exp(1/\epsilon)} \tag{13}$$

[KA03].

4.1 Solution of the Model Problem

4.1.1 Knot Sequence

In [HCB05] a knot sequence is defined slightly different compared to [Far97] and this paper. We have to add the left and right knot one time to be consistent with [HCB05]. Additionally, the numbering is different.

We choose L and n (cf. Section 2.7) and a knot sequence

$$[x_0 \quad x_1 \quad \dots \quad x_{L+2n-2}]$$

with domain knots $x_{n-1}, \dots, x_{L+n-1}$. Suppose that the boundary knots x_0 and x_{L+2n-2} are of multiplicity n, these knots are called open knots. We assume that $x_0 = \dots = x_{n-1} = 0$ and $x_{L+n-1} = \dots = x_{L+2n-2} = 1$. If the domain Ω cannot be represented by a one dimensional knot sequence then a d-dimensional knot sequence as described in [BBdVC+06] can be chosen. Note that the restriction to the interval $[0, 1]$ is not mandatory.

4.1.2 Recursive Construction of the B-splines

Given the knot sequence

$$[x_0 \quad x_1 \quad x_2 \quad x_3 \quad x_4 \quad x_5] = [0 \quad 0 \quad \frac{3}{10} \quad \frac{4}{5} \quad 1 \quad 1],$$

i.e. $n = 2$ and $L = 3$. The multiplicity of the knots 0 and 1 is n. As a consequence every B-spline curve will interpolate the points associated to the knots 0 and 1. The domain knots are $[x_1 \quad x_2 \quad x_3 \quad x_4]$. There exist $n + L = 5$ basis functions $N_i^n = N_i^2$ for $i = 0, \dots, n + L - 1$. According to (5), the B-splines are calculated recursively. We define the B-splines of degree zero:

$$N_i^0(x) = \begin{cases} 1 & \text{for } x_{i-1} \le x < x_i \\ 0 & \text{else} \end{cases}.$$

Due to the multiplicity of the knots 0 and 1 we find $N_0^0 = N_1^0 = N_5^0 = N_6^0 = 0$. In Figure 1 the results of the recursion process are illustrated, e.g.

$$N_2^2(x) = \begin{cases} \frac{25}{6}x^2 & \text{for } 0 \le x < \frac{3}{10} \\ \frac{1}{10}(4 - 5x)^2 & \text{for } \frac{3}{10} \le x < \frac{4}{5} \\ \frac{50}{7}(x - 1)^2 & \text{for } \frac{4}{5} \le x < 1 \end{cases}.$$

Obviously the set of B-splines forms a partition of unity. Every B-spline has a local support (cf. Section 2.7), i.e.

$$N_i^n(x) \neq 0 \quad \text{only for } x \in [x_{i-1}, x_{i+n}]. \tag{14}$$

That is the reason why B-splines can be used easily in a finite element method. B-spline curves are C^{n-r} at domain knots with multiplicity $r \ge 1$.

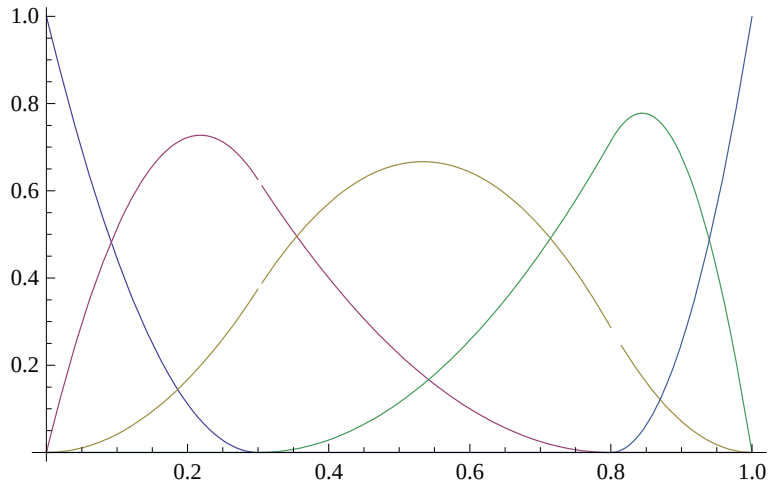

Figure 1: Quadratic B-splines for the knot sequence $[0 \quad 0 \quad \frac{3}{10} \quad \frac{4}{5} \quad 1 \quad 1]$

4.1.3 Representation of the Domain and Basis Functions

By the help of rational B-spline curves it is possible to model various complex geometries in $\mathbb{R}^d$ with only a few control points. This is done by means of an invertible push forward operator F that maps the parametric domain $[0,1]^d$ into the physical space $\Omega \subset \mathbb{R}^d$. The parametric domain consists of the d-dimensional knot sequences. Given a set of basis functions $N_i^{n,d}$ in $[0,1]^d$. We construct via $N_i^{n,d} \circ F^{-1}$ a set of basis functions in Ω.

4.1.4 Application to the Model Problem

Given a knot sequence on $\Omega = (0,1)$ with open boundary knots. In this case the map F introduced in 4.1.3 is nothing but the identity. We want to find solutions of the model problem (10) in the finite dimensional space

$$S_h := \{v(x) = \sum_{i=0}^{L+n-1} m_i N_i^n(x), \quad m_i \in \mathbb{R}, \quad v|_{\partial\Omega} = 0\}.$$

We see $m_0 = m_{L+n-1} = 0$. To simplify the discretization process we assume that the coefficient functions $\epsilon(x)$, $c(x)$ und $r(x)$ are constant in every finite element. Hence, the variational formulation of the discrete problem reads as follows: Find $u_h = \sum_{i=1}^{L+n-2} m_i N_i^n(x) \in S_h$ with

$$a(u_h, v) = \tilde{b}(v) \quad \text{for } v \in S_h.$$

We establish the stiffness matrix A and the right-hand side b. The variational problem is solved by the system $Am = b$ with $m = \begin{pmatrix} m_1 & m_2 & \dots & m_{n+L-2} \end{pmatrix}^T$. The entries of A are

$$(A)_{ji} = a(N_i^n, N_j^n) = \int_\Omega \left[\epsilon(x)(N_i^n)'\,(N_j^n)' + c(x)(N_i^n)'\,(N_j^n) + r(x)(N_i^n)\,(N_j^n)\right] dx$$

for $i, j = 1, \dots, L+n-2$. The integrals over the domain Ω are interpreted as integrals over $[0,1]^d$ using the transformation formula. The derivatives of the B-splines can be calculated by means of the recursion formula (6). Even so it is more efficient to use the polynomial representation of the B-splines. The stiffness matrix is assembled from local element matrices $A_{\text{loc},k}$. Due to (14) only

the basis functions $N^n_{k+1-n}, \ldots, N^n_{k+1}$ are non-zero in the element $E_k = [u_k, u_{k+1}]$. That shows

$$A_{\mathrm{loc},k} = \begin{pmatrix} a_{\mathrm{loc},k}(N^n_{k+1-n}, N^n_{k+1-n}) & \ldots & a_{\mathrm{loc},k}(N^n_{k+1-n}, N^n_{k+1}) \\ \vdots & \ddots & \vdots \\ a_{\mathrm{loc},k}(N^n_{k+1}, N^n_{k+1-n}) & \ldots & a_{\mathrm{loc},k}(N^n_{k+1}, N^n_{k+1}) \end{pmatrix}$$

with

$$a_{\mathrm{loc},k}(u, v) = \int_{E_k} \epsilon u'v' + cu'v + ruv \; dx.$$

As the basis functions are piecewise polynomials all values can be determined exactly. For the right-hand side b we apply the same ideas. Because of the boundary condition $g(0) = 0$ and $g(1) = 1$ we choose $w = N^n_{L+n-1}$. The j-th equation of the system is

$$a(u_h, N^n_j) = \sum_{i=1}^{L+n-2} d_i a(N^n_i, N^n_j) = \int_\Omega f(x)\, N^n_j(x)\; dx - a(N^n_{l+n-1}, N^n_j) = \tilde{b}(N^n_j).$$

Therefore

$$(b)_j = \int_\Omega f(x)\, N^n_j(x)\; dx \; - a(N^n_{l+n-1}, N^n_j) = \sum_{k=j-1}^{j+n} \int_{E_k} f(x)\, N^n_j(x)\; dx \; - a(N^n_{l+n-1}, N^n_j)$$

for $j = 1, \ldots, L+n-2$. In Figure 2 we see how the local element matrix is inserted into the global stiffness matrix.

