A Subdivision-based Implementation of Non-uniform Local Refinement with THB-splines

Xinhui Ma[1], Robert Cripps[2], Qingde Li[1], Ping Lin[3]

[1]Department of Computer Science, University of Hull, HU6 7RX, UK

[2]School of Mechanical Engineering, University of Birmingham , B15 2TT, UK

[3]Division of Mathematics, University of Dundee, DD1 4NH, UK

Abstract

Local refinement of spline basis functions is an important process for spline approximation and local feature modelling in computer aided design (CAD). This paper develops an efficient local refinement method for non-uniform and general degree THB-splines(Truncated hierarchical B-splines). A non-uniform subdivision algorithm is improved to efficiently subdivide a single non-uniform B-spline basis function. The subdivision scheme is then applied to locally hierarchically refine non-uniform B-spline basis functions. The refined basis functions are non-uniform and satisfy the properties of linear independence, partition of unity and are locally supported. The refined basis functions are suitable for spline approximation and numerical analysis. The implementation makes it possible for hierarchical approximation to use the same non-uniform B-spline basis functions as existing modelling tools have used. The improved subdivision algorithm is faster than classic knot insertion. The non-uniform THB-spline approximation is shown to be more accurate than uniform low degree hierarchical local refinement when applied to two classical approximation problems.

1 Introduction

Efficient local unstructured refinement is an extremely important research focus in Isogeometric Analysis (IGA), since geometric models created by designers will most likely be too coarse for analysis [10, 17]. IGA aims to seamlessly integrate CAD and FEA (finite element analysis), using the same spline basis functions for the representation of geometry in the design

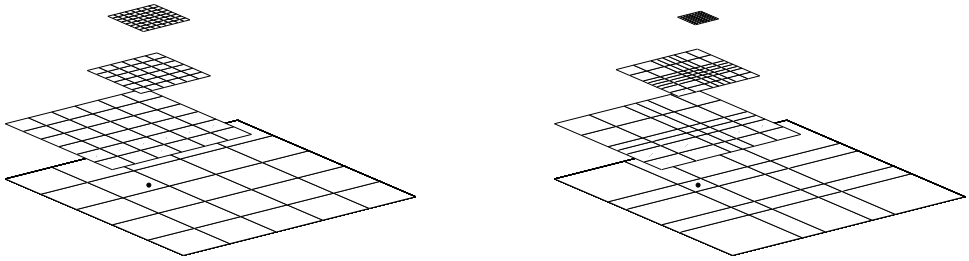
stage and the approximation in the analysis stage. Spline approximation also has many other applications, such as approximation of curve segments [5, 6], reduction of dense data points [12] and fitting dense meshes [13]. Furthermore, adaptive local mesh refinement is a most effective approach for FEA as shown in [1]. Local refinement can increase the accuracy of approximation whilst using fewer basis functions.

The THB-spline basis [8] is a normalized basis for hierarchical spline spaces. Thanks to the truncation mechanism, it forms a convex partition of unity. It has smaller support than hierarchical B-splines, hence enables a local mesh refinement in an effective and easy way. Furthermore, the THB-spline basis is strongly stable for adaptively refined multilevel spline spaces [9]. Consequently, THB-splines seem to be a promising adaptive approximation tool. However, the calculation of the weights of non-uniform truncated refined basis functions is not clear. This paper will focus on efficient implementation of the THB-spline in non-uniform settings.

Non-uniform general degree hierarchical refinement is essential in NURBS-based IGA which requires the same basis functions in both the design model and the approximation of the solution of the partial differential equation. Since the design stage adopts non-uniform B-splines, the approximation stage should adopt the same format of non-uniform B-splines. Non-uniform hierarchical refinement gives finer refinement and more localization than uniform hierarchical refinement, as shown in Fig. 1. Furthermore, general degree B-splines gives more accurate approximation than low degree. Hence, non-uniform general degree hierarchical refinement is very important for improving the adaptive IGA. However, application of non-uniform B-splines to hierarchical refinement is more of a challenging task due to the complexity of non-uniform knot insertion.

B-spline subdivision schemes provide a feasible solution to the problem of non-uniform general degree hierarchical refinement. Knot insertion is the basic operation in both subdivision surface generation and hierarchical refinement of basis functions. The Boehm algorithm [2] is for single knot insertion. The Oslo algorithm [4] extends it to parallel knot insertion. Recently, efficient parallel knots insertion algorithms have been developed [3] with much reduced number of iterations, which greatly improves the efficiency and speed of the subdivision. The generated subdivision surface is compatible with NURBS. This suggests that efficient hierarchical refinement of NURBS basis functions is feasible.

The aim of the present paper is to develop an efficient subdivision-based implementation of THB-splines in general degree and non-uniform settings. Firstly, a non-uniform subdivision algorithm is developed to determine the weights of the refined basis functions derived from a single original basis function. The resulting subdivision weights are then used



(a)Uniform hierarchical refinement (b)Non-uniform hierarchical refinement

FIGURE 1. Non-uniform hierarchical refinement gives finer refinement and smaller localization than uniform hierarchical refinement at the marked black point of the knot meshes

with hierarchical refinement to generate a refined basis with properties of linear independence, partition of unity and local support. Finally, the refined basis functions are used to improve the accuracy of approximation. The refined basis functions provide a richer set of basis functions that can be used for local feature modelling and adaptive IGA approximation. The main contribution of the paper is improving the non-uniform subdivision algorithm to be suitable for approximation, and implementing efficiently local refinement of THB-splines by applying the algorithm.

The remainder of this paper is organized as follows. Section 2 reviews related work in THB-splines and subdivision of B-spline curves and surfaces. Section 3 improves a non-uniform subdivision algorithm for approximation. Implementation of univariate non-uniform THB-splines of general degree is given in section 4. Section 5 extends the implementation to the bi-variate case. The performance of the implementation of non-uniform THB-splines is demonstrated by adaptive approximations in Section 6. Section 7 concludes with a few remarks and some directions for future research.

2 Related work

A B-spline basis of degree p is formed from a sequence of knots, $\tau_i \in \mathbb{R}$, where $\tau_0 \leq \tau_1 \leq, ..., \leq \tau_{n+p+2}$. Let $\tau = \{\tau_0, \tau_1, ..., \tau_{n+p+2}\}$, a univariate B-spline basis function $N_{i,p}(\tau)$ is defined using a recurrence relation.

2.1 Truncated hierarchical B-spline basis

Hierarchical refinement constructs a basis of a hierarchical spline space through local basis refinement. Suitable hierarchical B-splines are proposed in [7, 18], which are locally supported, linearly independent and non-negative. The approach is extended by [8] so that the resulting basis forms a partition of unity along with smaller support, and are called truncated hierarchical B-splines.

Let

$$V^0 \subset V^1 \subset \cdots \subset V^{n-1}$$

be a nested sequence of univariate B-splines function spaces defined on the domain Γ^0. Each spline space $V^l, l = 0, \ldots, n-1$, is spanned by a given normalized B-spline basis $\mathcal{N}^l$.

