On the similarities and differences between Classical Hierarchical, Truncated Hierarchical and LR B-splines

Kjetil André Johannessen, Filippo Remonato and Trond Kvamsdal
Department of Mathematical Sciences
Norwegian University of Science and Technology, Trondheim, Norway
e–mail: Kjetil.Johannessen@math.ntnu.no, Filippo.Remonato@math.ntnu.no,
Trond.Kvamsdal@math.ntnu.no

Abstract

Smooth spline functions such as B-splines and NURBS are already an established technology in the field of computer-aided design (CAD) and have in recent years been given a lot of attention from the computer-aided engineering (CAE) community. The advantages of local refinement are obvious for anyone working in either field, and as such, several approaches have been proposed. Among others, we find the three strategies *Classical Hierarchical B-splines*, *Truncated Hierarchical B-splines* and *Locally Refined B-splines*. We will in this paper present these three frameworks and highlight similarities and differences between them. In particular, we will look at the function space they span and the support of the basis functions. We will then analyse the corresponding stiffness and mass matrices in terms of sparsity patterns and conditioning numbers. We show that the basis in general do not span the same space, and that conditioning numbers are comparable. Moreover we show that the weighting needed by the Classical Hierarchical basis to maintain partition of unity has significant implications on the conditioning numbers.

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 differences 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.

In any finite element analysis of real world problems, it is of great importance that the quality of the computed solution may be determined. Furthermore, 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 a posteriori 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]) for an excellent overview. The senior author of the present paper have been working with error estimation and adaptivity for more than two decades (see e.g. [29], [36], [35], and [31]), and are well aware of the fact that adaptive methods are not yet an industrial tool, partly because the need for a link between the finite element program and traditional CAD-system. 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.

The new paradigm of Isogeometric Analysis, which was introduced by Hughes *et al.* [22] (see also [11]), demonstrates that for smooth (enough) problems much is to be gained with respect to efficiency, quality and accuracy in analysis by replacing traditional Finite Elements by volumetric NURBS elements. However, a fundamental constraint of traditional NURBS is that they are (global) tensor product and lack the potential for local refinement. The need for local refinement has always been an issue, and several proposed solutions to local refinement has been derived such as T-splines [42],[41], Hierarichal B-splines [15],[27], Truncated Hierarichal B-splines [18] and Locally Refined B-splines [13]. The use of these techniques in CAD allows for more freedom since it is often enough that a forward mapping exist and can be efficiently manipulated or evaluated, and some conversion algorithms are available [48]. With their applications into isogeometric analysis [4], [3], [14], [38], [5], [44], [45], [47], [34], [46], [6], [39], [23], [24], [28], and [43] came new requirements of the basis. Linear independence [10], [9], [40], [2], [30], [20], [8], stability [19] and partition of unity [18] became center topics of research as isogeometric analysis is impractical or sometimes impossible without these properties. The research is ongoing for most of these basis and the community has yet to settle on a single technology which encompass every desired property without restrictions on the mesh.

With many different technologies addressing the same problem of local refinement, it is to be expected that there is overlap. We hope to shed some light on this topic by presenting some of these spline families, highlighting similarities and differences between them. In particular we will be looking at the Classical Hierarchical, the Truncated Hierarchical and the Locally Refined B-splines.

We will during this paper compare a set of metrics on the different basis, and immediately comes the question of what is a fair comparison. While choices such as "the same number of basis functions" at first seem appropriate, this would depend on the particular refinement strategy (or how did you get to those basis functions), which are not always compatible across technologies. Instead we say that a fair comparison is to consider the different basis built on the *exact same mesh*. This will ensure that we study the basis itself and not the refinement parameters [23]. However we cannot include analysis suitable T-splines in this particular comparison, since they form a different set of admissible meshes; See Figure 1.

1.2 Aim and outline of the paper

The aim of this paper is to present several different local refinement strategies that currently exist. We will emphasize differences in both their mathematical and numerical properties.

The paper is organized as follows:

In Section 2, we give a brief introduction to the approximation theory. We describe the finite element method and least squares method with focus on derivation of the two matrices we will be looking at: the stiffness matrix A and the L_2 projection matrix M.

In Section 3, we define Hierarichal B-splines, Truncated Hierarichal B-splines and Locally Refined (LR) B-splines. Our aim is to provide a common framework and notation to better highlight their particular properties.

Numerical experiments are performed in Section 4. As our main measurement, we will be looking at conditioning number and sparsity pattern of the system matrices over several illustrative meshes.

We end this paper concluding upon our findings in Section 5.

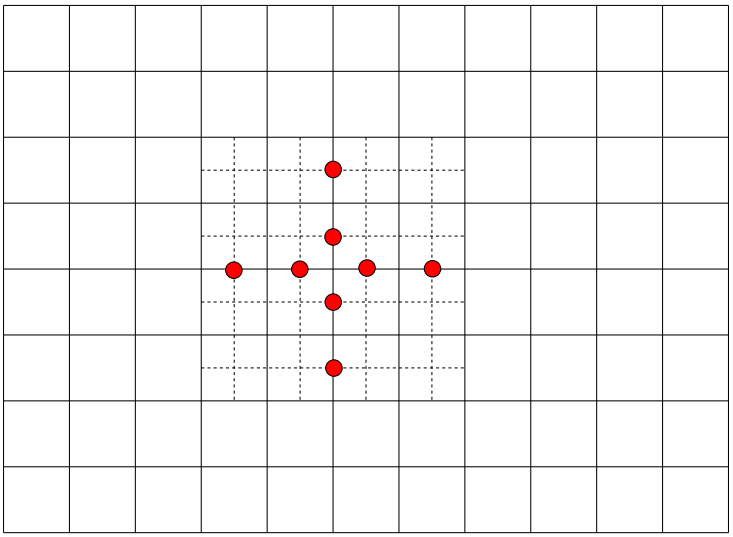


Figure 1: **T-splines:** The dotted lines are the extended T-mesh induced by the basis function highlighted in red. This is the corresponding T-mesh which would be equivalent to the refinement of one basis function of Hierarchical or LR B-splines. The mesh however, is not an analysis suitable T-spline since the meshline extensions intersect. We will in this paper not consider analysis suitable T-splines since they form a different set of admissible meshes than Hierarchical and LR B-splines does.

2 Finite Element Theory

In this section we give a very short introduction to the finite element theory, with the only aim of presenting the quantities we will be using as our performance indicators. A thorough explanation of the theory behind finite elements can be found in many sources like [25, 37, 7, 21].

One of the first steps when applying a finite element method to a problem, is to derive its so called *variational formulation* and write the problem in a structure like: Find $u \in \mathcal{V}$ such that

$$a(u, v) = l(v) \quad \forall v \in \mathcal{V} \tag{1}$$

where $\mathcal{V}$ is a Hilber space, $a(\cdot\,, \cdot)$ is a continuous bilinear forms, and $l(v)$ is a continuous functional on the dual space of $\mathcal{V}$. The problem is also normally associated with some type of conditions, like boundary or initial conditions. The existence and uniqueness of solutions is then guaranteed by the Lax-Milgram theorem.

The Galerkin approach to this kind of problems consists of producing a finite-dimensional approximation $\mathcal{V}_h$ of the infinite-dimensional function space $\mathcal{V}$, and search for solutions $u_h \in \mathcal{V}_h$. Specifically, we have $\mathcal{V}_h \subset \mathcal{V}$, and the problem reads: Find $u_h \in \mathcal{V}_h$ such that

$$a(u_h, v_h) = l(v_h) \quad \forall v_h \in \mathcal{V}_h\,. \tag{2}$$

The space $\mathcal{V}_h$ is defined to be the span of selected basis functions $\{\varphi_1, \varphi_2, \dots \varphi_n\}$. In the Isogeometric setting, these functions are chosen to be spline functions, for which we use the notation $\{B_1, B_2 \dots B_n\}$.

We now give some classical examples to introduce the stiffness and mass matrices.

2.1 Poisson equation

Poisson equation is the classical model problem for Elliptic PDEs. It arises in several engineering problems like elastic membranes or magnetic fields and also appears as an important part of more complicated problems like Navier-Stokes. Given a domain $\Omega \subset \mathbb{R}^2$ and a continuous function $f : \Omega \to \mathbb{R}$, we want to find a function $u : \Omega \to \mathbb{R}$ such that

$$-\Delta u = f \quad \text{in } \Omega \tag{3}$$

and satisfies certain prescribed conditions on the boundary of the domain $\partial\Omega$. We typically have two types of boundary conditions, namely Dirichlet:

$$u = \bar{u} \text{ at } \partial\Omega_D \tag{4}$$

and Neumann:

$$\frac{\partial u}{\partial n} = \bar{t} \text{ at } \partial\Omega_N \tag{5}$$

Here $\bar{u}$ is prescribed boundary value of the unknown u along $\partial\Omega$, $\bar{t}$ the prescribed (Neumann) flux along $\partial\Omega$ and $\partial u/\partial n$ is the normal derivative of u, i.e. the directional derivative respect to the outward normal vector n. Furthermore, we assume $\partial\Omega_D \cup \partial\Omega_N = \Omega$ and $\partial\Omega_D \cap \partial\Omega_N = \emptyset$. In case of pure Neumann problem we introduce the following notation: $\Gamma = \partial\Omega_N = \partial\Omega$

We now multiply by a test function $v \in \mathcal{V}$ and integrate over the domain, and write (3) as

$$\int_\Omega \Delta u\, v \,\mathrm{d}\Omega = \int_\Omega f\, v \,\mathrm{d}\Omega \quad \forall v \in \mathcal{V}\,.$$

Using Green's formula and pure Neumann boundary condition we can rewrite the problem as: Find $u \in \mathcal{V}$ such that

$$\int_\Omega \nabla u \,\nabla v \,\mathrm{d}\Omega = \int_\Omega f\, v \,\mathrm{d}\Omega + \int_\Gamma \bar{t}\, v \,\mathrm{d}S \quad \forall v \in \mathcal{V} \tag{6}$$

The problem written as in (6) is the variational formulation for the Poisson's equation.

We now apply Galerkin's approach and choose $\mathcal{V}_h \subset \mathcal{V}$ where $\mathcal{V}_h = \text{span}\{B_1, B_2, \dots B_n\}$. Any $u_h \in \mathcal{V}_h$ can then be written as a linear combination of the basis functions

$$u_h = \sum_{j=1}^{n} B_j \, u_j$$

with $u_j \in \mathbb{R}$. Substituting this into (6) and systematically selecting $v = B_i, i = 1, \dots, n$ allows us to write

$$\sum_{j=1}^{n} \int_\Omega \nabla B_i \, \nabla B_j \, \mathrm{d}\Omega \; u_j = \int_\Omega f \, B_i \, \mathrm{d}\Omega + \int_\Omega \bar{t} B_i \mathrm{d}\Omega \quad \forall i = 1, \dots, n$$

which is simply a linear system of equation of the form

$$A\boldsymbol{u} = \boldsymbol{b}$$

where

$$A_{i,j} = \int_\Omega \nabla B_i \, \nabla B_j \, \mathrm{d}\Omega \tag{7}$$

$$\boldsymbol{u} = [u_1, u_2, \dots u_n]^T$$

$$b_i = \int_\Omega f \, B_i \, \mathrm{d}\Omega + \int_\Gamma \bar{t} \, B_i \, \mathrm{d}S \, . \tag{8}$$

Due to historical reasons, the matrix A is called *Stiffness Matrix* and the vector $\boldsymbol{b}$ is called *Load Vector*, a nomenclature common in structural/solid mechanics for which the Finite Element method was developed in the late 50's.

2.2 Least Squares fitting

Performing a least-square fit of a surface is often encountered as a geometrical problem. In this case, given a smooth continuous function $f : \Omega \to \mathbb{R}$, we are searching for a function $u_h \in \mathcal{V}_h$ such that $\|u_h - f\|_{L^2}$ is as small as possible. It is possible to show that the solution u_h is the L^2-projection of f and

$$u_h = \arg\min_{u \in \mathcal{V}_h} \|u - f\|_{L^2} \iff \int_\Omega u_h \, v_h \, \mathrm{d}\Omega = \int_\Omega f \, v_h \, \mathrm{d}\Omega \quad \forall v_h \in \mathcal{V}_h \, .$$

Applying the same procedure as in the Poisson's equation case, we can write the problem as a solution of a linear system of equations

$$M\boldsymbol{u} = \boldsymbol{b}$$

where now the matrix M is called *Mass Matrix* and is defined as

$$M_{i,j} = \int_\Omega B_i \, B_j \, \mathrm{d}\Omega \, . \tag{9}$$

The load vector $\boldsymbol{b}$ is given by

$$b_i = \int_\Omega f \, B_i \, \mathrm{d}\Omega \, .$$

2.3 Helmholtz equation

Helmholtz equation often arises in problems involving waves; in particular it is the time-independent version of the wave equation and is written as

$$\Delta u + k^2\, u = f\,. \tag{10}$$

The solution u of Helmholtz equation represents the amplitude configuration of the wave in space, and k is the wavenumber.

It is easy now to see that the first term in (10) is the Laplacian operator we already encountered in the Poisson equation, while the second term is, besides the constant k, the same as in the least-squares fitting problem. Applying the same procedure as in the previous cases we are therefore able to rewrite the problem as searching for solutions of the linear system of equations

$$A\boldsymbol{u} + k^2 M\boldsymbol{u} = \boldsymbol{b} \;\Rightarrow\; \left(A + k^2 M\right)\boldsymbol{u} = \boldsymbol{b}\,.$$

As we have seen with the above examples, the matrices A and M play an important role in the solution of partial differential equations using a Galerkin approach. Simple elliptic problems may use only the stiffness matrix A; simple geometrical problems may use only the mass matrix M; while more complex or time-dependant problems use both.

The space $\mathcal{V}_h$ can be defined using several different types of basis functions. This choice is of great importance as it will dictate the properties of the solution space. Different types of basis functions will yield different system matrices and consequently this will affect the convergence rate of the numerical methods for solving linear systems of algebraic equations. For this reason, we will herein investigate the impact different classes of spline bases functions (presented in Section 3 below) have on important properties of these matrices as conditioning number, bandwidth, and sparsity.

3 Spline functions

In this section we present the theory of the three classes of spline functions considered in this paper: Classical Hierarchical, Truncated Hierarchical and LR B-splines. An effort has been made in order to unify the different concepts under a common framework of notations, to ease both the understanding and the comparison of the different technologies. We have included only the essentials and we refer the interested reader to the papers on which we based our studies for an in-depth introduction and details [13, 18, 23, 46].

3.1 Notation and common definitions

The Hierarchical (both Classical and Truncated) and the LR B-splines methodologies use quite different points of view when considering meshes and refinements. As such, different notations have been developed in the corresponding publications. We will in this paper use the following notation when we will need to address mesh-related quantities:

- ϵ for meshlines;
- Ω for domains, i.e. regions of the mesh (excluding mesh lines);
- V for full tensor product meshes;
- $\mathcal{M}$ for general meshes.

In particular, the Hierarchical setting focuses more on regions of the mesh and their underlying full tensor product meshes. For these reasons, Ω and V are often used in this context. The LR B-splines setting instead focuses more on meshlines and meshes as a whole. To provide a formal description of these different point of views we can write that a mesh $\mathcal{M}$ is seen as

$$\mathcal{M} = \bigcup_l (\Omega^l \cap V^l) \quad \text{in the Hierarchical setting}$$

$$\mathcal{M} = \bigcup_i \epsilon_i \quad \text{in the LR B-splines setting}$$

where the index l denotes the Hierarchical level and i runs over all meshlines.

The notation we will use for basis functions is the following:

- $N \in \mathcal{N}$ for *uniform* (in the index domain) tensor product basis functions.
- $B \in \mathcal{B}$ for tensor product basis functions (possibly non-uniform).
- $H \in \mathcal{H}$ for Classical Hierarchical basis functions.
- $T \in \mathcal{T}$ for Truncated Hierarchical basis functions.
- $L \in \mathcal{L}$ for LR B-splines basis functions.

Of course, there exist cases where for some indices $N_i = B_i = H_i = T_i = L_i$, but we hope the different notation will ease the understanding of the technologies.

We have from elementary spline theory that a *knot vector* is a nondecreasing sequence of coordinates in the parameter space of the form $\Xi = [\xi_1, \xi_2, \ldots, \xi_{n+p+1}]$, where each $\xi_i \in \mathbb{R}$ is called a *knot*. If the knot values are equidistant the knot vector is called *uniform*, and *non-uniform* otherwise. If the first and last knots have multiplicity $p+1$, the knot vector is called *open*. A knot vector comprising of $n+p+1$ knot values will generate n univariate linearly independent basis functions of degree p. We will focus our analysis on B-splines built from uniform, non-open knot vectors.

Corresponding to Ξ, we have the *index domain* $I = [1, \dots, n+p+1]$. The index domain is useful for considering non-uniform knots and also determine the support of functions. For uniform knot vectors Ξ we have $\Xi = \gamma I$ for some scaling factor $\gamma \in \mathbb{R}$, and this is what we will be working with in our examples. We would however like to stress that it is possible to generalize the same numerical tests using open or non-uniform knot vectors. For bivariate meshes, we consider the index domain to be the finest level tensor mesh, i.e. V^M, see Figure 3 for an example.

Definition 1. Given a knot vector $\Xi = [\xi_1, \xi_2, \dots, \xi_{n+p+1}]$ and a polynomial degree p, the n univariate basis functions $B_{1,p}, \dots, B_{n,p}$ are recursively defined in the following way:

$p = 0$:

$$B_{i,0}(\xi) = \begin{cases} 1 & \text{for } \xi_i \leqslant \xi < \xi_{i+1} \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

$p > 0$:

$$B_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} B_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} B_{i+1,p-1}(\xi) \tag{12}$$

The above definition is known as the *Cox-de Boor recursion formula*. From this definition it follows that each basis function depends only on $p+2$ knot values. For instance, for $p=2$ the knot vectors $\Xi = [0, 1, 2, 3, 4, 5]$ will generate three basis functions corresponding to the *local* knot vectors $\Xi_1 = [0, 1, 2, 3]$, $\Xi_2 = [1, 2, 3, 4]$, and $\Xi_3 = [2, 3, 4, 5]$. Due to this, we will often refer to the basis functions using their local knot vectors. The notation will also be adjusted on a case-to-case basis depending on what we need to emphasize, and we will use $B_i = B_{\Xi_i}$ to keep track of the local knot vector on which the function is built.

From the univariate basis functions it is possible to define multivariate functions using the tensor product structure of B-splines:

Definition 2. A d-variate B-spline $B(\boldsymbol{\xi})$ of degrees $\boldsymbol{p} = [p_1, p_2, \dots, p_d]$ is a separable function $B : \mathbb{R}^d \longrightarrow \mathbb{R}$ defined as:

$$B_{\boldsymbol{\Xi}}(\boldsymbol{\xi}) = \prod_{i=1}^{d} B_{\Xi_i}(\xi_i)$$

where $\Xi_i \in \mathbb{R}^{p_i+1}$ is the local knot vector for the univariate basis function of degree p_i along the i-th parametric dimension.

