Hierarchical B-splines for the adaptive solution of one-dimensional advection-diffusion problems

Final project for the course of
Matematica Numerica, 2019-2020

Author:
Ariel S. Boiardi [*]

Supervisor:
Prof.ssa Alessandra Aimi

Dipartimento di Scienze Matematiche Fisiche e Informatiche
Università degli Studi di Parma

In this work we present an adaptive discretization method for one dimensional advection-diffusion problems based on hierarchical B-splines. After reviewing the basic theory of hierarchical B-splines and their use in FEM and similar methods, we develop algorithms for the implementation of said technique. Numerical examples are provided to test the algorithms and their MATLAB implementations.

1 Introduction

Adaptive methods have been developed in the framework of finite element methods (FEM) since the seventies and have now reached a defining place in the theory and practice of FEM [16]. *A posteriori* error estimators are an essential ingredient for adaptivity, allowing to make judicious local refinements of the discretization, based on computable estimates of the numerical error.

Spline functions and B-splines stemmed from approximation theory, in which they play a fundamental role. Their usage in FEM-like discretization methods was proposed in the nineties [9] and has found a renewed interest in recent years with the introduction of Isogeometric Analysis (IgA) [10]. In this new framework, the usage of B-splines (and similar) as *finite elements* is aimed to bridge the gap between computer aided design (CAD) and finite elements analysis (FEA). In addition it has been shown that the use of smoother basis functions leads in many cases to better convergence properties [17].

B-splines do not provide a natural and effective way for local refinement in the multivariate case [14]. Univariate B-splines can be refined by knot insertion, but the resulting

[*]Email address: `arielsurya.boiardi@studenti.unipr.it`

basis can become strongly non-uniform and even lose smoothness.

Hierarchical B-splines were introduced by R. H. Bartles and D. Forsey [6] in order to allow for a finer local control of surfaces in geometry modelling. Their approach takes advantage of the excellent refinability properties of *uniform* B-splines by resolving finer details of the geometry with the superimposition of patches of coarser and finer B-spline surfaces. An alternative construction for HB-splines was given a decade later by R. Kraft [11]. Following an *ad hoc* selection procedure, this construction leads to a *basis* for the hierarchical splines space, which makes them a viable option as a discretization space in FEM-like methods.

In this work we present HB-splines following the definitions in [7, 17]; we then introduce a model advection-diffusion problem and its discretization with HB-splines. We recall error estimates for advection-diffusion problems discussed in [1] and adapt them to the novel HB-spline techniques. All the theoretical discussion will be coupled with data structures and algorithms for the implementation. Finally computational examples will be discussed to test the proposed methods on some benchmark problems. Numerical simulations were carried out with MATLAB implementations presented in appendix B.

2 Hierarchical B-splines

In this section we present some basic theory of B-splines and their refinement properties. We then introduce hierarchical B-splines (HB-splines) and discuss some properties useful for FEM and FEM-like methods.

2.1 B-splines

A space of univariate B-splines of degree p is uniquely defined by its *knots sequence*, which is associated to a non-decreasing partition of the domain Ω of the B-splines:

$$a = x_1 < x_2 < \ldots < x_n = b. \tag{1}$$

Every break point can be repeated up to $p+1$ times in the knots sequence. If m_i denotes the multiplicity of the i-th break point x_i as a knot and $M = \sum_{i=1}^{n} m_i$, the knot sequence can be written as

$$\Xi = \begin{bmatrix} \xi_1 & \cdots & \xi_{N+p+1} \end{bmatrix} \tag{2}$$

where $N = M - p - 1$ and since the knot sequence counts M knots, and every B-spline is supported on $p+2$ knots, the dimension of the B-spline space on the knots Ξ is exactly N.

The B-spline basis on the knots Ξ can be constructed through the Cox-De Boor recursion [3, 4]:

$$\begin{aligned} B_{i,0}(x) &= \begin{cases} 1 & x \in [\xi_i, \xi_{i+1}] \\ 0 & \text{otherwise,} \end{cases} \\ B_{i,p}(x) &= \frac{x-\xi_i}{\xi_{i+1}-\xi_i} B_{i,p-1}(x) + \frac{\xi_{i+p+1}-x}{\xi_{i+p+1}-\xi_i} B_{i+1,p-1}(x). \end{aligned} \tag{3}$$

Many definitions for spline spaces and B-splines are possible, but we consider (3) as a definition, because it emphasises many features of B-splines that will be used in the following, and provides a very efficient way to compute values of B-splines [15]. The B-splines defined in (3) span a space of splines $\mathcal{S}$ defined by Ξ, whose regularity is $\mathcal{C}^{p-m_i}$ at x_i. In order to preserve the partition on unity property of the B-splines on the whole domain, we will consider the first and last knots with multiplicity $p+1$. The basis for cubic B-splines is represented in fig. 1 for a given knot vector.

Derivatives of B-splines of order up to $k \leq \min_i \{p - m_i\}$ can also be computed recursively with the following formula [13]:

$$B_{i,p}^{(k)}(x) = p \left(\frac{B_{i,p-1}^{(k-1)}(x)}{\xi_{i+p} - \xi_i} - \frac{B_{i+1,p-1}^{(k-1)}(x)}{\xi_{i+p+1} - \xi_{i+1}} \right), \tag{4}$$

where fractions with zero denominator are considered to be zero, as also the i-th B-spline of degree $p-1$ is zero if the i-th knot is repeated p times.

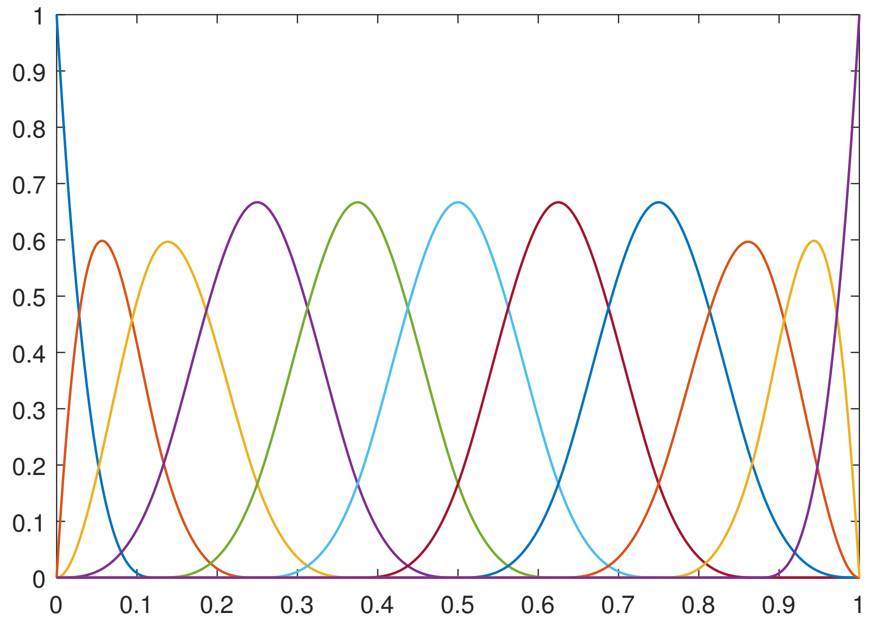


Figure 1: Cubic B-spline basis constructed on the uniformly spaced knot vector $\Xi = \left[\, 0\ 0\ 0\ 0\ 0.1250\ 0.2500\ 0.3750\ 0.5000\ 0.6250\ 0.7500\ 0.8750\ 1\ 1\ 1\ 1 \,\right]$.

2.1.1 Implementation aspects

An efficient way to build a B-spline basis is to store the degree p and the knot sequence Ξ as a vector. The dimension of the space follows, being $\dim(\mathcal{S}) = N$. Every B-spline is then identified by its index, which coincides with the index of the first knot of its support in the knot vector Ξ. Every B-spline is efficiently evaluated at any point with (3). Note that we decided to keep indices starting from 1 for the sake of clarity in the transition to our MATLAB implementations.

2.2 Refinement and subdivision of B-splines

B-splines are more *flexible* in regions of the domain where the knots are more dense, the refinement of the knot vector is therefore a necessary procedure in order to be able to resolve small details in B-spline based approximation.

We will consider a knot vector Θ obtained by dyadic refinement of the knot vector Ξ, dividing the interval between two knots in halves. The B-spline space obtained by dyadic refinement will be the space $\mathcal{S}_\Theta$ based on the refined knot vector. Being a refinement means that the new space is also capable to express every B-spline of the *coarser* space $\mathcal{S}_\Xi$ by the following two scale relation [9]:

$$B_{i,p}^{(\Xi)}(x) = \frac{1}{2^p} \sum_{k=0}^{p+1} \binom{p+1}{k} B_{2i+k-p-1,p}^{(\Theta)}(x), \tag{5}$$

for $i = p+1, \ldots, N-p$. The original B-spline is said to be *parent* of its $p+2$ children, which are the B-splines of the finer space whose indices are in the summation range in (5). Clearly every parent B-spline is not linearly independent of its children. The reconstruction of a B-spline from the basis in fig. 1 by its children is shown in fig. 2.

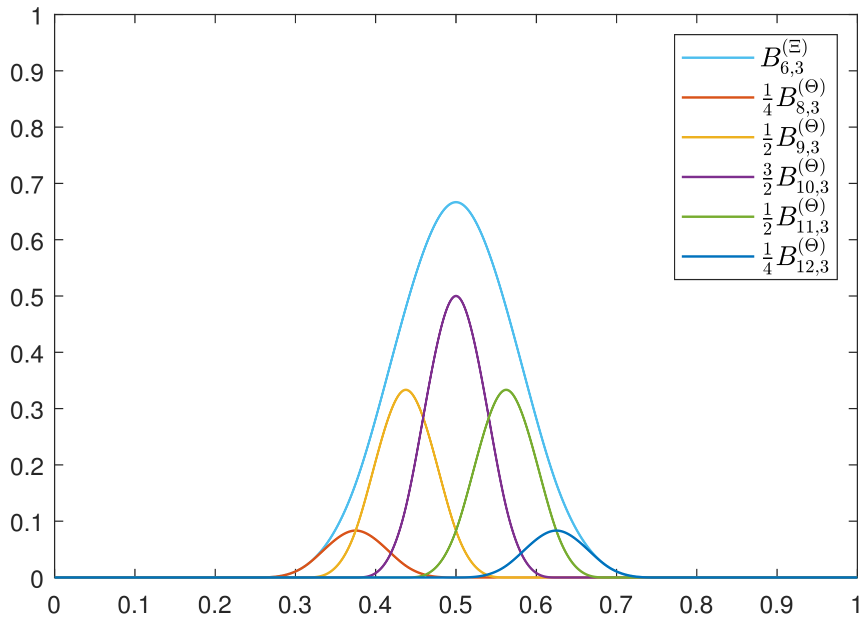


Figure 2: Subdivision of the 6-th B-spline on the Ξ knot vector in weighted sum of its children built on the knot vector Θ obtained by dyadic refinement of Ξ.

2.3 Hierarchical B-splines (HB-splines)

A hierarchical B-spline space can be thought as a nested sequence of uniform B-spline spaces, in which only some B-splines are considered in every level.

