Isogeometric analysis using LR B-splines

Kjetil André Johannessen, Trond Kvamsdal, and Tor Dokken
Department of Mathematical Sciences
Norwegian University of Science and Technology, Trondheim, Norway
Department of Applied Mathematics, SINTEF ICT, Norway
e–mail: Kjetil.Johannessen@math.ntnu.no, Trond.Kvamsdal@math.ntnu.no, and Tor.Dokken@sintef.no

Abstract

The recently proposed locally refined B-splines, denoted LR B-splines, by Dokken *et al.* [6] may have the potential to be a framework for isogeometric analysis to enable future interoperable computer aided design and finite element analysis. In this paper, we propose local refinement strategies for adaptive isogeometric analysis using LR B-splines and investigate its performance by doing numerical tests on well known benchmark cases. The theory behind LR B-spline is not presented in full details, but the main conceptual ingredients are explained and illustrated by a number of examples.

1 Introduction

1.1 Background

Computer Aided Design (CAD) and Finite Element Analysis (FEA) are essential technologies in modern product development. However, the interoperability of these technologies is severely disturbed by inconsistencies in the mathematical approaches used. The main reason for inconsistencies is that the technologies evolved in different communities with the focus on improving disjoint stages in product development processes, and taking little heed on relations to other stages. Efficient feedback from analysis to CAD and refinement of the analysis model are essential for computer-based design optimization and virtual product development. The current lack of efficient interoperability of CAD and FEA makes refinement and adaptation of the analysis model cumbersome, slow and expensive.

The new paradigm of Isogeometric Analysis, which was introduced by Hughes *et al.* [11], demonstrates that much is to be gained with respect to efficiency, quality and accuracy in analysis by replacing traditional Finite Elements by volumetric NURBS elements.

NURBS are not flexible enough to be a common basis for future CAD and FEA merely due to some required properties in design and analysis such as locally refineable, accommodate extraordinary points, and trimless option. T-splines are a recently developed generalization of NURBS [3], [7], [20], they were introduced to cure the above geometric limitations and to generate local refinements in the mesh. In context of isogeometric analysis, a new sub-class of T-splines as analysis-suitable (AS) T-splines [19] have emerged, which is a significant step towards more versatility. Recently there has also been published works related to hierarchical refinement of splines introduced by Forsey and Bartels [8]; see [22], [21], [9], [4], [17], and [16].

We believe that the recently proposed locally refined LR B-splines by Dokken *et al.* [6] may have the potential to form an alternative framework for future interoperable CAD and FEA systems. The new approach directly operates on the spline spaces, and in this way a broad spectrum of piecewise spline functions may be obtained. LR B-splines consist of smooth, piecewise polynomial basis functions that constitute a partition of unity. Among other advanced features they may facilitate local h-refinement. Since this class of splines is rich and versatile, it may break new ground and seems to be attractive as foundation for integrating CAD and FEA on one computational platform.

Our long term vision is to create a radically new computational platform with powerful and versatile refinement and adaptation procedures based on the concept of LR B-splines. Downward compatibility to existing NURBS-based models and the synergy of CAD and FEA expertise in each development stage will be essential and, at the same time, promote the broad acceptance and dissemination in both academia and the software industry.

In any finite element analysis of real world problems, it is of great importance that the quality of the computed solution may be determined. However, the assessment of the quality of a computed solution

is challenging, both mathematically and computationally. Thus traditionally, the quality of the solution is assessed manually by the scientist or engineer doing the simulation, but this is unreliable. Numerical simulation of many industrial problems in civil, mechanical and naval industry often require large computational resources. It is therefore of utmost importance that computational resources are used as efficiently as possible to make new results readily available and to expand the realm of which processes may be simulated. We thus identify reliability and efficiency as two challenges in simulation based engineering.

These two challenges may be addressed by error estimation combined with adaptive refinements. A lot of research has been performed on error estimation and adaptive mesh refinement, see e.g. (Ainsworth and Oden, 2000 [1]). However, adaptive methods are not yet an industrial tool, partly because the need for a link to traditional CAD-system makes this difficult in industrial practice. Here, the use of an isogeometric analysis framework may facilitate more widespread adoption of this technology in industry, as adaptive mesh refinement does not require any further communication with the CAD system.

1.2 Aim and outline of the paper

The aim of this paper is to present local refinement strategies using LR B-splines and investigate its performance in adaptive isogeometric analysis by means of showing numerical results on well known benchmark examples.

The paper is organized as follows:

In Section 2, we stated the preliminaries definitions of *B-splines* and *meshes* to illustrate the local refinement of B-spline using knot insertion. Then the basic important ingredients to understand *LR B-splines* concept such as *LR-mesh*, *LR B-spline space*, and *meshline extension* are given. Our aim here is to fix the notations, for a detailed mathematical description related to LR B-splines we refer the reader to Dokken *et al.* [6].

In Section 3, we give a brief introduction to the finite element method and the need for adaptive refinement in real world problems. The main characteristics of isogeometric finite element methods using B-splines (or NURBS) and LR B-splines is presented. Further we describe a general approach, that suits LR B-splines, to perform local h-refinement in adaptive isogeometric finite element method.

Section 4 is devoted to illustrate the local refinement strategies using LR B-splines. A more general discussion on different options for local refinement is given. Then we presented three specific local refinement strategies which we shall investigate in the numerical examples section. At the last the conceptual similarities between adaptive refinement in classical FEM *versus* isogeometric methods using LR B-splines (for $p = 1$ and 2) are given.

Numerical experiments are performed in Section 5. The aim of this section is to illustrate the performance of the local refinement strategies of Section 4. In particular, we investigate whether adaptive refinement using LR B-splines achieves optimal convergence rate, in terms of better accuracy per degrees of freedom (dofs) compared to the uniform refinement case, for non-smooth elliptic problems. For the purpose we consider one synthetic case of refinement along the diagonal and elliptic PDEs with known solutions.

We end this paper by giving some conclusion upon our findings in Section 6.

(a) Initial mesh (b) Tensor product refinement (c) Truly local refinement

Figure 1: Lack of local refinement of tensor B-splines.

2 Spline theory

The problem with traditional B-splines and NURBS is that they are formulated as tensor products of univariate B-splines. This means that refinement in one of the univariate B-splines will cause the insertion of an entire new row or column of knots in the bivariate spline space. As an example of refinement around a local point is achieved which also refine the other area of mesh. This is illustrated in Figure 1, where we have recursively refined the lower right corner. Ideally we do not want to insert any knot in the upper right and lower left part of the mesh, but with B-splines and NURBS, this is unavoidable. Thus to achieve truly local refinement we need some new structure to the mesh which is not based on global tensor products. This is what T-splines, Hierarchical B-splines, and LR B-splines address. T-splines were first introduced by Sederberg *et al.* [20] and have, like NURBS, primarily been used in computer aided design (CAD). In recent years T-splines have, however, been introduced to isogeometric analysis [2, 7, 18, 19].

In the following subsections we will present the LR B-splines, first we establishing a vocabulary that contains several definitions in section 2.1 and then we discuss the algorithms in section 2.2 followed by some properties of LR splines in Section 2.3

2.1 LR-splines

We start the introduction by describing the local knot vectors. From elementary B-spline theory we know that a knot vector of size $n + p + 1$ will generate n linearly independent basis functions of degree p. Usually this knot vector is required to start and end with a knot of multiplicity $p + 1$, ensuring at least $p+1$ basis functions to be generated. If we ignore this restriction, it is clear that we can generate a single basis function using a knot vector of size $p+2$. The purpose of open knot vectors (knot vectors with knots of multiplicity $p+1$ at the start and end) is only to ensure interpolating end points which is advantageous in a number of ways, for instance to simplify the handling of Dirichlet boundary conditions. From the evaluation algorithms of B-splines, it follows that every single basis function will depend on not more than $p+2$ knots, each basis function using different knots. For instance, consider a set of quadratic basis function from the knot vector Ξ. We then have

$$\begin{aligned}
\Xi &= [0,0,0,1,2,3,3,4,4,4] \\
\Xi_1 &= [0,0,0,1 \qquad\qquad\quad] \\
\Xi_2 &= [\ \ 0,0,1,2 \qquad\quad] \\
\Xi_3 &= [\ \ \ \ 0,1,2,3 \qquad] \\
\Xi_4 &= [\ \ \ \ \ \ 1,2,3,3 \quad] \\
\Xi_5 &= [\ \ \ \ \ \ \ \ 2,3,3,4 \ ] \\
\Xi_6 &= [\ \ \ \ \ \ \ \ \ \ 3,3,4,4 \ ] \\
\Xi_7 &= [\ \ \ \ \ \ \ \ \ \ \ \ 3,4,4,4],
\end{aligned} \tag{1}$$

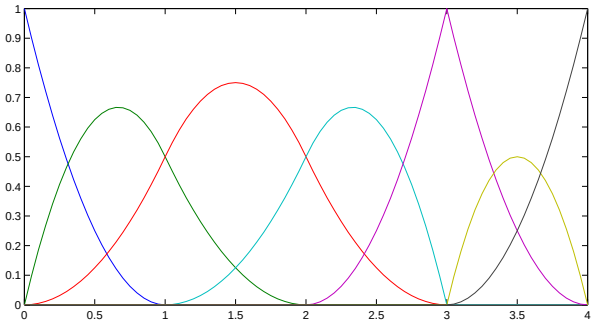


Figure 2: All quadratic basis functions generated by the knot $\Xi = [0, 0, 0, 1, 2, 3, 3, 4, 4, 4]$. Each individual basis function can be described using a local knot vector of 4 knots each $(p + 2)$.

where the seven basis functions will be separately generated by the *local knot vectors* $\Xi_1, ..., \Xi_7$. One might add here that we will not need the entire set of basis functions, and remove a subset of these, keeping only the ones we are interested in. Even though it might be instructive to look at local basis functions as a subsequence of a global knot vector, this is of little practical value. Instead we will not require any global knot vector Ξ, but rather create the local knot vectors Ξ_i in a different manner. The concept local knot vectors is important for LR B-splines as they are used as the building blocks. We have illustrated the basis functions given by Equation (1) in Figure 2. Using local knot vectors, we define a single B-spline function as

Definition 1. A **B-spline** $B(\boldsymbol{\xi})$ of degrees $\boldsymbol{p}$ is a separable function $B : \mathbb{R}^n \to \mathbb{R}$

$$B_{\boldsymbol{\Xi}}(\boldsymbol{\xi}) = \prod_{i=1}^{n} B_{\Xi^i}(\xi^i) \tag{2}$$

defined by the n nondecreasing local knot vectors $\Xi^i \in \mathbb{R}^{p_i+2}$ and the degrees p_i, where each $B_{\Xi^i}(\xi^i)$ are univariate B-spline functions of degree p_i over the knot vector Ξ^i.

Note that the degree is implicitly defined by the number of knots in each local knot vector.

Definition 2. The **parametric coordinate space** of dimension 1 2 and 3 is denoted using the greek letters ξ, η and ζ and is related in (2) as

$$(\xi^1, \xi^2, \xi^3) = (\xi, \eta, \zeta) \tag{3}$$

with the corresponding knot vectors begin denoted as $\Xi, \mathcal{H}, \mathcal{Z}$ such that

$$(\Xi^1, \Xi^2, \Xi^3) = (\Xi, \mathcal{H}, \mathcal{Z}). \tag{4}$$

For any B-spline in higher dimension than 3 it is custom to use index notation. The univariate, bivariate and trivariate cases are as following

$$\begin{array}{rcccccl} B_{\boldsymbol{\Xi}}(\xi^1) & = & B_{\boldsymbol{\Xi}}(\xi) & = & B_{\Xi^1}(\xi) & = & B_{\Xi}(\xi) \\ B_{\boldsymbol{\Xi}}(\xi^1, \xi^2) & = & B_{\boldsymbol{\Xi}}(\xi, \eta) & = & B_{\Xi^1}(\xi) B_{\Xi^2}(\eta) & = & B_{\Xi}(\xi) B_{\mathcal{H}}(\eta) \\ B_{\boldsymbol{\Xi}}(\xi^1, \xi^2, \xi^3) & = & B_{\boldsymbol{\Xi}}(\xi, \eta, \zeta) & = & B_{\Xi^1}(\xi) B_{\Xi^2}(\eta) B_{\Xi^3}(\zeta) & = & B_{\Xi}(\xi) B_{\mathcal{H}}(\eta) B_{\mathcal{Z}}(\zeta). \end{array}$$

We will in the remainder of the text regard bivariate B-splines unless otherwise stated and use the short hand notation

$$B[\xi_0 \xi_1 ... \xi_{p+1}; \eta_0 \eta_1 ... \eta_{p+1}] := B_{\Xi}(\xi) B_{\mathcal{H}}(\eta), \tag{5}$$

where the local knot vectors are known (integers), i.e. $B[0123; 00145]$ for $\Xi^1 = [0, 1, 2, 3], \Xi^2 = [0, 0, 1, 4, 5]$. This particular B-spline would be of polynomial degree $p_1 = 2$ and $p_2 = 3$ due to the number of elements in the local knot vectors.

Also note that we are distinguishing between subscripts and superscripts on the local knot vectors as the former refers to the index in a *set* of B-splines while the latter is the parametric dimension. Consider the set of biquadratic B-splines

$$\{B[0123;0012], B[2345;2245], B[1255;0112]\} = \{B_{\boldsymbol{\Xi}_1}, B_{\boldsymbol{\Xi}_2}, B_{\boldsymbol{\Xi}_3}\},$$

where

$$\begin{aligned} B_{\boldsymbol{\Xi}_1}(\xi^1,\xi^2) &= B_{\Xi_1^1}(\xi^1)B_{\Xi_1^2}(\xi^2) \\ B_{\boldsymbol{\Xi}_2}(\xi^1,\xi^2) &= B_{\Xi_2^1}(\xi^1)B_{\Xi_2^2}(\xi^2) \\ B_{\boldsymbol{\Xi}_3}(\xi^1,\xi^2) &= B_{\Xi_3^1}(\xi^1)B_{\Xi_3^2}(\xi^2) \end{aligned}$$

and

$$\begin{array}{llll} \Xi_1^1 = [0,1,2,3] & \Xi_1^2 = [0,0,1,2] \\ \Xi_2^1 = [2,3,4,5] & \Xi_2^2 = [2,2,4,5] \\ \Xi_3^1 = [1,2,5,5] & \Xi_3^2 = [0,1,1,2]. \end{array}$$

Definition 3. A **weighted B-spline** is defined as

$$B_{\boldsymbol{\Xi}}^{\gamma}(\boldsymbol{\xi}) = \gamma \prod_{i=1}^{n} B_{\Xi^i}(\xi^i),$$

where $\gamma \in (0,1]$.

The weighted B-spline is simply a B-spline multiplied by a scalar weight γ. This is to ensure that LR B-splines maintain the partition of unity property, and should not be confused with the rational weights w which is common in NURBS (non-uniform rational B-splines). For simplicity, we will denote both weighted and non-weighted B-splines as B and assume that it is clear from the context if it is one or the other.

Definition 4. A **Box Mesh** or T-mesh is a partitioning of a two-dimensional rectangular domain $[\xi_0,\xi_n]\times[\eta_0,\eta_n]$ into smaller rectangles by horizontal and vertical lines.

Definition 5. A **Tensor Mesh** is a Box Mesh where there are no T-joints, i.e. all horizontal and vertical lines span the entire length $[\xi_0,\xi_n]$ or $[\eta_0,\eta_n]$.

Definition 6. An **LR-Mesh** $\mathcal{M}_n$ is a Box Mesh which is the result from a series of single line insertions $\{\varepsilon_i\}_{i=1}^n$ from a initial tensor mesh $\mathcal{M}_0$, i.e. $\mathcal{M}_n \supset \mathcal{M}_{n-1} \supset ... \supset \mathcal{M}_1 \supset \mathcal{M}_0$ and each intermediate state $\mathcal{M}_{i+1} = \{\mathcal{M}_i \cup \varepsilon_i\}$ is a also a Box Mesh.

In other words, it must be possible to create the mesh by inserting one line at a time, where these lines never stop in the center of an element (knot span). See Figure 3 for examples of the different meshes.

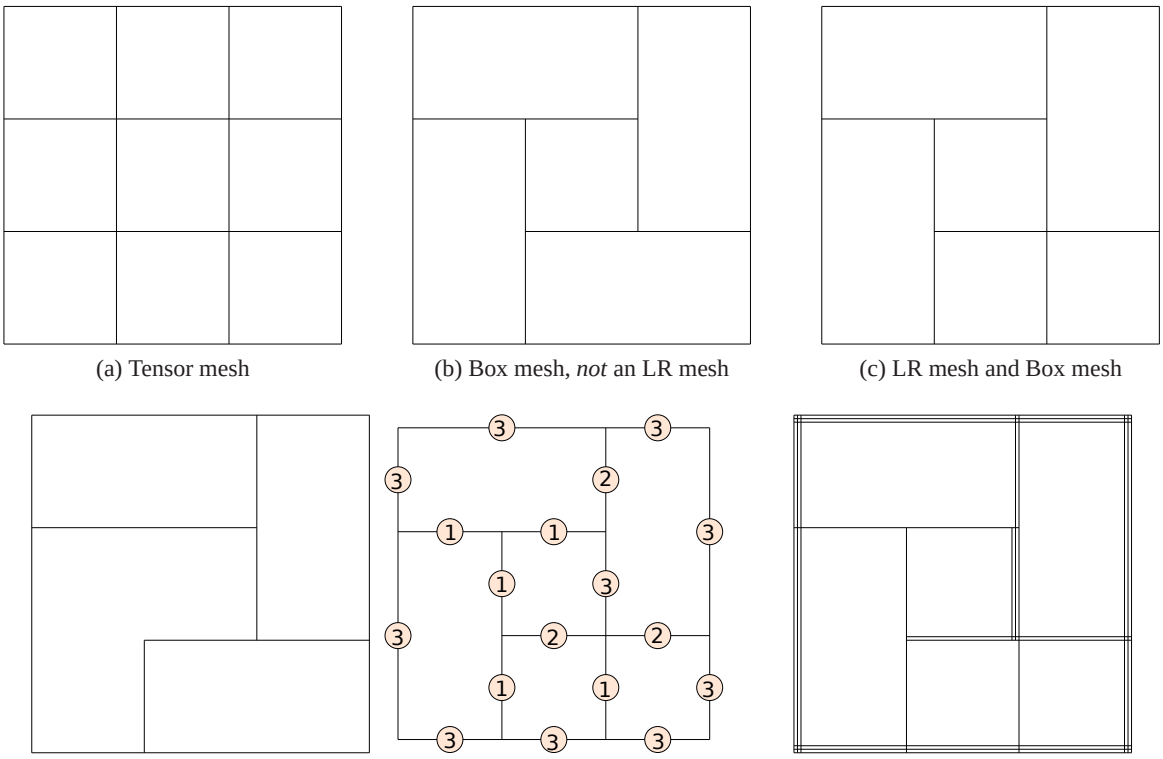
Definition 7. A Box Mesh, Tensor Mesh or LR-Mesh **with multiplicities** is a Mesh where each line segment has a corresponding integer value n, called the line multiplicity. Each multiplicity must satisfy $0 < n \leq p$, where p is the polynomial degree (in ξ-direction for vertical lines and in η-direction for horizontal lines).

Note that it is possible to create a C^{-1}-basis if using knot lines of multiplicity $n = p$.

Definition 8. The **support** of a (weighted) B-spline $B : \mathbb{R}^2 \to \mathbb{R}$

$$\begin{aligned} B(\xi,\eta) &= \gamma B_{\Xi}(\xi) B_{\mathcal{H}}(\eta) \\ \Xi &= [\xi_0,\xi_1,...,\xi_{p_1+1}] \\ \mathcal{H} &= [\eta_0,\eta_1,...,\eta_{p_2+1}] \end{aligned} \tag{6}$$

is the closure of all points where it takes nonzero value, i.e. $(\xi,\eta) \in [\xi_0,\xi_{p_1+1}] \times [\eta_0,\eta_{p_2+1}]$.



(a) Tensor mesh (b) Box mesh, *not* an LR mesh (c) LR mesh and Box mesh

(d) Not an LR-mesh, nor a box mesh (e) LR mesh with multiplicities (f) Alternative way of drawing (e)

Figure 3: Note that there is no way to create the box mesh (b) from single line insertions (starting at tensor mesh) where every intermediate state is also a box mesh. This is a prerequisite for all LR meshes.

Definition 9. A meshline ε is said to **traverse** the support of a (weighted) B-spline $B : \mathbb{R}^2 \to \mathbb{R}$ (see (6)) if

- a horizontal line $\varepsilon = [\xi_0^*, \xi_1^*] \times \eta^*$ satisfies

$$\begin{aligned} \xi_0^* \leq \xi_0, \quad \xi_{p_1+1} \leq \xi_1^* \\ \eta_0 \leq \eta^* \leq \eta_{p_2+1}, \end{aligned}$$

- a vertical line $\varepsilon = \xi^* \times [\eta_0^*, \eta_1^*]$ satisfies

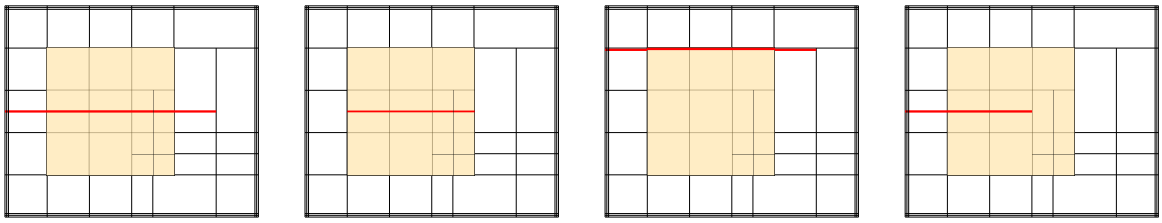
$$\begin{aligned} \xi_0 \leq \xi^* \leq \xi_{p_1+1} \\ \eta_0^* \leq \eta_0, \quad \eta_{p_2+1} \leq \eta_1^*. \end{aligned}$$

A horizontal line is said to traverse **the interior** if $\eta_0 < \eta^* < \eta_{p_2+1}$ and traverse **the edge** if $\eta_0 = \eta^*$ or $\eta_{p_2+1} = \eta^*$. Similarly for vertical lines it is said to traverse the interior if $\xi_0 < \xi^* < \xi_{p_1+1}$ and traverse the edge if $\xi_0 = \xi^*$ or $\xi_{p_1+1} = \xi^*$.

See Figure 4 for examples on traversing meshlines.

Definition 10. A (weighted) B-spline $B : \mathbb{R}^2 \to \mathbb{R}$ (see (6)) has **minimal support** on a LR Mesh $\mathcal{M}$ if

1. for every horizontal line $\varepsilon = [\xi_0^*, \xi_1^*] \times \eta^*$ of multiplicity n in the mesh $\mathcal{M}$ that traverses the support of B, there exist $\begin{cases} n \text{ unique } i \text{ such that } \eta_i = \eta^* & \text{, if } \varepsilon \text{ traverses the interior of } B \\ \text{an } i \text{ such that } \eta_i = \eta^* & \text{, if } \varepsilon \text{ traverses the edge of } B \end{cases}$
2. for every vertical line $\varepsilon = \xi^* \times [\eta_0^*, \eta_1^*]$ of multiplicity n in the mesh $\mathcal{M}$ that traverses the support of B, there exist $\begin{cases} n \text{ unique } i \text{ such that } \xi_i = \xi^* & \text{, if } \varepsilon \text{ traverses the interior of } B \\ \text{an } i \text{ such that } \xi_i = \xi^* & \text{, if } \varepsilon \text{ traverses the edge of } B \end{cases}$



(a) Line traversing the interior of B

(b) Line traversing the interior of B

(c) Line traversing the edge of B

(d) Line neither traversing the interior nor the edge of B

Figure 4: Traversing the support of a basis function. Note that we distinguish between traversing the edge and the interior of the support of B.

See Figure 5 for examples on minimal support.

Definition 11. Let $\mathcal{M}$ be an LR-mesh with multiplicities. A function $B : \mathbb{R}^2 \to \mathbb{R}$ is called an **LR B-spline** on $\mathcal{M}$ if

1. $B^{\gamma}_{\underline{\Xi}}(\boldsymbol{\xi}) = \gamma B_{\Xi}(\xi) B_{\mathcal{H}}(\eta)$ is a weighted B-spline where all knot lines (and the knot line multiplicities) in Ξ and $\mathcal{H}$ is also in $\mathcal{M}$.
2. B has minimal support on $\mathcal{M}$.

Definition 12. A **meshline extension** ε on an LR mesh $\mathcal{M}_n$ is either

- a new meshline,
- an elongation of an existing meshline,
- a joining of two existing meshlines or
- increasing the multiplicity of an existing line

which causes one or more of the LR B-splines on $\mathcal{M}_n$ to not have minimal support on $\mathcal{M}_{n+1}$.