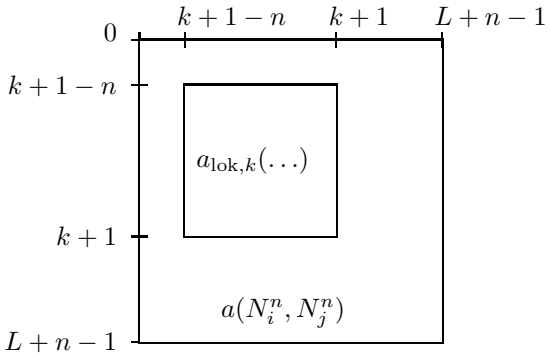

Figure 2: Inserting the element matrix into the stiffness matrix

4.2 Matlab Solution

We use Matlab and its Spline Toolbox [dB08] to find a numerical solution. Given $n, L \in \mathbb{N}$. Assume that all domain knots are simple. Without considering the index shift that is necessary in Matlab we use:

- Knot sequence $[x_0 \quad x_1 \quad \ldots \quad x_{L+2n-2}]$.
- Domain knots $x_{n-1},\ x_n,\ \ldots,\ x_{L+n-1}$.
- B-splines N^n_i, $i = 0, \ldots, L+n-1$, i.e. $L+n$ basis functions of polynomial degree n.
- Elements $E_k = [x_k, x_{k+1}]$, $k = n-1, \ldots, L+n-2$. There are L intervals.
- B-Splines with non-vanishing support on E_k are $N^n_{k+1-n}, \ldots, N^n_{k+1}$, i.e. $n+1$ functions. Thus the element matrix is a $(n+1) \times (n+1)$ matrix.

- The stiffness matrix is formally a $(L+n)\times(L+n)$ matrix. Due to the boundary condition first and last row as well as first and last column can be deleted.

In Matlab we have to be aware of some index shifts:

- In the knot sequence we identify $i \to i+2$. In Matlab, vectors start with index 1. Moreover, the open left knot appears $n+1$ times in the knot sequence.
- The degree of the polynomial is n, whereas $n+1$ is the order of the polynomial.
- In the Spline Toolbox, `bspline(1)` describes N_0^n.

In the start sequence all knots appear only once, i.e. $[x_1,\dots,x_L]$. We derive the appropriate knot sequence in the Matlab notation $[\tilde{x}_1,\dots,\tilde{x}_{L+2n+1}]$. Note, that the open knots appear $n+1$ times. There are two ways of describing the elements E_k, the B-form: $E_k=[\tilde{x}_{k+2},\tilde{x}_{(k+1)+2}]$ with $k=n-1,\dots,L+n-2$ and the pp-Form $E_k=[x_k,x_{k+1}]$ with $k=1,\dots,L$.

In Matlab, the polynomial representation of a B-spline curve always refers to the point of origin. The polynomial vector needed in the integration process can be calculated using a generalization of the Horner scheme, cf. [Ang]. Note that the right-hand side can only be integrated exactly if the function f is polynomial.

4.3 Choice of the Stabilization Parameter

We stabilize the solution by means of sdFEM, cf. Section 3. In [HCB05] no information on the appropriate choice of the stabilization parameter is given. It should depend on the polynomial degree and the mesh width. For polynomial basis functions [MS] proposes

$$\delta_k = \delta_1 \cdot h_i \cdot \frac{1}{2\sqrt{3}n^2}$$

with suitable δ_1. The length of the i-th interval is denoted by h_i. During our numerical experiments we realized that this approach is not applicable to the isogeometric analysis context. To solve model problem (10) we propose: Let $\delta_1 = 1.3$ and set

$$\delta_k = \delta_1 \cdot h \cdot \frac{1}{n} = \delta_1 \cdot \frac{1}{L} \cdot \frac{1}{n} \tag{15}$$

with the number of equidistant intervals L whose length is h.

In all numerical tests we solved (10) with the parameters (12). For comparative purposes the exact solution (13) appears in all figures. We consider $L=100$. In Figure 3 we plot the solution for $n=2$ and in Figure 4 for $n=5$. The unstabilized solution is always oscillatory. The stabilization terms remove these oscillations. Hence, formula (15) is suitable for a moderate polynomial degree.

A more detailed analysis reveals that (15) works reliably up to polynomial degree 16. The boundary layer is becoming smaller. Nevertheless, for $n=20$ any choice of the stabilization parameter leads to an oscillatory solution. Table 1 shows how the stabilization parameter δ_k depends on the polynomial degree n.

n	1	2	3	4	5	6	7	8
δ_k	0.013	0.0065	0.00433	0.00325	0.0026	0.002166	0.0018571	0.001625

n	9	10	12	16
δ_k	0.001444	0.0013	0.0010833	0.0008125

Table 1: Stabilization parameter δ_k according to (15) for $L=100$

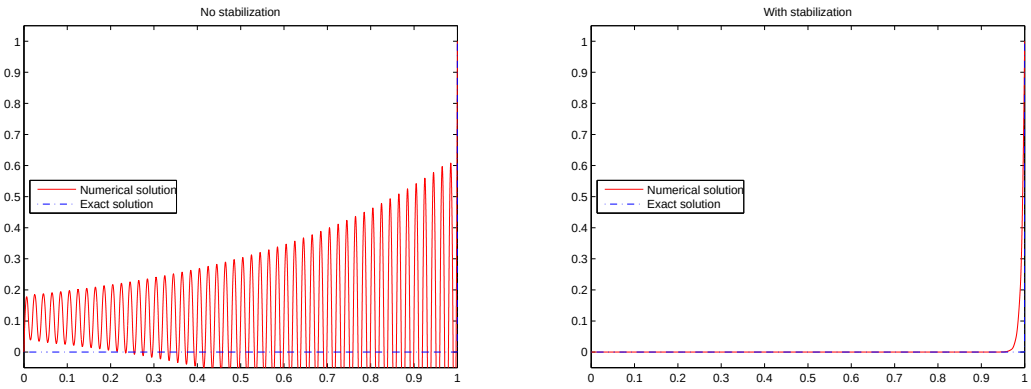

Figure 3: Numerical solution of (12) with $\epsilon = 10^{-4}$, $n = 2$ and $L = 100$

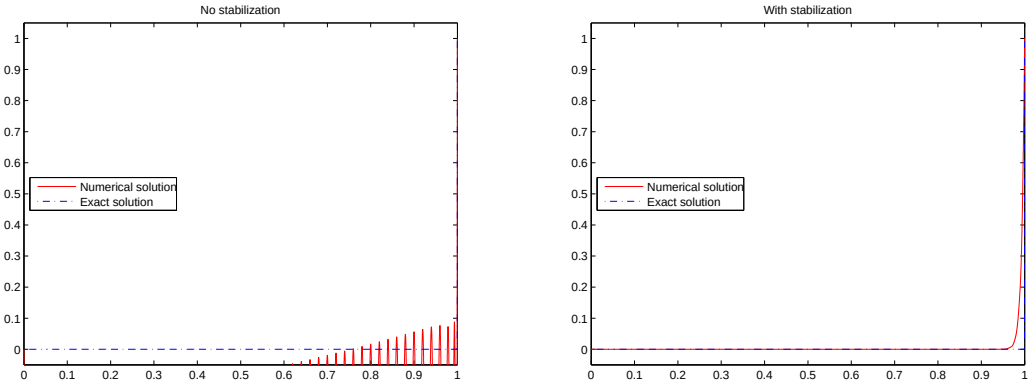

Figure 4: Numerical solution of (12) with $\epsilon = 10^{-4}$, $n = 5$ and $L = 100$

5 Two-dimensional Model Problem

5.1 The Model Problem

Let $\Omega = (0,1)^2 \subset \mathbb{R}^2$ be a bounded Lipschitz-domain. We consider the boundary value problem

$$-\nabla \cdot (K(x)\nabla u) + a(x) \cdot \nabla u + r(x)u = f \quad \text{for } x \in \Omega$$

with

$$K : \Omega \to \mathbb{R}^{2\times 2}, \qquad a : \Omega \to \mathbb{R}^2, \qquad r, f : \Omega \to \mathbb{R}.$$

To be more specific, let $\kappa > 0$ and $\theta \in [0, 2\pi]$:

$$\kappa(x) = \begin{pmatrix} \kappa & 0 \\ 0 & \kappa \end{pmatrix}, \qquad a(x) = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}, \qquad r \equiv 0 \qquad \text{und} \qquad f \equiv 0. \tag{16}$$

The problem simplifies to

$$\begin{aligned} -\kappa\Delta u + a \cdot \nabla u &= 0 \qquad \text{in } \Omega \\ u &= g \qquad \text{on } \partial\Omega \end{aligned} \tag{17}$$

with a suitable function $g \in L^2(\partial\Omega)$. For the model problem we choose, cf. [HCB05],

$$\begin{aligned} g(x) &= 0 \quad \text{for } x \in \quad \{1\} \times [0,1] \quad \cup \quad [0,1] \times \{1\} \quad \cup \quad \{0\} \times (0.2, 1]\,, \\ g(x) &= 1 \quad \text{for } x \in \quad [0,1) \times \{0\} \quad \cup \quad \{0\} \times [0, 0.2]\,. \end{aligned} \tag{18}$$