2.3.1 Nested spaces and domains

We consider a nested sequence of B-spline spaces of degree p

$$\mathcal{S}_1 \subset \mathcal{S}_2 \subset \ldots, \tag{6}$$

determined by their respective knot vectors. For $l \in \mathbb{N}$, the B-spline basis corresponding to $\mathcal{S}_l$ is denote by

$$\mathcal{B}_l := \left\{\beta_i^l \mid i = 1, \ldots, N_l\right\}, \tag{7}$$

where N_l is the dimension of the space $\mathcal{S}_l$. Furthermore we denote by $\mathcal{Q}_l$ the grid of level l associated to the partition of the domain defined by the knot vector of $\mathcal{S}_l$, and we call $Q \in \mathcal{Q}_l$ a *cell* of level l.

In the following we consider every level being obtained by dyadic refinement of the previous one as explained in section 2.2, and the *two scale relation* [7] between successive levels, which gives the expression of a B-spline of level l as a linear combination of B-splines of the finer $l+1$-th level,

$$\beta_i^l = \sum_{k=1}^{N_{l+1}} c_{k,l+1}\left(\beta_i^l\right) \beta_k^{l+1} \tag{8}$$

is given by (5).

Definition 1. For $n \in \mathbb{N}$ we consider a *hierarchy of subdomains* of depth n as

$$\Omega = \Omega_1 \supset \Omega_2 \supset \ldots \subset \Omega_n \subset \Omega_{n+1} = \emptyset \tag{9}$$

which defines the nested domains of the HB-splines. $\Omega_1 = \Omega$ is the domain of definition if the HB-spline space, and every subdomain Ω_l, for $l \geq 2$, is union of some cells of level $l-1$.

2.3.2 The HB-spline basis

With the previous notation, we define the basis $\mathcal{H}$ of the hierarchical space $\mathcal{S}^{\mathcal{H}}$ as follows:

Definition 2. Let $\{\mathcal{S}_l\}_l$ a sequence of spaces like (6) with the corresponding bases $\{\mathcal{B}_l\}_l$, and $\{\Omega_l\}_l$ a hierarchy of subdomains of depth n like (9). The hierarchical B-spline basis $\mathcal{H}$ is defined by taking $\mathcal{H} = \mathcal{H}_{n-1}$ in the following recursion

$$\begin{cases} \mathcal{H}_1 & := \mathcal{B}_1, \\ \mathcal{H}_{l+1} & := A_{l+1}^l \cup A_{l+1}^{l+1}, \quad l = 1, \ldots, n-2, \end{cases} \tag{10}$$

where

$$A_{l+1}^{l} := \{\beta \in \mathcal{H}_l \mid \operatorname{supp}(\beta) \not\subset \Omega_{l+1}\}\,,$$
$$A_{l+1}^{l+1} := \{\beta \in \mathcal{B}_{l+1} \mid \operatorname{supp}(\beta) \subset \Omega_{l+1}\}\,.$$

The initialization step in (10) selects all the basis functions of the underlying space $\mathcal{S}_1$. In the recursive step we first add to $\mathcal{S}^{\mathcal{H}}$ the B-splines of the previous level whose support is not entirely contained in Ω_{l+1} (A_{l+1}^{l}), then Ω_{l+1} is covered with A_{l+1}^{l+1}, i.e. by basis functions in $\mathcal{B}_{l+1}$ which are added to the hierarchical space at level $l+1$. An example of resulting basis $\mathcal{H}$ is shown in fig. 3.

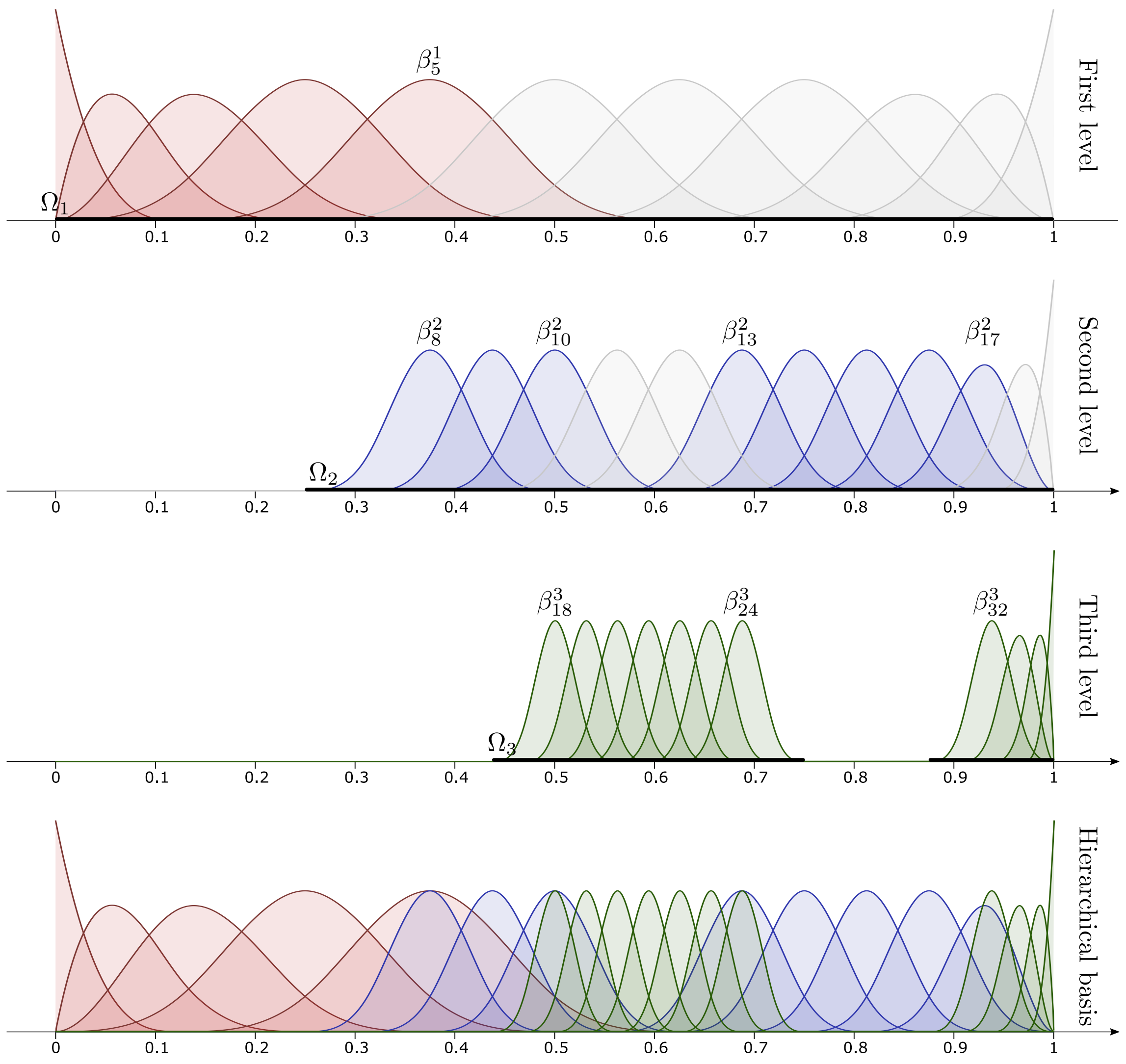


Figure 3: Hierarchical cubic B-spline basis of three levels. Deactivated functions are greyed out from each level.

Definition 3. We say that a B-spline β is *active* if $\beta \in \mathcal{H}$. In particular $\beta \in \mathcal{S}^{\mathcal{H}}$ is an active function of level l if $\beta \in \mathcal{A}_l := \mathcal{H} \cap \mathcal{B}_l$, and it is a *deactivated* function of level l if $\beta \in \mathcal{D}_l := \mathcal{H}_l \setminus \mathcal{H}_{l+1}$.

The hierarchical basis $\mathcal{H}$ is associated to an underlying *hierarchical mesh* $\mathcal{Q}$ given by

$$\mathcal{Q} := \bigcup_{l=1}^{n-1} \{Q \in \mathcal{Q}_l \mid Q \subset \Omega_l,\ Q \not\subset \Omega_{l+1}\}. \tag{11}$$

As for basis functions, Q is said to be an *active cell* if $Q \in \mathcal{Q}$ and it is an active cell of level l if $Q \in \mathcal{Q} \cap \mathcal{Q}_l$.

Linear independence of the hierarchical basis follows from its construction:

Lemma 1. *The functions in $\mathcal{H}$ are linearly independent.*

Proof. Consider a linear combination of functions of the hierarchical basis that gives zero, and rearrange it according to the level of the functions:

$$0 = \sum_{\beta \in \mathcal{H}} c_\beta \beta = \sum_{l=1}^{n-1} \left[\sum_{\beta \in \mathcal{H} \cap \mathcal{B}_l} c_\beta \beta \right].$$

Since $\mathcal{H} \cap \mathcal{B}_l \subset \mathcal{B}_l$, the functions in $\mathcal{H} \cap \mathcal{B}_l$ are linearly independent. Notice that only these functions have support on $\Omega_l \setminus \Omega_{l+1}$, and by linear independence $c_\beta = 0$ for $\beta \in \mathcal{H} \cap \mathcal{B}_l$. Repeating for $l = 1, \dots, n-1$ we have linear independence of the hierarchical basis. $\square$

As can be seen from definition 2, every iteration of the constructive procedure (10) induces a local refinement of the hierarchical basis [17]:

Lemma 2. *Let $\mathcal{H}_1, \dots, \mathcal{H}_{n-1}$ the hierarchical B-spline bases considered in definition 2. We have that*

$$\operatorname{span}(\mathcal{H}_l) \subset \operatorname{span}(\mathcal{H}_{l+1}), \quad l = 1, \dots, n-2. \tag{12}$$

With this construction, refinement seems possible only within an already refined region, since the hierarchy of subdomains (9) which select the regions of refinement at every level is in fact decreasing. As proved in [17], refinement is possible in any region as enlarging a domain $\Omega_{\bar{l}}$ of the hierarchy (9) produces an enlarged sequence of hierarchical bases, constructed by the procedure (2). However the implementation to achieve this flexibility is rather complicated, and since adaptive methods should take just a handful of iteration to be advantageous we will limit our scope to purely hierarchical refinement. In other word we will only be able to refine in smaller and smaller regions. The obvious drawback of this choice is a limited (or no) ability to compensate for a bad refinement step.

2.3.3 Implementation aspects

In order to represent a HB-spline basis for a hierarchical spline space one needs the degree of the splines, the sequence of B-spline bases (7), represented - as outlined in section 2.1.1 - by their respective knot vectors, and a (structured) list of active functions at each level. However, for the sake of a more natural notation in the codes and lighter computations, at the expense of higher memory usage, our implementation uses a bigger data structure, resumed in table 1.

Table 1: Properties of the data structure to represent a HB-spline basis.

Property	Description
`deg`	Degree of the splines.
`dim`	Dimension of the space.
$\{\ \texttt{hknots}_l\ \}$	Indices of active knots of the hierarchical mesh.
$\{\ \texttt{hcells}_l\ \}$	Indices of active elements of the hierarchical mesh.
`nlev`	Depth of the hierarchical space, i.e. total number of levels.
$\{\ \mathcal{A}_l\ \}$	Active functions by level.
$\{\ \mathcal{D}_l\ \}$	Deactivated functions by level.
$\{\ \mathcal{S}_l\ \}$	B-spline space from the underlying sequence for each level.