2.2 Refining LR B-splines

For local refinement, we again turn to existing spline theory. Tensor product B-splines form a subset of the LR B-splines and they obey some of the same core refinement ideas (globally not locally). From tensor product B-spline theory we know that one might insert extra knots to enrich the basis without changing the geometric description. This comes from the fact that we have available the relation between B-splines in the old coarse spline space and in the new enriched spline space. For instance if we want to insert the knot $\hat{\xi}$ into the knot vector Ξ between the knots ξ_{i-1} and ξ_i, then the relation is given by

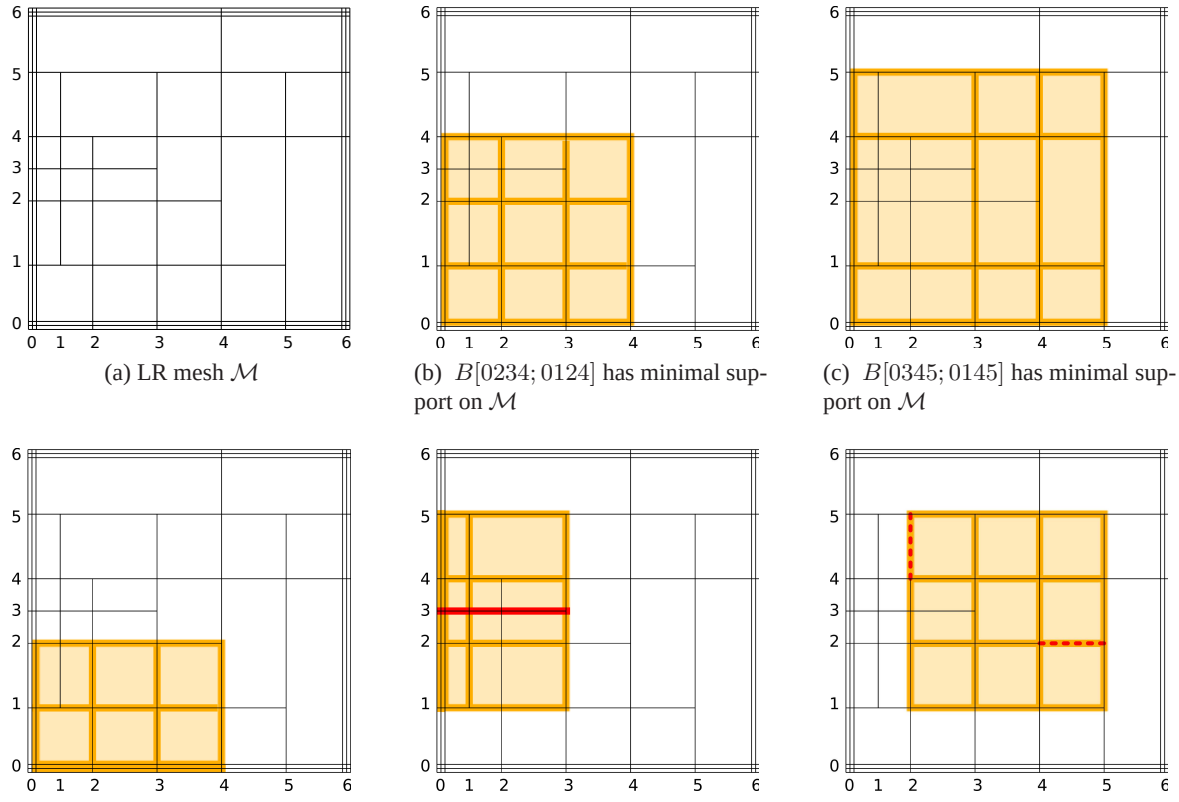
$$B_{\Xi}(\xi) = \alpha_1 B_{\Xi_1}(\xi) + \alpha_2 B_{\Xi_2}(\xi), \tag{7}$$

where

$$\begin{aligned} \alpha_1 &= \begin{cases} 1, & \xi_{p+1} \le \hat{\xi} \le \xi_{p+2} \\ \frac{\hat{\xi}-\xi_1}{\xi_{p+1}-\xi_1}, & \xi_1 \le \hat{\xi} \le \xi_{p+1} \end{cases} \\ \alpha_2 &= \begin{cases} \frac{\xi_{p+2}-\hat{\xi}}{\xi_{p+2}-\xi_2}, & \xi_2 \le \hat{\xi} \le \xi_{p+2} \\ 1, & \xi_1 \le \hat{\xi} \le \xi_2 \end{cases} \end{aligned} \tag{8}$$

and the knot vectors are

$$\begin{aligned} \Xi &= [\xi_1, \xi_2, ...\xi_{i-1}, \quad \xi_i, ...\xi_{p+1}, \xi_{p+2}] \\ \Xi_1 &= [\xi_1, \xi_2, ...\xi_{i-1}, \hat{\boldsymbol{\xi}}, \ \xi_i, ...\xi_{p+1} \quad\ ] \\ \Xi_2 &= [\quad \xi_2, ...\xi_{i-1}, \hat{\boldsymbol{\xi}}, \ \xi_i, ...\xi_{p+1}, \xi_{p+2}]. \end{aligned}$$



(a) LR mesh $\mathcal{M}$

(b) $B[0234; 0124]$ has minimal support on $\mathcal{M}$

(c) $B[0345; 0145]$ has minimal support on $\mathcal{M}$

(d) $B[0234; 0012]$ has minimal support on $\mathcal{M}$

(e) $B[0013; 1245]$ has **not** minimal support on $\mathcal{M}$ due to the meshline at $\eta = 3$

(f) $B[2345; 1245]$ has minimal support on $\mathcal{M}$, but is not an LR B-spline on $\mathcal{M}$ as the two highlighted lines are missing from $\mathcal{M}$

Figure 5: Minimal support ensures that every meshline traversing the support of a B-spline should appear in the local knot vector. Being an LR B-spline ensures the converse: that every line in the knot vector appears in the mesh $\mathcal{M}$

Note that the insertion of the knot $\hat{\xi}$ into Ξ yields a knot vector of size $p + 3$, meaning that it is generating two B-splines. These two B-splines are the one being described by the local knot vectors Ξ_1 and Ξ_2, both of size $p + 2$.

Let us look at an example using this technique. Say we want to insert $\hat{\xi} = \frac{3}{2}$ into the B-spline $\Xi_3 = [0, 1, 2, 3]$. This would give us $\alpha_1 = \alpha_2 = \frac{3}{4}$ and the three functions are plotted in Figure 6. If one were to insert the knot $\hat{\xi} = \frac{3}{2}$ into the *set* of B-splines in Figure 2, then this will require two more functions to be split, namely the function $\Xi_2 = [0, 0, 1, 2]$ and $\Xi_4 = [1, 2, 3, 3]$. All the three splitting shown in Figure 6–7 will then take place. This insertion will replace three old B-splines with four new linearly independent B-splines (see the knot vectors in the figure legend to identify the four distinctive new B-splines).

Bivariate functions are refined in one parametric direction at a time. By pairing two local knot vectors, one for each of the parametric directions we are able to create a bivariate B-spline. For instance if we have the knot vector Ξ in the first parametric direction, and $\mathcal{H}$ in the second, we will have the B-spline $B_{\Xi,\mathcal{H}}(\xi, \eta) = B_{\Xi}(\xi)B_{\mathcal{H}}(\eta)$.

By using the splitting algorithm in Equation (7) for the 2D case when splitting in one direction, we obtain:

$$\begin{aligned} B_{\boldsymbol{\Xi}}(\xi, \eta) &= B_{\Xi}(\xi)B_{\mathcal{H}}(\eta) \\ &= (\alpha_1 B_{\Xi_1}(\xi) + \alpha_2 B_{\Xi_2}(\xi))\, B_{\mathcal{H}}(\eta) \\ &= \alpha_1 B_{\boldsymbol{\Xi}_1}(\xi, \eta) + \alpha_2 B_{\boldsymbol{\Xi}_2}(\xi, \eta). \end{aligned} \tag{9}$$

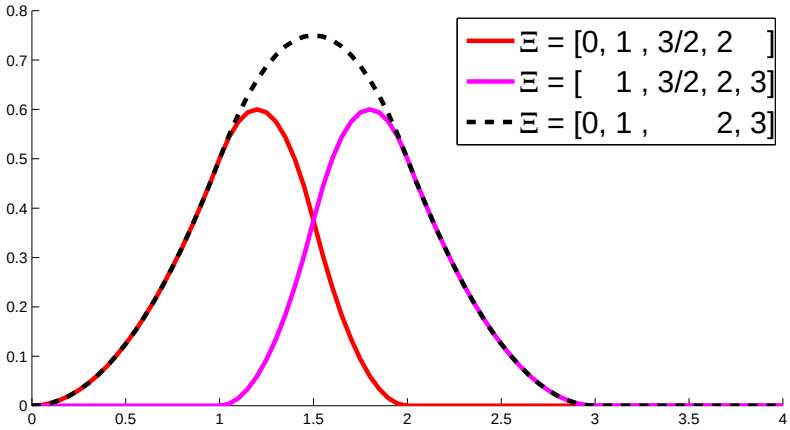
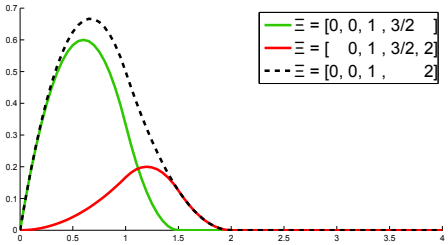
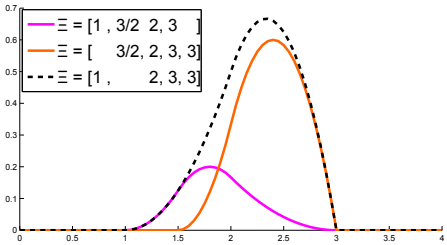


Figure 6: Splitting the B-spline $\Xi = [0, 1, 2, 3]$ into two separate B-splines by inserting the knot $\frac{3}{2}$.



(a) Inserting $\xi = \frac{3}{2}$ in $\Xi = (0, 0, 1, 2)$.



(b) Inserting $\xi = \frac{3}{2}$ in $\Xi = (1, 2, 3, 3)$.

Figure 7: Displaying function splitting in the case that $\hat{\xi}$ is not at the knotvector center.

For weighted B-splines, this becomes

$$\begin{aligned} B^{\gamma}_{\boldsymbol{\Xi}}(\xi,\eta) &= \gamma B_{\boldsymbol{\Xi}}(\xi,\eta) \\ &= \gamma\left(\alpha_1 B_{\Xi_1}(\xi) + \alpha_2 B_{\Xi_2}(\xi)\right) B_{\mathcal{H}}(\eta) \\ &= B^{\gamma_1}_{\boldsymbol{\Xi}_1}(\xi,\eta) + B^{\gamma_2}_{\boldsymbol{\Xi}_2}(\xi,\eta), \end{aligned}$$

where

$$\begin{aligned} \gamma_1 &= \alpha_1\gamma \\ \gamma_2 &= \alpha_2\gamma. \end{aligned}$$

We now have everything we need to formulate the refinement rules. This will be implemented by keeping track of the mesh $\mathcal{M}_n$ and the spline space $\mathcal{S}_n$. Note that we do not need to keep track of the refinement history $\mathcal{M}_i, \quad i = 1...n-1$, we only need to store the current state. For each B-spline $B^{\gamma_i}_{\boldsymbol{\Xi}_i}$ we store the following:

- $\Xi_i \in \mathbb{R}^{p+2}$ - the local knot vector in the first parametric direction
- $\mathcal{H}_i \in \mathbb{R}^{p+2}$ - the local knot vector in the second parametric direction
- $\gamma_i \in \mathbb{R}$ - the scaling weight
- $\boldsymbol{c}_i \in \mathbb{R}^d$ - control points in d-dimensional space.

Through the refinement we aim at two points: keeping the partition of unity and leaving the geometric mapping unchanged, i.e. $\sum_i \gamma_i B_i = 1$ and $\boldsymbol{f} = \sum_i \gamma_i B_i \boldsymbol{c}_i$ for all levels of refinement.

Assuming a meshline extension is inserted, the refinement process is characterized by two steps.

- **Step 1:** Split any B-spline which support is traversed by the *new* meshline - update the weights and control points
- **Step 2:** For all new B-splines, check if their support is completely traversed by any *existing* meshline

We will here describe these two steps in detail.

2.2.1 When to split a B-spline

A B-spline may need to be split at either of the two refinement steps. In Step 1 we test *every* B-spline against *one* meshline. In Step 2 we test every *newly created* B-spline against *all* existing meshlines. This is just a conceptual understanding of the process. In a computational realization of this technique, both of these searches could be done locally.

A B-spline is split whenever a meshline is traversing the interior of that B-spline (see Definition 9).

In the refinement process we will at multiple stages perform checks to see if one particular function is split by one particular meshline. The algorithm is in essence testing B-splines against meshlines, one for one, and splitting every function that satisfies the splitting criterion. The rest is just formulating which B-splines are going to be checked against which meshlines. Note that in the case of a meshline extension being an elongation of an existing meshline or a joining of two existing ones, then we will use the full length of the meshline to flag B-splines for splitting. Thus, in the case of an elongation, we will be using the union of the old line with the elongation and use this combined length when testing if lines are traversing B-splines.

2.2.2 How to split a B-spline

The splitting itself is done through the use of Equation (8) and (9). Let us assume that the function B_i will be split and the result is the functions B_1 and B_2 with corresponding α_1 and α_2. We will now have to make sure that we keep the geometric mapping unchanged and preserves the partition of unity. There are two cases which can arise:

- The new function (B_1 or B_2) already exist in our spline space (due to previous splitting).
- The new function is not present and must be added.

In the latter case of the function not already existing, we will need to create it. We simply add it to our list of B-spline and give it weight and control point equal to it's parent function, i.e. $\gamma_1 := \alpha_1\gamma_i$ and $\boldsymbol{c}_1 := \boldsymbol{c}_i$. We then proceed to add it to the list of newly created functions which will be subsequently tested for splitting in Step 2. In the former case of the function already being present, we simply update the weight and control point and continue with the refinement process. In this case, the control point will be given as $\boldsymbol{c}_1 := \left(\boldsymbol{c}_1\gamma_1 + \boldsymbol{c}_i\gamma_i\alpha_1\right)/(\gamma_1+\gamma_i\alpha_1)$ and the weight will be given by $\gamma_1 := \gamma_1+\gamma_i\alpha_1$. Finally, we remove the old function from our list of B-splines. This is illustrated in Algorithm 1 where we have assumed that the inserted knot is in the Ξ-vector (the $\mathcal{H}$-case being completely analogue). Note that we are keeping the unrefined knot vector $\mathcal{H}_i$ unchanged in line 6-7, as it is apparent in Equation (9). We are also storing all newly created B-splines in $\mathcal{S}_{new}$ as these will be required in Step 2 of the refinement algorithm.

2.2.3 LR spline definition

We define an LR spline as an application of the refinement algorithm.

Definition 13. An **LR spline** $\mathcal{L}$ is a pair $(\mathcal{M}_n, \mathcal{S})$, where $\mathcal{M}_n$ is an LR mesh and $\mathcal{S}$ is a set of LR B-splines on $\mathcal{M}_n$, and

- for each intermediate step $\mathcal{M}_{i+1} = \{\mathcal{M}_i \cup \varepsilon_i\}$ the new line ε_i is a meshline extension
- $\mathcal{S} = \left\{B_{\boldsymbol{\Xi}_i}(\boldsymbol{\xi})\right\}_{i=1}^{m}$ is the set of all LR B-splines on $\mathcal{M}_n$ resulting from Algorithm 2.

Algorithm 1 Local ξ-split

parameters: $\hat{\xi}$ {new knot}; B_i {B-spline to be split ($B_i \in \mathcal{S}$)}; $\mathcal{S}$ {Spline space}; $\mathcal{S}_{new}$ {Functions not present in $\mathcal{S}$}

calculate (α_1, α_2) from (8)
$\Xi \leftarrow$ SORT($\Xi \cup \hat{\xi}$)
$\Xi_1 \leftarrow [\xi_1, ..., \xi_{p+2}]$
$\Xi_2 \leftarrow [\xi_2, ..., \xi_{p+3}]$
$\mathcal{H}_1 \leftarrow \mathcal{H}_i$
$\mathcal{H}_2 \leftarrow \mathcal{H}_i$
if $(\Xi_1, \mathcal{H}_1) \in \mathcal{S}$ **then**
    $\boldsymbol{c}_1 \leftarrow (\boldsymbol{c}_1\gamma_1 + \boldsymbol{c}_i\gamma_i\alpha_1)\,/(\gamma_1 + \alpha_1\gamma_i)$
    $\gamma_1 \leftarrow \gamma_1 + \alpha_1\gamma_i$
else
    $\boldsymbol{c}_1 \leftarrow \boldsymbol{c}_i$
    $\gamma_1 \leftarrow \alpha_1\gamma_i$
    add B_1 to $\mathcal{S}_{new}$
end if
if $(\Xi_2, \mathcal{H}_2) \in \mathcal{S}$ **then**
    $\boldsymbol{c}_2 \leftarrow (\boldsymbol{c}_2\gamma_2 + \boldsymbol{c}_i\gamma_i\alpha_2)\,/(\gamma_2 + \alpha_2\gamma_i)$
    $\gamma_2 \leftarrow \gamma_2 + \alpha_2\gamma_i$
else
    $\boldsymbol{c}_2 \leftarrow \boldsymbol{c}_i$
    $\gamma_2 \leftarrow \alpha_2\gamma_i$
    add B_2 to $\mathcal{S}_{new}$
end if
remove B_i from $\mathcal{S}$

Algorithm 2 LR B-spline refinement

parameters: $\mathcal{S}$ {Spline space}; $\mathcal{M}$ {LR mesh}; $\mathcal{E}$ {Meshline extension}

for every B-spline $B_i \in \mathcal{S}$ **do**
    if $\mathcal{E}$ splits B_i **then**
        perform split according to Algorithm 1
    end if
end for
for every B-spline $B_i \in \mathcal{S}_{new}$ **do**
    for every existing edge $\mathcal{E}_j \in \mathcal{M}$ **do**
        if $\mathcal{E}_j$ splits B_i **then**
            perform split according to Algorithm 1
            {note that this may enlarge $\mathcal{S}_{new}$ further}
        end if
    end for
end for

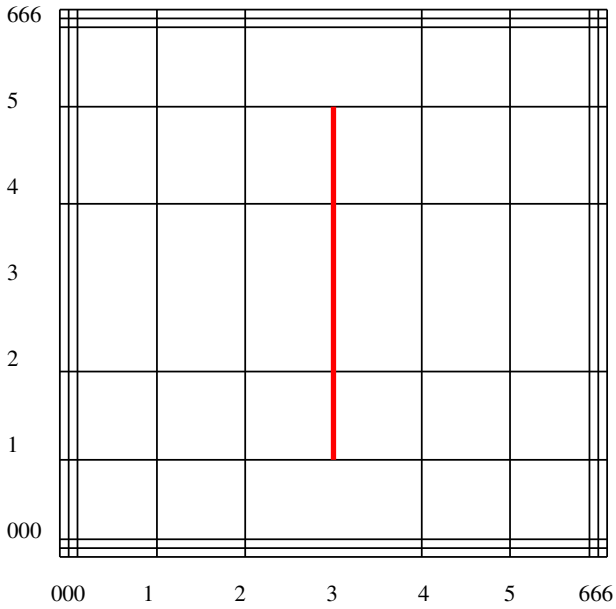


Figure 8: Inserting a local (vertical) meshline into a tensor product mesh.

We note that, there is no backwards dependence on the mesh, meaning that while the index in $\mathcal{M}_n$ seems to suggest that the LR spline is a sequence of meshes, it is enough that there exist one possible sequence. After we have constructed the set of LR B-splines on $\mathcal{M}_n$, it is safe to discard any link to the previous mesh $\mathcal{M}_{n-1}$.

Further, it does not matter if it is possible to make the mesh $\mathcal{M}_n$ in multiple ways. Indeed *any* ordering of the meshline insertions will produce the exact same end function space $\mathcal{S}$. See Section 2.3. As such, for any given LR mesh $\mathcal{M}_n$, the set of LR B-splines $\mathcal{S}$ is unique.

While it is possible to define LR splines by using non-weighted B-splines, as done in [6], we will here only consider weighted ones as to maintain the partition of unity which is important in finite element methods.

Definition 14. The **cardinality** of an LR spline $\mathcal{L} = (\mathcal{M}_n, \mathcal{S})$, where $\mathcal{S} = \left\{ B^{\gamma}_{\boldsymbol{\Xi}_i}(\boldsymbol{\xi}) \right\}_{i=1}^{m}$ is the number of B-splines in the set $\mathcal{S}$, and is denoted

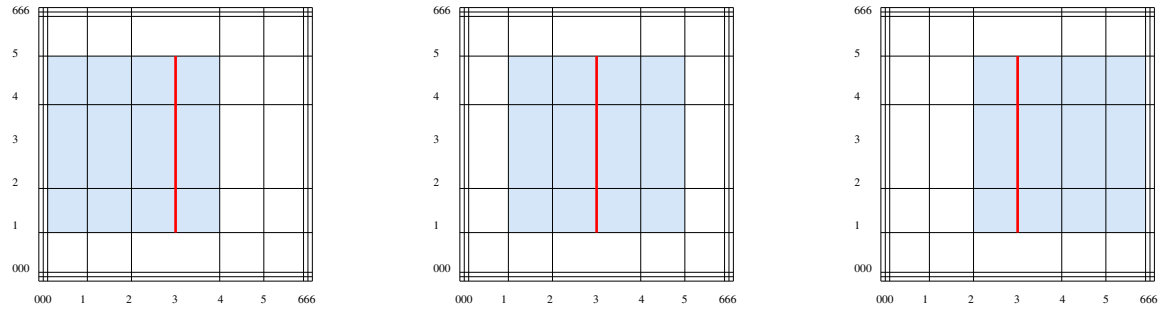
$$|\mathcal{L}| = m. \tag{10}$$

2.2.4 Example

As an example we look at the insertion of two local knot lines in the tensor product mesh given by the global knot vectors $\Xi = \mathcal{H} = [0, 0, 0, 1, 2, 4, 5, 6, 6, 6]$. We first introduce the line spanning $(\xi, \eta) \in (3, 1) \rightarrow (3, 5)$, see Figure 8. The line will split the three B-splines illustrated in Figure 9. We calculate the corresponding α-values from Equation (8) and get

$$\begin{aligned} B[0124;1245] &= B[0123;1245] + \tfrac{1}{3}B[1234;1245] \\ B[1245;1245] &= \tfrac{2}{3}B[1234;1245] + \tfrac{2}{3}B[2345;1245] \\ B[2456;1245] &= \tfrac{1}{3}B[2345;1245] + B[3456;1245] \end{aligned} \tag{11}$$

Updating these splits sequentially, we get the following. Let the numerical indices $i = 1, 2, 3, 4$ denote



(a) $B[0124;1245] = B[0123;1245] + \frac{1}{3}B[1234;1245]$

(b) $B[1245;1245] = \frac{2}{3}B[1234;1245] + \frac{2}{3}B[2345;1245]$

(c) $B[2456;1245] = \frac{1}{3}B[2345;1245] + B[3456;1245]$

Figure 9: B-splines split by the new meshline.

	Splitting B_a		Splitting B_b		Splitting B_c	
	γ_i	$\boldsymbol{c}_i$	γ_i	$\boldsymbol{c}_i$	γ_i	$\boldsymbol{c}_i$
B_1	1	$\boldsymbol{c}_a$	1	$\boldsymbol{c}_a$	1	$\boldsymbol{c}_a$
B_2	1/3	$\boldsymbol{c}_a$	1	$\frac{1}{3}\boldsymbol{c}_a + \frac{2}{3}\boldsymbol{c}_b$	1	$\frac{1}{3}\boldsymbol{c}_a + \frac{2}{3}\boldsymbol{c}_b$
B_3			2/3	$\boldsymbol{c}_b$	1	$\frac{2}{3}\boldsymbol{c}_b + \frac{1}{3}\boldsymbol{c}_c$
B_4					1	$\boldsymbol{c}_c$

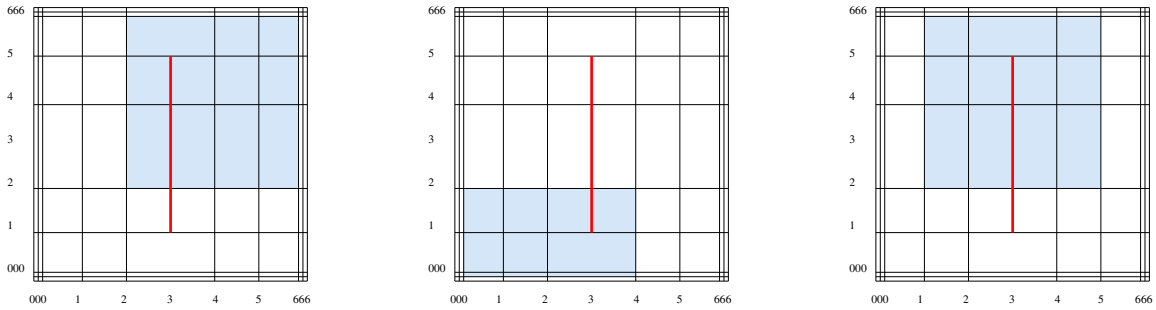
Table 1: Numerical values for weights and control points as Algorithm 2 iterates to insert the meshline in Figure 8.

the new B-splines and alphabetical indices $i = a, b, c$ denote the old basis, i.e.

$$\begin{aligned} B_1 &= B[0123;1245] \\ B_2 &= B[1234;1245] \\ B_3 &= B[2345;1245] \\ B_4 &= B[3456;1245] \\ B_a &= B[0124;1245] \\ B_b &= B[1245;1245] \\ B_c &= B[2456;1245] \end{aligned}$$

Note that at the tensor product case, all weights γ will be equal to one. After the first split of the old function B_a, we establish the new functions B_1 and B_2. Their weights will simply be the α-values 1 and 1/3 and the control points will remain unchanged $\boldsymbol{c}_1 = \boldsymbol{c}_2 = \boldsymbol{c}_a$. Splitting the second function in Equation (11) B_b will cause one of the results B_2 to be already present, so we update the corresponding weights and control point according to line 9 - 10 in Algorithm 1. The process is shown in Table 2 and the numerical values are tabulated sequentially according to whenever each of the B-splines B_a, B_b and B_c are being split.