To find a solution of the problem described by (16), (17) and (18) isogeometric analysis [HCB05] is used. First, we derive the weak formulation of (17). Let $w \in H^1(\Omega)$ such that $w|_{\partial\Omega} = g$ and

$$\begin{aligned} V &= \{v \in H^1(\Omega) \ : \ v|_{\partial\Omega} \equiv 0\}, \\ \tilde{V} &= \{v \in H^1(\Omega) \ : \ v|_{\partial\Omega} \equiv g\} = \{v \in H^1(\Omega) \ : \ v - w \in V\}. \end{aligned}$$

Partial integration

$$\int_\Omega \kappa\nabla u_1 \cdot \nabla u_2 \, dx = -\int_\Omega (\kappa\Delta u_1) u_2 \, dx + \int_{\partial\Omega} \kappa \, (\nabla u_1 \cdot \nu) \, u_2 \, d\sigma$$

with $u_1 \in H^2(\Omega)$ and $u_2 \in H^1(\Omega)$ yields for $v \in H^1(\Omega)$, $u \in H^2(\Omega)$ and $f \equiv 0$

$$\begin{aligned} &-\int_\Omega (\kappa\Delta u)\, v \, dx + \int_\Omega (a \cdot \nabla u)\, v \, dx = \int_\Omega f v \, dx \\ \Leftrightarrow &\int_\Omega \kappa\nabla u \cdot \nabla v \, dx - \int_{\partial\Omega} \kappa(\nabla u \cdot \nu) v \, d\sigma + \int_\Omega (a \cdot \nabla u) v \, dx = \int_\Omega f v \, dx\,. \end{aligned}$$

The weak formulation of (17) is: Find $\tilde{u} \in \tilde{V}$ with

$$a(\tilde{u}, v) = b(v) \quad \text{for all } v \in V.$$

As in Section 4 we transform the formulation to be able to find a solution in a vector space. Let $\tilde{u} := u + w$ with $w \in \tilde{V}$, $w|_{\partial\Omega} \equiv g$ and $u \in V$. This leads to

$$a(u, v) = a(\tilde{u} - w, v) = a(\tilde{u}, v) - a(w, v) = b(v) - a(w, v) \quad \text{for all } v \in V.$$

With $\tilde{b}(v) := b(v) - a(w, v)$ the second weak formulation reads as: Find $u \in V$ with

$$a(u, v) = \tilde{b}(v) \quad \text{for all } v \in V. \tag{19}$$

The assumptions (16) ensure that the assumptions of the Lax-Milgram lemma are satisfied.

5.2 Discretization of the Model Problem

Considering the two-dimensional domain Ω as tensor product $\Omega = \Omega_1 \times \Omega_2$ with $\Omega_1 = \Omega_2 = (0,1)$ allows us to apply the same methods as in Section 4. The space of basis functions is the tensor product of one-dimensional B-splines. Given the polynomial degrees of the ansatz functions n_1 on the x-axis resp. n_2 on the y-axis and the number of intervals in case of simple knots L_1 on the x-axis resp. L_2 on the y-axis. We consider the knot sequences

$$[x_0^{(1)} \quad x_1^{(1)} \quad \dots \quad x_{L_1+2n_1-2}^{(1)}]$$

on the x-axis with $x_0^{(1)} = 0$, $x_{L_1+2n_1-2}^{(1)} = 1$ and

$$[x_0^{(2)} \quad x_1^{(2)} \quad \dots \quad x_{L_2+2n_2-2}^{(2)}]$$

on the y-axis with $x_0^{(2)} = 0$, $x_{L_2+2n_2-2}^{(2)} = 1$. Note that the polynomial degrees in the two spatial dimensions may be different. The end knots $x_0^{(i)}$ and $x_{L_i+2n_i-2}^{(i)}$ are supposed to be of multiplicity n_i for $i \in \{1,2\}$. The x-axis resp. y-axis is denominated $\Omega_1 = (0,1)$ resp. $\Omega_2 = (0,1)$. Let $N_i^{n_1}$ with $i = 0, \dots, L_1 + n_1 - 1$ be the $L_1 + n_1$ B-splines on Ω_1 and $M_j^{n_2}$ with $j = 0, \dots, L_2 + n_2 - 1$ the $L_2 + n_2$ B-splines on Ω_2. A basis function $N_{ij}^{n_1 n_2}$ on Ω is the tensor product

$$N_{ij}^{n_1 n_2}(x) := N_i^{n_1}(x_1) \cdot M_j^{n_2}(x_2), \quad x = (x_1, x_2)^T,\ x_1 \in \Omega_1,\ x_2 \in \Omega_2$$

with $i = 0, \dots, L_1 + n_1 - 1$ and $j = 0, \dots, L_2 + n_2 - 1$. Hence, on Ω we have $(L_1 + n_1) \cdot (L_2 + n_2)$ basis functions. The discrete space S_h of the ansatz functions is

$$S_h := \left\{ v(x) = \sum_{i=0}^{L_1+n_1-1} \sum_{j=0}^{L_2+n_2-1} m_{ij} N_i^{n_1}(x_1) M_j^{n_2}(x_2) \right\}.$$

A function $v \in S_h$ is uniquely described by a matrix $M = (m_{ij}) \in \mathrm{Mat}(L_1 + n_1, L_2 + n_2, \mathbb{R})$, so $\dim S_h = (L_1 + n_1) \cdot (L_2 + n_2)$. The discrete space S_h^0 of such functions from S_h that vanish on the boundary of Ω is

$$S_h^0 := \left\{ v(x) = \sum_{i=1}^{L_1+n_1-2} \sum_{j=1}^{L_2+n_2-2} m_{ij} N_i^{n_1}(x_1) M_j^{n_2}(x_2) \right\}.$$

We suppose that the function $w \in S_h$ fulfils approximately the boundary condition (17). Now we can formulate the discrete version of (19): Find $u \in S_h^0$ with

$$a(u, v) = \tilde{b}(v) = -a(w, v) \quad \text{for all } v \in S_h^0.$$

We calculate the stiffness matrix A and the right side b. In a first step we compute $\tilde{A}$ and $\tilde{b}$ without taking the boundary conditions into consideration. We renumber the functions in S_h, i.e.

$$S_h = \left\{ v(x) = \sum_{s=1}^{(L_1+n_1)(L_2+n_2)} m_s N_s^{n_1 n_2} \right\}.$$

So we project the rectangle $[0, L_1+n_1-1] \times [0, L_2+n_2-1]$ onto the interval $[1, (L_1+n_1)\cdot(L_2+n_2)]$ via

$$(i, j) \mapsto s := i + j(L_1 + n_1) + 1\ .$$

Table 2 gives an example for $n_1 + L_1 = 5$ and $n_2 + L_2 = 4$. We identify $N_s^{n_1 n_2} = N_{ij}^{n_1 n_2}$ and $m_s = m_{ij}$. The entries of the stiffness matrix $\tilde{A}$ are

$$\tilde{A}_{rt} = a(N_t^{n_1 n_2}, N_r^{n_1 n_2}) \quad \text{for } r, t = 1, \dots, (L_1 + n_1)(L_2 + n_2),$$

			i $\rightarrow$				
			0	1	2	3	4
j	$\downarrow$	0	1	2	3	4	5
		1	6	7	8	9	10
		2	11	12	13	14	15
		3	16	17	18	19	20

Table 2: Projection of a rectangle onto the straight line

with the numbers r and t calculated by means of the projection map introduced. To find A we delete from the matrix $\tilde{A}$ rows and columns that result from basis functions that are included in S_h but not in S_h^0. The same procedure applies to the vector b. We first calculate $\tilde{b}$, a vector of length $(L_1 + n_1)(L_2 + n_2)$:

$$\tilde{b}_r = -a(w, N_r^{n_1 n_2}).$$

Then we derive b by erasing some entries of $\tilde{b}$. Due to the structure of the function w we can easily compute the vector $\tilde{b}$ by copying appropriate entries of the matrix $\tilde{A}$.

5.3 Matlab Solution

5.3.1 Domain and Finite Elements

The representation of the domain $\Omega = (0, 1)^2$ as a tensor product of one-dimensional B-splines is trivial. The entries of the global stiffness matrix are computed element by element. We consider the element $E_{kl} = [x_k^{(1)}, x_{k+1}^{(1)}] \times [x_l^{(2)}, x_{l+1}^{(2)}]$. Here k and l are indices of the knot sequences of the x and y spatial dimensions with

$$k \in \{(n_1 - 1), \ldots, (L_1 + n_1 - 2)\} \quad \text{and} \quad l \in \{(n_2 - 1), \ldots, (L_2 + n_2 - 2)\}.$$

An example of the domain decomposition is visualized in Figure 5. In the two-dimensional case

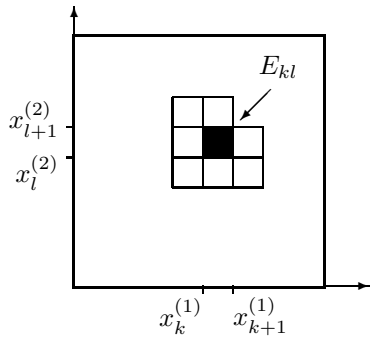

Figure 5: Decomposition of Ω using two knot sequences

we do not provide local element matrices. We insert the element contribution directly into the global stiffness matrix.