The order in the HB-spline basis is not the same as that in the classical B-spline basis, but it needs to count also for the subdivision between levels, in particular we consider the basis functions to be ordered according to their level and within the same level with the usual B-spline basis ordering. If $u \in \text{span}(\mathcal{H})$, u is expressed by a linear combination of basis function with the following form:

$$u = \sum_{l=1}^{\texttt{nlev}} \left[\sum_{k \in \mathcal{A}_l} u_{k,l} \beta_k^l \right], \tag{13}$$

where $u_{k,l}$ is the coefficient of u with respect to the k-th B-spline of the l-th level of the hierarchical space.

3 Adaptive hierarchical B-spline approximation of a model problem

Adaptive methods are largely used in FEM as a mean of obtaining finer approximations while keeping a low number of degrees of freedom (DoFs), particularly for the approximation of functions characterised by harsh local behaviours. Some simple adaptive FEM methods for the numerical solution of advection-diffusion problems have been the subject of my Bachelor's thesis [1], from which some results will be referenced in the following.

A general adaptive procedure can be sketched as the iteration of the following steps:

$$\cdots \text{ Solve} \rightarrow \text{Estimate} \rightarrow \text{Refine } \cdots \tag{14}$$

until a stopping criterion is satisfied [5, 12]. Every module of this procedure will be discussed in the following for a model problem.

In this section we introduce advection-diffusion problems, toward which our work is oriented, and develop some FEM-like discretization techniques using the HB-spline spaces introduced in the previous section as approximation spaces. We then describe the adaptive refinement procedures and adjust some of the results from the FEM technique studied in [1] to the novel HB-spline framework. Finally we present some algorithms for the implementation of the adaptive procedure.

3.1 The advection-diffusion model problem and its discretization

The one-dimensional stationary advection-diffusion equation is a second-order boundary value problem of the form

$$\begin{cases} -\mu u'' + bu' = f, \quad x \in \Omega := (0,1) \\ u(0) = u_0, \ u(1) = u_1. \end{cases} \tag{15}$$

The solution of this problem for $f = 0$ and $u_0 = 0$, $u_1 = 1$, which is easily derived analytically, as shown in fig. 4 presents a *boundary layer* at $x = 1$ that becomes steeper and steeper as the ration $\frac{b}{\mu}$ increases. This problem has been chosen as a benchmark to test our local refinement techniques because local singular behaviours, characterised by steep gradients and harsh variations are exactly what makes adaptive methods more compelling than classical uniform h-refinement methods in FEM [1].

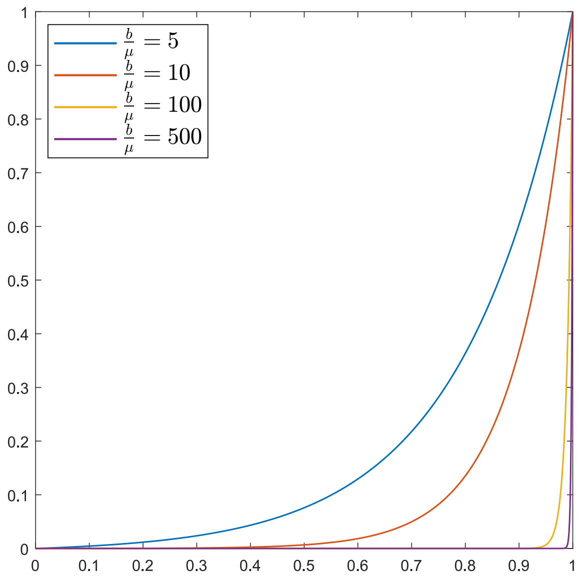


Figure 4: Exact solution of the advection-diffusion problem for some choices of $\frac{b}{\mu}$.

3.1.1 Weak form of the problem

The weak form of the advection diffusion problem (15) was derived in [1]:

$$\text{Find } u \in \mathrm{H}_0^1(\Omega) \mid a\left(u,v\right) = \mathcal{F}\left(v\right) \ \ \forall v \in \mathrm{H}_0^1(\Omega), \tag{16}$$

where $a\left(\cdot,\cdot\right)$ is the bilinear form associated to the differential problem in (15)

$$a\left(u,v\right) = \int_\Omega \mu u'v' - bu'vdx, \tag{17}$$

and the functional $\mathcal{F}\left(\cdot\right)$ depends on boundary conditions of the problem and on the source term f in the right hand side of the ODE in (15):

$$\mathcal{F}\left(v\right) = \int_\Omega fvdx - a\left(B,v\right), \tag{18}$$

where B is built to express the boundary conditions in (15), in the sense that $B(0) = u(0)$ and $B(1) = u(1)$. Note that B can be any function that represents the boundary conditions, and its choice in $\mathrm{H}^1(\Omega)$ or other function spaces is arbitrary, but can make calculations much more convenient or possible at all. Given the solution $u \in \mathrm{H}_0^1$ of the weak problem (16), the solution to the problem with non homogeneous boundary conditions as in (15) is given by $\tilde{u} = u + B$.

As proved in [1], the weak problem (16) has a unique solution by Lax-Milgram's Lemma, since the bilinear form (17) is continuous end coercive on $\mathrm{H}^1(\Omega)$ [2].

3.1.2 HB-spline discretization

Let us now approximate the weak problem (16) in the finite-dimensional space

$$\mathcal{S}_0^{\mathcal{H}} = \operatorname{span}\left(\beta \in \mathcal{H} \mid \beta|_{\partial\Omega} = 0\right), \tag{19}$$

of hierarchical splines that vanish on the boundary of the domain. The discrete form of the weak problem (16) is the following *Galerkin equation*:

$$\text{Find } u_h \in \mathcal{S}_0^{\mathcal{H}} \mid a\left(u_h,v\right) = \mathcal{F}\left(v\right) \quad \forall v \in \mathcal{S}_0^{\mathcal{H}}. \tag{20}$$

A basis for $\mathcal{S}_0^{\mathcal{H}}$, be it

$$\mathcal{H}_0 := \mathcal{H} \cap \mathcal{S}_0^{\mathcal{H}},$$

is clearly given by $\mathcal{H}$, excluded the first and last B-splines, that is those whose support contains the first or the last knot of their respective knots vectors (see fig. 3). Note that identifying the said *first* and *last* basis functions do not immediately correspond to some indices, but need to be looked for though all levels, as described in algorithm 3.

Expressing the approximated solution u_h with respect to the basis $\mathcal{H}_0$ as in (13), and testing it against all basis functions with (20), ordered by level, we get to the following system of linear equations:

$$\sum_{l=1}^{\texttt{nlev}} \sum_{k\in\mathcal{A}_l} u_{k,l} a\left(\beta_k^l, \beta\right) = \mathcal{F}\left(\beta\right) \quad \forall \beta \in \mathcal{A}_\lambda, \ \forall \lambda = 1, \ldots, \texttt{nlev} \tag{21}$$

The stiffness matrix associated to the discretization of the problem can be described by blocks, where every block represents the interaction between two levels of the hierarchical basis:

$$A = [A_{\lambda,l}]_{\substack{\lambda = 1,\dots,\mathtt{nlev} \\ l = 1,\dots,\mathtt{nlev}}} \quad \text{where} \quad [A_{\lambda,l}]_{j,k} = \left[a\left(\beta^l_k, \beta^\lambda_j\right)\right]_{\substack{j \in \mathcal{A}_\lambda \\ k \in \mathcal{A}_l}} . \tag{22}$$

The load vector of the system on the other hand accounts for the source f and for the interaction with the boundary and is expressed as

$$F = [F_\lambda]_{\lambda=1,\dots,\mathtt{nlev}} \quad \text{where} \quad [F_\lambda]_j = \left[\mathcal{F}\left(\beta^\lambda_j\right)\right]_{j\in\mathcal{A}_\lambda} . \tag{23}$$

Note that, alas, the stiffness matrix is not tridiagonal as in the FEM case, nor banded as in a non hierarchical B-spline discretization. The matrix given by the HB-spline discretization, described in (22), will only present a symmetric sparsity pattern, strongly dependent on the choice of order between and within the levels of the hierarchical basis. We will nonetheless take advantage of the sparse nature of the stiffness matrix in order to optimize the computation of its elements, as described in algorithm 1. An efficient computation of the load vector is described in algorithm 2.

3.2 Error estimates

The refinement of the approximation space needs to be guided by some local estimate of the approximation error, which quantifies how well the solution obtained with the discretization (21) can represent the exact solution of the weak-form problem (16).

In [1] a simple residual based *a posteriori* local error estimator for one-dimensional advection-diffusion problems has been proposed for adaptive finite elements methods. As shown therein, the global approximation error $e_h := |u - u_h|$ can be estimated by a sum of local contributions:

$$\|e_h\|_{\mathrm{H}^1(\Omega)} \le \frac{1}{\alpha}\eta = \frac{C}{\alpha}\left(\sum_{Q\in\mathcal{Q}} \eta_Q^2\right)^{\frac{1}{2}} , \tag{24}$$

where C is an unknown constant, α is the coercivity constant of the bilinear form $a(\cdot,\cdot)$, and the *local residual* η_Q is a local estimate for the error on the element Q of the mesh (11) defined as:

$$\eta_Q = \mathrm{meas}\,(Q)\left\|f + \mu\left(B'' + u_h''\right) - b\left(B' + u_h'\right)\right\|_{L^2(Q)} , \tag{25}$$

where f is the source in the differential equation ($f = 0$ in (15)), B is a function that represent the boundary conditions and u_h is the approximate solution of (16). It has also been shown in [1] by numerical evidence that η_Q can represent the local distribution of the approximation error element by element.

The active cells of the hierarchical space are structured in levels, similarly we consider a *hierarchical residual estimate*

$$\mathtt{heta} = \begin{bmatrix}\eta_1 & \cdots & \eta_{\mathtt{nlev}}\end{bmatrix} , \tag{26}$$

containing the local error estimates

$$[\eta_l]_j = [\eta_{Q_j}]_{Q_j \in \mathcal{Q} \cap \mathcal{Q}_l} \tag{27}$$

of the residuals on the active cells of each level.

In our implementation only the contribution of the deepest level is computed at every iteration and added to the global error estimate, in place of the contributions on the deactivated cells as described in algorithm 4. Even if the refinement is local it affects the solution on the whole domain, therefore algorithm 4 does not exactly compute the local error estimate. At every iteration of the adaptive procedure, the computed value $\eta_{\mathtt{nlev}}$ reflects the actual value of the local residuals as from (25), while the previous levels reflect the value of η_l given by (25) with respect to coarser approximations of u.

This partial evaluation of the local residuals is faster and proves to be more useful than the complete exact evaluation as will be discussed in section 3.3.3. Moreover local error estimates relative to previous levels have no use in the adaptive procedure, since our error estimator is only used to represent the distribution of the error.

3.3 Adaptive hierarchical refinement

Once the local error is estimated, it its possible to choose which elements need refinement according to a *marking strategy*: B-splines are selected to be refined and are replaced by finer B-splines which are added to a new level of the hierarchical space. The refinement is carried on respecting definition 2. The problem is then approximated in the refined space, and the procedure is repeated until a stopping criterion is satisfied.

3.3.1 Marking strategies

The main difference between FEM and B-spline discretization is that FEM is centred on the geometric elements of the mesh, while B-splines methods are centred on basis functions. This basic difference becomes more apparent in the marking procedure, when elements or functions need to be selected for refinement. Since the local and global error estimates proposed in section 3.2, together with their properties, are based on the geometric discretization of FEM, our procedure will mark the elements and then select for refinement the B-splines supported on marked elements.