In addition, let

$$\Gamma^0 \supseteq \Gamma^1 \supseteq \cdots \supseteq \Gamma^{n-1}$$

be a sequence of nested domains.

Let $h \in V^l$ and let

$$h = \sum_{N \in \mathcal{N}^{l+1}} W_N^{l+1}(h) N.$$

be its representation with respect to the finer basis of V^{l+1}, where W_N^{l+1} is the weight derived from Eq.(1). The truncation of h with respect to $\mathcal{N}^{l+1}$ and Γ^{l+1} is defined as

$$\begin{aligned} \text{trunc}^{l+1} h &= h - \sum_{N \in \mathcal{N}^{l+1}, supp N \subseteq \Gamma^{l+1}} W_N^{l+1}(h) N \\ &= \sum_{N \in \mathcal{N}^{l+1}, supp N \nsubseteq \Gamma^{l+1}} W_N^{l+1}(h) N. \end{aligned}$$

Although THB-splines gives the general definition of a hierarchical basis with small local support, the calculation of the weights of non-uniform refined basis functions is not clear. The Boehm knot insertion algorithm [2] or the Oslo algorithm [4] could be used to calculate the weights of refined basis functions. However, there is an issue with efficiency due to producing refined basis functions one by one. Non-uniform subdivision algorithms provide feasible solutions to this problem. The efficiency issue is addressed by [3] to insert any number of knots at a time using *refine* and *smooth* steps to produce refined basis function simultaneously, as explained below.

2.2 Non-uniform subdivision of B-spline curves and surfaces

Refinement by subdivision is a remarkable property of B-splines. For a univariate B-spline basis function $N_{0,p}$ of polynomial degree p, the subdivision property leads to the relation [16]:

$$N_{0,p}(\tau) = \sum_{j=0}^{p+1} W_j \bar{N}_{j,p}(\tau). \tag{2.1}$$

where $\bar{N}_{j,p}$ are the subdivided basis functions and $W_j \in \mathbb{R}$ are the weights of the subdivided basis functions. (The W_j are used to construct the original basis function, not the weights of basis functions to construct a B-spline curve).

For the uniform case, the weights W_j can be expressed in terms of binomial coefficients as [14]:

$$W_j = 2^{-p} \frac{(p+1)!}{j!(p+1-j)!} \tag{2.2}$$

Unfortunately for non-uniform B-splines, Eq. (2.2) does not hold. Nevertheless the subdivision weights for the non-uniform case can be constructed using certain subdivision algorithms [3] based on blossoming. The blossom [15] of a degree p polynomial $P(\tau)$ is the unique polynomial in p variables $b(\tau_1, \tau_2, \ldots, \tau_p)$ satisfying the following three properties:

- Symmetry: $b(\tau_{\sigma(1)}, \tau_{\sigma(2)}, \ldots, \tau_{\sigma(p)}) = b(\tau_1, \tau_2, \ldots, \tau_p)$ for any permutation σ of $\{1, 2, \ldots, p\}$
- Multi-affine: $b((1-\alpha)\tau_{11}+\alpha\tau_{12}, \tau_2, \ldots, \tau_p) = (1-\alpha)b(\tau_{11}, \tau_2, \ldots, \tau_p)+ \alpha b(\tau_{12}, \tau_2, \ldots, \tau_p)$
- Diagonal: $b(\tau, \tau, \ldots, \tau) = P(\tau)$

These properties are compatible with B-spline basis functions $N(\tau)$. The relationship between B-spline basis functions can be represented by their corresponding blossoms. The properties of blossom can be used for subdivision of B-spline basis functions. According to the dual function property, they are also suitable for control points construction.

Based on blossoming [15] and using the *refine* and *smooth* algorithm [11], a non-uniform subdivision algorithm is introduced for general degree B-spline curves [3]. This algorithm works by inserting new knots between each pair of original knots. A case distinction is required for odd and even degrees. For odd degree, the refine stage adds a new point between each

pair of the original control points. Each smoothing stage copies half of the points themselves. Each of the other half points is an affine combination of three consecutive points. For even degree, the refine stage just doubles the original control points. Each smoothing stage places a pair of new points somewhere on the edge between a pair of old points. As a result, the algorithm efficiently doubles the control points of a non-uniform B-spline curve. The subdivision property can be used to improve spline approximation by construction of an optimal spline space.

3 Improved Non-uniform Subdivision Algorithm for Approximation

For non-uniform B-spline basis functions, a refinability formula equivalent to the uniform case, formula (2.2), does not exit. The aim here is to develop a subdivision algorithm to generate the weights for the refined basis functions and apply these weights for non-uniform hierarchical basis refinement. By simply replacing the coefficients in formula (2.1) with the generated non-uniform subdivision weights, the existing uniform hierarchical refinement algorithm can be used to achieve non-uniform refinement of the basis functions.

An efficient basis function refinement algorithm can be constructed from the subdivision algorithm in [3] which inserts control points into the original control meshes. This procedure can also be called knot insertion, since the addition of control points to a control mesh must be accompanied by knots inserted into neighboring basis functions. Operating on the knots of a single non-uniform B-spline basis function instead of the control points, and eliminating the refine step, a simplified refinement algorithm may be obtained. For odd degree, we find that the refine step can be combined with the smooth step. Setting the level variable λ to zero, the refine step is identical to the smooth step of level zero. For even degree, the refine step is replaced by setting two unit weights to the original basis function. Extra steps are added to check whether a subdivided basis function is an end basis function, since the weights associated with end subdivided basis functions are different from the weights associated with internal subdivided basis functions.

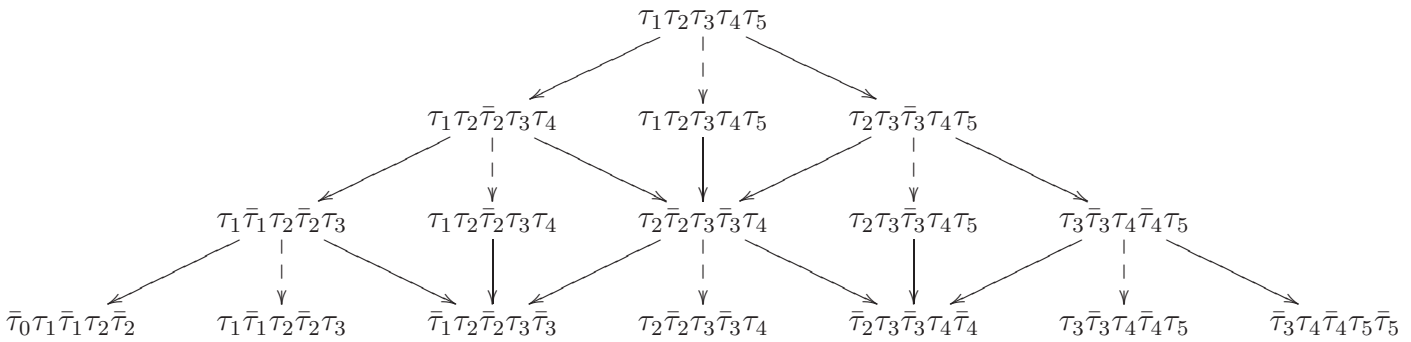
To illustrate the subdivision process, two examples of blossoming diagrams for the quintic and sextic algorithms are shown in Fig. 2. Each basis function is represented as the blossom function $b(\tau_1, \tau_2, \ldots, \tau_p)$. Note that the variables of the blossom function are the internal knots of its corresponding B-spline basis function. For example, the blossom variables of the knot vector $\{\tau_0, \tau_1, \ldots, \tau_p, \tau_{p+1}\}$ are $\{\tau_1, \tau_2, \ldots, \tau_p\}$. Here,