5.3.2 The Stiffness Matrix

The B-splines $N_{k+1-n_1}^{n_1}, \ldots, N_{k+1}^{n_1}$ are non-vanishing on the interval $[x_k^{(1)}, x_{k+1}^{(1)}]$. On the interval $[x_l^{(2)}, x_{l+1}^{(2)}]$ the B-splines $M_{l+1-n_2}^{n_2}, \ldots, M_{l+1}^{n_2}$ are non-vanishing. Hence, on the element E_{kl} the

basis functions

$$N_{ij}^{n_1 n_2} \text{ with } \begin{cases} i = k+1-n_1, \ldots, k+1 \\ j = l+1-n_2, \ldots, l+1 \end{cases}$$

are non-vanishing, at all these are $(n_1+1)(n_2+1)$ functions.

Given the two functions

$$u = N_{i_1 i_2}^{n_1 n_2} = N_{i_1}^{n_1} M_{i_2}^{n_2} \quad \text{and} \quad v = M_{i_1 i_2}^{n_1 n_2} = N_{j_1}^{n_1} M_{j_2}^{n_2}.$$

We compute $a(v|_{E_{kl}}, u|_{E_{kl}})$. Therefore, we need

$$\nabla u = \nabla N_{i_1 i_2}^{n_1 n_2} = \begin{pmatrix} \left(N_{i_1}^{n_1}(x_1)\right)' M_{i_2}^{n_2}(x_2) \\ N_{i_1}^{n_1}(x_1) \left(M_{i_2}^{n_2}(x_2)\right)' \end{pmatrix} \quad \text{and} \quad \nabla v = \nabla N_{j_1 j_2}^{n_1 n_2} = \begin{pmatrix} \left(N_{j_1}^{n_1}(x_1)\right)' M_{j_2}^{n_2}(x_2) \\ N_{j_1}^{n_1}(x_1) \left(M_{j_2}^{n_2}(x_2)\right)' \end{pmatrix}.$$

Now, we derive

$$\begin{aligned} a(v|_{E_{kl}}, u|_{E_{kl}}) &= \int_{E_{kl}} \kappa \left(\nabla v \cdot \nabla u\right) \, dx + \int_{E_{kl}} (a \cdot \nabla v)\, u \, dx \\ &= \left(\int_{x_k^{(1)}}^{x_{k+1}^{(1)}} N_{i_1}^{n_1}(x_1) N_{j_1}^{n_1}(x_1) \, dx_1 \right) \left(\int_{x_l^{(2)}}^{x_{l+1}^{(2)}} \left(M_{j_2}^{n_2}(x_2)\right)' \left(\kappa \left(M_{i_2}^{n_2}(x_2)\right)' + (\sin\theta) M_{i_2}^{n_2}(x_2) \right) \, dx_2 \right) \\ &\quad + \left(\int_{x_l^{(2)}}^{x_{l+1}^{(2)}} M_{j_2}^{n_2}(x_2) M_{i_2}^{n_2}(x_2) \, dx_2 \right) \left(\int_{x_k^{(1)}}^{x_{k+1}^{(1)}} \left(N_{j_1}^{n_1}(x_1)\right)' \left(\kappa \left(N_{i_1}^{n_1}(x_1)\right)' + (\cos\theta) N_{i_1}^{n_1}(x_1) \right) \, dx_1 \right). \end{aligned}$$

After computing the local value we need to apply the projection described in Section 5.2. To do so we reconstruct from the local two-dimensional numbering the global two-dimensional numbering and project it onto the one-dimensional numbering.

5.3.3 Right-hand Side and Further Steps

From the stiffness matrix $\tilde{A}$ we can extract the values of the vector $\tilde{b}$. We represent the boundary condition by a function $w \in S_h$ with

$$w = \sum \sum a_{i_1 i_2} N_{i_1 i_2}^{n_1 n_2} = \sum \psi_s N_s^{n_1 n_2}. \tag{20}$$

The coefficients $a_{i_1 i_2}$ ensure that w fulfils the boundary condition (18) which is discontinuous. Hence, this condition cannot be represented exactly as a linear combination of B-splines, see figure 6.

The entries of the vector $\tilde{b} \in \mathbb{R}^{(L_1+n_1)(L_2+n_2)}$ are nothing but $-a(w, v)$ for all basis functions $v = N_{j_1 j_2}^{n_1 n_2} \in S_h$. We compute $\tilde{b}$ by multiplying the stiffness matrix $\tilde{A}$ with a vector $-\Psi$ where $(\Psi)_s = \psi_s$. We also use Ψ to identify all rows and columns of $\tilde{A}$ that have to be erased from the matrix. Likewise, the vector $\tilde{b}$ is treated. Now the linear system can be solved numerically.

5.4 Stabilization and Results

The streamline diffusion method, cf. Section 3, adds local stabilization terms to the stiffness matrix. We give an explicit representation of the term

$$\delta_k \langle -\kappa \Delta v + a \cdot \nabla v + r v, a \cdot \nabla u \rangle_{E_{kl}}$$

with the stabilzation parameter δ_k. Given the element $E_{kl} = [x_k^{(1)}, x_{k+1}^{(1)}] \times [x_l^{(2)}, x_{l+1}^{(2)}]$ and the two functions

$$u = N_{i_1 i_2}^{n_1 n_2} = N_{i_1}^{n_1} M_{i_2}^{n_2} \quad \text{und} \quad v = N_{j_1 j_2}^{n_1 n_2} = N_{j_1}^{n_1} M_{j_2}^{n_2}.$$

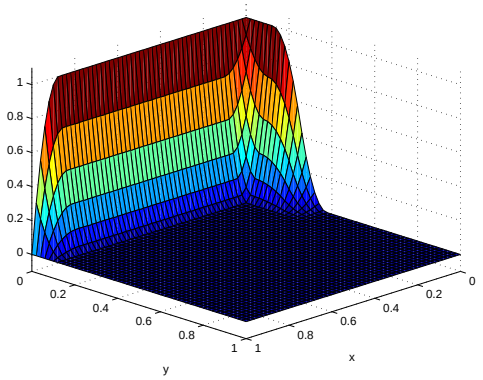

Figure 6: Boundary condition for $n_1 = n_2 = 8$ and $L_1 = L_2 = 8$

We find

$$\Delta v = \frac{\partial^2}{\partial x^2} v + \frac{\partial^2}{\partial y^2} v = \left(N_{j_1}^{n_1}(x_1)\right)'' M_{j_2}^{n_2}(x_2) + N_{j_1}^{n_1}(x_1) \left(M_{j_2}^{n_2}(x_2)\right)'',$$
$$\Delta u = \frac{\partial^2}{\partial x^2} u + \frac{\partial^2}{\partial y^2} u = \left(N_{i_1}^{n_1}(x_1)\right)'' M_{i_2}^{n_2}(x_2) + N_{i_1}^{n_1}(x_1) \left(M_{i_2}^{n_2}(x_2)\right)''.$$

For

$$\langle -\kappa \Delta v, a \cdot \nabla u \rangle_{E_{kl}} = r_1 + r_2$$

we have

$$\begin{aligned} r_1 =& - \int_{E_{kl}} \kappa \left(N_{j_1}^{n_1}(x_1)\right)'' M_{j_2}^{n_2}(x_2) \left((\cos\theta) \left(N_{i_1}^{n_1}(x_1)\right)' M_{i_2}^{n_2}(x_2) + (\sin\theta) N_{i_1}^{n_1}(x_1) \left(M_{i_2}^{n_2}(x_2)\right)' \right) dx \\ =& - \kappa \cos\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} \left(N_{j_1}^{n_1}(x_1)\right)'' \left(N_{i_1}^{n_1}(x_1)\right)' dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} M_{j_2}^{n_2}(x_2) M_{i_2}^{n_2}(x_2)\, dx_2 \\ & - \kappa \sin\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} \left(N_{j_1}^{n_1}(x_1)\right)'' N_{i_1}^{n_1}(x_1)\, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} M_{j_2}^{n_2}(x_2) \left(M_{i_2}^{n_2}(x_2)\right)' dx_2 \end{aligned}$$

and

$$\begin{aligned} r_2 =& - \int_{E_{kl}} \kappa N_{j_1}^{n_1}(x_1) \left(M_{j_2}^{n_2}(x_2)\right)'' \left((\cos\theta) \left(N_{i_1}^{n_1}(x_1)\right)' M_{i_2}^{n_2}(x_2) + (\sin\theta) N_{i_1}^{n_1}(x_1) \left(M_{i_2}^{n_2}(x_2)\right)' \right) dx \\ =& - \kappa \cos\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} N_{j_1}^{n_1}(x_1) \left(N_{i_1}^{n_1}(x_1)\right)' dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} \left(M_{j_2}^{n_2}(x_2)\right)'' M_{i_2}^{n_2}(x_2)\, dx_2 \\ & - \kappa \sin\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} N_{j_1}^{n_1}(x_1) N_{i_1}^{n_1}(x_1)\, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} \left(M_{j_2}^{n_2}(x_2)\right)'' \left(M_{i_2}^{n_2}(x_2)\right)' dx_2 \,. \end{aligned}$$