The cell marking if performed following the *equilibrium strategy* by Dörfler [5], coupled with a *premarker* (as suggested in [16] and discussed in [1]), and a *threshold marker*, which marks for refinement the elements where the local relative residual is higher than a fixed threshold γ.

3.3.2 Local refinement procedure

After a subset of the basis B-splines of the last hierarchical level, be l, have been selected for refinement according to the marking strategies outlined in section 3.3.1, the union of their supports becomes the refinement region Ω_{l+1}, and a finer $l+1$-th level is added to the hierarchical space according to the recursion in definition 2.

The new level will be a subset of the basis of a B-spline space created inserting knot that split every non-empty knot interval into halves, as in algorithm 8. In the new levels are activated the B-splines that are children of marked functions of the previous level; and in order to keep linear independence, we need to deactivate functions from previous level that can be expressed as linear combinations of newly introduced B-splines. The `refine` procedure in algorithm 5 assembles all the steps of the refinement.

3.3.3 Stopping criteria

Every iteration of the adaptive algorithm requires to solve the discrete problem and to evaluate the local residual (25), which adds some very relevant computational costs. In order to get a useful adaptive algorithm it is then necessary to optimize as much as possible the number of iterations and the increase of DoFs at every iterations: if too many DoFs are added, one gets a procedure which is very similar to uniform refinement (with the added costs due to local error and hierarchical basis); on the other hand adding too few DoFs will require a lot of iterations of the procedure to get useable results. As is often the case, *in medio stat virtus*, and the balance is a subtle game left to the sensitivity of the analyst.

Since our implementation only allows for refinement at the deepest level of the hierarchy, it is often impossible to meet requirements on global error estimates, which still are not really significant in a numerical sense, since the constant C in (24) is unknown, and might be different at every iteration. Finer stopping criteria need then to be set in place, in order to stop the procedure at the optimal point. In the rest of this section we propose two stopping criteria that seem reasonable to us.

Best possible improvement Refinement is only performed within the deepest level, therefore one could - and it is often the case - build a hierarchical space in which the possible improvement is only marginal compared to the overall accuracy.

The best theoretical - and indeed optimistic - improvement attainable by refinement a given hierarchical space is reached if all residuals from the deepest level become zero. Therefore, the best relative improvement with respect to the last computed global residual estimate η coincides with the relative error due to the last level

$$\eta^{\text{rel}}_{\texttt{nlev}} := \frac{\eta_{\texttt{nlev}}}{\eta}. \tag{28}$$

If the best relative improvements is less than a significant fraction - say 10% - the refinement of the hierarchical space is no longer advantageous. It is now clear why the partial evaluation of the residual highlighted in section 3.2 makes sense: the partial evaluation of `heta` gives the exact residual on the last level and allows to compare it with other *first attempt* on other levels, that were considered *good enough* by previous iterations of the algorithm. In other words, since the approximation available on the deepest level is a *first attempt* in the new space, it makes sense to compare it with other - *good* - first attempts.

Iteration improvement We - hope to - expect an improvement at every iteration of the procedure. If the improvement gained with one iteration with respect to the previous one is not satisfying, i.e. less than a fixed threshold - typically 3% - the procedure is stopped.

3.4 Algorithms for the implementation

One of the main parts of this work was the development of algorithms for the implementation of the described adaptive hierarchical B-spline discretization techniques.

An adaptive algorithm has been sketched in a very general yet expressive form in (14), and we will not provide a complete algorithm, which would be a mere assembly of the components. Algorithms for the main components of the procedure are instead listed in appendix A. We tried to keep algorithms abstract and symbolic; for this reason some procedures described here might not find a direct counterpart in the MATLAB implementations (appendix B). The rationale was to keep the algorithms as language-agnostic as possible.

3.4.1 Algorithms for the solution

The first step of every iteration of the adaptive procedure is the solution of the discrete problem in the hierarchical space. One crucial step is therefore to evaluate the stiffness matrix (22) and the load vector(23).

An efficient evaluation of the stiffness matrix is performed with the procedure `assembly` in algorithm 1. The main idea is to initialize the matrix A with zeroes and then evaluate only the cells which give a non-zero. Following the same idea the load vector (23) is computed by the procedure `load_eval` in algorithm 2.

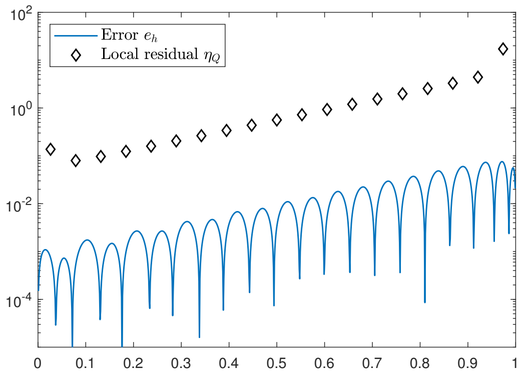
4 Computational examples

In this section we test our adaptive hierarchical B-spline discretization method to numerically solve three one-dimensional boundary value problems in the advection-diffusion form (15) on the domain $\Omega = (a, b)$, whose general weak form was discussed in section 3.1.1. Initial approximations spaces are constructed on uniformly spaced knot vectors. We generally used cubic B-splines, if not differently specified. All the results were obtained with the MATLAB codes in appendix B.

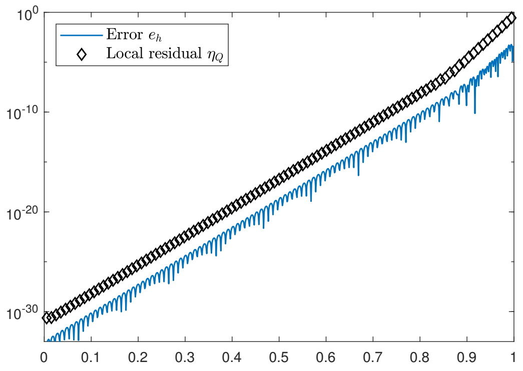
4.1 Model advection diffusion problem

This problem was presented in (15), with non homogeneous boundary conditions. To represent the non-homogeneous boundary condition $u(1) = 1$, we introduce a boundary operator $B = \beta^{\mathtt{L}BD}_{\mathtt{last}}$ containing the right-hand boundary basis function, as found by algorithm 3. The exact solution of the problem, plotted in fig. 4, has a very steep boundary layer around $x = 1$ which becomes more steep as the ratio between advection and diffusion parameters $\frac{b}{\mu}$ is increased.

Since the solution we are approximating has a local singularity, we hope that our error estimator, which proved to be good in [1], can represent the error distribution also for B-spline approximation. In fig. 5 the local residual distribution is compared with the point evaluation of the error. We see that the estimated error, although more pessimistic, is able to represent the distribution of the actual error on the domain.



(a) 20 DoFs



(b) 100 DoFs

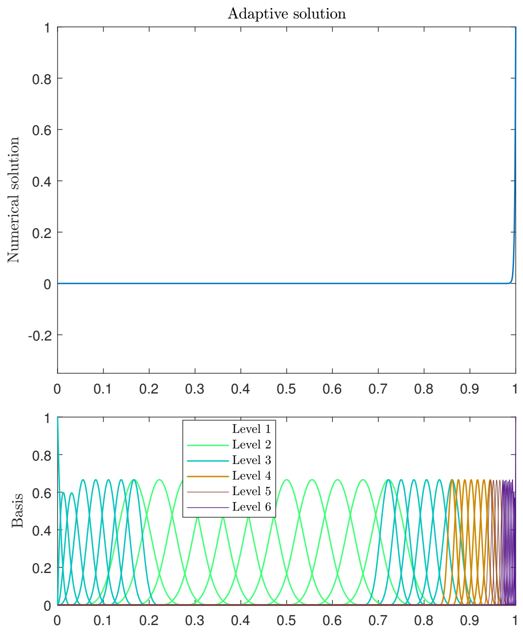
Figure 5: Pointwise error and local residual estimates for the advection diffusion problem with $\frac{b}{\mu} = 100$ approximated in uniform cubic B-splines. Residuals are plotted in the middle of the cell on which it is evaluated.

The numerical solution obtained with the hierarchical adaptive method with parameters in table 2 is plotted in fig. 6a with the hierarchical basis of the approximation space. As expected the hierarchical basis is much more dense around the boundary layer. The approximate adaptive solution can represent much better the local singularity, as from the comparison with fig. 6b, in which numerical instabilities can be observed around the boundary layer.

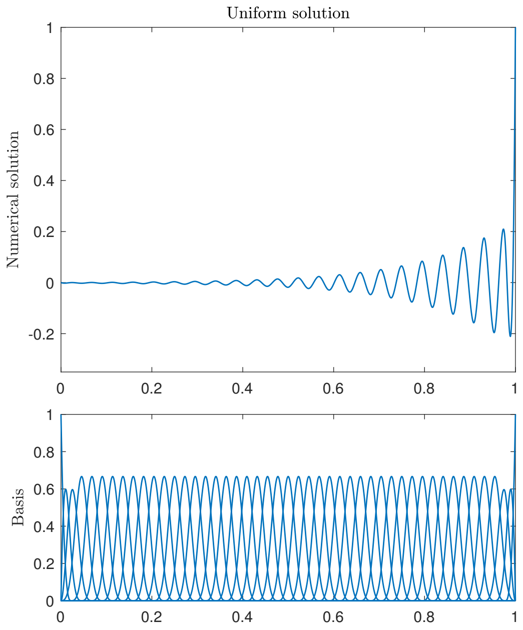
Table 2: Basic parameters for the adaptive solution of the advection diffusion problem. Parameters are only specified if different from this table.

Parameter	Value
Initial DoF	40
`minPercImpr`	5
`minPercIterImpr`	1
`maxRelResLoc`	1
θ	0.5
`PreMark`	`true`
`PreMarkPerc`	3

The same results as fig. 6 can be observed in the fifth mark of the convergence plot in fig. 7b, which also shows that our stopping criterion has worked perfectly in this test,



(a) Adaptive approximation and basis of the approximation space. The initial approximation was built in a uniform B-spline space with 10 DoFs, $\theta = 0.25$, `maxRelResLoc`=0.5 and no premarking. The solution was found in 6 iteration of the adaptive procedure with 45 DoFs. The procedure was stopped by low improvement in the last iteration.



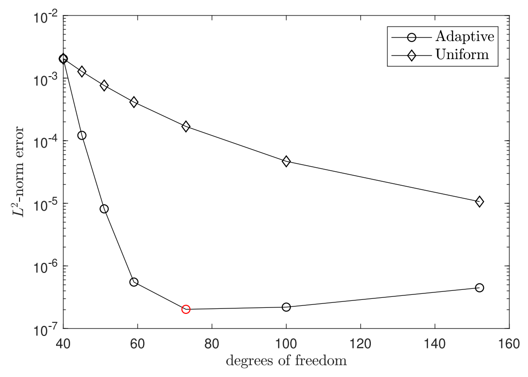
(b) Numerical solution on uniform B-spline space with 45 DoFs.

Figure 6: Comparison between uniform and adaptive numerical solution of the advection diffusion problem with $\frac{b}{\mu} = 500$ for the same number of degrees of freedom. The L^2-norm error in the adaptive solution is $1.903\,03 \times 10^{-4}$, while uniform solution with 76 DoFs attained an L^2 error of $0.052\,895\,4$ and still shows strong instabilities.