We would now proceed to Step 2, and test every new B-spline B_1, B_2, B_3 and B_4 against all previously inserted meshlines, but as this is the first inserted line, this is unnecessary. Some of the supports of the unrefined B-splines are depicted in Figure 10.



(a) $B[24565; 2456]$ (b) $B[0124; 0012]$ (c) $B[1245; 2456]$

Figure 10: B-splines *not* split by the new meshline.

Next we insert another line, this time spanning $(\xi, \eta) \in (1, 3) \rightarrow (5, 3)$ as shown in Figure 11. We first iterate through **Step 1** of the refinement. Here, 4 B-splines will be completely traversed by the new meshline as illustrated in Figure 12. We keep our old convention of numbering the old B-splines by alphabetical letters $i = a, b, c, d$ and the new B-splines by numerical numbers $i = 1, 2, \ldots$.

$$\begin{aligned}
B[1234; 1245] &= \frac{2}{3}B[1234; 1234] + \frac{2}{3}B[1234; 2345] \\
B[2345; 1245] &= \frac{2}{3}B[2345; 1234] + \frac{2}{3}B[2345; 2345] \\
B[1245; 2456] &= \frac{1}{3}B[1245; 2345] + B[1245; 3456] \\
B[1245; 0124] &= B[1245; 0123] + \frac{1}{3}B[1245; 1234]
\end{aligned}$$

or more compact

$$\begin{aligned}
B_a &= \frac{2}{3}B_1 + \frac{2}{3}B_2 \\
B_b &= \frac{2}{3}B_3 + \frac{2}{3}B_4 \\
B_c &= \frac{1}{3}B_5 + B_6 \\
B_d &= B_7 + \frac{1}{3}B_8
\end{aligned}$$

We note that none of the new B-splines on the right hand side are equal, so all will be considered as new functions with corresponding weights and control points set by line 12 - 13 in Algorithm 1. Again, we show some of the B-splines in the vicinity of the newest meshline that are not splitted, see Figure 13.
We now proceed to **Step 2** of Algorithm 2. There are 8 new B-splines and all of these would have to be checked against the previous line and see if they are to be further split. As it turns out, two of the new functions are now completely traversed by the previous line since their support have decreased with the knot insertion. These are B_5 and B_8 as depicted in Figure 14

$$\begin{aligned}
B[1245; 2345] &= \frac{2}{3}B[1234; 2345] + \frac{2}{3}B[2345; 2345] \\
B[1245; 1234] &= \frac{2}{3}B[1234; 1234] + \frac{2}{2}B[2345; 1234]
\end{aligned}$$

or

$$\begin{aligned}
B_5 &= \frac{2}{3}B_2 + \frac{2}{3}B_4 \\
B_8 &= \frac{2}{3}B_1 + \frac{2}{2}B_3
\end{aligned}$$

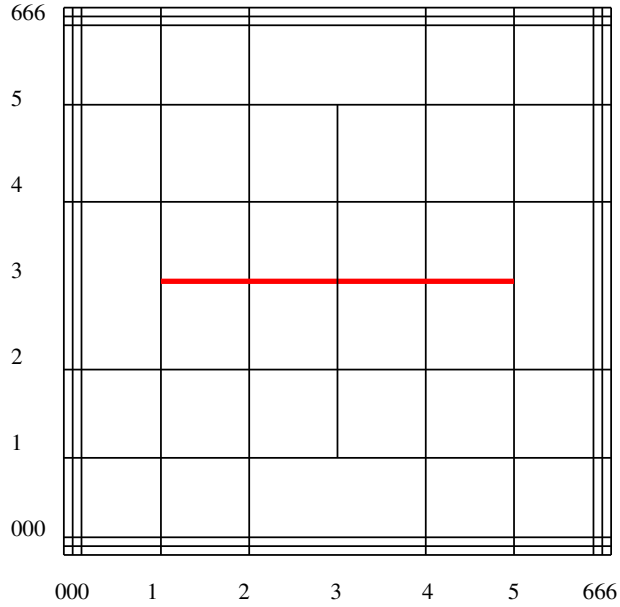
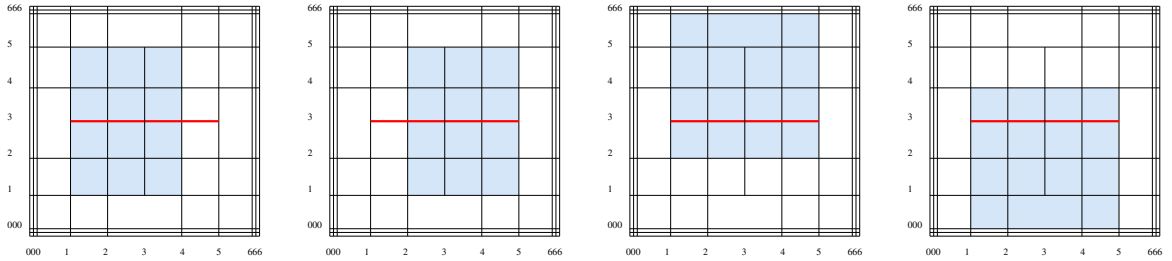


Figure 11: Inserting another local (horizontal) meshline.



(a) Basis B_a split, $B[1234;1245] = \frac{2}{3}B[1234;1234] + \frac{2}{3}B[1234;2345]$

(b) Basis B_b split, $B[2345;1245] = \frac{1}{3}B[2345;1234] + B[2345;2345]$

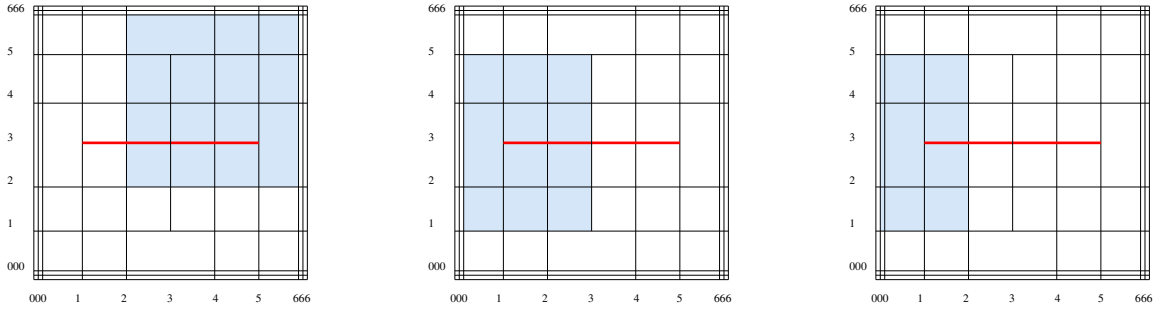
(c) Basis B_c split, $B[1245;2456] = B[1245;2345] + \frac{1}{3}B[1245;3456]$

(d) Basis B_d split, $B[1245;0124] = \frac{1}{3}B[1245;0123] + B[1245;1234]$

Figure 12: B-splines split by the new meshline.

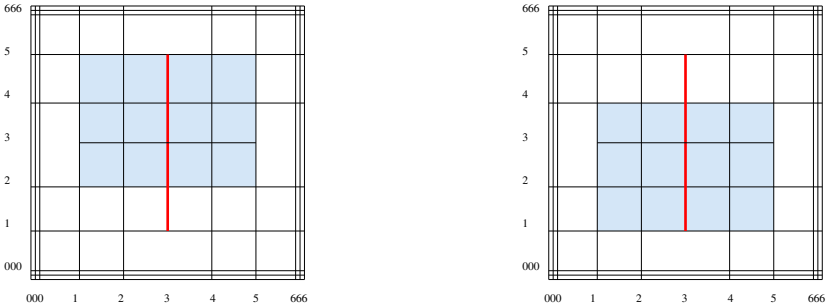
	Step 1		**Step 2**			
	Splitting B_a, B_b, B_c and B_d		Splitting B_5		Splitting B_8	
	γ_i	$\boldsymbol{c}_i$	γ_i	$\boldsymbol{c}_i$	γ_i	$\boldsymbol{c}_i$
B_1	2/3	$\boldsymbol{c}_a$	2/3	$\boldsymbol{c}_a$	8/9	$\frac{1}{8}(6\boldsymbol{c}_a + 2\boldsymbol{c}_d)$
B_2	2/3	$\boldsymbol{c}_a$	8/9	$\frac{1}{8}(6\boldsymbol{c}_a + 2\boldsymbol{c}_c)$	8/9	$\frac{1}{8}(6\boldsymbol{c}_a + 2\boldsymbol{c}_c)$
B_3	2/3	$\boldsymbol{c}_b$	2/3	$\boldsymbol{c}_b$	8/9	$\frac{1}{8}(6\boldsymbol{c}_b + 2\boldsymbol{c}_d)$
B_4	2/3	$\boldsymbol{c}_b$	8/9	$\frac{1}{8}(6\boldsymbol{c}_b + 2\boldsymbol{c}_c)$	8/9	$\frac{1}{8}(6\boldsymbol{c}_b + 2\boldsymbol{c}_c)$
B_5	1/3	$\boldsymbol{c}_c$	<Remove B_5>			
B_6	1	$\boldsymbol{c}_c$	1	$\boldsymbol{c}_c$	1	$\boldsymbol{c}_c$
B_7	1	$\boldsymbol{c}_d$	1	$\boldsymbol{c}_d$	1	$\boldsymbol{c}_d$
B_8	1/3	$\boldsymbol{c}_d$	1/3	$\boldsymbol{c}_d$	<Remove B_8>	

Table 2: Numerical values of weights and control points as Algorithm 2 iterates to insert the meshline in Figure 11.



(a) $B[2456; 2456]$ (b) $B[0123; 1245]$ (c) $B[0012; 1245]$

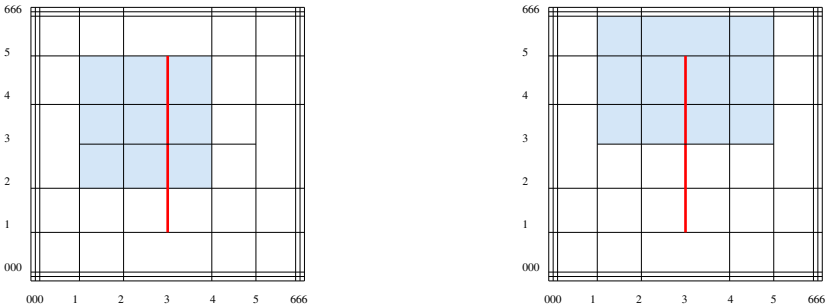
Figure 13: B-splines *not* split by the new meshline.



(a) B-spline $B_5 = B[1245; 2345]$ split by the old meshline

(b) B-spline $B_8 = B[1245; 1234]$ split by the old meshline

Figure 14: *New* B-splines split by an *old* meshline in Step 2 of the refinement algorithm.



(a) B-spline $B_2 = B[1234; 2345]$ not split by the old meshline (knotline already present)

(b) B-spline $B_6 = B[1245; 3456]$ not split by the old meshline

Figure 15: New B-splines unchanged by the existing meshline in Step 2 of the refinement algorithm.

2.3 LR spline properties

Consider a LR spline $(\mathcal{M}_n, \mathcal{S})$. Then

1. $\sum_{i=1}^{m} \gamma_i B_i(\boldsymbol{\xi}) = 1$, i.e. the LR B-splines form a partition of unity.
2. $(\mathcal{M}_{i+1}, \mathcal{S}_{i+1}) \supset (\mathcal{M}_i, \mathcal{S}_i)$, i.e. the LR spline is nested.
3. If there exists two meshline insertions lists $\{\varepsilon_0, \varepsilon_1, ..., \varepsilon_{n-1}\}$ and $\{\tilde{\varepsilon}_0, ..., \tilde{\varepsilon}_{n-1}\}$ such that $\mathcal{M}_{i+1} = \{\mathcal{M}_i \cup \varepsilon_i\}$, $\tilde{\mathcal{M}}_{i+1} = \{\tilde{\mathcal{M}}_i \cup \tilde{\varepsilon}_i\}$ and the final mesh is equal $\mathcal{M}_n = \tilde{\mathcal{M}}_n$, then the spline space is equal $\mathcal{S}_n = \tilde{\mathcal{S}}_n$, i.e. the LR spline refinement is order independent.
4. $\mathcal{S} = \{B_i(\boldsymbol{\xi})\}_{i=1}^{m}$ does in general not form a linearly independent set.

2.3.1 Partition of unity

The set of LR B-splines form a partition of unity, i.e.

$$\sum_{i=1}^{m} \gamma_i B_i(\boldsymbol{\xi}) = 1 \tag{12}$$

Proof: Since the refinement consists of repeated use of Algorithm 1, we will only show that the partition of unity is preserved through one step of this algorithm. The global result then follows from induction. Following our convention of enumerating old B-splines alphabetical and new B-splines numerical, let us name the three functions $\{a, 1, 2\}$ such that

$$B_a(\boldsymbol{\xi}) = \alpha_1 B_1(\boldsymbol{\xi}) + \alpha_2 B_2(\boldsymbol{\xi}) \tag{13}$$

There are three outcomes of the algorithm:

- B_1 and B_2 already exist in the spline space
- B_1, but not B_2 already exist in the spline space
- neither B_1 nor B_2 exist in the spline space.

We will here show that this holds for the first case, since the proof for the two other cases are completely analog. Assume that the partition of unity holds before splitting, i.e. (12), then

$$\begin{array}{llll} \sum_{\substack{i=1 \\ i\neq 1,2,a}}^{m} \gamma_i B_i(\boldsymbol{\xi}) & +(\gamma_1 + \alpha_1\gamma_a) B_1(\boldsymbol{\xi}) & +(\gamma_2 + \alpha_2\gamma_a) B_2(\boldsymbol{\xi}) & = \\ \sum_{\substack{i=1 \\ i\neq 1,2,a}}^{m} \gamma_i B_i(\boldsymbol{\xi}) & +\gamma_1 B_1(\boldsymbol{\xi}) & +\gamma_2 B_2(\boldsymbol{\xi}) & +\gamma_a(\alpha_1 B_1(\boldsymbol{\xi}) + \alpha_2 B_2(\boldsymbol{\xi})) = \\ \sum_{\substack{i=1 \\ i\neq 1,2,a}}^{m} \gamma_i B_i(\boldsymbol{\xi}) & +\gamma_1 B_1(\boldsymbol{\xi}) & +\gamma_2 B_2(\boldsymbol{\xi}) & +\gamma_a B_a(\boldsymbol{\xi}) = 1 \end{array} \tag{14}$$

2.3.2 Nested space

A spline space $\mathcal{S}_{i-1} \subset \mathcal{S}_i$ is said to be nested, if for any $f \in \mathcal{S}_{i-1}$ there exist an $\hat{f} \in \mathcal{S}_i$ such that $f = \hat{f}$. For LR splines the functions f and $\hat{f}$ can be represented by their control points as $f = \sum_{i=1}^{n} B_i c_i$ and $\hat{f} = \sum_{i=1}^{m} B_i \hat{c}_i$. In order to find the relation between an arbitrary $f = [\boldsymbol{c}]$ and $\hat{f} = [\hat{\boldsymbol{c}}]$, we need to find the relationship between their control points defining them.

As the refinement algorithm, is a repeated use of (7), which is a linear relation, we can formulate the relations between the set of old B-splines and the set of enriched B-splines as a matrix $C \in \mathbb{R}^{n \times m}$, satisfying

$$B_{old} = C B_{new}. \tag{15}$$

Hence given any $f = [\boldsymbol{c}]$, we can find $\hat{f} = [\hat{\boldsymbol{c}}]$ by $\hat{\boldsymbol{c}} = C^T \boldsymbol{c}$.

The LR meshes $\mathcal{M}$ are as such nested by construction.

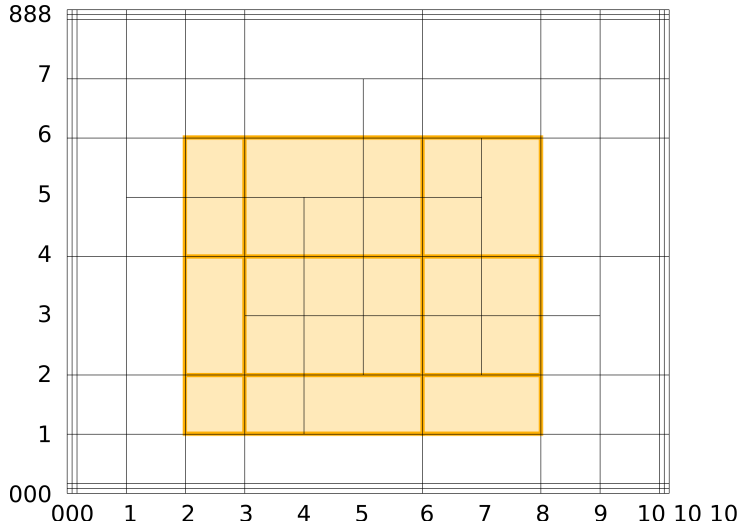


Figure 16: Example of a linearly dependent LR mesh using biquadratic B-splines. The shaded B-spline $B[2356; 1246]$ is a linear combination of 7 smaller B-splines; their relation given in (16)

2.3.3 Independence of meshline insertion order

LR splines are independent of the ordering at which the meshline extensions are inserted. That means if $\mathcal{L} = \{\mathcal{M}_n, \mathcal{S}_n\}$, $\hat{\mathcal{L}} = \{\hat{\mathcal{M}}_m, \hat{\mathcal{S}}_m\}$ and the meshes are equal $\mathcal{M}_n = \hat{\mathcal{M}}_m$, then $\mathcal{S}_n = \hat{\mathcal{S}}_m$. For this proof, see Dokken *et al.* [6].

2.3.4 Linear dependence

The LR splines are not linearly independent, in general. As an example for linearly dependent set of LR B-splines, see Figure 16. Here the linear dependence relation is given as:

$$\begin{aligned} 720 \cdot B[2368; 1246] \quad = \quad & 108 \cdot B[5678; 2346] + 135 \cdot B[2356; 2456] + \\ & 108 \cdot B[3567; 3456] + 268 \cdot B[3456; 2345] + \\ & 324 \cdot B[4567; 2345] + 360 \cdot B[2346; 1245] + \\ & 384 \cdot B[3468; 1234]. \end{aligned} \tag{16}$$

2.4 Linear independence of LR splines

As shown above, one cannot guarantee that an arbitrary LR-mesh is producing a linearly independent set of functions, however there are several ways to ensure that the system of functions you get is in fact linearly independent. We will here briefly describe three methods, but for full details we reference the work of Dokken et.al [6].

2.4.1 Hand-in-hand principle

Mourrain [15] presented a formula independent of choice of basis for the dimensionality of a spline space over a T-mesh. This result is generalized in [6] to also address general multiplicities and any dimension. Used in the bivariate setting it provides a topological equation based on the polynomial degrees, the elements, the edges and their multiplicities and the vertices. By observing the change of these components, we are able to predict the dimension increase of the spline space for classes of meshline insertions.

Definition 15. A **primitive meshline extension** is a meshline extension which increases the dimension of the spline space by one.

In particular we note that any meshline extension of the following type

- inserting a new meshline spanning p elements,
- elongation of a meshline by one element and
- increasing the multiplicity of a meshline of length p

are primitive.

Proposition 1. *Let $\mathcal{L} = \{\mathcal{M}, \mathcal{S}\}$ be a refinement of $\hat{\mathcal{L}} = \{\hat{\mathcal{M}}, \hat{\mathcal{S}}\}$, where $\mathcal{M} = \{\hat{\mathcal{M}} \cup \varepsilon\}$. If ε is a primitive meshline extension on $\hat{\mathcal{M}}$ and $\hat{\mathcal{S}}$ is linearly independent then $\mathcal{S}$ is linearly independent on $\mathcal{M}$ if and only if*

$$|\mathcal{L}| = \left|\hat{\mathcal{L}}\right| + 1. \tag{17}$$

Proof: We know from the dimension formula formula that the dimension of the function space should be equal to one greater than the dimension of the old space. What is left to show is that the LR Spline space does in fact increase in size, i.e. the new function is not a linear combination of the existing ones. This can be seen from a continuity perspective. Any meshline extension will decrease the continuity of the basis at the point where the mesh is extended and at least one B-spline will in the splitting scheme include this new knot within its support. Thus $\hat{\mathcal{S}} \subset \mathcal{S}$, with strict inclusion and the theorem is proved.

Take note that many meshline extensions can be formulated as a series of primitive meshline extensions. One such example is the insertion of a new meshline of length n which can be formulated as one new meshline of length p and $p - n$ meshline extensions of length one. One can as such carefully go hand-in-hand and ensure that the LR spline at all stages of the refinement is coinciding with the theoretical dimension proved by Mourrain.

2.4.2 Peeling algorithm

Another option is by the peeling algorithm and the notion of local linear independence.

Definition 16. An **element** $\mathcal{R} = (\xi_1, \xi_2) \times (\eta_1, \eta_2)$ in a box mesh is an open set where no horizontal or vertical lines cross.

Note that elements on LR splines means that all B-splines are C^∞ on $\mathcal{R}$ since all reduced continuity appears across meshlines.

Definition 17. An element $\mathcal{R}$ in an LR spline is said to be **locally linearly independent** if there exist no choice of coefficients $\{c_i\}$ such that

$$\sum_{i \in \mathcal{S}_\mathcal{R}} c_i B_i(\boldsymbol{\xi}) = 0, \quad \forall \boldsymbol{\xi} \in \mathcal{R} \tag{18}$$

except for the trivial solution $c_i = 0, \forall i$. Here $\mathcal{S}_\mathcal{R}$ denotes the set of all B-splines with support on the element $\mathcal{R}$.

Since all B-splines are polynomials when restricted to one particular element, it is clear that an element is locally linearly independent if and only if the set $\mathcal{S}_\mathcal{R}$ consists of exactly $(p+1)(q+1)$ B-splines, where p and q is the polynomial degree of the LR spline.

The Peeling algorithm is given in Algorithm 3. Here, we keep track of two things: The set of B-splines that may appear in a (global) linear dependence relation

$$\sum_i c_i B_i(\boldsymbol{\xi}) = 0 \tag{19}$$

and the set of possible areas where this may occur. Line 2-11 is just initialization where we remove all locally linearly independent elements and the B-splines with support on these. The next lines comes

Algorithm 3 Peeling algorithm

1: **parameters:** $\mathcal{S}$ {Spline space}; Ω {Parametric domain}; $\mathcal{S}_{LD}$ {possible linearly dependent B-splines}; Ω_{LD} {areas of possible linear dependence}
2: $\mathcal{S}_{LD} \leftarrow \mathcal{S}$
3: $\Omega_{LD} \leftarrow \Omega$
4: **for** every element $\mathcal{R} \in \Omega$ **do**
5: **if** $\mathcal{R}$ is locally linearly independent **then**
6: $\Omega_{LD} \leftarrow \Omega_{LD} \setminus \mathcal{R}$
7: **for** every B-spline B_i with support on $\mathcal{R}$ **do**
8: $\mathcal{S}_{LD} \leftarrow \mathcal{S}_{LD} \setminus B_i$
9: **end for**
10: **end if**
11: **end for**
12: **while** Ω_{LD} or $\mathcal{S}_{LD}$ changed **do**
13: **for** all elements $\mathcal{R} \in \Omega_{LD}$ **do**
14: **if** $\mathcal{R}$ has support of exactly one B-spline $B_i \in \mathcal{S}_{LD}$ **then**
15: $\Omega_{LD} \leftarrow \Omega_{LD} \setminus \mathcal{R}$
16: $\mathcal{S}_{LD} \leftarrow \mathcal{S}_{LD} \setminus B_i$
17: **end if**
18: **end for**
19: **end while**

from the realization that (19) may never contain only one term. There have to be at least two B-splines with coefficients $c_i \neq 0$ for a nontrivial relation to exist. We may then in line 14-16 remove this B-spline and the element from possible linear combinations. The removal of B_i in line 16 will in turn decrease the number of B-splines with support on several rectangles which may cause the if-statement in line 14 to trigger more. As such, one may peel away B-spline after B-spline until one of two things happen: There are no B-splines left in the set $\mathcal{S}_{LD}$ and $\mathcal{S}$ is proven linearly independent or all elements $\mathcal{R} \in \mathcal{M}_{LD}$ have support of two or more B-splines. In the latter case there *may* exist a linear combination, and further investigation is required.

2.4.3 The tensor expansion

From any LR spline $\mathcal{L} = \{\mathcal{M}, \mathcal{S}\}$ expand the LR mesh $\mathcal{M}$ to a full tensor mesh. This will create a map from the LR spline basis to the tensor product basis and can be described as (15). Since we know that the tensor product B-splines are linearly independent, the LR spline basis $\mathcal{S}$ will also be linearly independent if and only if the matrix C has full rank [14].