For the second stabilization term

$$\langle a \cdot \nabla v, a \cdot \nabla u \rangle_{E_{kl}} = r_3$$

we find

$$
\begin{aligned}
r_3 = &\int_{E_{kl}} \left((\cos\theta) \left(N_{j_1}^{n_1}(x_1)\right)' M_{j_2}^{n_2}(x_2) + (\sin\theta) N_{j_1}^{n_1}(x_1) \left(M_{j_2}^{n_2}(x_2)\right)' \right) \cdot \ldots \\
&\ldots \cdot \left((\cos\theta) \left(N_{i_1}^{n_1}(x_1)\right)' M_{i_2}^{n_2}(x_2) + (\sin\theta) N_{i_1}^{n_1}(x_1) \left(M_{i_2}^{n_2}(x_2)\right)' \right) \, dx \\
=&(\cos\theta)^2 \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} \left(N_{j_1}^{n_1}(x_1)\right)' \left(N_{i_1}^{n_1}(x_1)\right)' \, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} M_{j_2}^{n_2}(x_2) M_{i_2}^{n_2}(x_2) \, dx_2 \\
&+ \sin\theta\cos\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} \left(N_{j_1}^{n_1}(x_1)\right)' N_{i_1}^{n_1}(x_1) \, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} M_{j_2}^{n_2}(x_2) \left(M_{i_2}^{n_2}(x_2)\right)' \, dx_2 \\
&+ \sin\theta\cos\theta \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} N_{j_1}^{n_1}(x_1) \left(N_{i_1}^{n_1}(x_1)\right)' \, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} \left(M_{j_2}^{n_2}(x_2)\right)' M_{i_2}^{n_2}(x_2) \, dx_2 \\
&+ (\sin\theta)^2 \int_{x_k^{(1)}}^{x_{k+1}^{(1)}} N_{j_1}^{n_1}(x_1) N_{i_1}^{n_1}(x_1) \, dx_1 \int_{x_l^{(2)}}^{x_{l+1}^{(2)}} \left(M_{j_2}^{n_2}(x_2)\right)' \left(M_{i_2}^{n_2}(x_2)\right)' \, dx_2 \, .
\end{aligned}
$$

For the streamline diffusion method the article [HCB05] proposes to choose

$$\delta_k = \frac{h_a}{2|a|} \quad \text{with} \quad h_a = \frac{h}{\max\{\cos\theta, \sin\theta\}}.$$

Here h is the mesh width, θ the angle which appears in the convection term a. If we choose uniform spacing in both spatial dimensions with $L_1 = L_2 = 20$ intervals we find $h = \frac{1}{L_1} = \frac{1}{L_2} = \frac{1}{20}$. In the cases $\theta = \frac{\pi}{4}$ resp. $\theta = \arctan(2)$ the stabilization terms are

$$\delta_k = \frac{1}{20 \cdot \sin\left(\frac{\pi}{4}\right) \cdot 2} = \frac{1}{20 \cdot \sqrt{2}} \approx 0.035355 \quad \text{resp.} \quad \delta_k = \frac{1}{20 \cdot \sin(\arctan(2)) \cdot 2} \approx 0.02795.$$

In Figures 7 to 16 numerical solutions of the model problem from [HCB05] are visualized. For comparison we always give a 21×21 plot with linear interpolation, too. Table 3 shows the computing times for different dimensions n of the B-splines.

n	Running time [seconds]	
	Test 1	Test 2
1	21.33	20.95
2	112.15	100.65
3	313.04	313.03
4	794.15	729.38
8	7830.86	7921.55

Table 3: Running time on Intel(R) Pentium(R) 4 CPU 3.06GHz with 768 MB RAM

6 Model Problem with Variable Convection

In this section the more complex model problem from [Ang95] with variable convection is solved using isogeometric analysis. Given the domain $\Omega = (0,1)^2 \subset \mathbb{R}^2$. The problem is

$$-\nabla \cdot (K(x)\nabla u) + a(x) \cdot \nabla u(x) + r(x)u = f \quad \text{for } x \in \Omega$$

with

$$K : \Omega \to \mathbb{R}^{2\times 2}, \qquad a : \Omega \to \mathbb{R}^2, \qquad r, f : \Omega \to \mathbb{R}.$$