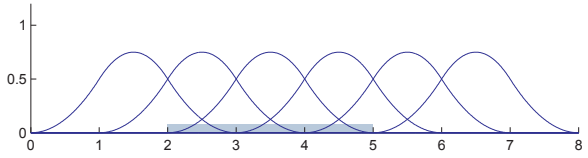
Note that the polynomial degree is implicitly defined by the number of knots in the local knot vectors. In the bivariate setting, it is customary to denote the two parametric coordinates as ξ and η, and the corresponding polynomial orders as p and q. We denote a *tensor product basis* $\mathcal{B} = \{B_1, B_2, \dots, B_n\}$ as a basis of functions defined by taking a tensor product of 1D basis functions.

In the following we will construct the mesh such that the actual domain will be the unit square, i.e. the initial tensor product basis will form a partition of unity on $\Omega_0 = [0, 1] \times [0, 1]$. Since we will use uniform knot vectors, we will need to extend the mesh beyond Ω_0 trough the use of a *ghost domain* G. In this way the full parametric domain will be given by $G \cup \Omega_0$.

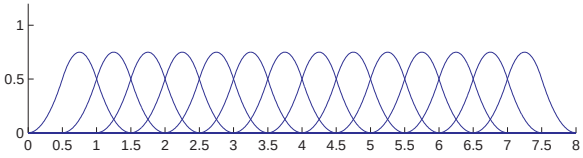
We will represent bivariate basis functions on the same plot through the use of anchors. A common choice for the coordinates of the anchor are the *Greville abscissae*. The Greville abscissae $(\bar{\xi}, \bar{\eta})$ corresponding to a basis function are defined as

$$\bar{\xi} = \frac{1}{p} \sum_{j=2}^{p+1} \xi_j, \quad \bar{\eta} = \frac{1}{q} \sum_{j=2}^{q+1} \eta_j \tag{13}$$

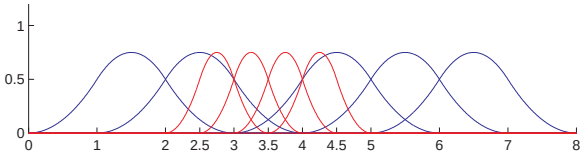
where ξ_j and η_j are the knot values in the local knot vector. The choice of Greville Abscissa for non-rectangular supports is not an obvious one, and turning to Giannelli et al. [19] it is



(a) The coarse mesh with the initial basis (level 0)



(b) The fine-scale basis (level 1)



(c) The Hierarchical basis, constructed as a suitable combination of the coarse and fine scale basis functions

Figure 2: **Hierarchical Basis:** Construction of a univariate basis using quadratic basis functions. The highlighted area is selected for refinement, and the coarse functions contained therein are substituted by finer basis functions.

defined as the coefficients which generates the linear monomoials. While this has many appealing properties, it means that the Truncated and Hierarchical basis functions' anchors will coincide. In order to highlight the difference between the methods we will use *area-averaged coordinates* defined as

$$\bar{\xi} = \frac{\sum_{i:E_i \in \text{supp}T} A_i \, \xi_i^c}{\sum_{i:E_i \in \text{supp}T} A_i}, \quad \bar{\eta} = \frac{\sum_{i:E_i \in \text{supp}T} A_i \, \eta_i^c}{\sum_{i:E_i \in \text{supp}T} A_i} \tag{14}$$

where the sum runs over all elements E_i in the support of the function, (ξ_i^c, η_i^c) are the coordinates of the centre of the element E_i and A_i is its area. The weighting by the area ensures that each element contributes in the right way to the resulting coordinate of the anchor. This method of calculating the coordinates allows to see the difference when the support is non-rectangular.

3.2 Hierarchical B-Splines

The application of the Hierarchical framework in Isogeometric Analysis is very well explained by Vuong et al. in [46], and Giannelli et al. in [18]. We will look at how an admissible mesh is constructed, and how the construction procedure defines a sequence of nested bounded domains linked to the different Hierarchical levels.

3.2.1 Introduction and general idea

The basic idea underlying Hierarchical B-splines is very simple, yet results in a good and flexible method to locally refine the mesh. A one-dimensional example for quadratic basis functions is illustrated in Figure 2: One portion of the initial level 0 mesh is selected for refinement. The coarse basis functions contained in that area are substituted by finer basis functions, and we thus obtain the Hierarchical basis.

Figure 3 presents the Hierarchical approach on a simple 2D example with biquadratic basis functions. When a selected area of the mesh is refined, the knot spans are halved in each direction and this introduces one new level in the hierarchy. The basis functions from the previous level that are contained in the refined region are then substituted by the corresponding finer basis functions defined on the new knot spans.

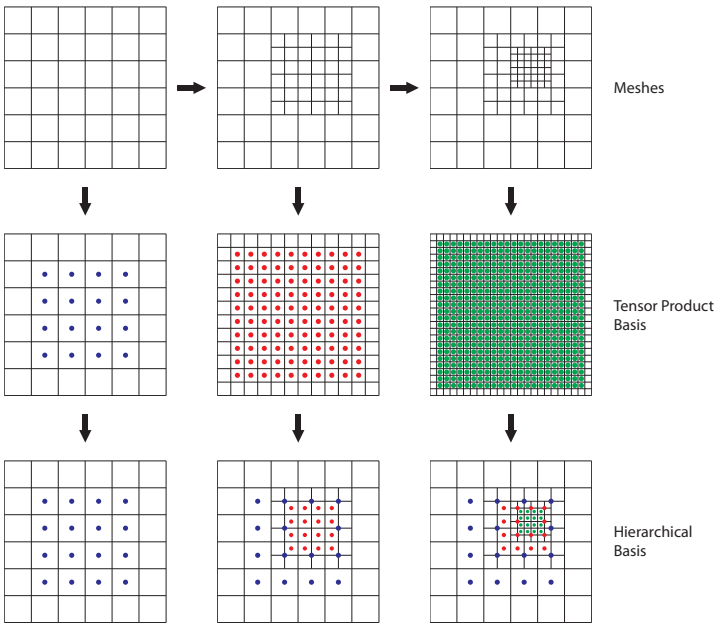


Figure 3: **2D Hierarchical Basis:** Example using biquadratic basis functions. **Upper row**: A two-step refinement is applied to the initial mesh displaying Ω^0, Ω^1 and Ω^2. **Middle row**: The tensor-product basis defined on the finest knot span available showing the functions $\mathcal{N}^l$ on the mesh V^l for all levels $l = 0, 1, 2$. From here we select the appropriate basis functions to include in the Hierarchical basis. **Lower row**: The actual Hierarchical basis defined on the refined mesh $\mathcal{M}$ above. At each step, the basis functions from the previous level that are contained in the refined region are substituted by the finer ones, showing the Hierarchical basis $\mathcal{H}$. The anchors are positioned using Greville Abscissae.

In the following we will focus mainly on the two dimensional case, and several examples will be presented.

3.2.2 The Classical Hierarchical Basis

The construction of the mesh on which the Hierarchical basis is defined is a direct application of the idea presented above: starting from an initial, tensor product mesh V^0, some areas are selected for refinement. Once the areas have been selected, several new meshlines are introduced, halving the knot spans of the local knot vectors of all the functions contained therein.

This gives a hierarchy of nested domains. We will remove certain functions from the coarser domains, and add certain functions of the finer domains. These functions are going to be a subset of the corresponding tensor product functions for the full mesh on that level.

We are now ready to construct the Hierarchical basis:

Definition 3. The *Hierarchical B-spline basis* $\mathcal{H}$ is recursively constructed as follows:

1. Initialization: $\mathcal{H}^0 = \{N \in \mathcal{N}^0 : \operatorname{supp} N \neq \emptyset\}$
2. Recursive case: $\mathcal{H}^{l+1} = \mathcal{H}_A^{l+1} \cup \mathcal{H}_B^{l+1}$ for $l = 0, \dots, M-1$, where
$$\mathcal{H}_A^{l+1} = \{N : N \in \mathcal{H}^l, \operatorname{supp} N \nsubseteq \Omega^{l+1}\}$$
$$\mathcal{H}_B^{l+1} = \{N : N \in \mathcal{N}^{l+1}, \operatorname{supp} N \subseteq \Omega^{l+1}\}$$
3. $\mathcal{H} = \mathcal{H}^M$

The recursive definition ensures that we always select the correct functions to include in the basis. The first step initializes the Hierarchical basis with all the relevant functions of the underlying tensor product basis $\mathcal{N}^0$. The recursive procedure then updates the basis by removing the coarse functions contained inside the refined region and including the finer ones substituting them.

Figure 4 presents some of the basis functions defined on the same mesh used in Figure 3. In the Classical Hierarchical case, all the functions have rectangular support since they are plain tensor product of univariate functions.

As a result of the definition, the Classical Hierarchical B-spline basis and the associated spaces have the following properties, as proved in [46]:

- The functions in $\mathcal{H}$ are linearly independent.
- The spaces spanned by the basis are nested, i.e. $\operatorname{span}\mathcal{H}^l \subseteq \operatorname{span}\mathcal{H}^{l+1}$.

Borrowing the terminology introduced for T-splines [42, 41], we classify different hierarchical basis by the following definition

Definition 4. We denote a basis $\{N_i\}$

- *standard* if $\sum N_i = 1$
- *semi-standard* if there exists a choice of weights $w_i > 0$, such that $\sum w_i N_i = 1$
- *non-standard* no choice of $w_i > 0$ exist to ensure partition of unity

The general definition of Hierarchical B-splines allows for all three kinds of basis. A tensor product mesh will yield a standard basis, but this will not be the case for arbitrary meshes. We will define our set of admissible meshes to be given as the following, which will ensure all basis to be semi-standard.

Definition 5. In the Hierarchical setting, we will call a mesh *admissible* if at all levels the area selected for refinement Ω^l is defined as the union of the supports of previous-level basis functions $N \in \mathcal{N}^{l-1}$.

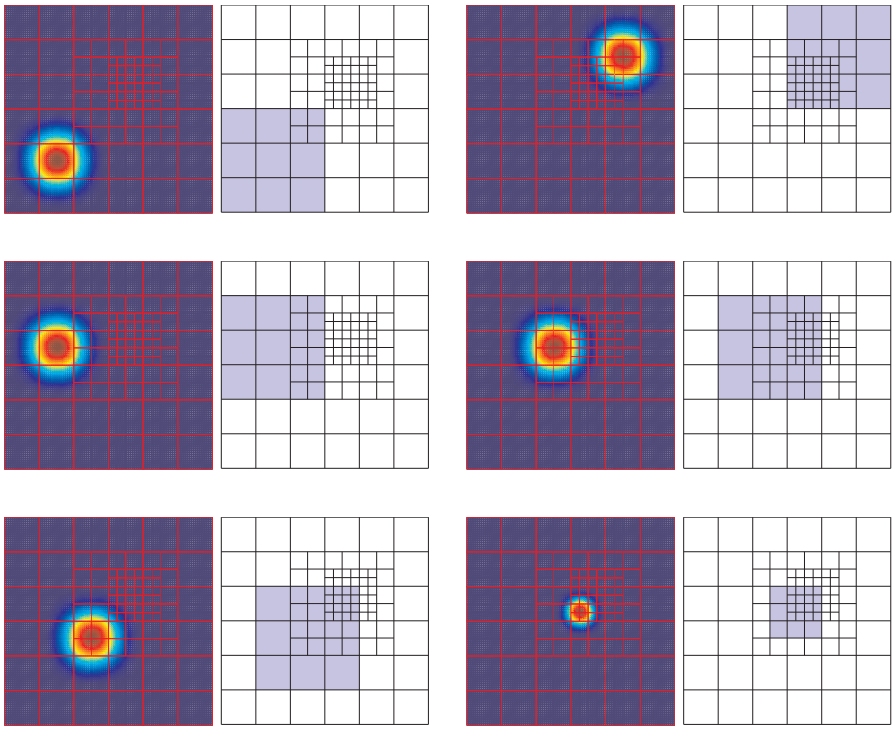
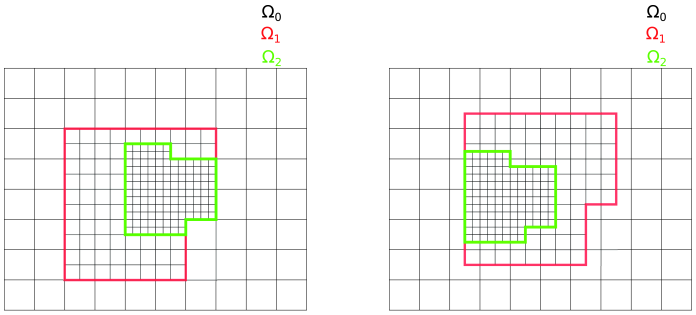


Figure 4: **Classical Hierarchical Basis:** Some of the biquadratic basis functions defined on the same mesh used in Figure 3. For each function, on the left is presented a top view of the evaluation plot and on the right the elements constituting its actual support.

The set of admissible meshes as defined above is a subset of the Hierarchical B-splines as it rules out among other, all non-standard basis. A generalization, which is not considered in this paper is that domain boundaries may not coincide with previous levels. This is known as weak condition on domain boundaries, and is illustrated in Figure 5. Another generalization is small refinement regions in which finer level functions appear, but coarser functions are not removed, see Figure 6. It is important to note here that non-standard basis may be perfectly valid for computations as they are linearly independent and well defined, but they form a different set of admissible meshes, which is not covered by LR B-splines and it is thus not possible to construct a basis on the same mesh and do a comparison study.

We consider Definition 5 to be highly relevant for mesh refinement. This is due to the fact that it is customary for iterative refinement to produce an error measure and refine regions of large errors. For Hierarchical B-splines, this error measure is often defined on an error per basis function level, followed by refinement of the largest error functions [23], [18].



(a) Strong condition on domain boundaries. The boundary of Ω^{l+1} is aligned with the knot lines of $\mathcal{V}^l$.

(b) Weak condition on domain boundaries. The boundary of Ω^{l+1} is aligned with the knot lines of $\mathcal{V}^{l+1}$.

Figure 5: **Hierarchical setting:** Different conditions on the domain boundaries. Weak boundary condition may produce non-standard basis and is not considered in this paper.

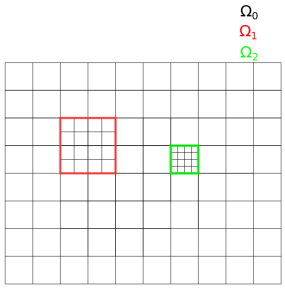


Figure 6: **Hierarchical setting:** Small domain sizes, may not be large enough to remove coarse level functions. This results in a non-standard basis and will not be considered in this paper. For this particular example, neither Ω^1 nor Ω^2 is large enough to remove any level 0 functions (for $p > 1$), but they are both large enough to create level 1 and level 2 functions of degree $p = 2$ or $p = 3$.

3.2.3 The Truncated Hierarchical Basis

While the Hierarchical B-splines presented above provide good flexibility and allow for localized refinement, the number of overlapping basis functions can increase very rapidly with the introduction of new levels. This happens because the large support of the coarse basis functions may overlap with the support of several fine-scale ones. See for example in Figure 4, where the top-right function, defined at level 0, overlaps with all the fine-scale functions in level 2.

This behaviour has a negative impact on the formation and solution of linear system of algebraic equation associated to the solution of the discrete (finite element) variational problem: A higher number of overlaps means we need to perform more functions evaluations and add more elements in the system matrices. This on one side increases the assembly time required to build such matrices as well as the sparsity. In order to, among others, address these problems, a new basis for the Hierarchical space was proposed in [18]. The key idea is that we can appropriately *truncate* the coarse basis functions, thus reducing their support and significantly decreasing the number of overlaps.

To start, we note that the function spaces are nested. That is: coarse functions can be represented as a linear combination of finer functions. For general knot vectors consider Equation (18) in section 3.3, but for uniform meshes, this corresponds to a simpler expression. Each coarse function can be represented as a sum of scaled, translated copies of itself.

$$N_{\Xi}(\xi) = \sum_{i=1}^{p+2} 2^{-p} \binom{p+1}{i-1} N_{\Xi}(2\xi - \xi_i) \tag{15}$$

Note that $N_{\Xi}(2\xi - \xi_i) = N_{\Xi_i}$ where the new knot vector Ξ_i is constructed from Ξ halving all the knot spans and taking $p+2$ subsequent knots. For example, for a quadratic basis function defined on $\Xi = [0, 1, 2, 3]$ we would have

$$\Xi_1 = [0, 0.5, 1, 1.5] \quad \Xi_2 = [0.5, 1, 1.5, 2]$$
$$\Xi_3 = [1, 1.5, 2, 2.5] \quad \Xi_4 = [1.5, 2, 2.5, 3]$$

The relation given in Equation (15) is at the core of the truncation: It tells us which functions of level $l+1$ we need to represent a function of level l.

The truncation of a basis function is defined as follows:

Definition 6. Let T be a basis function defined at level l, and let

$$T = \sum_{j \,:\, N_j \in \mathcal{N}^{l+1}} \alpha_j \, N_j$$

be its representation respect to the fine-scale basis associated to level $l+1$. The *truncation* of T respect to $\mathcal{N}^{l+1}$ and Ω^{l+1} is defined as

$$\operatorname{trunc}^{l+1} T = \sum_{\substack{j \,:\, N_j \in \mathcal{N}^{l+1}, \\ \operatorname{supp} N_j \not\subseteq \Omega^{l+1}}} \alpha_j \, N_j \tag{16}$$

It is clear that the coefficients α_j depend not only on the component N_j they refer to, but also on the function T considered. We omitted the explicit dependence to ease the notation.

The Truncated Hierarchical basis is then defined as follows [18] :

Definition 7. The *Truncated Hierarchical B-spline basis* $\mathcal{T}$ is recursively constructed as follows:

1. Initialization: $\mathcal{T}^0 = \mathcal{H}^0$

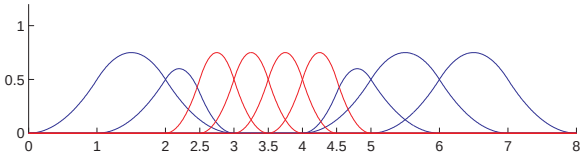


Figure 7: **Truncated Hierarchical Basis:** The quadratic basis on the same mesh as in Figure 2. Partition of unity is automatically achieved by the truncation procedure.