$\tau_i,\ i = 0, 1, ..., p+1$ is an original knot and $\bar{\tau}_i,\ i = 0, 1, ..., p$ is a new knot to be inserted. An affine combination is represented by arrows entering a basis function. A dashed line means no actual calculation.

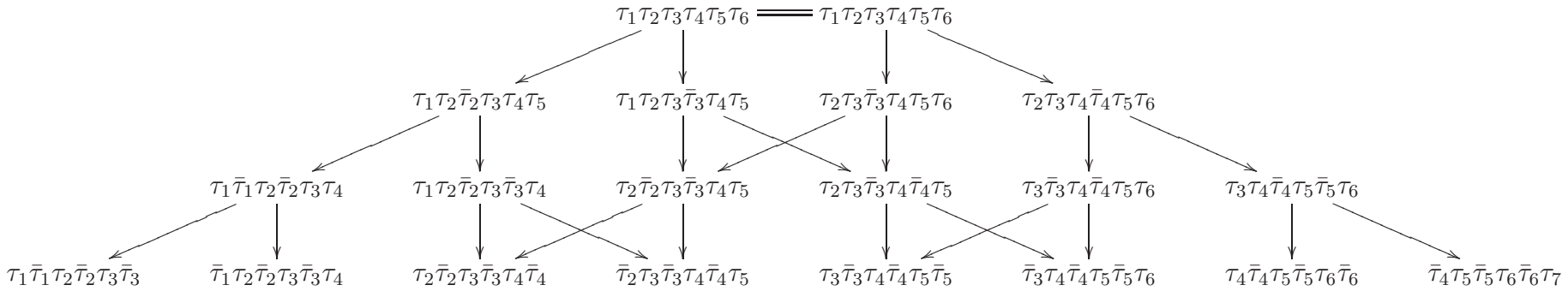
Algorithms 1 and 2 give generic pseudo-code implementations of the subdivision of a single non-uniform B-spline basis function for odd and even degree respectively. The original weight value is one, which is the weight of the original basis function for itself. The output are the weights of the refined basis functions which make up the original basis function, i.e. which can be used to reconstruct the original basis function.

Two examples of the subdivision algorithms are given in Fig. 3. The original basis functions are quintic and sextic non-uniform B-splines functions. Applying the subdivision algorithms to the knots of the original basis functions generates the weights of the refined basis functions. According to equation (2.1), the original basis functions are the sum of the weighted refined basis functions.

The proposed subdivision algorithm is more efficient than classic knot insertion algorithms, e.g. Oslo algorithm [4]. The Oslo algorithm can be treated as a local knot insertion algorithm, i.e. inserting into only one interval at a time. The proposed algorithms are global, i.e. acting on all knot intervals at once. The proposed algorithm is likely to overlap computations as much as possible. The generated subdivision weights can be used for hierarchical refinement of non-uniform B-spline basis functions as discussed in the following section.



(a) Quintic



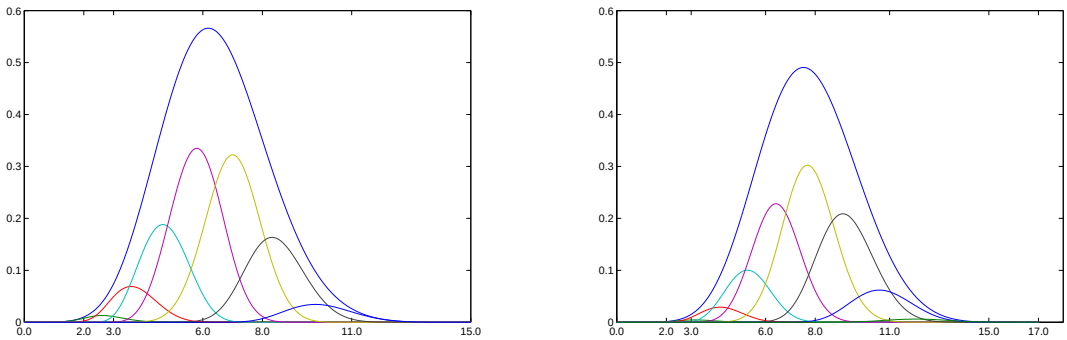
(b) Sextic

FIGURE 2. Blossoming diagrams for subdivision of a single non-uniform B-spline basis function (a) $p = 5$ and (b) $p = 6$

Algorithm 1: Subdivision algorithm for a non-uniform B-spline basis function of odd degree

input : Degree p, where $p \bmod 2 = 1$
input : Knot vector $\boldsymbol{\tau}$ of the original basis function, where $|\boldsymbol{\tau}| = p + 2$
input : Knots to be inserted $\bar{\boldsymbol{\tau}}$, where $\tau_i \le \bar{\tau}_i \le \tau_{i+1}, 0 \le i \le p+1$
output: Weights W_j, $0 \le j \le p+1$, of the refined basis functions

$\mathbf{t} \leftarrow \boldsymbol{\tau} \cup \bar{\boldsymbol{\tau}}$ // Such that each $\tau_i = t_{2i}$ and each $\bar{\tau}_i = t_{2i+1}$
$w^0_{(p+1)/2} \leftarrow 1.0$ // Assign initial weight
for $\lambda \leftarrow 0$ **to** $(p-1)/2$ **do** // For each smoothing step
  for $i \leftarrow p - \lambda$ **to** $p + \lambda + 2$ **do** // For each new basis
    $j \leftarrow i - (p+1)/2$ // Shifting of index
    if $\lambda \bmod 2 = i \bmod 2$ **then** // Copy the weight
      $w^{\lambda+1}_j = w^\lambda_j$
    else // Calculate weights by affine combination
      $c_0 = \frac{t_{i+p-\lambda} - t_{i+\lambda}}{t_{i+p-\lambda} - t_{i-p+\lambda}}$; $c_1 = \frac{t_{i+\lambda} - t_{i-\lambda}}{t_{i+p-\lambda} - t_{i-p+\lambda}}$; $c_2 = \frac{t_{i-\lambda} - t_{i-p+\lambda}}{t_{i+p-\lambda} - t_{i-p+\lambda}}$
      if $i = p - \lambda$ **then** // Left end basis
        $w^{\lambda+1}_j = c_2 w^\lambda_{j+1}$
      else if $i = p + \lambda + 2$ **then** // Right end basis
        $w^{\lambda+1}_j = c_0 w^\lambda_{j-1}$
      else // Internal basis
        $w^{\lambda+1}_j = c_0 w^\lambda_{j-1} + c_1 w^\lambda_j + c_2 w^\lambda_{j+1}$
for $j \leftarrow 0$ **to** $p+1$ **do** // Transfer weight notation
  $W_j = w^{(p+1)/2}_j$



