S V E U Č I L I Š T E U S P L I T U
FAKULTET ELEKTROTEHNIKE, STROJARSTVA I BRODOGRADNJE

POSLIJEDIPLOMSKI DOKTORSKI STUDIJ STROJARSTVA

KVALIFIKACIJSKI ISPIT

GEOMETRIJSKO MODELIRANJE OBLAKA TOČAKA PARAMETARSKIM KRIVULJAMA I PLOHAMA

Domagoj Samardžić

Split, Srpanj 2024.

U N I V E R S I T Y O F S P L I T
FACULTY OF ELECTRICAL ENGINEERING, MECHANICAL ENGINEERING AND NAVAL ARCHITECTURE

POSTGRADUATE DOCTORAL MECHANICAL ENGINEERING STUDY

QUALIFICATION EXAM

GEOMETRIC MODELING OF POINT CLOUDS USING PARAMETRIC CURVES AND SURFACES

Domagoj Samardžić

Split, July 2024.

Contents

1. Introduction

Geometric modeling plays important part in computational engineering applications, such as *Computed-Aided Design* (CAD), *Computed-Aided Manufacturing* (CAM) and *Computed-Aided Engineering* (CAE). Geometric model is obtained using some geometric modeling software where new object is defined, or by parameterizing some existing engineering object through reverse engineering process which is main focus of this paper. Reverse engineering, in context of CAD, is a process of obtaining geometric and physical representation of given object using CAD technologies. In first step of geometric representation, descriptive data of a object is obtained with scanning of a object. Scanned data is given in a form of *Point Cloud* (PC), which is basically set of triangulated points of scanned object. PC in some cases can contain very large set of points, depending on the scanner resolution, and sometimes highly dense and noisy data which sometimes require additional filtering of segmentation [1, 2]. PC of scanned object contains points coordinates and their triangulation, which is not applicable for any slightly advanced numerical procedure. Therefore, given data need to be represented with proper geometric and shape parameters for establishing representative and functional geometric object [3]. Resulting set of parameters needs to be compact and minimal, but sufficient enough to represent initial PC. Achieving this parameterization often includes sets of highly nonlinear numerical and optimization procedures.

Second step is choice of adequate parametric model for data description and shape optimization. The choice of parametric model is arbitrary, but in certain cases some model shape variables can be insufficient or lead to high dimensionality of design space [3]. This can lead to many discontinuities or numerical problems. Mostly, choice of parametric model is a matter of expertise. After defining the model, next step is to fit data with given model. Fitting can be done either by approximation or interpolation. Approximation is mostly used, because interpolation requires much higher number of shape parameters. This leads to highly time consuming, computationally expensive procedures and possible discontinuities of the model. In that case, relatively "cheaper" approximation is used which results with satisfying fit.

With fitting, parametric model is obtained but it can be further enhanced [3]. Certain geometric features (edges, peaks, etc.) can be extracted from PC and refitted, because features usually represent locations with higher geometric error, i.e. distance between fitted model and input data. Features are extracted using some statistical based methods [4] and shape parameters are usually densely positioned around them to minimize error.

This procedure allows to perform of shape synthesis of initial PC data into finite geometric

(a) 3D geometric parameterization of portion of small cylinder head.

(b) 3D geometric parameterisation of formula-student body shell.

Figure 1.1: Examples of geometric parameterisations of engineering objects [3].

CAD model, which is a tool of some geometric modeling software. Obtained CAD model can than be manufactured again or used for some geometric or numerical analysis. Further on, it can be enhanced or modified through some numerical analysis as FEA or CFD, depending on the excellence criteria of the desired application.

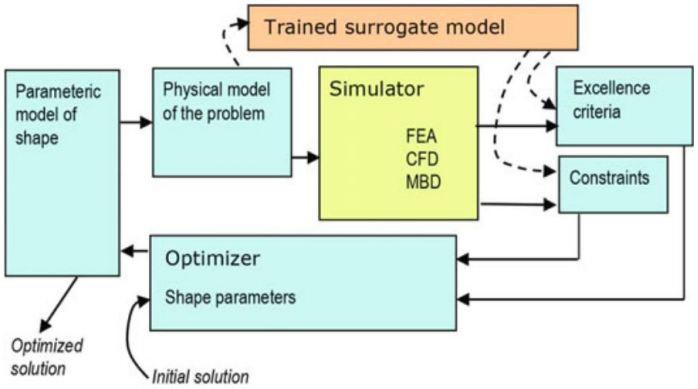


Figure 1.2: Changing initial shape toward desired criteria [3].

In this paper, parametric models and optimization procedures required for defining CAD model will be discussed. Standard CAD parametric models will be defined, along with some advanced, adaptive models. Fitting procedures and algorithms regarding given parametric models will also be described. Standard CAD models along with geometric algorithms required for definition of advanced parametric models are addressed in Chapter 2. Fitting procedures of standard and advanced parametric models are explained in Chapter 3 and Chapter 4, respectively.

2. Parametric Curves and Surfaces

In CAD technologies, relatively complex geometric data sets which cannot be described with some established mathematical functions (lines, circles, cylinders, spheres, etc.) require proper geometrical parameterization. For any set of geometric data points, shape parameterization can be achieved with piecewise polynomial parametric models. This chapter provides detailed description of these models, such as *de facto* standard parametric models (Bèzier, B-Spline and NURBS) and some advanced, adaptive parametric models based on standard B-Spline or NURBS. Also, fundamental geometric algorithms for manipulation of standard models are explained. These algorithms make necessary contribution for establishing adaptive models as T-Splines or hierarchical B-Splines.

2.1. Standard Parametric Curves and Surfaces

2.1.1. Bèzier Curve and Surface

One of the first polynomial parametric curves, introduced by Pierre Bèzier [5], is Bèzier curve, named after him.
An n-th degree Bèzier curve is defined by [6]

$$\mathbf{C}(u) = \sum_{i=0}^{n} B_{i,n}(u)\mathbf{Q}_i, \tag{2.1}$$

where u is parametric value, usually defined as $0 \leq u \leq 1$. Notation of parametric value as $0 \leq u \leq 1$ will be used throughout this paper. The basis or blending functions $B_{i,n}$ are standard n-th degree *Bernstein* polynomials, introduced by S. Bernstein [7], and geometric coefficients denoted as $\mathbf{Q}_i$ are called *control points* (in most literature, control points are denoted as $\mathbf{P}_i$, but in this paper mostly $\mathbf{Q}_i$ notation will be used).
Bernstein polynomials are defined explicitly as [6, 8]

$$B_{i,n} = \binom{n}{i} u^i (1-u)^{n-i} = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i}. \tag{2.2}$$

Polygon formed by control points $\{\mathbf{Q}_i\}$ is called *control polygon*, which approximates the final shape of the curve.
Basis functions determines the geometric characteristics of the curve with its properties [6]. In

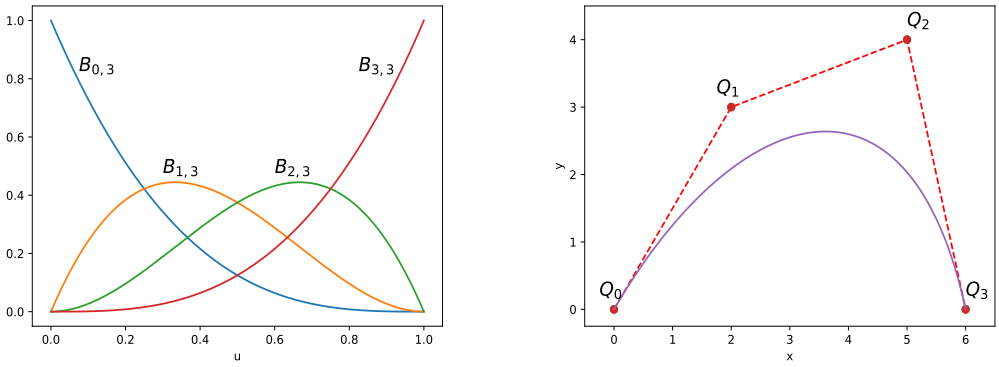


Figure 2.1: Bernstein polynomials and Bèzier curve with degree $n = 3$.

this case, properties of Bernstein polynomials are:

P_B1. Non-negativity: $B_{i,n} \geq 0$, for all i, n and for any $u \in [0,1]$.

P_B2. Partition of Unity (PoU): $\sum_{i=0}^{n} B_{i,n} = 1$ for any $u \in [0,1]$.

P_B3. For $u = 0$ and $u = 1$, it is valid that $B_{0,n}(0) = B_{n,n}(1) = 1$.

P_B4. $B_{i,n}$ attains one minimum in $u \in [0,1]$ and that is at $u = i/n$.

P_B5. Symmetry: for any number of basis functions n, set of $B_{i,n}(u)$ is symmetric with respect to parametric value $u = 1/2$.

Properties of Bèzier curve defined with Bernstein polynomials are [6, 8]:

P_C1. Degree of curve is defined by $n+1$ control points.

P_C2. Curve passes through first and last control point, $\mathbf{Q}_0 = \mathbf{C}(0)$ and $\mathbf{Q}_n = \mathbf{Q}(1)$.

P_C3. Endpoint ($u = 0$ and $u = 1$) tangent directions of the curve are parallel to directions of first two and last two control points polygon, i.e. $\mathbf{C}'(0) = \overline{\mathbf{Q}_1 - \mathbf{Q}_0}$ and $\mathbf{C}'(1) = \overline{\mathbf{Q}_n - \mathbf{Q}_{n-1}}$.

P_C4. Convex hull property: curve is contained inside convex hull defined by control points polygon.

P_C5. Variation diminishing property: no straight line intersects the curve more times than it intersects the control polygon.

P_C6. Affine invariance: curve is invariant to affine transformations (translations, rotations, scaling, ...), i.e. affine transformations to the curve can be applied by applying it to the control points.

P_C7. Changing the position of any control point $\mathbf{Q}_i$ changes the global picture of the curve, which corresponds to the non-negativity property of Bernstein polynomials

Most of the basis function and curve properties can be observed on Fig. 2.1.
Bernstein polynomials derivation is defined as

$$B'_{i,n}(u) = \frac{dB_{i,n}(u)}{du} = n\left(B_{i-1,n-1}(u) - B_{i,n-1}(u)\right), \tag{2.3}$$

from which expression Bèzier curve derivative can be obtained

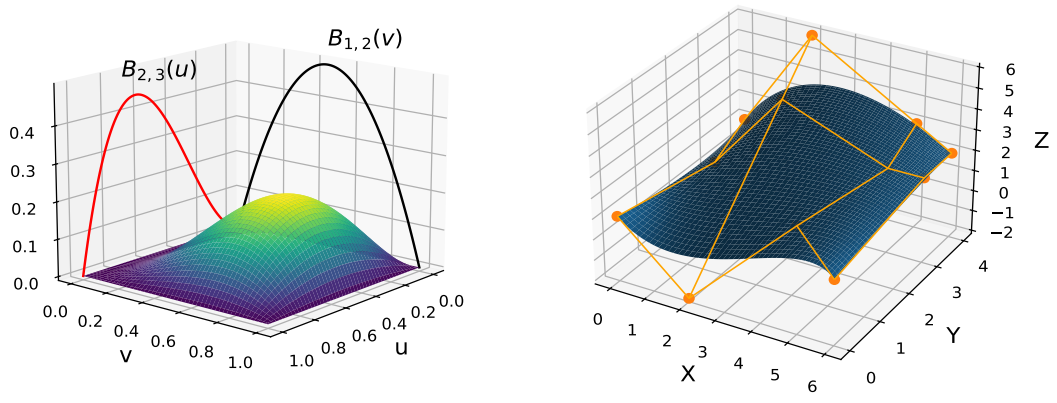
$$\mathbf{C}'(u) = \sum_{i=0}^{n} B'_{i,n}\mathbf{Q}_i = n\sum_{i=0}^{n} B_{i,n-1}(u)\left(\mathbf{Q}_{i+1} - \mathbf{Q}_i\right). \tag{2.4}$$

Surface is vector-valued function of two parameters, u and v, and represents mapping of region in (u,v) parametric plane into 3D Euclidean space [6]. Surfaces are mostly defined in tensor product scheme. Basis functions are bivariate functions of u and v, obtained by tensor product of univariate basis functions. Therefore, Bèzier surface is represented as tensor product of univariate Bernstein polynomials $B_{i,n}(u)$ and $B_{j,m}(v)$ (Fig. 2.2a), multiplied with grid of control points $\mathbf{Q}_{i,j}$, i.e. control net [9] (Fig. 2.2b).

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_{i,n}(u)B_{j,m}(v)\mathbf{Q}_{i,j}, \tag{2.5}$$

where parametric values are mostly defined as $0 \leq u,v, \leq 1$. Control net $\mathbf{Q}_{i,j}$ is given as

$$\mathbf{Q}_{i,j} = \begin{bmatrix} Q_{0,0} & \cdots & Q_{0,m} \\ \vdots & \ddots & \vdots \\ Q_{n,0} & \cdots & Q_{n,m} \end{bmatrix} \in \mathbb{R}^{3(n+1)\times(m+1)}. \tag{2.6}$$



(a) (b)

Figure 2.2: (a) Bèzier tensor product of cubic $B_{2,3}(u)$ and quadratic $B_{1,2}(v)$ basis functions, (b) example of tensor product Bèzier surface with control net.

2.1.2. The de Casteljau Algorithm

When it's necessary to determine the value of Bèzier curve $\mathbf{C}(u)$, widely used and numerically stable algorithm is *de Casteljau algorithm*. Algorithm is using recursive linear operator which subdivides control polygon in a ratio defined by parametric value u [8]. Main idea is to split every polygon leg in ratio $u:(1-u)$ which creates new polygon. Procedure is repeated $n-1$ times until last $(n-1)$ polygon intersects curve, which is wanted value $\mathbf{C}(u)$ [10]. Subdivision points $\mathbf{Q}_i^r$ are usually evaluated by recursive affine combinations

$$\mathbf{Q}_i^{r-1}(1-u)+\mathbf{Q}_{i+1}^{r-1}(u)=\mathbf{Q}_i^r. \tag{2.7}$$

Given algorithm can be arranged in triangular scheme as

$$\begin{array}{cccc} \mathbf{Q}_0^0 & & & \\ \mathbf{Q}_1^0 & \mathbf{Q}_0^1 & & \\ \vdots & \vdots & \ddots & \\ \mathbf{Q}_n^0 & \mathbf{Q}_{n-1}^1 & \cdots & \mathbf{Q}_0^n=\mathbf{Q}(u) \end{array} \tag{2.8}$$

Last polygon leg value for a given parametric coordinate u results with wanted curve evaluation $\mathbf{C}(u)=\mathbf{Q}_0^n$. In this manner, any point on a curve $\mathbf{C}(u)$ can be evaluated and thus entire curve can be obtained.

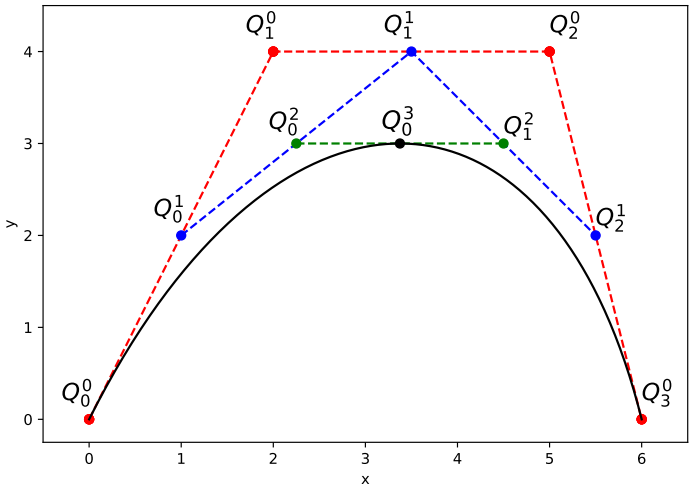


Figure 2.3: De Casteljau algorithm for cubic Bèzier at $u=1/2$.

On Fig. 2.3 de Casteljau algorithm is presented for cubic Bèzier where final control point $\mathbf{Q}_0^3$ represents parametric curve value $\mathbf{C}(1/2)=[C_x(1/2),C_y(1/2)]$.
The de Casteljau algorithm can be utilized in several applications, such as curve/surface subdivision in two or more curves/surfaces without changing the initial geometry. In this way, Bèzier parameterization enables local influence on desired segments.

2.1.3. B-Spline Curve and Surface

Bèzier curve is globally defined in entire parametric domain, which means that slightest change in parameters alters entire curve. Local influence can be obtained through subdivision of curve, but to avoid this numerically expensive procedure, *B-Spline* curves was introduced [6]. B-Spline is defined as piecewise polynomial curve in parametric domain, where the domain consists of $m-1$ parametric values $0 \leq \overline{u}_0 \leq \overline{u}_1 \leq \cdots \leq \overline{u}_{m+1} \leq \overline{u}_m \leq 1$ called *knots* $\overline{u}_i$.
An p-th degree B-Spline curve is defined as [6, 8]

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i, \tag{2.9}$$

where the parametric value u are mostly defined in $u \in [0,1]$, $\{\mathbf{Q}_i\}$ is vector of control points and $N_{i,p}$ are B-Spline basis functions. Basis functions are defined with Cox-de Boor recursive formula [11, 12]

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1, & \text{if } \overline{u}_i \leq u < \overline{u}_{i+1} \\ 0, & \text{otherwise} \end{cases} \\ N_{i,p}(u) &= \frac{u-\overline{u}_i}{\overline{u}_{i+p}-\overline{u}_i} N_{i,p-1}(u) + \frac{\overline{u}_{i+p+1}-u}{\overline{u}_{i+p+1}-\overline{u}_{i+1}} N_{i+1,p-1}(u). \end{aligned} \tag{2.10}$$

Visual example of Cox-de Boor recurrence can be seen on Fig. 2.4, where is shown how a cubic polynomial is constructed recursively from zeroth polynomial, with equivalent triangular scheme on Fig. 2.5.
First use of Cox-de Boor recurrence equation was by Gordon and Riesenfeld [13], where it was used to define parametric B-Spline curve. Ever since, it's most popular way of defining B-Spline basis functions. Alternative ways of defining basis function are described closely by de Boor in [14]. B-Spline basis function can be evaluated with de Boor algorithm [12], which is using Cox-de Boor recurrence equation (Eq. 2.10) and only nonzero values are taken into account. It's derived from de Casteljau algorithm into more general approach to locally defined basis functions.
Knot vector $\{\overline{\mathbf{u}}\}$ is usually defined as

$$\overline{\mathbf{u}} = \{\underbrace{0,\ldots,0}_{p+1}, \overline{u}_{p+1}, \ldots, \overline{u}_{m-p-1}, \underbrace{1,\ldots,1}_{p+1}\}, \tag{2.11}$$

where $p+1$ multiplicity of first and last knot values ensures that B-Spline curve passes through first and last control point Q_i. Knot vector values can be equidistantly spaced or otherwise.
In contrast to Bèzier curve, additional knot vector $\{\overline{u}_i\}$ increases the number of parameters required to define parametric curve and with that better parameterization of geometry. On the

other hand, increase in the number of parameters also increases computational efforts.

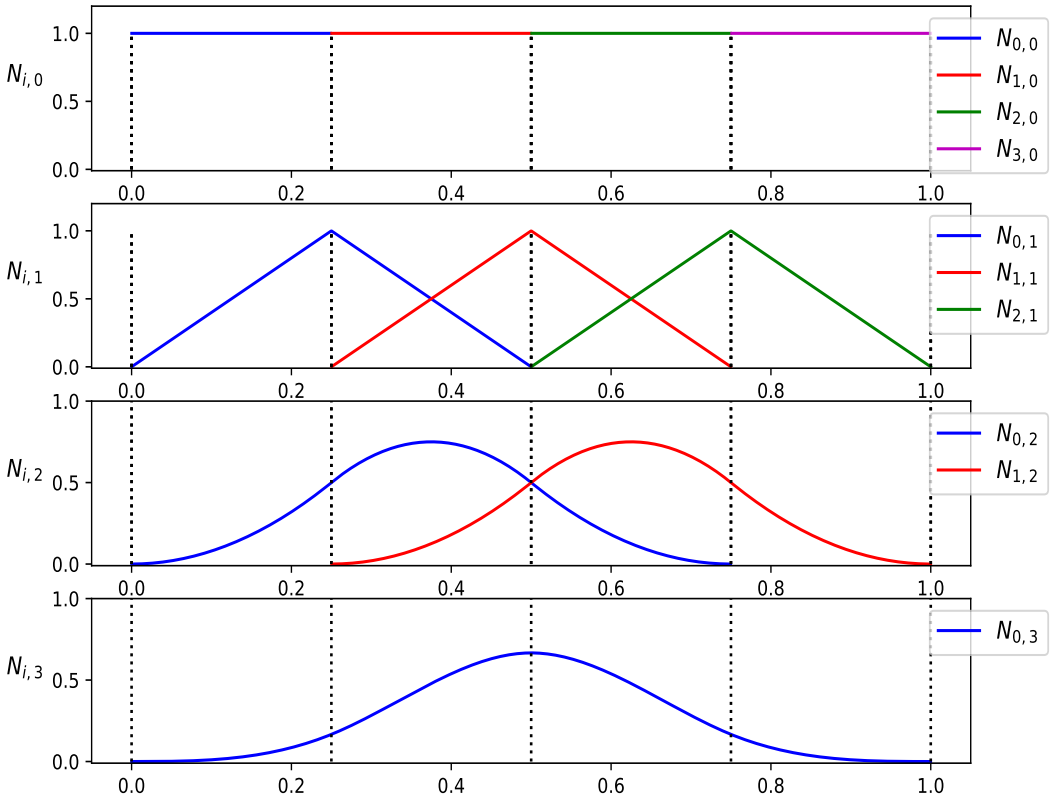


Figure 2.4: Recursive calculation of cubic B-Spline basis function on knot interval $\bar{\mathbf{u}} = \{0, 0.25, 0.5, 0.75, 1\}$, from zeroth to third polynomial degree.

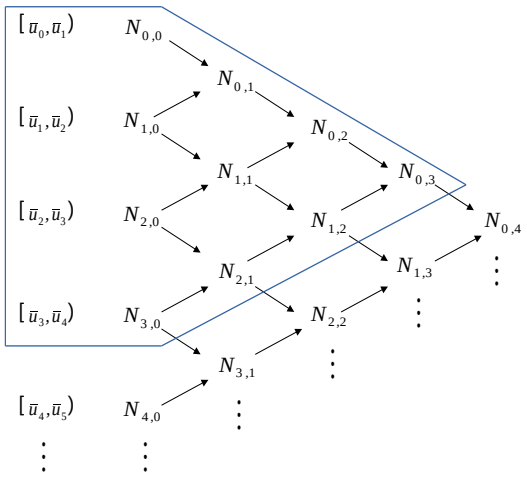


Figure 2.5: Triangular scheme of recursive Cox-de Boor formula. Blue polygon represents triangular scheme of Fig. 2.4.

Characteristics of B-Spline basis functions are [6, 8, 14]:

P_B1. $N_{i,p}$ is polynomial function of degree p.

P_B2. Local support property: any $N_{i,p}$ is nonzero in interval $[\overline{u}_i, \overline{u}_{i+p+1})$, i.e. $N_{i,p}$ is only defined in interval $[\overline{u}_i, \overline{u}_{i+p+1})$. This property can be seen on Fig. 2.4 and Fig. 2.5.

P_B3. Non-negativity property: $N_{i,p} \geq 0$, for any i, p and u.

P_B4. In any arbitrary knot span $[\overline{u}_i, \overline{u}_{i+1})$, there is most $p+1$ basis functions $N_{i,p}$ that are nonzero.

P_B5. For any arbitrary knot span $[\overline{u}_i, \overline{u}_{i+1})$, $\sum_{j=i-p}^{i} N_{j,p} = 1$ for any $u \in [\overline{u}_i, \overline{u}_{i+1})$ (PoU).

P_B6. For a number of knots $m+1$ and polynomial degree p, there exist $n+1$ basis functions which satisfy the condition $m = n+p+1$.

P_B7. $N_{i,p}$ defined in knot interval $[\overline{u}_i, \overline{u}_{i+p+1})$ is $p-k$ continuously differentiable at a knot values, where k is multiplicity of the knot. Therefore, degree p increases continuity and increasing knot multiplicity k decreases continuity.

P_B8. If some knot $\overline{u}_i$ is multiplied k times, then there is most $p-k+1$ nonzero basis function at $\overline{u}_i$ value.