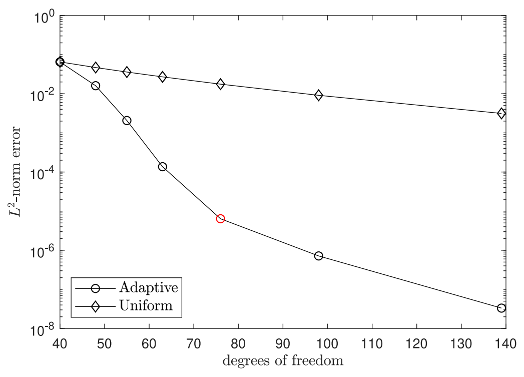
as convergence slows reduces after the fifth iteration. The same can be said for the test with $\frac{b}{\mu} = 100$ in fig. 7a, where a stopping criterion is satisfied at best possible iteration. This plot also shows how important stopping criteria are in adaptive methods based on a posteriori error estimates.

Convergence plots fig. 7 also show that the increase in the number of degrees of freedom is higher at every iteration. One way to avoid this phenomenon is to disable premarking, since it tends to accelerate the growth of the number of DoF, as shown in fig. 8. This also might require to adjust some parameters in the stopping criteria or marking strategies to get optimal results. For example fig. 8b shows that with no premarking the procedure stops because the estimated possible improvement is less than 5%, and to get another iteration (which improves the approximation) we need to restrict the stopping conditions.

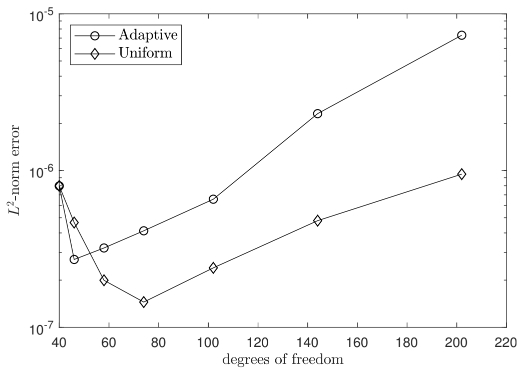
The comparison of convergence plots in figs. 7 and 8 shows, as expected, that the adap-



(a) $\frac{b}{\mu} = 100$



(b) $\frac{b}{\mu} = 500$



(c) $\frac{b}{\mu} = 10$

Figure 7: Convergence plot of uniform and adaptive method for advection-diffusion problem. Red mark indicates that a stopping criterion is satisfied.

tive method is advantageous with respect to uniform refinement only when the solution of the problem has a strong local singularity. First reason is that if the singularity is not very localised, the hierarchical refinement quickly builds a HB-spline space which can only be refined in regions where refinement is no more beneficial. The reason why adaptive approximation in fig. 7c loses accuracy faster than the uniform one might also be due to the worse conditioning of the stiffness matrix (see fig. 9).

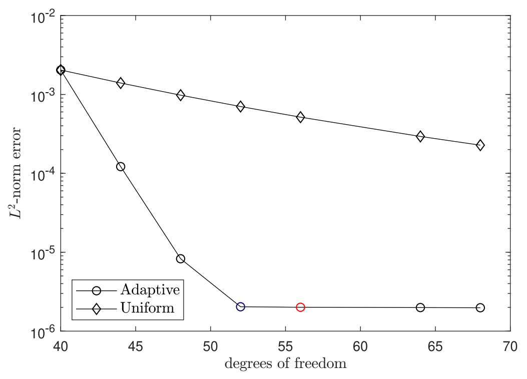
4.2 Inner layer problem

For this example we build a problem of form (15) whose solution is a modified Runge function

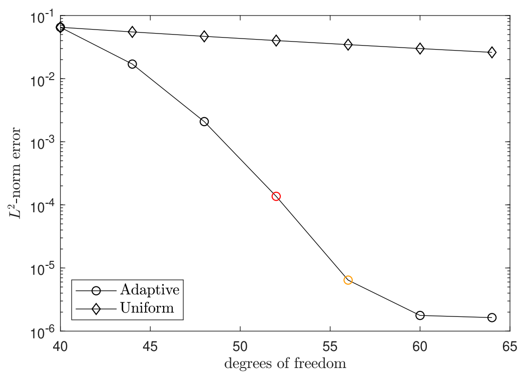
$$u(x) = \frac{1}{1+\alpha\left(x-\frac{1}{2}\right)^2} - \frac{1}{1+\frac{\alpha}{4}}, \tag{29}$$

on the domain $\Omega = (0,1)$. The solution plotted in fig. 10.

The construction of the differential problem with such solution and its weak formulation were discussed in [1] and do not add much to the present work, since the machinery of the

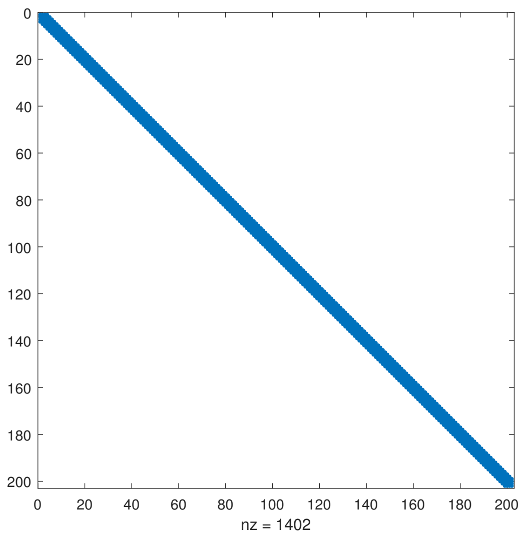


(a) $\frac{b}{\mu} = 100$

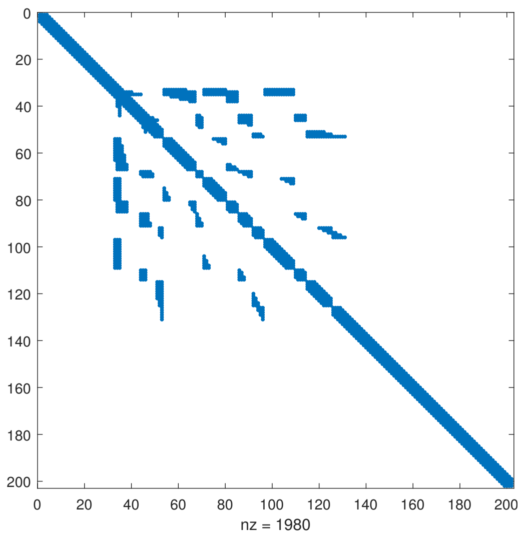


(b) $\frac{b}{\mu} = 500$

Figure 8: Convergence plot of uniform and adaptive method for advection-diffusion problem without premarking. Coloured markers indicate that a stopping criterion is satisfied: red marker for parameters in table 2, orange if `minPercImpr` $=2$ and blue if `minPercImpr` $=7$.



(a) Uniform, $\text{cond}\,(A) = 3.2057 \times 10^3$.



(b) Adaptive, $\text{cond}\,(A) = 1.3354 \times 10^4$.

Figure 9: Sparsity patterns of stiffness matrices from the last iteration in fig. 7c.

HB-spline adaptive methods only depend on some parameters of the problem. For this example $b = \mu = 1$ and boundary conditions are homogeneous. The source in (15) turns the out to be

$$f_\alpha(x) = \frac{\alpha(3-2x)}{\left[\alpha\left(x-\frac{1}{2}\right)^2+1\right]^2} - \frac{2\alpha^2(2x-1)^2}{\left[\alpha\left(x-\frac{1}{2}\right)^2+1\right]^3}, \tag{30}$$

but since is can be generated within MATLAB with the symbolic engine or using any computer algebra system, we will skip this step in the next examples.

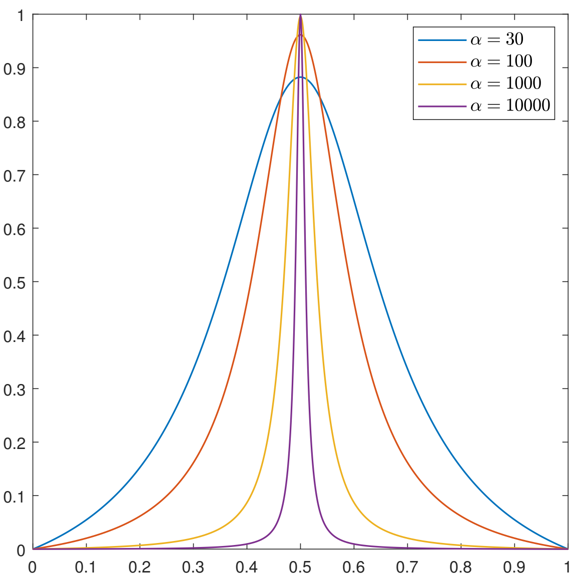
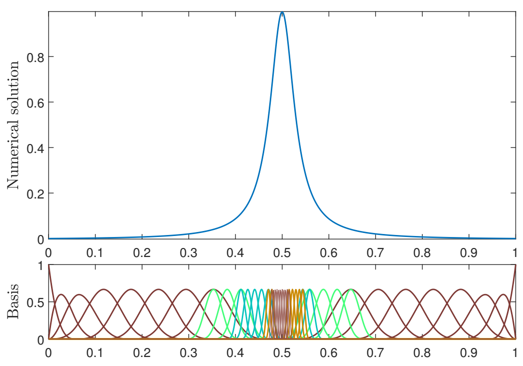
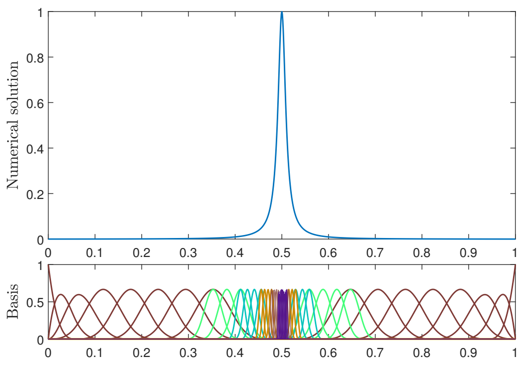


Figure 10: Exact solution (29) for the problem with inner layer varying the parameter α.

Numerical solutions for this problem are plotted in fig. 11; the region where refinement is concentrated is, correctly, that around the inner layer, both for stronger and weaker gradients.



(a) $\alpha = 1000$, adaptive procedure stopped after 5 iteration with 46 DoF and L^2-error 0.000 017. Corresponding uniform solution has L^2-error 0.004 767.



(b) $\alpha = 10000$, adaptive procedure stopped after 6 iteration with 53 DoF and L^2-error 0.000 008. Corresponding uniform solution has L^2-error 0.016 557.

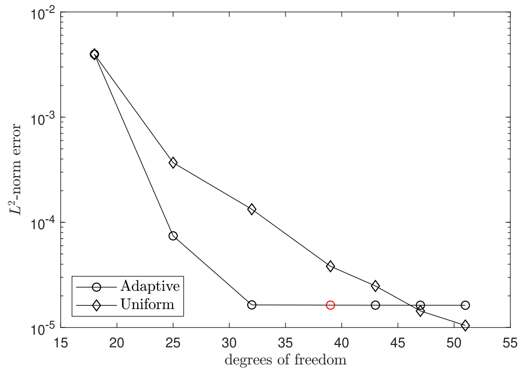
Figure 11: Adaptive numerical solution for the problem with Runge-like solution. Solver parameters in table 3.