(a) Quintic (b) Sextic

FIGURE 3. Subdivision of a single non-uniform spline basis function

Algorithm 2: Subdivision algorithm for a non-uniform B-spline basis function of even degree

input : Degree p, where $p \bmod 2 = 0$
input : Knot vector $\boldsymbol{\tau}$ of the original basis function, where $\mid \boldsymbol{\tau} \mid = p+2$
input : Knots to be inserted $\bar{\boldsymbol{\tau}}$, where $\tau_i \leq \bar{\tau}_i \leq \tau_{i+1}, 0 \leq i \leq p+1$
output: Weights W_j, $0 \leq j \leq p+1$, of the refined basis functions

$\mathbf{t} \leftarrow \boldsymbol{\tau} \cup \bar{\boldsymbol{\tau}}$ // Construct refined knot vector
$w_{p/2}^1 \leftarrow 1.0$; $w_{p/2+1}^1 \leftarrow 1.0$ // Assign initial weights
for $\lambda \leftarrow 1$ **to** $p/2$ **do** // For each smoothing step
  for $i \leftarrow p-\lambda$ **to** $p+\lambda+1$ **do** // For each new basis
    $j \leftarrow i - p/2$ // Shifting of index
    // Calculate weights by affine combination
    if $\lambda \bmod 2 = i \bmod 2$ **then**
      $c_0 = \frac{t_{i+p-\lambda+2} - t_{i-\lambda+1}}{t_{i+p-\lambda+2} - t_{i-p+\lambda}}$; $c_1 = \frac{t_{i-\lambda+1} - t_{i-p+\lambda}}{t_{i+p-\lambda+2} - t_{i-p+\lambda}}$
      if $i = p-\lambda$ **then** // Left end basis
        $w_j^{\lambda+1} = c_1 w_{j+1}^{\lambda}$
      else if $i = p+\lambda$ **then** // Second right end basis
        $w_j^{\lambda+1} = c_0 w_j^{\lambda}$
      else // Internal basis
        $w_j^{\lambda+1} = c_0 w_j^{\lambda} + c_1 w_{j+1}^{\lambda}$
    else
      $c_0 = \frac{t_{i+p-\lambda+1} - t_{i+\lambda}}{t_{i+p-\lambda+1} - t_{i-p+\lambda-1}}$; $c_1 = \frac{t_{i+\lambda} - t_{i-p+\lambda-1}}{t_{i+p-\lambda+1} - t_{i-p+\lambda-1}}$
      if $i = p-\lambda+1$ **then** // Second left end basis
        $w_j^{\lambda+1} = c_1 w_j^{\lambda}$
      else if $i = p+\lambda+1$ **then** // Right end basis
        $w_j^{\lambda+1} = c_0 w_{j-1}^{\lambda}$
      else // Internal basis
        $w_j^{\lambda+1} = c_0 w_{j-1}^{\lambda} + c_1 w_j^{\lambda}$
for $j \leftarrow 0$ **to** $p+1$ **do** // Transfer weight notation
  $W_j = w_j^{p/2+1}$

4 Implementation of Univariate Non-uniform THB-splines of General Degree

In this section, we derive a local refinement scheme to construct non-uniform THB-splines step by step, using the weights of the refined basis functions generated by the subdivision algorithm introduced in section 3.

4.1 Supporting small localization and maintaining the original partitioning

Given a knot interval marked to be refined, determining the refined domain and thus choosing the number of refined basis functions is an essential issue in hierarchical refinement. Reducing the local support and maintaining the original domain are two requirements that need to be considered.

A straightforward approach is to apply the subdivision equation (2.1) to all the original basis functions with support in the marked knot interval [18]. However, this approach is inefficient as it results in more refined basis functions than are necessary, which in turn leads to non-localization.

Alternatively, the contracted approach only adds three refined basis functions for a cubic B-spline basis function [16]. It is not clear how to use the contracted approach for general degrees. In the cubic case, this approach can achieve very small localization and satisfies the requirements of the approximation. A disadvantage of this approach is that it will add new knots to the original domain, which makes the refinement more complex in both geometry and approximation.

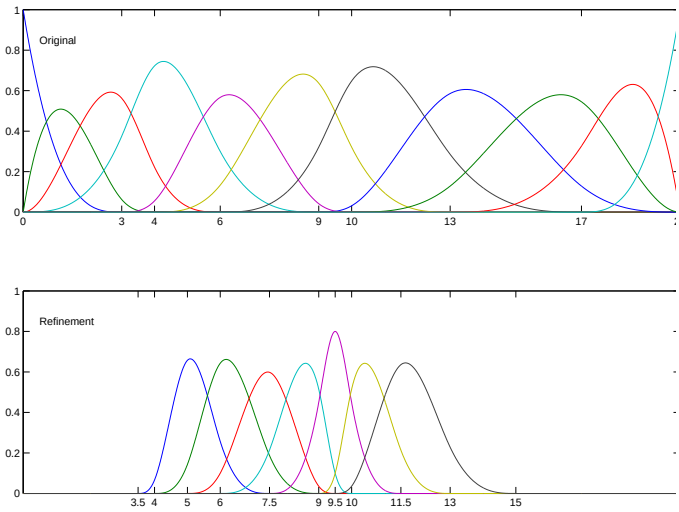
The approach here is to just add the refined basis functions whose support overlaps the marked knot interval. This approach can achieve smaller localization than straight subdivision and keep the partition of the original domain [16]. Fig. 4(a) illustrates the refinement of the marked knot span $[6, 10]$ for a cubic non-uniform B-spline. The support of each refined basis function overlaps with the marked knot span.