P_B9. Derivative of basis function is given by

$$N'_{i,p}(u) = \frac{p}{\overline{u}_{i+p} - \overline{u}_i} N_{i,p-1}(u) + \frac{p}{\overline{u}_{i+p+1} - \overline{u}_{i+1}} N_{i+1,p-1}(u). \tag{2.12}$$

P_B10. Knot vector $\overline{\mathbf{u}}=\{\underbrace{0,\cdots 0}_{p+1}, \underbrace{1,\cdots 1}_{p+1}\}$ gives Bernstein polynomials of degree p. Therefore, with appropriate knot vector $\{\overline{\mathbf{u}}\}$, B-Spline basis functions can represent Bernstein polynomials.

P_B11. $N_{i,p}$ are linearly independent, i.e. they are defining a basis for the vector space of basis functions.

$$\sum_{i=0}^{N} \alpha_i N_{i,p}(u) = 0 \quad \text{for any } \; u \in \mathbb{R}. \tag{2.13}$$

Eq. 2.13 equals to 0 only if every linear coefficient $\alpha_i = 0$, which defines linear independency between basis functions.

Like B-Spline basis functions, characteristics of B-Spline curve are [6, 8]:

P_C1. B-Spline curve $\mathbf{C}(u)$ is piecewise polynomial curve (because of basis functions), with the degree p, $n+1$ control points and $m+1$ knots in knot vector. Like B-Spline basis functions, curve is also defined by relation $m = n+p+1$.

P$_C$2. If $n = p$ and knot vector $\overline{\mathbf{u}}_i = \{\underbrace{0,\cdots,0}_{p+1},\underbrace{1,\cdots,1}_{p+1}\}$, then B-Spline curve is Bèzier curve.

P$_C$3. Endpoints interpolation, i.e. $\mathbf{C}(0) = \mathbf{Q}_0$ if $\overline{u}_0 = \cdots = \overline{u}_p = 0$ and $\mathbf{C}(1) = \mathbf{Q}_n$ if $\overline{u}_{m-p} = \cdots = \overline{u}_m = 1$.

P$_C$4. Affine transformations of the curve are applied through the control points.

P$_C$5. Convex hull property: curve is contained inside control polygon convex hull.

P$_C$6. Local modification scheme: modification of $\mathbf{Q}_i$ only changes curve in the interval $[\overline{u}_i, \overline{u}_{i+p+1})$. Because by Eq. (2.9) every control point has its own basis function which is nonzero in interval $[\overline{u}_i, \overline{u}_{i+p+1})$.

P$_C$7. Control polygon represents piecewise linear approximation of the curve. So with lower degree of the curve, the curve is better approximation of the control polygon. For $p = 1$, control polygon is the curve.

P$_C$8. Variation dimishing property: no straight line intersects the curve more times than it intersects control polygon.

P$_C$9. Curve is infinitely differentiable for an u value in any knot interval and at least $p-k$ times continuously differentiable at a knot of k multiplicity.

P$_C$10. Inserting the new knot in knot vector or raising the degree of the curve results with new control point because of relation $m = n + p + 1$.

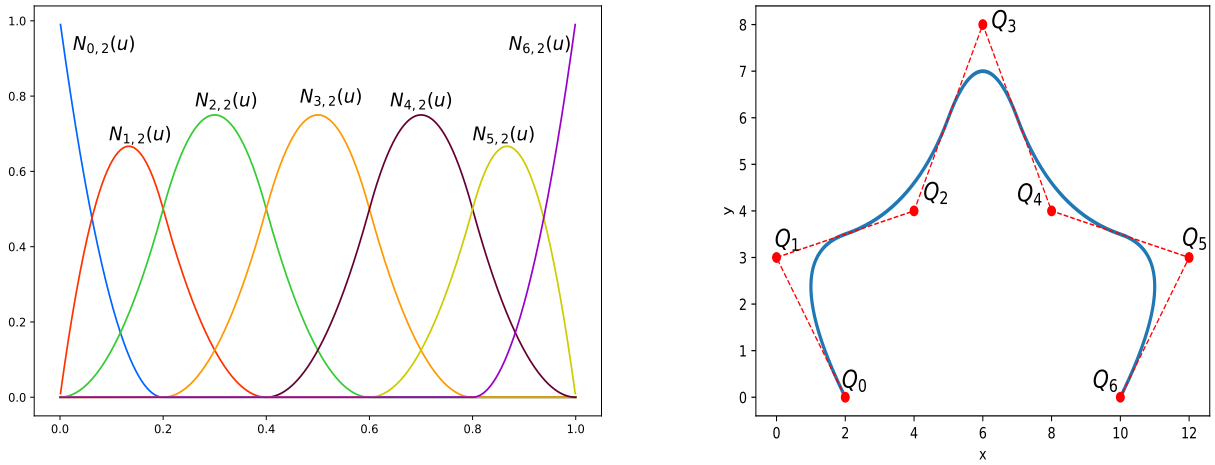
P$_C$11. The kth derivative of B-Spline curve is given by

$$\mathbf{C}^{(k)}(u) = \sum_{i=0}^{n} N_{i,p}^{(k)}(u)\mathbf{Q}_i. \tag{2.14}$$

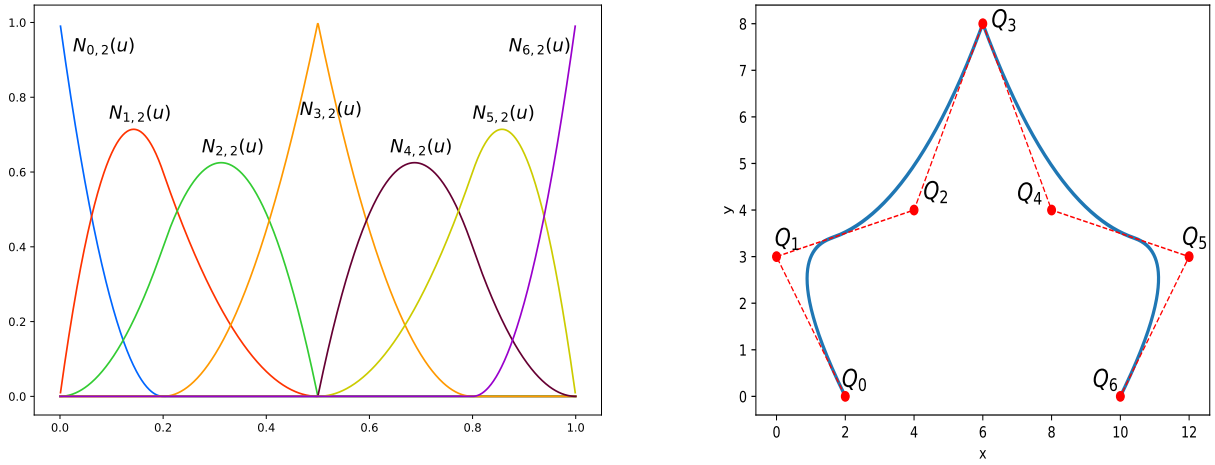
Calculation of B-Spline curve value is presented in several simplified steps in a Algorithm 2.1 [6, 14]:

Algorithm 2.1 B-Spline curve calculation

- **Input parameters**: $u \in \overline{\mathbf{u}}$
  1. Knot span $[\overline{u}_i, \overline{u}_{i+1}]$ where the u is located has to be found.
  2. For an i-th basis function and p-th degree, $N_{i,p}$ is calculated with Eq. (2.9.)
  3. $N_{i,p}$ value is multiplied with corresponding control point $\mathbf{Q}_i$ and added to the sum of B-Spline curve (Eq. (2.8)).
- Steps 1-3 are repeated for every $p+1$ nonzero $N_{i,p}$ in interval $[\overline{u}_i, \overline{u}_{i+1}]$.
- **Output parameter**: $\mathbf{C}(u)$ for a given $u \in \mathbb{R}blu$



(a) Knot vector $\overline{\mathbf{u}} = \{0,0,0,0.2,0.4,0.6,0.8,1,1,1\}$.



(b) Knot vector $\overline{\mathbf{u}} = \{0,0,0,0.2,0.5,0.5,0.8,1,1,1\}$.

Figure 2.6: Two quadratic B-Spline curves and its degree basis functions with exact control points but different knot vectors.

B-Spline surface is obtained through bidirectional net of control points (control net), with two knot vectors $\overline{u}_i$ and $\overline{v}_i$ and the tensor product of univariate B-Spline basis functions $N_{i,p}$ (Fig. 2.7) [6, 8]

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{Q}_{i,j}, \tag{2.15}$$

where the knot vectors are usually defined as

$$\overline{\mathbf{u}} = \{\underbrace{0,\ldots,0}_{p+1},\overline{u}_{p+1},\ldots,\overline{u}_{r-p-1},\underbrace{1,\ldots,1}_{p+1}\}, \tag{2.16a}$$

$$\overline{\mathbf{v}} = \{\underbrace{0,\ldots,0}_{q+1},\overline{u}_{q+1},\ldots,\overline{u}_{s-q-1},\underbrace{1,\ldots,1}_{q+1}\}. \tag{2.16b}$$

where $\overline{\mathbf{u}}$ has $r+1$ knots with polynomial degree p and $n+1$ control points (or basis functions). Respectively, $\overline{\mathbf{v}}$ has $s+1$ knots with polynomial degree q and $m+1$ control points.
B-Spline surface follows same characteristics as B-Spline curve, analogously defined for bivariate parametric domain (u,v).

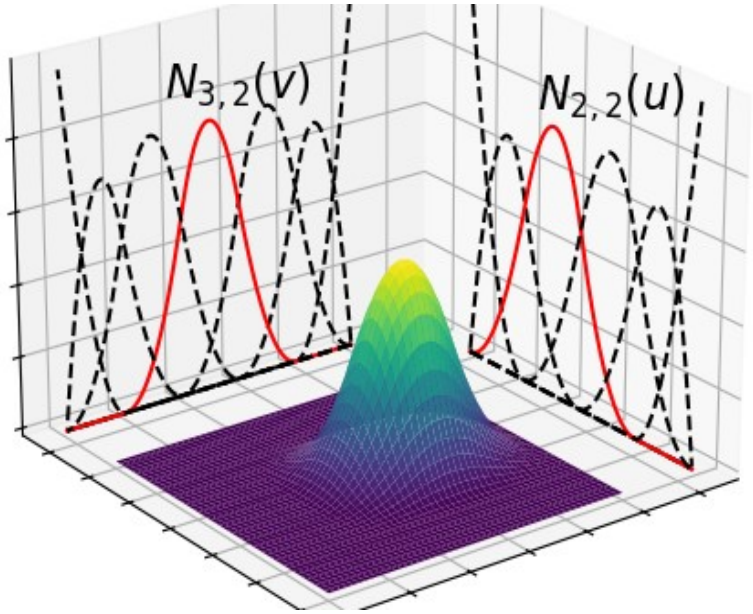


Figure 2.7: Tensor product of quadratic B-Spline basis functions $N_{2,2}(u)$ and $N_{3,2}(v)$.

2.1.4. NURBS Curve and Surface

B-Spline parameterization offers many possibilities when it comes to geometric modeling, but due to polynomial basis functions they are unable to represent rational functions and shapes, such as circles, conic sections or some free-form curves [15, 16]. To overcome this difficulty, rational B-Spline basis functions were introduced by Versprille in his dissertation [17], and their use in curves and surface representation was expanded as NURBS (Non-Uniform Rational B-Spline) curves [15, 16].
A p-th degree NURBS curve is defined as [6, 15, 16]

$$\mathbf{C}(u) = \sum_{i=0}^{n} R_{i,p}(u)\mathbf{Q}_i, \tag{2.17}$$

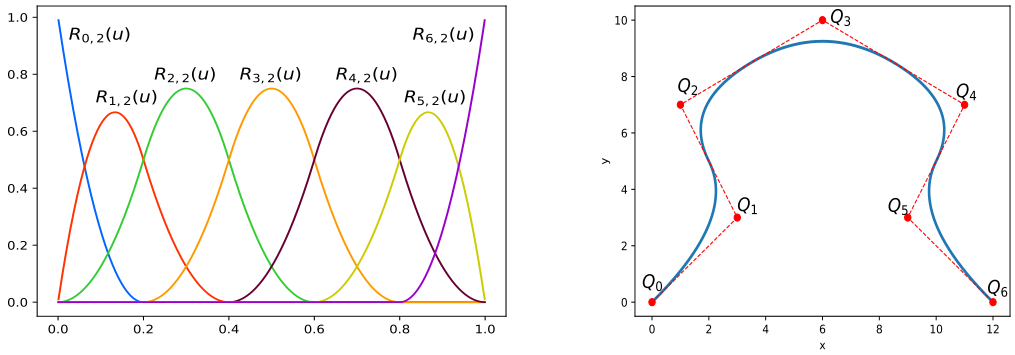
where $R_{i,p}$ are piecewise rational B-Spline basis functions defined as

$$R_{i,p}(u) = \frac{N_{i,p}(u)w_i}{\sum_{j=0}^{n} N_{j,p}w_j}. \tag{2.18}$$

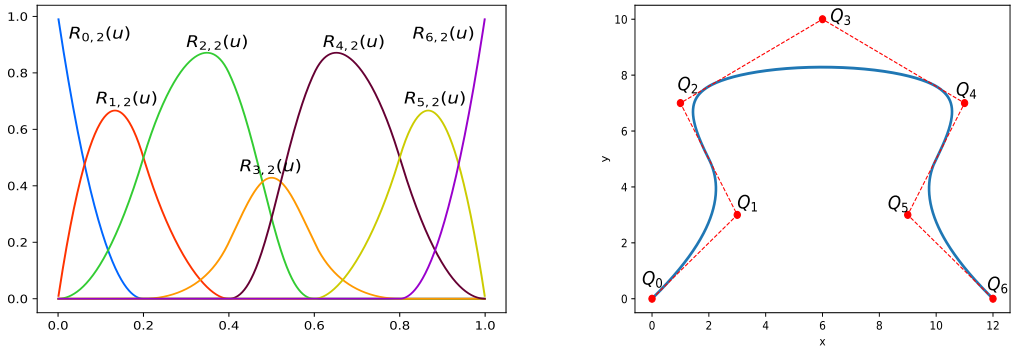
In contrast to B-Spline curve, NURBS curve adds more parameters to the curve definition. These new parameters are defined through *weights* vector $\mathbf{w} = \{w_0, ..., w_n\}$, where each weight corresponds to one basis function or control point. Weights add more flexibility and degrees of

freedom in curve modeling and consequently, NURBS model becomes more computationally expensive and numerically complex.

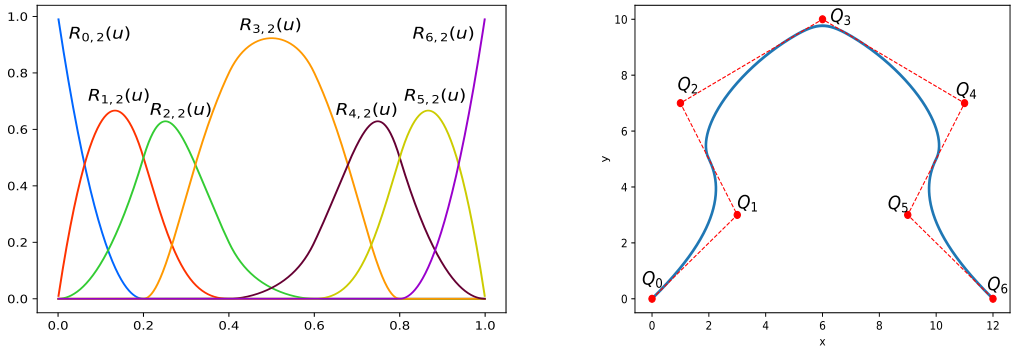
NURBS curve follows same characteristics as B-Spline curve, described in Sec. 2.1.3, except one additional remark regarding weights in NURBS curve. When all weights are set to $w_i = c$, for every $i = 0, \ldots, n$ and where c is any constant $c \in \mathbb{R} \setminus \{0\}$, then rational B-Spline basis functions represent B-Spline basis functions and therefore NURBS curve represents B-Spline curve. In most applications, weights are $w_i > 0$.



(a) Weights vector $\mathbf{w} = \{1, 1, 1, 1, 1, 1, 1\}$.



(b) Weights vector $\mathbf{w} = \{1, 1, 1, 0.25, 1, 1, 1\}$.



(c) Weights vector $\mathbf{w} = \{1, 1, 1, 4, 1, 1, 1\}$.

Figure 2.8: Quadratic NURBS basis functions and curve with same control points and knot vector, shown through different weights vector.

NURBS surface is defined as [6, 15, 16]

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)\mathbf{Q}_{i,j}, \tag{2.19}$$

where the bivariate rational basis functions are given by

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}}. \tag{2.20}$$

NURBS surface has same parameters as B-Spline surface (Sec. 2.1.3), with exception of weight net $w_{i,j}$ defined as

$$\mathbf{w} = \begin{bmatrix} w_{0,0} & \cdots & w_{0,m} \\ \vdots & \ddots & \vdots \\ w_{n,0} & \cdots & w_{n,m} \end{bmatrix} \in \mathbb{R}^{3(n+1)\times(m+1)}. \tag{2.21}$$

NURBS surface holds identical characteristics as B-Spline surface and analogously, it can represent B-Spline surface and therefore Bèzier surface (Fig. 2.9).

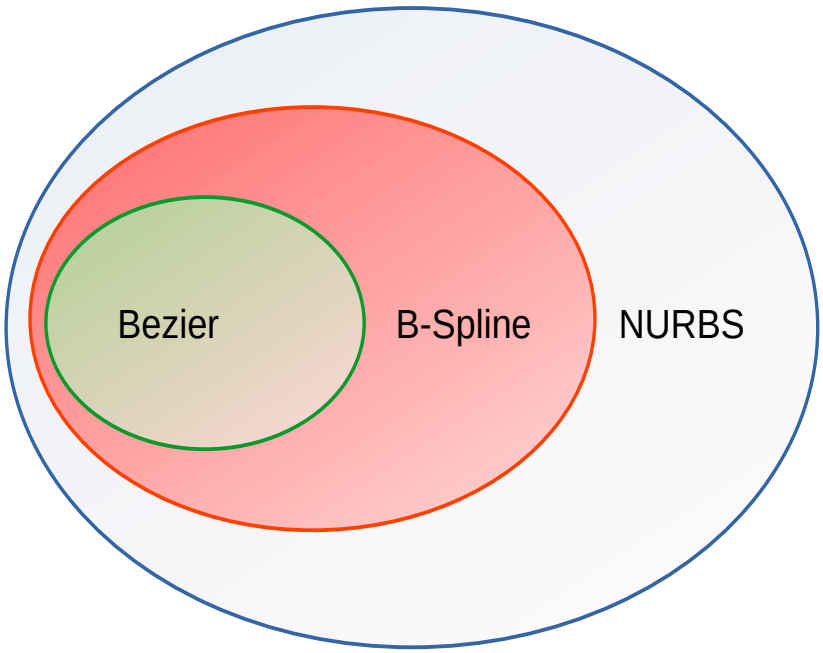


Figure 2.9: Relation between parametric models.

2.2. Fundamental Geometric Algorithms

So far mentioned parameterization methods can, in certain cases, produce satisfying parameterized geometric models. Sometimes, obtained model can require additional improvements which can reduce the error or be used in other applications, e.g. refinement or subdivision. In this chapter, several algorithms will be discussed which are used for parametric model improvements. Similarly, these methods can increase or decrease the number of parameters in model and thus change model complexity and computational efforts.

2.2.1. Knot Insertion Algorithm

NURBS model (B-Spline as well) improvements can be achieved through *knot insertion* algorithm, which refines initial knot vector and thus adds more parameters to the model (Sec. 2.1.3, property $P_B6.$). Knot insertion in one of the most important geometric algorithms, because it can be used for multiple applications such as: adding control points to increase flexibility, subdivision of the curve/surface, evaluation of curves/surfaces values and derivatives, etc. [6, 18]. Knot insertion algorithm was introduced simultaneously by Cohen et al. [18] and Boehm [19], where authors in [18] named the algorithm *Oslo algorithm* while algorithm in [19] has no particular title. Both algorithms work directly with B-Spline coefficients while Oslo algorithm can be expanded to the linear system which is defined by basis transformations between refined spaces, where every approach has its own pros and cons [20]. Both algorithms are derived from Cox-de Boor algorithm. For simplicity sake, single knot insertion will be closely described in this paper [21, 6, 8].
Initially, we assume that we have NURBS curve $\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}\mathbf{Q}_i$ which is defined on a knot sequence $\overline{\mathbf{u}} = \{u_i\}_{i=0}^{n+p+1}$. We want to insert knot $\overline{u}$ in some knot interval $\left[\overline{u}_j, \overline{u}_{j+1}\right)$ which will result in new knot sequence $\widehat{\mathbf{u}} = \{u_i\}_{i=0}^{n+p+2}$. New NURBS curve can then be formulated as

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}\mathbf{Q}_i = \sum_{i=0}^{n+1} \widehat{N}_{i,p}(u)\widehat{\mathbf{Q}}_i, \tag{2.22}$$

where the system of linear equation is obtained and which is to be solved.

$$\sum_{i=0}^{n} N_{i,p}\mathbf{Q}_i = \sum_{i=0}^{n+1} \widehat{N}_{i,p}(u)\widehat{\mathbf{Q}}_i. \tag{2.23}$$

For every $u \in \left[\overline{u}_j, \overline{u}_{j+1}\right)$ it's valid

$$\begin{aligned} N_{i,p}(u) &= \widehat{N}_{i,p}(u), \qquad i = 0, \cdots, j-p-1, \\ N_{i,p}(u) &= \widehat{N}_{i+1,p}(u), \quad i = j+1, \cdots, n. \end{aligned}$$

Upper equations and linear independence property of basis functions imply on equalities

$$\begin{aligned} \mathbf{Q}_i &= \widehat{\mathbf{Q}}_i, \qquad i = 0, ..., j-p-1, \\ \mathbf{Q}_i &= \widehat{\mathbf{Q}}_{i+1}, \quad i = j+1, ..., n. \end{aligned} \tag{2.24}$$

Expanding the Eq. (2.23) yields

$$\begin{aligned} &\widehat{\mathbf{Q}}_{j-p} = \mathbf{Q}_{j-p}, \\ &\widehat{\mathbf{Q}}_i = \alpha_i \mathbf{Q}_i + (1-\alpha_i)\mathbf{Q}_{i-1}, \qquad j-p+1 \leq i \leq j, \\ &\widehat{\mathbf{Q}}_{j+1} = \mathbf{Q}_j, \end{aligned} \tag{2.25}$$

where

$$\alpha_i = \begin{cases} 1, & i \leq j-p, \\ \frac{\overline{u}_i - u_i}{\overline{u}_{i+p} - \overline{u}_i}, & j-p+1 \leq i \leq j, \\ 0, & i \geq j+1. \end{cases} \tag{2.26}$$

Given procedure is mostly based on Boehm method [19]. Single knot procedure can be given in matrix form as $\widehat{\mathbf{Q}} = \mathbf{R}\mathbf{Q}$, where $\mathbf{R}$ is $(n+1) \times n$ *transfer matrix* [22]

$$\mathbf{R} = \begin{bmatrix} \alpha_0 & 0 & 0 & \cdots & \cdots & 0 \\ (1-\alpha_1) & \alpha_1 & 0 & \cdots & \cdots & 0 \\ 0 & (1-\alpha_2) & \alpha_2 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & (1-\alpha_{n-1}) & \alpha_{n-1} \\ 0 & 0 & 0 & 0 & \cdots & (1-\alpha_n) \end{bmatrix}. \tag{2.27}$$

Given matrix form is generalization of Oslo algorithm linear system.
When inserting knot who has multiplicity k and needs to be inserted r times ($k+r \leq p$, because it's not practical to have knots of multiplicity greater than p), then new control points $\widehat{\mathbf{Q}}_{i,r}$ are given as [6]

$$\widehat{\mathbf{Q}}_{i,r} = \alpha_{i,r}\widehat{\mathbf{Q}}_{i,r-1} + (1-\alpha_{i,r})\widehat{\mathbf{Q}}_{i-1,r-1}, \tag{2.28}$$

where

$$\alpha_{i,r} = \begin{cases} 1, & i \leq j-p+r-1, \\ \frac{\overline{u}_i - u_i}{\overline{u}_{i+p-r+1} - \overline{u}_i}, & j-p+r \leq i \leq j-k, \\ 0, & i \geq j-k+1. \end{cases} \tag{2.29}$$

Eq. (2.29-2.28) actually represents Eq. (2.25-2.26) repeated in a loop r times. Knot refinement is algorithm when multiple knots have to be inserted at the same time. Knot refinement can be solved using Oslo algorithm [18] where linear system has to be solved or inserting one knot

at a time following the procedure above. Basically, knot refinement is generalization of knot insertion algorithm. Regarding the NURBS surface, best way is to insert new knots is one knot at a time per parametric axes.