In a computational realization of these methods it is possible to describe the matrix C using rational numbers under the assumption that the initial tensor product mesh $\mathcal{M}_0$ consisted of integer or rational knots. This is due to the fact that all refinements are done by halving each knot interval, and all splitting of B-splines, results in rational expressions seen in (8). It is then possible to compute the rank of the matrix using *exact* arithmetics, and this is what is done in the work within this paper. For a more computational efficient implementation of this method, consider using integers modulus some high Mersenne prime, for instance $p = 2^{31} - 1$. This gives a faster, more robust method at the cost of the very unlikely event that one of the matrix entries by chance becomes a multiple of p and the method produces the wrong result.

2.4.4 Choice of methodology

The tensor expansion for checking for linear independence does not scale to well with increasing problem sizes, but it has the advantage of always working for all meshes and continuities. The hand-in-hand principle has the drawback that it disallows a few refinements and the peeling algorithm, while necessary, it is not sufficient for linearly dependent meshes, meaning that it can prove that an LR spline is linearly independent, but it cannot prove it linearly dependent.

It is of course possible to combine these techniques, where one could for instance narrow the possible areas of linear dependence down to only a subset of the mesh using the peeling algorithm and proceeding with tensor expansions in only these areas for full verification.

For all numerical experiments presented in this paper, we tested for linear independence using the tensor expansion method, and no linearly dependent cases were discovered when using the full span and the structured mesh refinement techniques of Section 4 (below).

In fact we conjecture that by virtue of the particular choice of refinement scheme (full span and structured mesh) you will always be in a subset of the linearly independent LR Splines, similar to the subclass of T-splines which are the analysis suitable T-splines [14].

3 Isogeometric analysis

3.1 The Galerkin finite element method

3.1.1 The variational formulation

Many problems in science and engineering can be addressed by solving a variational problem. Given a Hilbert space $\mathcal{V}$, a continuous, coercive bilinear form $a(\cdot,\cdot)$ and a continuous linear functional $l \in \mathcal{V}^*$, where $\mathcal{V}^*$ is the dual space to $\mathcal{V}$, the variational formulation is defined by: find $u \in \mathcal{V}$ such that such that

$$a(u,v) = l(v) \quad \forall v \in \mathcal{V}. \tag{20}$$

The existence and uniqueness of the solution to this continuous problem is guaranteed by the Lax-Milgram theorem. The Galerkin Finite Element (FE) approximation to this variational problem may then be given as follow: Given a finite subspace $\mathcal{V}_h \subset \mathcal{V}$ and $l \in \mathcal{V}^*$, find $u_h \in \mathcal{V}_h$ such that

$$a(u_h,v_h) = l(v_h) \quad \forall v_h \in \mathcal{V}_h. \tag{21}$$

3.1.2 A priori error estimates

For cases when the bilinear form $a(\cdot,\cdot)$ is selfadjoint the FE-solution u_h is the optimal approximation to the analytical solution u as measured in the "a-norm" (often denoted "energy-norm" symbolized with $_E$):

$$||u-u_h||_E = \sqrt{a(u-u_h,u-u_h)}. \tag{22}$$

If the analytical solution (of a variational problem involving first order differentiation) is sufficiently smooth, i.e. $u \in H^{p+1}$, and the FE mesh $\mathcal{M}_0$ is regular and quasi-uniform, the error in the approximate FE-solution on a family of uniformly refined meshes $\{\mathcal{M}_k\}$, is bounded by

$$||u-u_h||_E \leq Ch^p||u||_{H^{p+1}}, \tag{23}$$

where C is some problem-dependent constant, h is the characteristic size of the finite elements, p denotes the highest degree of a complete polynomial in the FE basis and $||u||_{H^{p+1}}$ denotes the Sobolev norm of order $p+1$.

For problems where the solution is not sufficiently smooth, $u \notin H^{p+1}$, e.g. problems with singularities within the solution domain or on its boundary, we have the error bound

$$||u-u_h||_E \leq Ch^\alpha||u||_{H^{\alpha+1}}, \tag{24}$$

where the value of the non-negative real parameter α depends on how the family of meshes $\{\mathcal{M}_k\}$ are created. Assume that λ is a real number characterizing the strength of the singularity. For a sequence of uniformly, or nearly uniformly, refined meshes we then have

$$\alpha = \min\{p,\lambda\}. \tag{25}$$

Thus, when $\lambda < p$ the rate of convergence is limited by the strength of the singularity, and not to the polynomial order.

3.1.3 Adaptive mesh refinement (AMR)

For classical FEM the main method for obtaining an optimal grid for minimizing the global energy error (a-norm), has been to do adaptive grid refinement with the aim of obtaining an (quasi) uniform element error distribution. This approach has shown to be effective in order to eliminate any "pollution" from singularities in the domain or at the boundary as well as achieving optimal convergence rates for problems involving rough right hand sides. Thus, by means of adaptive mesh refinement we may achieve

$$\alpha = p \tag{26}$$

An important step in such adaptive refinement processes is *a posterirori* error estimation that provides a reliable element error distribution, see Ainsworth and Oden [1]. To achieve (quasi) uniform element error distribution we may subdivide those elements that have an element error that is above the average, or we may restrict ourself to subdivide those with the β percent elements with largest error contribution.

In classical FEM, the traditional way of refining a quadrilateral element is by subdivision, i.e. inserting a cross to obtain four new elements. If the aspect ratio (width to length ratio) is undesirable large, one may extend the algorithm to inserts only a single line, splitting the element into two new elements. This way of adaptive refinement give raise to so-called "hanging nodes" for which there are several techniques to reestablish the appropriate C^0- continuity.

3.2 The isogeometric finite element method

3.2.1 Spline spaces

In isogeometric FE methods we introduce splines as basis functions. The most common spline bases are tensor, either defined by B-splines or NURBS. Herein we will use tensor-product B-splines as well as locally refined B-splines denoted LR B-splines. Given the global knot vectors $\Xi = [\xi_1, \xi_2, ..., \xi_{m+p+1}]$ and $\mathcal{H} = [\eta_1, \eta_2, ..., \eta_{n+q+1}]$ and an $m \times n$ grid of control points $\mathbf{C}_{i,j}$, then a tensor 2D B-splines surface of polynomial order p in x-direction and q in y-direction may be written as follows:

$$\mathbf{F}(\xi, \eta) = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{C}_{i,j} \, B_{p,\Xi_i}(\xi) \cdot B_{q,\mathcal{H}_j}(\eta) \tag{27}$$

Here, the local knot vectors Ξ_i and $\mathcal{H}_j$ are defined as a subsequence of the corresponding global knot vectors Ξ and $\mathcal{H}$, respectively. The function space we use is given by

$$\mathcal{S}_{p,q} = \text{span}\left\{B_{p,i}(\xi) \otimes B_{q,j}(\eta)\right\}_{i=1,j=1}^{m,n} \tag{28}$$

For LR B-splines, these will instead be defined over a single running global index i using the local knot vectors Ξ_i and $\mathcal{H}_i$ by

$$\mathbf{F}(\xi, \eta) = \sum_{i=1}^{n_{dof}} \gamma_i \mathbf{C}_i B_{p,\Xi_i}(\xi) \cdot B_{q,\mathcal{H}_i}(\eta) \tag{29}$$

where the regularity is given by the local knot vectors and γ_i is the weighting factors needed to obtained partition of unity, see the Section 2. Let the function space spanned by LR B-spline basis functions be denoted by

$$\mathcal{L}_{p,q} = \text{span}\left\{B_{\boldsymbol{\Xi}_i}(\xi, \eta)\right\}_{i=1}^{n} \tag{30}$$

A proper function space for the FE trial functions v_h and the FE test functions w_h to achieve a compatible FE space is as follows:

$$\mathcal{V}_h(\Omega) = \left\{w_h \in \mathcal{V}(\Omega) \mid w_h(\mathbf{F}^{-1}(x_1, x_2)) \in \mathcal{L}_{p,q}(\xi, \eta)\right\} \tag{31}$$

where the coordinate mapping F is assumed to be an onto and invertible mapping between the parameter domain denoted by $\hat{\Omega}$ and the true domain Ω, i.e. for any $(x_1, x_2) \in \Omega$ there exist $(\xi^*, \eta^*) \in \hat{\Omega}$ such that $(x_1, x_2) = F(\xi^*, \eta^*)$.

3.2.2 Convergence rates for splines

Our model problem herein is the Poisson problem for which we have that:

$$||u - u_h||_E = \sqrt{a(u - u_h, u - u_h)} = (\nabla(u - u_h), \nabla(u - u_h)) = |u - u_h|_{H^1}. \tag{32}$$

Thus, for our model problem the a-norm is equal to the H^1 semi-norm. The a priori convergence estimate given in Equation (23) is proven to hold for tensor B-splines (or NURBS), see Bazilevs *et al.* [3]. We conjecture that it also holds for the LR B-splines that we are using herein. Note that we are not actively using the a priori convergence rates in our adaptive refinement strategies. However, we will in the numerical studies compare the obtained convergence rates towards the ones given by Equation (23).

3.2.3 Adaptive refinement of isogeometric finite elements

As shown in Section 3.1.3 by proper adaptive mesh refinement (AMR) we may utilize the full power of higher order methods and that is highly relevant for Isogeometric FE-methods. One important application of LR B-splines is to use them as an enabling technology for achieving optimal convergence order, i.e. accurate and efficient FE-models. Herein, our aim is to demonstrate and test the performance obtain by adaptive refinement using LR B-splines. Thus, we have chosen to investigate this by solving benchmark problems with known analytical solution — the exact error is thus computable.

The refinement algorithm chosen herein is based on increasing our solution space by $\beta \cdot n_{dof}$ new degrees of freedom for each iteration, where β is a prescribed growth parameter. This is achieved by continued refinement of the elements having the greatest elemental error contribution, ρ_e, to the global relative error, ρ

$$\rho_e = \frac{||u - u_h||_{E(\Omega_e)}}{||u||_{E(\Omega_e)}} \quad \text{and} \quad \rho = \frac{||u - u_h||_{E(\Omega)}}{||u||_{E(\Omega)}} = \sqrt{\sum_e^{n_{el}} \rho_e^2} \tag{33}$$

Typical we choose $5\% \leq \beta \leq 20\%$. Small values for β gives more accurate refinement process, whereas larger values result in fewer refinement steps.

Regarding adaptive refinement of isogeometric finite elements there have been some recent attempts using T-splines and hierarchical B-splines, see [5, 7, 21, 22]. It is common to mark those elements (knot-span) with highest energy error and subdivide into into four new elements by inserting a cross (ignoring large aspect ratio elements for now) through the element center.

As discussed in the previous section, the length of the crossing LR meshlines will have to be of a certain length in order to actually split a basis function properly. The actual length is depending on the surrounding topology of the mesh, and may split some neighboring elements into two new elements. In order to do a splitting we will need to compute the length of the new knot lines to ensure a proper meshline extension. This can be extracted by the element to basis function correspondence, which lists all basis functions which have support on each particular element. This is already available as it is needed in the assembly of the stiffness matrix, so no extra computations are required. Moreover, this eliminates the need for expensive topological searches.

4 Adaptive mesh refinement using LR B-splines

4.1 LR spline refinements

Although Dokken *et al.* [6] describe how to manipulate the LR B-splines when inserting knot lines, it is still up to the implementer to choose exactly which knot lines to use for refinement purposes. The inserted knot lines must at least entirely split an existing B-spline, which puts a minimum length requirement on it. This is to ensure a proper meshline extension which causes a B-spline in a state of having not minimal support, according to Definition 12. We have several options available when doing the refinement. Not only the length and the position of the knot lines, but also their multiplicity as splines in general open for duplicate knots. We will in the numerical examples investigate how much impact these choices made in the refinement process have on the properties of the resulting LR B-splines space.

As a starting point for the refinement algorithm we have a list of element error contributions ρ_e, see (33). This may be an estimated error based on some a posterior error estimator, or exact error in case of a known exact solution. For all results presented here, we are using the latter. A straightforward implementation would be to refine the β percent elements with largest error contribution. However, this will not suffice for our purposes, since we will be comparing different refinement schemes. To ensure a proper growth of the solution space $\mathcal{S}$ we propose to continuously insert new knot lines according to some algorithm until we have β percent more B-splines in our spline space. This will ensure that different refinement schemes are comparable.

To see an example of the converse, consider inserting a new knot in a 1D univariate set of B-splines. Using multiplicity 1, this will increase your spline space by 1, while using multiplicity p will cause p new B-splines, even though the number of elements refined is the same in both cases.

Definition 18. The **refinement parameter** β of an iterative scheme is defined such that two LR splines $\mathcal{L}_i$ and $\mathcal{L}_{i-1}$ satisfy

$$\begin{aligned} \mathcal{L}_{i-1} &\subset \mathcal{L}_i \\ (1+\beta)\,|\mathcal{L}_{i-1}| &\leq |\mathcal{L}_i| \end{aligned}$$

Which simply states that $\mathcal{L}_i$ should be a refinement of $\mathcal{L}_{i-1}$ and the number of B-splines (not elements) should grow by at least β percent each iteration.

Assume we have a LR-mesh as given in Figure 17 and we want to refine the element $T_e = [2,4] \times [1,2]$. To find out what are appropriate LR meshline extensions, we list all B-splines with support on this element and these are shown in Figure 18 and tabulated in Table 3. An obvious choice when choosing the refinement line, is to make sure that it is refining *every* B-spline over that element. This can simply be done by making it run from the smallest ξ-knot value to the largest, and likewise in the other parametric direction. In this particular case one would then have a ξ-line going from $\xi = 0$ to $\xi = 6$ with constant

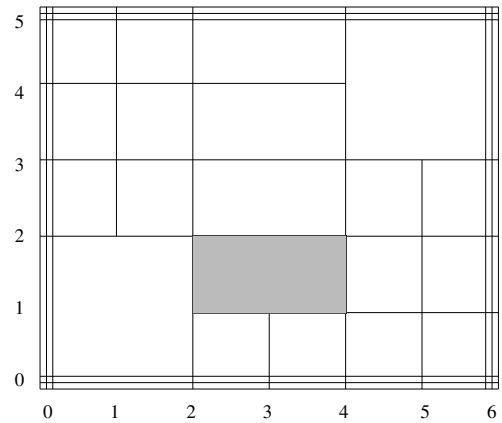


Figure 17: LR-mesh with an element marked for refinement

Table 3: B-splines with support on $[2,4] \times [1,2]$ in Figure 17 - 18.

i	Ξ_i	$\times$	$\mathcal{H}_i$
1	$[0024]$	$\times$	$[0002]$
2	$[0245]$	$\times$	$[0002]$
3	$[2456]$	$\times$	$[0012]$
4	$[0024]$	$\times$	$[0023]$
5	$[0245]$	$\times$	$[0023]$
6	$[2456]$	$\times$	$[0123]$
7	$[0024]$	$\times$	$[0234]$
8	$[0246]$	$\times$	$[0235]$
9	$[2466]$	$\times$	$[1235]$

$\eta = 1.5$ such that it passes through the center of the element. For the other parametric line, this would then have to run from $\eta = 0$ to $\eta = 5$ through $\xi = 3$. This is the "safe" way of refining, as one will have little reason to prioritize refining some B-splines over others given the limited information available.

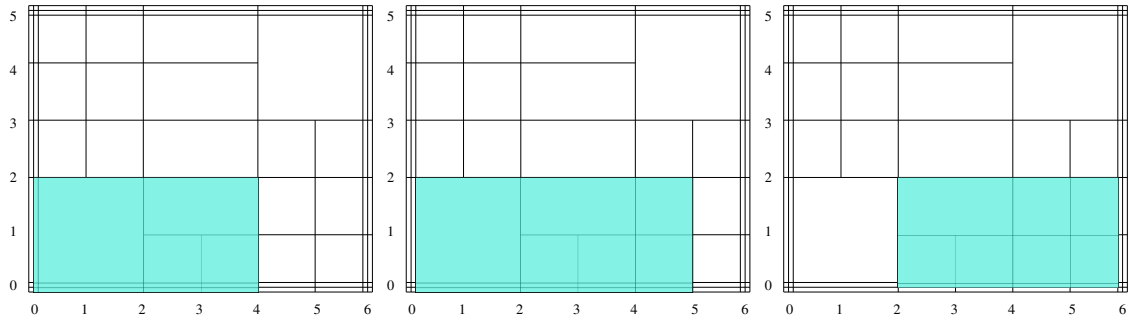
However another obvious option also comes to mind. Since we are doing *local* refinement, it is natural to want the refinement lines to be as local as possible. One might argue that we should insert the smallest possible line, while still being long enough that it actually splits an entire B-spline. This information can directly be computed from the local B-spline list in Table 3 as one can iterate through the list and choose the smallest knot vector and choose this. Note that this comes with the loss of uniqueness, since both Ξ_1 and Ξ_3 has parametric ξ-length 4 in the above example. We now must decide on whether to use Ξ_1 and insert $[0,4]$ or to use Ξ_3 and insert $[2,6]$. Of course this will have implications on the resulting spline space, but without more knowledge of the underlying problem, it is not possible to say which one is advantageous over the other. We thus propose to pick a *random* function of all the available with smallest parametric support. Notice again we don't need any topological information about the surrounding mesh topology to insert the knot lines. We are simply extracting every information we need locally, and the resulting refinement will not yield any more than what is extracted here.

As all generalizations of B-spline spaces, LR B-splines do also allow for duplicate knots. Duplicate knots work just in the same way as they do for regular tensor product B-splines, in which the B-splines all have continuity C^{p-m} where p is the spline polynomial degree and m is the knot multiplicity. We may choose to insert not regular knot lines, but also knot lines of higher multiplicity. This is obviously a good idea if we actually *want* the lower continuity for instance if we either are doing geometry modelling or if we know something about the underlying problem. However this is not the only reason to include the double knot lines. As will become apparent later, this will help us keep the refinement more localized and reduce the propagation effects.

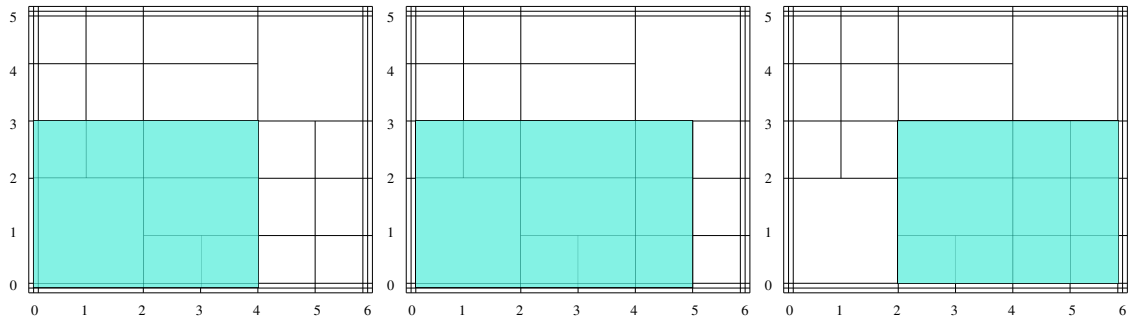
So to sum up: we have several design parameters when choosing a refinement scheme. We may choose

- which B-splines with support on the element to refine
- the location of the element split
- the multiplicity of the inserted knot lines

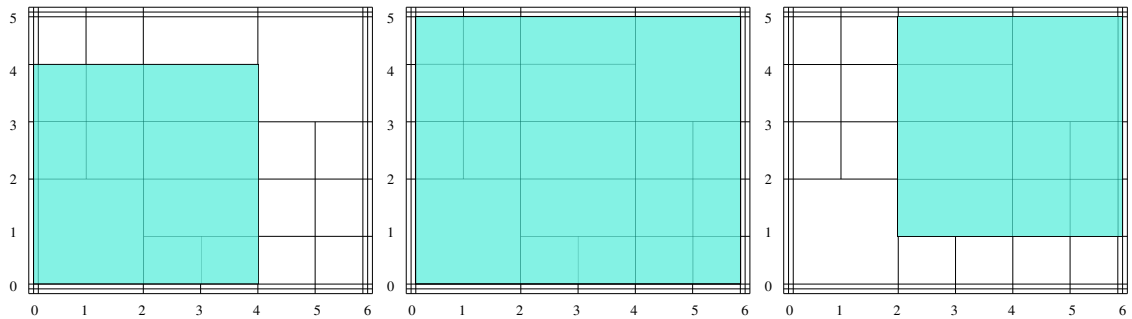
the second point refers to the fact that one doesn't necessary need to create a cross through the element *center* as described in the example above. There is nothing preventing us from inserting two lines in both directions through each element, placing them one third from the edge each, effectively splitting the element into 9. For this discussion, we will restrict ourself to inserting crosses through the element centers. These are all illustrated in Figure 19 where we refine a shaded element using different techniques.



(a) $B[0024; 0002]$ (b) $B[0245; 0002]$ (c) $B[2456; 0012]$

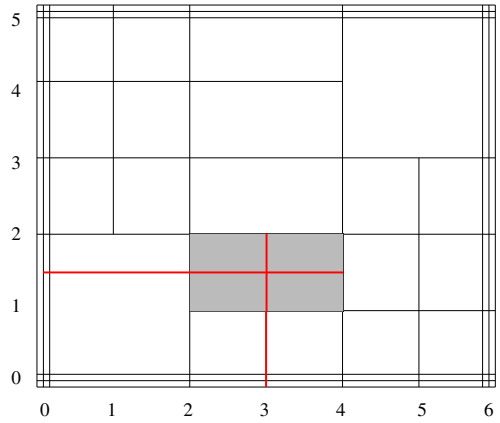


(d) $B[0024; 0023]$ (e) $B[0245; 0023]$ (f) $B[2456; 0123]$

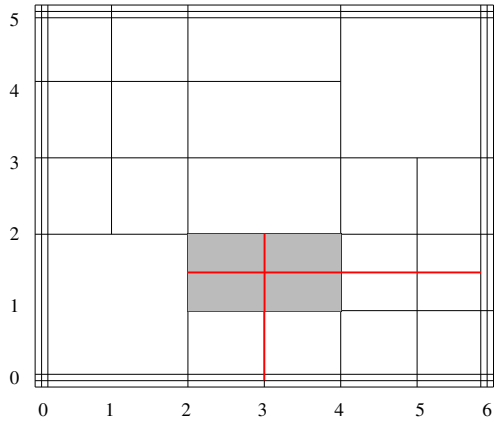


(g) $B[0024; 0234]$ (h) $B[0246; 0235]$ (i) $B[2466; 1235]$

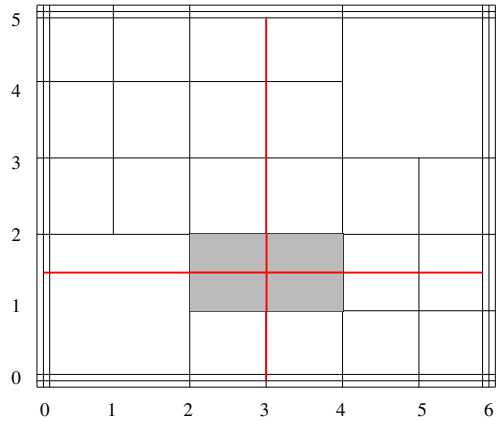
Figure 18: All B-splines with support on the element $[2, 4] \times [1, 2]$.



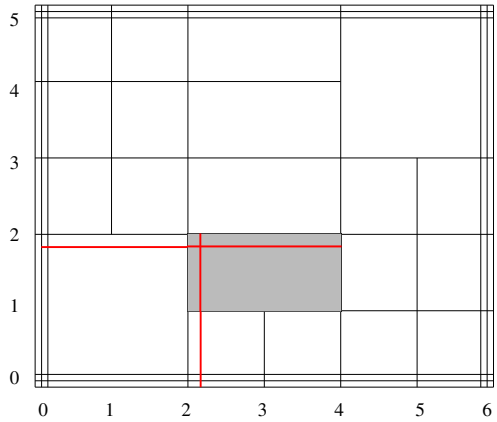
(a) Minimal span refinement - refining B-spline in Figure 18a



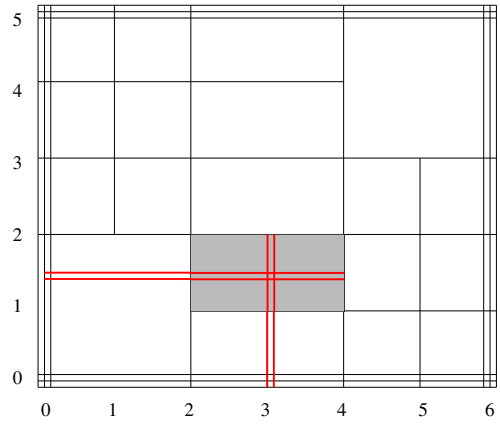
(b) Minimal span refinement - refining B-spline in Figure 18c



(c) Refining all B-splines with support (see Figure 18)



(d) Off center line insertion splitting same as in (a)



(e) Duplicate knot line insertion, splitting the same as in (a)

Figure 19: Different choices for refining the shaded element.

4.2 Local refinement strategies for LR B-splines

We will here present three different local refinement strategies that will be used in our numerical examples. The starting point for all of these is the assumption that we have identified a set of elements which needs refinement and proceed to refine these using one of three strategies. The goal is to split the marked element (knot span) into four new elements by inserting a cross. However, as already discussed, this cross cannot be limited to only spanning the marked element.

4.2.1 Full span

Our strategy here is to refine *every* B-spline with support on the marked element. The inserted meshline in the ξ-direction will then have to span from the minimum ξ-knot to the maximum ξ-knot of all functions with support on the marked element. Likewise for the meshline in the η-direction. The exact length of the two inserted meshlines are extracted by using the list of element to B-splines correspondence.