2. Recursive case: $\mathcal{T}^{l+1} = \mathcal{T}_A^{l+1} \cup \mathcal{T}_B^{l+1}$ for $l = 0, \ldots, M-1$, where

$$\mathcal{T}_A^{l+1} = \{\text{trunc}^{\,l+1} T : T \in \mathcal{T}^l \wedge \text{supp}\, T \not\subseteq \Omega^{l+1}\}$$
$$\mathcal{T}_B^{l+1} = \mathcal{H}_B^{l+1}$$

3. $\mathcal{T} = \mathcal{T}^M$

Note that the representation of T in terms of next-level functions is easily obtained through Equation (15). The truncation mechanism removes all those components of T that are explicitly included in the basis by the recursive step of the definition. This procedure appropriately shrinks the support of all functions that cross over multiple levels in the mesh, effectively reducing the number of overlaps. Also note that the way of expressing the truncation as given in Equation (16) is what we will call an *additive* representation. It is also possible to use a *subtractive* representation, expressing the truncation as

$$\text{trunc}^{\,l+1} T = T - \sum_{\substack{j\,:\,N_j \in \mathcal{N}^{l+1}, \\ \text{supp}\, N_j \subseteq \Omega^{l+1}}} \alpha_j\, N_j \tag{17}$$

Both these representations have advantages and disadvantages which are discussed in Section 4.

Figure 7 shows the Truncated Hierarchical basis constructed on the same mesh as in Figure 2. Note that no weights are needed to maintain the partition of unity.

Figure 8 presents the same basis functions as in Figure 4 with the truncation procedure applied. As we can see, the support of each Truncated function is modified in order to reduce the number of overlaps with finer levels. This, however, makes some functions lose the rectangular shape of their support.

The Truncated Hierarchical basis naturally inherits the properties of the Classical Hierarchical basis, and also adds some more. In particular:

- The functions in $\mathcal{T}$ are linearly independent.
- The spaces are nested, i.e. $\text{span}\mathcal{T}^l \subseteq \text{span}\mathcal{T}^{l+1}$.
- The basis maintains partition of unity.

In addition, if we consider the Classical Hierarchical basis $\mathcal{H}$ defined on the same mesh as $\mathcal{T}$, then:

- The cardinality of the basis is the same: $|\mathcal{H}| = |\mathcal{T}|$.
- The spaces spanned are the same: $\text{span}\mathcal{H} = \text{span}\mathcal{T}$.

Proofs for the above can be found in [18].

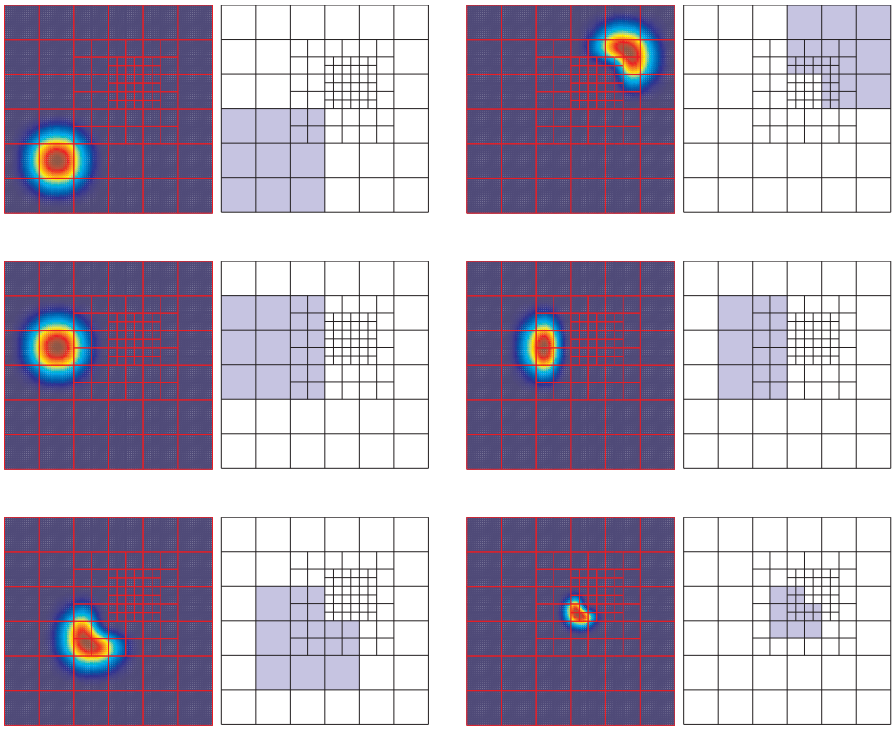


Figure 8: **2D Truncated Hierarchical basis:** The biquadratic basis constructed on the same mesh, and the corresponding functions, as in Figure 4. For each function, on the left is presented a top view of the evaluation plot and on the right the elements constituting its actual support.

3.3 LR B-splines

LR B-splines were recently proposed by Dokken et al. in [13] and later applied to Isogeometric Analysis by Johannessen et al. in [23]. We report here some of the theory contained in those papers, while taking a different approach that focuses on clarity and ease of understanding.

LR B-splines differentiate themselves from the Hierarchical cases in the way the refinement is applied: while Hierarchical functions rely on the subdivision rule given in Equation (15) and generate up to $p+2$ new functions from each original B-spline, LR B-splines use the knot insertion procedure, inserting one knot at a time and splitting old B-splines into 2 new ones. The fact the the knots are inserted one at a time is crucial, especially in the bivariate setting: We will show that even when inserting the same knot values as produced by the subdivision rule, the resulting refined B-spline basis may be different.

LR B-splines are locally refined in the same way the standard tensor-product B-splines are. From basic spline theory we know that it is possible to perform knot insertion to enrich the spline space while leaving the geometry description unchanged. In the univariate case, if we want to insert the knot $\hat{\xi}$ between the knots ξ_{i-1} and ξ_i we have

$$B_{\Xi}(\xi) = \alpha_1 \, B_{\Xi_1}(\xi) + \alpha_2 \, B_{\Xi_2}(\xi) \tag{18}$$

where

$$\begin{aligned} \alpha_1 &= \begin{cases} \frac{\hat{\xi}-\xi_1}{\xi_{p+1}-\xi_1} & \xi_1 \leqslant \hat{\xi} \leqslant \xi_{p+1} \\ 1 & \xi_{p+1} \leqslant \hat{\xi} \leqslant \xi_{p+2} \end{cases} \\ \alpha_2 &= \begin{cases} 1 & \xi_1 \leqslant \hat{\xi} \leqslant \xi_2 \\ \frac{\xi_{p+2}-\hat{\xi}}{\xi_{p+2}-\xi_2} & \xi_2 \leqslant \hat{\xi} \leqslant \xi_{p+2} \end{cases} \end{aligned} \tag{19}$$

Figure 9: **LR B-splines:** Examples of knot insertion for $\hat{\xi} = \frac{3}{2}$. Dashed lines: The original functions. Colors: The new functions resulting from the splitting.

and the knot vectors are

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

As we can see, inserting one knot splits the original B-spline into two new B-splines described by the local knot vectors Ξ_1 and Ξ_2. The weights α_1 and α_2 are needed to maintain partition of unity. Figure 9 shows some examples of the application of Equation (18).

In the bivariate case, functions are refined one parametric direction at a time. In this case we obtain:

$$\begin{aligned}
B_{\boldsymbol{\Xi}}(\xi, \eta) &= B_{\Xi}(\xi)\, B_{\Psi}(\eta) \\
&= (\alpha_1\, B_{\Xi_1}(\xi) + \alpha_2\, B_{\Xi_2}(\xi))\; B_{\Psi}(\eta) \\
&= \alpha_1\, B_{\boldsymbol{\Xi_1}}(\xi, \eta) + \alpha_2\, B_{\boldsymbol{\Xi_2}}(\xi, \eta)
\end{aligned} \tag{20}$$

In the following we will call *meshline extension* all mesh-altering actions like inserting a new meshline, prolonging existing meshlines (possibly connecting two existing ones) or increasing the multiplicity of meshlines. When a new meshline extension is inserted, we need to know which basis functions are affected by it. For this purpose, we give the following definition:

Definition 8. A meshline ϵ is said to *traverse the support* of a function $B_{[\xi_1 \dots \xi_{p_1+2}; \eta_1 \dots \eta_{p_2+2}]}$ if

- ϵ is a horizontal line $\epsilon = [\xi_1^*, \xi_2^*] \times \eta^*$ such that

$$\xi_1^* \leqslant \xi_1, \;\; \xi_{p_1+2} \leqslant \xi_2^*, \;\; \eta_1 \leqslant \eta^* \leqslant \eta_{p_2+2}$$

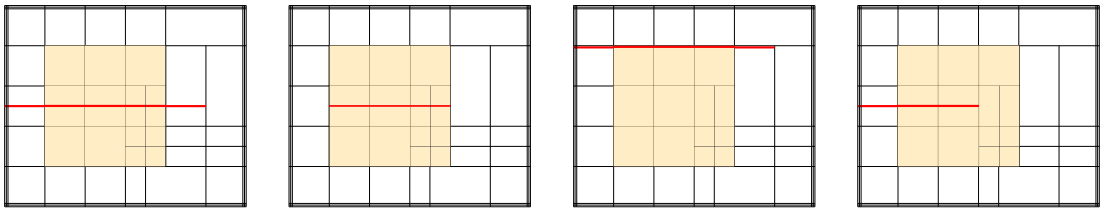
- ϵ is a vertical line $\epsilon = \xi^* \times [\eta_1^*, \eta_2^*]$ such that

$$\xi_1 \leqslant \xi^* \leqslant \xi_{p_1+2}, \;\; \eta_1^* \leqslant \eta_1, \;\; \eta_{p_2+2} \leqslant \eta_2^*$$

In particular, a horizontal line is said to traverse *the interior* of $B_{[\xi_1 \dots \xi_{p_1+2}; \eta_1 \dots \eta_{p_2+2}]}$ if $\eta_1 < \eta^* < \eta_{p_2+2}$ and traverse *the edge* if $\eta^* = \eta_1$ or $\eta^* = \eta_{p_2+2}$. Similarly, a vertical line is said to traverse the interior if $\xi_1 < \xi^* < \xi_{p_1+2}$ and traverse the edge if $\xi^* = \xi_1$ or $\xi^* = \xi_{p_1+2}$.

Figure 10 shows some examples of lines traversing the support of a basis function.
When a meshline extension is applied, the refinement process is composed of two steps:

1. Split any function which support is traversed by the *new* meshline.
2. For all new functions, check if their support is traversed by any *existing* meshline, and split again if this happens.



(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 edge nor the interior of B

Figure 10: **LR B-splines:** Examples of lines traversing the support of a basis function.

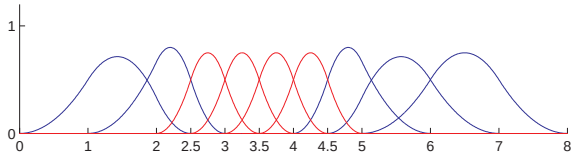


Figure 11: **LR B-splines:** The basis constructed on the same mesh as in figures 2 and 7.

In step 1 we test all current functions against one meshline. In step 2 we test all newly created functions against all existing meshlines. Note that when the meshline extension is an actual elongation, possibly connecting two separate existing meshlines, we will use the full length of the resulting line to test the functions for splitting. When a function is flagged for splitting, this is performed through the use of Equations (19) and (20).

In view of the above, we can give the following two definitions:

Definition 9. In the LR B-splines setting, an *admissible mesh* is any mesh which can be obtained by a sequence of meshline extensions starting from an initial tensor product mesh. Each extension must cause at least one basis function to be split, and the meshlines must end at existing knot values (they cannot stop at the centre of an element). All tensor product meshes are admissible.

Definition 10. An *LR B-spline* is a function which results from the application of the refinement scheme and Equations (18)-(19). All tensor product B-splines are LR B-splines.

Figure 11 shows the 1D LR B-splines basis defined on the same mesh as in the previous examples at Figures 2, and 7. Note that in the univariate setting the LR B-spline refinement coincide with the normal knot insertion. In this case all the weights sum up to 1 and so the LR B-spline basis is the normal B-splines basis originating from the non-uniform knot vector $\Xi = [0, 1, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8]$, and automatically maintains partition of unity. Also note that in the general LR B-spline setting the notion of levels is not as present as in the Hierarchical setting; we can however define the level of an LR B-spline function using the maximum knotspan contained in its local knot vector.

For a thorough example in the bivariate case we refer the reader to [23, p. 481-483].

Given an initial tensor product mesh $\mathcal{M}_0$, a sequence of meshline extensions $\{\epsilon_i\}_{i=1}^n$ and corresponding admissible meshes $\mathcal{M}_i = \{\mathcal{M}_{i-1} \cup \epsilon_i\}$ and LR B-splines $\mathcal{L}^i$, the following properties hold [13, 23]:

- The spaces are nested: $\text{span}\mathcal{L}^i \subseteq \text{span}\mathcal{L}^{i+1}$.
- The LR B-splines defined on a mesh are not affected by the order in which the meshline extensions have been inserted, i.e. if $\mathcal{M}$ and $\hat{\mathcal{M}}$ are two identical LR B-splines meshes that differ only for the order in which the meshlines extensions have been applied, then the resulting LR B-splines functions are the same.

- The LR B-splines form a partition of unity.

Note that, in general LR B-splines may be linear dependent. This is mostly due to the fact that the single-line insertion mechanism used in this setting allows for several different types of refinement strategies, and the particular choice naturally affects the spline space. However, there are several ways to recover the linear independency as proposed in [13, 23]. The linear independence of LR B-splines depending on the type of refinement strategy used is currently object of research.

In all our examples we will use the *Structured Mesh* refinement presented in [23]. This strategy focuses on refining functions, instead of elements. The idea of refining elements is indeed a legacy from the classical Finite Element methodology. Using the Structured Meshes approach, one instead selects which basis functions to refine. This can be done through the use of custom-built criteria, just as one would do in an adaptive refinement scheme. The idea proposed in [23] is to compute the error pertaining to each basis function as

$$\|e\|^2_{\text{supp}B_i} = \sum_{K \in \text{supp}B_i} \|e\|^2_K \tag{21}$$

i.e. we define the *B-spline error* as the sum of the normal error $\|e\|$ measured in the energy norm over all elements in the support of B_i. Once the functions to be refined are identified, we proceed to insert several knot lines in both directions, halving the knot spans of the largest supported knot interval. Note that the Structured Mesh strategy will yield the same results on the mesh as the subdivision rule used in the Hierarchical setting. This means that all meshes which are admissible in the Hierarchical setting, i.e. they satisfy the conditions of Definition 5, are also admissible in the LR B-splines setting and can be obtained using the Structured Mesh refinement. For this reason we have always used this approach for our examples, as it provides a better ground for comparison.

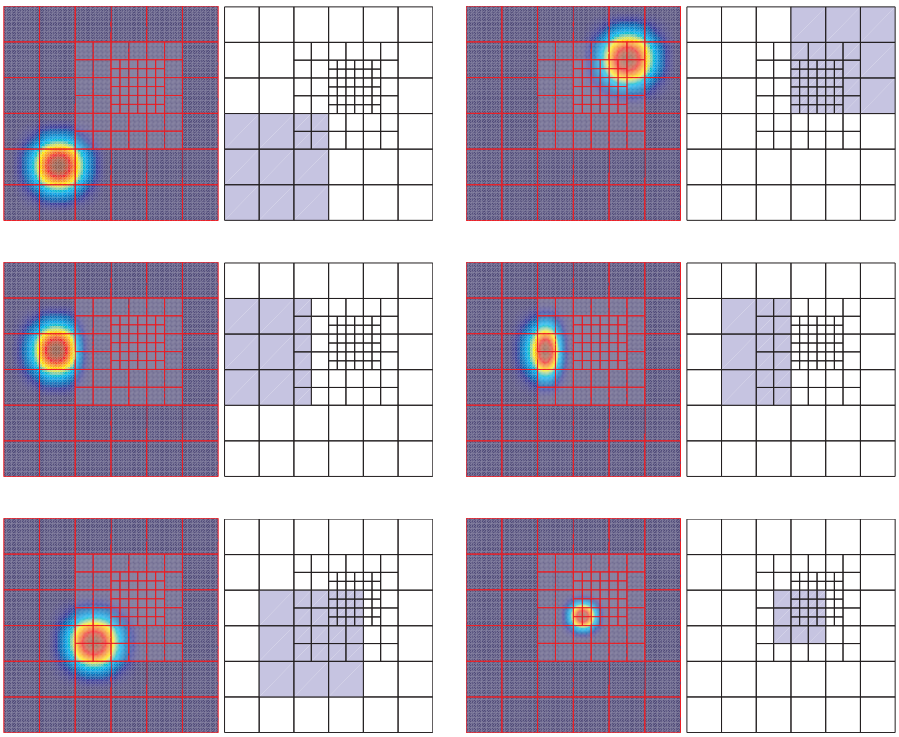


Figure 12: **LR B-splines:** The biquadratic basis constructed on the same mesh, and the corresponding functions, as in Figure 4 and 8. For each function, on the left is presented a top view of the evaluation plot and on the right the elements constituting its actual support.

4 Results

We present here the results of our analysis on the different type of basis functions outlined in Section 3. The *Qualitative analysis* sections collects results regarding the mathematical properties of the various basis, many of which were already briefly listed in the corresponding sections. The *Quantitative analysis* section focuses on implementation and numerical quantities and discusses the properties of the stiffness and mass matrices generated using the different splines functions.

4.1 Qualitative analysis

We would like to start pointing out that, under normal mesh refinement iterations (i.e. excluding special constructed cases), on the qualitative level all the three classes of splines give comparable results. However, there are some interesting distinctive features that are worth to be mentioned.

4.1.1 Different functions

With the notation introduced at page 7 we have that the general functions $H \in \mathcal{H}$, $T \in \mathcal{T}$, and $L \in \mathcal{L}$ can be written as

$$H = N$$
$$T = \sum \alpha_i N_i$$
$$L = \alpha B$$

for appropriate indices i and weights α

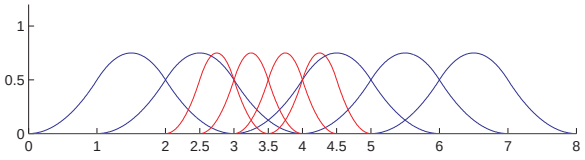
This can be seen as follows. Consider a set of nested domains $\Omega^0 \supset \Omega^1 \supset \ldots \supset \Omega^M$, with corresponding tensor mesh $V^0 \subset \ldots \subset V^M$, where $V^l = \Xi^l \otimes \Psi^l$, $\Xi^l \subset \Xi^{l+1}$ and $\Psi^l \subset \Psi^{l+1}$. The knot vectors of $N_i^l \in \mathcal{N}^l$ is picked as *connected subsets* of Ξ^l and Ψ^l. These are uniform both in the index domain of Ξ^l and Ξ^M. This is in contrast to how the LR B-splines are constructed, as these are in general a *unconnected* subset of Ξ^M and Ψ^M and hence non-uniform in the index domain.

For uniform starting meshes under dyadic refinement this becomes slightly more apparent as every N will be comprised of uniform local knot vectors, while any B will potentially be non-uniform. The Classical Hierarchical functions (Definition 3) are then uniform B-splines; Truncated Hierarchical functions (Definitions 6 and 7) are generally a linear combination of these uniform B-splines; LR B-splines functions (Equation (18) and Definition 10) are non-uniform B-splines.