2.2.2. Knot Removal Algorithm

Knot removal is the reverse algorithm of knot insertion algorithm. First major use of knot removal was as a reduction of parameters [23, 24], but slightly later it was developed further as a tool for many applications, such as: degree reduction, converting the B-Spline segments to Bèzier patches and vice versa, etc. [25, 6].
Idea is to represent a given NURBS curve with new knot vector $\widehat{\mathbf{u}}$ which is obtained when the knot $\overline{u}_r$ (r is index of the knot in knot vector) of multiplicity k is removed t times ($1 \leq t \leq k$).

$$\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}\mathbf{Q}_i = \sum_{i=0}^{n-t} \widehat{N}_{i,p}\widehat{\mathbf{Q}}_i. \tag{2.30}$$

For the start, knot removal algorithm must determine is the knot $\overline{u}_r$ removable and if it is, how many times. From that, algorithm must compute new control points $\widehat{\mathbf{Q}}_i$ [6]. If a knot can be removed t times, curve must be C^{p-t} continuous. Basically, if a knot $\overline{u}_r$ is to be removed t times, equations are set like the knot $\overline{u}_r$ is to be inserted t times (knot insertion algorithm for knot $\overline{u}_r$, t times). Obtained set of equations is then solved backwards, starting from the last to the first. For the $n = p-k+1$, where the k is multiplicity of the knot to be removed, knot removal produces n equations in $n-1$ unknowns and destroys n control points, replacing them with $n-1$ new control points [25].
General algorithm for removing the knot $\overline{u}_r$ t times, where $1 \leq t \leq k$ [25].

Algorithm 2.2 Knot removal of knot $\overline{u}_r$ t times

Initialize:
$first = r-p+1$; $last = r-k-1$;
for $iter = 1, \ldots, t$ **do**
$first = first - 1$; $last = last + 1$;
$i = first$; $j = last$
while $j - i \geq iter - 1$ **do**

$$\widehat{\mathbf{Q}}_i^{iter} = \frac{\widehat{\mathbf{Q}}_i^{iter-1} - (1-\alpha_i)\widehat{\mathbf{Q}}_{i-1}^{iter}}{\alpha_i},$$

$$\widehat{\mathbf{Q}}_j^{iter} = \frac{\widehat{\mathbf{Q}}_j^{iter-1} - (1-\alpha_j)\widehat{\mathbf{Q}}_{j+1}^{iter}}{1-\alpha_j},$$

where $\alpha_i = \frac{\overline{u}_t - \overline{u}_i}{\overline{u}_{i+p+iter} - \overline{u}_i}$; $\alpha_j = \frac{\overline{u}_t - \overline{u}_{j-iter+1}}{\overline{u}_{j+p+1} - \overline{u}_{j-iter+1}}$;

$i = i+1$; $j = j-1$;
end while
end for

Given algorithm decrements multiplicity k and knot position index r with each iteration

(each iteration removes knot one time), and accounts the gap in knot vector and control points sequence due to knot removal.

2.2.3. Degree Elevation Algorithm

Basic steps of degree elevation algorithm are decomposition of NURBS curve into Bèzier segments, degree elevation of Bèzier segments and knot removal of unnecessary knots to form NURBS representation of the curve. When we have multiple curves which are used to construct one surface or NURBS curve, then some of the curves require degree elevation so that all curves have common degree [6]. When elevating the degree of NURBS curve $\mathbf{C}_p(u)$ from p to $p+1$, there must exist control points $\widehat{\mathbf{Q}}_i$ and knot vector $\widehat{\mathbf{u}}$ such that [6]

$$\begin{aligned} \mathbf{C}_p(u) &= \mathbf{C}_{p+1}(u), \\ \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i &= \sum_{i=0}^{\widehat{n}} N_{i,p+1}(u)\widehat{\mathbf{Q}}_i. \end{aligned} \tag{2.31}$$

In Eq. (2.31) there are three unknowns, $\widehat{n}$, $\widehat{\mathbf{u}}$ and $\{\widehat{\mathbf{Q}}_i\}$. To determine $\widehat{n}$ and $\widehat{\mathbf{u}}$, it's assumed that knot vector has the form

$$\overline{\mathbf{u}} = \{\overline{u}_0, ..., \overline{u}_m\} = \{\underbrace{0, ..., 0}_{p+1}, \underbrace{\overline{u}_1, ..., \overline{u}_1}_{k_1}, ..., \underbrace{\overline{u}_s, ..., \overline{u}_s}_{k_s}, \underbrace{1, ..., 1}\},$$

where $k_1, ..., k_s$ denotes multiplicity of interior knots (s is the number of interior knots). At a knot $\overline{u}_i$ of multiplicity k curve is $p - k_i$ continuous, which states that elevated curve $\mathbf{C}_{p+1}(u)$ must have same continuity. From there, same knot must have $k_i + 1$ continuity, which yields

$$\widehat{n} = n + s + 1, \tag{2.32a}$$

$$\hat{\mathbf{u}} = \{\overline{u}_0, ..., \overline{u}_{\hat{m}}\} = \{\underbrace{0, ..., 0}_{p+2}, \underbrace{\overline{u}_1, ..., \overline{u}_1}_{k_1+1}, ..., \underbrace{\overline{u}_s, ..., \overline{u}_s}_{k_s+1}, \underbrace{1, ..., 1}_{p+2}\}, \tag{2.32b}$$

where the number of knots is now $\hat{m} = m + s + 2$. With $\widehat{n}$ and $\hat{\mathbf{u}}$ defined, control points $\{\hat{\mathbf{Q}}_i\}$ is obtained from system of linear equations

$$\sum_{i=0}^{n} N_{i,p}\mathbf{Q}_i = \sum_{i=0}^{\widehat{n}} N_{i,p+1}\hat{\mathbf{Q}}_i. \tag{2.33}$$

Evaluating $N_{i,p}(u)$ and $N_{i,p+1}(u)$ at suitable $\widehat{n}+1$ values yields system of $\widehat{n}+1$ linear equations where unknowns are $\{\widehat{\mathbf{Q}}_i\}$. This method presents basic procedure to elevate degree, but somewhat inefficient. Prautzsch and Piper [26, 27] presented complex but more efficient method where new knots are inserted (based on Eq. 2.32b) and in combination with knot insertion algorithm, new control points are obtained. So far, given algorithms raise the degree by one, $p+1$.

In paper [28], Piegl and Tiller developed simpler algorithm in which degree can be raised by any degree ($p+r$, $r \geq 1$).

2.2.4. Degree Reduction Algorithm

Main application of degree reduction is to reverse degree elevation algorithm [6]. Let there be defined NURBS curve $\mathbf{C}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i$ on a knot vector

$$\overline{\mathbf{u}} = \{\underbrace{0,...,0}_{p+1}, \underbrace{\overline{u}_1,...,\overline{u}_1}_{k_1}, ..., \underbrace{\overline{u}_s,...,\overline{u}_s}_{k_s}, \underbrace{1,...,1}_{p+1}\}.$$

Curve $\mathbf{C}(u)$ is degree reducible if it can be presented in a form

$$\begin{aligned} \mathbf{C}_p(u) &= \mathbf{C}_{p-1}(u), \\ \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i &= \sum_{i=0}^{\widehat{n}} N_{i,p-1}\hat{\mathbf{Q}}_i, \end{aligned} \tag{2.34}$$

on the knot vector

$$\widehat{\mathbf{u}} = \{\underbrace{0,...,0}_{p}, \underbrace{\overline{u}_1,...,\overline{u}_1}_{k_1-1}, ..., \underbrace{\overline{u}_s,...,\overline{u}_s}_{k_s-1}, \underbrace{1,...,1}_{p}\}. \tag{2.35}$$

with $\widehat{n} = n-s-1$. It is evident that k_i can be $k_i = 1$, which implies that knot $\overline{u}_i$ is not in vector $\widehat{\mathbf{u}}$ and that knot $\overline{u}_i$ was removed from $\mathbf{C}(u)$. Elevation of a curve degree always possible, but curve may not be a degree reducible. Just like at knot removal, problem can be over-determined, i.e. it can produce more equations than unknowns $\widehat{\mathbf{Q}}_i$. Because of floating point error, obtained curve $\hat{\mathbf{C}}(u)$ may not always coincide with $\mathbf{C}(u)$, and therefore some error must be accounted [6]

$$E(u) = \|\mathbf{C}(u) - \widehat{\mathbf{C}}(u)\| \leq TOL. \tag{2.36}$$

Most notable algorithm is defined by Piegl and Tiller [29], which is extension of paper [28] by the same authors with distinction of degree reduction. Briefly, NURBS curve is decomposed into Bèzier segments, then for each segment degree is reduced and unnecessary knots are removed to form NURBS representation. After decomposition in Bèzier segments, knot removal algorithm is applied to reduce the degree of each segment and finally to merge segments into NURBS representation. Depending on degree parity, approximation errors terms are derived in both steps of knot removal, first for Bèzier segments degree reduction and second for removal of excess knots to unify segments for NURBS representation. Later, Wolters et al. [30] developed an algorithm that works with NURBS directly. Curve is represented in blossom form (based on de Boor algorithm), from which equations are obtained and solved using least squares approach. Finally, with weighting scheme unknown coefficients are obtained. Following the optimization scheme, Yong at al. [31] also reduced the problem to constrained optimization. Using the continuity property of NURBS curve (Sec. 2.1.3, property $P_B 7.$), NURBS derivatives were evaluated

using B divided differences and where unknowns $\widehat{\mathbf{Q}}_i$ are obtained through constrained optimization.

2.3. Advanced Parametric Curves and Surfaces

In this chapter curves and surfaces defined with multiple level domains or locally defined domains will be discussed. Their common thing is that they are defined by two or more tensor products of basis functions in domains.

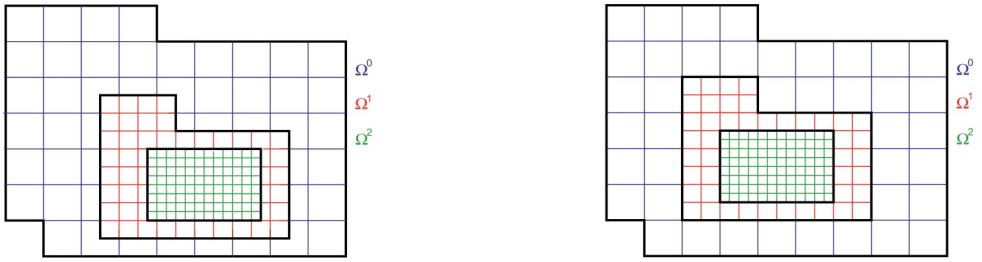
2.3.1. Hierarchical B-Spline

So far discussed, NURBS theory together with B-Spline and Bèzier provides sufficient degrees of freedom regarding parametric modeling. But in certain cases, additional finer control of some regions is required. This is achieved with knot refinement, which gives more local control to desired areas. B-Spline (or NURBS) refinement was first introduced by Forsey and Bartels [32] where authors presented adaptive refinement procedure for one or more basis functions in initial domain. Using only knot refinement would result with new global knot vector, but the idea was to keep the refinement in desired areas. To keep changes only in desired areas, authors introduced hierarchical overlays where every new refined area was considered as new hierarchical domain, keeping the changes strictly localized. Later, with adaptive hierarchical refinement, Kraft [33, 34] defined multilevel hierarchical spline space as linear span of tensor product B-Splines defined on different grids of knots mesh.
Let $\{\mathcal{V}^l\}_{l=0,\dots,N-1}$ be a sequence of N nested tensor-product spline spaces so that $\mathcal{V}^l \subset \mathcal{V}^{l+1}$. The index l of a space $\mathcal{V}^l$ is defined as its *level* in hierarchy and N is depth of hierarchy. Each spline space $\mathcal{V}^l$ is spanned by tensor product of B-Spline basis $\mathcal{B}^l$, defined on one (univariate B-Spline) knot sequence or on two (bivariate B-Spline) knot sequences $\{\overline{\mathbf{u}}_i^l\}_{i=0,\dots,p(l)}$ and $\{\overline{\mathbf{v}}_i^l\}_{i=0,\dots,q(l)}$, where p and q denotes polynomial degree per each parametric axis. As a result, every space $\mathcal{V}^l$ has its corresponding knot grid [35, 36]

$$\mathcal{G}^l = \{(\overline{u}_{i-1}^l, \overline{u}_i^l) \times (\overline{v}_{i-1}^l, \overline{v}_i^l) : i = 1,\dots,p(l), j = 1,\dots,q(l)\}.$$

New level spline space $\mathcal{V}^{l+1}$ grid is obtained with iterative refinement of knots (frequently dyadic knot insertion algorithm, Sec. 2.2.1), where the old knot grid is kept and refined with new knots in the middle of old knots as $\overline{u}^j = (\overline{u}_i^j + \overline{u}_{i+1}^j)/2$. Refinement over new regions can be done in several directions as [22]:

- Rectangular (over rectangular regions)
- Linear (along diagonal layer)
- Curvilinear (along curvilinear trajectory)



(a) Weak condition.

(b) Strong condition.

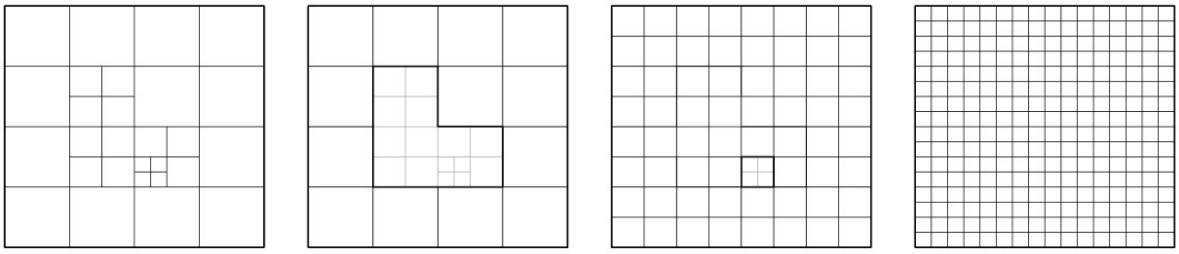
Figure 2.10: Strong and weak condition for bivariate subdomains boundaries [37].

.

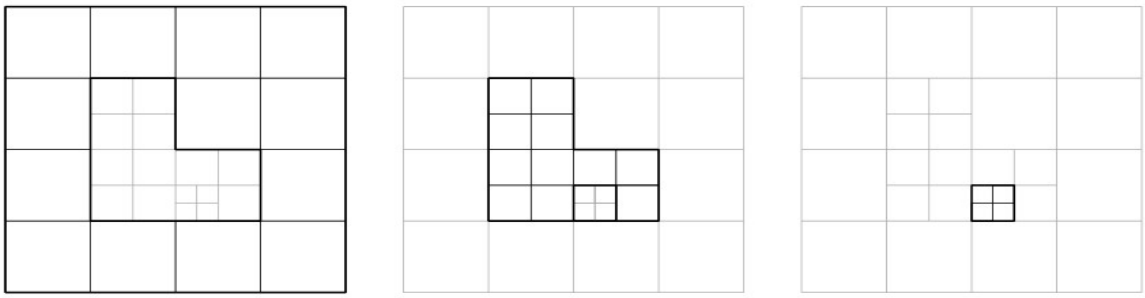
Additionally, we consider a sequence of nested subdomains $\{\Omega^l\}_{l=0,\dots,N-1}$ such that $\Omega^l \subseteq \Omega^{l+1}$ for $l = 1,\dots,N-1$. Each Ω^l represents region to be refined at level l and its boundary $\partial\Omega^l$ can be in line with knot grid of spline space $\mathcal{V}^{l-1}$ (strong condition) or in line with knots of $\mathcal{V}^l$ (weak condition), which can be seen on Fig. 2.10. [37]. Support of any function f when restricted to domain Ω^0 is

$$\text{supp } f = \{(u,v) : f(u,v) \neq 0 \wedge (u,v) \in \Omega^0\}.$$

The term *ring*, related to domain $R^l = \Omega^0 \backslash \Omega^{l+1}$ was introduced by Giannelli and Jüttler [36]. Ring R^l conceptually represents Ω^0 with a hole given by Ω^{l+1}. R^l consists of cells given by tensor product grid of level l and its called *multi-cell domain*. Regarding the difference between two successive subdomains $\Omega^l \backslash \Omega^{l+1}$, new sequence of multi-cell domains is defined. Union of these domains is called *multi-grid multi-cell domain* (Fig. 2.11).



(a) From left to right: hierarchical mesh, $R^0 = \Omega^0 \backslash \Omega^1$, $R^1 = \Omega^0 \backslash \Omega^2$, $R^2 = \Omega^0 \backslash \Omega^3$.



(b) From left to right: $\Omega^0 \backslash \Omega^1$, $\Omega^1 \backslash \Omega^2$, $\Omega^2 \backslash \Omega^3$

Figure 2.11: Hierarchical mesh defined by restricting the grid $\mathcal{V}^l$ to Ω^l, for $l = 0,\dots,3$, with rings R^l (a) and differences in $\Omega^l \backslash \Omega^{l+1}$ (b) [36].

With given spline spaces $\mathcal{V}^l$ and B-Spline bases $\mathcal{B}^l$ in nested subdomains Ω^l with defined rings R^l, the construction of hierarchical basis (HB) is given in following definition [35]

Definition 1. *For a given subdomain hierarchy* $\{\Omega^l\}_{l=0,\dots,N-1}$*, hierarchical spline basis* $\mathcal{H}$ *is recursively constructed as*

(I) Initialization:

$$\mathcal{H}^0 = \left\{\beta \in \mathcal{B}^0 : supp\ \beta \neq 0\right\},$$

(II) Recursive construction of $\mathcal{H}^{l+1}$:

$$\mathcal{H}^{l+1} = \mathcal{H}_A^{l+1} \cup \mathcal{H}_B^{l+1}, \quad for\ l = 0,\dots,N-2,$$

where

$$\mathcal{H}_A^{l+1} = \left\{\beta \in \mathcal{H}^l : supp\ \beta \not\subseteq \Omega^{l+1}\right\},$$
$$\mathcal{H}_B^{l+1} = \left\{\beta \in \mathcal{B}^{l+1} : supp\ \beta \subseteq \Omega^{l+1}\right\}.$$

(III) Result:

$$\mathcal{H} = \mathcal{H}^{N-1}.$$

Finally, $\mathcal{S} = span\,\mathcal{H}$ *is multilevel spline space defined by subdomain hierarchy* $\{\Omega^l\}_{l=0,\dots,N-1}$.

For initial hierarchy domain Ω^0, all basis functions in $\mathcal{B}^0$ are selected. Spline hierarchy is completed by adding basis functions supported in current hierarchy Ω^l ($\mathcal{H}_A^{l+1}$ in Def. 1) and basis function contained in successive hierarchy Ω^{l+1} ($\mathcal{H}_B^{l+1}$ in Def. 1), which is done recursively for all levels $l = 0,\dots,N-2$. Basis functions β contained in some arbitrary level $\mathcal{B}^l$ which are present in current $\mathcal{H}^l$ are called *active* functions, while functions not considered in current level are called *passive* functions. HB hold most of the B-Spline basis properties, such as nonegativity and local support [33]. Key properties, from Def. 1 are (see on Fig. 2.12 for curve and Fig. 2.13 for a surface): [35]

(H1) Linear independence: $\sum_{\beta\in\mathcal{H}} d_\beta\beta = 0 \Leftrightarrow d_\beta = 0, \forall\beta \in \mathcal{H}$.

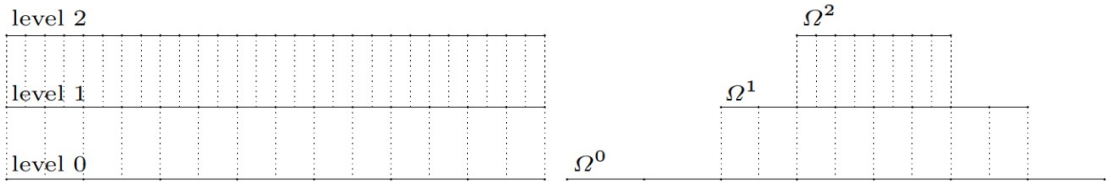
(H2) Non-negativity: $\forall\beta \in \mathcal{H}, \beta \geq 0$.

(H3) Nested nature of the spline spaces: $span\,\mathcal{H}^l \subseteq \mathcal{H}^{l+1}$ for $l = 0,\dots,N-2$.

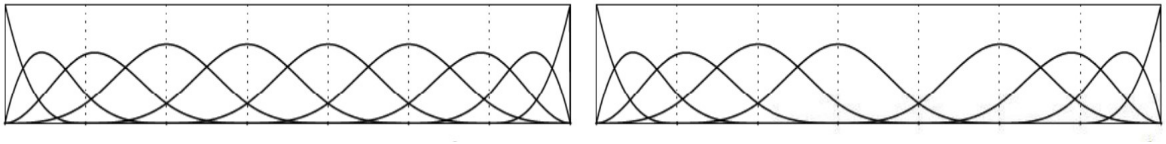
Non-negativity property comes directly from definition of B-Spline basis functions (Sec. 2.1.3), while linear independence and nested nature property are more detailed in [33, 34] and [37], respectively. Partition of unity (PoU) property is not inherited in HB. Simple way to recover PoU property is through *weighted basis* $\mathcal{W}$, discussed in [37]. For every basis function β in HB space $\mathcal{H}$ there is a related weighted basis function $\omega = w_\beta\beta$ from which normalized HB is obtained as

$$\mathcal{W} = \left\{\omega = w_\beta\beta : \beta \in \mathcal{H} \wedge \sum_{\beta\in\mathcal{H}} w_\beta\beta = 1\right\}.$$

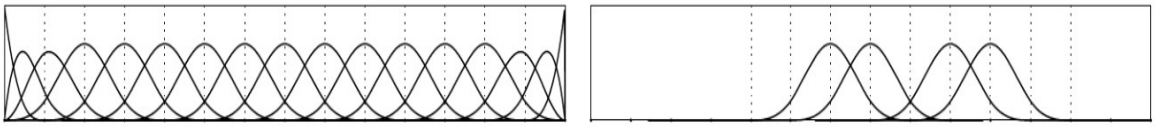
which forms a PoU as $\sum_{\omega \in \mathcal{W}} \omega = 1$. For some hierarchical configurations, some weights in refined levels can be zero and thus cancel some contributions. To avoid this, truncated HB can be used which will be explained in next section.



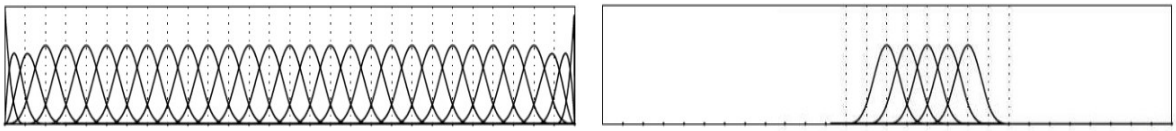
(a) Knot vectors (left) and subdomain hierarchy (right).



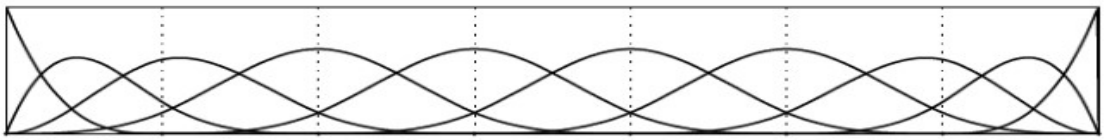
(b) Level 0: B-Spline basis of $\mathcal{B}^0$ (left) and hierarchical B-Splines $\mathcal{H} \cap \mathcal{B}^0$ (right).