This strategy will make sure that all B-splines with support on the marked element (knot span) are treated equally and all of them will be split by the refinement. However, the drawback of this strategy is that one get a somewhat large footprint. Moreover, the neighbouring elements will be split by a single line which will in essence double their aspect ratio. This is depicted in Figure 20a where one can clearly see the rectangular shaped neighbouring elements arising from this strategy.

4.2.2 Minimum span

This refinement strategy inserts a cross through the marked element center with the aim of making the refinement footprint as small as possible. Thus, we want the inserted meshlines to be as short as possible, but still splitting at least one B-spline. From the list of element to B-spline correspondence we may deduct which B-splines having the smallest ξ- and η-support. Note that this comes with the loss of uniqueness as there may be several B-splines with the same length of the ξ-span, but with different local origin, i.e. for the B-spline i and j we may have $\xi^i_{p+1} - \xi^i_0 = \xi^j_{p+1} - \xi^j_0$ but $\xi^i_0 \neq \xi^j_0$ (see Figure 18a - 18c). This local refinement strategy is depicted in Figure 20b where we have chosen to split only one of the several available B-splines. Due to the (in general) lack of uniqueness of which B-splines to split, we will herein propose to do a *random* choice of which B-spline to refine. This will cause properties such as symmetry to be lost.

4.2.3 Structured mesh

The idea of refining elements is a legacy from the finite element method where every inserted vertex would correspond to an additional degree of freedom. With LR B-splines this is not the case and as seen from the two previous schemes the required length of the inserted meshlines may vary from element to element. Another way of refining LR B-splines is identifying *B-splines* which needs to be refined as opposed to which *elements*. In the case of the synthetic diagonal refinement problem presented later, these are easily extracted as all functions along the diagonal that satisfy $\xi_i = \eta_i$ for $i = 0, ..., p+1$. However, for general isogeometric finite element computations we need a criteria to identify which B-spline to be refined. We propose the following definition

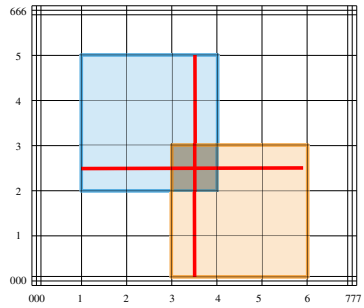
Definition 19. The **B-spline error** is the sum of element error over all supported elements, i.e.

$$\|e\|^2_{\mathcal{M}(N_i)} = \sum_{K \in \mathcal{M}(N_i)} \|e\|^2_K \tag{34}$$

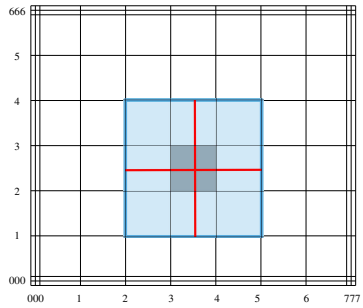
where $\mathcal{M}(N_i)$ is the set of elements on which the B-spline N_i is nonzero, and $\|e\|_K$ is the usual element error measured in energy norm,

$$\|e\|^2_K = a(u - u_h, u - u_h)_{\Omega_K} \tag{35}$$

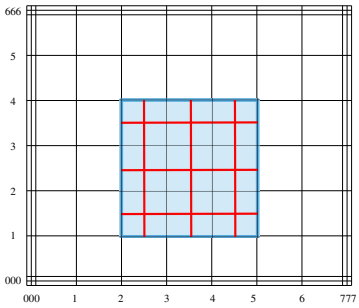
Once the B-splines which are subject to refinement are identified, we proceed to refine these by inserting a net of knot lines halving the largest supported knot intervals as shown in Figure 21.



(a) Full span - split all functions on one element, here only two of all the nine functions with support on this element is depicted

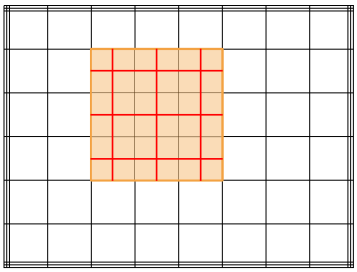


(b) Min span - split one random functions on one element, note that for odd order splines (or by a poor random pick), the symmetry would be lost

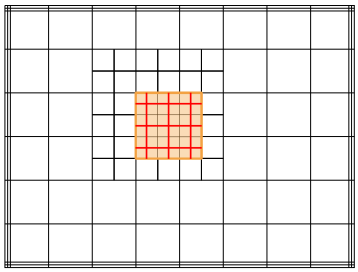


(c) Structured Mesh - split all knot spans on one B-spline, notice that no bad aspect ratio elements are created

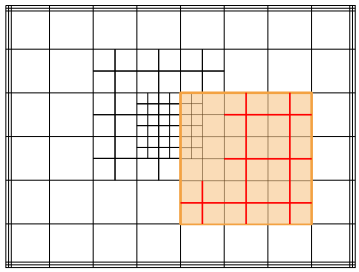
Figure 20: The ideas behind the different refinement strategies, here illustrated on a quadratic tensor product mesh. Notice the fundamental difference in that 20a - 20b is refining an element, while 20c is refining a B-spline.



(a) Iteration 1



(b) Iteration 2



(c) Iteration 3

Figure 21: Three iterations of an example structured mesh refinement. Notice that we at each iteration halve the largest supported elements. A selection of LR B-splines over the mesh from iteration 3 is depicted in Figure 22

4.2.4 Regularity

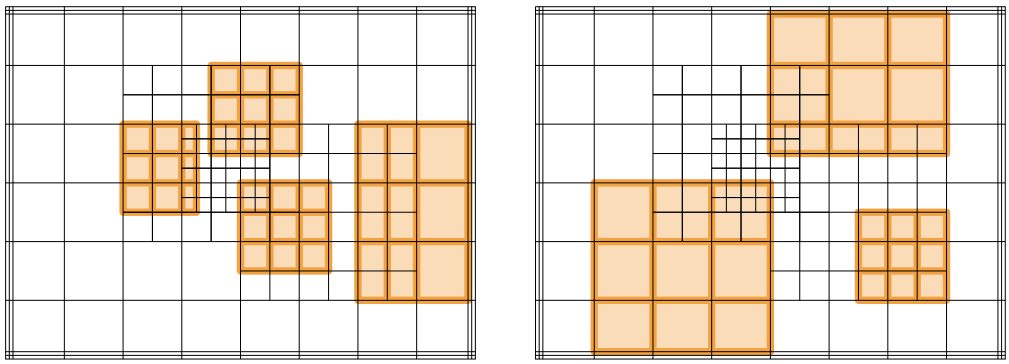
We note that just like tensor product B-splines, LR B-splines also allow for duplicate knots. The effects of duplicate knots is twofold. Firstly it reduces the regularity, such that a knot of multiplicity m will give rise to a C^{p-m} function across that knot, where p is the polynomial degree. In addition to the decreased regularity, one also decreases the support of the function. This will in turn diminish the propagation effects of the refinement. We will investigate refinement using different regularities.

4.3 Hanging nodes in FEM versus LR B-splines

Adaptive refinement of classical quadrilateral (Lagrange) FE has been achieved by means of many different approaches, e.g: Subdivision of marked

- patch of elements into smaller elements with transition zones to contain C^0 continuity
- elements into four new elements using multipoint constraints to contain C^0 continuity
- elements into four new elements using transition elements to contain C^0 continuity

To give some insight into the developed local refinement strategy using LR B-splines we will below illustrate how it for $p = 1$ compares to the the concept of using transition elements for adaptive refinement of FE grid. The comparison is chosen for a basic refinement case and we would like to emphasize that adaptive refinement using LR B-splines is in general more versatile than the concept of using transition elements.



(a) (b)

Figure 22: Some example quadratic LR B-splines over the LR mesh from Figure 21c

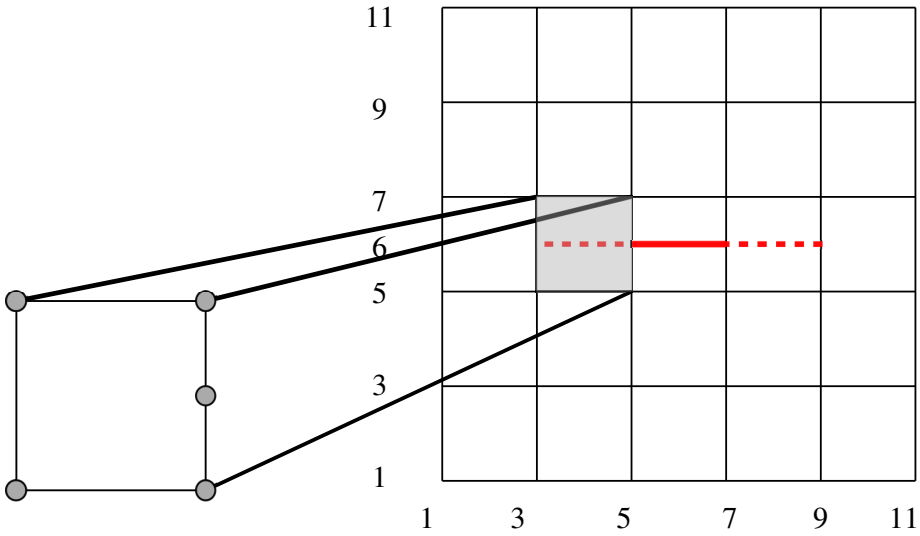


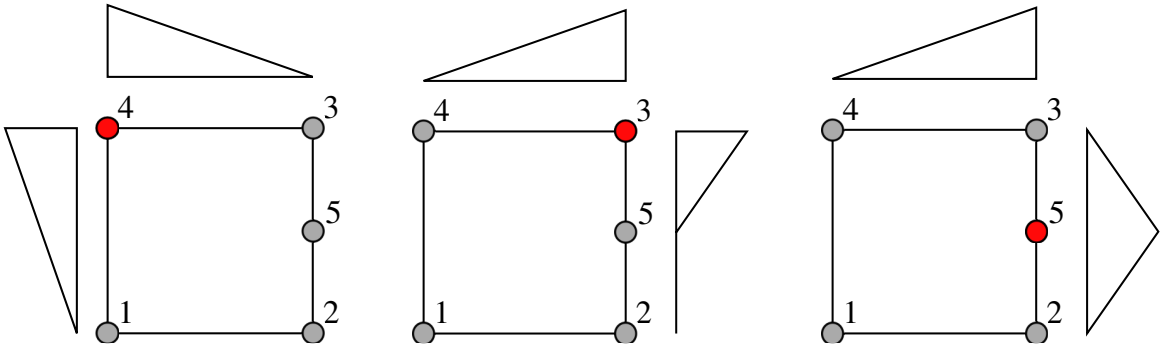
Figure 23: The transition element concept: Horizontal split of the centre element into two new elements.

4.3.1 Refinement for $p = 1$

Assume that we want to divide the centre FE element in the grid shown in Figure 23 in two elements by a horizontal split[1]. The concept of using transition elements then implies that we to the left and right of the centre element introduce 5-noded transition elements, see Hughes [10]. In Figure 24 we have displayed the element nodal shape functions for three of the five nodes. For node 1 and 4 the nodal shape functions for the 5-noded transition element are identical to those for node 1 and 4 for the standard bilinear 4-noded quadrilateral. However, for node 2 and 3 we have to modify the element nodal shape functions compared to the 4-noded in order to achieve that the shape function in node i evaluated at node j is either equal to 1 if $i = j$ or equal to 0 if $i \neq j$. Finally the element shape function for the new inserted node 5 is as displayed in Figure 24c. It is easy to verify that C^0-continuity is attained for the refined FE grid displayed in Figure 24.

To achieve the same refinement using LR B-splines we would insert a horizontal meshline as displayed in Figure 25. The nodal basis functions corresponding to those in Figure 24a–24c are displayed in the Figures 26–28, respectively. The leftmost column is showing an alternative way of plotting LR B-splines and is to be understood in the following way. Each continuity reduction line i.e. the LR meshlines, is plotted. For each B-spline we plot an ellipse with center at the Greville point (average value of the local knot vector) and with a size corresponding to the size of the support of that particular B-spline.

[1] A horizonal split is chosen for simplicity instead of a cross, but the this comparison apply for a cross as well.



(a) Element nodal shape function for element node 4 is unchanged

(b) Element nodal shape function for node 3 is modified

(c) Element nodal shape function for the inserted node 5

Figure 24: The transition element concept: The 5-noded transition element and its element nodal shape functions.

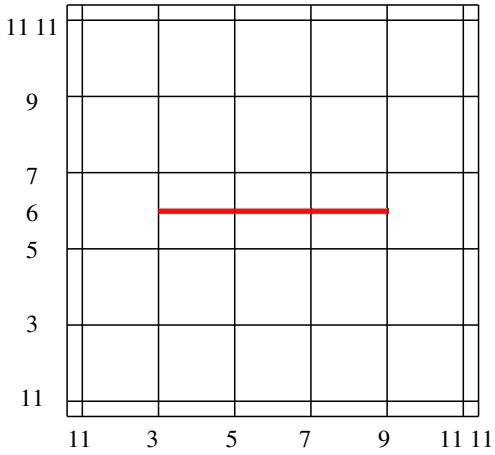
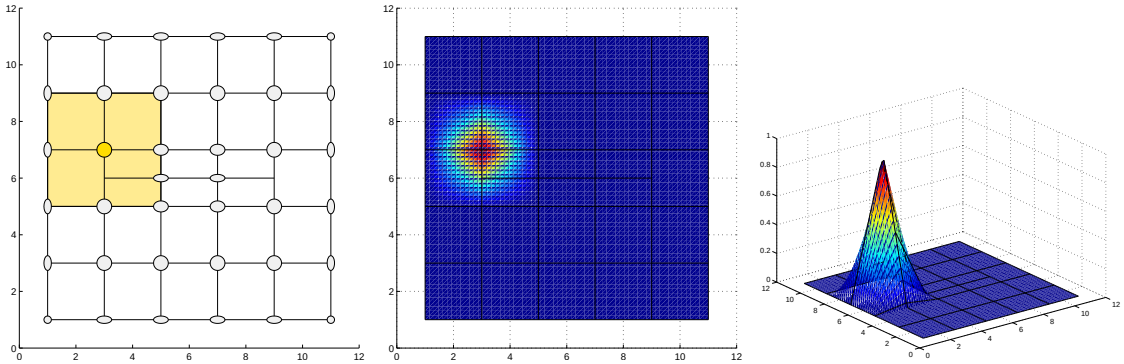


Figure 25: LR B-spline refinement: Inserting a horizontal meshline in order to to split the centre element into two new elements.

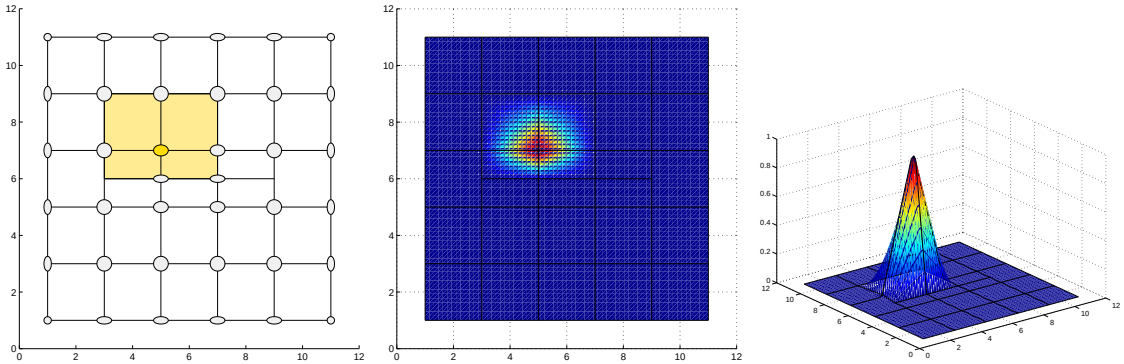
Furthermore, we have shaded the ellipse for the particular B-spline which is shown, as well as the support of that function.

The discrete function space after refinement is identical for the FE and LR B-spline case. The only difference is that we in the LR B-spline case get three more elements, whereas for the FE case we only get one new element. However, notice that we in Figure 23 have stipulated the transition elements to indicate that one should treat them as two elements when performing numerical integration. The reason being that the element nodal shape functions 2, 3, and 5, are not infinitely smooth across the stipulated line (they are only C^0). Thus, in practice we need to do the same amount of work related to numerical integration for both the FE and LR B-spline case.



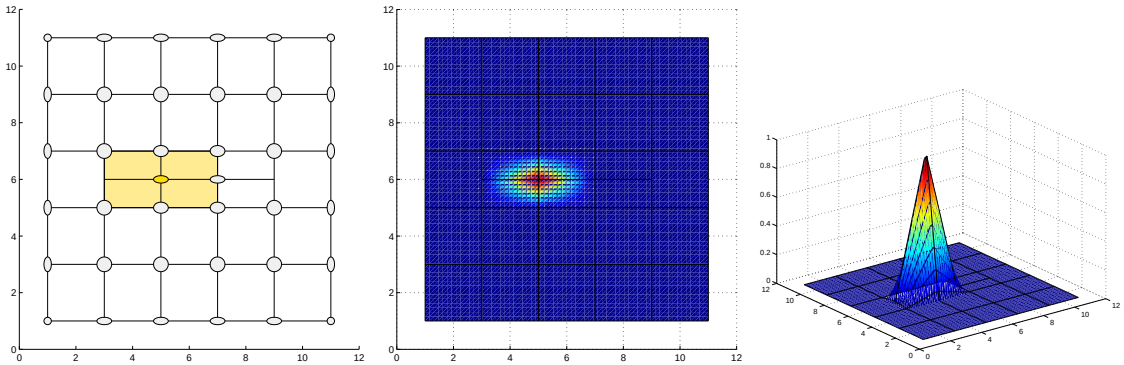
(a) Function space and support (b) Top view (c) Perspective view

Figure 26: B-spline $B[135; 579]$.



(a) Function space and support (b) Top view (c) Perspective view

Figure 27: B-spline $B[357; 679]$.



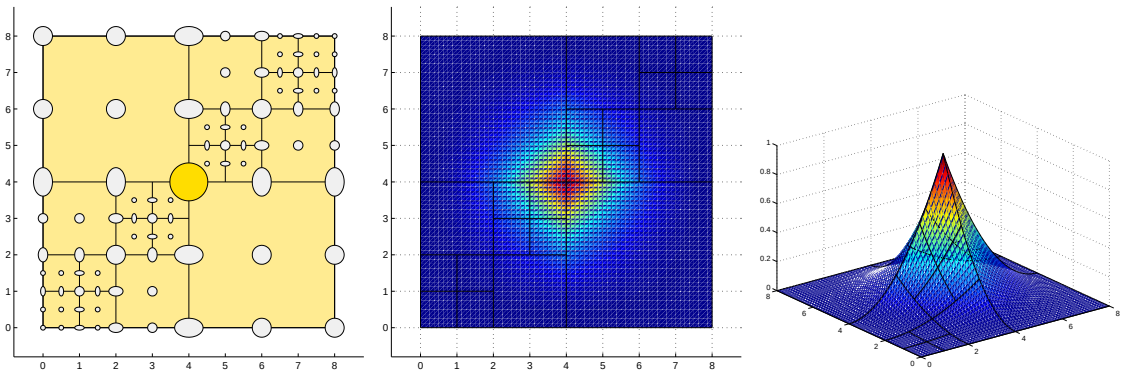
(a) Function space and support (b) Top view (c) Perspective view

Figure 28: B-spline $B[357; 567]$.

4.3.2 C^0-Refinement for $p = 2$

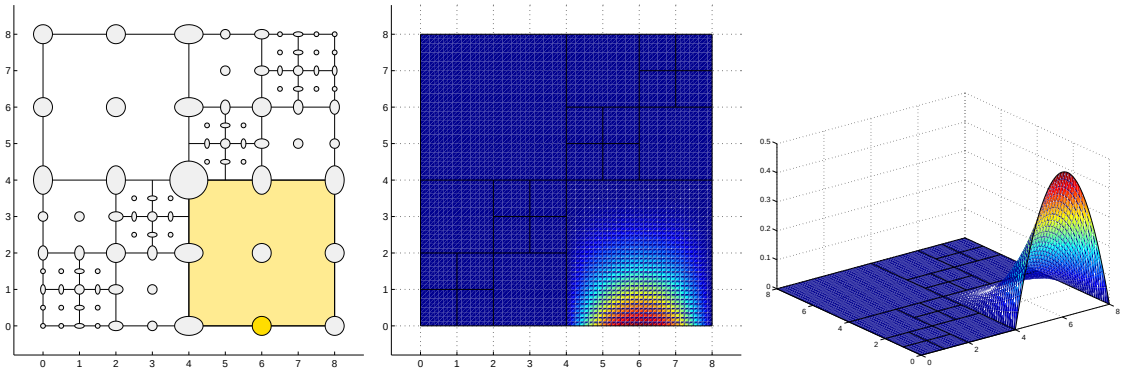
To make a comparison between classical Lagrange functions and LR splines with $p = 2$, we must consider C^0 elements. This is perfectly possible by using double knot lines, which is done here. The example mesh is taken from the diagonal refinement case for $p = 2$ and $m = 2$ which is going to be discussed in the Section 5.2. We see that the Greville points for $p = 2$ do in fact line up with the traditional way of drawing Lagrangian biquadratic finite element nodes. We have the usual 9 nodes for each element, provided that there are no hanging nodes nearby. Note however that the basis functions themselves are different. LR B-splines are non-negative, while Lagrange functions do take negative values. Nevertheless, they have the same support, and we see that around the hanging nodes, the basis functions vanish. This is equivalent to setting multipoint-constraints on these nodes, effectively removing them as a degree of freedom. This is can be seen in Figure 29 - 33, where several of the B-splines are shown. Also note that there is no upper bound on the number of hanging nodes on a single element, as several elements have 2 hanging nodes in this example.

It is interesting to see that one might recreate the Lagrangian function space, also with hanging nodes. However it is important to note that this is something that we in general will not try to do. Doing this causes us to loose the smoothness which is characteristic of all spline spaces, and this is a property which we would like to preserve.



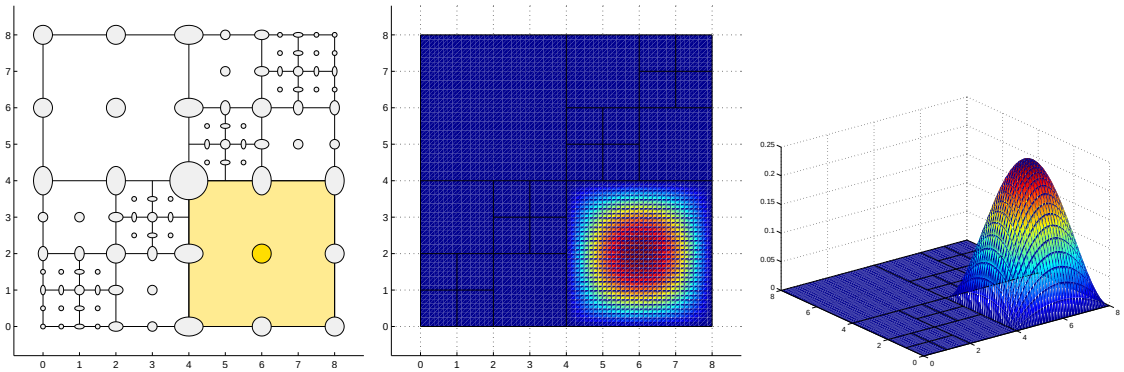
(a) Function space and support (b) Top view (c) Perspective view

Figure 29: B-spline $B[0448; 0448]$



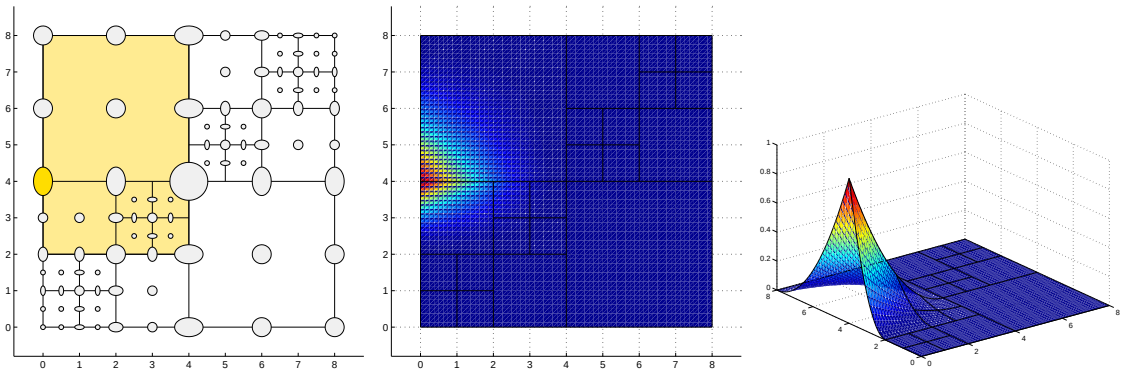
(a) Function space and support (b) Top view (c) Perspective view

Figure 30: B-spline $B[4488; 0004]$



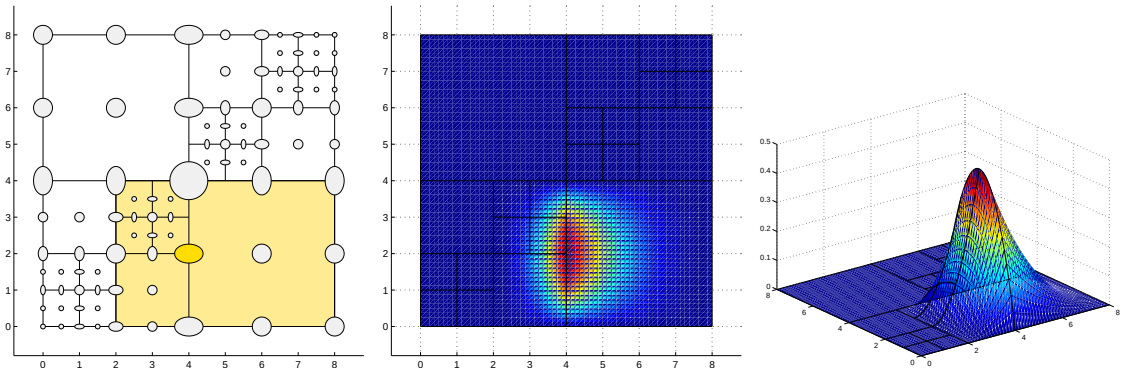
(a) function space and support (b) Top view (c) Perspective view

Figure 31: B-spline $B[4488; 0044]$



(a) function space and support (b) Top view (c) Perspective view

Figure 32: B-spline $B[0224; 2448]$



(a) function space and support (b) Top view (c) Perspective view

Figure 33: B-spline $B[2448; 0044]$

5 Numerical Results

5.1 Preliminaries

To demonstrate the performance of adaptive refinement using LR B-splines, we study one synthetic case and two Poisson type problems with known analytic solutions. The synthetic case denoted *Diagonal Refinement* is chosen as it illustrates very well the spreading effect of the refinement schemes and has been addressed by other researchers in the isogeometric community ([5, 7, 13]). The first Poisson example denoted *L-shape* is chosen as it has a point singularity at the boundary that causes reduced convergence rate when performing uniform refinement. Whereas the next Poisson problem *Interior Layer* has a rough right hand side that impose a sharp layer along a circular arc in the interior of the domain. The asymptotic convergence rate is here suboptimal for uniform refinement until the uniform element size h becomes smaller than a threshold given by the width of the interior layer.