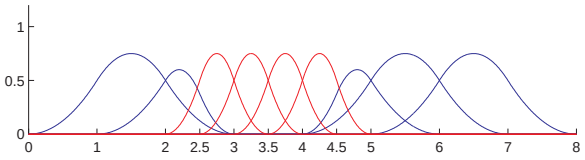
Figure 13 shows the Classical Hierarchical, Truncated Hierarchical and LR B-splines basis for the knot vector $\Xi = [0, 1, 2, 2.5, 3, 3.5, 4, 4.5, 5, 6, 7, 8]$. Note that on the uniform vector $[0 : 8]$ all three families of functions would be exactly the same. In the Classical Hierarchical case, partition of unity is not preserved. In the Truncated Hierarchical case this is achieved automatically by the truncation procedure, which removes some components from the old-level functions. The LR B-splines are instead defined by non-uniform local knot vectors, and also use weights to maintain partition of unity.

Figure 14 shows a comparison of the support for some of the basis functions presented in the Figures 4, 8, and 12.

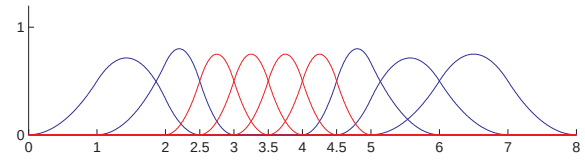
Another way of interpreting the difference between the functions is the following. While all technologies are built on the same relation of nested functions (18) in general or (15) for uniform knot vectors, the application of this equation is enforced at different steps. For Hierarchical splines, one will only remove coarse functions and add new ones, while keeping existing functions unchanged (save the ones being removed). For Truncated Hierarchical functions, one will alter



(a) Classical Hierarchical

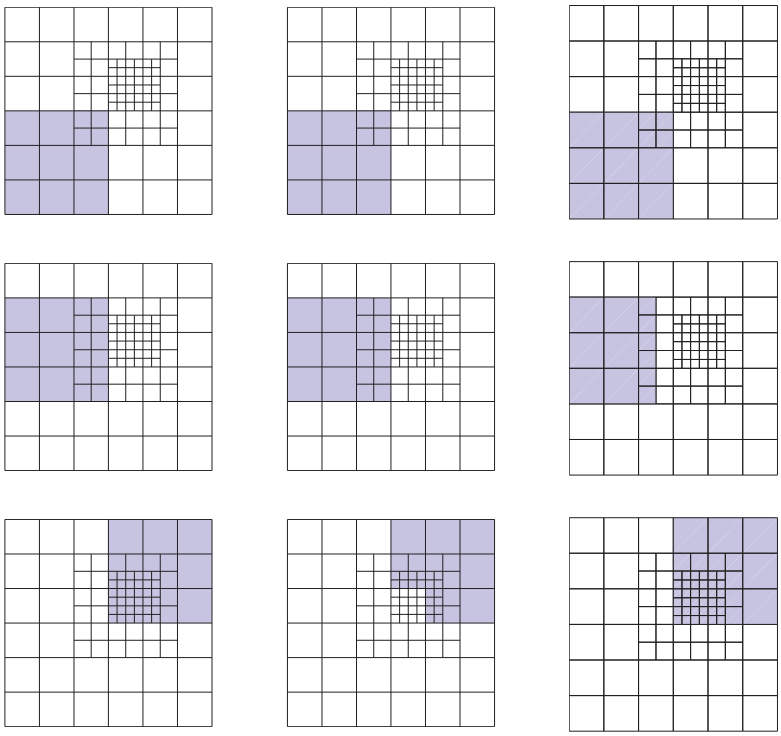


(b) Truncated Hierarchical



(c) LR B-splines

Figure 13: **Qualitative analysis:** The different quadratic bases constructed using the same knot vector.

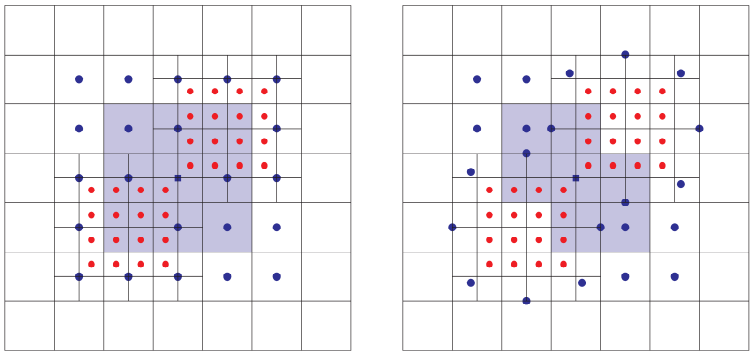


(a) Left Column: Classical Hierarchical

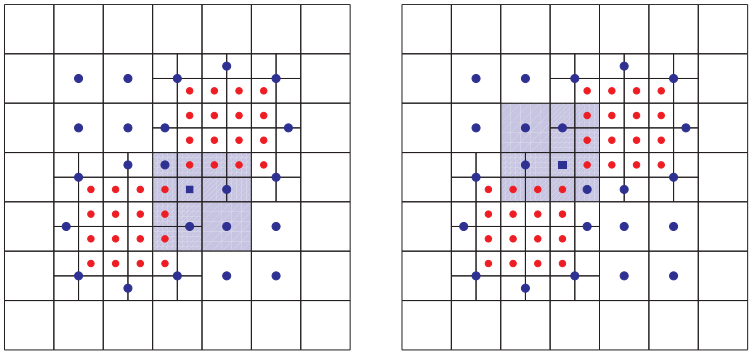
(b) Centre Column: Truncated Hierarchical

(c) Right Column: LR B-splines

Figure 14: **Qualitative analysis - Different Functions:** The support of corresponding bi-quadratic basis functions in the three spline families presented in the Figures 4, 8, and 12.



(a) Classical Hierarchical (b) Truncated Hierarchical



(c) LR B-splines, first function (d) LR B-splines, second function

Figure 15: **Qualitative analysis - Different spaces:** An example of mesh on which the biquadratic Hierarchical bases span different spaces than the LR B-splines basis. The central level 0 function in the Hierarchical cases corresponds to two distinct functions in the LR B-splines basis. Both the Hierarchical bases are constituted of 55 functions; the LR B-splines basis contains 56 functions. The highlighted area is the support of the selected function, represented with a square as anchor symbol. The anchors are placed as described at page 8.

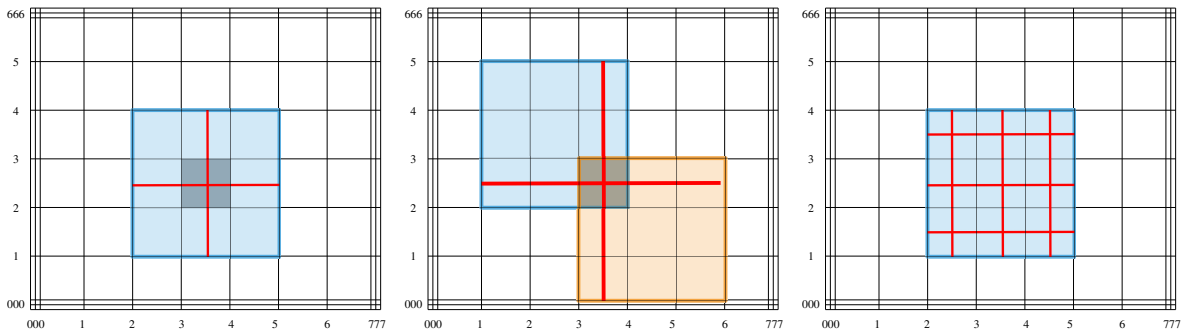
any coarse functions whose support contains the entire span of a fine function, keeping all else unchanged. For LR B-splines one will alter the shape of any coarse function whose support overlaps (completely in one direction, partly in the other) with a fine domain. A consequence of this is that truncated basis functions have smaller support on the diagonal or at level corners, but LR functions have smaller horizontal and vertical support.

4.1.2 Different spaces

Perhaps the most important and interesting difference is that for some meshes the Hierarchical basis and the LR B-splines set span different spaces. One such example is given in figure 15.

The central function appearing in the Classical and Truncated Hierarchical setting corresponds to two distinct functions in the LR B-splines set. This is due to the way the refinement works in the LR B-splines setting: As we can see there is one new meshline which completely traverses the support of the central function. When this meshline is inserted, it triggers the LR B-splines refinement algorithm which splits the original B-spline into two new ones as expected. This does not happen in the Hierarchical framework, which leaves the function unchanged in the Classical case or appropriately reduces its support in the Truncated case.

Mourrain [33] presented a formula for the maximum dimension of the space of piecewise



(a) Minimum Span (b) Full Span (only two functions are shown) (c) Structured Mesh

Figure 16: **Qualitative analysis:** Different types of refinement strategies using LR B-splines.

polynomials with given continuity on a mesh: given a planar mesh with F faces (the elements), ϵ_H horizontal and ϵ_V vertical internal edges and P vertices, the maximum dimension of the space of bivariate piecewise polynomials of degrees (p, q) with continuity (k, l) along element edges is given by

$$\mathcal{S} = (p+1)(q+1)F - (p+1)(l+1)\epsilon_H - (q+1)(k+1)\epsilon_V + (k+1)(l+1)P + \mathbb{H} \qquad (22)$$

where $\mathbb{H}$ is the homology factor of the mesh, which is equal to zero for all the refinement schemes used here. For Hierarchical B-splines, as well as Truncated Hierarchical, it is possible to add constraints on the mesh topology to ensure spanning the entire space [32], [20]. For LR B-splines one may assure this by the so called "hand in hand" process [13], which again puts restrictions on mesh topologies. For meshes under Definition 5, this however cannot be guaranteed.

4.1.3 Different refinement strategies

Hierarchical functions rely on Equation (15) to apply the refinement. This procedure halves the local knot vectors of the function and replaces the original B-spline with up to $p + 2$ new functions in the univariate case, or $(p + 2)(q + 2)$ in the bivariate case.

LR B-splines use the knot insertion given in Equation (18), which introduces two new B-splines functions. This procedure allows for more flexibility in the refinement approach as there are no prescriptions on the number or positions of the new knots. In addition to the Structured Mesh strategy, already presented in Section 3, in [23] two other different refinement strategies are proposed: *Minimum Span* and *Full Span*. While all these strategies insert the meshlines so that they halve the knotspans, this is not a requirement as the use of non-uniform knot vectors is already built-in in the definitions of LR B-splines.

The Minimum Span strategy aim is to keep the refinement as localized as possible. Once an element is marked for refinement, a cross is inserted through its centre and the meshlines are made to be as short as possible, while still splitting at least one function.

In the Full Span strategy the idea is to split *all* B-splines with support on a selected element. This is done inserting two meshlines in a cross through the centre of the element. The new meshlines will have to span from the minimum to the maximum knot values of all functions with support on the marked element in both parametric direction. This strategy makes sure that all B-splines with support on the marked element are treated equally, but on the other hand this results in an extension of the refinement away from the selected element, in particular for high polynomial degrees.

A common drawback of both the Full and Minimum Span is that some elements will be traversed by only one meshline and therefore will be split into two rectangular elements, effectively doubling their aspect ratio.

While the possibility of applying other refinement strategies is allowed by the definitions and the theory of LR B-splines, some may lead to linearly dependant sets. The research in these cases is still ongoing and, as of today, the Structured Mesh is the best candidate for a stable refinement algorithm.

4.1.4 Different admissible meshes

A direct consequence of the various refinement approaches available with LR B-splines is that some meshes which are legal in a LR setting cannot be reproduced using Hierarchical splines. On the other hand, LR B-splines are not capable of achieving some configurations of the weak conditions on domain boundaries available in the Hierarchical framework. For an example see Figure 5b. In that case, the meshlines of Ω^1 are stopping in the centre of the elements, a behaviour that is disallowed by the LR definitions.

Another difference is that LR B-splines are currently defined starting from a global tensor-product mesh, i.e. only on rectangular parametric domains. Conversely, Hierarchical B-splines can be defined on non-rectangular parametric domains.

Meshes that are defined on a rectangular domain and also satisfy the conditions of Definition 5 are admissible in both the Hierarchical and LR B-splines framework.

4.2 Quantitative analysis

Here, we first discuss details related to different representation of Truncated Hierarchical basis, and then present the numerical results obtained for different meshes and polynomial degrees,

4.2.1 Representation of Truncated functions

As we briefly mentioned earlier, Truncated Hierarchical B-splines can be represented in an additive or subtractive fashion; we restate Equations (16) and (17) for reading convenience:

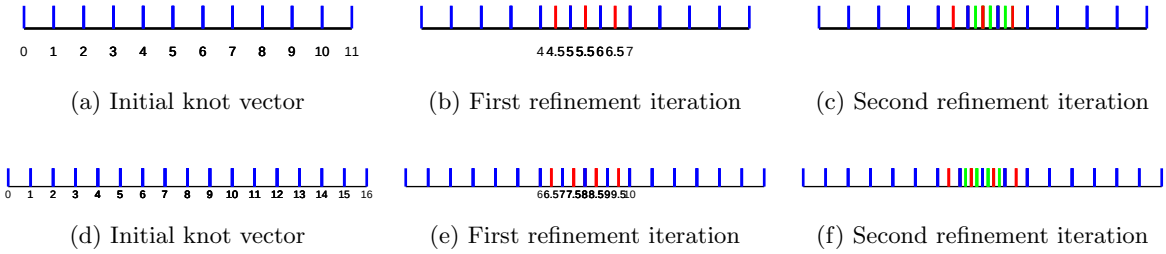
$$\operatorname{trunc}^{l+1} T = \sum_{\substack{j \,:\, N_j \in \mathcal{N}^{l+1}, \\ \operatorname{supp} N_j \not\subseteq \Omega^{l+1}}} \alpha_j \, N_j \qquad \text{Additive representation} \tag{23}$$

$$\operatorname{trunc}^{l+1} T = T - \sum_{\substack{j \,:\, N_j \in \mathcal{N}^{l+1}, \\ \operatorname{supp} N_j \subseteq \Omega^{l+1}}} \alpha_j \, N_j \qquad \text{Subtractive representation} \tag{24}$$

where N_j are the components of T with respect to the finer level basis functions and α_j the corresponding weights as given by Equation (15).

When implementing the code we found that choosing one representation over the other yield important consequences. In a typical finite element code one has to deal with two important aspects: determining which basis functions are active over a given element, and then evaluating such functions.

To address the former a convenient way to retrieve or store the support of the functions is essential. In the case of B-splines this is generally easy since the support is identified with the local knot vectors. When the B-spline is a standard tensor product, and the support is therefore rectangular, this becomes even easier since one only needs to check the starting and ending points of the local knot vectors. As we have seen, however, Truncated Hierarchical functions do not always have rectangular support, hence we need a representation that allows for an easy way to retrieve it. The subtractive representation (24) is unfortunately not very helpful in this sense: the fact that a given component is subtracted does not automatically guarantee that the function itself vanishes in that area. Given an element $E \in \operatorname{supp} T$ we should check if *all* possible components on that element are removed in order to know if $\operatorname{trunc} T$ has support on E



(a) Initial knot vector (b) First refinement iteration (c) Second refinement iteration

(d) Initial knot vector (e) First refinement iteration (f) Second refinement iteration

Figure 17: **1D Central Refinement:** The first three steps of the refinement process in the cases $p = 2$ (above) and $p = 3$ (below). When two functions are equally close to the centre, the rightmost one is selected for refinement.

or not. The additive representation (23), on the other hand, is much more convenient: We can simply loop over all components and check if *any* of them has support on E.

To address the latter point we need an efficient way to evaluate basis functions. This is even more important in an Isogeometric setting: Since the Cox-de Boor algorithm is a typical bottleneck of the code, we would like to perform as few basis evaluations as possible. In this case the additive representation (23) is not efficient. In a biquadratic case a representation in terms of next-level basis functions comprises of 16 fine-scale functions. This number clearly increases when increasing the polynomial degree: 25 for bicubic functions, 36 for biquartic etc. In addition, an additive representation may require to store the function in terms of the finest-available scale, which would greatly increase the amount of components needed; let's assume, for simplicity, that this is not the case. To give an example, look at the biquadratic basis functions of Figure 8. For each of the Truncated Hierarchical basis functions only 4 components are removed. This means that in an additive representation we would still need to evaluate 12 fine-scale functions in order to compute the value of the B-spline we are interested in. In a subtractive representation we would need to evaluate only 5 functions: The original tensor product B-spline and the 4 components we need to subtract.

To summarize, we have the following:

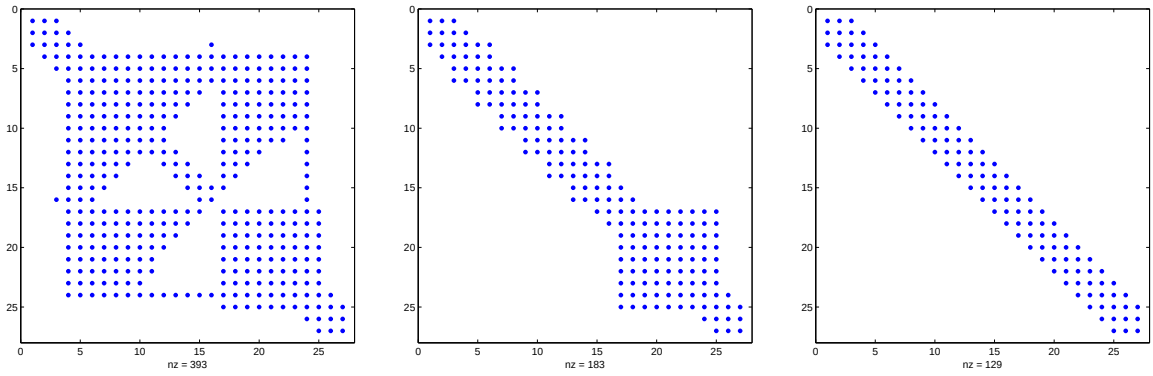
- An additive representation (23) is useful when determining the support but not efficient in the function evaluation process;
- A subtractive representation (24) does not allow to easily identify the support of the function but is more efficient in its evaluation.

The above discussions are overall considerations: The disadvantages of the representations might be accounted for by programming the algorithm in a smart efficient way. On the other hand, this is still something that needs to be taken into consideration. For an in-depth discussion on the implementation of Truncated Hierarchical B-splines we refer to [26].