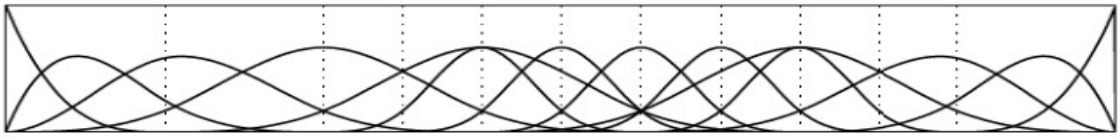
(c) Level 1: B-Spline basis of $\mathcal{B}^1$ (left) and hierarchical B-Splines $\mathcal{H} \cap \mathcal{B}^1$ (right).



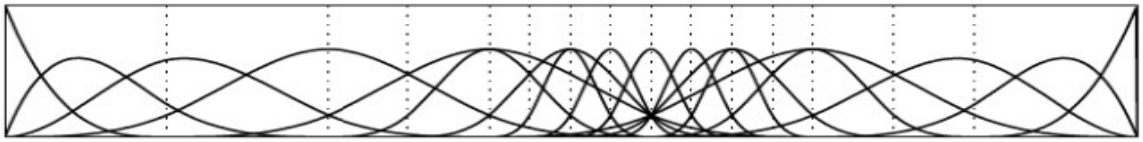
(d) Level 2: B-Spline basis of $\mathcal{B}^2$ (left) and hierarchical B-Splines $\mathcal{H} \cap \mathcal{B}^2$ (right).



(e) HB basis space $\mathcal{H}^0$.



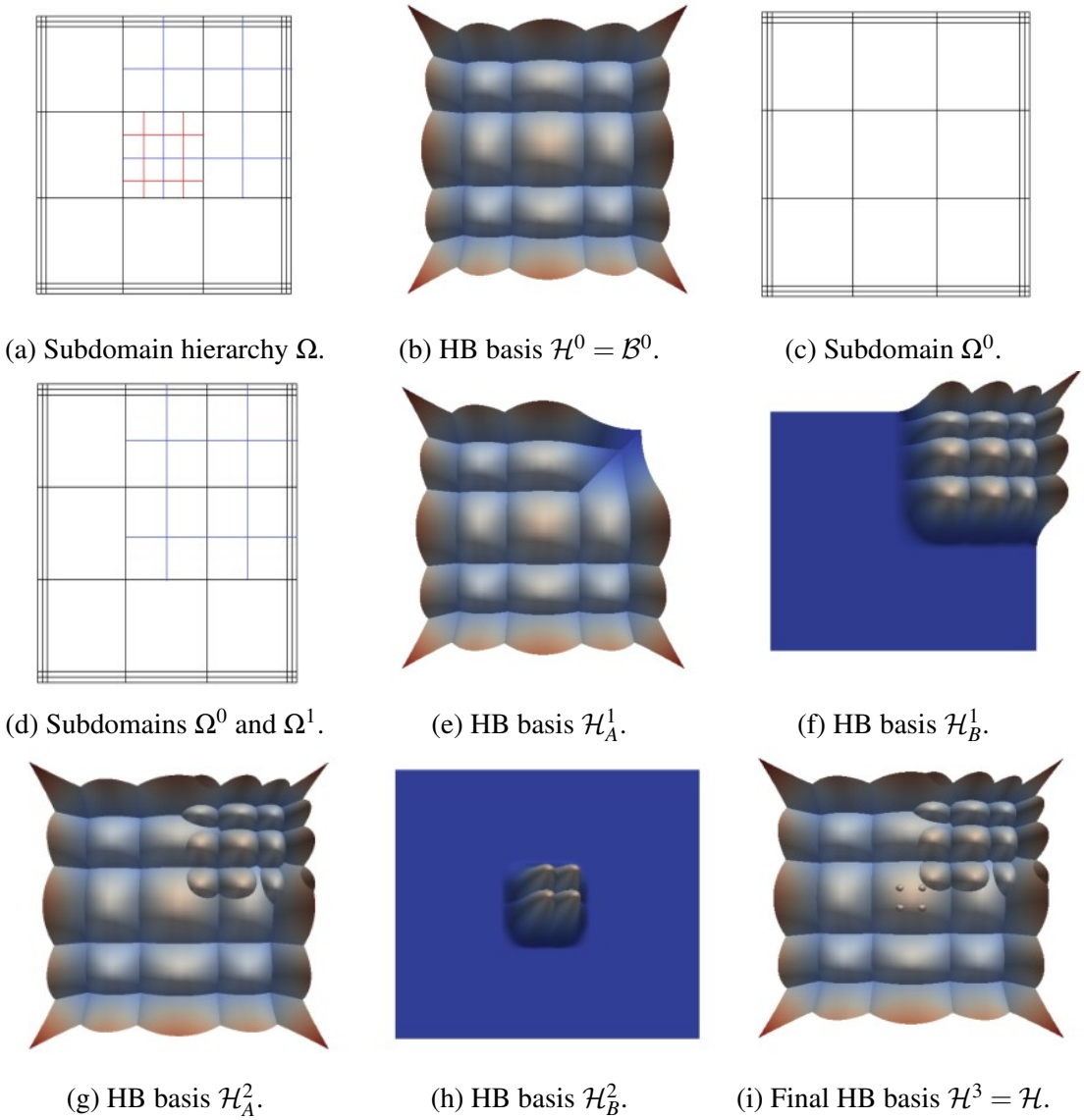
(f) HB basis space $\mathcal{H}^1$.



(g) HB basis space $\mathcal{H}^2 = \mathcal{H}$ from Fig. 2.12a subdomain hierarchy.

Figure 2.12: Univariate hierarchical basis of $p = 3$, with three levels of hierarchy [35]. Figures (b-d) shows B-Spline basis for each level of knots and its corresponding functions contained in subdomain hierarchy (a). Figures (e-g) represent HB bases $\mathcal{H}^l$ with active functions defined recursively per each level.

Univariate HB of three levels, with every basis space $\mathcal{B}^l$ and nestedness of subdomain hierarchies Ω^l and HB spaces $\mathcal{H}^l$ can be seen on Fig. 2.12 for 2D space and on Fig. 2.13 for 3D space, together with resulting HB space $\mathcal{H}^2 = \mathcal{H}$ based on subdomain hierarchy.



(a) Subdomain hierarchy Ω. (b) HB basis $\mathcal{H}^0 = \mathcal{B}^0$. (c) Subdomain Ω^0.

(d) Subdomains Ω^0 and Ω^1. (e) HB basis $\mathcal{H}^1_A$. (f) HB basis $\mathcal{H}^1_B$.

(g) HB basis $\mathcal{H}^2_A$. (h) HB basis $\mathcal{H}^2_B$. (i) Final HB basis $\mathcal{H}^3 = \mathcal{H}$.

Figure 2.13: Bivariate HB space with three levels of refinement, with symbolds from Def. 1 [22]. Subdomain hierarchy (a) is shown throughout HB for every hierarchy level ($\mathcal{H}^l_A \cup \mathcal{H}^l_B$), starting from initial domain and bases (b-c), through recursive definition for both levels of refinement (d-h) until final HB space (i).

With given construction of HB space, HB-Spline surface (curve) can be defined as [21, 22]

$$\mathbf{S}(u,v) = \sum_{l=0}^{N-1} \sum_{i \in \mathcal{A}^l} \beta^l_i(u,v)\mathbf{Q}^l_i, \tag{2.37}$$

where β^l_i are B-Spline basis functions and $\mathcal{A}^l$ is set of active functions per level l.
With further analysis of HB-Spline space, Giannelli and Jüttler [36] analyzed dimensions of

HB space to achieve maximum order of smoothness on certain grids, based on strong condition. Thus, admissible domain configuration was proposed. Similarly, Giannelli at al. [35] introduced weak and strong stability of hierarchical basis, based on weak and strong condition, where local refinement can be achieved in a more effective way. Completeness of HB tensor product was explored by Mokriš et al. [38], where the condition for completeness of hierarchical spline space was introduced. Given condition guarantees that any piecewise polynomial basis function can be defined in hierarchical tensor product B-Spline basis. This paper was extension (or complement) of work in [36]. In order to construct hierarchical data structure, binary tree is used for 1D problems, quad-tree and octree for two or three dimensions, respectively. For construction of 3D surface defined in 2D adaptive mesh domain, mostly used is concept of quad-tree from [39]. Every node or leaf holds information about corresponding knot span and basis functions in respective hierarchical level. Additionally, each node or leaf contains pointers to neighboring nodes in considered level which makes it easier to establish relations between neighboring basis functions (Fig. 2.14).

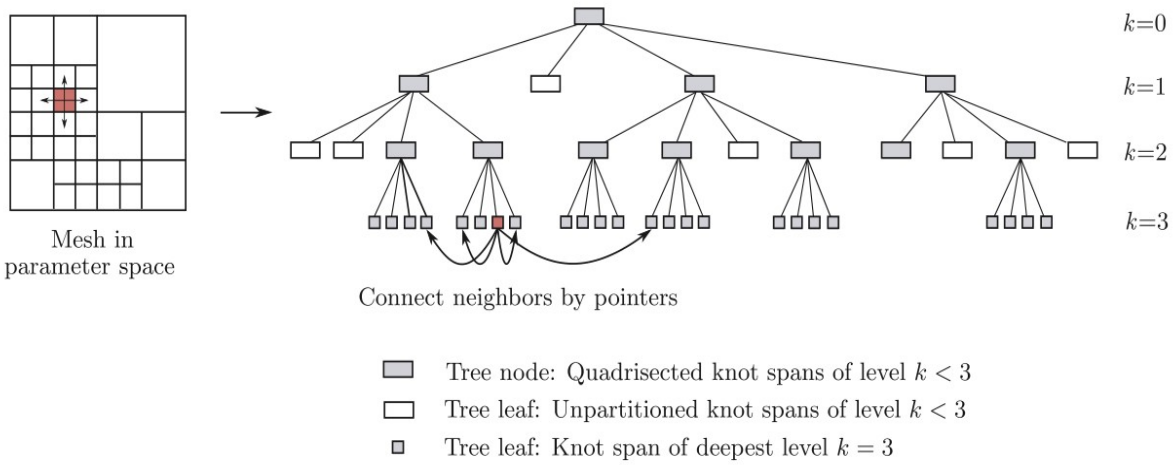


Figure 2.14: Quad-tree example for given hierarchical subdomain hierarchy [39]. Pointers are directing to neighboring partitions of desired element (red element) on the same level but on different leaves of quad-tree.

2.3.2. Truncated Hierarchical B-Spline

Adaptive local refinement can be greatly achieved with HB but they lack one of major properties, which is PoU. This can be realized with use of normalized basis for refined spline spaces. HB-Spline construction can be altered with suitable *truncation* of basis functions in finer levels of hierarchy. Main idea is to eliminate contributions of basis functions which are included in finer level. This mechanism preserves PoU and defines smaller support of basis functions. Due to truncation mechanism, this type of HB is named *truncated hierarchical basis* (THB) from which THB-Splines are obtained. First major introduction of THB was by Giannelli at al. [40] where truncation mechanism on HB was applied and thus spline space which contains all HB

properties along preservation of PoU was obtained. Also, THB introduced several more advantages over classical HB. Similar approach has been proposed by Speleers at al. [41], where authors defined normalized hierarchical basis on Powell-Sabin triangulations. Also, due to increasing popularity of isogeometric analysis (IGA) [42] which initially used standard NURBS models, authors started implementing adaptively refined models in IGA. Most notably, Vuong et al. [37] discussed HB application in IGA. This application resulted in more research interest for adaptive models where further developments (e.g. THB from HB) were achieved.
Like already mentioned, THB model is extension of HB model. With already defined spline space $\mathcal{V}^l$ and corresponding B-Spline basis functions space $\mathcal{B}^l$ in subdomain hierarchy Ω^l (Sec. 2.3.1) hierarchical basis are obtained with Def. 1, where authors in [40] progressed to THB introduction which mostly relied on following definition [43]

Definition 2. *Any B-Spline basis function $\beta \in \mathcal{B}^l$ in level l can be expressed as linear combination of basis functions in finer level $l+1$ as*

$$\tau = \sum_{i=0}^{p+1} c_i^{l+1} \beta_i^{l+1}, \tag{2.38}$$

where τ is B-Spline basis function denoted in THB finer basis $\mathcal{V}^{l+1}$ and c_i^{l+1} are called subdivision (or refinement) coefficients [44, 45, 46], which can be obtained by knot insertion algorithm (de Boor algorithm) or as binomial coefficients with equation below

$$c_i^{l+1} = \frac{1}{2^p} \binom{p+1}{i}. \tag{2.39}$$

This is called refinement of B-Spline basis functions. With dyadic refinement, any B-Spline basis function is refined into $p+2$ finer basis functions (Fig. 2.15). Def. 2 and upper equations corresponds to dyadic refinement, although refinement can be achieved for arbitrary number of new knots via knot insertion. It should be noted that Eq. 2.39 is binomial form of knot insertion when every knot interval in knot vector is bisected. Some authors have formulated refinement (Eq. 2.38) in matrix form as [43]

$$\mathbf{T}^l = \mathbf{C}^{l+1} \mathbf{B}^{l+1},$$

where $\mathbf{C}^{l+1}$ is subdivision matrix.

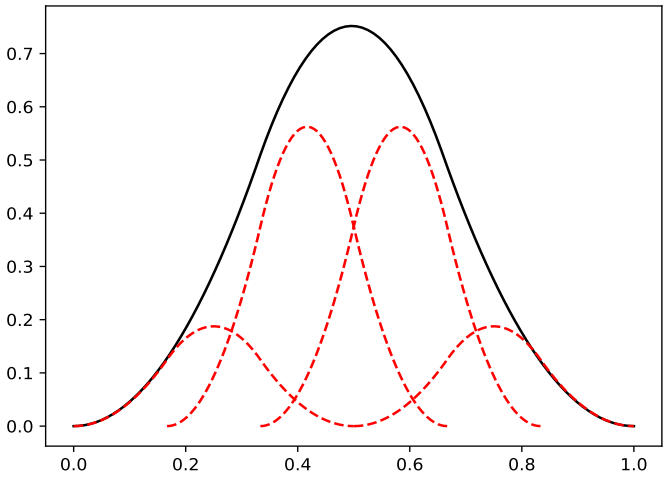


Figure 2.15: Refinement of quadratic B-Spline basis function, according to Def. 2 and Eq. 2.38. B-Spline basis function τ defined on knot vector $\overline{\mathbf{u}} = \{0, 0.333, 0.667, 1\}$ is shown in black solid line. With dyadic refinement, new knot vector is obtained $\overline{\mathbf{u}}_{ref} = \{0, 0.1667, 0.333, 0.5, 0.667, 0.833, 1\}$ and thus new $p+2$ refined basis function β_{ref} (shown in red dashed lines).

Truncation of τ with respect to $\mathcal{B}^{l+1}$ and Ω^{l+1} is defined as [40] (Fig. 2.16)

$$trunc^{l+1}\tau = \sum_{\beta \in \mathcal{B}^{l+1},\, supp\,\beta \subseteq \Omega^{l+1}} c_{\beta}^{l+1}(\tau)\beta. \tag{2.40}$$

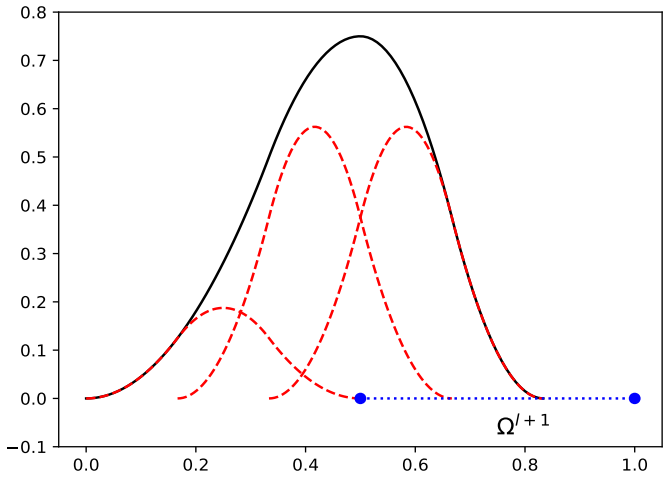


Figure 2.16: Truncation of B-Spline basis function shown on Fig. 2.15, according to Eq. 2.40. If a knot span $[0.5, 1)$ in $\overline{\mathbf{u}}_{ref}$ (Fig. 2.15) belongs to a new level subdomain hierarchy Ω^{l+1} (shown in blue dotted lines), then every refined function β^{l+1} (red dashed lines) with support in Ω^{l+1} is truncated from τ. Finally, black solid line represents $trunc\,\tau$, which is active basis function in Ω^{l}.

With truncation mechanism applied to HB-Splines, the THB-Splines can be introduced as [40, 22]

Definition 3. *The truncated hierarchical B-Spline basis $\mathcal{T}$ is recursively defined as:*

(I) Initialization:

$$\mathcal{T}^0 = \mathcal{H}^0,$$

(II) Recursive construction of $\mathcal{T}^{l+1}$:

$$\begin{aligned}
&\mathcal{T}^{l+1} = \mathcal{T}_A^{l+1} \cup \mathcal{T}_B^{l+1}, \quad \textit{for } l = 0,...,N-2, \\
&\textit{where} \\
&\mathcal{T}_A^{l+1} = \left\{ trunc^{l+1}\tau \in \mathcal{T}^l : supp\, \tau \nsubseteq \Omega^{l+1} \right\}, \\
&\mathcal{T}_B^{l+1} = \left\{ \beta \in \mathcal{B}^{l+1} : supp\, \beta \subseteq \Omega^{l+1} \right\} = \mathcal{H}_B^{l+1}.
\end{aligned}$$

(III) Result:

$$\mathcal{T} = \mathcal{T}^{N-1}.$$

Construction is exact to the construction of HB-Splines (Def. 1), with addition of truncation mechanism on basis functions β^l whose finer basis functions τ have support in Ω^l. Truncation leads to truncated basis $trunc^{l+1}\tau$, whose support is minimally contained in Ω^{l+1} and thus preserves the PoU property. For each truncated basis function τ at level l, there exists one B-Spline basis function β^l which satisfies

$$\tau = trunc^{N-1}(trunc^{N-2}...(trunc^{l+1}(\beta))...), \tag{2.41}$$

from which B-Spline basis β is denoted as *mother* of τ ($\beta = mot(\tau)$) and τ is child of β ($\tau = child(\beta)$). Because THB is derived from HB, it inherits some properties and acquires some new ones, such as (see Fig. 2.17 and Fig. 2.18 for curve and surface, respectively): [40, 35]

(T1) Linear independence: $\sum_{\tau\in\mathcal{T}} d_\tau\tau = 0 \Leftrightarrow c_\tau = 0,\ \forall\tau \in \mathcal{T}$.

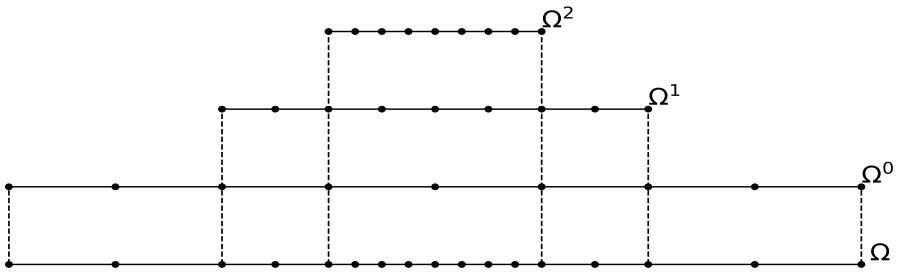
(T2) Non-negativity: $\forall\tau \in \mathcal{T},\ \tau \geq 0$.

(T3) Nested nature of spline spaces: $span\,\mathcal{T}^l \subseteq \mathcal{T}^{l+1}$ for $l = 0,...,N-2$.

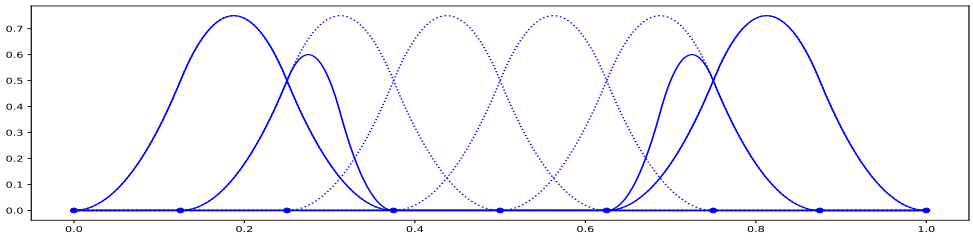
(T4) $span\,\mathcal{T}$ is multilevel spline space $\mathcal{S}$ (Def. 1), determined by subdomain hierarchy $\mathcal{S} = span\,\mathcal{H} = span\,\mathcal{T}$.

(T5) Truncated hierarchical B-Spline basis $\mathcal{T}$ forms PoU: $\sum_{\tau\in\mathcal{T}^l} \tau = 1$ for all $l = 0,...,N-1$.

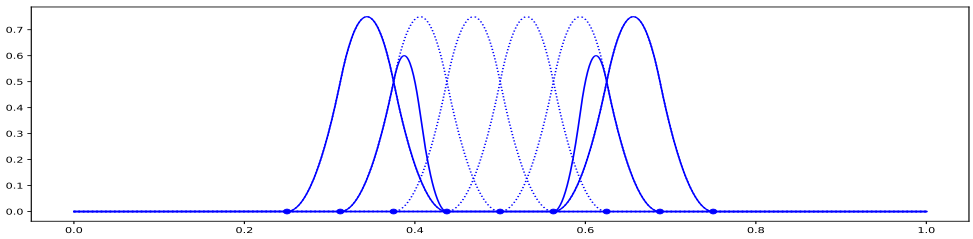
(T6) Truncated hierarchical basis $\mathcal{T}$ forms convex PoU on Ω^0.



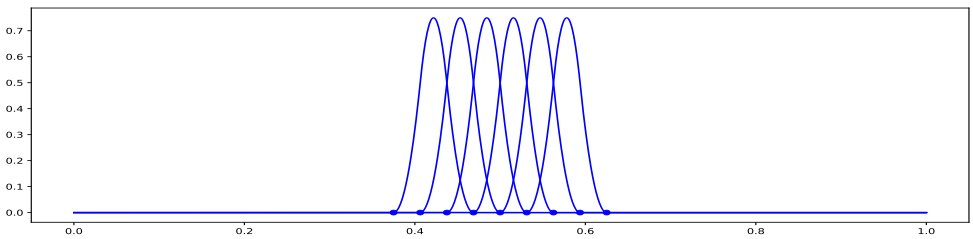
(a) Subdomain hierarchy with three nested levels (0,1 and 2).



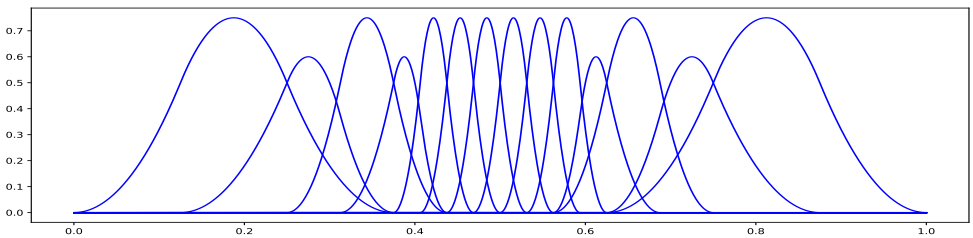
(b) Level 0: Truncated hierarchical basis $\mathcal{T} \cap \mathcal{B}^0$.



(c) Level 1: Truncated hierarchical basis $\mathcal{T} \cap \mathcal{B}^1$.