Convergence plots in fig. 12 show again that the adaptive procedure gives best results when the solution to approximate has stronger local gradients. Nonetheless, even for

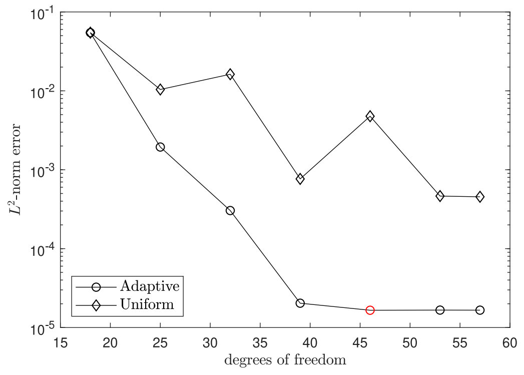
Table 3: Basic parameters for the adaptive solution of the inner layer problem.

Parameter	Value
Initial DoF	18
`minPercImpr`	5
`minPercIterImpr`	1
`maxRelResLoc`	1
θ	0.15
`PreMark`	`false`

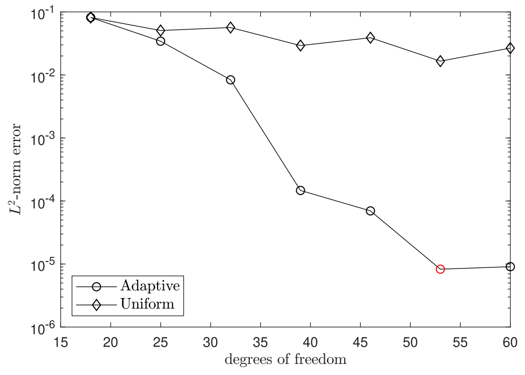
relatively smooth solution, like the case in the convergence plot fig. 12a, a couple of adaptive steps can still be a valuable way to accelerate convergence. This is a feature of HB-spline based adaptive methods that does not appear in the results obtained with FEM in [1], in which adaptivity is clearly non competitive with uniform refinement for smooth solutions.



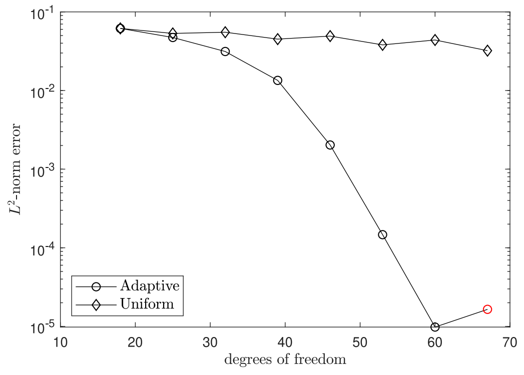
(a) $\alpha = 100$



(b) $\alpha = 1000$



(c) $\alpha = 10000$



(d) $\alpha = 100000$

Figure 12: Convergence plot of uniform and adaptive method for inner layer problem. Red mark indicates that a stopping criterion is satisfied.

Table 4: Comparison of L^2-errors of adaptive and uniform solver varying parameters from table 3.

Changed parameters	Final DoF	L^2-error	L^2-error uniform
$\alpha = 100$			
$\theta = 0.5$	42	$1.172\,990\,60 \times 10^{-5}$	$2.916\,768\,48 \times 10^{-5}$
$\theta = 0.75$	50	$7.513\,315\,03 \times 10^{-6}$	$1.112\,886\,50 \times 10^{-5}$
`PreMarkPerc`=5	52	$1.145\,510\,90 \times 10^{-5}$	$1.972\,617\,23 \times 10^{-5}$
Initial DoF=30	48	$3.700\,329\,36 \times 10^{-6}$	$1.323\,652\,77 \times 10^{-5}$
Initial DoF=40	62	$7.162\,599\,27 \times 10^{-6}$	$5.200\,805\,65 \times 10^{-6}$
$\alpha = 1000$			
$\theta = 0.5$	49	$7.821\,556\,58 \times 10^{-6}$	$4.112\,069\,60 \times 10^{-4}$
$\theta = 0.75$	45	$7.424\,798\,48 \times 10^{-6}$	$2.377\,792\,31 \times 10^{-3}$
`PreMarkPerc`=3	53	$8.427\,282\,73 \times 10^{-6}$	$4.637\,570\,54 \times 10^{-4}$
$\alpha = 10000$			
$\theta = 0.75$	71	$2.576\,105\,61 \times 10^{-5}$	$7.642\,952\,70 \times 10^{-3}$
`PreMarkPerc`=1	56	$8.613\,692\,67 \times 10^{-6}$	$2.969\,682\,96 \times 10^{-2}$
Initial DoF=30	65	$1.264\,646\,21 \times 10^{-5}$	$9.954\,047\,64 \times 10^{-3}$
$\alpha = 100000$			
`PreMarkPerc`=2	107	$8.046\,978\,08 \times 10^{-5}$	$1.963\,798\,39 \times 10^{-2}$
$\theta = 0.75$	77	$3.866\,227\,80 \times 10^{-5}$	$2.844\,169\,46 \times 10^{-2}$
Initial DoF=30	72	$1.536\,166\,91 \times 10^{-5}$	$3.978\,598\,54 \times 10^{-2}$
Initial DoF=400	421	$2.262\,852\,01 \times 10^{-5}$	$8.101\,536\,74 \times 10^{-5}$

Another remarkable difference with the adaptive FEM results in [1] is that the new HB-spline adaptive methods converges much faster. For example, to obtaine the same result as the one in the last marker of fig. 12d, the AFEM method would still require 9 iterations and roughly 650 DoF, almost ten times the degrees of freedom required for the HB-spline method. Unfortunately this does not necessarily make HB-splines more efficient since for linear finite elements the stiffness matrix was computed explicitly and almost no numerical quadrature was required.

This advantage still does not mean that the HB-spline adaptive method can obtain any accuracy. In fact too many iterations of the adaptive procedure do not necessarily converge, as from many of the convergence plots show so far; on the other hand, starting with a very fine initial approximation space, makes adaptivity almost useless, as in the last row of table 4.

Table 5: Parameters for the solver for the step function problem.

Parameter	Value
Initial DoF	17
`minPercImpr`	10
`minPercIterImpr`	5
`maxRelResLoc`	5
θ	0.15
`PreMark`	`false`

4.3 Step function problem

The next example is inspired by Gibbs phenomenon [8], we test our method to numerically solve a problem which has as solution a regularized step function

$$u(x) = \frac{1}{1 + e^{-2kx}} \quad k \in \mathbb{N} \tag{31}$$

on $\Omega = (-1, 1)$. As in the previous example $b = \mu = 1$ and the source term is automatically symbolically computed in MATLAB . The solution is plotted in fig. 13.

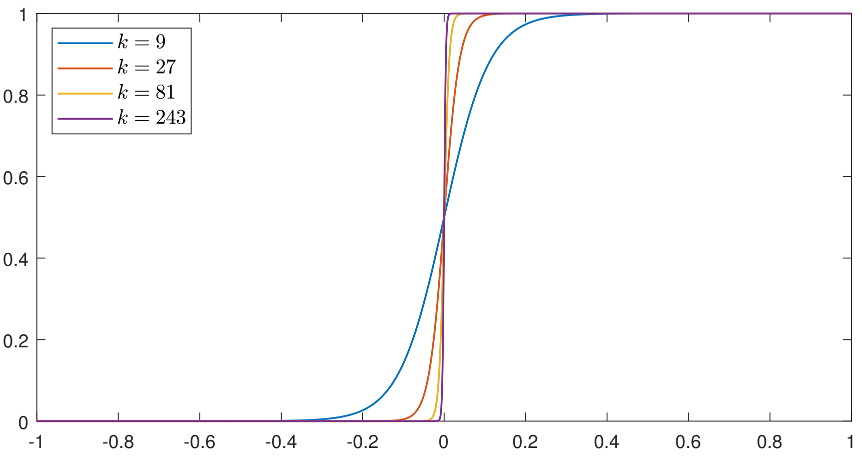


Figure 13: Exact solution (31) for the problem with step function solution varying parameter $k \in \mathbb{N}$.

Numerical solution obtained with the hierarchical adaptive procedure are plotted in fig. 14. The evolution of this approximation throughout the successive refinements is represented in fig. 15. Convergence plot of the same scenario in shown in fig. 16a.

Comparing convergence plots in fig. 16, we see that the stopping criteria are working well in both instances, but unfortunately the procedure turns out again to be sensitive on the choice of the initial approximation space in unpredictable ways, sicne starting with a fine space in this case stops convergence earlier and gives an overall worse result (even if marginally).

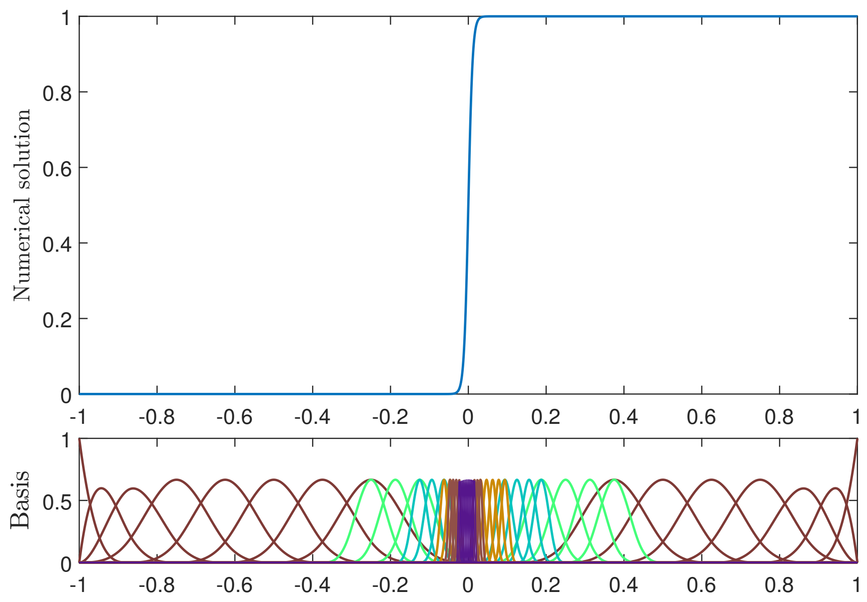


Figure 14: Adaptive numerical solution for the step function problem with $k = 91$. Parameters of the adaptive solver are in table 5. The solution was found after 6 iterations with 52 degrees of freedom and the procedure required 3.35 seconds. L^2-error is $5.954\,482\,17 \times 10^{-6}$. Computation of the uniform solution with the same number of DoF required 0.39 seconds and has an L^2-error of $0.019\,237\,17$.

Quadratic B-splines have been tested, but they do not seem to give good results. An explanation might be the symmetry of their shape with respect to the knots distribution. This hypothesis is (partially) confirmed by a comparison with fourth and fifth degree B-splines: the former tend to behave like quadratic, while the latter like cubic. Anyway, more tests are necessary in this regard.