The aim of the numerical experiments herein is to investigate whether adaptive refinement using LR B-splines achieves optimal convergence rate for non-smooth problems such that it gives better accuracy per dof compared to uniform refinement. Thus, the adaptive strategy is based on refining a prescribed portion of the the elements, i.e. $\beta \cdot n_{\text{el}}$ having the greatest elemental contribution, ρ_e to the global error, ρ in order to achieve uniform element error distribution. Furthermore, we want to investigate the sensitivity in accuracy and convergence rates towards relevant parameters e.g. polynomial order p, regularity C^r and local refinement strategies.

All the cases are analysed with LR B-splines of polynomial order $p = 2, 3, 4$. We have performed the tests with different regularity, C^r, were $0 \leq r \leq p-1$, obtained by using multiple knot lines. The multiplicity $m = 1$ corresponds to maximum regularity $r = p - m = p - 1$ whereas $m = p$ corresponds to minimum regularity $r = p - m = 0$. We have used two different local refinement strategies denoted *full span* and *structured mesh*.

For the synthetic case *Diagonal Refinement* we present the following results:

- Refined grids: Representative examples of refined grids
- Tables showing the number of dofs and elements for each refinement step
- Graphs showing the relation between number of elements and number of dofs

For the two Poisson cases (*L-shape* and *Interior Layer*) we present the following results:

- Convergence plot: Log of relative error vs log of number of degrees of freedoms (n_{dof})
- Refined grids: Representative examples of refined grids
- Error distribution: Elemental contribution, ρ_e, to the total error ρ
- Root mean square error of the element error distribution

The exact error $e = u - u_h$ is measured in the energy norm (a-norm) $||e||_E$ as given in Equation (22). Let $||e||_{E(\Omega)}$ and $||e||_{E(\Omega_e)}$ be the global and element error, respectively. Then we define the root mean square of exact element error:

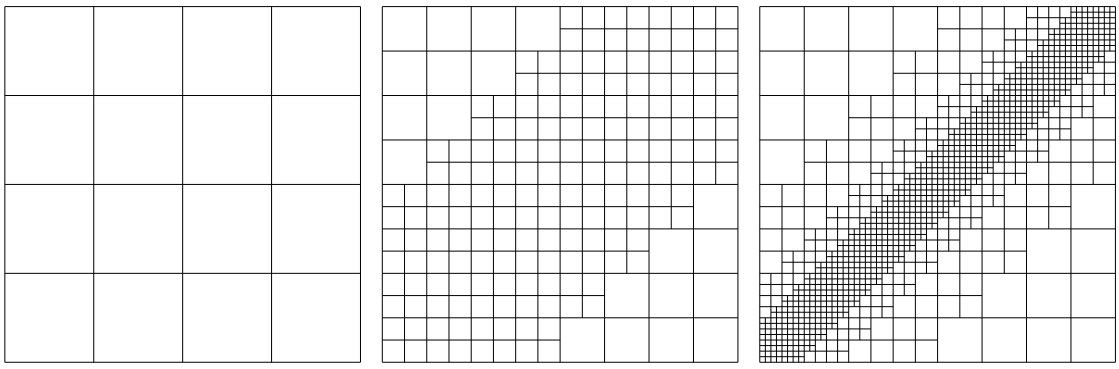
$$||e||_{\text{RMS}} = \left(\frac{1}{n_{\text{el}}} \sum_{e=1}^{n_{\text{el}}} (||e||_{E(\Omega_e)} - ||e||_{\text{avg}})^2 \right)^{1/2} / ||e||_{\text{avg}} \tag{36}$$

where the average exact element error is defined as

$$||e||_{\text{avg}} = \frac{1}{n_{\text{el}}} \sum_{e=1}^{n_{\text{el}}} ||e||_{E(\Omega_e)} \tag{37}$$

The quantity root mean square of exact element error given in Equation (36) measures the deviation from an uniform element error distribution. For uniform element error distribution we have $||e||_{\text{RMS}} = 0$. Thus, we refer to *asymptotically optimal mesh refinement* (see Kvamsdal and Okstad [12]) as a sequence of meshes satisfying

$$\lim_{h \to 0} ||e||_{\text{RMS}} = 0 \tag{38}$$



(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 34: The Diagonal Refinement problem: *Full span* and *structured mesh* refinement strategy using single knot lines and bicubic B-splines. Note that in the special case of diagonal refinement, these strategies coincide completely

5.2 Diagonal refinement

As an introductory example we look at the diagonal refinement. This example highlights some of the problems that local refinement strategies face since the request for refinement is in conflict with the parametrization direction. With the parametrization being parallel to the coordinate axes, and the diagonal 45 degrees on this, one will have to refine both parametric directions equally. Dörfel *et al.* [7] showed that under T-spline refinement this could provoke a worst-case scenario where the mesh lines propagate through the entire domain.

We will here present three different refinement strategies for this particular problem and analyse the resulting spline space resulting from these. The starting point for all of these is the assumption that we have identified a set of elements which needs refinement (here: the diagonal elements) and proceed to refine these using one of three strategies.

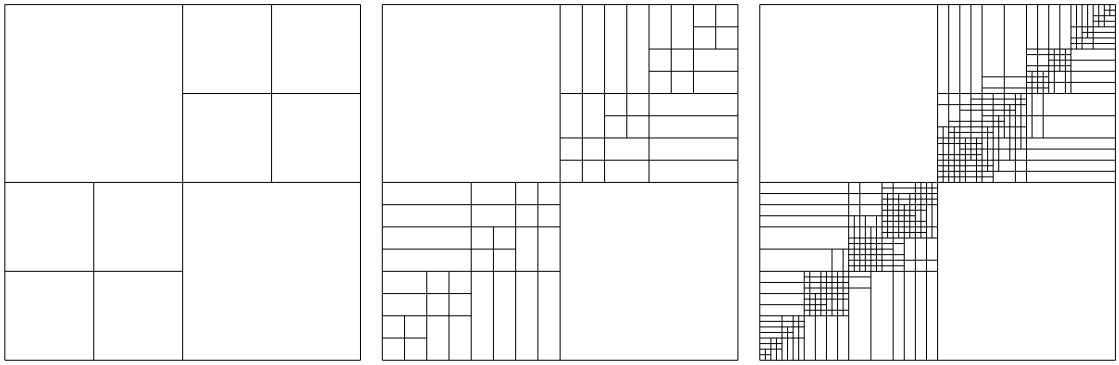
As discussed already, every inserted mesh line must at least span the support of at least one basis function. Thus it is in general not possible to only insert a single cross through one element when refining. If the inserted knot lines are limited to that element, then they will in general not be long enough to traverse the entire support of a basis function.

5.2.1 Results

Several refinement strategies was tested to see their performance on this benchmark test for local refinement. The setup was using $p = 3$ in each parametric direction and trying to refine the elements along the diagonal. No a priori knowledge on the problem was used as the input was just a given set of elements to be refined on a general LR B-spline. The first strategy that we tested was the *full span* strategy which for all tagged elements, chooses to refine all B-splines with support on that element. To accomplish this, we need a rather long mesh line in both ξ- and η-direction. Thus the characteristic propagation effect is rather large as can be seen in Figure 34 where several steps of the refinement process have been illustrated. Even if the propagation is clearly apparent, the refinement is still very much contained in a band along the diagonal and global refinement is avoided.

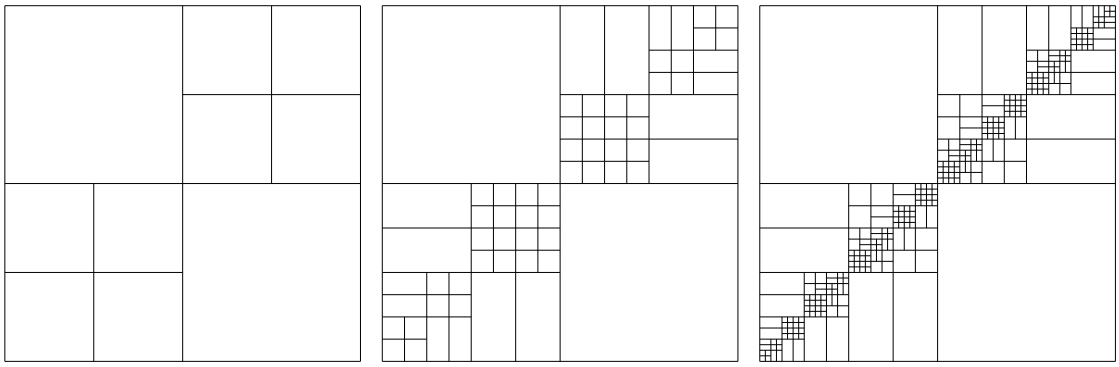
The diagonal test case is very exceptional in the sense that bad aspect ratio elements that are characteristic of the *full span* refinement are all canceled out by the next level on the diagonal. Thus the *full span* strategy produces identical meshes as the *structured mesh* strategy. However, the differences between these two strategies become apparent when they are used in an adaptive refinement process as shown in the next two subsections.

The final option is the *minimum span* refinement. Due to the fact that this picks a random function we introduce stochastic effects in our refinement strategy and things such as symmetry is in general lost. It



(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 35: The Diagonal Refinement problem: *Minimum span* refinement strategy using single knot lines.



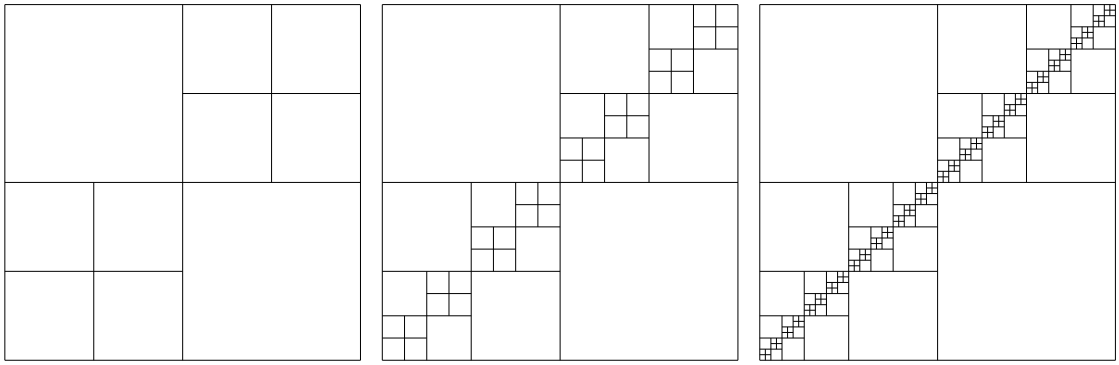
(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 36: The Diagonal Refinement problem: *Minimum span* refinement using double knot lines.

does however turn out to be more local than in the previous two cases. The results of a series of iterations using this refinement strategy is plotted in Figure 35. Note that after the first refinement, not a single line is added to the top left and bottom right portion of the mesh. This is due to the interpolatory basis functions at the edge which only span one element. Since the algorithm compares meshline lengths, it will always favor crossing zero-span elements such as the ones located at the edges.

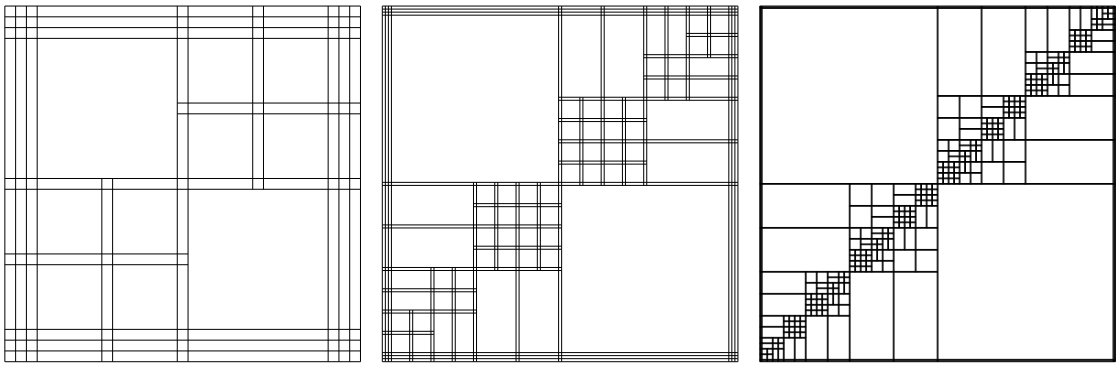
Although the latter refinement strategy does indeed reduce the effect of propagation, it is still apparent. Inspired by similar results for T-splines [5], we now test duplicate knot lines. The effect of splitting the elements using both double and triple knot lines is here shown in Figure 36 and 37. The results are quite promising as the triple knot line insertion removes *all* propagation into neighbouring elements. It is kept perfectly local and results in a very good mesh. One thing to keep in mind though when inserting duplicate knots is that each B-spline is splitted into more than two new B-splines. This results in a larger growth of the total number of B-splines than when using single knot lines. So even if the mesh seems tighter, or more compact, Figure 37 contains more B-splines than the mesh in Figure 35.

The corresponding *index* mesh to Figure 36-37 is given in Figure 38-39. Notice the high aspect ratio of some of the elements in Figure 35 and to a lesser degree in 36. This is a side effect which happens when inserted knot lines are traversing neighbouring elements and is not restricted to the element being refined. Of course, it is possible to combat this effect by recursively inserting more lines to compensate for this aspect ratio, but that would be in contrast with what we are trying to achieve here, which is to keep the refinement as local as possible.



(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 37: The Diagonal Refinement problem: *Minimum span* refinement using triple knot lines.



(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 38: The Diagonal Refinement problem: Index domain for *minimum refinement* using double knot lines.

5.2.2 Degrees of freedom versus elements

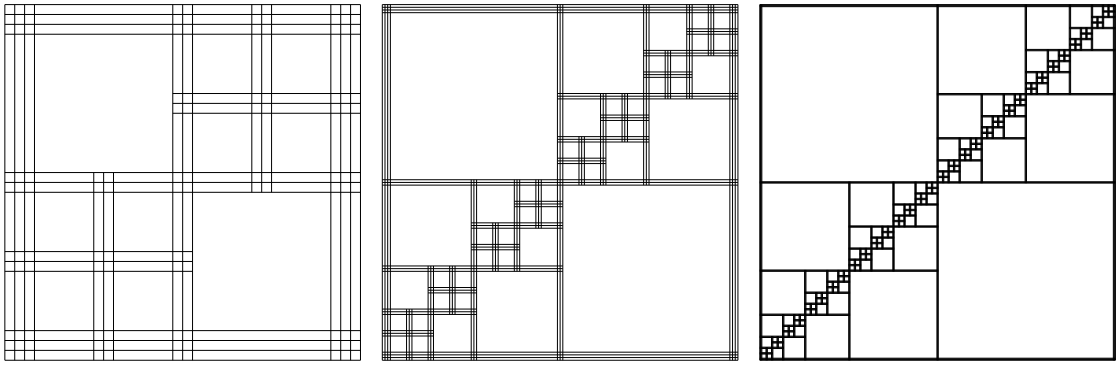
From the numerical experiments we observe that the number of basis functions (i.e. dofs) versus number of elements varies significantly with the regularity. If we compare the mesh using single knot line refinement (C^2-refinement) in Figure 35 with the mesh using triple knot lines refinement (or C^0-refinement) in Figure 37 we get the numbers displayed in the Table 5. The most refined C^0-mesh is containing approximately 5 times as many degrees of freedom and *less* than half the number of elements when compared with the most refined C^2-mesh.

We may take a deeper look into exactly how much this effect is apparent by plotting the number of basis functions and elements for our diagonal refinement case. We compare the minimum span refinement using single, double and triple knot lines. The results are given in Table 5. We see a clear tendency that the C^0 refinement keeps above 7:1 ratio between the number of degrees of freedom and the number of elements, whereas for the C2-refinement the ratio is dropping below 1:1 for the most refined grid.

For any univariate C^r-regular B-spline basis we note that n_{el} elements gives

$$n_{\text{dof}} = n_{\text{el}}(p - r) + r - 1 \tag{39}$$

where p is the polynomial order and the knot multiplicity is $m = p - r$. For a tensor product spline with n_{el}^2 elements it is clear that this gives $n_{dof}^2 = ((p-r)n_{\text{el}})^2 + \mathcal{O}(n_{\text{el}})$. For our particular (*minimum span*) case we have $p = 3$ and $r = 0, 1, 2$ and it seems reasonable that n_{dof} is approximately 8 times larger than n_{el} for C^0 and a factor 0.6 for C^2 at the most refined grid. We see that LR B-splines shows a somewhat similar growth of basis function to elements as regular tensor product B-splines does. This is shown in Figure 40 where the tabulated values are plotted.



(a) 1st refinement (b) 3rd refinement (c) 5th refinement

Figure 39: The Diagonal Refinement problem: Index domain for *minimum refinement* using triple knot lines.

Table 4: The Diagonal Refinement problem: Number of elements vs degrees of freedom using the *full span* strategy.

refinement count	C^2-elements	C^2-DOFs	C^1-elements	C^1-DOFs	C^0-elements	C^0-DOFs
1	4	25	4	36	4	49
2	16	49	16	100	16	169
3	64	121	46	220	46	439
4	196	253	112	452	112	1009
5	496	505	250	908	250	2179
6	1132	997	532	1812	532	4549
7	2440	1969	1102	3612	1102	9319

For the solution of stationary problems such as the ones considered in this paper this doesn't have too many implications. However, for the solution of a time-dependent non-linear elasticity problem, where the global coefficient (stiffness) matrix must be assembled for each iteration, this might be a drawback. The cost of numerical integration (by Gaussian quadrature) is dominated by the number of Gaussian integration points and hence the number of elements. Due to this huge discrepancy between the number of dofs and the number of elements, one might argue that measuring convergence rates and running time should no longer be plotted as a function of dofs, but rather as a function of elements. For our purposes however we note that the bottleneck is still the solution of the linear system of equations, and hence we keep the convergence plots with dof along the x-axis.

Table 5: The Diagonal Refinement problem: Number of elements vs degrees of freedom using the *minimum span* strategy.

refinement count	C^2-elements	C^2-DOFs	C^1-elements	C^1-DOFs	C^0-elements	C^0-DOFs
1	4	25	4	36	4	49
2	10	31	10	60	10	103
3	26	47	26	124	22	199
4	66	87	66	248	46	379
5	198	191	150	484	94	727
6	506	364	325	956	190	1411
7	1215	747	682	1904	382	2767

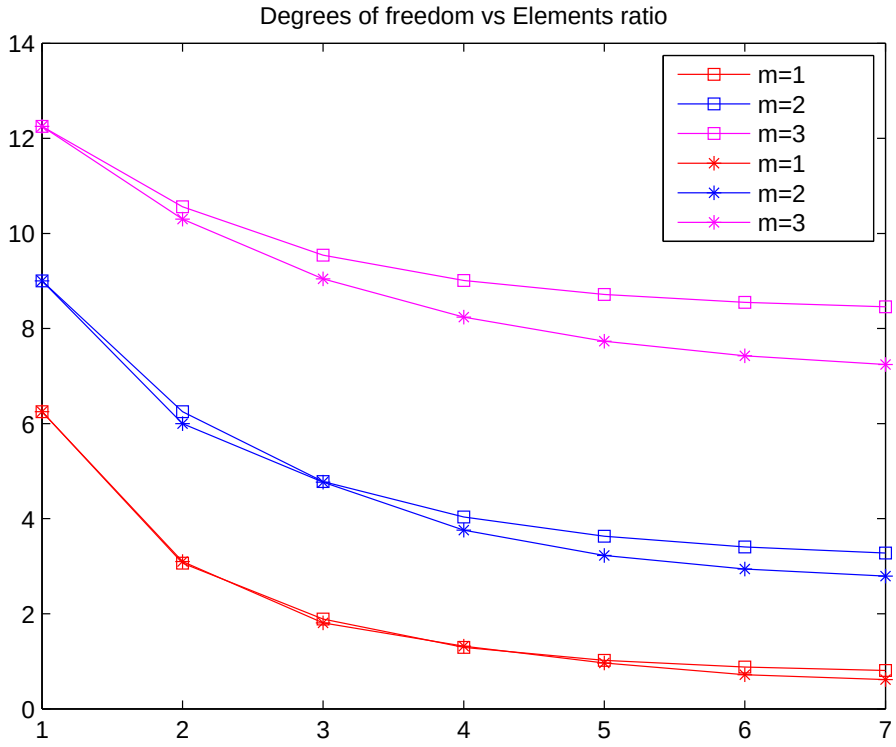


Figure 40: The Diagonal Refinement problem: Ratio of degrees of freedom versus elements using both *full span* (squares) and *minimum span* (stars) local refinement strategy. For tensor product bicubic splines, we have the asymptotic limit of 9 for $m = 3$, 4 for $m = 2$ and 1 for $m = 1$, see Equation (39).

5.3 L-shape

5.3.1 Problem definition

The problem consist of solving the stationary heat equation, or Laplace equation $\nabla^2 u = 0$ on a L-shaped domain $\Omega = [-1,1]^2 \setminus [0,1]^2$ with appropriate boundary conditions, see Figure 41.

$$\begin{aligned} \nabla^2 u &= 0 \quad \text{in} \quad \Omega \\ u &= 0 \quad \text{on} \quad \partial\Omega_D \\ \tfrac{\partial u}{\partial n} &= g \quad \text{on} \quad \partial\Omega_N \end{aligned} \tag{40}$$

with $g(x,y)$ given by the exact solution at the Neumann edge and $\boldsymbol{n}$ being an outward unit normal. It can be shown that

$$u_{ex}(r,\theta) = r^{2/3} \sin\left(\frac{2\theta+\pi}{3}\right) \tag{41}$$

is a solution to the Laplace equation $\nabla^2 u = 0$, and this is what we will be using as our analytical solution. The generation of g is straightforward from u_{ex} but is not given as a simple expression and the details are omitted here. The homogeneous Dirichlet boundary condition is given at $y = 0, x \in [0,1]$ and $x = 0, y \in [0,1]$, while all other edges are given with Neumann conditions (see Figure 41a). Note that the exact solution, which is pictured in Figure 41b exhibit a singularity at the origin. The function has a sharp edge at that point, and the derivative is not well defined there.

In Figure 42 we see that the convergence for uniform mesh refinement is limited by the strength of the singularity, i.e. the convergence rate is equal to $-q/2 = -1/3$. For problems where the solution is not sufficiently smooth, $u \notin H^{p+1}$, as the L-shape with a singular point on its boundary, we do not obtain optimal convergence. In particular, the use of high order polynomials is then inefficient.