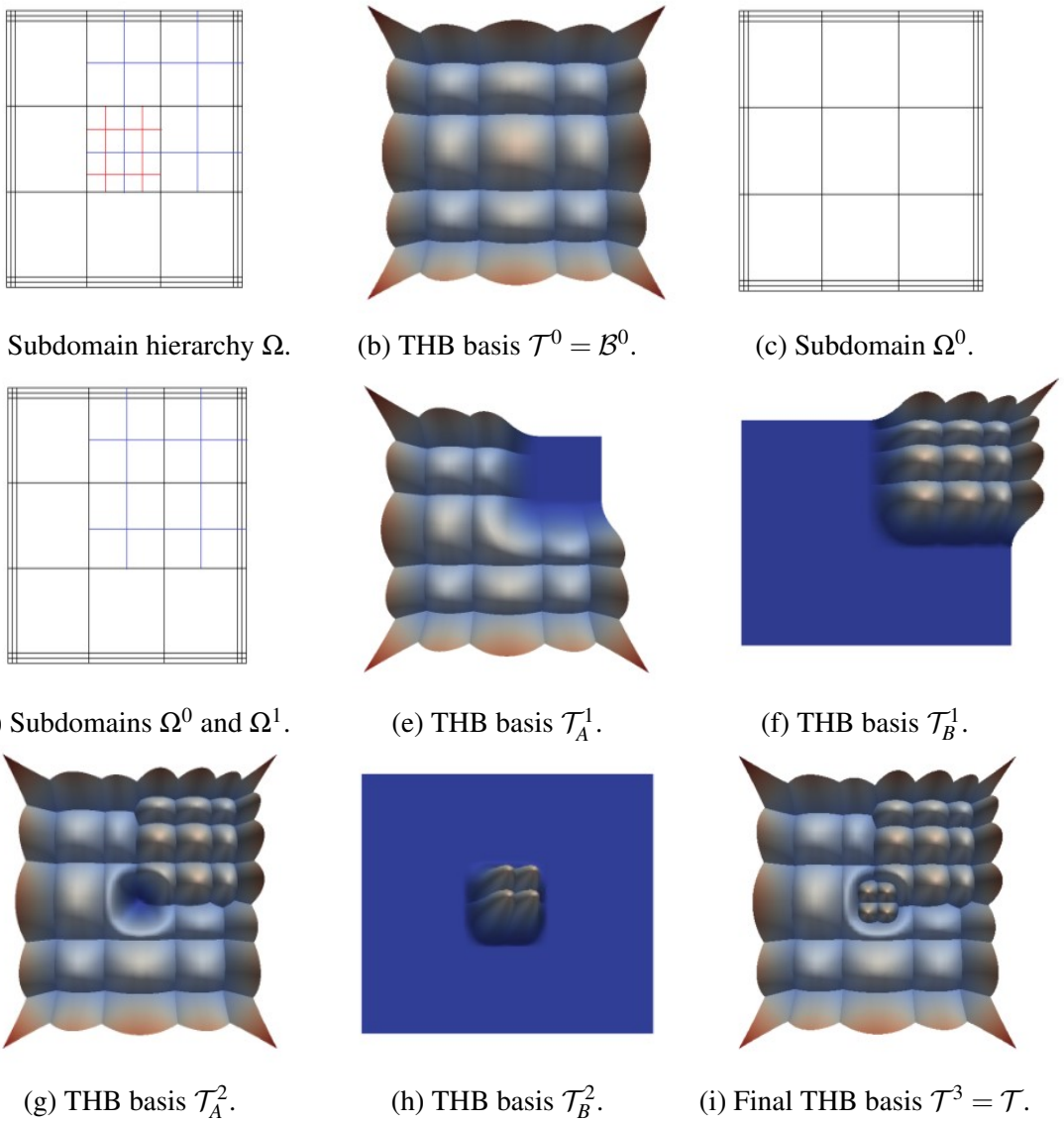


(d) Level 2: Truncated hierarchical basis $\mathcal{T} \cap \mathcal{B}^1$.



(e) Resulting truncated hierarchical basis $\mathcal{T} = \mathcal{T}^2$ from subdomain hierarchy (Fig. 2.17a).

Figure 2.17: Univariate truncated hierarchical basis of $p = 2$. For a given subdomain hierarchy (a), figures (b-d) show active truncated hierarchical basis functions per each level. From these active functions, final THB is constructed (e).



(a) Subdomain hierarchy Ω. (b) THB basis $\mathcal{T}^0 = \mathcal{B}^0$. (c) Subdomain Ω^0.

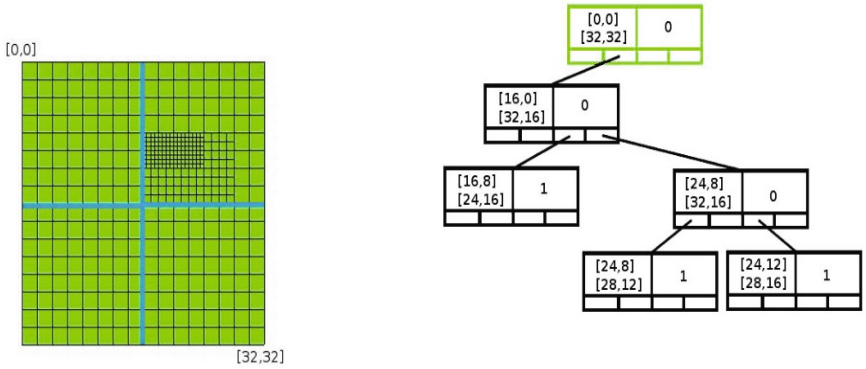
(d) Subdomains Ω^0 and Ω^1. (e) THB basis $\mathcal{T}_A^1$. (f) THB basis $\mathcal{T}_B^1$.

(g) THB basis $\mathcal{T}_A^2$. (h) THB basis $\mathcal{T}_B^2$. (i) Final THB basis $\mathcal{T}^3 = \mathcal{T}$.

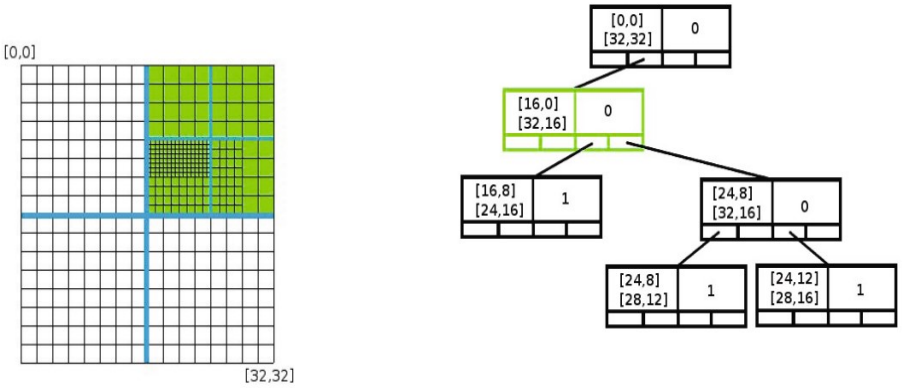
Figure 2.18: Bivariate THB space with three levels of refinement, with symbolds from Def. 3. Subdomain hierarchy (a) is shown throughout THB for every hierarchy level ($\mathcal{T}_A^l \cup \mathcal{T}_B^l$), starting from initial domain and bases (b-c), through recursive definition for both levels of refinement (d-h) until final THB space (i) [22].

Similarly like with HB-Spline, authors in [35] also accounted THB-Spline in bases stability analysis and shown that they are always strongly stable in comparison to HB-Spline. Buffa and Giannelli [47] introduced new term regarding the refinement of the grid. This so called *admissible mesh* defines the admissible class of arbitrary element (cell) and defines the bounding number of active basis functions per that element. It's connected to the truncation mechanism and defines that number of non-zero active functions on one element is not a function of number of the levels, but rather the function of admissible class if the refinement is done in admissible narrative. It's a way of gaining more structured refined grid and it can be quite useful in IGA when dealing with error estimators and convergence. With further analysis of mesh admissibility, Buffa et al. [48] have developed complexity estimation which ensures constant ratio

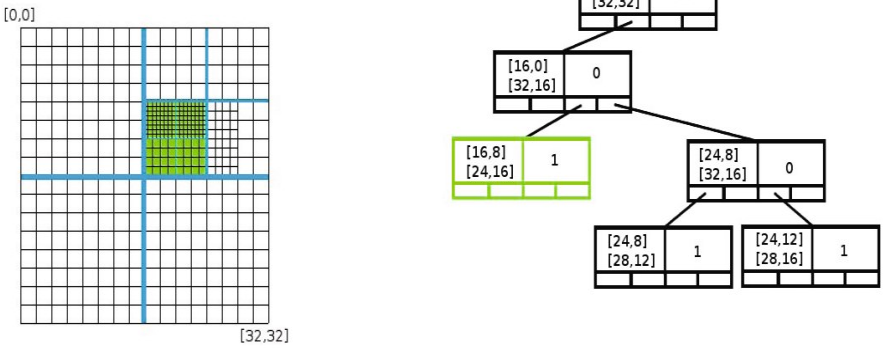
between old and refined elements, if the refinement is done in admissible way. Also, this estimation parameter can be used as a way to track (or control) mesh refinement in convergence of IGA. Kiss et al. [49, 22] developed implementation of THB-Spline algorithms for construction and evaluation of THB-Spline model, along with quad-tree data structure for storing data required to construct THB-Spline (Fig. 2.19). Quad-tree is similar to the one defined for HB-Spline (Sec. 2.3.1) but with additional functionalities and queries for THB-Spline definition. Also, *characteristic matrices* were introduced to collect information about active or passive B-Spline basis functions.



(a) First split (left) and its quad-tree representation (right).



(b) Second split (left) and its quad-tree representation (right).



(c) Third split (left) and its quad-tree representation (right).

Figure 2.19: Quad-tree representation of hierarchical domain [49].

Each node on Fig. 2.19 contains information about cells (or partition) coordinates (upper left data), corresponding level of the cell (upper right data) and pointers to the four children cells of current cell (four lower entries in node). Required algorithms and functionalities are also described in [49, 22] with the use of sparse data structures. Song et al. [50] defined kd-tree representation of hierarchical domain which does not split parametric domain in four parts like quad-tree, but rather in half (per u or v axis) and thus creates deeper and narrower tree representation than quad-tree (Fig. 2.20). Authors compared kd-tree subdivision to standard quad-tree and octree subdivision, which proved to be more efficient than quad-tree or octree and that generates less cells.

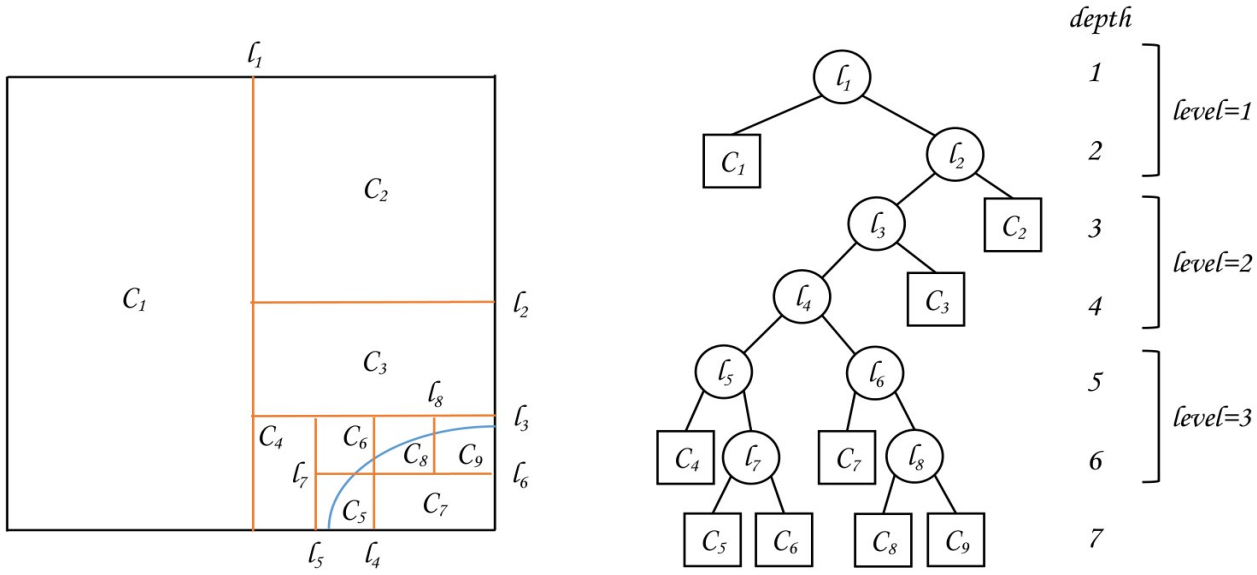


Figure 2.20: Kd-tree based subdivision, where each level consists of two depths that represent splitting directions l_i and cells C_i [50].

2.3.3. T-Splines

Tensor product NURBS or hierarchical spline (HB or THB) is constructed on one rectangular knot grid or on multiple rectangular knot grids, respectively. With local refinement, standard rectangular knot grids can get very dense, locally or globally. So in order to manipulate only desired rows and columns of knots or control points, T-junctions were introduced in by Sederberg et al. [51, 52] which allow creation of so called T-mesh from which T-Splines are obtained. From this perspective, T-mesh is actually rectangular knot grid with T-junctions (Fig.2.21) [51]. Each line in T-mesh is a line segment of constant u or constant v (in some literature it is denoted as s and t, i.e. (s,t) parametric domain). Each edge is labeled with a knot interval and constrained by two rules:

Rule 1. Sum of knot intervals on opposing edges of any face must be equal. E.g. for face **F** in Fig. 2.21 must be valid that $d_2 + d_6 = d_7$ and $e_6 + e_7 = e_8 + e_9$.

Rule 2. If a two T-junctions on opposite edges can be connected, thus splitting the face into two faces following the Rule 1, that edge must be included in the T-mesh.

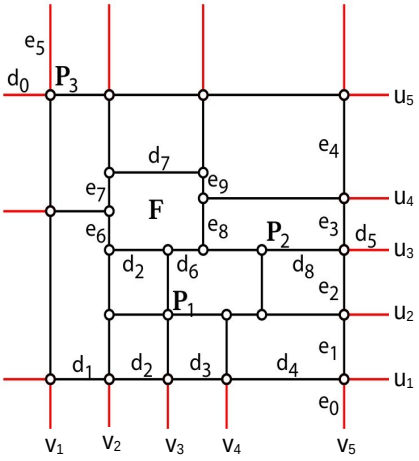


Figure 2.21: Example of T-mesh in (u,v) parametric domain with control points $\mathbf{P}_i$ [51].

To show how the T-spline basis functions are calculated, first the concept of *point based splines* (PB-Splines) must be introduced [51]. The control points in PB-Splines posses no topological relationship with each other.

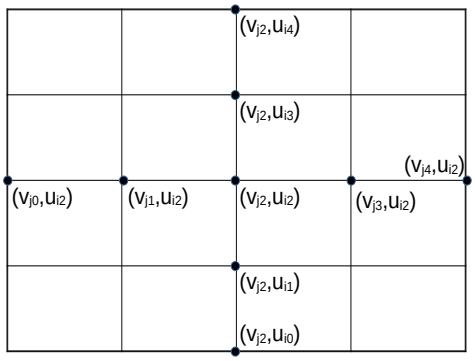


Figure 2.22: Knot lines for PB-Spline basis function $B_{i,j}(u,v)$ [51].

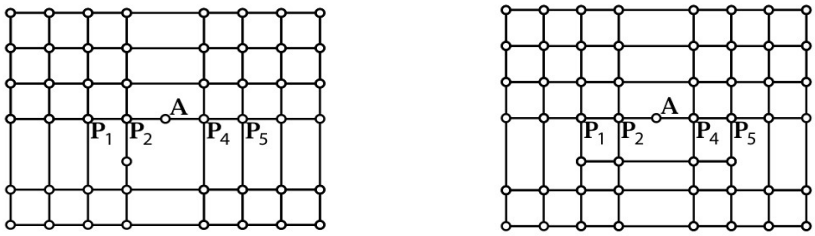
From Fig. 2.22, basis function $B_{i,j}(u,v)$ is given as tensor product of cubic univariate basis functions $B_{i,j}(u,v) = N_{i,3}(u)N_{j,3}(v)$. Every univariate basis function is associated with its knot vector, where $N_{i,3}(u)$ corresponding knot vector is $\overline{\mathbf{u}} = \{u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}\}$ and $N_{j,3}(v)$ knot vector is $\overline{\mathbf{v}} = \{v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$. This results with equation for a PB-Spline surface

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n-1} B_{i,j}(u,v)\mathbf{P}_i}{\sum_{i=0}^{n-1} B_{i,j}(u,v)}. \tag{2.42}$$

In order to specify a PB-Spline, set of control points with its pair of knot vector is required. PB-Spline satisfies convex hull property. Also, it has no notion of control mesh, knot vector

of any basis function is independent of any other basis function knot vectors. With the notion of PB-Spline basis function, T-Spline basis function knot vector can be introduced. As already mentioned, control point $\mathbf{P}_i$ and its corresponding basis function $B_{i,j}(u,v)$ (Eq. 2.42) are defined on knot vectors $\overline{\mathbf{u}} = \{u_{i0}, u_{i1}, u_{i2}, u_{i3}, u_{i4}\}$ and $\overline{\mathbf{v}} = \{v_{i0}, v_{i1}, v_{i2}, v_{i3}, v_{i4}\}$ where the $\mathbf{P}_i$ knot coordinates are (v_{i2}, u_{i2}). If some knots are not visible on T-mesh, they are defined with a ray in parameter space $\mathbf{R}(\alpha)$. So for example, for some defined $u_{i,j}$, v_{ij} is given as $\mathbf{R}(\alpha) = (v_{ij} \pm \alpha, u_{ij})$ and vice versa. For a $\mathbf{P}_1$ from Fig. 2.21, knots are $\overline{\mathbf{v}} = \{v_1, v_2, v_3, v_4, v_5 - d_8\}$ and $\overline{\mathbf{u}} = \{u_1 - e_0, u_1, u_2, u_3, u_4 + e_9\}$, for $\mathbf{P}_2$ is $\overline{\mathbf{v}} = \{v_3, v_3 + d_6, v_5 - d_8, v_5, v_5 + d_5\}$ and $\overline{\mathbf{u}} = \{u_1, u_2, u_3, u_4, u_5\}$. For a boundary point $\mathbf{P}_3$ knots are $\overline{\mathbf{v}} = \{v_1 - d_0, v_1 - d_0, v_1, v_2, v_2 + d_7\}$ and $\overline{\mathbf{u}} = \{u_1, u_5 - e_4 + e_9 - e_7, u_5, u_5 + e_5, u_5 + e_5\}$. With defined knot vectors, T-Spline is evaluated as PB-Spline with Eq. 2.42, although some authors use weights in T-Spline surface evaluation like with NURBS [52], thus creating rational blending functions. New control points are inserted into T-mesh via knot insertion algorithm (Sec. 2.2.1). While performing knot insertion, additional rule has to be followed

Rule 3. New control point **A** can be inserted only if $v_1 = v_2 = v_4 = v_5$ for horizontal edge and $u_1 = u_2 = u_4 = u_5$ for a vertical edge.



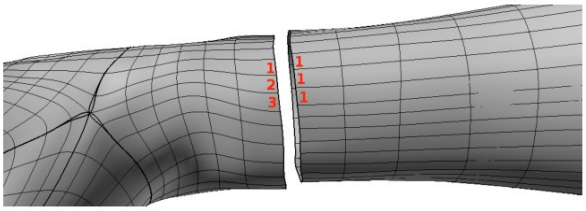
(a) **A** cannot be inserted. (b) **A** can be inserted.

Figure 2.23: T-mesh knot insertion [51].

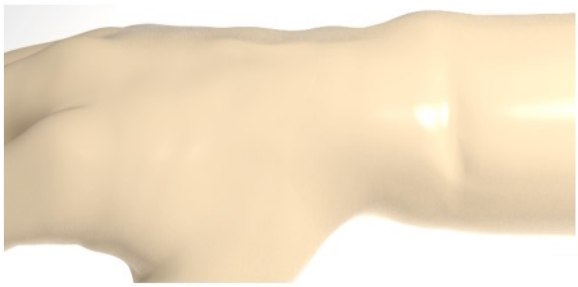
From Rule 3. and Fig. 2.23, new knot can only be inserted on an existing edge, so sometimes new faces have to be created with new knots. So it is obvious that insertion of a new control point can sometimes lead to multiple control points insertion in order to follow Rules 1.-3, where the number of new control points and knots can grow significantly. This problem was solved by Sederberg et al. in [52] with new local refinement. When inserting one control point with this local refinement, the number of additional control points can be highly reduced. Cubic basis functions can be refined as linear combination of $m-4$ B-Spline basis function, where m is number of new knots which contains old knots and one new which is to be inserted. Local refinement must not violate several rules during execution. Using the linear combination of basis function reduces the number of additional control points for insertion and it's termination is guaranteed. It's worth noting that T-Spline conversion to B-Spline and T-Spline knot removal is also described in [52], while merging B-Splines into T-Spline is defined in [51]. Sederberg et al. [52] also introduced three types of T-Splines:

- **Standard T-Spline**: $\sum_i B_{i,j}(u,v) = 1, \quad \forall (u,v) \in \Omega$.
- **Semi-standard T-Spline**: $\sum_i w_{i,j} B_{i,j}(u,v) = 1, \quad \forall (u,v) \in \Omega$, where not all $w_{i,j} = 1$.
- **Non-standard T-Spline**: $\sum_i w_{i,j} B_{i,j}(u,v) \neq 1, \quad \forall (u,v) \in \Omega$.

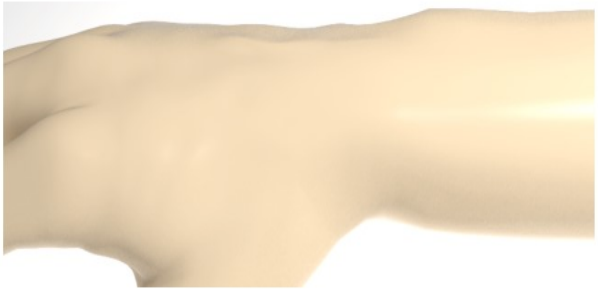
Regarding the problem of merging several NURBS models into single T-Spline, gaps can easily occur at intersections. This problem was exploited by Sederberg et al. [53] for two trimmed NURBS merging into one T-Spline. Key part of the algorithm is control points insertion and T-mesh creation at boundary where the surfaces are to be merged, thus creating C^2 continuity (Fig. 2.24).



(a) Mismatching knots of hand and arm model.



(b) Merge using the algorithm in [51].



(c) Merge using the algorithm in [53].

Figure 2.24: Merging NURBS hand and arm model with NU-NURBS algorithm in [53].

Buffa et al. [54] analyzed linear independence of T-Spline blending functions and proved it on several considered T-meshes for any odd polynomial degree. It has been shown that T-Spline blending functions are not always linearly independent and such as, authors gave method for defining a class of T-Splines blending functions who are linearly independent. Continuing on that work, Li et al. [55] defined class of T-Splines who are guaranteed to be linearly independent. These T-Splines obey several properties, such as non-negativity, convex hull property,

affine invariance and with some additional condition, PoU is retained. This class of T-Spline blending functions is called *analysis suitable T-Splines* because they are well suitable for IGA implementation, thanks to its properties. Analysis suitable T-Splines are obtained by T-junction extensions and are defined for cubic blending functions. Scott et al. [56] further developed local refinement algorithm for analysis suitable T-Splines using the nestedness T-Spline spaces. Regarding the dimensionality and nesting behavior of T-Spline space, Li and Scott [57] developed dimension formula for smooth polynomial spaces defined over Bèzier mesh of a T-Spline and defined nesting properties of T-Spline based on analysis suitable local refinement defined in [56]. About the T-Spline data structures, Asche and Berkham [58] presented half-edge data structure model for T-grids based on cell incidence relation, Xiao et al. [59] modeled T-Spline as object-oriented data and composed it into three layers, thus creating three layer model with great advantages for data storage, access and operations. Wang et al. [60] proposed half edge data structure for unstructured T-mesh with corresponding local refinement algorithm.
With the importance of keeping the B-spline properties (non-negativity, PoU, linear independence, compact support, etc.) in T-Splines and the usage of T-Spline in IGA and geometric modelling, Kang et al. [61] developed so called *modified T-Splines*. Modified T-Splines are locally refined T-Splines, constructed as linear combination of T-Splines blending functions defined in auxiliary mesh and are similar to THB-Splines. Blending functions are given as linear combination of the extended vertices, i.e. extensions of T-junctions in T-mesh for corresponding knot vector (just like extensions in analysis suitable T-splines). Extensions of the these knots define so called extended mesh T'. Evans et al. [62] introduced hierarchical analysis suitable T-Splines (HASTS), which are superset of both T-Splines and HB-Splines. With this formulation, complex T-Spline design can be encapsulated in the first hierarchy level, while higher levels can be used for adaptive multiresolution schemes. It's worthy noticing that authors in [62] considered only four-sided domains, while application of so far listed T-Splines and HASTS in arbitrary topological domains can be achieved with the use od spline forests [63] (Fig. 2.25).