5 Conclusions

In this work have discussed hierarchical B-splines in the framework of the adaptive numerical solution of advection-diffusion problems in one dimension. We first reviewed the definition and basic properties of HB-splines as a locally refined B-spline basis. Hierarchical refinement turns out to be very flexible, because it is suitable for any degree of smoothness in combination with B-splines. We then developed adaptive methods for advection-diffusion problems based on HB-splines, borrowing and adapting some techniques and results from adaptive finite element methods. Algorithms for the main parts of the procedure have been developed. Numerical results obtained on some test problems show that the adaptive procedure we presented is able to resolve localised singularities in the solutions with a modest increase in the computational cost. Moreover adaptive HB-spline approximation can also be a valuable tool for more regular solutions, contrarily to adaptive FEM, which is almost exclusively competitive for very strong local singularities.

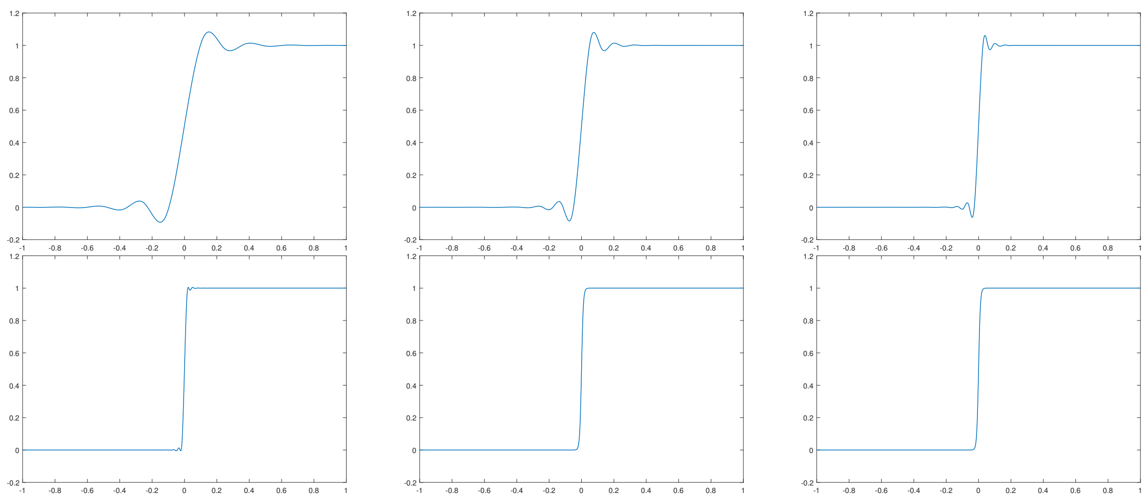
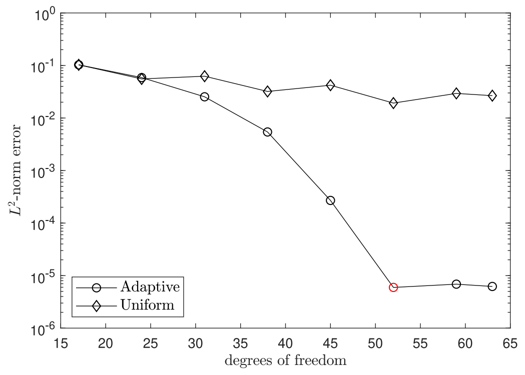
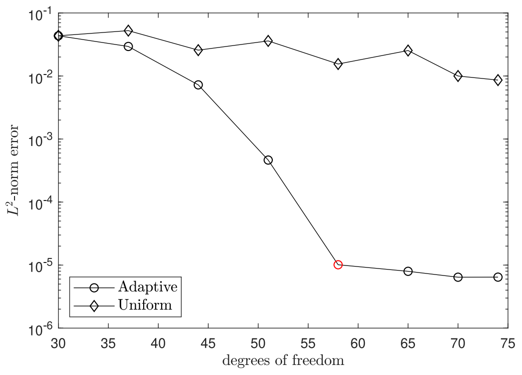


Figure 15: Sequence of refinements of the numerical solution in fig. 14.



(a) Initial space with 17 DoF.



(b) Initial space with 30 DoF.

Figure 16: Convergence plots for the step function problem with $k = 91$.

The regularity of B-splines made for much faster convergence with respect to FEM, with almost one tenth of the degrees of freedom needed on average, but, on the other hand, convergence of the HB-spline approximation is much more unpredictable and in some instances very sensitive on the parameters of the solver, and mainly to the choice of the initial approximation space. It is unclear to us weather this sensitivity is inherent to the method or induced by our algorithms or coding. The main shortcoming of our approach is the inability of our algorithms to deal with refinement on a previously unrefined region. This locks the refinement in only one direction and seriously limits the possibility of improving the approximation; more work need to be done in this direction.

A Algorithms

Algorithm 1: Efficient assembly of the stiffness matrix (22).

```
Input: Data of the problem, hspace with data of the HB-spline space $\mathcal{S}^{\mathcal{H}}$
Output: Stiffness matrix $A$ for the discretization of the problem in $\mathcal{S}^{\mathcal{H}}$
procedure assembly(probdata, hspace)
    L = nlev                                                    // Deepest level of the space
    $[A]_{j,k} \leftarrow 0,\ \forall j,k = 1,\dots,\dim\left(\mathcal{S}^{\mathcal{H}}\right) - 2$          // Initialize $A$ with zeroes
    $j \leftarrow 1$                                                       // Row index
    for $lr = 1,\dots,L$ do                                           // Loop on all levels
        $\mathcal{A}_{\mathtt{lr}} \setminus \left\{\beta_1^{\mathtt{lr}}, \beta_{N_{\dim(\mathcal{S}_{\mathtt{lr}})}}^{\mathtt{lr}}\right\}$          // Exclude first and last B-splines from active
        forall $jlr \in \mathcal{A}_{lr}$ do
            $k \leftarrow 1$                                                // Column index
            for $lc = 1,\dots,L$ do
                $\mathcal{A}_{\mathtt{lc}} \setminus \left\{\beta_1^{\mathtt{lc}}, \beta_{N_{\dim(\mathcal{S}_{\mathtt{lc}})}}^{\mathtt{lc}}\right\}$
                if $\#\mathcal{A}_{lc} > 0$ then        // Consider only levels with active B-splines
                    // We want to skip all levels which contain functions with no
                       support in common with the selected B-spline
                    if $\mathrm{supp}\left(\beta_{jlr}^{lr}\right) \cap \bigcup_{\beta \in \mathcal{A}_{lc}} \mathrm{supp}\,(\beta) \neq \emptyset$ then
                        forall $klc \in \mathcal{A}_{lc}$ do          // Column index in current level lc
                            if $\mathrm{supp}\,(\beta_{klc}^{lc}) \cap \mathrm{supp}\left(\beta_{jlr}^{lr}\right) \neq \emptyset$ then
                                $[A]_{j,k} \leftarrow a\left(\beta_{klc}^{\mathtt{lc}}, \beta_{jlr}^{\mathtt{lr}}\right)$
                            $k \leftarrow k + 1$
                    else
                        $k \leftarrow k + \#\mathcal{A}_{\mathtt{lr}}$               // Account for all skipped functions
            $j \leftarrow j + 1$
    return $A$                                                  // Stiffness matrix
```

B Matlab codes

The main original content of this work has been the development of algorithms for hierarchical B-spline adaptive discretization technique and the implementation in Matlab . The complete library `adHBsplineFEM` (for advection-diffusion HB-spline FEM), is available at the following GitHub repository: `https://github.com/arielsboiardi/adAHBsplineFEM`.

Algorithm 2: Efficient evaluation of the load vector.

```
Input: Data of the problem, hspace with data of the HB-spline space $\mathcal{S}^{\mathcal{H}}$
Output: Contribution of the boundary conditions to the load vector $F$
procedure load_eval(probdata, hspace)
    L = nlev                                                  // Total number of levels
    $[F]_r = 0,\ \forall r = 1, \ldots, \dim\left(\mathcal{S}^{\mathcal{H}}\right) - 2$          // Initialize $F$ with zeroes
    if $u_0 \neq 0 \vee u_L \neq 0$ then           // Only if boundary conditions are not trivial
        BDfun ← "left"                                  // First loop is the left boundary
        $r \leftarrow 1$
        d ← 1
        first ← 1, last ← L
        foreach $u_{BD} \in \{u_0, u_L\}$ do               // Repeat on both boundary conditions
            if $u_{BD} \neq 0$ then
                $\beta_{i_{BD}}^{l_{BD}}$ = get_bd_fun(BDfun, $\mathcal{S}^{\mathcal{H}}$)               // Basis function for $u_{BD}$
                for l = first : d : last do
                    $\mathcal{A}_l \setminus \left\{\beta_1^l, \beta_{\dim(\mathcal{S}_l)}^l\right\}$
                    if $\#\mathcal{A}_l > 0$ then              // Only of levels with active functions
                        if $\operatorname{supp}\left(\beta_{i_{BD}}^{l_{BD}}\right) \cap \bigcup_{\beta \in \mathcal{A}_l} \operatorname{supp}(\beta) \neq \emptyset$ then
                            if d=-1 then                      // Second loop right boundary
                                Reverse order of $\mathcal{A}_l$
                            forall $a \in \mathcal{A}_l$ do             // Loop on all active functions
                                if $\operatorname{supp}(\beta_a^l) \cap \operatorname{supp}\left(\beta_{i_{BD}}^{l_{BD}}\right) \neq \emptyset$ then
                                    $[F]_r \leftarrow [F]_r - u_{BD} \cdot a\left(\beta_{BD}^{l_{BD}}, \beta_a^l\right)$
                                    $r \leftarrow r + d$
                                else
                                    $r \leftarrow r + d \cdot \#\{i \geq a \mid \beta_i^l \in \mathcal{A}_l\}$
                                    break
                        else
                            $r \leftarrow r + d \cdot \#\mathcal{A}_l$                // Account for skipped functions
            BDfun ← "right"        // After the first loop, the right bc is selected
            first ← L, last ← 1                    // Levels are now read in reverse order
            d ← −1
            $r \leftarrow \dim\left(\mathcal{S}^{\mathcal{H}}\right) - 2$                        // $F$ is filled from the end
    if $f \neq 0$ then                          // Contribution source term in the equation
        $r \leftarrow 1$
        for l= 1,...,L do
            forall $a \in \mathcal{A}_l \setminus \left\{\beta_1^l, \beta_{\dim(\mathcal{S}_l)}^l\right\}$ do
                $\operatorname{supp}(\beta_a^l) = [\xi_a, \xi_{a+p+1}]$
                $[F]_r \leftarrow [F]_r + \int_{\xi_a}^{\xi_{a+p+1}} f \beta_a^l$
                $r \leftarrow r + 1$
    return $F$                                                       // Load vector
```

Algorithm 3: Find boundary functions in the hierarchical space.

```
Input: Boundary function BDfun to search, hierarchical B-spline space $\mathcal{S}^{\mathcal{H}}$ where
       function is searched
Output: Level of the space $\mathtt{L}_{BD}$ where the boundary function is found, index $i$ of the
        function in the space $\mathcal{S}_{\mathtt{L}_{BD}}$
procedure get_bd_fun()
    L = nlev
    $i \leftarrow 1$                // It is assumed that BDfun is the left-boundary function
    for $l = 1, \dots, L$ do
        if BDfun is "last" then
            $i = \dim(\mathcal{S}_1)$          // Index of right-boundary function of level l
        if $\beta_i^l \in \mathcal{A}_l$ then
            return $L, i$
```