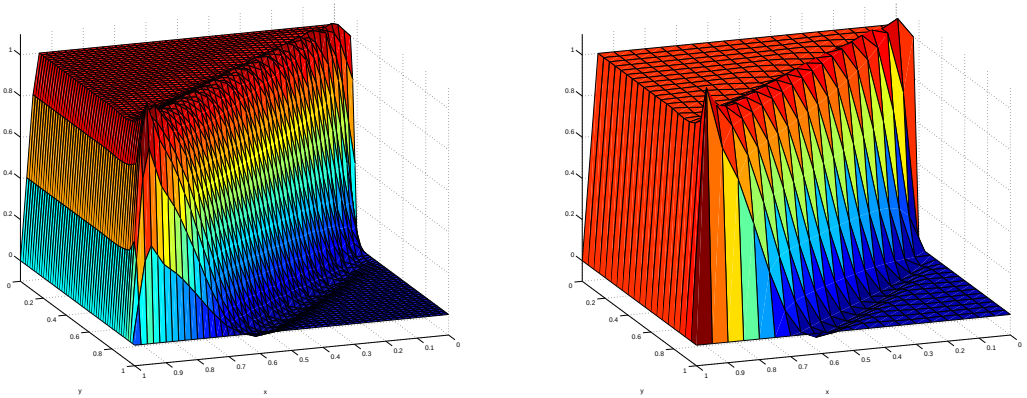

Figure 7: $\theta = \frac{\pi}{4}$, $n = 1$

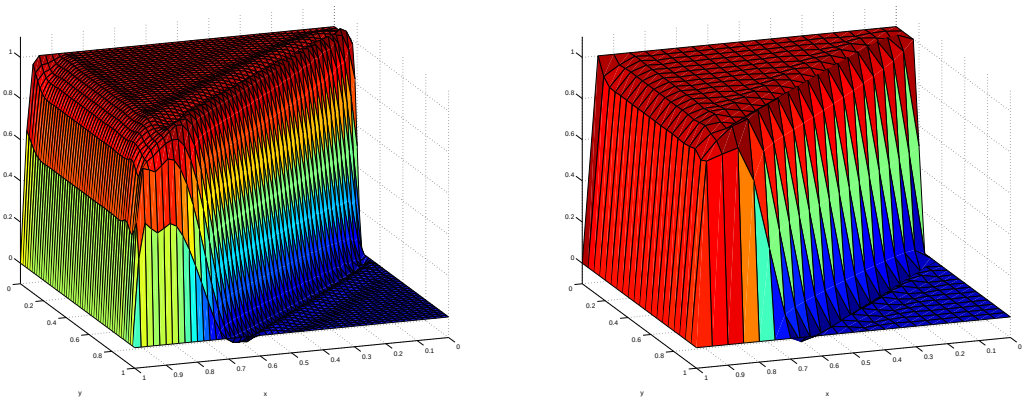

Figure 8: $\theta = \frac{\pi}{4}$, $n = 2$

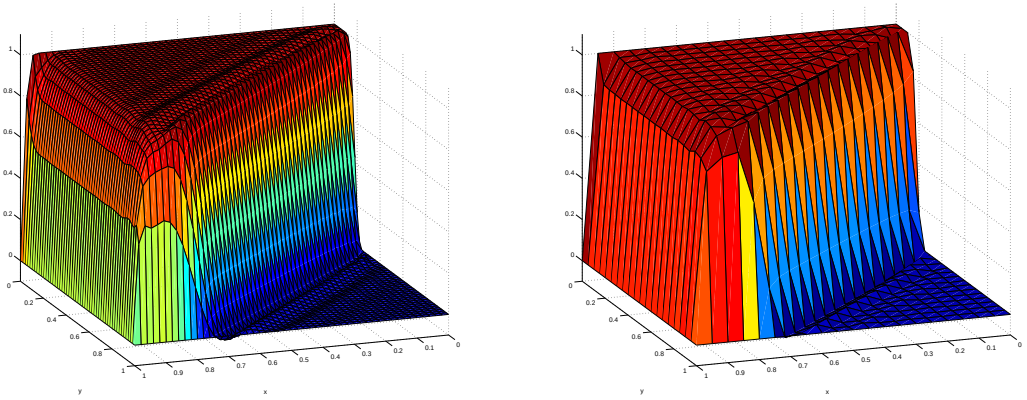

Figure 9: $\theta = \frac{\pi}{4}$, $n = 3$

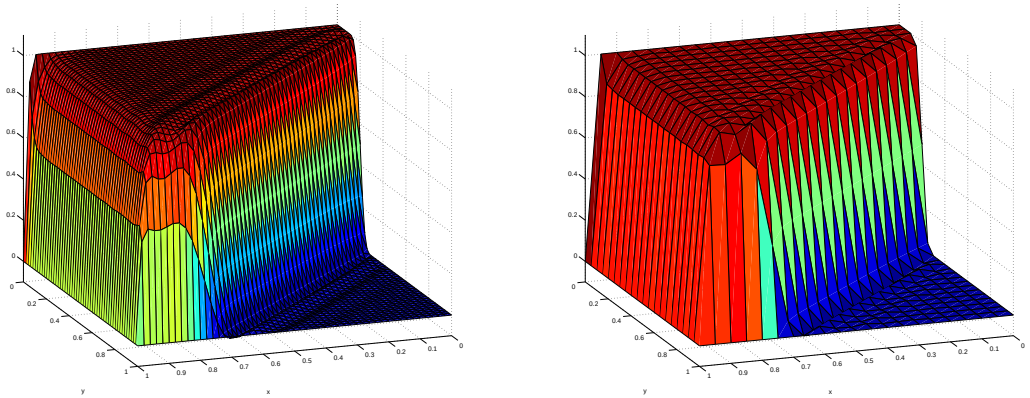

Figure 10: $\theta = \frac{\pi}{4}$, $n = 4$

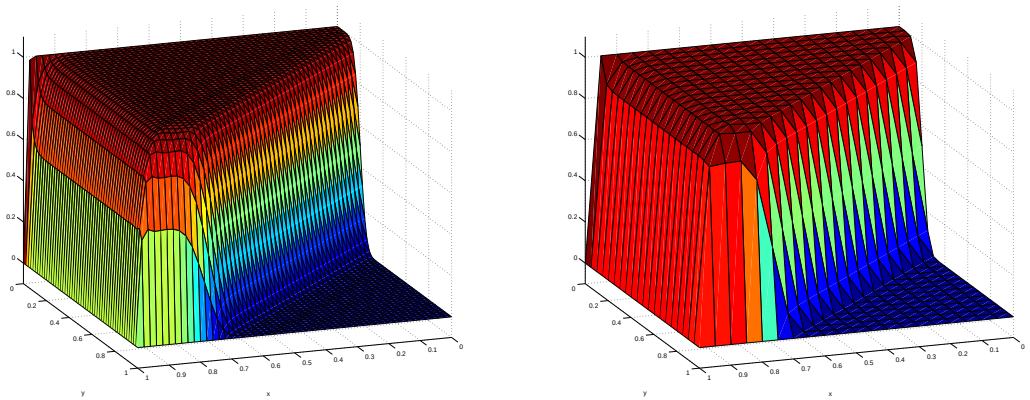

Figure 11: $\theta = \frac{\pi}{4}$, $n = 8$

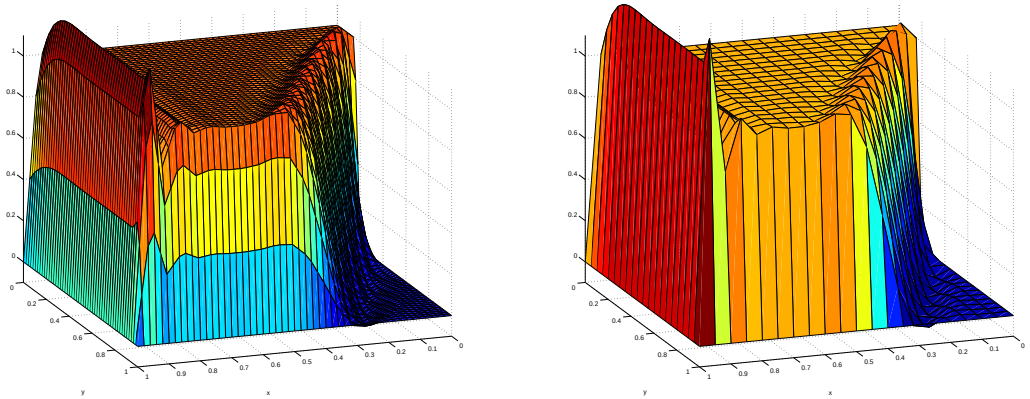

Figure 12: $\theta = \arctan 2$, $n = 1$

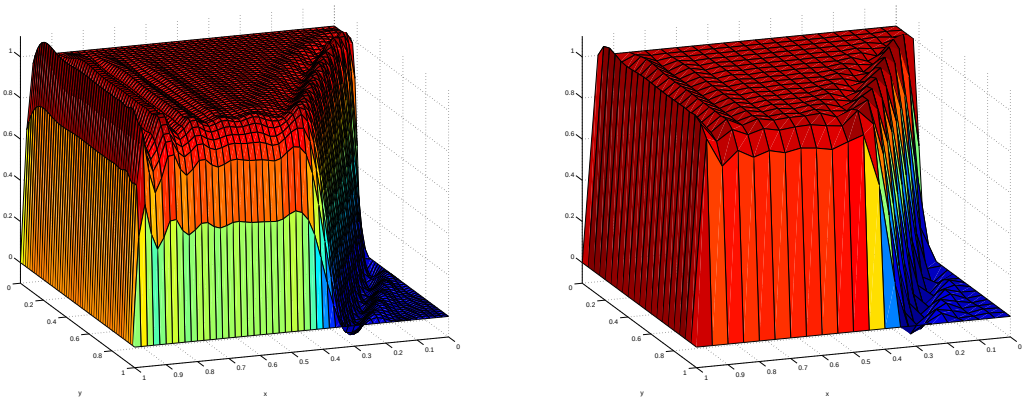

Figure 13: $\theta = \arctan 2$, $n = 2$

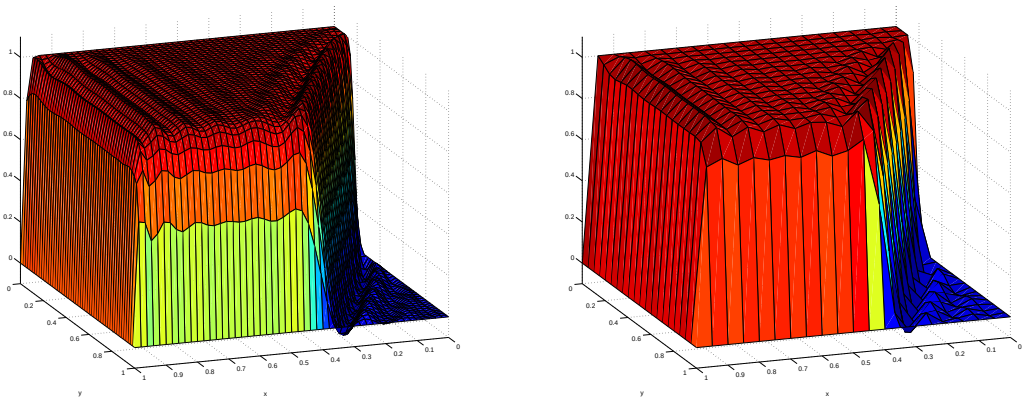

Figure 14: $\theta = \arctan 2$, $n = 3$

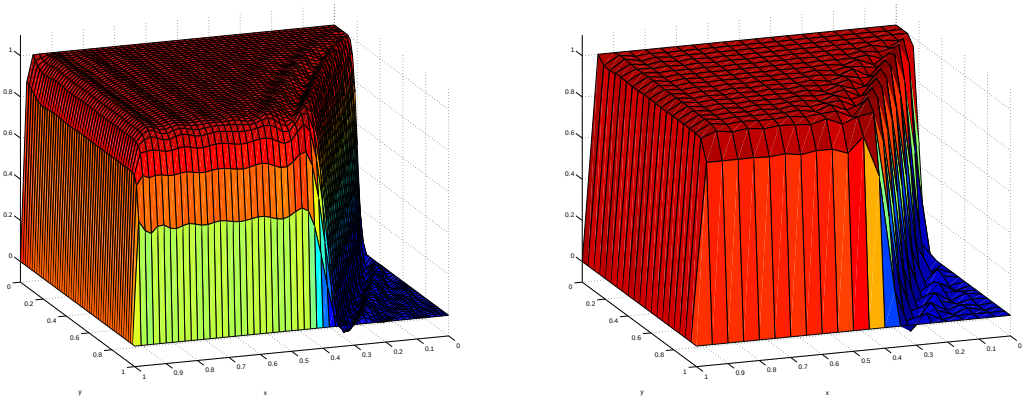

Figure 15: $\theta = \arctan 2$, $n = 4$

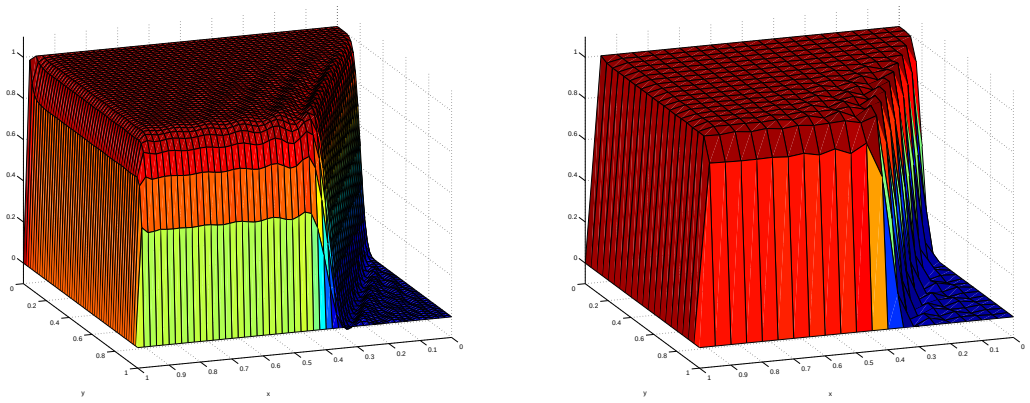

Figure 16: $\theta = \arctan 2$, $n = 8$

For $\kappa > 0$ and $x = (x_1, x_2)^T \in \mathbb{R}^2$ we define

$$\kappa(x) = \begin{pmatrix} \kappa & 0 \\ 0 & \kappa \end{pmatrix}, \qquad a(x) = \begin{pmatrix} c_1 x_1 + c_2 x_2 + c_3 \\ d_1 x_1 + d_2 x_2 + d_3 \end{pmatrix}, \qquad r \equiv 0 \qquad \text{and} \qquad f \equiv 0.$$

In the model we set

$$c_1 = 9.975, \quad c_2 = -19.22, \quad c_3 = 9.61, \quad d_1 = 6.41, \quad d_2 = -9.975, \quad d_3 = 4.9875.$$

The boundary conditions are given by the function $g(x)$ with

$$g(x_1, x_2) = 1 \quad \text{for } x_1 = 0, \quad x_2 \in [\frac{1}{4}, \frac{9}{20}] \quad \text{and} \quad g(x_1, x_2) = 0 \quad \text{otherwise.}$$

The solution needs the same steps as in Section 5. Only two adaptions are necessary. Of course, the boundary condition has to be changed. Furthermore, the integrals of the convection terms have to be re-computed.