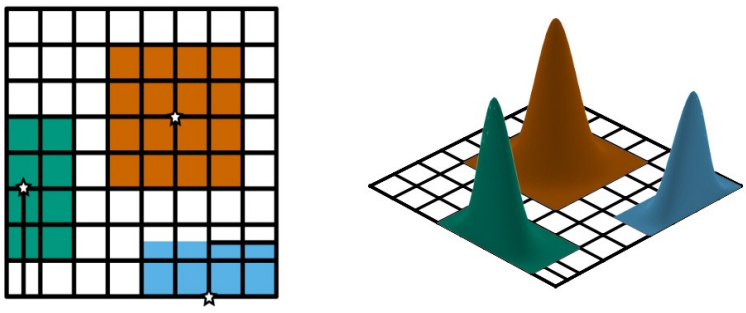


Figure 2.25: T-Spline basis functions and its support. Star denotes center of the basis function, i.e. basis function anchor [62].

HASTS are analysis suitable T-Splines [55] defined in hierarchical T-Spline space as a finite sequence of N nested analysis suitable T-Spline spaces. Sequence of N analysis suitable T-meshes $\mathcal{T}^l \subseteq \mathcal{T}^{l+1}$, $l = 0,...,N-1$ is constructed as follows:

1. $\mathcal{T}^{l+1}$ is created from $\mathcal{T}^{l}$ by subdivision of each cell in Ω^{l} in four congruent cells.

2. Extended T-junctions are inserted into $\mathcal{T}^{l+1}$ until it is analysis suitable

With defined hierarchical T-meshes, HASTS space is constructed by definition [62]:

Definition 4. *The hierarchical analysis suitable T-spline basis $\mathcal{T}$ is recursively defined as:*

(I) Initialization:

$$\mathcal{T}^0 = \mathcal{H}^0.$$

(II) Recursive construction of $\mathcal{T}^{l+1}$:

$$\mathcal{T}^{l+1} = \mathcal{T}_A^{l+1} \cup \mathcal{T}_B^{l+1}, \quad for\ l = 0, ..., N-2,$$

where

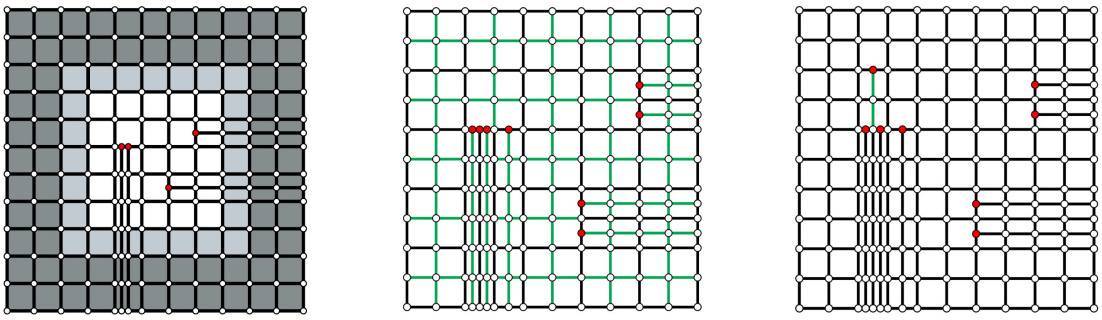
$$\mathcal{T}_A^{l+1} = \left\{ \beta \in \mathcal{T}^l : supp\ \tau \not\subseteq \Omega^{l+1} \right\},$$

$$\mathcal{T}_B^{l+1} = \left\{ \beta \in \mathcal{B}^{l+1} : supp\ \beta \subseteq \Omega^{l+1} \right\}.$$

(III) Result:

$$\mathcal{T} = \mathcal{T}^{N-1}.$$

HASTS preserve several important properties from analysis suitable T-splines, such as: linear independence, PoU, non-negativity, affine invariance, convex hull property, locally refinable, etc. Bezier extraction can be applied on HASTS, which converts T-spline hierarchy (or any other spline hierarchy) into single level elements and enables finite element formulation [62].



(a) Initial mesh $\mathcal{T}^0$.

(b) Creation of $\mathcal{T}^{l+1}$ from $\mathcal{T}^l$ by subdividing Bèzier elements.

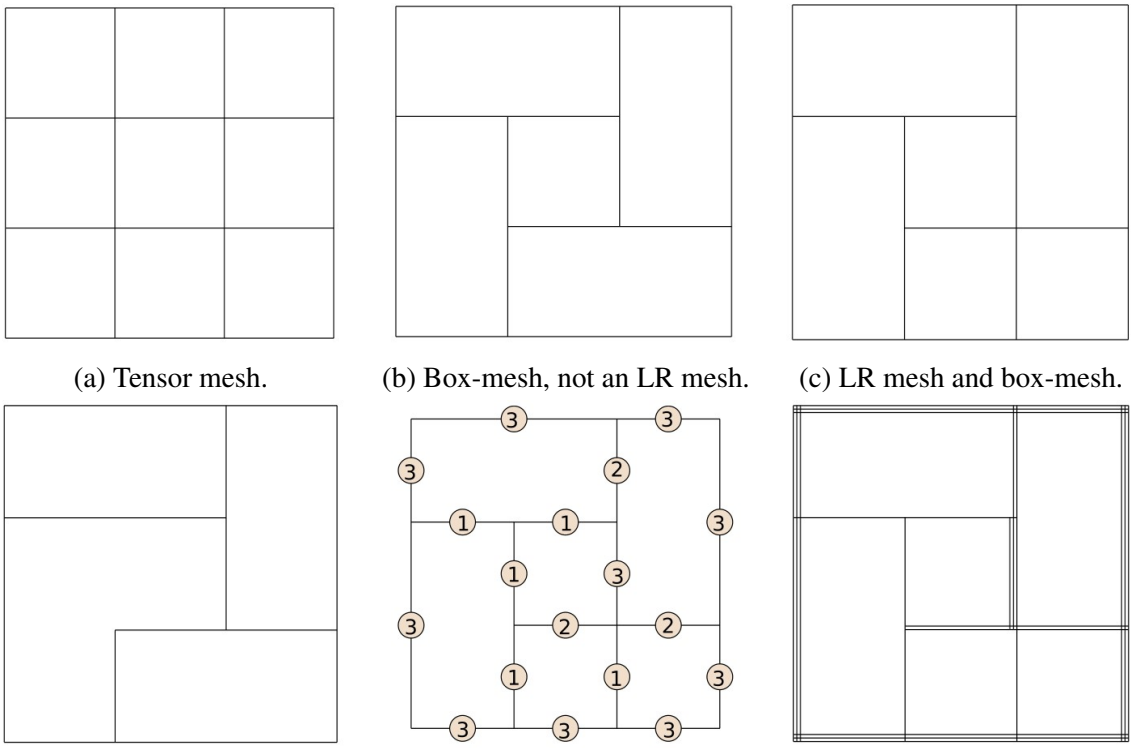
(c) Extended T-junctions until $\mathcal{T}^{l+1}$ is analysis suitable.

Figure 2.26: Construction of hierarchical analysis suitable T-meshes [62].

More details about T-Spline application in IGA and arbitrary polynomial degree can be found in [64] and [65], respectively.

2.3.4. Locally Refined B-Splines

New progressive, adaptively refined space of splines was introduced by Dokken et al. [66] with local refinement of axes parallel box partitions and box meshes, more known as *locally refined B-splines* (LR B-Splines).
First of all, *box-mesh* (or T-mesh) is defined as partitioning of 2D rectangular domain $[u_0, u_n] \times [v_0, v_n]$ into smaller rectangles by vertical or horizontal lines. Tensor mesh is a special case of box-mesh (or vice versa) without T-joints where all horizontal and vertical lines span entire domain. *LR mesh* $\mathcal{M}_n$ is a box-mesh gained through series of single line insertions $\{\varepsilon\}_{i=1}^{n}$ from initial tensor mesh $\mathcal{M}_0$, where $\mathcal{M}_n \subset \mathcal{M}_{n-1} \subset ... \subset \mathcal{M}_0$ and where every mesh state $\mathcal{M}_{i+1} = \{\mathcal{M}_i \cup \varepsilon_i\}$ is also a box-mesh [67].



(a) Tensor mesh. (b) Box-mesh, not an LR mesh. (c) LR mesh and box-mesh.

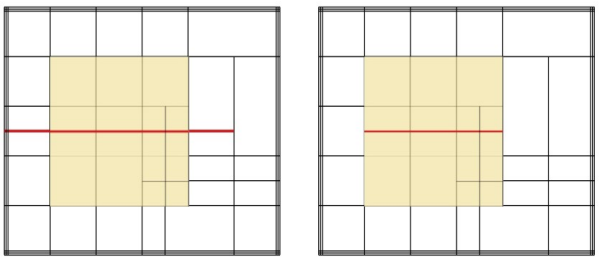
(d) Not an LR mesh, nor a box-mesh. (e) LR mesh with multiplicities. (f) Alternative representation of LR with multiplicities.

Figure 2.27: Examples of meshes used in definition of LR-Splines [67].

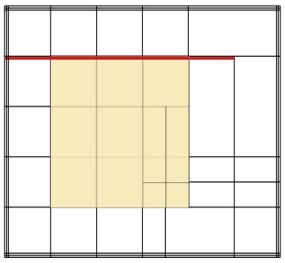
A box-mesh, tensor mesh or LR mesh with multiplicities is a mesh where each line segment has corresponding multiplicity integer n, where $0 < n \leq p$ and p is polynomial degree. From Fig. 2.27 can be seen that there is no way to create box-mesh on Fig. 2.27b from single line insertions starting at tensor mesh (Fig. 2.27a), where every state is also a box-mesh. This is prerequisite for all LR meshes.
Support of a B-Spline basis function $\beta : \mathbb{R}^2 \to \mathbb{R}$ is given as $B(u,v) = \beta(u)\beta(v)$, where knot vectors are $\overline{\mathbf{u}} = [u_0, ..., u_{p+1}]$ and $\overline{\mathbf{v}} = [v_0, ..., v_{q+1}]$. Sometimes, weighted B-Spline (or refined) is used to ensure PoU property (Sec. 2.3.2, Eq. 2.38). A meshline ε is said to traverse support

of a B-Spline β if [67]:

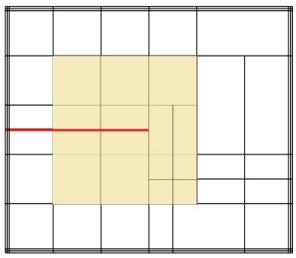
- horizontal line $\varepsilon = [u_0^*, u_1^*] \times v^*$ satisfies $u_0^* \leq u_0,\ u_{p+1} \leq u_1^*$ and $v_0 \leq v^* \leq v_{q+2}$
- vertical line $\varepsilon = [v_0^*, v_1^*] \times u^*$ satisfies $v_0^* \leq v_0,\ v_{q+1} \leq v_1^*$ and $u_0 \leq u^* \leq u_{p+2}$



(a) Line traversing the interior of β.



(b) Line traversing the edge of β.



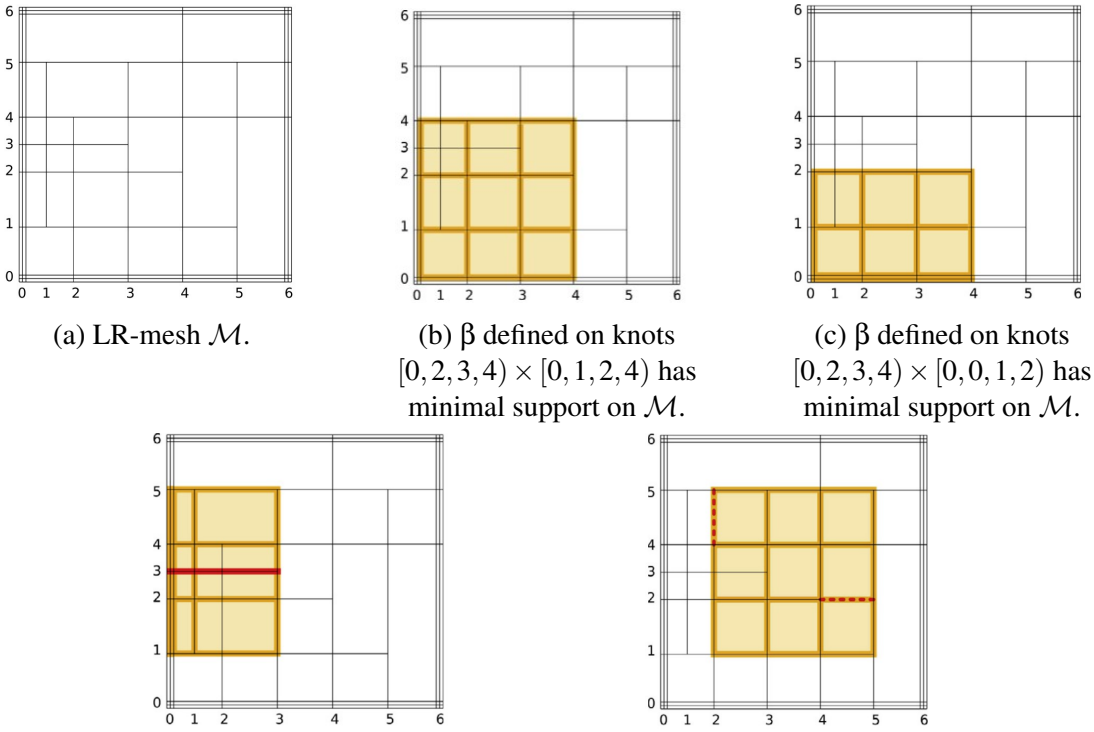
(c) Line neither traversing the interior nor the edge of β.

Figure 2.28: Examples of meshline ε traversing the support of basis function β [67].

A B-Spline basis function β has minimal support on a LR mesh $\mathcal{M}$ if for every horizontal line $\varepsilon = [u_0^*, u_1^*] \times v^*$ or vertical line $\varepsilon = [v_0^*, v_1^*] \times u^*$ of multiplicity n that traverses support of β in $\mathcal{M}$, there is:

- n number of unique i such that $v_i = v^*$ (horizontal ε) or $u_i = u^*$ (vertical ε), if ε traverses the interior if β
- an i such that $v_i = v^*$ (horizontal ε) or $u_i = u^*$ (vertical ε), if ε traverses the edge of β1

It can be seen on Fig. 2.28 that there is distinction when meshline ε traverses the interior or the edge of basis function support β. Minimal support ensures that every meshline traversing the support of B-Spline basis function appears in local knot vector and LR B-Spline ensures that every line in knot vector appears in LR mesh $\mathcal{M}$ (Fig. 2.29). This property of B-Spline basis functions defined as LR B-Spline basis was established by Dokken et al. [66]. B-Spline basis function β is called LR B-Spline on mesh $\mathcal{M}$ if all knot lines of β are contained in mesh $\mathcal{M}$ and if β has minimal support on $\mathcal{M}$. Also, meshline extension ε on LR-mesh $\mathcal{M}_n$ can be defined as [67]: a new meshline, an elongation of existing meshline, an union (join) of two or more existing meshlines or an increased multiplicity od existing meshline. New meshlines in $\mathcal{M}_n$ causes that one or more LR B-Splines does not have minimal support on $\mathcal{M}_{n+1}$.



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

(b) β defined on knots $[0,2,3,4)\times[0,1,2,4)$ has minimal support on $\mathcal{M}$.

(c) β defined on knots $[0,2,3,4)\times[0,0,1,2)$ has minimal support on $\mathcal{M}$.

(d) β defined on knots $[0,0,1,3)\times[1,2,4,5)$ does not have minimal support on $\mathcal{M}$ due to meshline $v=3$.

(e) β defined on knots $[2,3,4,5)\times[1,2,4,5)$ has minimal support on $\mathcal{M}$, but is not an LR B-Spline basis on $\mathcal{M}$ because two highlighted lines are missing.

Figure 2.29: Examples of minimal support basis function on LR mesh $\mathcal{M}$ [67].

By inserting new meshlines ε, local refinement of LR mesh $\mathcal{M}$ is achieved through knot insertion of new ε. New refined basis function is obtained by Eq. 2.25 (Fig. 2.15 is example of refined basis function where every initial knot vector is bisected through refinement, or Fig. 2.30 where quadratic basis function is refined by inserting one knot). For univariate case, when inserting single knot $\widehat{u}$ into knot vector $\overline{\mathbf{u}}$ between knots $\overline{u}_i$ and $\overline{u}_{i+1}$, refined basis β is given as [67]

$$\beta(u) = \alpha_1\beta_1(u) + \alpha_2\beta_2(u),$$

where

$$\alpha_1 = \begin{cases} 1, & \overline{u}_p \le \widehat{u} \le \overline{u}_{p+1}, \\ \frac{\widehat{u}-\overline{u}_0}{\overline{u}_p-\overline{u}_0}, & \overline{u}_0 \le \widehat{u} \le \overline{u}_p. \end{cases}$$

$$\alpha_2 = \begin{cases} \frac{\overline{u}_{p+1}-\widehat{u}}{\overline{u}_{p+1}-\overline{u}_1}, & \overline{u}_1 \le \widehat{u} \le \overline{u}_{p+1}, \\ 1, & \overline{u}_0 \le \widehat{u} \le \overline{u}_1. \end{cases}$$

LR mesh is given by successive insertion of new meshlines ε and splitting the box-mesh in two or by increasing the multiplicity of given meshline [66]. Note that there is no need to keep track of refinement history, only the current mesh $\mathcal{M}_n$ and spline space $\mathcal{S}_n$ is of interest. Through local refinement Pou is preserved and geometric mapping is unchanged. More detailed analysis

of LR mesh refinement can be found in [66, 67].

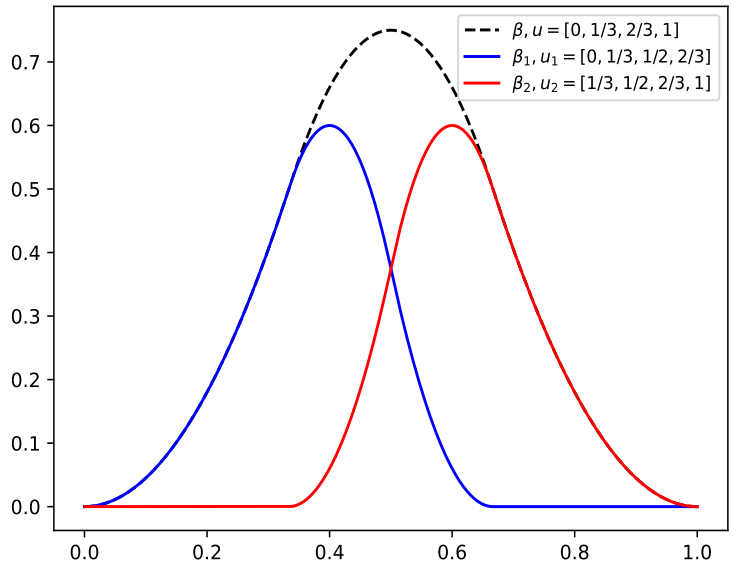


Figure 2.30: Refinement (or splitting) quadratic B-Spline basis function defined on knot $\overline{\mathbf{u}} = [0, 1/3, 2/3, 1]$ by inserting knot $\widehat{u} = 1/2$.

LR-Spline $\mathcal{L}$ is a pair $(\mathcal{M}_n, \mathcal{S})$ of mesh $\mathcal{M}_n$ and set of LR B-Splines $\mathcal{S} = \{\beta_i(u)\}_{i=0}^{m-1}$ where m is the number of B-Splines in the set $\mathcal{S}$. Dokken et al. [66] defined LR-Spline space with standard B-Spline basis, but refined are mostly used to keep PoU property. LR-Splines properties are [67]:

(L1) PoU: $\sum_{i=0}^{m-1} \gamma_i \beta_i = 1$

(L2) Nested nature of spline spaces: $(\mathcal{M}_{i+1}, \mathcal{S}_{i+1}) \subset (\mathcal{M}_i, \mathcal{S}_i)$

(L3) LR-Spline refinement is order independent i.e. resulting mesh is accounted not the order of the meshline refinements.

(L4) Spline space generally does not form linear independence of LR B-Spline basis functions.

Dokken et al. proposed certain scenarios of LR mesh for achieving linear independence and gave two alternative sets of B-Spline basis functions (rationally scaled and weighted) to achieve PoU [66]. Bressan in [68] extended the paper [66] and gave instructions for constructing LR B-Spline with defined number of overlapping support from which PoU and linear independence can be achieved. Bracco et al. [69] expanded the construction of LR-Splines through the generalized B-Splines on prescribed T-mesh, without standard meshline refinement. Generalized B-Splines on LR mesh preserve major properties such as PoU, local support and linear independence. Applications of LR-Splines in IGA for single-patch and multi-patch geometry can

be inspected in more detail in [67] and [70], respectively. Zimmermann and Sauer [71] introduced LR NURBS as extension of LR B-Splines. Rational basis functions are used (Eq. 2.18) which can model complex geometries unlike standard polynomials. Basis properties as local support, Pou, linear independence and convex hull property are preserved from NURBS and LR B-Spline theory. Additionally, when performing local refinement in LR NURBS, weights have to be treated in the same manner as control points, i.e. during the refinement corresponding weights are also refined. Also, Bèzier extraction operator was developed which enables LR NURBS application in finite elements and IGA. Analogous to that, Chen and Borst [72] introduced LR T-Splines, which takes advantages of T-Splines together with flexibility of local mesh refinement in LR-Splines. First major difference to standard LR B-Splines is input mesh which is T-mesh, unlike standard tensor product mesh (Fig. 2.27a). Final LR T-mesh is achieved through described local refinement i.e. new meshlines ε insertion instead of T-mesh standard refinement with control points. LR T-Splines preserve properties major properties as PoU, linear independence, nested nature of spline space, which enables their usage in finite element applications and IGA.

Johannessen et al. analyzed and compared HB-Splines, THB-Splines and LR B-SPlines in [73]. Each spline space, construction, properties and refinement procedure is described and compared, with emphasis on IGA. Authors analyzed sparsity of matrices and conditioning numbers of each spline for same numerical problems, where they established that exception of PoU in HB-Splines presents big deficiency in numerical applications and that THB-Splines or LR B-Splines should be preferred. From this results, it's logical to conclude that analysis suitable T-Splines should also be preferred over HB-Splines due to PoU preservation.

3. Methods for Fitting Standard Parametric Models

Fitting data points with parametric curve or surface is frequent problem in many scientific and engineering areas, such as computer vision, reverse engineering, CAD, etc. Goal is to minimize distance (errors) between parametric model and given data points. Data points can be in ordered or unordered form. Fitting the data points can be done by interpolation or approximation, where main focus of this paper will be on approximation methods. Fitting is defined as nonlinear optimization problem, where error or objective function is formulated through *distance based* or *coordinate based* algorithms [74]. Most popular objective function is defined as *least squares error* (LSE) minimization, from distance based algorithms, where the distances between data points and parametric model are squared and added up [75, 76, 77, 78]. In initial form, LSE equation is function of parameterization coefficients (parametric values of data, knots, weights, etc.) and control points, which makes problem nonlinear

$$E(\mathbf{u},\mathbf{Q}) = \sum_{i=1}^{N} ||\mathbf{C}(u_i) - \mathbf{P}_i||^2, \tag{3.1}$$

where $\mathbf{C}(u_i)$ is curve/surface value for given parametric value u_i, $\mathbf{P}$ are data points (mostly from PC), N is number of data points, $\mathbf{u}$ are parameterization coefficients and $\mathbf{Q}$ are control points. Additional formulations of LSE problems and solution methods can be seen in [78], but the one in Eq. 3.1 is most popular and widely used.

When parameterization values are obtained, LSE problem becomes function of just control points ($E = f(\mathbf{Q})$). Hence, LSE minimization problem is reduced to linear LSE minimization which yields linear system of equations for a curve [76, 77, 78]

$$\mathbf{Q} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{P}, \tag{3.2}$$

and for a surface

$$\mathbf{Q} = \left[\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\right]\mathbf{P}\left[\left(\mathbf{B}^T\mathbf{B}\right)^{-1}\mathbf{B}^T\right]. \tag{3.3}$$

where $\mathbf{A}$ is $(n \times N)$ matrix of n basis (blending) functions $N_{i,p}(u)$ for N data points and analogously $\mathbf{B}$ is $(m \times N)$ matrix of m basis functions $N_{j,q}(v)$.

In most cases, fitting problems are occupied with defining optimal parameterization coefficients. After the coefficients are obtained, problem is then reduced to trivial linear LSE solution. Main focus of the following chapters will be a way of obtaining parameterization coefficients.