Algorithm 4: Partial evaluation of local error estimate.

```
Input: Approximated solution $u_h$, data of the problem, approximation space $\mathcal{S}^{\mathcal{H}}$, last
       computed local error estimate heta
Output: Local error estimates heta for $u_h \in \mathcal{S}^{\mathcal{H}}$.
procedure hLocRes($u_h, \mathcal{S}^{\mathcal{H}}$, heta)
    L = nlev                                   // Current deepest level of hspace
    $\mathcal{S}_{\mathtt{L}}$,                                    // B-spline space of deepest level
    $\Xi = \begin{bmatrix} \xi_1 & \cdots & \xi_{\dim(\mathcal{S}_{\mathtt{L}})+p+1} \end{bmatrix}$                    // Knot vector of $\mathcal{S}_{\mathtt{L}}$
    $\mathcal{Q}_{\mathtt{L}} = \begin{bmatrix} Q_1 & \cdots & Q_N \end{bmatrix}$                       // Cells of mesh of level L
    $\eta = 0$                                          // Initilize
    // Remember that $u_h$ here is the solution (16), therefore $u_h \in \mathcal{S}_0^{\mathcal{H}}$.
    for $j = 1, \dots, N$ do
        if $Q_j \in \mathcal{Q} \cap \mathcal{Q}_L$ & $Q_j \neq \emptyset$ then    // Only consider active non empty cells
            $[\eta]_j = \mathrm{meas}(Q_j) \, \| f + \mu (B'' + u_h'') - b (B' + u_h') \|_{L^2(Q_j)}$
    $\mathtt{heta}_{\mathtt{L}} = \eta$
    // In order not to cout two times the same error, we set to 0 the error
       estimates of the previous level that sit on deactivated cells.
    for $Q_j \in \mathcal{Q}_{L\text{-}1}$ do
        if $Q_j \notin \mathcal{Q}$ then
            $[\mathtt{heta}_{\mathtt{L-1}}]_j = 0$
```

Algorithm 5: Refinement procedure for a hierarchical B-spline space.

```
Input: HB-spline space hspace, marked functions of hspace to be refine.
Output: Refinement of hspace
procedure refine(hspace, marked_fun)
    L = nlev                                                    // Deepest level of hspace
    marked_fun ← marked_fun ∪ get_supp_fun($\mathcal{S}_L$, marked_fun)
    $\mathcal{D}_{L}$ ← $\mathcal{D}_{L}$ ∪ marked_fun
    $\mathcal{A}_{L}$ ← $\mathcal{A}_{L}$ \ marked_fun
    hcells_L ← hcells_L \ get_cells(marked_fun)
    hknots_L ← knots_from_cells(hcells_L)                     // Get knots delimiting cells
    nlev ← L+1                                                  // Add new level
    $\mathcal{S}_{L+1}$ ← dyad_ref($\mathcal{S}_L$)
    $\mathcal{A}_{L+1}$ ← get_children(marked_fun)
    $\mathcal{D}_{L+1}$ ← $\mathcal{A}^C_{L+1}$
    hcells_{L+1} ← get_cells($\mathcal{A}_{L+1}$)
    hknots_{L+1} ← knots_from_cells(hcells_{L+1})
    $\dim$(hspace) = $\dim$(hspace) − #marked_fun + #$\mathcal{A}_{L+1}$      // New dim.
    return hspace                                               // Refined hierarchical space
```

Algorithm 6: Find B-splines in $\mathcal{S}$ which have common support with selected functions of $\mathcal{S}$.

```
Input: B-spline space $\mathcal{S}$, indices fun_indof B-splines in $\mathcal{S}$
Output: Indices of the B-splines in $\mathcal{S}$ with common support with those of indices
        fun_ind
procedure get_supp_fun($\mathcal{S}$, fun_ind)
    $\Xi = \begin{bmatrix}\xi_1 & \cdots & \xi_{N+p+1}\end{bmatrix}$                          // Knot vector of $\mathcal{S}$
    forall $i \in$ fun_ind do
        $\text{supp}(\beta_i) = [\xi_i, \xi_{i+p+1}]$                       // Support of the $i$-th B-spline
        selected ← $\{j \le i \mid \xi_j = \xi_i\} \cup \{j > i \mid \xi_{j+p+1} = \xi_{i+p+1}\}$
    selected = $\{i_{k_1} < \ldots < i_{k_{n+1}}\}$                     // Indices of selected functions
    // Now if two selected B-splines have adjacent supports, all B-splines between
       them have support in common, and are therefore to be selected
    for $j = 1, \ldots, n$ do
        dist ← $i_{k_{j+1}} - i_{k_j}$
        if $1 <$ dist $\le p + 1$ then
            selected ← selected ∪ $\{i \mid i_{k_j} < i < i_{k_{j+1}}\}$
    return selected
```

Algorithm 7: Get cells in the support of given B-splines.

```
Input: Indices fun_ind of B-splines in $\mathcal{S}$
Output: Indices cell_ind of the cells that support the given functions
procedure get_cells(fun_ind)
    forall $i \in$ fun_ind do
        cell_ind = cell_ind $\cup \{j \mid i \leq j \leq i+p\}$        // $\mathrm{supp}(\beta_i) = [\xi_i, \xi_{i+p+1}]$
        // To get knots in the support of $\beta_i$ it is sufficent to let $j$ go up to
            $i+p+1$
    return cell_ind        // Cells in the support of B-splines with indice fun_ind
```

Algorithm 8: Dyadic refinement of B-spline space $\mathcal{S}$.

```
Input: B-spline space $\mathcal{S}$
Output: Refinement of $\mathcal{S}$ by dyadic subdivision of the knot vector
procedure dyad_ref($\mathcal{S}$)
    $\Xi = \begin{bmatrix}\xi_1 & \cdots & \xi_{M+2p+2}\end{bmatrix}$        // Knot vector of $\mathcal{S}$
    for $i = 1, \ldots, M+2p+1$ do        // Total number of knots -1
        $\theta_k \leftarrow \xi_i$
        $k \leftarrow k+1$
        $h \leftarrow \xi_{i+1} - \xi_i$
        if $h > 0$ then
            $\theta_k \leftarrow \xi_i + \frac{h}{2}$
            $k \leftarrow k+1$
    $\theta_k = \xi_{M+2p+2}$
    $\Theta = \begin{bmatrix}\theta_1 & \cdots & \theta_k\end{bmatrix}$        // Refined knot vector
    return $\mathcal{S}_\Theta$        // B-spline space built on $\Theta$
```

Algorithm 9: Determines the children of given B-splines in the refined space, according to the subdivision formula (8).

```
Input: Indices fun_indof B-splines in $\mathcal{S}$
Output: Indices C of B-splines in the refinement of $\mathcal{S}$ that are children of the given
        B-splines
procedure get_children(fun_ind)
    C $\leftarrow \emptyset$        // Initialize
    forall $i \in$ fun_ind do
        C$_i \leftarrow \{j > 0 \mid 2i - p - 1 \leq j \leq 2i\}$
        C $\leftarrow$ C $\cup$ C$_i$
    return C        // Indices of children B-splines of fun_ind
```

References

[1] A. S. Boiardi. “FEM adattivo per problemi di trasporto e diffusione”. Bachelor’s Thesis. Università degli Studi di Parma, Apr. 2019. URL: https://github.com/arielsboiardi/adAFEM/blob/master/Report/FEM_adattivo_problemi_diffusione_trasporto.pdf.

[2] S. C. Brenner and R. Scott. *The mathematical theory of finite element methods.* Vol. 15. Springer Science & Business Media, 2007. DOI: 10.1007/978-0-387-75934-0.

[3] M. G. Cox. “The numerical evaluation of B-splines”. In: *Journal of the Institute of Mathematics and its Applications* 10.2 (Oct. 1972), pp. 134–149. DOI: 10.1093/imamat/10.2.134.

[4] C. De Boor. “On calculating with B-splines”. In: *Journal of Approximation Theory* 6 (1972). Collection of articles dedicated to J. L. Walsh on his 75th birthday, V (Proc. Internat. Conf. Approximation Theory, Related Topics and their Applications, Univ. Maryland, College Park, Md., 1970), pp. 50–62. DOI: 10.1016/0021-9045(72)90080-9.

[5] W. Dörfler. “A Convergent Adaptive Algorithm for Poisson’s Equation”. In: *SIAM Journal on Numerical Analysis* 33.3 (1996), pp. 1106–1124. DOI: 10.1137/0733054.

[6] D. R. Forsey and R. H. Bartels. “Hierarchical B-spline Refinement”. In: *ACM SIGGRAPH Computer Graphics* 22.4 (June 1988), pp. 205–212. DOI: 10.1145/378456.378512.

[7] E. M. Garau and R. Vázquez. “Algorithms for the implementation of adaptive isogeometric methods using hierarchical B-splines”. In: *Applied Numerical Mathematics. An IMACS Journal* 123 (2018), pp. 58–87. DOI: 10.1016/j.apnum.2017.08.006.

[8] E. Hewitt and R. E. Hewitt. “The Gibbs-Wilbraham phenomenon: An episode in fourier analysis”. In: *Archive for History of Exact Sciences* 21.2 (June 1979), pp. 129–160. DOI: 10.1007/BF00330404. URL: https://doi.org/10.1007/BF00330404.

[9] K. Höllig. *Finite element methods with B-splines.* Vol. 26. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2003. DOI: 10.1137/1.9780898717532.

[10] T. J. R. Hughes, J. A. Cottrell, and Y. Bazilevs. “Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement”. In: *Computer Methods in Applied Mechanics and Engineering* 194.39 (2005), pp. 4135–4195. DOI: 10.1016/j.cma.2004.10.008.

[11] R. Kraft. “Adaptive and linearly independent multilevel B–splines”. In: *Surface fitting and multiresolution methods.* Ed. by A. Le Méhauté, C. Rabut, and L. L. Schumaker. Vanderbilt University Press, Nashville, TN, 1997, pp. 209, 218.

[12] K. Mekchay and R. H. Nochetto. “Convergence of adaptive finite element methods for general second order linear elliptic PDEs”. In: *SIAM Journal on Numerical Analysis* 43.5 (2005), pp. 1803–1827. DOI: 10.1137/04060929X.

[13] L. Piegl and W. Tiller. *The NURBS book*. Monographs in Visual Communication. Springer-Verlag Berlin Heidelberg, 1997. DOI: 10.1007/978-3-642-59223-2.

[14] D. Schillinger et al. "An isogeometric design-through-analysis methodology based on adaptive hierarchical refinement of NURBS, immersed boundary methods, and T-spline CAD surfaces". In: *Computer Methods in Applied Mechanics and Engineering* 249-252 (Dec. 2012). Higher Order Finite Element and Isogeometric Methods, pp. 116–150. DOI: 10.1016/j.cma.2012.03.017.

[15] L. Schumaker. *Spline Functions: Basic Theory*. 3rd ed. Cambridge Mathematical Library. Cambridge University Press, 2007. DOI: 10.1017/CBO9780511618994.

[16] R. Verfürth. *A Posteriori Error Estimation Techniques for Finite Element Methods*. Numerical mathematics and scientific computation. Oxford University Press, 2013. DOI: 10.1093/acprof:oso/9780199679423.001.0001.

[17] A.-V. Vuong et al. "A hierarchical approach to adaptive local refinement in isogeometric analysis". In: *Computer Methods in Applied Mechanics and Engineering* 200.49 (2011), pp. 3554–3567. DOI: 10.1016/j.cma.2011.09.004.