In Figures 17 to 24 numerical solutions are visualized. The stabilization parameter is only roughly estimated. In this paper it is only revealed whether isogeometric analysis is suitable to solve such problems. In the examples is $n_1 = n_2$ and $L_1 = L_2$. It is obvious that the boundary condition blurs when using higher dimensional B-splines.

In Figure 17 we show a test computation without stabilization with $\kappa = 10^{-2}$. All the other tests include the stabilization terms. In Figures 18 and 19 the cases $\kappa = 10^{-3}$ and $\kappa = 10^{-6}$ can be compared.

In the subsequent tests we choose $n_1 = 1$, $L_1 = 20$, $\kappa = 10^{-6}$ and different stabilization parameters $\delta_k = 0.02$ (Figure 19), $\delta_k = 0.01$ (Figure 20), $\delta_k = 0.005$ (Figure 21).

For $n_1 = 2$ we used $L_1 = 50$ intervals, we visualize the two cases $\delta_k = 0.02$ (Figure 22) and $\delta_k = 0.01$ (Figure 23).

Concluding, we treat a high-dimensional example $n_1 = 8$ in Figure 24.

7 Concluding Remarks

Isogeometric analysis can be used to solve convection dominated convection-diffusion problems. The numerical solution of the problem presented in Section 5 is of good quality. On the other hand, the more challenging problem in Section 6 shows the limits of the method.

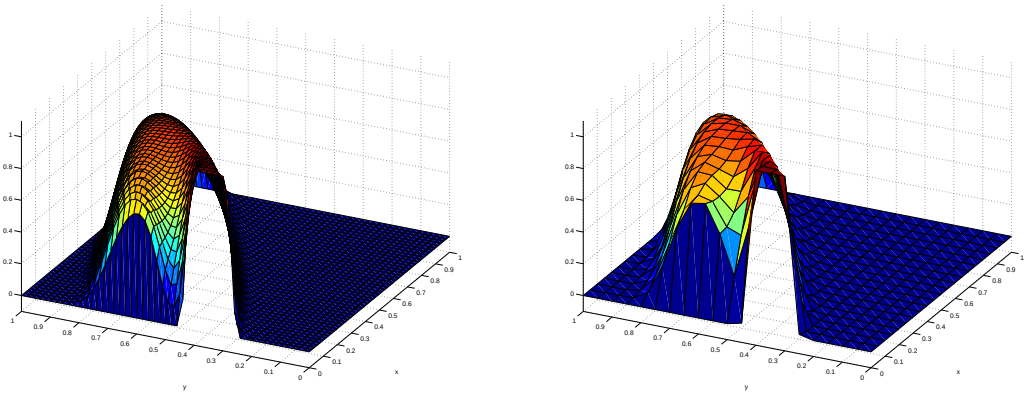

Figure 17: $n_1 = n_2 = 3$, $L_1 = L_2 = 50$, $\kappa = 10^{-2}$, no stabilization

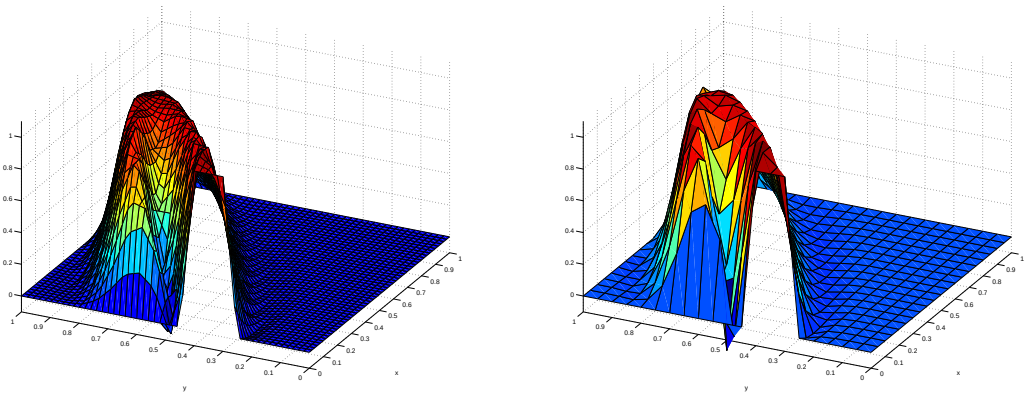

Figure 18: $n_1 = n_2 = 1$, $L_1 = L_2 = 20$, $\kappa = 10^{-3}$, $\delta_k = 0.02$

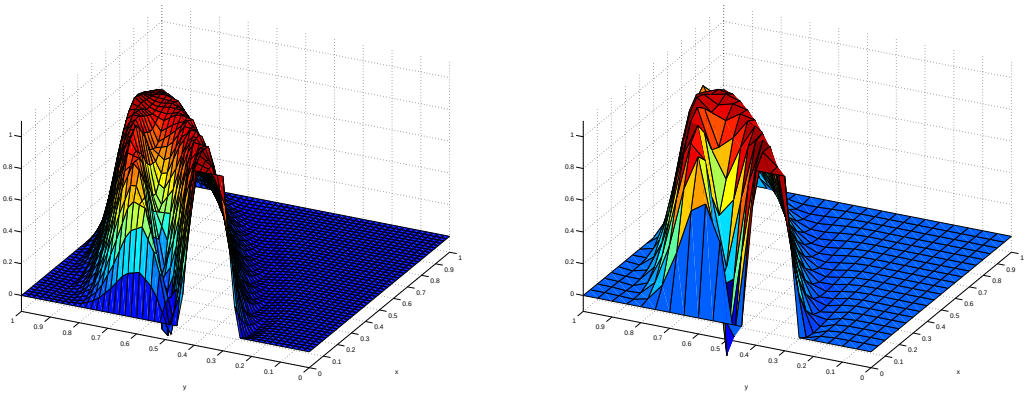

Figure 19: $n_1 = n_2 = 1$, $L_1 = L_2 = 20$, $\kappa = 10^{-6}$, $\delta_k = 0.02$

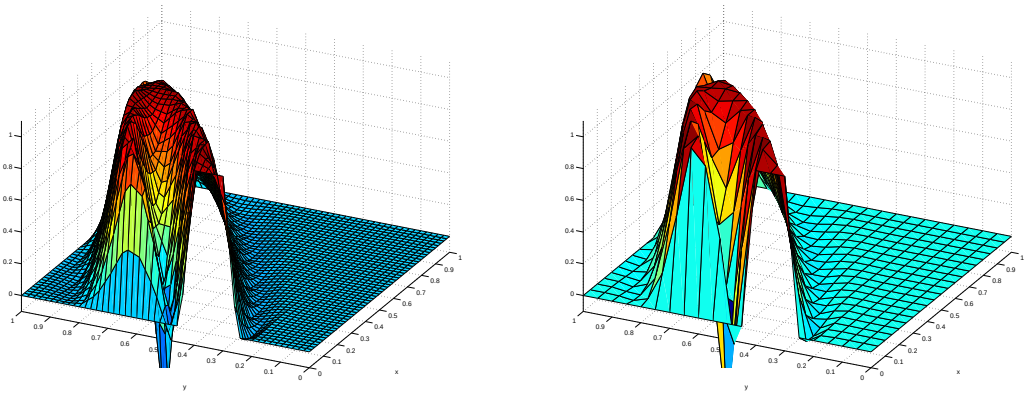

Figure 20: $n_1 = n_2 = 1$, $L_1 = L_2 = 20$, $\kappa = 10^{-6}$, $\delta_k = 0.01$