3.1. Data parameterization

First step when performing fitting procedure is to define parametric values of data points $\mathbf{P}$, regardless of the chosen parametric curve or surface. Parametric domain is consisted of u axis and/or v axis, depending if it is a curve or surface. Like already seen in Chapter 2, parametric models are (mostly) defined between $[0,1]$ so in order to evaluate LSE error (Eq. 3.1) every data point needs to have corresponding parametric values u_i. In earlier works [6, 8, 79] ordered data sets were parametrized using uniform method (Eq. 3.4a), chord length method (Eq. 3.4b) or centripetal method (Eq. 3.4c)

$$u_i = \frac{i-1}{N-1}, \quad i = 1,...,N-1. \tag{3.4a}$$

$$u_i = u_{i-1} + \frac{||\mathbf{P}_i - \mathbf{P}_{i-1}||}{\sum_{k=1}^{N} ||\mathbf{P}_i - \mathbf{P}_{i-1}||}, \quad i = 1,...,N-1. \tag{3.4b}$$

$$u_i = u_{i-1} + \frac{\sqrt{||\mathbf{P}_i - \mathbf{P}_{i-1}||}}{\sum_{k=1}^{N} \sqrt{||\mathbf{P}_i - \mathbf{P}_{i-1}||}}, \quad i = 1,...,N-1. \tag{3.4c}$$

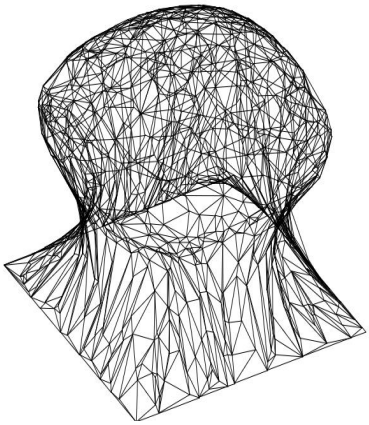
Where $u_1 = 0$ and $u_N = 1$. In the cases of B-Splines or NURBS, parametric values can be averaged per knot spans where. For unordered data sets, two often ways of parameterization are through mapping or nonlinear optimization. Similar iterative procedure was developed by Hoschek [80], where the distance error D_i between data point $\mathbf{P}_i$ and curve $\mathbf{C}(u_i)$ was minimized by projecting D_i on $\mathbf{C}(u_i)$ tangent.

Several mapping or projection techniques were developed over the years, where data points are mapped into parametric domain. Li et al. [81] introduced harmonic mapping, governed by Laplace equation with Dirichlet boundary conditions.

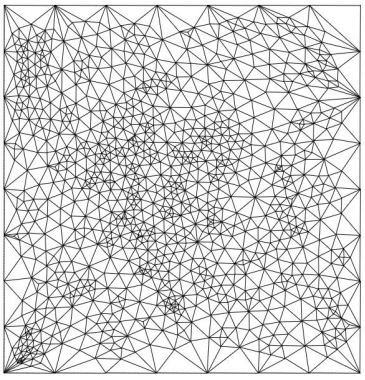
$$\Delta f = 0. \tag{3.5}$$

Equation yields linear system $\mathbf{Ku} = 0$, where internal nodes u_i are solved from values of boundary nodes u_j. For 3D surface parameterization, problem needs to be solved for u and v axis. Authors proposed FEM and FDM solutions of given problem for obtaining matrix of coefficients $\mathbf{K}$. Maillot et al. [82] presented solution of mapping Eq. 3.5 where matrix $\mathbf{K}$ is defined as minimization of deformation energy measured by Green-Lagrange deformation tensor from theory of elasticity [83]. Floater [84, 85, 86] developed several methods for mapping surface triangulations into 2D parametric domain, with different methods for obtaining coefficients $K_{i,j}$. When mapping into parametric domain on unit square (or unit line for 2D curves), corner points are selected and boundary values between corner points are mapped using one of methods in Eq. 3.4a-3.4c. In [84], Floater proposed uniform, weighted least squares and shape preserving parametrizations for mapping triangulated surfaces (Fig. 3.1). Afterwards, parameterized data are interpolated on desired grid from which new surface is created.

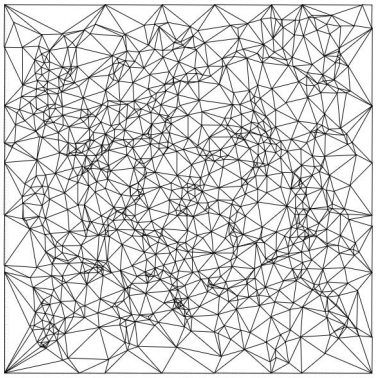
Floater and Reims generalized Floater's work in [84] and extended parameterization for un-



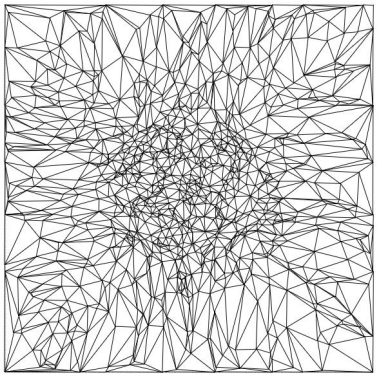
(a) Salt dome triangulation.



(b) Uniform parameterization of salt dome on Fig. 3.1a.



(c) Weighted least squares parameterization of salt dome on Fig. 3.1a.



(d) Shape preserving parameterization of salt dome on Fig. 3.1a.

Figure 3.1: Examples of proposed parameterizations by Floater [84].

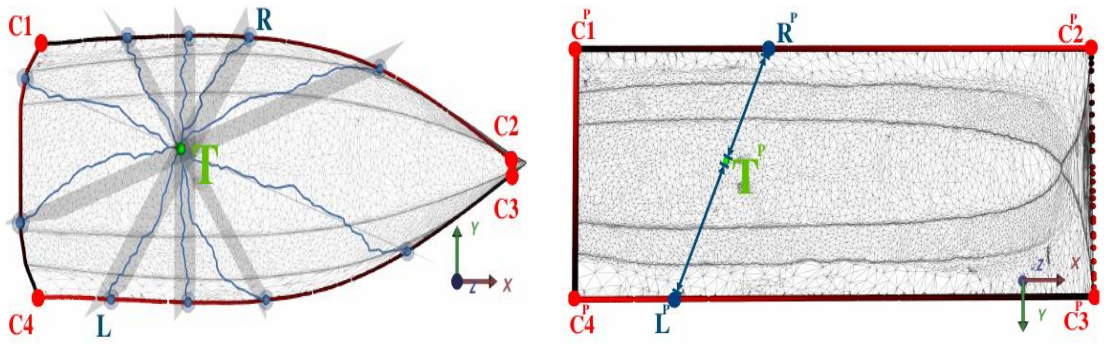
organized points where coefficient $K_{i,j}$ for each (i, j) point obtained as convex combination of its neighbouring points [85]. Coefficients $K_{i,j}$ are evaluated with methods from [84] but with account of neighbouring points in some arbitrary radius. Floater combined already mentioned works and developed solution for harmonic mapping where coefficients are obtained by the Mean Value Theorem for harmonic functions [86]. Ma and Kruth [87] parameterized unorganized data points by projecting surface into base surface, defined as unit square. Projection is done by minimizing distance between target surface and base surface, where the distance is expressed by normal vector from base surface to target surface. Harmonic mapping is most popular choice among mapping algorithms because its based on minimizing distortion in angles or areas (Dirichlet energy) and can be easily implemented with finite element or finite difference formulation [88]. Generally, harmonic mapping is not conformal (preservation of angles) but because it minimizes Dirichlet energy of angles or areas, results are very satisfactory. Mapping model based on elastic springs was used by Greiner and Hormann [89], where every triangulation edge is modeled as elastic spring. Model is based on elastic spring energy minimization where every matrix coefficient $K_{i,j}$ is defined as $K_{i,j} = 1/L_{i,j}$ for $i \neq j$ and for $i = j$ $K_{i,j}$ is eval-

uated from equation $\sum_{j=0}^{n} K_{i,j} = 0$ where n are all neighbouring vertices and $L_{i,j}$ is length of (i, j) vertex. Kragić et al. [90] developed hybrid mapping method for combining harmonic and elastic spring mapping

$$\mathbf{K_H} = \lambda \mathbf{K}_S + (1 - \lambda)\mathbf{K}_M, \tag{3.6}$$

where λ is blending ratio and $\mathbf{K}_H$, $\mathbf{K}_S$, $\mathbf{K}_M$ are coefficient matrices for hybrid, elastic spring and harmonic mapping, respectively. Lai et al. [91] proposed feature sensitive parameterization where the parametric data are allocate densely around sharp and complex features of given data. Data points are mapped into higher dimensional space of Cartesian coordinates and features deviation metric. Based on feature metric, data are projected into 2D parametric domain with clustering sharp points. Gaussian mapping can also be applied for unordered data set [92, 93]. Mapping is done by projecting curvature lines on a unit sphere which is afterwards stretched to unit square. Then, the parametric values are easily extracted from unit square. Conformal mapping is concept from complex analysis in which surface is mapped into complex plane with Riemann mapping theorem. This type of mapping preserves angles. For more details, please refer to [94, 95].

Jin et al. [96] presented conformal mapping using Euclidean surface Ricci flow [97]. Ricci flows is generalization of geometry circle packing. Authors presented this method for arbitrary surface topologies, where problem is solved using discrete Euclidean surface flow. Su et al. [98, 99] developed projection methods using optimal mass transportation theorem. In [98] they presented area preserving parameterization for surfaces with multiple boundaries. In first step, conformal mapping is done using Ricci flow and in second step, area distortion is corrected using mass transport optimization. In [99], presented volume preserving optimization. Boundary is mapped using harmonic projection, which is later used to compute volumetric harmonic map. In final step optimal mass transportation map is solved which gives volume preserving parameterization. Procedure is similar to harmonic mapping in a way that problem is reduced to linear system with different sets of differential equations. Ćurković et al. [100] presented novel projection od 3D geometry into 2D rectangular parametric domain. First, the boundary is extracted and four corner points are selected with nodes in between corner points linearly placed. Then for any interior point **T** and some arbitrary number of sections, closest distances are calculated between point **T** and intersections of defined sections with boundary. For evaluated point, plane is defined and parametric coordinates of point **T** are linearly evaluated (Fig. 3.2). Developed method obtains well structured geometries without wrinkling, but its computationally intensive. More detailed of existing PC parameterization methods can be seen in review paper by Zhu et al. [101]. Recently, authors wanted to train *Neural Networks* (NN) [102] for parameterization of data points. Scholz and Jüttler [103] trained deep neural network for parameterizing and approximating polynomial curves with small set of input data. Giannelli et al. [104] implemented convolutional neural network for predicting coefficients of Floater and Reims parameterization [85]. Rios et al. [105] also trained neural network for predicting Floaters and Reims param-



(a) Geometry to be projected. (b) Rectangular projection.

Figure 3.2: Procedure of projection by [100].

eterization, but for several complex combinations of coefficients. From this set of combinations, stochastic gradient descent was applied to find optimal global aprameterization. When using some of projections (mapping) methods, every parametrized point can be input in fitting method. In that case, parametrized data are in form of a vector and every point enters linear LSE fitting. Otherwise, parametric rectangular (or linear) domain can be discretized with arbitrary grid and using interpolation between grid and data points, parametric matrices can be obtained and used as input for linear LSE fitting.

3.2. Bèzier model fitting

When parametric values are defined with some method from previous section, next step when dealing with Bèzier curve fitting problem is solving linear LSE fitting problem. In order to get better solution, authors defined nonlinear optimization problems for Bèzier model fitting. Most authors defined data parameterization as first iteration of optimization from which linear LSE problem is obtained at the end of every iteration. With this formulation, problem is very nonlinear and some of gradient based or metaheuristic methods were used as optimization algorithms. First nonlinear fitting problems with Bèzier curve or surfaces included parametric data optimization with gradient based methods and linear LSE solution [79, 106]. Initial parametric values were define from uniform or chord length method (Eq. 3.4a-3.4c). Additionally, Borges and Pastva [106] added additional functional into LSE equation, which is Jacobian of standard LSE formulation and used Gauss-Newton method to gain solution. After solving nonlinear functional, linear system is obtained and solved per every iteration.
Metaheuristic algorithms were also apllied for finding optimal parametric values of data. Iglesias, Galvez and et al. applied different metaheuristic algorithms, such as *Genetic Algorithm* (GA) [107], *Particle Swarm Optimization* (PSO) [108], *Clonal Selection Algorithm* (CSA) [109] which is method from *Artificial Immune Systems* (AIS) algorithms, *Firefly Algorithm* (FA) [110, 111], *Bat Algorithm* (BA) [112, 113], *Cuckoo Search Algorithm* [110] and *Functional Networks* [114]. In most cases, metaheurstic approach was applied for data parameterization and afterwards linear LSE solution is obtained and for some data points and control points were

variables of optimization. Summary of their work can be seen in Table 3.1. In case of functional

Fitting method	Optimization algorithm	Strategy	Ref.
LSE	Genetic algorithm	Control points and parametric value vectors **u** (and **v**) are variables of optimization (Fig. 3.3).	[115]
LSE	Functional network	Parametric coordinates are input data. Network fits control points with Bernstein polynomials acting as constraints.	[115]
LSE	Particle swarm optimization	Control points and parametric value vectors **u** (and **v**) are variables of optimization.	[116]
LSE	Clone selection algorithm	Parametric values were gained via CSA, afterwards standard linear LSE fitting (Fig. 3.5).	[117]
LSE	Firefly algorithm	Parametric values were gained via FA, afterwards standard linear LSE fitting.	[118]
Root mean square error (RMSE)	Cuckoo search	Weighted Bayesian energy functional with data points, control points and constraints ino bjective function.	[119]
LSE	Bat algorithm	Parametric values were gained via BA, afterwards standard linear LSE fitting.	[120]

Table 3.1: Metaheuristic, nature-inspired optimization approaches for Bèzier curve/surface fitting by Galvez, Iglesias et al.

networks [115], functional networks were used to fit control points with functional constraints in form of Bernstein polynomials. Goal was to find surface $\mathbf{S}(u,v)$ which satisfies system of functional equations

$$\mathbf{S}(u,v) = \sum_{j=0}^{n} \alpha_j(u) f_j^*(v) = \sum_{i=0}^{m} \beta_i(v) f_i(u), \tag{3.7}$$

where $\alpha_j(u)$ and $\beta_i(v)$ are coefficients of linearly independent functions. Taking into account that desired surface is Bèzier surface, improved functional equation for functional network is

$$\mathbf{S}(u,v) = \sum_{i=0}^{m} \sum_{j=0}^{n} \mathbf{P}_{i,j} f_i(u) f_j^*(v) = \mathbf{f}(u).\mathbf{P}(\mathbf{f}^*(\mathbf{v}))^T. \tag{3.8}$$

LSE function was applied during learning process where each neural function f_i was approximated as linear combination of functions $\{\phi_{i0},...,\phi_{im}\}$. In short, for input coordinates from parametric domain and Bernstein polynomial constraints, network fits Bèziers control points which results with approximated surface (Fig. 3.4).

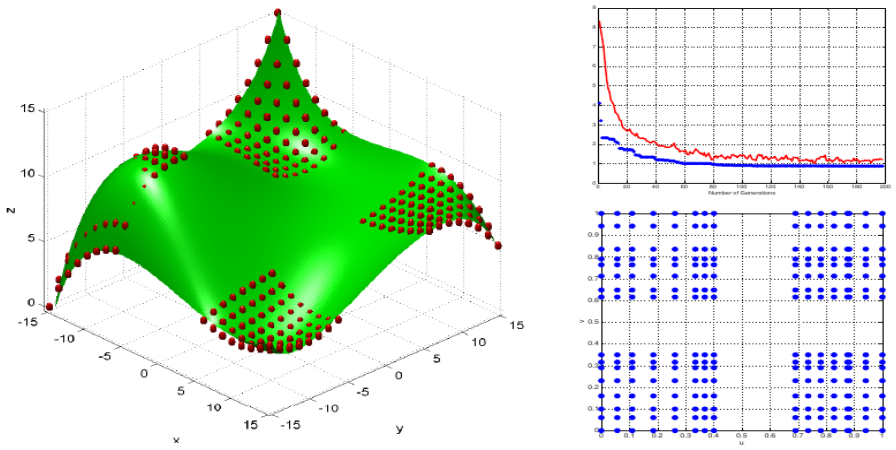


Figure 3.3: Bèzier surface fitting with GA: bicubic Bèzier surface and data points (left); evolution of mean errot and Euclidean errors (right-top); optimal parametric values for data points (right-bottom) [115].

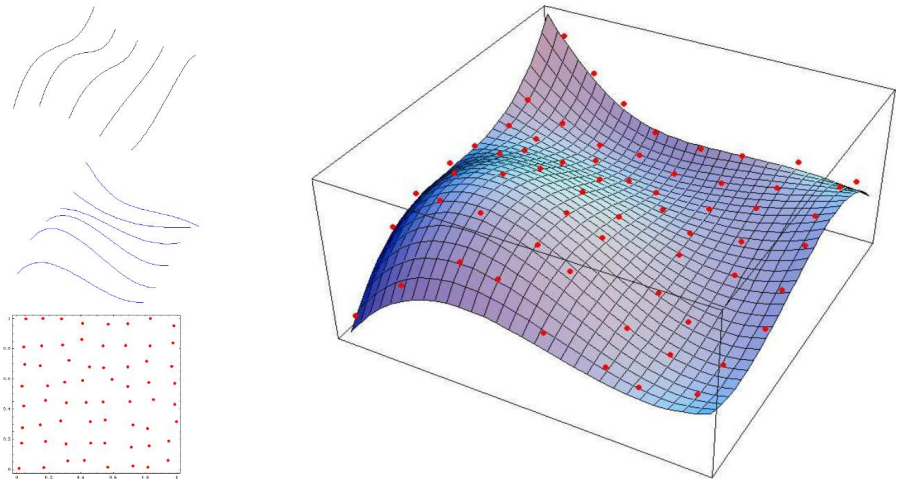


Figure 3.4: Functional constraints i.e. Bernstein polynomials (left, top and middle); data points parametric coordinates (left-bottom); fitted Bèzier surface (right) [115].

Interesting case is that in [120] bat algorithm optimization was first time applied in context of geometric modeling.

Pandunata and Shamsuddin [121] used *Differential Evolution* (DE) [122] (which is a generalization of GA) for fitting control points of Bèzier curve with data values defined by chord length method. Ueda et al. used *Simulated Annealing* (SA) [123] (probabilistic metaheuristic method) for fitting piecewise Bèzier curves [124]. Authors used piecewise Bèzier curve for fitting control points with C^1 continuity factors between curves, which were used as additional variables. Objective function was composed of LSE between curve and data points, fitting curve length and absolute difference between fitted curve length and data points length, where multi objective simulated annealing was used as an optimization algorithm.

Analytical optimization can be seen in [125] by Lifton et al., for nonlinear fitting of Bèzier

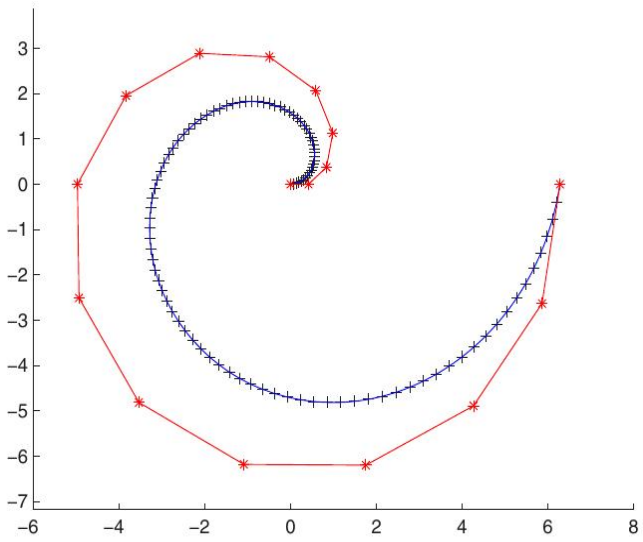


Figure 3.5: Archimedean spiral CSA fit: original data points (black symbols +), fitted Bèzier curve (blue line) and control polygon (red line + symbols *) [117].

surface on unstructured data points. Using fourth degree curve, fitting algorithm can evaluate complex noisy surfaces to the extent that the surface roughness can be evaluated. Given data points are uniformly spaced in parametric domain. Then algorithm follows four steps:

1. Linear LSE is used to calculate control points for initial uniform parametric values.
2. Jacobian is evaluated from linear LSE solution, then nonlinear LSE is used to calculate new parametric values.
3. Linear LSE calculates new control points for new parametric values from 2.
4. 2 and 3 are repeated, until convergence is achieved (relative squared distance between last two iteration is lesser than tolerance).

Developed algorithm proved to be stable, with smooth convergence (Fig. 3.6).

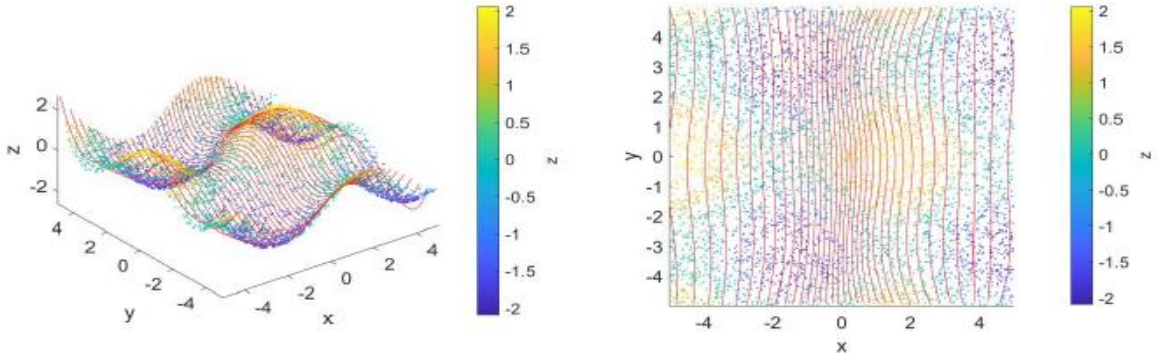


Figure 3.6: Unstructured data points (coloured dots) and fitted Bèzier surface (left); Top view of left figure [125].

Some authors presented fitting procedures for generalized Bèzier models. In [126], Iglesias,

Galvez and Loucera applied similar method like in their already mentioned papers. In this case, rational Bèzier curve was parametric model with blending functions [126]

$$R_i^n(u) = \frac{w_i B_i^n(u)}{\sum_{j=0}^{n} w_j B_j^n(u)}, \tag{3.9}$$

where w_i are weights and they make additional degree of freedom in optimization. Authors used metaheuristic algorithm SA, with two different schemes. Optimization variables were curve degree n, parametric values u_i, weights w_j and control points Q_j. For a range of degree n, parametric values and weights were optimized with SA, from where linear LSE solved control points. Then, using Bayesian information criterion (BIC) [127] objective function, best degree n is obtained. This optimization problem is highly nonlinear with all variables related to each other, thus it needed to be decomposed into several subproblems. Experiments showed that method is suitable for fitting, but it's performance is affected by noise intensity, meaning that some filtering operations are advisable before fitting procedure.