4.2 Recovering linear independence

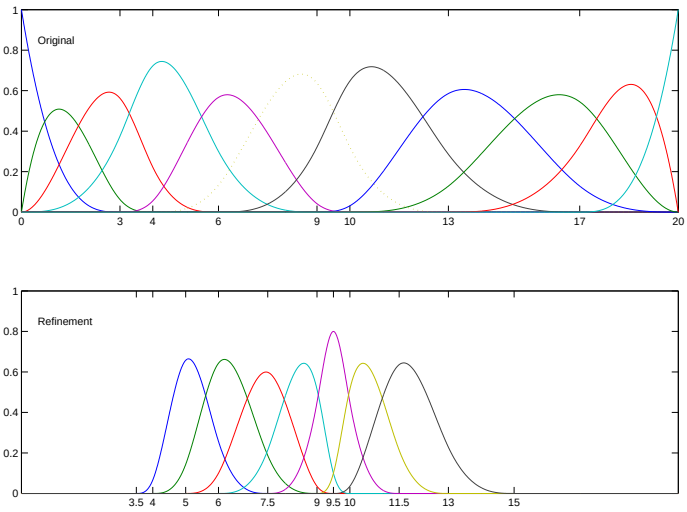
In order to construct a hierarchical B-spline space suitable for the approximation, the linear independence of the basis functions should be ensured. With reference to Fig. 4(b), in the refined level, there are more than $p+2$ consecutive refined B-spline basis functions. The linear combination of these refined basis functions will represent some of the basis functions in the previous level, as determined by equation (2.1). In order to determine the refined span of basis functions, it is necessary to determine all the B-spline basis functions (all the dotted functions in Fig. 4) to identify which basis functions can be removed from the coarser level.

4.3 Forming partition of unity

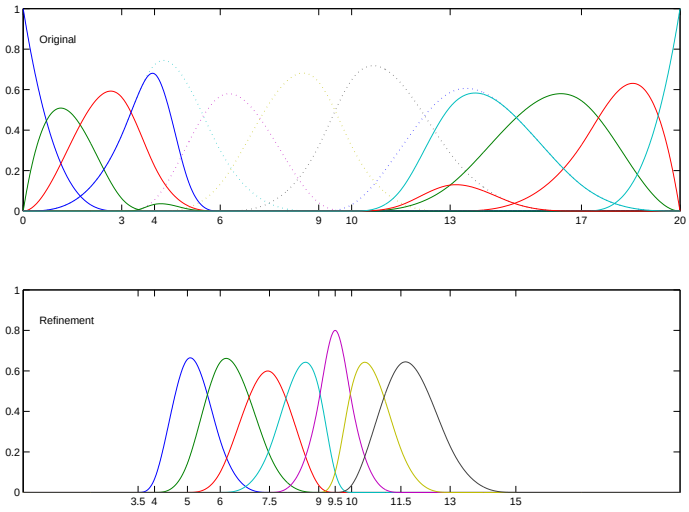
In the hierarchically refined basis of Fig. 4(b), some basis functions of the original level still implicitly contain some B-spline basis functions of the refined level. A further improvement of the conditioning and sparsity can be achieved by removing these multiplicities. If a refined B-spline basis



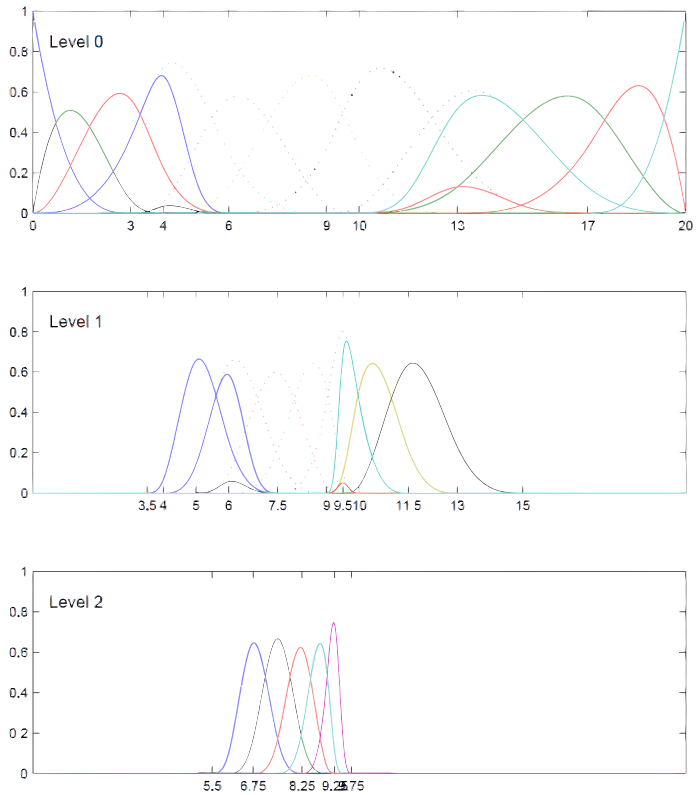
(a) Local refinement



(b) Recovering linear independence



(c) Forming unity partition



(d) Multi level refinement

FIGURE 4. Univariate hierarchical refinement

function has common support with a B-spline basis function on the current level and if it is a part of the linear combination of the subdivision B-spline basis functions generated from the current level then we subtract the B-spline basis function of the refined level, multiplied by the corresponding subdivision weight, from B-spline basis functions of the current level.

Applying the subtraction procedure to the basis in Fig. 4(b), we get the refined hierarchical basis in Fig. 4(c). The original basis functions are plotted as dotted lines, while the subtracted basis functions are plotted as solid lines. It illustrates that any overlap is further reduced whist maintaining the unity partition.

4.4 Multi-level hierarchical refinement

By repeating the procedure described in the previous paragraphs, the local knot support of the refinement can be reduced. An example of multi-level refinement procedure is illustrated in Fig. 4(d). To refine the position at 8, the node spans that are marked for refinement in levels 0 and 1 are $[6, 10]$ and $[7.5, 9]$ respectively.

The hierarchical local refinement is efficient, since only a small portion of coarse basis functions need calculating the associated subdivision weights. These basis functions' support partly overlaps the refined intervals. We do not need to do anything to the basis functions outside the refined interval. We just remove the basis functions within the refined intervals. Replacing the uniform subdivision weights by the weights generated from the proposed subdivision algorithms, we can realize the non-uniform hierarchical refinement using the existing uniform hierarchical refinement code.

5 Implementation of Bivariate non-uniform THB-splines

The extension of non-uniform subdivision from a single univariate B-spline basis functions to the bivariate case is straightforward due to their tensor product structure. The bivariate subdivision can be written as:

$$B_{p,q}(u,v) = \sum_{i=0}^{p+1} \sum_{j=0}^{q+1} W_{i,p} \bar{N}_{i,p}(u) W_{j,q} \bar{N}_{j,q}(v). \tag{5.1}$$

Where p, q are the polynomial degrees, u, v are the parametric coordinates in each parametric direction, i, j denotes the position of the subdivided basis functions in the tensor product structure, and $W_{i,p}, W_{j,q}$ are the

weights of the subdivided basis functions. The weights are calculated using Algorithms 1 and 2. Fig. 5 illustrates the subdivided non-uniform bi-quintic basis functions resulting from the subdivision algorithms proposed in Section 3.