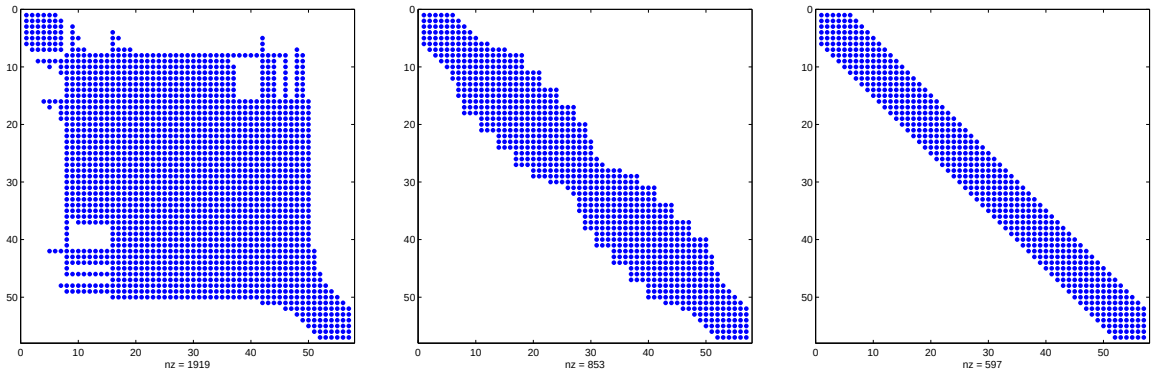
4.2.2 1D examples

We now present the results obtained in various 1D examples. We performed several experiments for polynomial degrees $p = 2, 3, 4$, and 5. In each case we started from a uniform, non-open knot vector $\Xi^0 = [0, 1 \dots 5\,p + 1]$ and successively applied 6 refinement steps, always refining the central basis function. Note that for odd polynomial degrees a central function always exists, while for even-degree normally there are two functions near the knot vector centre. In this case we chose to always refine the rightmost one. Figure 17 shows as examples the first two refinement iterations for $p = 2$ and $p = 3$.

For each refinement iteration we constructed the stiffness matrix A and the mass matrix M using the Classical Hierarchical, Truncated Hierarchical and LR B-splines functions defined



(a) Classical Hierarchical, $p = 2$ (b) Truncated Hierarchical, $p = 2$ (c) LR B-splines, $p = 2$



(d) Classical Hierarchical, $p = 5$ (e) Truncated Hierarchical, $p = 5$ (f) LR B-splines, $p = 5$

Figure 18: **1D Central Refinement:** Examples of sparsity patterns of the stiffness matrices at the last(6th) refinement iteration. The Cuthill-McKee Algorithm has been applied to optimize the bandwidth. Top: $p = 2$. Bottom: $p = 5$.

using the same knot vector. We then analysed some important numerical properties of these matrices, namely the sparsity pattern, the conditioning number, and the spectrum. Note that in the univariate case the LR B-splines basis coincides with the standard non-uniform B-splines generated via knot insertion.

Sparsity Figure 18 shows the sparsity patterns of the stiffness matrix at the last refinement iteration after a reordering using the Cuthill-McKee Algorithm [12] has been applied. The top row corresponds to $p = 2$, while to bottom row corresponds to $p = 5$.

As expected the Classical Hierarchical basis functions produce the densest stiffness matrices. This is normal since the support of coarse-level functions remains unaffected by the refinement in neighbouring regions. The values for all polynomial degrees are collected in Table 1.

Conditioning Numbers The conditioning number of the stiffness and mass matrices can be significantly influenced by the way the boundary conditions are imposed. In order to avoid any such effect we decided to look at the "pure" conditioning numbers, i.e. before any imposition of the boundary conditions. As is well known, with just pure Neumann boundary conditions the stiffness matrix is singular; the conditioning number can then be defined as the ratio between the largest eigenvalue and the smallest non-zero one. This means that for either the stiffness matrix A or the mass matrix M, given the ordered set of their eigenvalues $[\lambda_1, \lambda_2 \dots \lambda_n]$ we

p	HB	THB	LR	H/T	H/LR
2	393	183	129	215%	305%
3	803	315	247	255%	325%
4	1257	629	403	200%	312%
5	1919	853	597	225%	321%

Table 1: **1D Central Refinement:** Number of non-zero elements in the stiffness matrix at the last (6th) refinement iteration. The last two columns present the ratios, rounded to the nearest percentage point.

Stiffness Matrix

Iter.	0	1	2	3	4	5	6
HB	12.7425	28.0291	55.7519	111.4035	222.7908	445.5791	891.158
THB	12.7425	25.8255	52.0501	105.3161	213.368	432.4906	876.3622
LR	12.7425	27.2848	55.6005	112.6381	228.1518	462.2306	936.1914

Mass Matrix

Iter.	0	1	2	3	4	5	6
HB	46.7947	52.5238	65.8931	116.2265	225.4839	448.1175	894.9733
THB	46.7947	41.5164	42.6706	45.6839	88.2484	176.373	352.7153
LR	46.7947	38.0372	38.4295	38.5944	67.7769	135.5371	271.0706

Table 2: **1D Central Refinement:** The conditioning numbers for $p = 2$ throughout the mesh refinement. Stiffness Matrix above, Mass Matrix below.

define

$$\begin{aligned} \text{cond}(A) &= \frac{\lambda_n}{\lambda_2} \\ \text{cond}(M) &= \frac{\lambda_n}{\lambda_1} \end{aligned} \tag{25}$$

Figure 19 shows the plots for the conditioning numbers of both the stiffness and mass matrices for each polynomial degree considered. While all values are quite close to each other, and always remained in the same order of magnitude for our experiments, it is interesting to note that the Truncated Hierarchical and LR B-splines perform very similarly.

Looking at the plots for the stiffness matrix we can see that, with the exception of the lowest degree, i.e. $p = 2$, the conditioning numbers are ordered as

$$\text{cond}(A_T) < \text{cond}(A_{LR}) < \text{cond}(A_H)$$

while for the mass matrix we always have

$$\text{cond}(M_{LR}) < \text{cond}(M_T) < \text{cond}(M_H)$$

where the subscripts indicates the basis functions used.

We can also see that the conditioning numbers of the stiffness matrices are increasing with each refinement iteration, while the conditioning numbers for the mass matrices for $p = 4$ and 5 are bounded from above and below by a constant. This behaviour was already presented by Gahalaut and Tomar in [16] and Garoni et al. [17], although that result is proven for uniform refinement only.

The Tables 2 and 3 present the numerical data for $p = 2$ and $p = 3$, respectively.

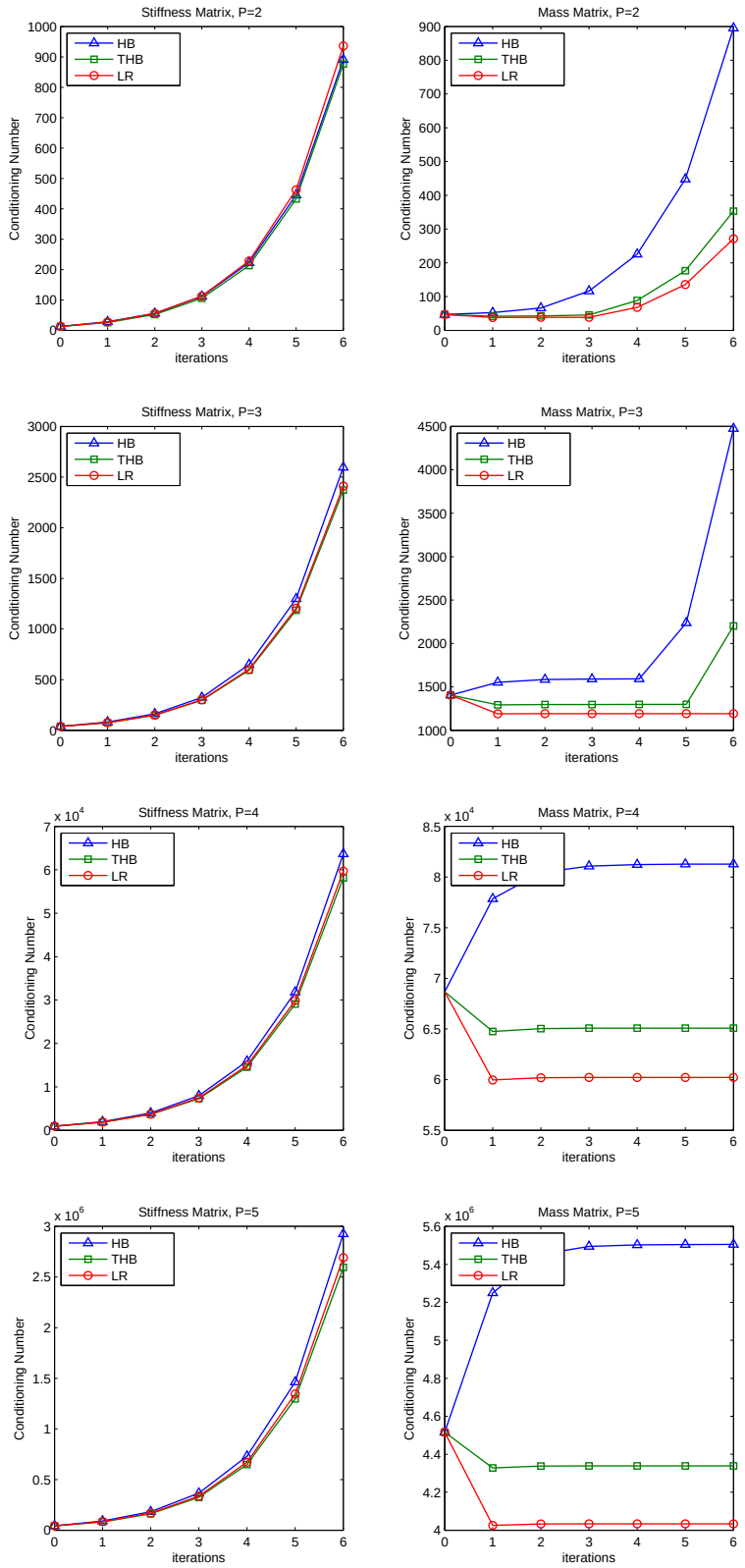


Figure 19: **1D Central Refinement:** Graphs of the conditioning numbers of stiffness matrices (left column) and mass matrices (right column) from $p = 2$ (top) to $p = 5$ (bottom).

Stiffness Matrix

Iter.	0	1	2	3	4	5	6
HB	37.5856	81.2603	162.2944	324.6481	649.3102	1298.622	2597.2442
THB	37.5856	74.0527	148.15	296.3336	592.6853	1185.3798	2370.7641
LR	37.5856	75.1932	150.6787	301.4619	602.9764	1205.9794	2411.9722

Mass Matrix

Iter.	0	1	2	3	4	5	6
HB	1405.2245	1553.052	1585.2845	1590.5673	1591.5617	2238.165	4476.3032
THB	1405.2245	1292.2619	1296.8079	1297.3633	1297.4726	1297.6033	2201.9071
LR	1405.2245	1190.1684	1191.548	1191.7976	1191.8173	1191.819	1191.8192

Table 3: **1D Central Refinement:** The conditioning numbers for $p = 3$ throughout the mesh refinement. Stiffness Matrix above, Mass Matrix below.

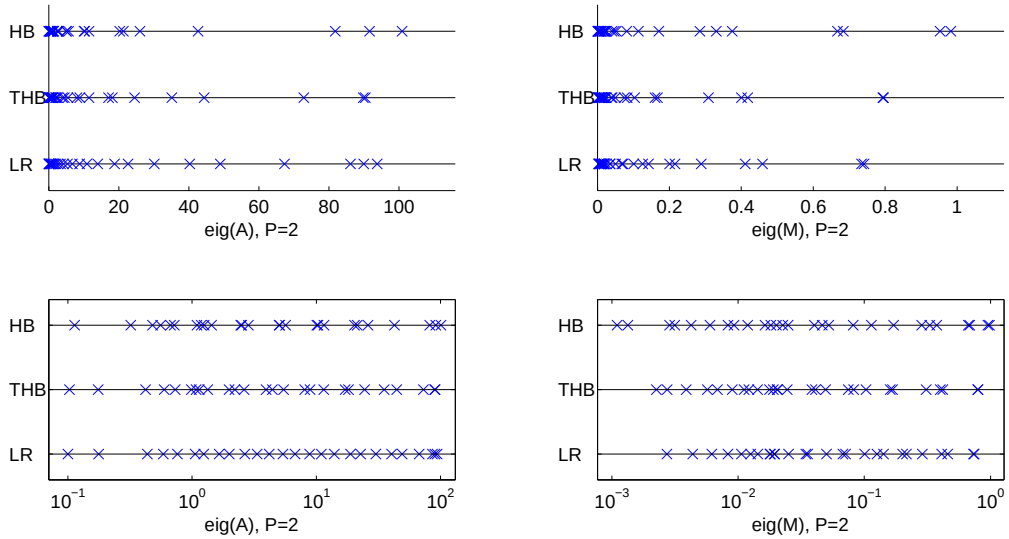


Figure 20: **1D Central Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 2$ at the last refinement iteration. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

Spectrum Figure 20 shows the spectra of the stiffness and mass matrices for $p = 2$ at the sixth refinement iteration. The eigenvalues of the stiffness matrix are spread over a large interval, while the eigenvalues of the mass matrix are much more clustered. In all cases the eigenvalues tend to be denser near the origin, but there is no substantial difference between the various basis functions.

Increasing the polynomial degree has different consequences on the eigenvalues of the stiffness and mass matrices: While the large eigenvalues of the stiffness matrix are reduced, those of the mass matrix are increased. The values of the smallest eigenvalues are instead reduced in all cases, as we would expect by the increase in the conditioning numbers. We noted, however, that only a small number of outliers quickly approaches zero, while the other smallest eigenvalues are still reduced but not as fast. In particular, for the Classical and Truncated Hierarchical basis functions the smallest eigenvalues seems to decrease faster than those associated with LR B-splines. Figure 21 shows the spectra of the stiffness and mass matrices produced with basis functions of degree $p = 5$.

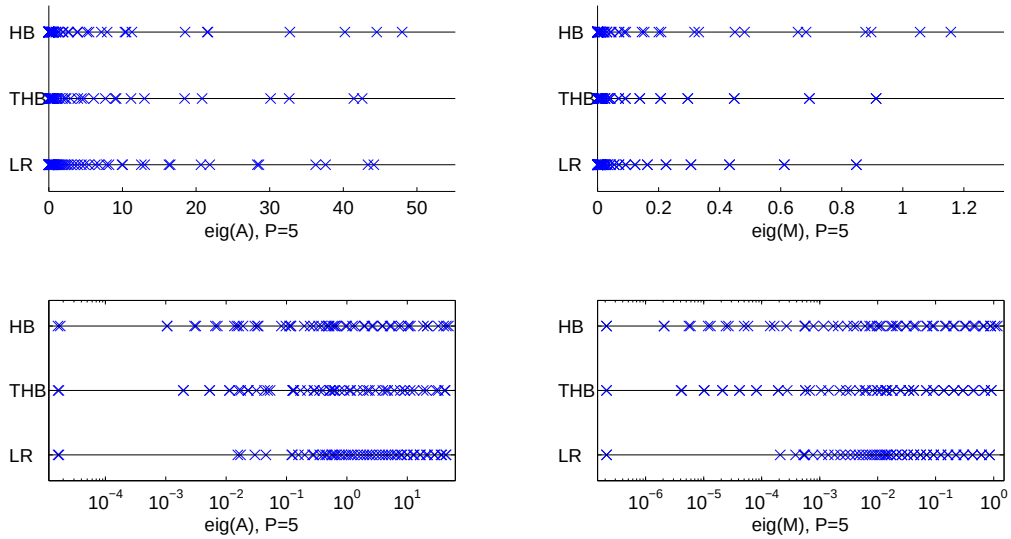


Figure 21: **1D Central Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 5$ at the last refinement iteration. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

4.2.3 2D Example: Central Refinement

The first of the 2D examples we present is the natural extension of the 1D cases considered above. Starting from a uniform tensor-product mesh, we performed five refinement iterations where at each step the central basis function was selected for refinement. Experiments were conducted for $p = 2, 3, 4$. As in the previous cases, for even polynomial degrees usually none of the basis functions is perfectly in the centre of the mesh; when this happened we chose to refine the lower-left function. Depending on the polynomial degree we also adjusted the knot vectors to have a ghost domain such that the initial tensor product basis constitutes a partition of unity in $\Omega^0 = [0, 1] \times [0, 1]$. Figure 22 shows the first three steps of the refinement process for $p = 2$ and $p = 3$.

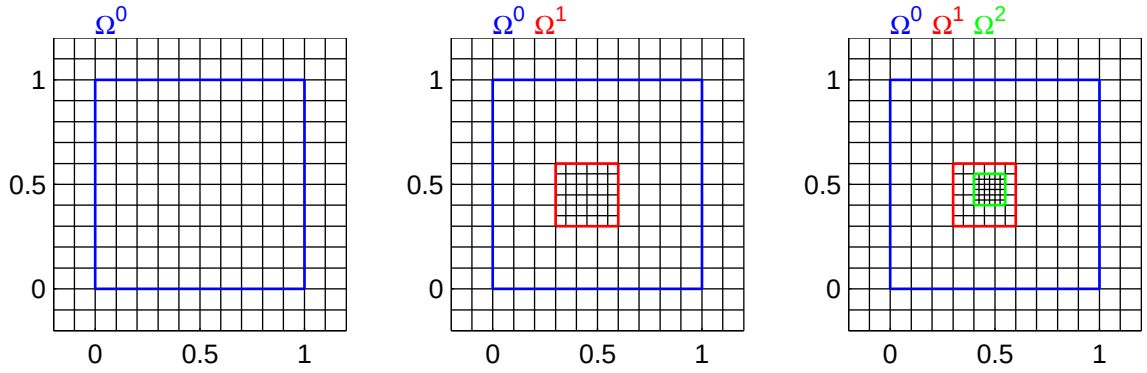
As in the previous examples, we constructed the stiffness and mass matrices using Classical Hierarchical, Truncated Hierarchical and LR B-splines basis functions on the same meshes, and compared their numerical properties.

Sparsity Figure 23 shows the sparsity pattern of the stiffness matrices after the fifth refinement iteration for biquadratic and biquartic basis functions. All results are reported in Table 4.

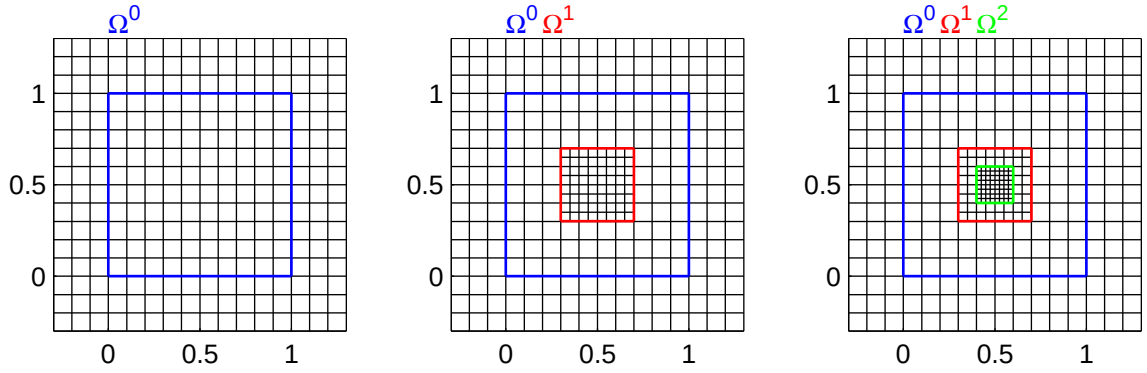
As expected the Classical Hierarchical basis produces significantly denser matrices due to the higher number of overlaps. While it may seem that the improvement gained by using the Truncated basis is less than in the 1D case, we have to take into consideration that the refined region is very small.

p	HB	THB	LR	H/T	H/LR
2	8403	6079	6711	138%	125%
3	23625	14909	18909	158%	125%
4	47913	36943	40839	130%	117%

Table 4: **2D Central Refinement:** Number of non-zero elements in the stiffness matrices at the last (5th) refinement iteration. The last two columns present the ratios, rounded to the nearest percentage point.