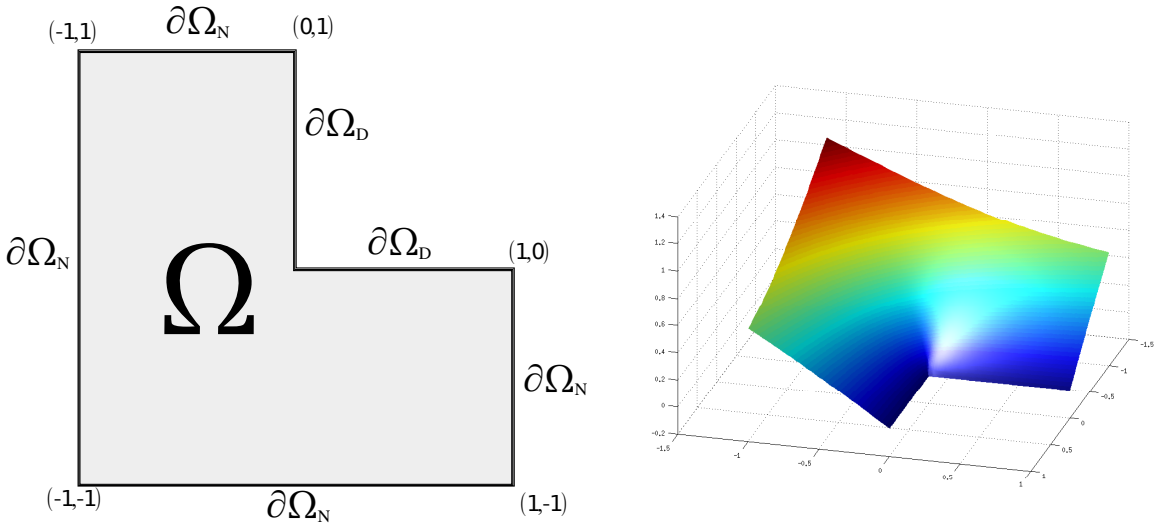
5.3.2 Results

In Figure 42 we show the results obtained by adaptive refinement using LR B-splines. The results are displayed using different polynomial order $p = \{2,3,4\}$, portion of refinement $\beta = \{5,20\}$, multiplicity/regularity $m = \{0, p-1\}$ and local refinement scheme.

The main observation is that we achieve optimal asymptotic convergence rate (valid for (quasi) uniform element error distribution), i.e. $-q/2 = -p/2 = \{-1, -3/2, -2\}$ for $p = \{2,3,4\}$, respectively.

We clearly see that high regularity (i.e. low multiplicity) is efficient when we compare relative error versus number of degrees of freedom. At first glance this seems odd, as the L-shape is one of the benchmarks for demonstrating the need for local refinement. Moreover, in the above example we concluded that the "perfect" local refinement was the one which introduced $p-1$ multiple knot lines as this did not propagate at all. However, introducing double knot lines will split each basis function into *three* new functions, as opposed to inserting a single knot line for splitting it into two. For triple knot lines this will of course split each function into four new ones. This means that the multiple knot line insertion actually gives a faster growth of the degrees of freedom. Furthermore, we see that the convergence results are more sensitive to the tested variation of local refinement strategies for higher polynomial order p and higher percentage of elements added in each refinement β. Notice that for $(p,m) = (4,1)$ the *structured mesh* refinement gives a higher error than for the *full span* refinement strategy. Furthermore, from our experiments we saw that for $\beta = 50$ we did not always obtain optimal convergence rate, i.e. the value of β should not be chosen too large.

The resulting grids are different for the different local refinement strategies. This is illustrated in the Figures 43–45 where we have displayed the effect on the refined grid using different local refinement strategies. As can be seen in the two Figures 43 and 44 the *full span* method have more elements with high aspect ratios than the *structured mesh* method, but the latter one give a more widespread stepwise uniform refinement towards the singularity point. However, when terminating at 3300 dofs, the two methods produce quite similar global energy error. In general, the two different refinement strategies are both able to refine sharply around the origin where the singularity appears.

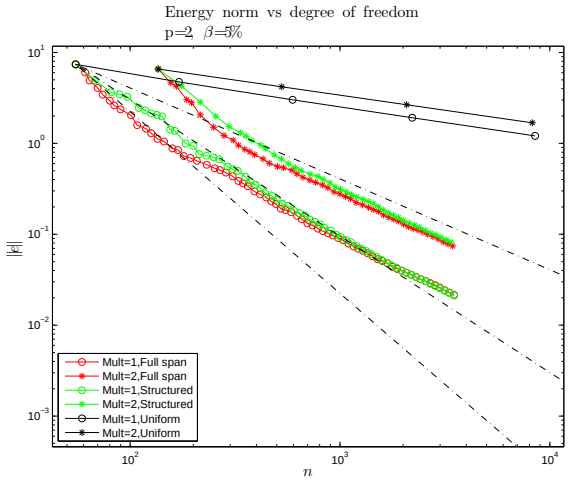


(a) The domain Ω and the boundary conditions

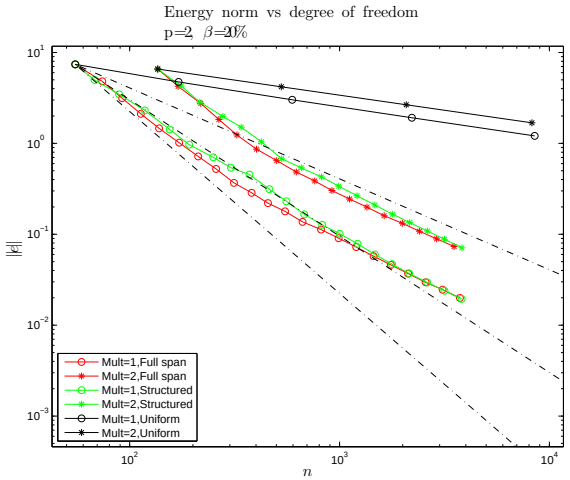
(b) The exact solution, see Equation (41)

Figure 41: The L-shape problem: A Poisson problem with a singularity point on the boundary.

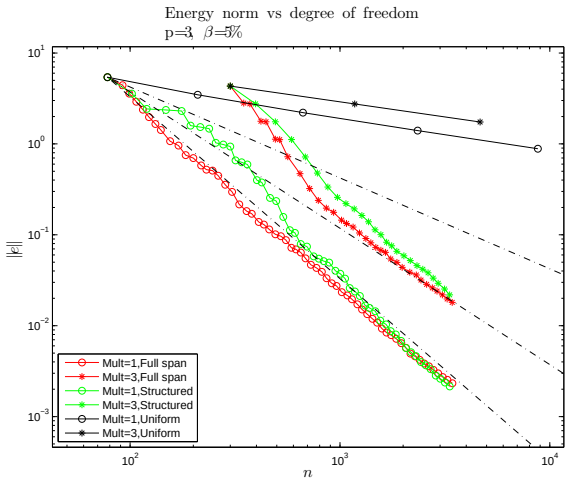
In Figure 46 we have displayed the root-mean square of the exact element error (in % measuring the deviation from uniform error distribution) versus n_{dof} obtained for uniform refinement using B-splines and adaptive refinement using LR B-splines. We see immediately that for the uniform refinement using B-splines the root mean square for the error distribution increases with number of uniform refinements. This is as expected as the error in the vicinity of the singularity point will be more and more dominant with uniform refinement. The highly non-uniform error distribution is consistent with the observed reduced convergence order in Figure 42. For the adaptive refined grids we see that for $p = 2$ the rms of the error distribution reduces in the first refinement steps and than becomes more or less constant $rms = 5 - 8 \cdot 10^{-1}$. As discussed in the paragraph above the lack of sufficient refinement around the singularity point prevent the rms to approach to zero. However, the values obtained may be classified as quasi-uniform, i.e. the rms is bounded when increasing n_{dof}. For $p = 3$ we see that the rms-values are a bit higher (around 1) and more differences between the different local refinement strategies. The results for $p = 4$ is even more spread and in particular for $\beta = 20$ we observe that the rms-values are slightly non-decreasing. This is consistent with the observation made above, i.e. that for $p = 4$ we get noticeable higher error for $\beta = 20$ than for $\beta = 5$, see Figure 42 f). Notice that the local refinement strategy *full span* have the lowest rms-value in all cases! Furthermore, that low multiplicity (i.e. $m = 1$) gives lower rms-values for *full span* than for high multiplicity, but this is not always the case for the *structured mesh* method.



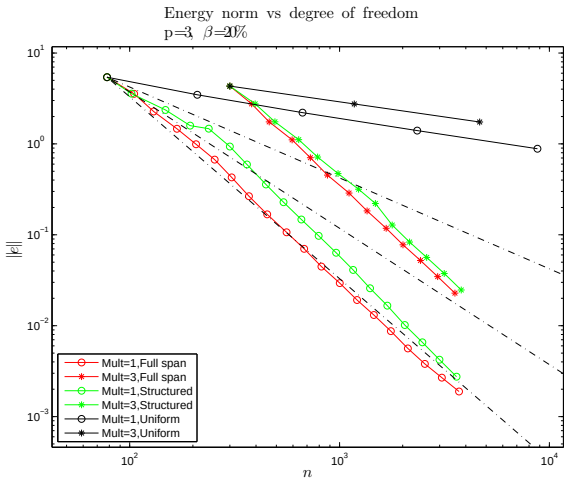
(a) $p = 2, \beta = 5$



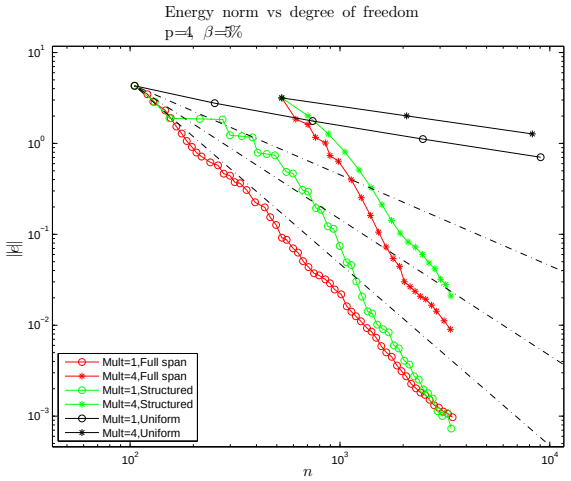
(b) $p = 2, \beta = 20$



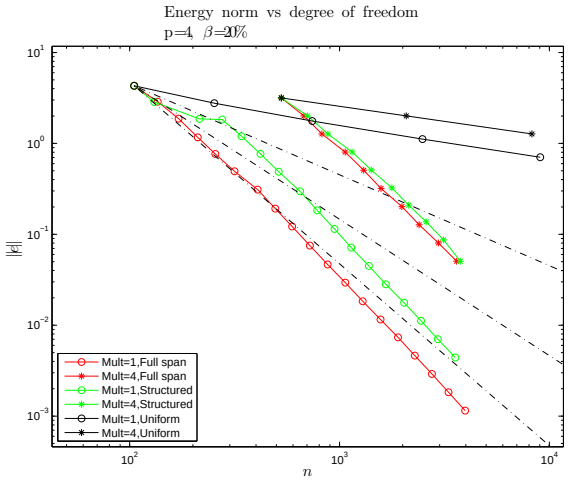
(c) $p = 3, \beta = 5$



(d) $p = 3, \beta = 20$

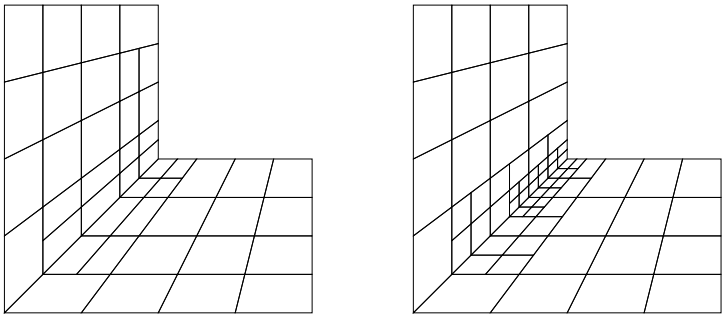


(e) $p = 4, \beta = 5$



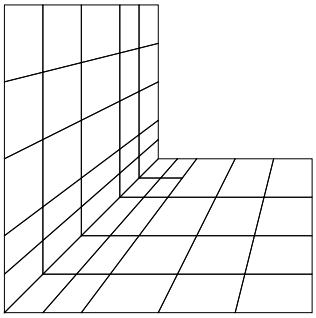
(f) $p = 4, \beta = 20$

Figure 42: The L-shape problem: Relative global errors (in %) (measured in the a-norm) versus n_{dof} obtained for uniform refinement using B-splines and adaptive refinements using LR B-splines. The dotted lines are the suboptimal convergence rate $\mathcal{O}(n_{\text{dof}}^{-1/3})$ valid for (quasi) uniform refinement and the optimal asymptotic convergence rates (valid for (quasi) uniform element error distribution) $\mathcal{O}(n_{\text{dof}}^{-1})$, $\mathcal{O}(n_{\text{dof}}^{-3/2})$, $\mathcal{O}(n_{\text{dof}}^{-2})$ for $p = \{2, 3, 4\}$, respectively.

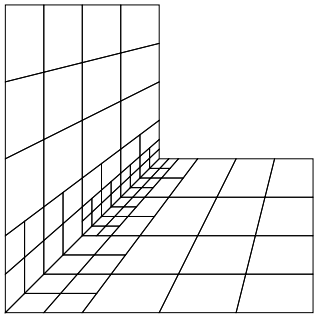


(a) (pmsb) = (2,1,0,5)

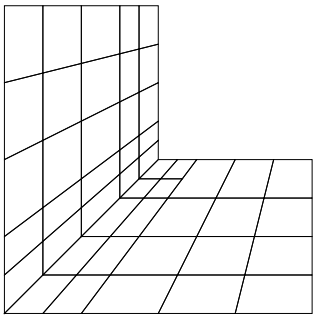
(b) (pmsb) = (2,1,2,5)



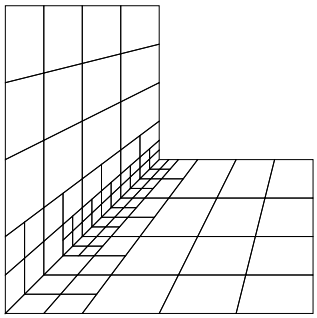
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)

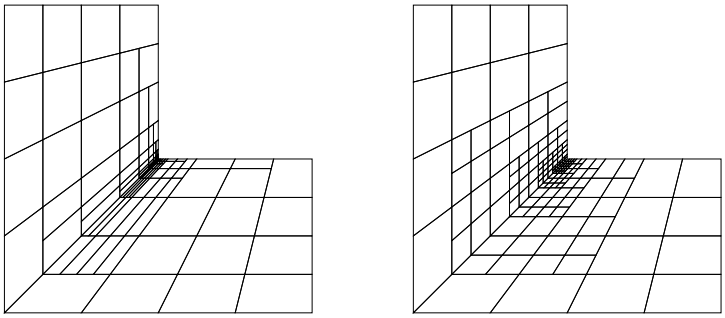


(e) (pmsb) = (4,1,0,5)



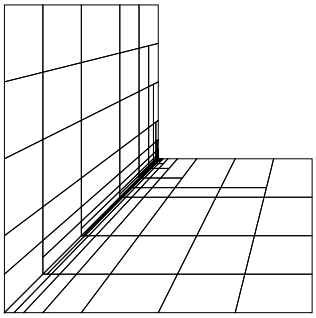
(f) (pmsb) = (4,1,2,5)

Figure 43: The L-shape problem: The 3rd adaptively refined grid $\mathcal{M}_3$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (notice that β is denoted b in the subfigure captions).

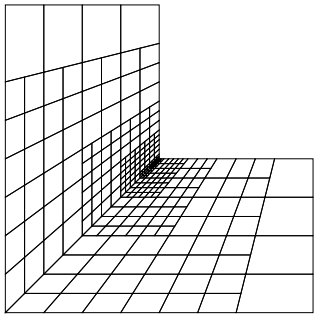


(a) (pmsb) = (2,1,0,5)

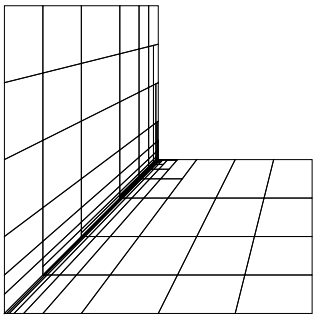
(b) (pmsb) = (2,1,2,5)



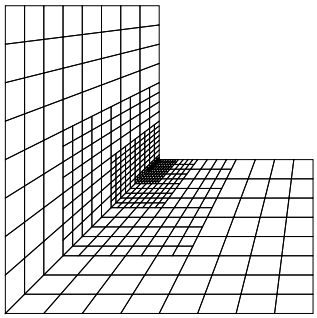
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)

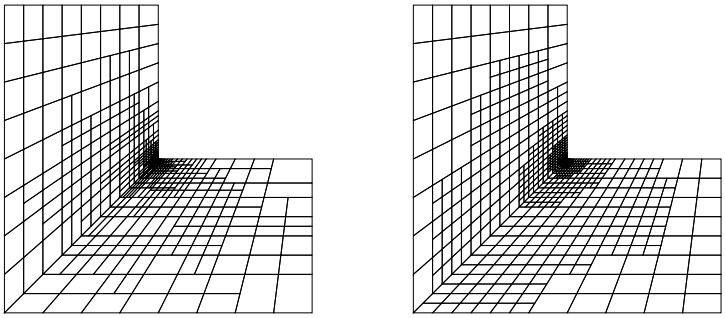


(e) (pmsb) = (4,1,0,5)



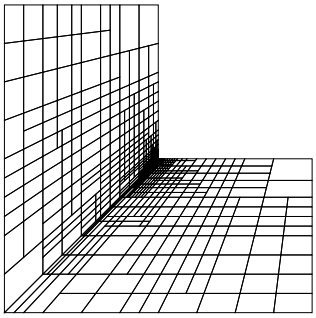
(f) (pmsb) = (4,1,2,5)

Figure 44: The L-shape problem: The 12th adaptively refined grid $\mathcal{M}_{12}$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (notice that β is denoted b in the subfigure captions).

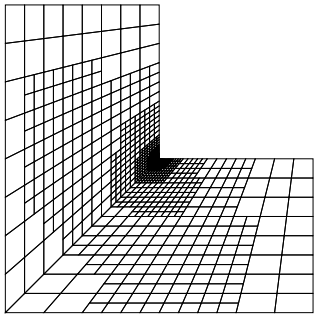


(a) (pmsb) = (2,1,0,5)

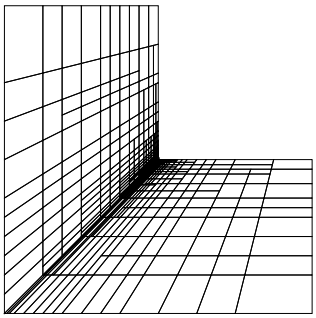
(b) (pmsb) = (2,1,2,5)



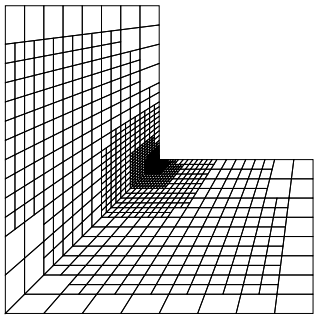
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)

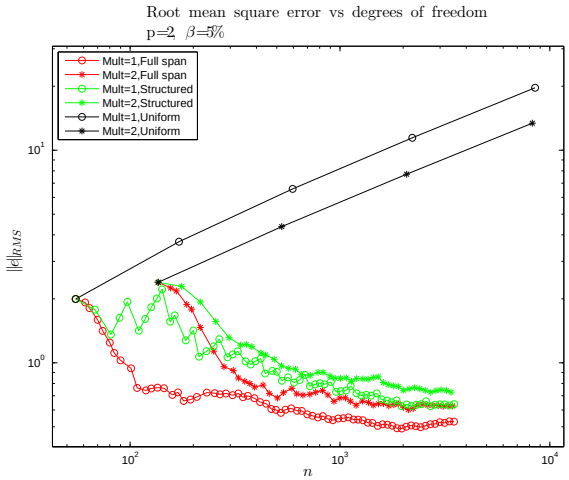


(e) (pmsb) = (4,1,0,5)

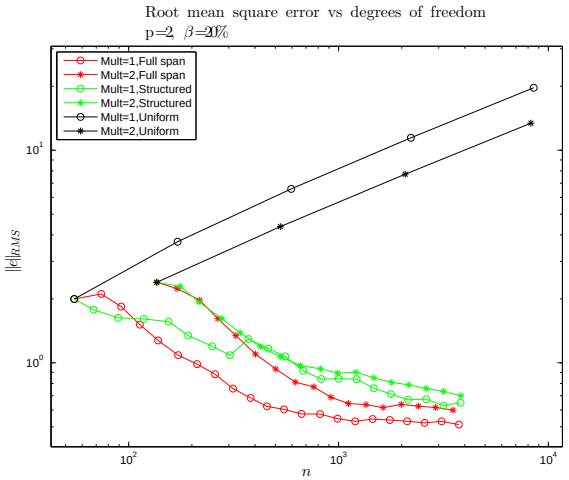


(f) (pmsb) = (4,1,2,5)

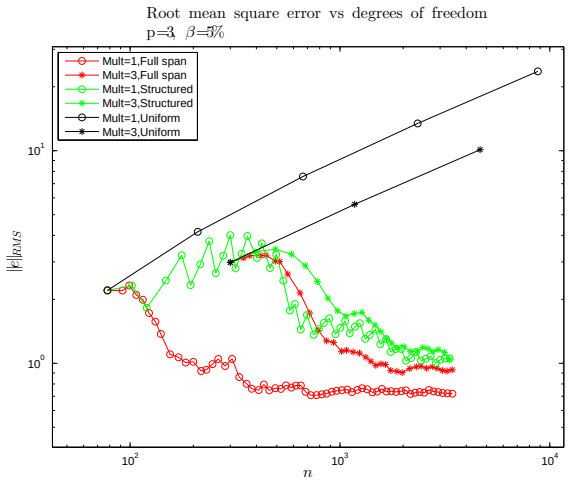
Figure 45: The L-shape problem: The final adaptively refined grid $\mathcal{M}_n$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (notice that β is denoted b in the subfigure captions).



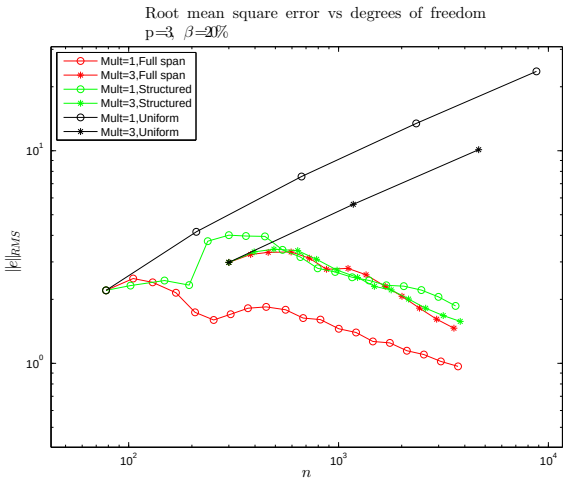
(a) $p = 2, \beta = 5$



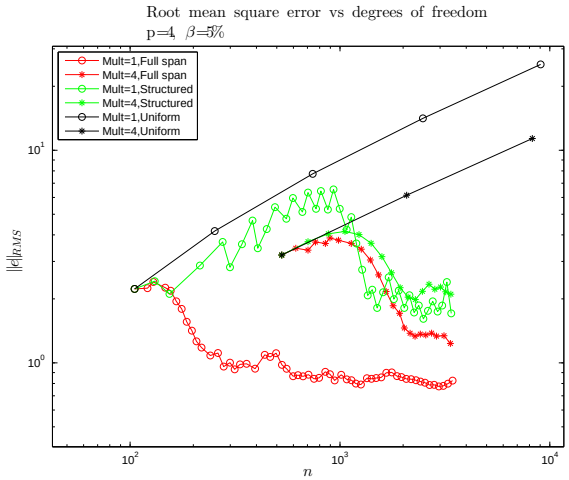
(b) $p = 2, \beta = 20$



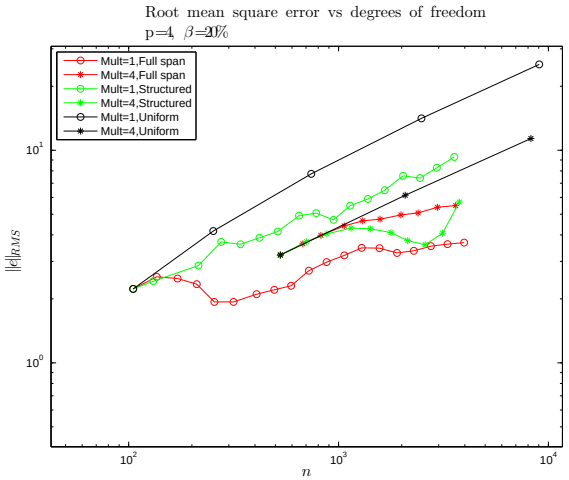
(c) $p = 3, \beta = 5$



(d) $p = 3, \beta = 20$



(e) $p = 4, \beta = 5$



(f) $p = 4, \beta = 20$

Figure 46: The L-shape problem: Root-mean square of exact element error (in %) (measuring the deviation from uniform error distribution) versus n_{dof} obtained for uniform refinement using B-splines and adaptive refinement using LR B-splines. Results displayed for $p = \{2, 3, 4\}$ (from top to bottom) and $\beta = \{5, 20\}$ (left to right).