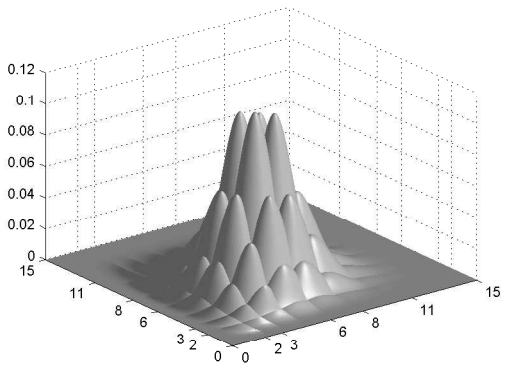
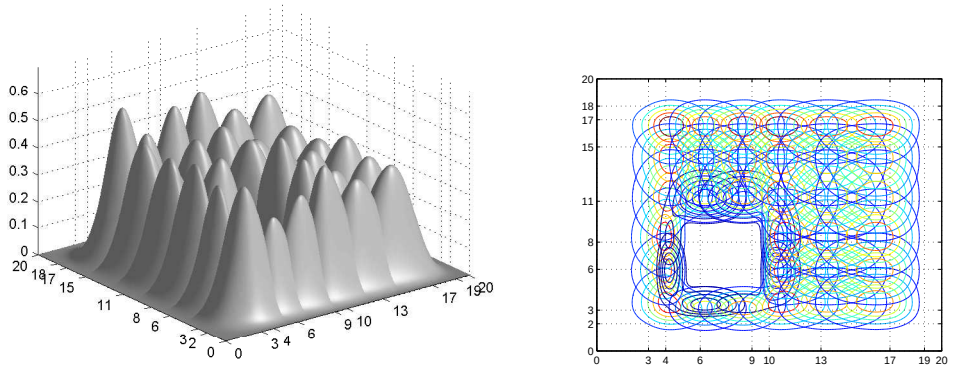


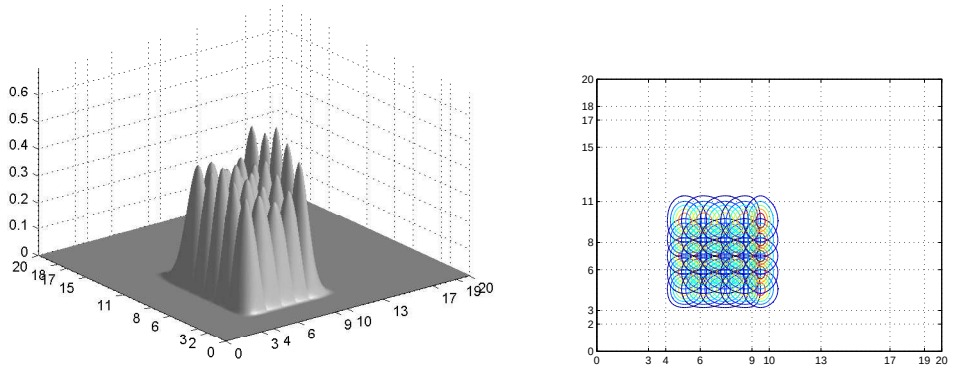
FIGURE 5. Subdivision of a non-uniform bi-quintic spline basis function

Similarly, the non-uniform hierarchical refinement approach can be extended to bivariate B-spline basis functions. Based on the subdivided bivariate basis functions calculated from Algorithms 1 and 2, the bivariate hierarchical refinement can be achieved using the steps illustrated in Section 4. Fig. 6 illustrates bivariate hierarchical refinement on non-uniform bi-cubic B-spline basis functions. The locally refined basis are linearly independent, form a partition of unity and suitable for approximation.

Given a knot interval marked to be refined, the proposed approach uses a slightly different way from those in THB-splines [8] to determine the refined domain. Note that the bases are the same as THB-splines, only the strategy to determine the refined domain is different. In order to achieve smaller localization, the proposed approach just adds the refined basis functions whose support overlaps the marked knot interval. If the present univariate refinement strategy is extended to the bivariate case, L-junction would appear during the refinement process. Thanks to the hierarchical approach, L-junction is acceptable in THB-splines as shown in Fig.2 of [8].



Level k=0



Level k=1

FIGURE 6. Non-uniform hierarchical refinement of bi-cubic B-spline basis functions

6 Approximation using non-uniform THB-splines

In this section, we compare the behavior of non-uniform high order THB-splines with uniform low order THB-splines, and the efficiency of the implemented subdivision algorithm with that of the classic knot insertion method. Their performances are tested on two classical approximation problems. Least-square fitting is iteratively implemented to approximate a given function until the residual error is within a specified tolerance. Given a test function and an initial knot vector defined on a domain, we iteratively approximate the test function using the hierarchically refined B-spline basis. The initial non-uniform meshes are manually constructed so that the meshes in singularity or sharp areas are denser than the meshes in the flat or smooth areas.The hierarchical refinement algorithm is de-

scribed as follows:

(a) Compute the least-square approximation using the initial B-spline basis.

(b) Flag all knot intervals whose error exceeds the given tolerance (30% of the maximum approximation error).

(c) Apply hierarchical refinement on the flagged knot intervals to generate one more level of hierarchical B-spline basis.

(d) Compute the least-square approximation using the refined hierarchical B-spline basis of step (c).

(e) Repeat from step (b) to (d) until the maximum residual error lies within the given tolerance.

We test two classical approximation problems using the above described approach, and compare the convergence of uniform low order THB-splines and non-uniform high order THB-splines. The first example compares uniform hierarchical refinement to non-uniform hierarchical refinement using the same order of cubic approximation. The second example compares low order non-uniform hierarchical refinement to high order non-uniform hierarchical refinement using quadratic and quartic approximations respectively. Note that the two test functions are taken from [8] with small modifications.

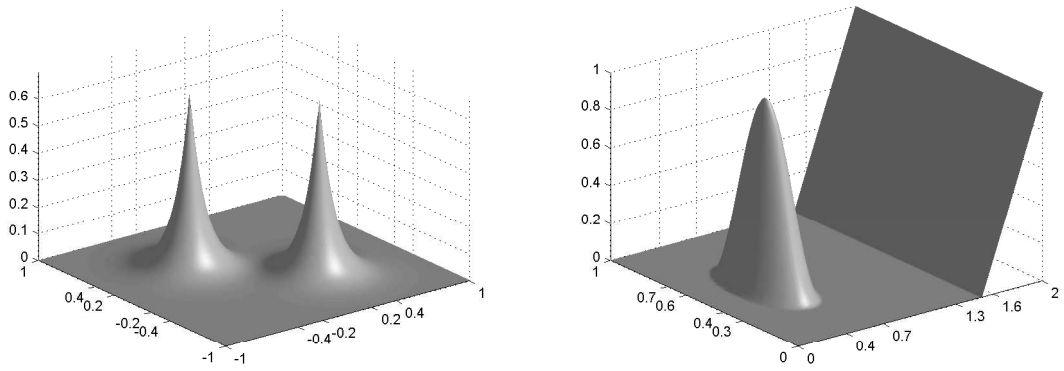


FIGURE 7. Functions considered in the examples

Example 1. We compute the least-squares approximations of the function

$$f(x) = \frac{2}{3\exp(\sqrt{(10x-3)^2+(10y+3)^2}} + \frac{2}{3\exp(\sqrt{(10x+3)^2+(10y-3)^2}},$$

as shown on the left of Fig. 7, by sampling the data points on a 150×150 uniform interval defined on the domain $[-1,\ 1] \times [-1,\ 1]$. The function is

approximated by uniform and non-uniform cubic B-splines basis functions respectively.