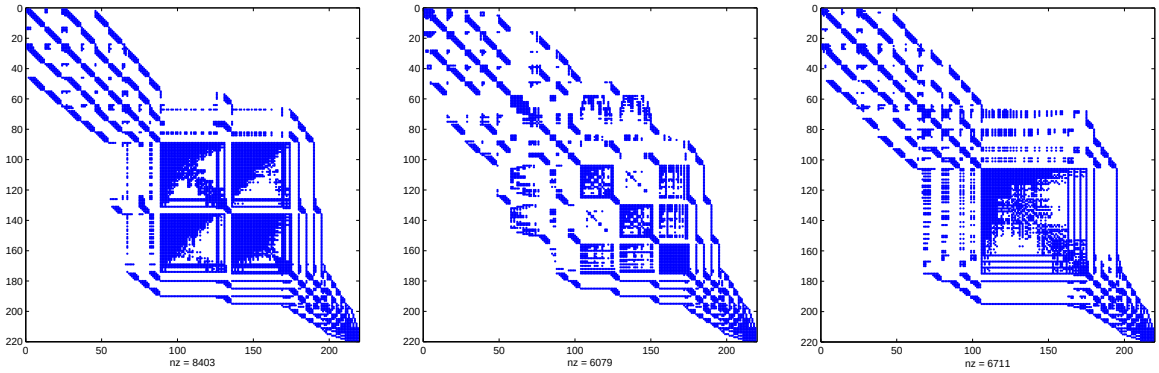


(a) Initial Mesh (b) First refinement iteration (c) Second refinement iteration

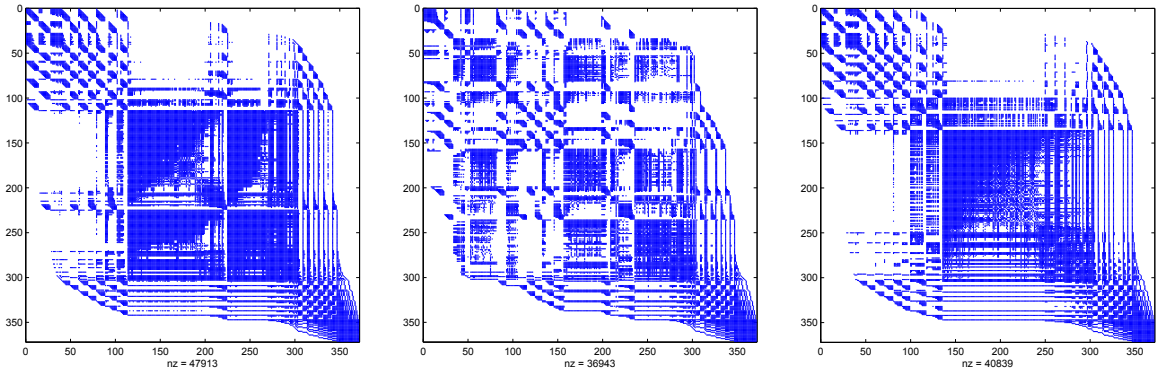


(d) Initial Mesh (e) First refinement iteration (f) Second refinement iteration

Figure 22: **2D Central Refinement:** The first three steps of the refinement process in the cases $p = 2$ (above) and $p = 3$ (below). When four functions are equally close to the centre, the lower-left one is selected for refinement.



(a) Classical Hierarchical, $p = 2$ (b) Truncated Hierarchical, $p = 2$ (c) LR B-splines, $p = 2$



(d) Classical Hierarchical, $p = 4$ (e) Truncated Hierarchical, $p = 4$ (f) LR B-splines, $p = 4$

Figure 23: **2D Central Refinement:** The sparsity patterns of the stiffness matrices at the last (5th) refinement iteration. The Cuthill-McKee Algorithm has been applied to optimize the bandwidth. Top: $p = 2$. Bottom: $p = 4$.

Stiffness Matrix

Iter.	0	1	2	3	4	5
HB	83.6793	125.4868	142.641	152.964	159.1009	162.9791
THB	83.6793	94.7517	99.6748	102.4115	103.8121	104.6173
LR	83.6793	91.0205	91.0144	91.0061	91.006	91.0105

Mass Matrix

Iter.	0	1	2	3	4	5
HB	2.241e+03	2.245e+03	2.245e+03	2.245e+03	5.808e+03	2.323e+04
THB	2.241e+03	2.155e+03	2.154e+03	2.154e+03	5.554e+03	2.221e+04
LR	2.241e+03	2.153e+03	2.152e+03	5.981e+03	3.245e+04	1.757e+05

Table 5: **2D Central Refinement:** The conditioning numbers for $p = 2$ in the various iterations of the central refinement. Stiffness Matrix above, Mass Matrix below.

Stiffness Matrix

Iter.	0	1	2	3	4	5
HB	2.323e+04	3.410e+04	3.792e+04	3.960e+04	4.043e+04	4.088e+04
THB	2.323e+04	2.714e+04	2.977e+04	3.112e+04	3.186e+04	3.231e+04
LR	2.323e+04	2.416e+04	2.421e+04	2.421e+04	2.421e+04	2.421e+04

Mass Matrix

Iter.	0	1	2	3	4	5
HB	1.975e+06	2.016e+06	2.019e+06	2.019e+06	2.019e+06	2.019e+06
THB	1.975e+06	1.837e+06	1.837e+06	1.837e+06	1.837e+06	1.837e+06
LR	1.975e+06	1.836e+06	1.836e+06	1.836e+06	1.836e+06	1.836e+06

Table 6: **2D Central Refinement:** The conditioning numbers for $p = 3$ in the various iterations of the central refinement. Stiffness Matrix above, Mass Matrix below.

Conditioning Numbers Figure 24 shows the plots of the conditioning numbers of the stiffness and mass matrices produced by the refinement procedure described above. As we can see, the conditioning numbers for the mass matrices are bounded for the higher degrees.

Tables 5 and 6 contain the numerical values of the conditioning numbers for $p = 2$ and 3, respectively.

Spectrum Figure 25 shows the spectra of the matrices obtained from the last refinement iteration using biquadratic basis functions. While there is no substantial difference in the eigenvalue distribution produced by the three refinement methodologies, we can see how the smallest eigenvalue of the mass matrix coming from LR B-splines functions is lower than its Hierarchical counterparts. This explains the higher conditioning number for LR B-splines than the Hierarchical refinement schemes.

Increasing the polynomial degree to $p = 4$ compacts the spectrum, reducing the lower eigenvalues but also the higher ones. As in the univariate case, the smallest eigenvalues are outliers and assume almost the exact same value for all three families of splines; the difference in the magnitude of the conditioning numbers is therefore dictated by the values of the greatest eigenvalues.

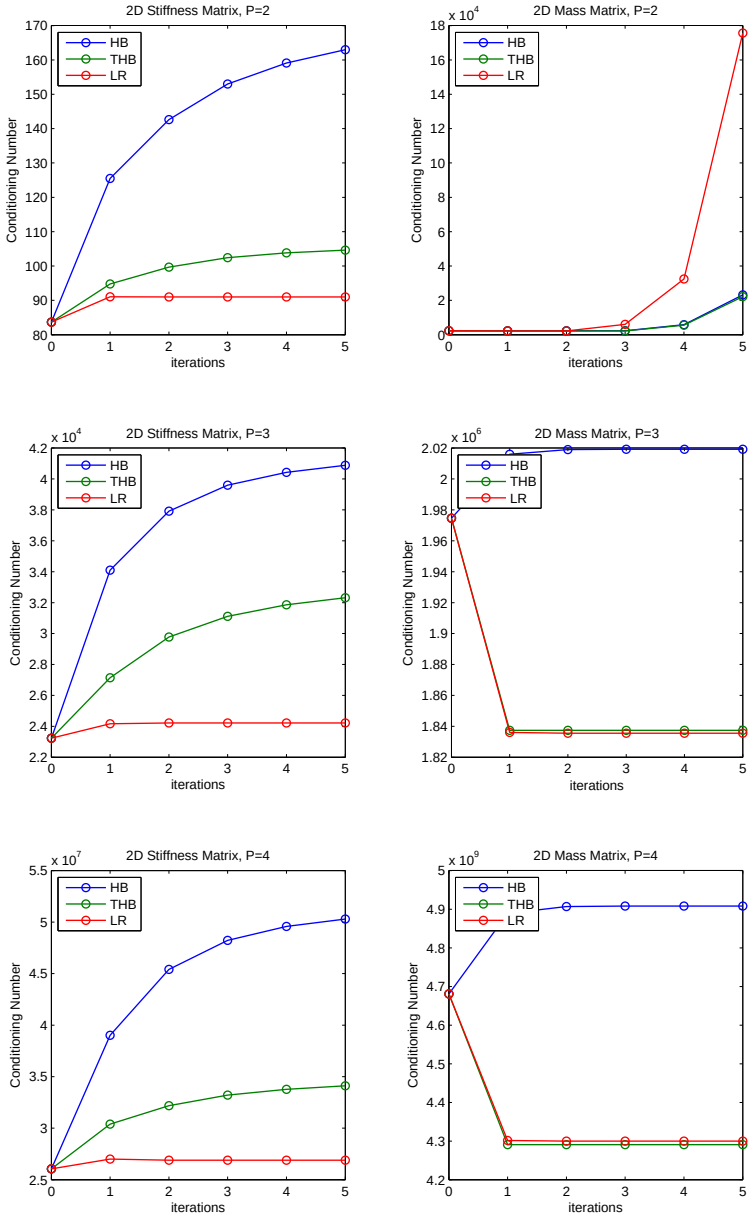


Figure 24: **2D Central Refinement:** Graphs of the conditioning numbers of stiffness matrices (left column) and mass matrices (right column) for the bivariate central refinement. $p = 2$ (top), $p = 3$ (middle), and $p = 4$ (bottom).

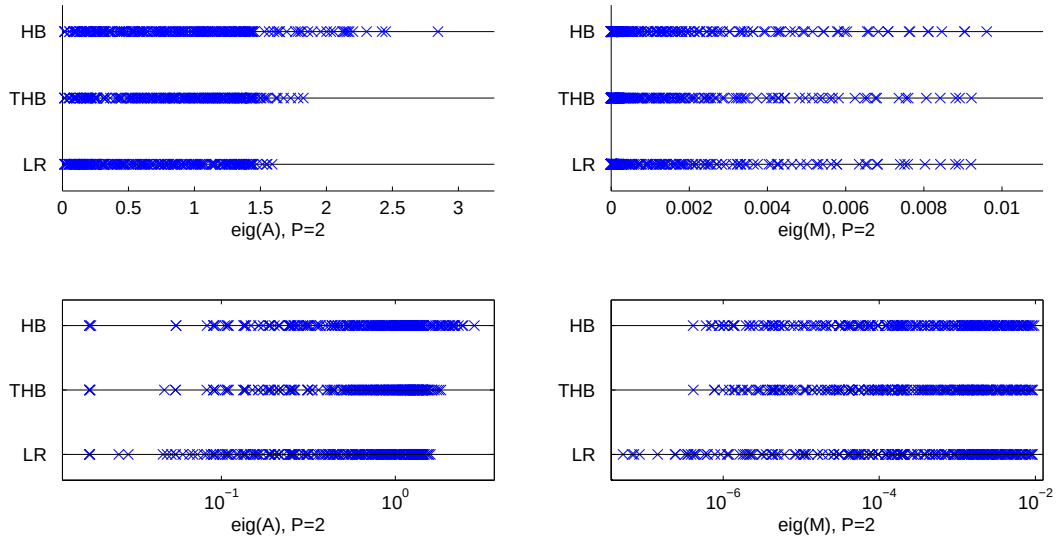


Figure 25: **2D Central Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 2$ at the last (5th) iteration of the central refinement, bivariate case. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

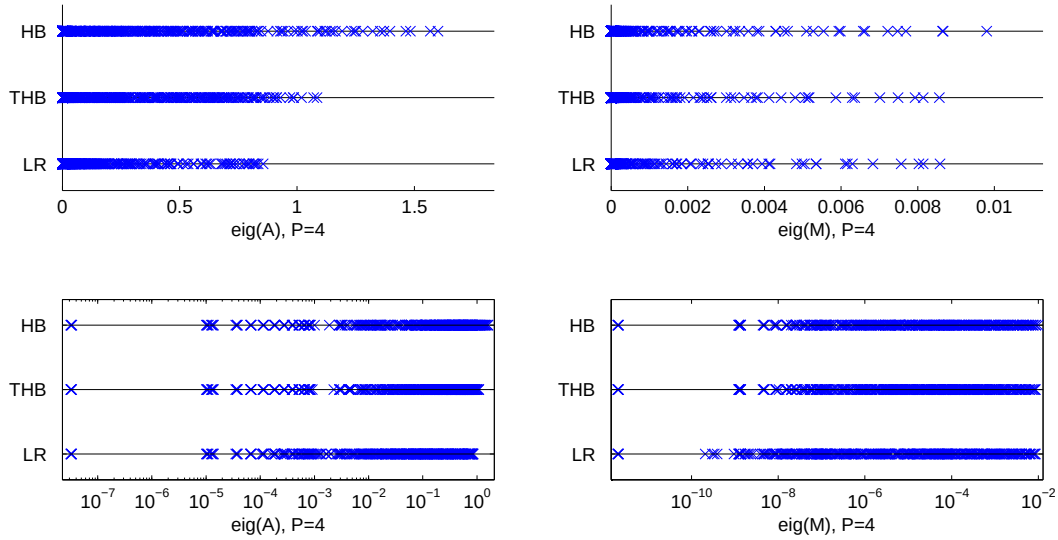


Figure 26: **2D Central Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 4$ at the last iteration of the central refinement, bivariate case. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

4.2.4 2D Example: Diagonal Refinement

We present here the results obtained applying a diagonal refinement. This configuration is a classical benchmark often used in publications and, since the refinement area is quite large, it provides a different point of view respect to the central refinement illustrated before.

Starting from the usual uniform tensor product mesh, we applied four refinement iterations where we refine at each step all the basis functions along the diagonal. As in the centre refinement case, we considered the polynomial degrees $p = 2, 3$, and 4. Again, the ghost domain was adjusted in order for the first tensor product basis to constitutes a partition of unity in Ω^0. Figure 27 shows the first three meshes in the biquadratic and bicubic cases. Note that we refined also a portion of the ghost domain: This was done in order to avoid having T-joints on the boundary of Ω^0. The integration, however, was carried out only for the elements inside Ω^0, as in the previous cases.

Sparsity Due to the extension of the refinement region, and the number of overlapping zones, we expected to see quite a difference in the sparsity pattern of the matrices produced by the different spline technologies. Figure 28 presents the sparsity patterns for $p = 2$ and $p = 4$. The results are presented in Table 7.

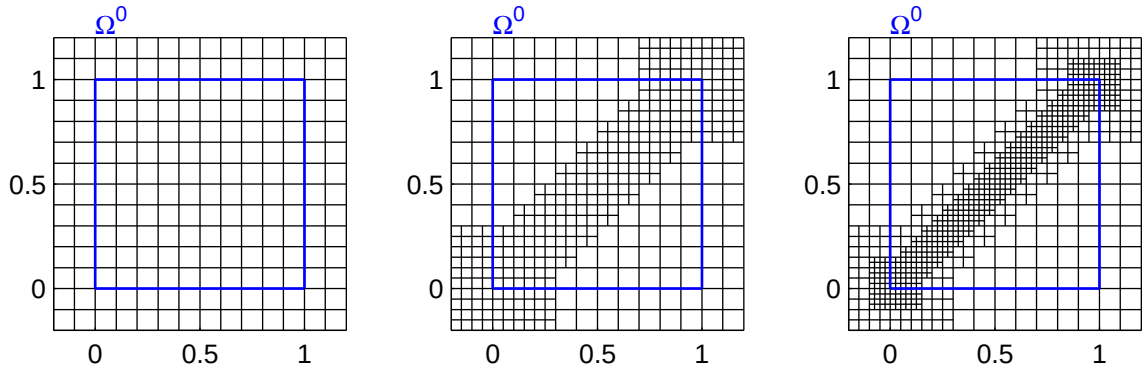
As we can see, due to the larger refined area the use of Truncated Hierarchical or LR B-splines basis functions has a huge impact on the sparsity of the matrices: The Classical Hierarchical basis produces almost twice as many non-zero elements as the Truncated basis, and more than twice those of the LR B-splines basis. It also seems that increasing the polynomial degree somewhat reduces the advantage of the Truncated basis, while the LR B-splines basis maintains the same ratio.

Conditioning Numbers Figure 29 shows the conditioning numbers of the stiffness and mass matrices obtained for this diagonal example. The numerical values of the conditioning numbers for the stiffness and mass matrices are presented in Table 8 for the biquadratic case, and Table 9 for the bicubic case.

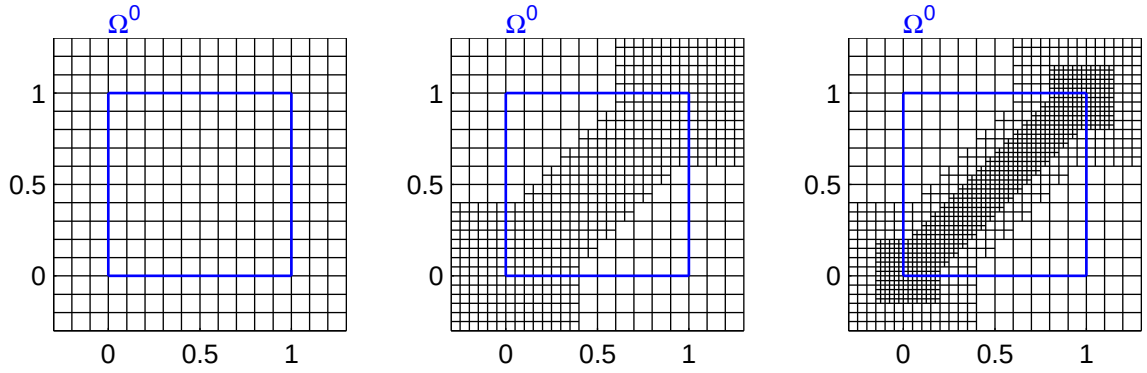
Spectrum Figures 30 and 31 present the spectrum of the stiffness and mass matrices in the cases $p = 2$ and 4, respectively. As before, the magnitude of the smallest eigenvalues is the same for all three types of basis functions considered. The value of the conditioning numbers depends therefore from the values of the highest eigenvalues, which is typically greater for the Classical Hierarchical functions.

p	HB	THB	LR	H/T	H/LR
2	116366	61330	53558	190%	217%
3	304671	164039	140047	186%	218%
4	628862	356042	287594	177%	219%

Table 7: **2D Diagonal Refinement:** Number of non-zero elements in the stiffness matrices at the last refinement iteration of the bivariate diagonal refinement. The last two columns present the ratios, rounded to the nearest percent point.