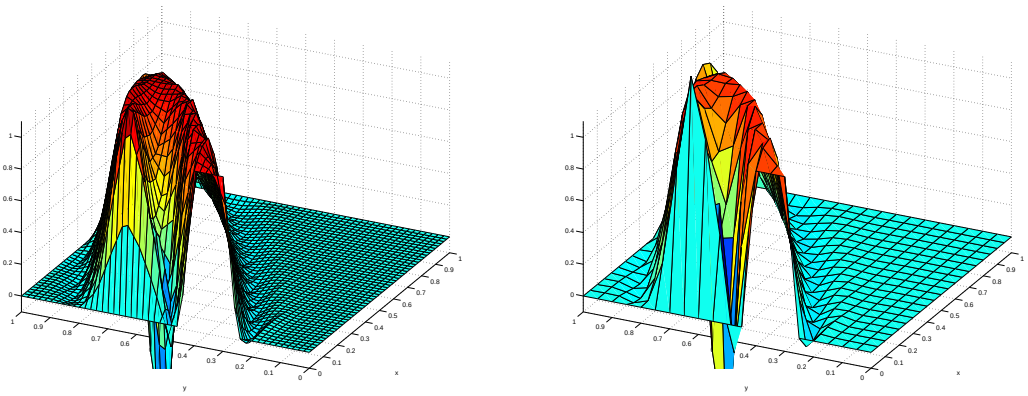

Figure 21: $n_1 = n_2 = 1$, $L_1 = L_2 = 20$, $\kappa = 10^{-6}$, $\delta_k = 0.005$

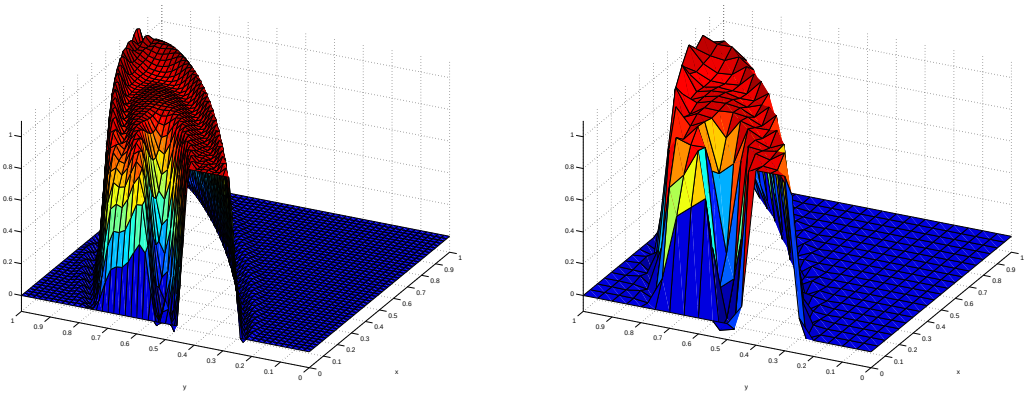

Figure 22: $n_1 = n_2 = 2$, $L_1 = L_2 = 50$, $\kappa = 10^{-6}$, $\delta_k = 0.02$

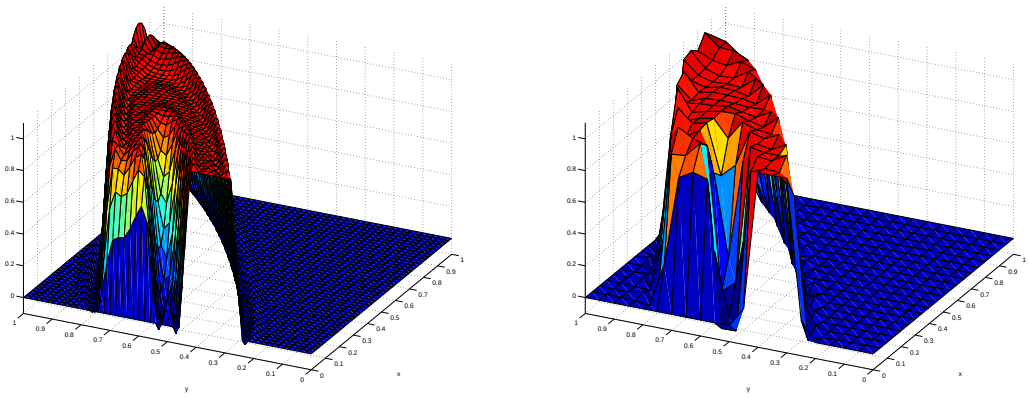

Figure 23: $n_1 = n_2 = 2$, $L_1 = L_2 = 50$, $\kappa = 10^{-6}$, $\delta_k = 0.01$

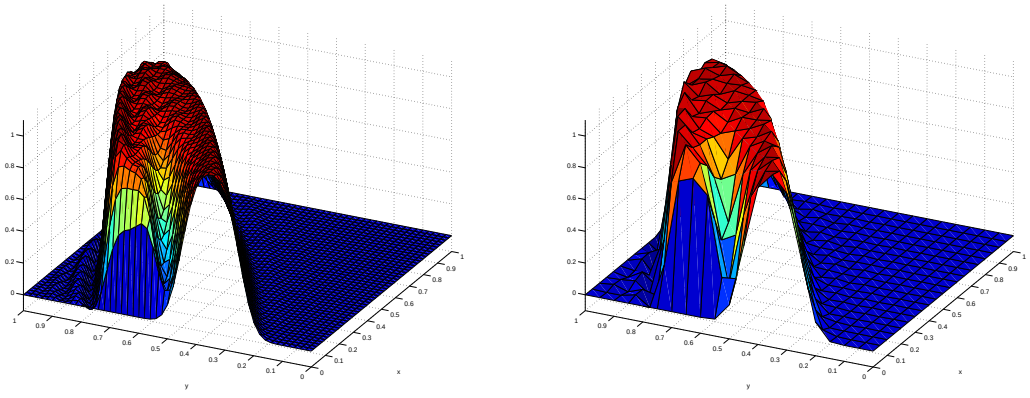

Figure 24: $n_1 = n_2 = 8$, $L_1 = L_2 = 20$, $\kappa = 10^{-6}$, $\delta_k = 0.01$

References

[Ang] Lutz Angermann. Numerische Mathematik I, Vorlesungsskriptum, Institut für Mathematik, Technische Universität Clausthal, 2005.

[Ang95] Lutz Angermann. Error estimates for the finite-element solution of an elliptic singularly perturbed problem. *IMA J. Numer. Anal.*, 15(2):161–196, 1995.

[BBdVC+06] Y. Bazilevs, L. Beirão da Veiga, J. A. Cottrell, T. J. R. Hughes, and G. Sangalli. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Math. Models Methods Appl. Sci.*, 16(7):1031–1090, 2006.

[Che05] Zhangxin Chen. *Finite Element Methods and Their Applications*. Springer Verlag, Berlin, Heidelberg, New York, 2005.

[dB08] Carl de Boor. *Matlab Spline Toolbox (TM) 3 User's Guide*. The MathWorks, October 2008.

[Far97] Gerald Farin. *Curves and Surfaces for Computer-Aided Geometric Design: A Practical Guide*. Academic Press, San Diego, London, 4th edition, 1997.

[FHK02] Gerald Farin, Josef Hoschek, and Myung-Soo Kim. *Handbook of Computer Aided Geometric Design*. Elsevier Science, Amsterdam, 2002.

[HCB05] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Comput. Methods Appl. Mech. Engrg.*, 194(39-41):4135–4195, 2005.

[KA03] Peter Knabner and Lutz Angermann. *Numerical Methods for Elliptic and Parabolic Partial Differential Equations*. Springer Verlag, New York, 2003.

[MS] J. M. Melenk and C. Schwab. The *hp* streamline diffusion finite element method for convection dominated problems in one space dimension. Research Report No. 98 10 (October 1998), Seminar für Angewandte Mathematik, Eidgenössische Technische Hochschule, Zürich.

[SS96] Christoph Schwab and Manil Suri. The p and hp versions of the finite element method for problems with boundary layers. *Math. Comp.*, 65(216):1403–1429, 1996.

[Wah91] Lars B. Wahlbin. Local behavior in finite element methods. In *Handbook of numerical analysis, Vol. II*, Handb. Numer. Anal., II, pages 353–522. North-Holland, Amsterdam, 1991.