Fig. 8 shows the non-uniform approximation results and errors on locally refined hierarchical meshes of iterations $0, 2$ and 4. Fig. 9(a) gives the comparison of the convergence of uniform and non-uniform THB-splines on this function. Fig. 9(a) illustrates that the non-uniform THB-splines are more accurate than the uniform THB-splines for the same number of degrees of freedom. Given a specified tolerance, the non-uniform refinement uses few degrees of freedom than that of the uniform refinement. Hence, the non-uniform hierarchical refinement converges faster compared with the uniform hierarchical refinement algorithm.

The absolute convergence of the approximation seems slow. The main reason is that most of the degrees of freedom are used to satisfy the boundary condition of the B-splines. Only a small number of the degrees of freedom contribute to the approximation of the problem. Since the same original domains are used for all the hierarchical refinements, this will not affect the comparison of convergence rate for different hierarchical refinement methods. The convergence rate can be improved by choosing alternative error estimators and refinement strategies. A true error estimator or a posteriori error estimator may be used to replace the residual error [1]. The error threshold which determines the mesh intervals to refine should be related to the approximation problem. Given the marked mesh interval in a level, a suitable refinement strategy [8] can be chosen to determine the refinement region in the next level.

Example 2. The data points are sampled on a 150×150 uniform intervals from the function

$$f(x) = \begin{cases} 2x - 3, & 3/2 \leq x \leq 2, \\ 1/2\cos(4\pi\sqrt{q(x,y)}) + 1/2, & q(x,y) \leq 1/16, \\ 0, & \text{otherwise}, \end{cases}$$

where $q(x,y) = (x-1/2)^2 + (y-1/2)^2$, defined on the domain $[0,\ 2] \times [0,\ 1]$ as shown on the right of Fig. 7.

Fig. 10 compares the quadratic and quartic non-uniform hierarchical meshes and approximation errors after 5 steps of refinement. It illustrates that the quartic refinement has smaller errors than quadratic refinement with similar density of refined meshes. The performance of the approximation may depend on the function to be tested. For the linear part of the test function, quartic approximation gives worse results than quadratic approximation. Larger errors and denser meshes are produced by quartic refinement than those of quadratic hierarchical refinement. For the cosine

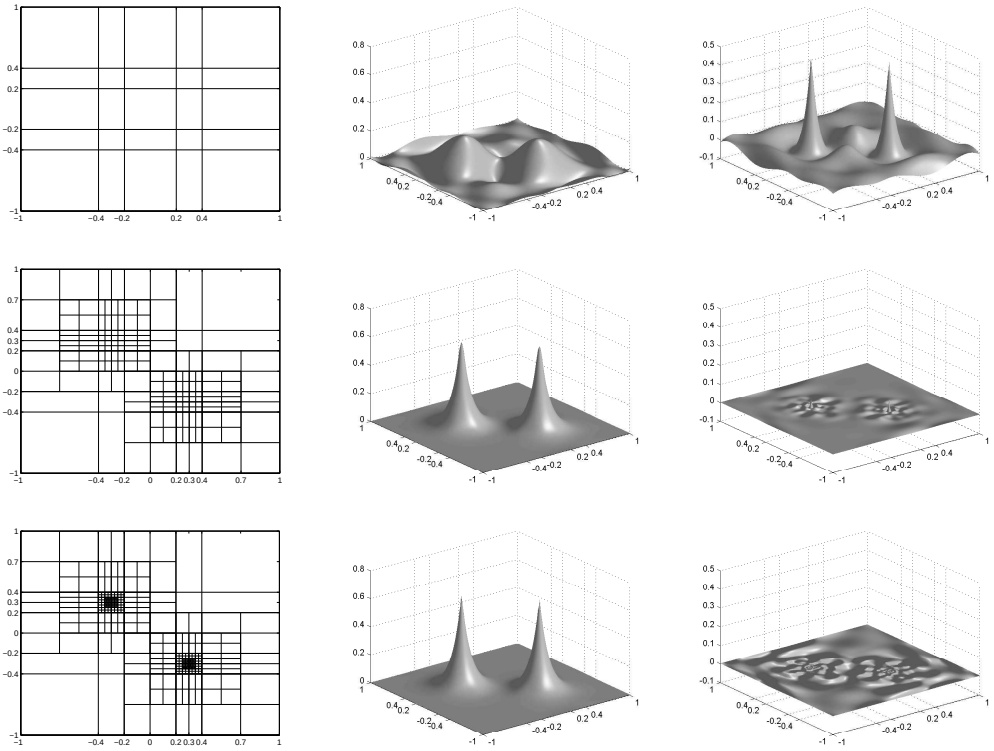
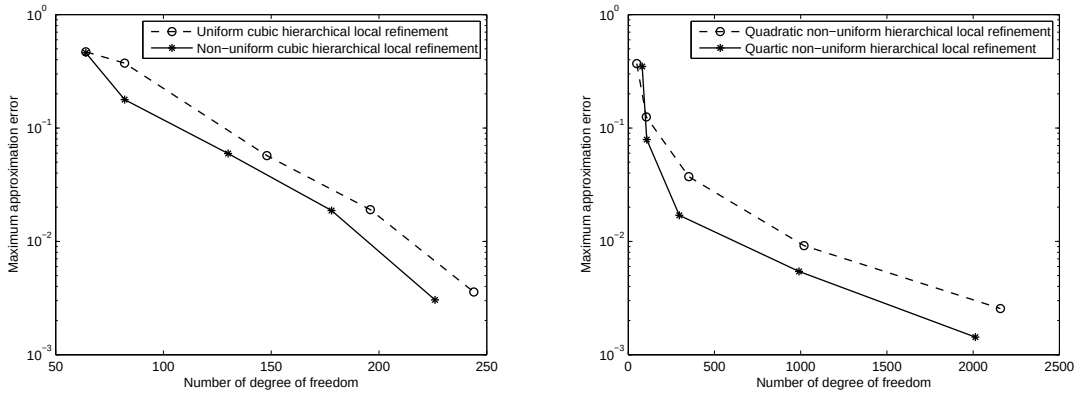
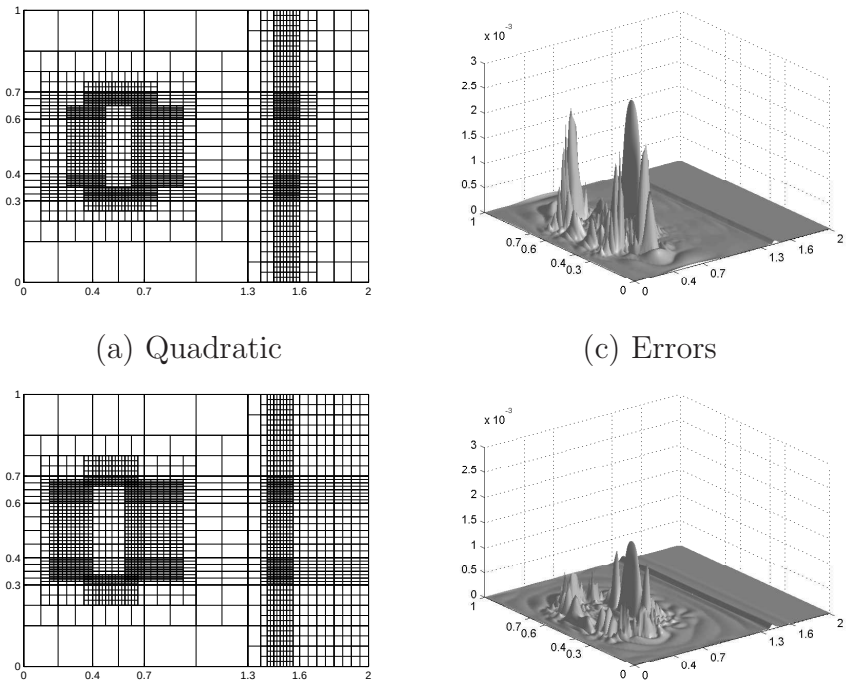