(a) Initial Mesh (b) First refinement iteration (c) Second refinement iteration



(d) Initial Mesh (e) First refinement iteration (f) Second refinement iteration

Figure 27: **2D Diagonal Refinement:** The first three steps of the refinement process in the cases $p = 2$ (above) and $p = 3$ (below).

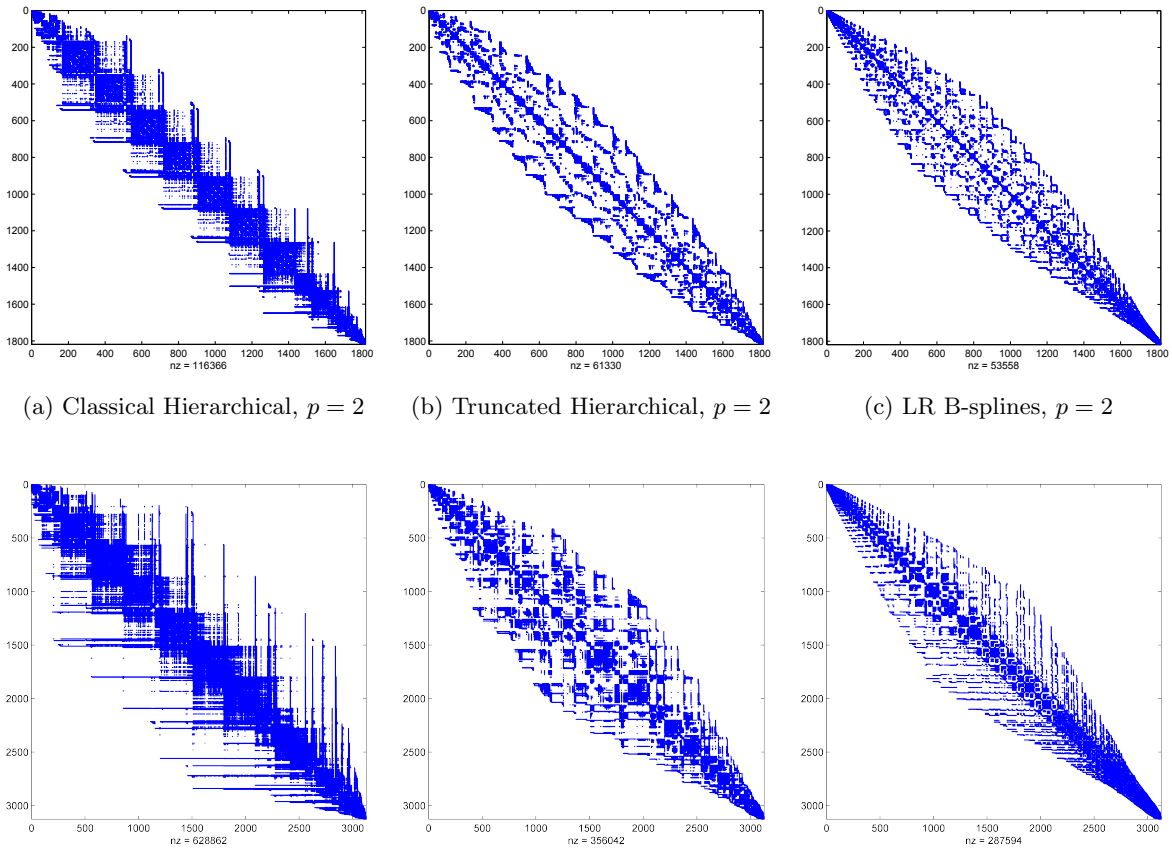
Stiffness Matrix

Iter.	0	1	2	3	4
HB	83.6793	139.708	169.439	225.641	318.5089
THB	83.6793	97.8692	173.2625	371.711	772.2455
LR	83.6793	95.6921	163.0508	346.4521	716.4894

Mass Matrix

Iter.	0	1	2	3	4
HB	2.241e+03	8.783e+03	3.511e+04	1.404e+05	5.617e+05
THB	2.241e+03	8.187e+03	3.275e+04	1.310e+05	5.240e+05
LR	2.241e+03	8.161e+03	3.264e+04	1.306e+05	5.223e+05

Table 8: **2D Diagonal Refinement:** The conditioning numbers for $p = 2$ in the various iterations of the diagonal refinement. Stiffness Matrix above, Mass Matrix below.



(a) Classical Hierarchical, $p = 2$ (b) Truncated Hierarchical, $p = 2$ (c) LR B-splines, $p = 2$

(d) Classical Hierarchical, $p = 4$ (e) Truncated Hierarchical, $p = 4$ (f) LR B-splines, $p = 4$

Figure 28: **2D Diagonal Refinement:** The sparsity patterns of the stiffness matrices at the last refinement iteration for the 2D diagonal refinement. The Cuthill-McKee Algorithm has been applied to optimize the bandwidth. Top: $p = 2$. Bottom: $p = 4$.

Stiffness Matrix

Iter.	0	1	2	3	4
HB	2.323e+04	3.661e+04	4.366e+04	4.588e+04	4.675e+04
THB	2.323e+04	2.666e+04	2.840e+04	2.927e+04	2.977e+04
LR	2.323e+04	2.522e+04	2.569e+04	2.581e+04	2.585e+04

Mass Matrix

Iter.	0	1	2	3	4
HB	1.975e+06	7.885e+06	3.160e+07	1.264e+08	5.057e+08
THB	1.975e+06	6.876e+06	2.750e+07	1.100e+08	4.400e+08
LR	1.975e+06	6.753e+06	2.701e+07	1.080e+08	4.321e+08

Table 9: **2D Diagonal Refinement:** The conditioning numbers for $p = 3$ in the various iterations of the diagonal refinement. Stiffness Matrix above, Mass Matrix below.

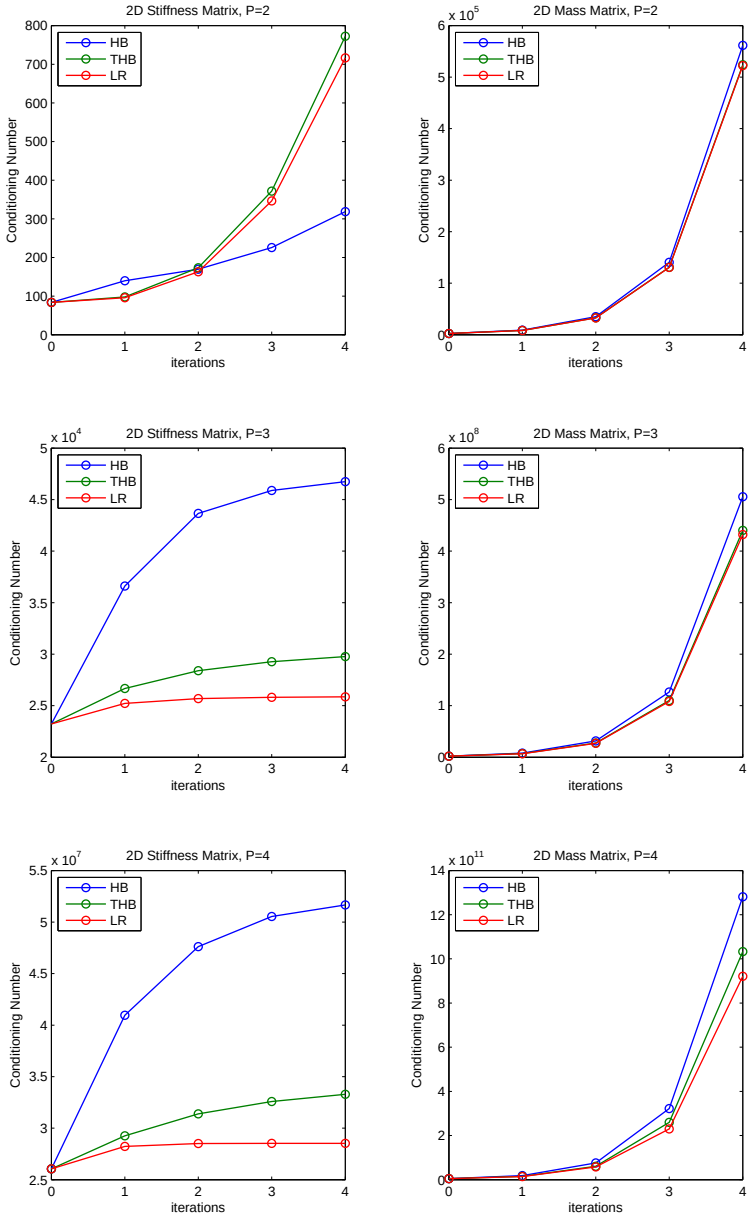


Figure 29: **2D Diagonal Refinement:** Graphs of the conditioning numbers of stiffness matrices (left column) and mass matrices (right column) for the bivariate diagonal refinement. $p = 2$ (top), $p = 3$ (middle), and $p = 4$ (bottom).

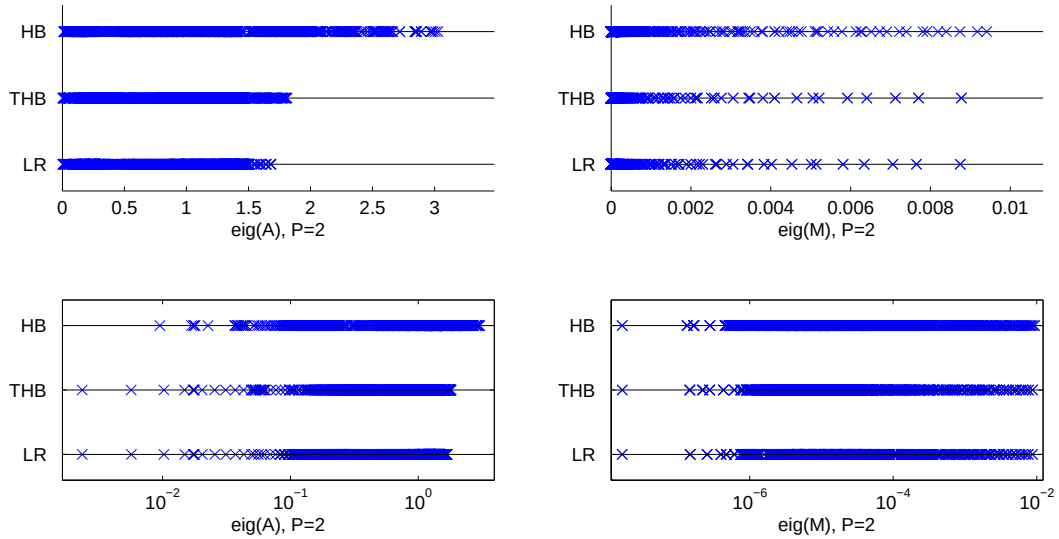


Figure 30: **2D Diagonal Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 2$ at the last iteration of the diagonal refinement. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

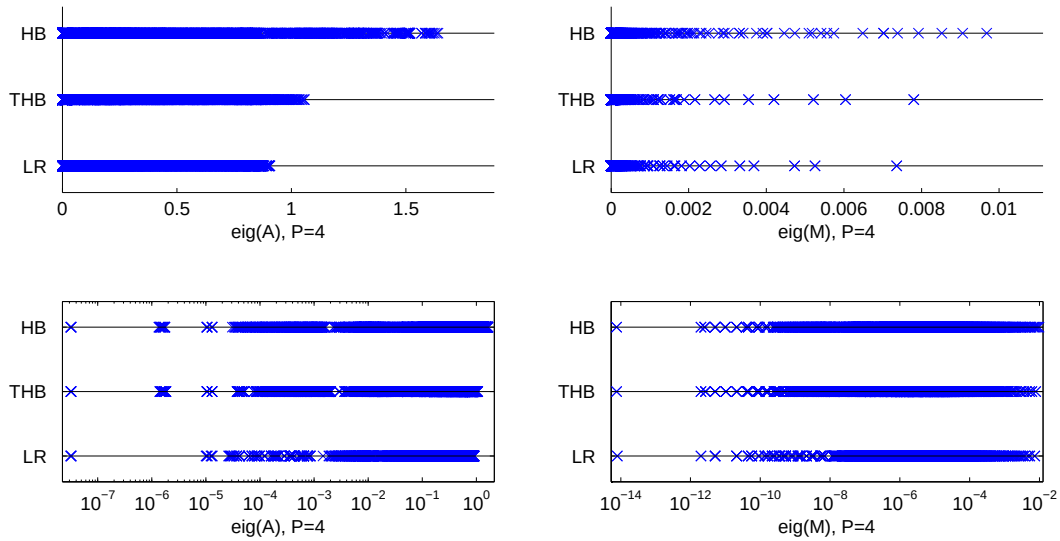


Figure 31: **2D Diagonal Refinement:** The eigenvalues of the Stiffness Matrix (left) and Mass Matrix (right) for $p = 4$ at the last iteration of the diagonal refinement. The plots are shown on a linear scale (top) and logarithmic scale (bottom). The zero eigenvalue of the Stiffness Matrix is omitted.

4.3 Additional Results

In this section we present some additional results. We feel it is unnecessary to give the same level of detail as in the previous examples, hence only the meshes and corresponding graphs of the conditioning numbers are shown. We feel that this gives a solid base of different refinement types that may appear in applications. Ranging from refinement around points, to curves to areas.

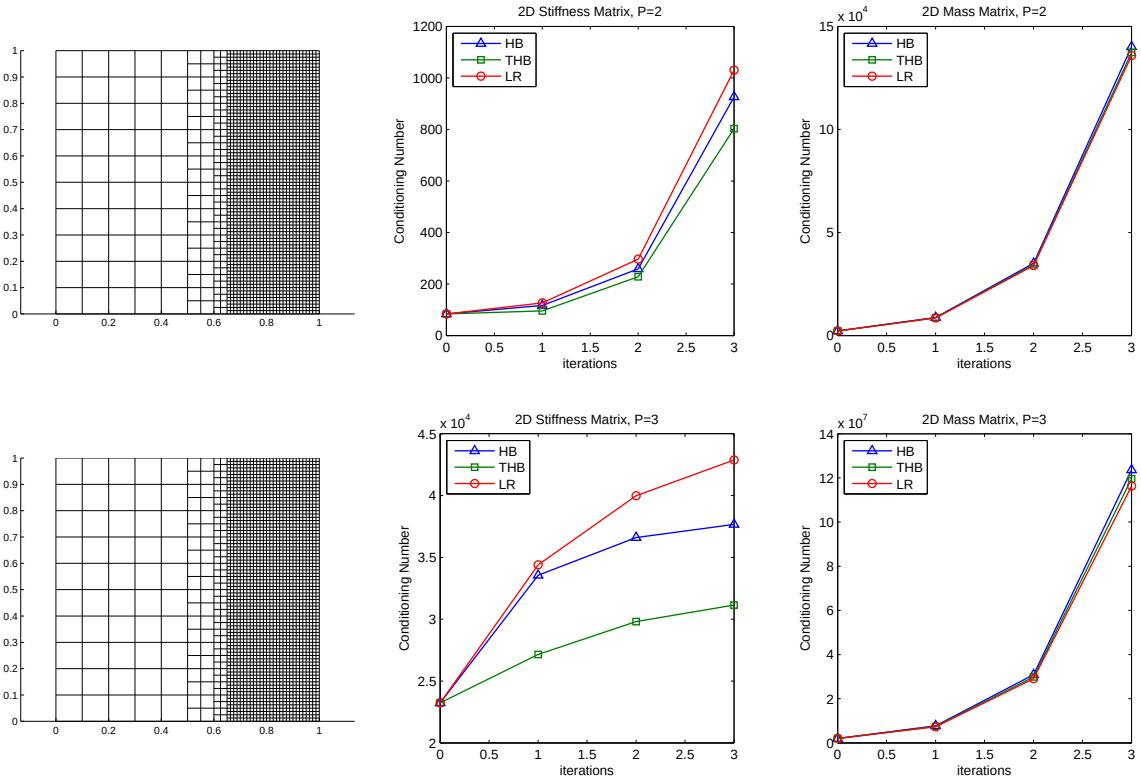


Figure 32: **Half-Side Refinement:** Refinement conducted only in the right half of the mesh. Top: Final mesh and conditioning numbers for $p = 2$. Bottom: Final mesh and conditioning numbers for $p = 3$.

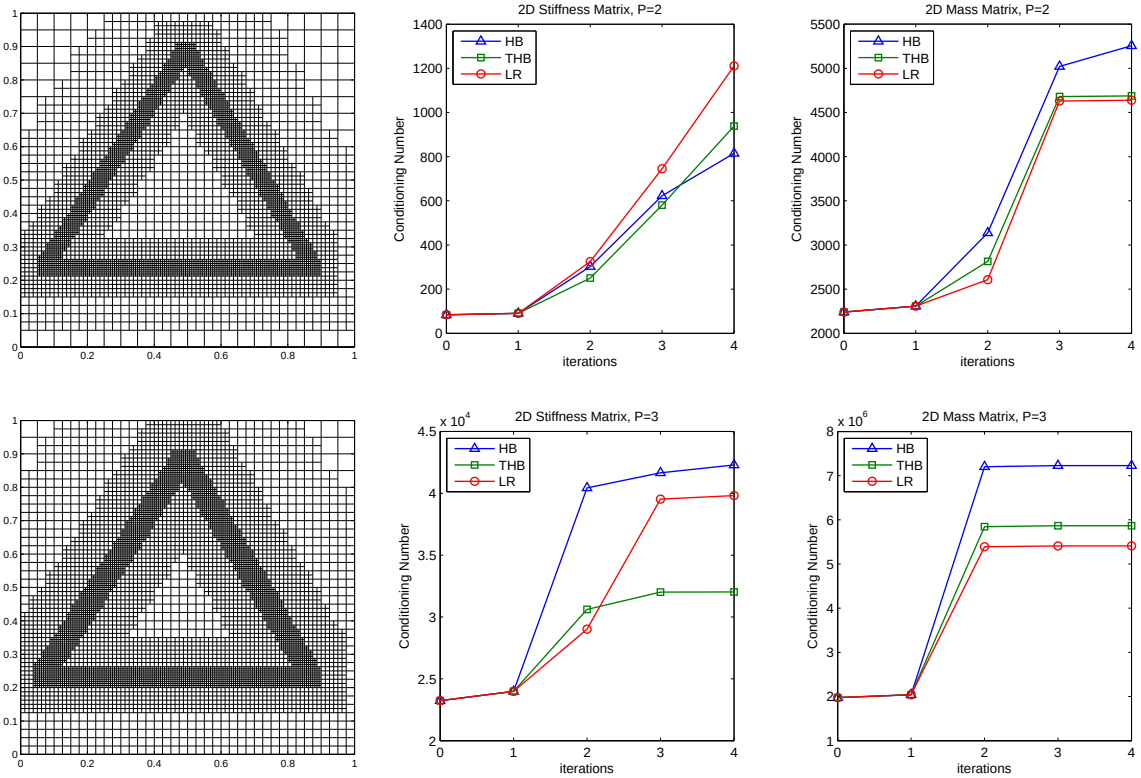


Figure 33: **Triangle Refinement:** Refinement along a triangular path. Top: Final mesh and conditioning numbers for $p = 2$. Bottom: Final mesh and conditioning numbers for $p = 3$.