5.4 Interior layer

5.4.1 Problem definition

The next test problem is a Poisson problem on a unit square with a sharp interior layer due to a highly varying right hand side (volumetric forcing). The problem is given as

$$\begin{array}{rll} \nabla^2 u &= f(x,y) & \text{in} \quad \Omega \\ u &= u_D(x,y) & \text{on} \quad \partial\Omega_D \\ \frac{\partial u}{\partial n} &= g(x,y) & \text{on} \quad \partial\Omega_N \end{array} \tag{42}$$

and has an exact solution given by

$$u(x,y) = \arctan\left(S(\sqrt{(x-1.25)^2+(y+0.25)^2} - \frac{\pi}{3})\right). \tag{43}$$

Note that the right hand side $f(x,y)$ is generated by taking the Laplacian (∇^2) of the analytical solution given in Equation (43) and similarly $g(x,y)$ is found by taking normal derivative, i.e. $\frac{\partial u}{\partial n}$, of the analytical solution. The analytical solution depicted in Figure 47b displays a "front"-type of behavior where the solution is rapidly changing across a circular band through the domain. This problem is mathematically smooth i.e. $u \in H^{p+1}(\Omega)$ for any finite p. However, due to the highly varying right hand side we may only expect optimal convergence order when the element size h is less than a given threshold that depends on the sharpness/bandwith of the interior layer. Hence, we may expect suboptimal convergence rate for uniform mesh refinement when the mesh is not fine enough.

In Figure 48 we see that the convergence for uniform mesh refinement is limited by the low regularity of the right hand side, i.e. the convergence rate is equal to $-q/2 = -1/2$. However, we see that for refined grids with small enough element size $h \lesssim 1/40$ (i.e. $n_{\text{dof}} \gtrsim 1600$) we obtain optimal convergence order. Thus, our refinement goal is here to resolve the interior layer as adequately as possible in order to obtain optimal convergence order, in an adaptive grid refinement process towards a global solution of a certain accuracy measured in the a-norm.

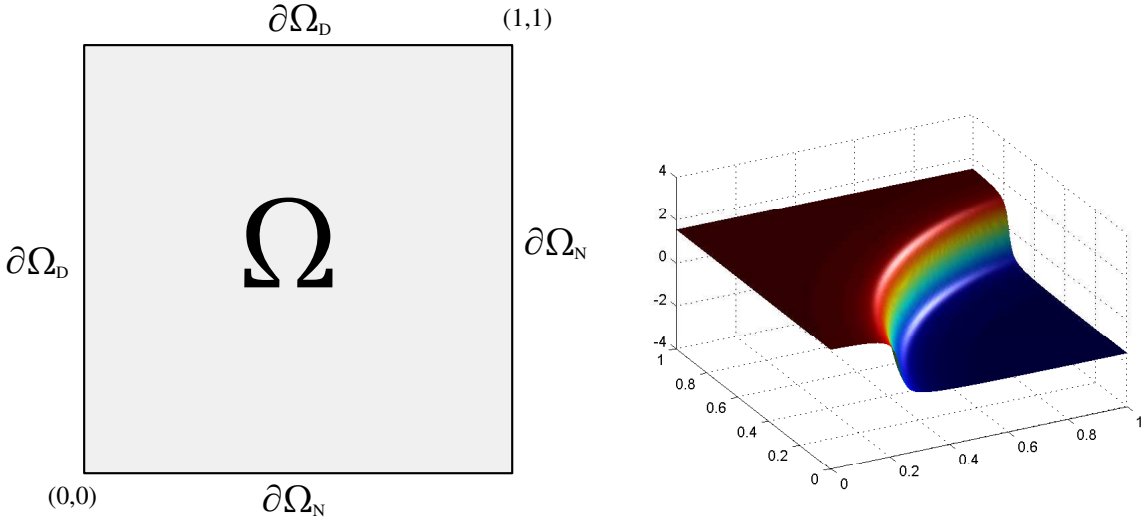
5.4.2 Results

In Figure 48 we show the results obtained by adaptive refinement using LR B-splines. The results are displayed using different polynomial order $p = \{2,3,4\}$, portion of refinement $\beta = \{5,10,20\}$, multiplicity $m = \{0, p-1\}$ and local refinement strategy.

The main observation is that we achieve optimal convergence rate, after some refinements, i.e. $-q/2 = -p/2 = \{-1, -3/2, -2\}$ for $p = \{2,3,4\}$, respectively. However, we see that for high polynomial order ($p = \{3,4\}$) we need more refinements than for low order to obtain optimal convergence rate. Furthermore, for $p = 4$ we observe some “extra” refinement along the Dirichlet boundary due to the fact that $u \notin S^h$. Compared to uniform refinement the errors for adaptive refined meshes using LR B-splines are about 10 times lower. The sharper the interior layer, the more pronounced this error difference will become.

We clearly see that high regularity (i.e. low multiplicity) is efficient when we compare relative error versus number of degrees of freedom. Furthermore, we see that the convergence results are not sensitive to variation of β, whereas local refinement strategies have some influence for high polynomial order.

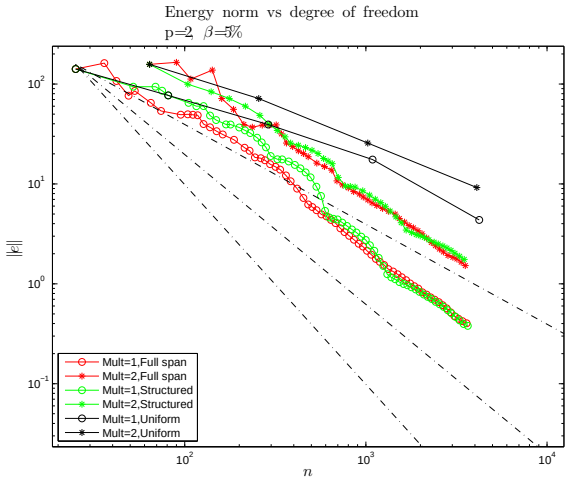
The resulting grids are different for the different local refinement strategies. This is illustrated in the Figures 49–51 where we have displayed the effect on the refined grid using different local refinement strategies. As seen, the LR B-splines makes it possible to refine sharply in the vicinity of the interior layer. Furthermore, we may see from the two Figures 49 and 50 that the *full span* method have more elements with high aspect ratios than the *structured mesh* method, whereas the latter one gives a more widespread uniform refinement on subdomains along the interior layer. In particular, the differences are pronounced at the $\mathcal{M}_3$ grid. However, at about $n_{dof} = 3000$ the two methods produce quite similar global energy error. Notice, that the final grid for $p = 4$ shows extra refinement along the Dirichlet part of the boundary due to the fact that the inhomogeneous Dirichlet boundary conditions are approximated.



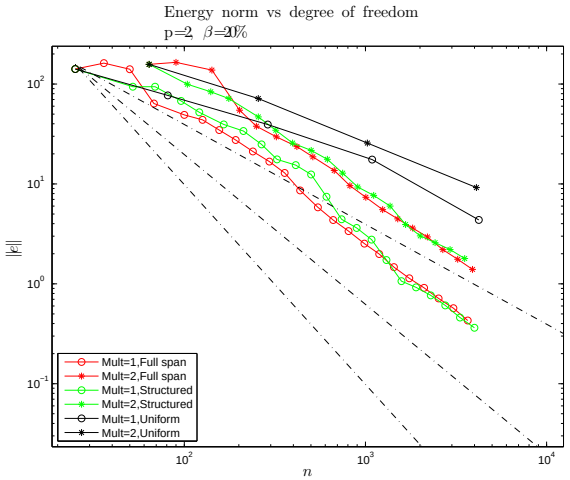
(a) The domain Ω and the boundary conditions

(b) The exact solution, see Equation (43)

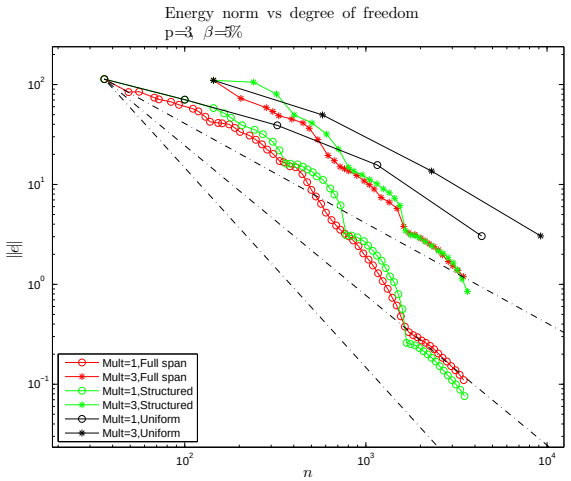
Figure 47: The Interior Layer problem: A Poisson problem with rough right hand side.



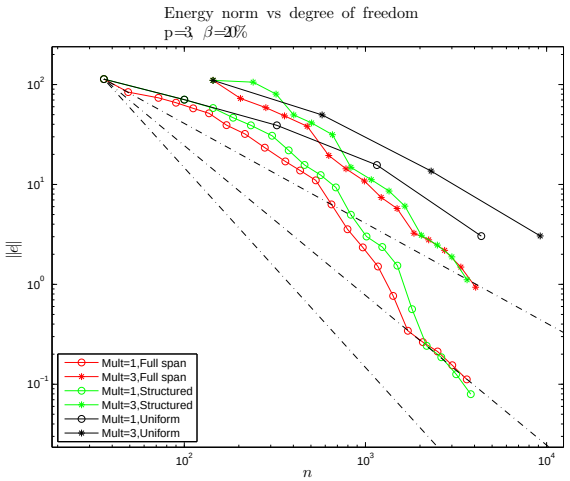
(a) $p = 2$, $\beta = 5$



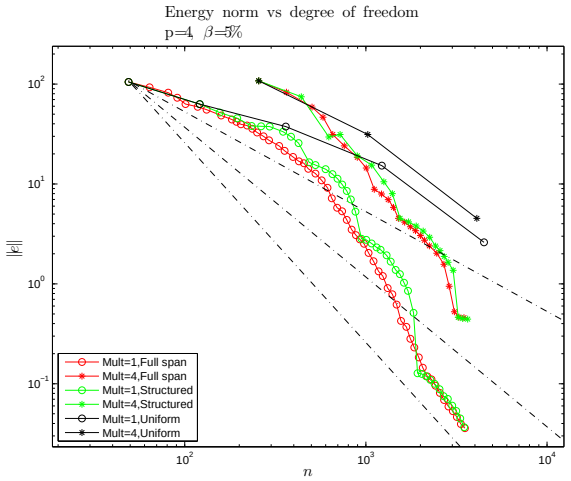
(b) $p = 2$, $\beta = 20$



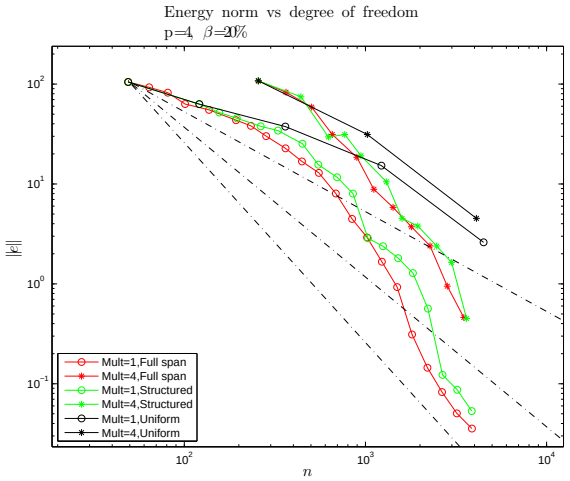
(c) $p = 3$, $\beta = 5$



(d) $p = 3$, $\beta = 20$

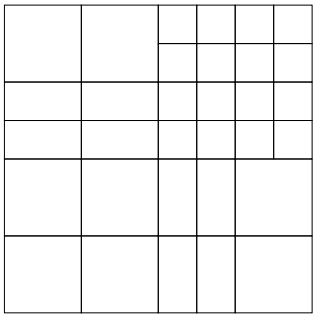


(e) $p = 4$, $\beta = 5$

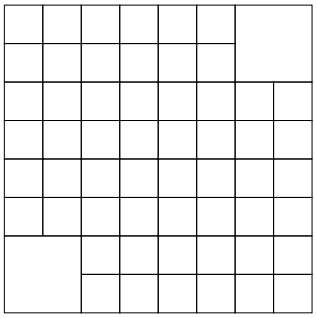


(f) $p = 4$, $\beta = 20$

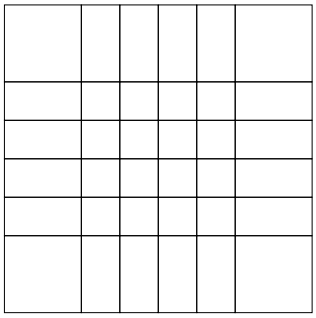
Figure 48: The Interior Layer problem: Relative global errors (in %) (measured in the a-norm) versus n_{dof} obtained for uniform refinement using B-splines and adaptive refinements using LR B-splines. The dotted lines are the optimal asymptotic convergence rates (valid for (quasi) uniform element error distribution) $\mathcal{O}(n_{\text{dof}}^{-1})$, $\mathcal{O}(n_{\text{dof}}^{-3/2})$, $\mathcal{O}(n_{\text{dof}}^{-2})$ for $p = \{2, 3, 4\}$, respectively.



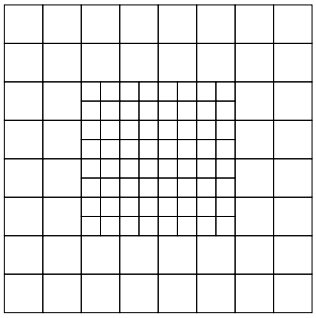
(a) (pmsb) = (2,1,0,5)



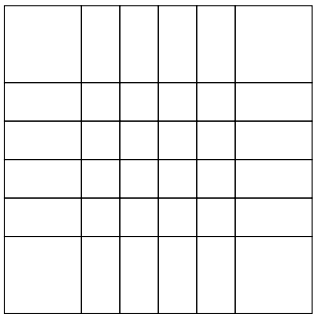
(b) (pmsb) = (2,1,2,5)



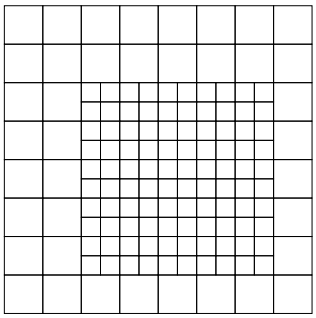
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)

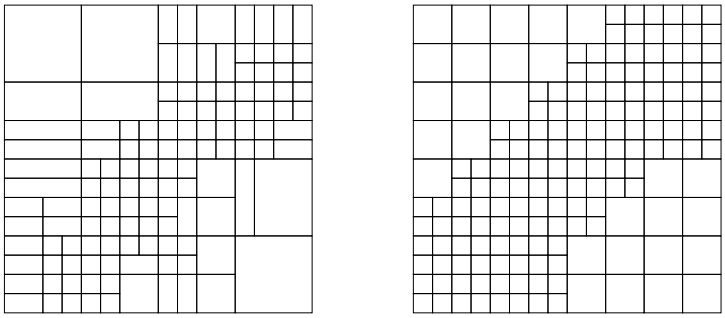


(e) (pmsb) = (4,1,0,5)



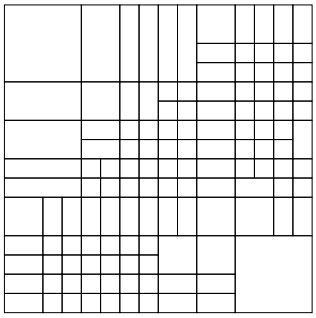
(f) (pmsb) = (4,1,2,5)

Figure 49: The Interior Layer problem: The 3rd adaptively refined grid $\mathcal{M}_3$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, , 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (β is denoted b in the subfigure captions).

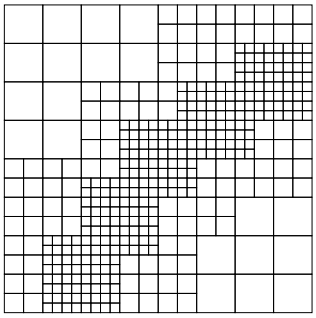


(a) (pmsb) = (2,1,0,5)

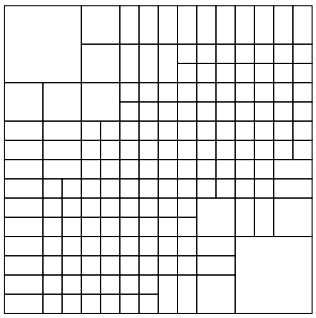
(b) (pmsb) = (2,1,2,5)



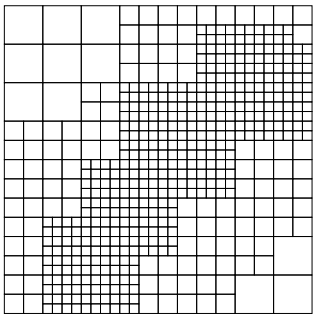
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)

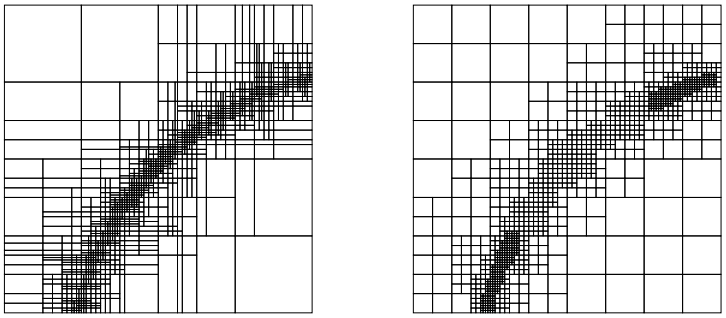


(e) (pmsb) = (4,1,0,5)



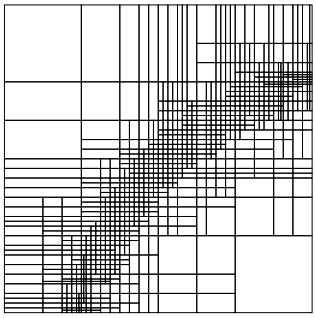
(f) (pmsb) = (4,1,2,5)

Figure 50: The Interior Layer problem: The 12th adaptively refined grid $\mathcal{M}_{12}$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (β is denoted b in the subfigure captions).

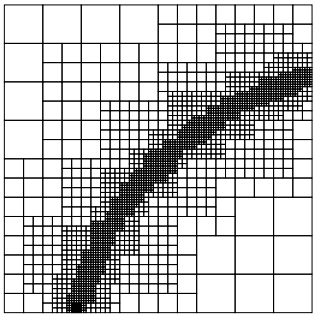


(a) (pmsb) = (2,1,0,5)

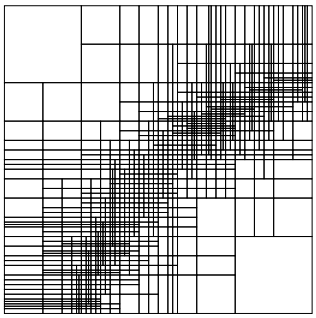
(b) (pmsb) = (2,1,2,5)



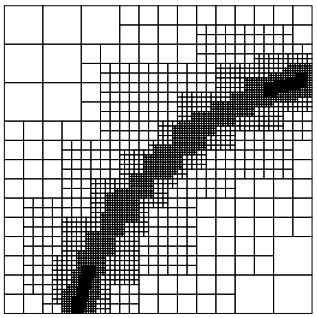
(c) (pmsb) = (3,1,0,5)



(d) (pmsb) = (3,1,2,5)



(e) (pmsb) = (4,1,0,5)



(f) (pmsb) = (4,1,2,5)

Figure 51: The Interior Layer problem: The final adaptively refined grid $\mathcal{M}_n$ obtained by using LR B-splines with different polynomial degrees $p = \{2, 3, 4\}$ and local refinement strategies, but same multiplicity $m = 1$ and $\beta = 5$ (β is denoted b in the subfigure captions).

6 Conclusions

In this paper we have investigated adaptive refinement in isogeometric analysis using LR B-splines. Traditional tensor product B-splines lack the ability of local refinement which is needed in order to achieve optimal convergence order in real world applications. In particular, higher order isogeometric methods based on tensor product B-splines are not able to exploit the full potential offered by isogeometric analysis when applied to problems involving singularities or rough right hand sides.

Herein, the newly developed LR B-splines have been applied as adaptive refinement in isogeometric analysis. Different local refinement strategies has been proposed and implemented in the object oriented code *IFEM*.

We have performed an extensive set of numerical tests to investigate the performance of using LR B-splines to achieve accurate results measured in energy norm (a-norm) and optimal convergence rates for the classical benchmark tests L-shape and Interior Layer. The results are very good and we achieved optimal convergence rates for both test cases, and the sensitivity towards different choices of local refinement strategies and prescribed portion of refinement was moderate. Furthermore, high regularity gives less error versus degrees of freedoms compared to low regularity (for a given polynomial order) in all cases.

We conjecture that the application of the full span and structured mesh refinement strategies, both generates a subclass of LR B-splines that are linearly independent, omitting the need for linear independence testing. No linearly dependent mesh has been discovered while using these strategies. The proof for this is left as a topic for future investigation.

We think the LR B-splines may serve well as a framework for adaptive isogeometric methods as they are both versatile and manageable with regards to development of general purpose finite element software. The framework open new vistas for interoperable CAD and FEA systems, and more research and developments should be pursued to fully exploit these possibilities.

7 Acknowledgement

The authors acknowledge the financial support from the Norwegian Research Council and the industrial partners of the ICADA-project; Ceetron, Det Norske Veritas and Statoil. They also acknowledge the support from the other co-workers in the ICADA-project. We thank in particular Dr Mukesh Kumar for many fruitful discussions and suggestions that have improved the manuscript.

References

[1] M. Ainsworth and J.T. Oden. *A Posterori Error Estimation in Finite Element Analysis*. Wiley-Interscience, 1st edition, January 2000.

[2] Y. Bazilevs, V. M. Calo, J. A. Cottrell, J. A. Evans, T.J.R. Hughes, S. Lipton, M. A. Scott, and T. W. Sederberg. Isogeometric analysis using T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):229–263, 2010.

[3] Y. Bazilevs, L. Beirao de Veiga, J.A. Cottrell, T.J.R. Hughes, and G. Sangalli. Isogeometric analysis: approximation, stability and error estimates for h-refined meshes. *Mathematical Models and Methods in Applied Sciences*, 16:1031–1090, 2006.

[4] P. B. Bornemann and F. Cirak. A subdivision-based implementation of the hierarchical b-spline finite element method. *Computer Methods in Applied Mechanics and Engineering*, 2012.

[5] A. Buffa, D. Cho, and M. Kumar. Characterization of T-splines with reduced continuity order on T-meshes. *Computer Methods in Applied Mechanics and Engineering*, (201-204):112–126, 2012.

[6] T. Dokken, T. Lyche, and K.F. Pettersen. Polynomial splines over locally refined box-partitions. *Comput. Aided Geom. Des.*, 30(3):331–356, March 2013.

[7] M. R. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with T-splines. *Computer Methods in Applied Mechanics and Engineering*, 199(5-8):264 – 275, 2010.

[8] D.R. Forsey and R.H. Bartels. Hierarchical B-spline refinement. *ACM SIGGRAPH Computer Graphics*, 22(4):205–212, 1988.

[9] C. Giannelli, B. Jüttler, and H. Speleers. THB-splines: The truncated basis for hierarchical splines. *Computer Aided Geometric Design*, 29(7):485 – 498, 2012.

[10] T.J.R. Hughes. *The finite element method: linear static and dynamic finite element analysis*. Prentice-hall, 1987.

[11] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39-41):4135–4195, October 2005.

[12] T. Kvamsdal and K. M. Okstad. Error estimation based on superconvergent patch recovery using statically admissable stress fields. *International Journal for Numerical Methods in Engineering*, 42:443–472, 1998.

[13] X. Li and M. A. Scott. On the nesting behavior of T-splines. Technical report, University of Texas at Austin, May 2011.

[14] X. Li, J. Zheng, T.W. Sederberg, T.J.R Hughes, and M.A. Scott. On linear independence of T-spline blending functions. *Comput. Aided Geom. Des.*, 29(1):63–76, January 2012.

[15] B. Mourrain. On the dimension of spline spaces on planar T-subdivisions. *Arxiv preprint arXiv:1011.1752*, November 2010.

[16] D. Schillinger, L. Dedé, M.A. Scott, J.A. Evans, M.J. Borden, E. Rank, and T.J.R. Hughes. An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces. *Computer Methods in Applied Mechanics and Engineering*, 249 - 252(0):116 – 150, 2012.

[17] D. Schillinger and E. Rank. An unfitted hp-adaptive finite element method based on hierarchical B-splines for interface problems of complex geometry. *Computer Methods in Applied Mechanics and Engineering*, 200(47 - 48):3358 – 3380, 2011.

[18] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes. Isogeometric finite element data structures based on Bézier extraction of T-splines. *International Journal for Numerical Methods in Engineering*, 88(2):126–156, 2011.

[19] M. A. Scott, X. Li, T. W. Sederberg, and T. J. R. Hughes. Local refinement of analysis-suitable T-splines. *Computer Methods in Applied Mechanics and Engineering*, 213:206–222, 2012.

[20] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. *ACM Transactions on Graphics*, 22(3):477–484, 2003.

[21] A.V. Vuong, C. Giannelli, B. Jüttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(49-52):3554–3567, 2011.

[22] P. Wang, J. Xu, J. Deng, and F. Chen. Adaptive isogeometric analysis using rational PHT-splines. *Computer-Aided Design*, 43:1438–1448, 2011.