FIGURE 8. Non-uniform hierarchical spline approximations ($p = 3$) to the function in example 1 of iterations $0, 2$ and 4. In each sub-figure, left is the locally refined hierarchical knot meshes, middle is the approximation to the function, right is the corresponding errors.



(a) Example 1 (b) Example 2

FIGURE 9. Convergence of approximation using hierarchical adaptive local refinement



(a) Quadratic (c) Errors

(b) Quartic (d) Errors

FIGURE 10. Comparison of quadratic and quartic non-uniform hierarchical approximation for the function in example 2. In each sub-figure, left is locally refined hierarchical knot meshes, right is the corresponding approximation errors.

part, quartic hierarchical refinement gives smaller errors than quadratic hierarchical refinement, producing similar density of the hierarchical meshes.

Fig. 9(b) illustrates that high-degree(quartic) hierarchical refinement is more accurate than low-degree (quadratic) hierarchical refinement for the combined cosine and linear functions. Given a specified tolerance, the high-degree refinement uses fewer Degrees of Freedoms (DOFs) than that of low-degree refinement. Note that the number of DOFs is not proportional to the density of meshes for different degree of approximation. The reason is that high-degree DOFs need more mesh knots than low-degree DOFs. For example, a univariate quartic DOF needs 6 mesh knots while a univariate quadratic DOF only needs 4 mesh knots. Hence the quartic refinement uses fewer DOFs than quadratic refinement although they have similar density of meshes.

The above examples and discussions indicate that the implementation of non-uniform general degree THB-splines converges faster compared with the implementation of low degree uniform. The implementation is

flexible and makes it possible for approximation to use the same non-uniform B-spline basis functions as existing modelling tools have used.

7 Conclusion

This paper proposed an efficient implementation on non-uniform THB-splines of general degree. A non-uniform subdivision algorithm is developed to subdivide a single non-uniform B-spline basis function and applied to construct THB-splines for use in approximation. The non-uniform subdivision algorithm is efficient due to simultaneous generation of all refined basis functions. The proposed approach can be easily integrated into the existing uniform code for uniform B-splines, by simply applying the subdivision algorithm to generate the subdivision weights to replace uniform subdivision weights.

An application of the proposed implementation is approximating classic approximation functions. This is possible because the proposed implementation makes it possible for both design and approximation stages to use the same format of non-uniform B-spline basis functions. It has been illustrated that the non-uniform general degree THB-splines converge faster in comparison with the uniform low degree THB-splines for the two classic approximation problems tested.

In the future, the hierarchical non-uniform refinement implementation can be directly applied on the B-spline basis instead of through the subdivision of an individual basis function. A careful indexing scheme for selective nodes during subdivision can be configured using a selective knot insertion scheme. The non-uniform hierarchical refinement may be extended to trivariate splines for approximation of 3D solids.

References

[1] M. Ainsworth and X. Ma. Non-uniform order mixed FEM approximation: Implementation, post-processing, computable error bound and adaptivity. *Journal of Computational Physics*, 231(2):436 – 453, 2012.

[2] W. Boehm. Inserting new knots into B-spline curves. *Computer-Aided Design*, 12(4):199 – 201, 1980.

[3] T. J. Cashman, N. A. Dodgson, and M. A. Sabin. A symmetric, non-uniform, refine and smooth subdivision algorithm for general degree B-splines. *Computer Aided Geometric Design*, 26(1):94 – 104, 2009.

[4] E. Cohen, T. Lyche, and R. Riesenfeld. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing*, 14(2):87 – 111, 1980.

[5] R. Cripps, M. Hussain, and S. Zhu. Smooth polynomial approximation of spiral arcs. *Journal of Computational and Applied Mathematics*, 233(9):2227 – 2234, 2010.

[6] B. Cross and R. J. Cripps. G3 quintic polynomial approximation for generalised cornu spiral segments. *Journal of Computational and Applied Mathematics*, 236(13):3111 – 3122, 2012.

[7] D. R. Forsey and R. H. Bartels. Surface fitting with hierarchical splines. *ACM Trans. Graph.*, 14(2):134–161, Apr. 1995.

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

[9] C. Giannelli, B. Jüttler, and H. Speleers. Strongly stable bases for adaptively refined multilevel spline spaces. *Advances in Computational Mathematics*, 40(2):459–490, 2014.

[10] T. Hughes, J. 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, 2005.

[11] J. M. Lane and R. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-2(1):35–46, 1980.

[12] X. Ma and R. Cripps. Shape preserving data reduction for 3D surface points. *Computer Aided Design*, 43(8):902–909, 2011.

[13] X. Ma, S. Keates, Y. Jiang, and J. Kosinka. Subdivision surface fitting to a dense mesh using ridges and umbilics. *Computer Aided Geometric Design*, 32:5 – 21, 2015.

[14] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline techniques*. Springer, 2002.

[15] L. Ramshaw. Blossoming a connect-the-dots approach to splines. *Tech. Rep. 19, Digital Systems Research Center*, 1987.

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

[17] H. Speleers, C. Manni, F. Pelosi, and M. L. Sampoli. Isogeometric analysis with Powell-Sabin splines for advection-diffusion-reaction problems. *Computer Methods in Applied Mechanics and Engineering*, 221-222(0):132 – 148, 2012.

[18] A.-V. Vuong, C. Giannelli, B. Juttler, and B. Simeon. A hierarchical approach to adaptive local refinement in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 200(492):3554 – 3567, 2011.