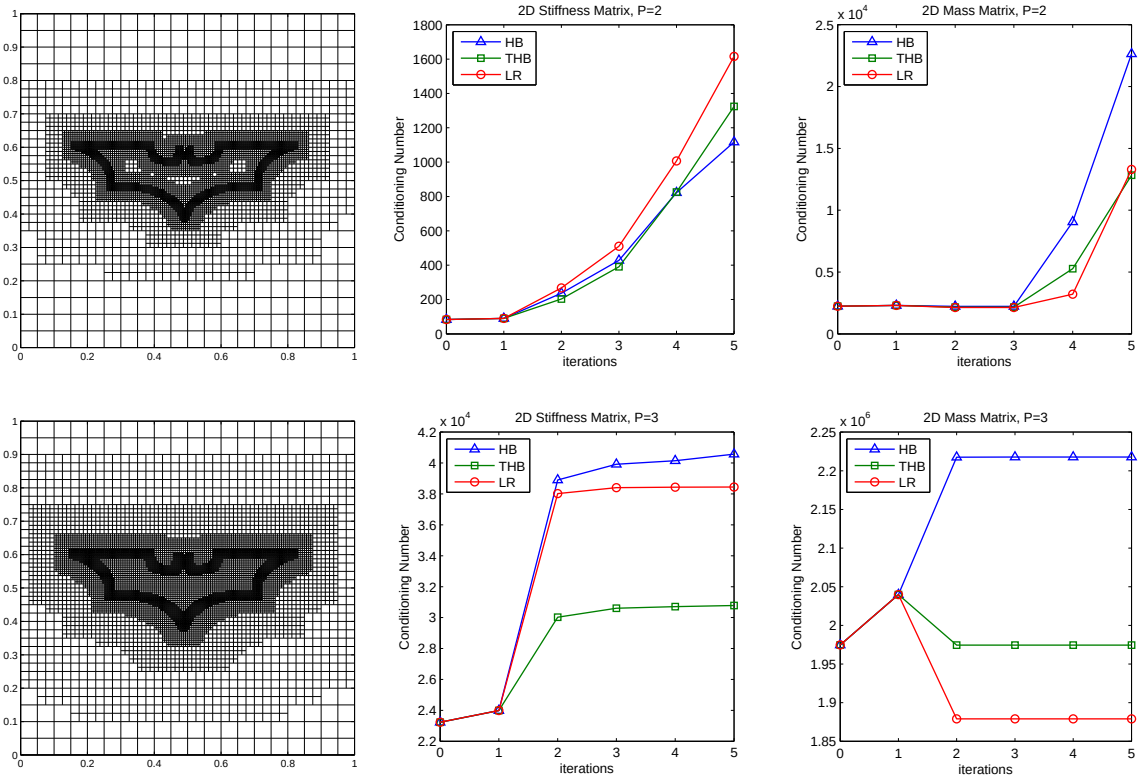


Figure 34: **Logo Refinement:** Refinement along a familiar logo. Top: Final mesh and conditioning numbers for $p = 2$. Bottom: Final mesh and conditioning numbers for $p = 3$.

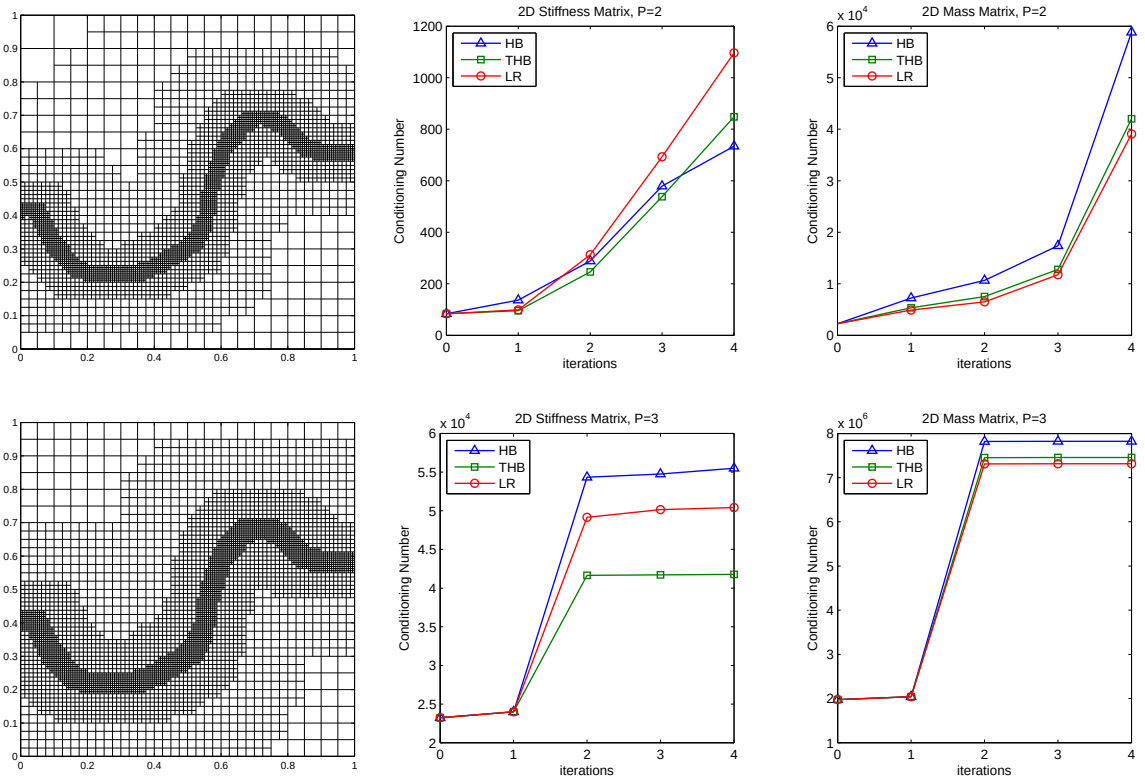


Figure 35: **Curve Refinement:** Refinement along a predefined curve. Top: Final mesh and conditioning numbers for $p = 2$. Bottom: Final mesh and conditioning numbers for $p = 3$.

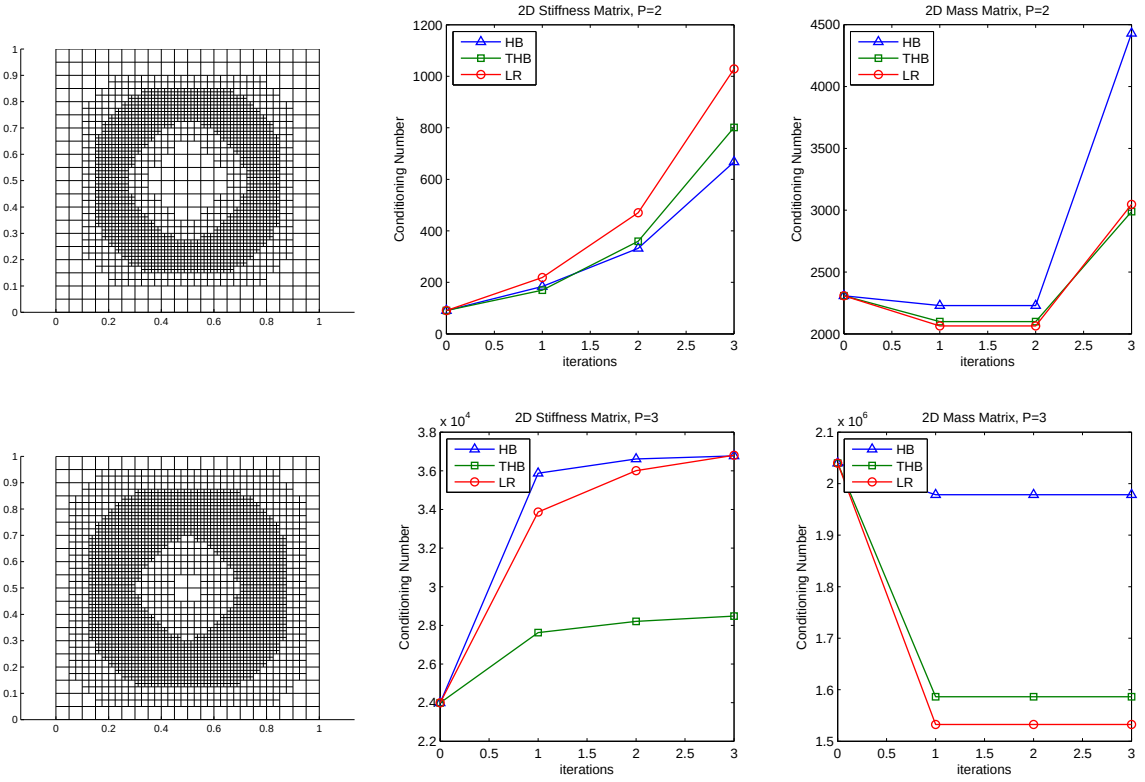


Figure 36: **Circle Refinement:** Refinement along a unit circle. Top: Final mesh and conditioning numbers for $p = 2$. Bottom: Final mesh and conditioning numbers for $p = 3$.

5 Conclusions

In this paper we have analysed the Classical Hierarchical, Truncated Hierarchical and LR B-splines basis on both a qualitative (more theoretical) and quantitative (more numerical) level. Regarding the qualitative differences we believe that the most important points are:

- The Classical Hierarchical basis does not constitute a partition of unity;
- For some meshes, the basis generated by the Hierarchical B-splines and the structured mesh LR B-spline refinement does indeed produce different function spaces;
- The Hierarchical and LR B-splines frameworks have different admissible meshes;
- While LR B-splines allow for more flexibility regarding the choice of refinement strategies, a formal proof for the linear independence of the resulting set of functions is still lacking.

The difference in the functions spaces is perhaps the most important point. Since both hierarchal and Truncated B-splines span the same space they are both resulting in the same discrete finite element solution, meaning that the differences in the basis functions are going to affect only the number of operations required to get to a certain precision. For LR B-splines versus Hierarchical B-splines, the situation becomes a bit more nuanced as the discrete solution itself can be different.

For the quantitative case we presented several numerical examples which have shown that there is a substantial difference between the three spline families especially for what concerns the sparsity pattern of the matrices. The Classical Hierarchical basis always produced the densest matrices, while those produced by the Truncated Hierarchical and LR B-splines were much more sparse. In particular, it seems that when the refinement region affects only a small portion of the mesh, the Truncated basis yields the best results regarding sparsity; if instead the refinement covers a large portion of the mesh, then the LR B-splines basis produces the most sparse matrices.

When it comes to the conditioning numbers, no clear and defined pattern emerged, and the results seemed very dependant on several factors: the dimension of the problem (1D vs 2D), the matrix considered (Stiffness Matrix vs Mass Matrix) and the polynomial degree. In particular, in the univariate setting we had, for the stiffness matrix,

$$\text{cond}(A_T) < \text{cond}(A_{LR}) < \text{cond}(A_H)$$

while for the mass matrix we had

$$\text{cond}(M_{LR}) < \text{cond}(M_T) < \text{cond}(M_H)$$

In the bivariate setting things are not so definite, but there seem to be a tendency of

$$\text{cond}(M_{LR}) \approx \text{cond}(M_T) < \text{cond}(M_H)$$

for the mass matrix, while the stiffness matrix doesn't seem to show as prominent patterns. An interesting observation is that for low p, the Classical Hierarchical have conditioning numbers of A on par, or below that of Truncated or LR, but for $p > 2$ this is no longer the case.

On a general level we can say that the Classical Hierarchical basis performed worse than the Truncated or the LR one, while these last two frameworks yielded very similar results. Therefore, we conclude that for any application where sparsity or conditioning numbers are important quantities, one of these two refinement schemes are to be preferred.

6 Acknowledgement

The authors acknowledge the financial support from the Norwegian Research Council and the industrial partners of the ICADA-project (NFR Projects 187993); Ceetron, Det Norske Veritas and Statoil.

References

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

[2] L. Beir ao da Veiga, A. Buffa, D. Cho, and G. Sangalli. Analysis-suitable t-splines are dual-compatible. *Computer Methods in Applied Mechanics and Engineering*, 249252(0):42 – 51, 2012. Higher Order Finite Element and Isogeometric Methods.

[3] 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.

[4] 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.

[5] M.J. Borden, C.V. Verhoosel, M.A. Scott, T.J.R. Hughes, and C.M. Landis. A phase-field description of dynamic brittle fracture. *Computer Methods in Applied Mechanics and Engineering*, 217220(0):77 – 95, 2012.

[6] 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.

[7] S. Brenner and L.R. Scott. *The mathematical theory of finite element methods*. Springer, 2005.

[8] A. Bressan. Some properties of LR-splines. *Comput. Aided Geom. Design*, 30(8):778–794, 2013.

[9] 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(0):112–126, 2012.

[10] A. Buffa, D. Cho, and G. Sangalli. Linear independence of the T-spline blending functions associated with some particular T-meshes. *Computer Methods in Applied Mechanics and Engineering*, 199(23-24):1437 – 1445, 2010.

[11] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & Sons, 2009.

[12] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th National Conference*, ACM '69, pages 157–172, New York, NY, USA, 1969. ACM.

[13] 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.

[14] 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.

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

[16] K.P.S. Gahalaut and S.K. Tomar. Condition number estimates for matrices arising in the isogeometric discretizations. Technical Report 23, RICAM, 2012.

[17] Carlo Garoni, Carla Manni, Francesca Pelosi, Stefano Serra-Capizzano, and Hendrik Speleers. On the spectrum of stiffness matrices arising from isogeometric analysis. *Numerische Mathematik*, 127(4):751799, 2014.

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

[19] C. Giannelli, B. Jüttler, and H. Speleers. Strongly stable bases for adaptively refined multilevel spline spaces. *Advances in Computational Mathematics*, 40:459–490, 2014.

[20] Carlotta Giannelli and Bert Jüttler. Bases and dimensions of bivariate hierarchical tensor-product splines. *Journal of Computational and Applied Mathematics*, 239(0):162–178, 2013.

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

[22] 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.

[23] K.A. Johannessen, T. Kvamsdal, and T. Dokken. Isogeometric analysis using LR B-splines. *Computer Methods in Applied Mechanics and Engineering*, 269:471–514, 2014.

[24] K.A. Johannessen, T. Kvamsdal, and M. Kumar. Divergence-conforming discretization for stokes problem on locally refined meshes using LR B-splines. *Submitted to Computer methods in applied mechanics and engineering*, 2014.

[25] C. Johnson. *Numerical solutions of partial differential equations by the finite element method*. Dover Publications, 2009.

[26] G. Kiss, C. Giannelli, and B. Jüttler. Algorithms and data structures for truncated hierarchical bsplines. Technical Report 14, Johannes Kepler University, 2012.

[27] R. Kraft. *Adaptive und linear unabhängige Multilevel B-Splines und ihre Anwendungen*. PhD thesis, Stuttgart, 1998.

[28] M. Kumar, T. Kvamsdal, and K.A. Johannessen. Superconvergent patch recovery and a posteriori error estimation technique in adaptive isogeometric analysis. *Submitted to Computer methods in applied mechanics and engineering*, 2014.

[29] T. Kvamsdal and K.M. Okstad. Error estimation based on superconvergent patch recovery using statically admissible stress fields. *International Journal for Numerical Methods in Engineering*, 42(3):443–472, 1998.

[30] X. Li and M.A. Scott. Analysis-suitable t-splines: Characterization, refineability, and approximation. *Mathematical Models and Methods in Applied Sciences*, 24(06):1141–1164, 2014.

[31] H. Melbø and T. Kvamsdal. Goal oriented error estimators for stokes equations based on variationally consistent postprocessing. *Computer methods in applied mechanics and engineering*, 192(5):613–633, 2003.

[32] Dominik Mokriš, Bert Jüttler, and Carlotta Giannelli. On the completeness of hierarchical tensor-product B-splines. *Journal of Computational and Applied Mathematics*, 271(0):53–70, 2014.

[33] B. Mourrain. On the dimension of spline spaces on planar T-meshes. *Math. Comp.*, 83(286):847–871, 2014.

[34] N. Nguyen-Thanh, H. Nguyen-Xuan, S.P.A. Bordas, and T. Rabczuk. Isogeometric analysis using polynomial splines over hierarchical t-meshes for two-dimensional elastic solids. *Computer Methods in Applied Mechanics and Engineering*, 200(2122):1892 – 1908, 2011.

[35] K.M. Okstad and T. Kvamsdal. Object-oriented programming in field recovery and error estimation. *Engineering with Computers*, 15(1):90–104, 1999.

[36] K.M. Okstad, T. Kvamsdal, and K.M. Mathisen. Superconvergent patch recovery for plate problems using statically admissible stress resultant fields. *International journal for numerical methods in engineering*, 44(5):697–727, 1999.

[37] A. Quarteroni. *Numerical models for differential problems*. Springer, 2008.

[38] 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.

[39] 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.

[40] 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.

[41] T.W. Sederberg, D.L. Cardon, G.T. Finnigan, N.S. North, J. Zheng, and T. Lyche. T-spline simplification and local refinement. *ACM Transactions on Graphics*, 23(3):276–283, 2004.

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

[43] A. Stahl and T. Kvamsdal. Post-processing and visualization techniques for isogeometric analysis results. *Submitted to Computer methods in applied mechanics and engineering*, 2014.

[44] C.V. Verhoosel, M.A. Scott, R. de Borst, and T.J.R. Hughes. An isogeometric approach to cohesive zone modeling. *International Journal for Numerical Methods in Engineering*, 87(1-5):336–360, 2011.

[45] C.V. Verhoosel, M.A. Scott, T.J.R. Hughes, and R. de Borst. An isogeometric analysis approach to gradient damage models. *International Journal for Numerical Methods in Engineering*, 86(1):115–134, 2011.

[46] 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.

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

[48] Y. Wang, J. Zheng, and H. S. Seah. Conversion between T-splines and hierarchical B-splines. In *Proceedings of the Eight IASTED International Conference*, Computer Graphics and Imaging, Honolulu, Hawaii, USA, August 2005.