Zaman and Chowdhury [128] used DE on optimization of modified Bèzier curves. Modified Bèzier curves were introduced by Yang and Zeng [129], where curve (or surface) has additional degree of freedom in form of *shape parameter*. This parameter is similar to weights in NURBS (Sec. 2.1.4), but in this case it only affects y coordinate for curves and analogously z for surfaces. Modifier Bèzier curve is then defined as [128]

$$\mathbf{x}(u) = \sum_{i=0}^{n} \binom{n}{i} u^i (1-u)^{n-1} \mathbf{Q}_{x,i}, \tag{3.10a}$$

$$\mathbf{y}(u) = \sum_{i=0}^{n} \Lambda_i u^i (1-u)^{n-1} \mathbf{Q}_{y,i}, \tag{3.10b}$$

where Λ_i are *shaping coefficients* expressed with *shaping parameters* ζ_i as

$$\Lambda_i = \zeta_i \binom{n}{i}, \quad i = 0, 1, 2, ..., n. \tag{3.11}$$

Achieving optimum shaping parameters ζ_i, curve follows control polygon. In this case, LSE is defined between curve and control polygon. For a discrete set of parametric values u, curve y values are evaluated with Eq. 3.10b and control polygon values are evaluated with piecewise-linear interpolation between control points for evaluated x from Eq. 3.10a. LSE between curve and control polygon slope is also accounted, which yields weighted LSE objective function

$$E = w_1 E_c + w_2 E_s = w_1 \sum_{i=0}^{N} [y_c - y_{cp}]^2 + w_2 \sum_{i=0}^{N} \left[\left(\frac{dy_c}{dx} \right) - \left(\frac{dy_{cp}}{dx} \right) \right]^2, \tag{3.12}$$

where N is number of discrete points, y_c is Bèzier curve value, y_{cp} is linear interpolation value of control polygon and w are weight factors. Authors formulated this optimization problem

with ζ_i as variables and used DE as optimization algorithm. Results showed that optimization converged and that optimum curve matches shape and slope of control polygon with much higher degree of accuracy (Fig. 3.7). Proposed method is general, i.e. can be used on set of discrete data points where data points are regarded as control points. Ueda et al. [130]

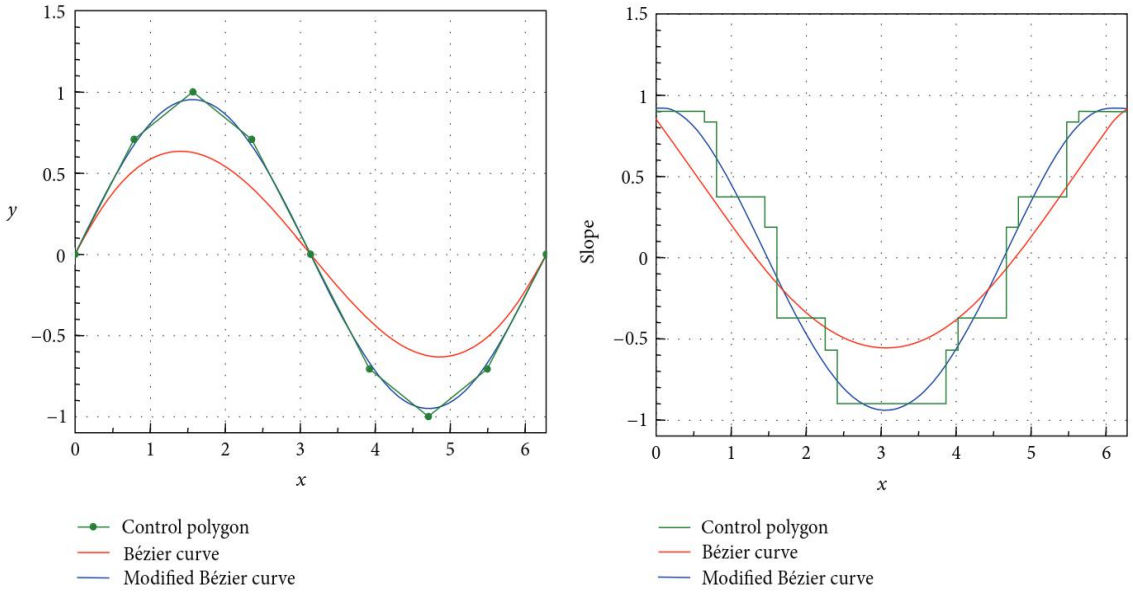


Figure 3.7: Comparison between modified and conventional Bèzier curve: fitted curve (left) and slope (right) [128].

extended his work from [124]. For a same formulation of the problem, single objective and multi objective optimization was performed with SA algorithm. Again, curve length error and LSE were formulated, where in multi objective optimization they were both objective functions and for single objective optimization they were weighted objective function. Single objective SA has achieved better approximation due to refinement parameters. Lu et al. [131] used elitist non-dominated sorting GA (NSGA-II) for fitting generalized cubic developable Bèzier-like surfaces. This class of Bèzier like surfaces has additional shape parameters λ_i, which are variables of optimization. A multi-objective optimization model is developed, based on shortest arc length, smallest energy and smallest curvature. Four optimization approaches are established by combining two of three objective criteria and finally combining all three. NSGA-II proved effective when solving this type of multi objective optimization, but with increased time complexity. Zain et al. [132] used fractional Bèzier curve as parametric model when fitting the data, which has two types of parameters: shape and fractional. With fractional parameters the improved continuity is developed, called fractional continuity. Parametric values can be chosen arbitrarily, fractional and shape parameters are then easily adjusted, but with higher computational time due to complex equations.

Regarding the fitting of Bèzier patches, few papers will be mentioned. Vučina et al. [133] proposed fitting of low degree partial surfaces (patches) with local control and imposed continuity, called chained piecewise Bèzier surfaces. Fortes and Medina [134] developed fitting

method for datasets containing a region in which no data is provided ("holes"). C^1 quadratic Bèzier patches were fitted with more emphasis on obtaining global fitting function, rather than local patches. Missing data are calibrated by several representative curves, which are then extended over missing data. Cui et al. [135] developed adaptive extension scheme when fitting Bèzier curve. Small segment of data points is approximated with curve, from where domain is enlarged and curve is extended up to another optimal data point. Using this scheme, data points can be fitted with one or several piecewise curves, depending on the data points. Method showed effectiveness in curve fitting, but with higher fluctuation of data points the number of curve segments can increase drastically. Utilizing higher curve degree can sometimes provide feasible approximation. Recently, new research interest has risen with development of autonomous mobile robotics, where Bèzier parametric model has found its application in trajectory optimization and path planning [136].

3.3. B-Spline model fitting

When dealing with B-Spline model, additional degree of freedom in terms of knots is included (Sec. 2.1.3). B-Spline basis functions aren't global, i.e. they are defined on knot spans $[\overline{u}_i, \overline{u}_{i+p+1})$ from which they have local support and thus more flexibility. When fitting B-Spline curve or surface, main problem is definition of the knot vector(s). Data points are in most cases parametrized with some method from Sec. 3.1 and in order to obtain B-Spline coefficients (control points) with linear LSE system (Eq. 3.2 and Eq. 3.3), knots position have to be determined. Main focus of B-Spline fitting are procedures for obtaining optimal knot positions, which will be discussed in this section.

When fitting B-Spline curve or surface, number of knots is usually predetermined with first $p+1$ knots being 0 and last $p+1$ knots are 1 (Eq. 2.11). With that, internal knots are variables of nonlinear optimization. Simplest way is that internal knots are defined by uniform, chord length or centripetal method (Eq. 3.4a-3.4c) [6, 8, 76, 137].

Knots can also be parametrized with methods based on data points values or deviation. Trivial method is averaging technique [6], where for fixed knot number knots are averaged from data points. Regarding data deviation, *Inverse chord length* (ICL) method [138, 93] is used which achieves high compression ratio by allocating fewer knots to regions with small deviations and more knots to regions with higher shape deviation, by uniform sampling. Park and Lee [139] introduced dominant points for knot placement in curve fitting. Data points are parameterized (chord or centripetal method), from which dominant points are calculated. Their calculation is based on parametric values deviation, which gives more dominant points around dense and complex regions and less at flat regions. Finally, dominant points are averaged which yields knot values. Also, some knots placement techniques are developed like KTP where knots are spanned in a manner that each knot span contains same number of parametric values. In the

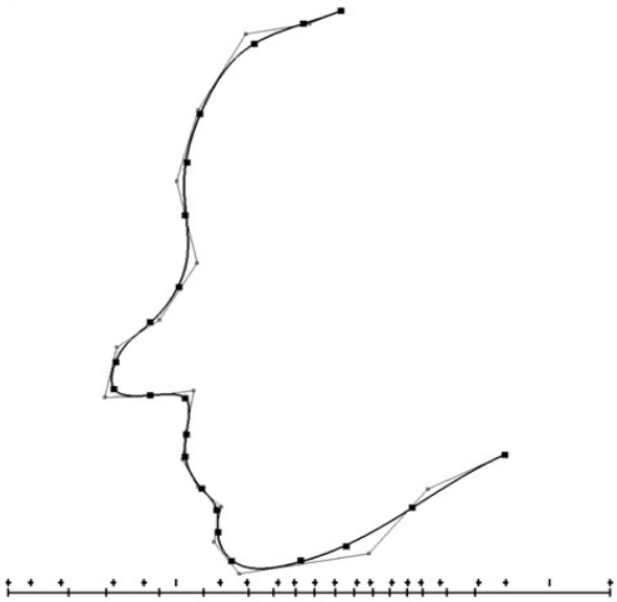


Figure 3.8: Curve obtained by LSE minimization using dominant points method proposed in [139]. Dominant points are denoted by '+' sign, with knot vector underneath.

framework of optimization, knot placement is nonlinear problem with constraints where knots should make increasing sequence. Piegl and Tiller developed similar method to KTP called NKTP [140], with adding flexibility to knots and knot removal algorithm. Initial knots can freely move and with knot removal, knot vector is reduced to keep distance error within tolerance. Resulting knot vector is generalization of averaging technique. Li et al. [141] developed adaptive knot placement algorithm based on curve's curvature. Highly noisy data are exposed with curvature characteristics. Afterwards, they are filtered and new knots are placed to reduce data noise and density.

Knot insertion technique can also be applied to improve flexibility when fitting B-Spline curve or a surface [137, 142]. Similarly, knot removal is also applicable, where initial dense knot vector is reduced [23, 24]. Some authors [143, 144] formulated unconstrained optimizations from constrained one, with LSE minimization and using gradient based optimization methods for achieving optimal B-Spline curve. Recently, sparse optimization in knot calculation began to develop. Kang et al. [145] developed sparse optimization algorithm for knot reduction based on works [146, 147]. Initially, dense knot vector is provided. From initial vector, subvectors are defined which reduce problem to sparse fitting. Satisfactory subvectors replace initial knot spans, and the process is repeated until the knot vector is small enough and approximated curve error is within error tolerance with initial curve. Occurred problems in algorithm are high time cost and deficiency in handling data with high noise.

Like with Bèzier model, many authors applied metaheuristic algorithms for defining knot vector. Some of interesting work has been done by applying GA to B-Spline fitting. Yoshimoto et al. [148] defined real-coded GA, where every individual in population was defined with real knot values and with variable size i.e. number of knots. Authors used BIC fitness function

defined as

$$BIC = Nlog_eE + (log_eN)(2n+p), \tag{3.13}$$

where E is point distance error term of LSE (Eq. 3.1), N is total number of data points, n and p are number of knots and polynomial degree, respectively. *Akaike Information Criterion* (AIC) [149] can also be used as fitness function and is written as

$$AIC = Nlog_eE + 2(2n+p). \tag{3.14}$$

According to [148], AIC gives redundant number of knots while BIC gives adequate number of knots which makes BIC better choice for fitness function. Parametric values and knot values can simultaneously be defined as variables of GA optimization [150]. With standard GA optimization, used fitness function was normalized root mean squared error (non dimensional form)

$$f = \frac{\sum_{i=1}^{N} \sqrt{(\mathbf{C}(u_i) - \mathbf{P}_i)^2}}{|P_{max} - P_{min}|}, \tag{3.15}$$

where $|P_{max} - P_{min}|$ is range of input data points. Kumar et al. [151] used parametric values as GA optimization variables. Knots are then averaged from resulting parametric values and finally, linear LSE solution is obtained. Fitness function was defined as maximization problem

$$f = \frac{1}{1 + E_{rms}}, \tag{3.16}$$

where E_{rms} is root mean squared error $E_{rms} = \sqrt{\sum_{i=1}^{N} e_i^2 / N}$ with e_i being Euclidean error for every data point. Garcia-Capulin et al. [152] used hierarchical GA definition for fitting B-Spline. Main novelty of this approach is hierarchical structure for representation of model structure (knots and its number) and model parameters (control points), called hierarchical gene representation. Bi-objective function was used and written as weighted combination of AIC and penalty functions for knot structure. As the result, algorithm finds best model with fewest knots, knots optimal position and optimal control points.

Like with Bèzier metaheuristic fitting, Galvez, Iglesias ad et al. contributed to that topic but with B-Spline model fitting. Summary of their work can be seen in Table 3.2. Functional networks are applied in same manner like with Bèzier model, where for parametric input data and B-Spline basis function act as constraints. The network fits corresponding control points [153, 154]. Galvez and Iglesias [155] used free knots as variables, where the number of knots and their values are unknowns. Results can be seen on Fig. 3.9. Also, multiplicity of a knots $k \geq 1$ can be achieved with free knots. Fitting problem with free knots leads to highly continuous multimodal and multivariate nonlinear optimization. In [156] Galvez, Iglesias et al. applied iterative procedure with two GA optimization as parts of iteration. GA optimizations for parametric values and knots, from which LSE system is solved with SVD (*singular valued decomposition*) or LU methods (Fig. 3.10). Finally, mean error is evaluated and new iteration

Fitting method	Optimization algorithm	Strategy	Ref.
LSE	Functional network	Parametric coordinates are input data. Network fits control points with B-Spline basis functions acting as constraints.	[153, 154]
LSE	Particle swarm optimization	Free knots as variables of optimization.	[155]
LSE	Genetic algorithm	Two steps optimization: (1.) parametric values for surface parameterization and (2.) knot values for surface fitting LSE system is solved using SVD decomposition. Iterative procedure until convergence is achieved.	[156]
LSE	Firefly algorithm	Knots are variables of optimization, from which linear LSE is solved.	[157]

Table 3.2: Metaheuristic, nature-inspired optimization approaches for B-Spline curve/surface fitting by Galvez, Iglesias et al.

begins until convergence is achieved. Galvez and Iglesias [157] optimized fixed knots with FA. They used four different fitness functions, LSE, RMSE, AIC and BIC and analyzed influence of each one on final result. Also, knots gained with FA were compared to some standard knot parameteriztion methods (uniform, chord length, ...).

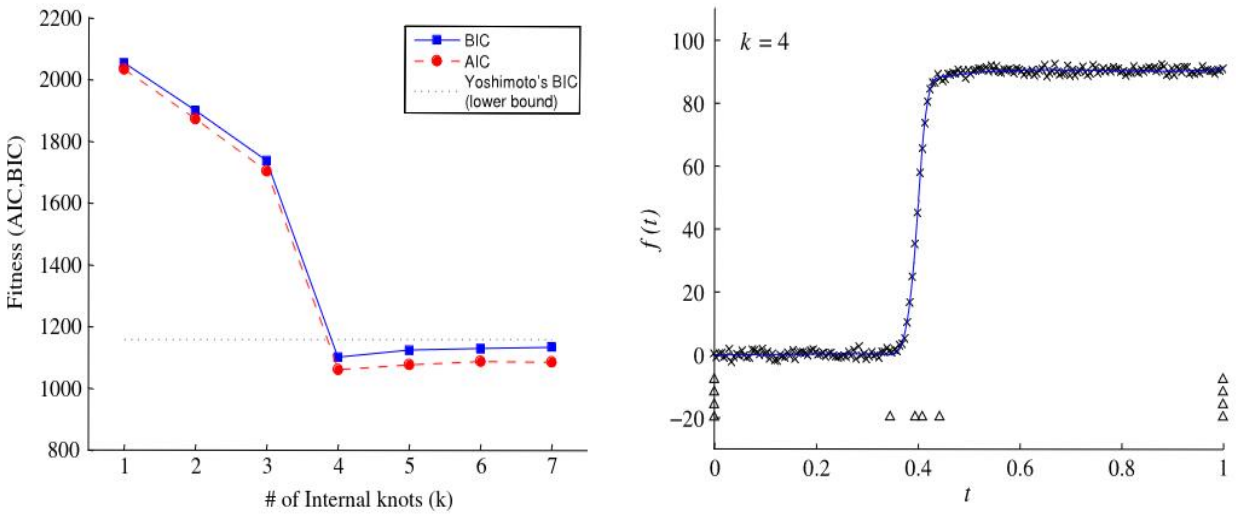


Figure 3.9: Curve obtained by PSO optimization of free knots (right) with AIC and BIC fitness functions [155].

So far discussed LSE objective function (Eq. 3.1) is also known as *point distance error*. LSE objective functions can also be formulated as [158] *tangent distance error*

$$E_{TD} = \sum_{i=1}^{N} || \left(\mathbf{C}(u_i) - \mathbf{P}_i\right) \mathbf{N}_i ||^2. \tag{3.17}$$

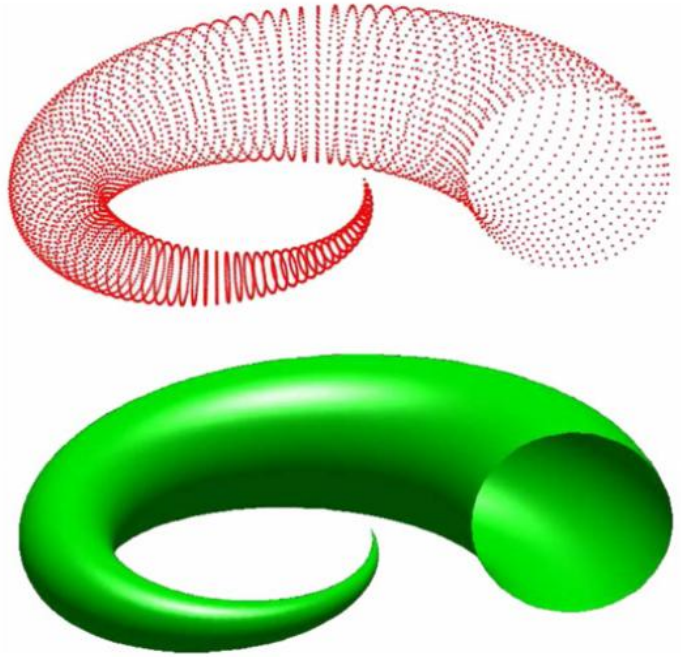


Figure 3.10: Reconstruction of horn surface form point cloud points (up) and (4,4)-order B-Spline surface [156] .

where $\mathbf{N}_i$ is unit normal vector of the fitted curve towards data point. Wang et al. [158] developed new LSE quadratic objective function called *squared distance error*

$$E_{SD} = \begin{cases} \dfrac{d}{d-\rho}\left[(\mathbf{C}(u_i)-\mathbf{P}_i)\mathbf{T}_i\right]^2 + \left[(\mathbf{C}(u_i)-\mathbf{P}_i)\mathbf{N}_i\right]^2, & \text{if } d < 0, \\ \left[(\mathbf{C}(u_i)-\mathbf{P}_i)\mathbf{N}_i\right]^2, & \text{if } 0 \geq d < \rho. \end{cases} \tag{3.18}$$

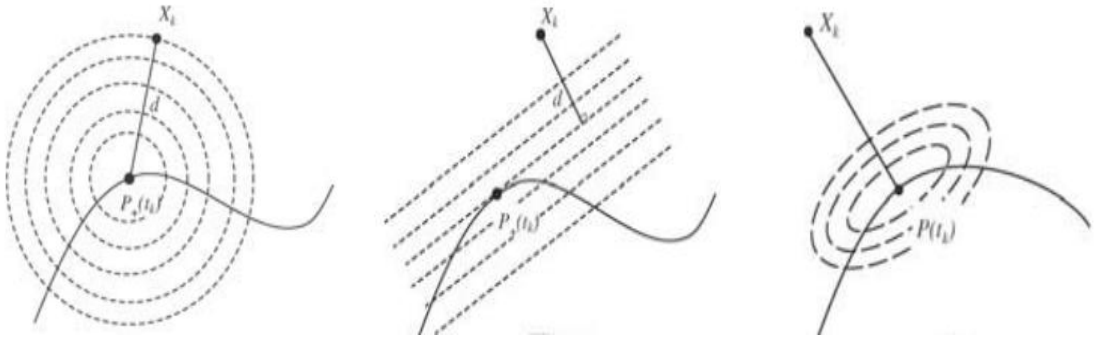
where d is closest distance between point $\mathbf{P}_i$ and curve point $\mathbf{C}(u_i)$, ρ is curvature radius of a curve $\mathbf{C}(u)_i$ at closest point and $\mathbf{T}_i$ is unit tangent vector between curve point and data point. Squared distance error becomes tangent distance error data point is sufficiently close to curve point, i.e. when $0 \geq d < \rho$.

LSE objective functions can also be formulated with additional smoothness functional as

$$f = E + \lambda f_s, \tag{3.19}$$

where E is arbitrary error distance term, λ positive weight and f_s smoothness functional. Weight λ is usually used as very small positive value. Smoothness functional acts as penalizing factor for deviating shapes [142]. They have to be defined as quadratic function at control points to keep system linear. Most commonly used smoothness functional is *thin plate energy* functional, involving partial derivatives of a curve or a surface [90, 159]

$$f_s = \int\int \left(||S_{uu}||^2 + 2||S_{uv}||^2 + ||S_{vv}||^2\right) dudv, \tag{3.20}$$



(a) Iso-values of point distance error.

(b) Iso-values of tangent distance error.

(c) Iso-values of squared distance error.

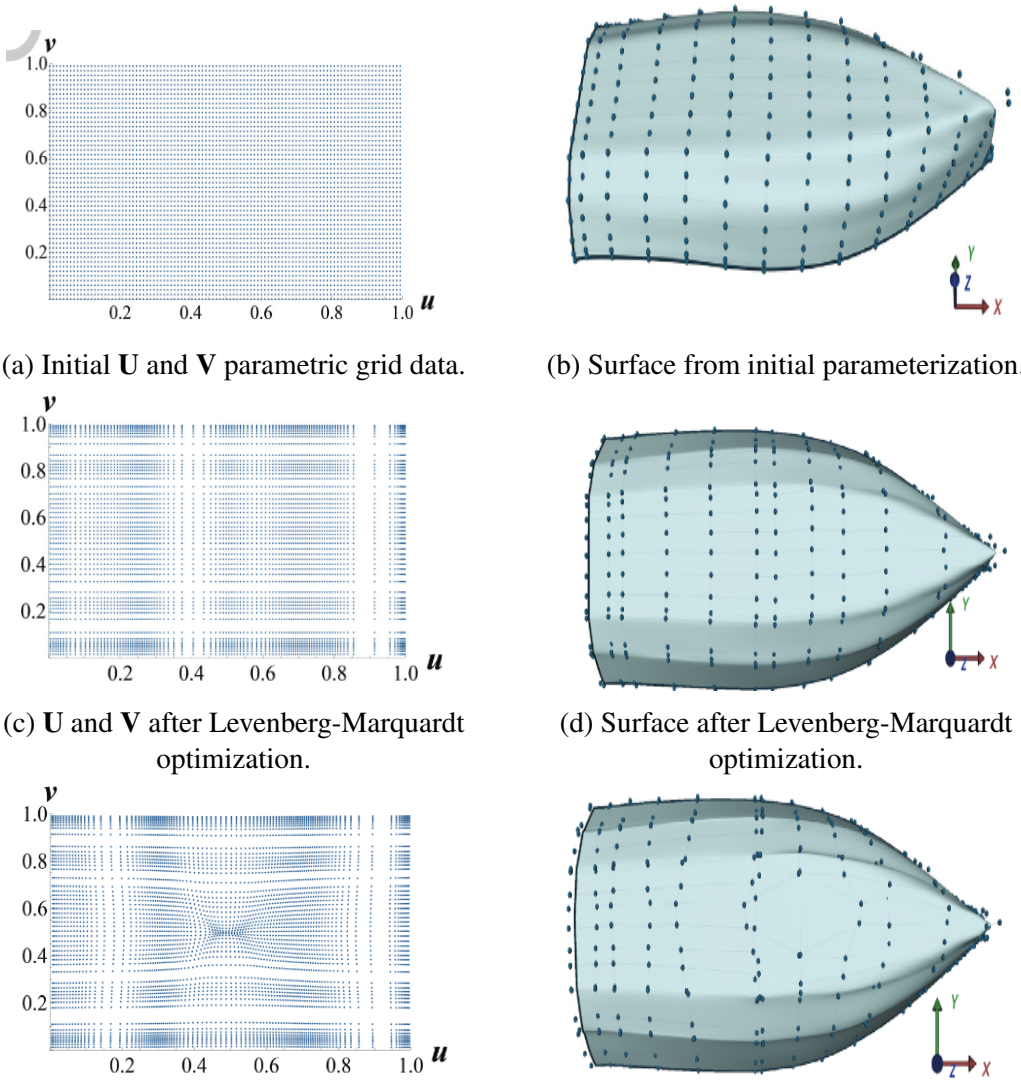
Figure 3.11: Iso-values of different LSE function formulation [158].

where indices denote partial derivatives of a surface $\mathbf{S}(u,v)$. Other functionals, such as *exact thin plate energy* which is based on principle curvatures [160] or *data dependent thin plate energy* [161] can also be applied for smoothing effect.
Ćurković et al. [162, 163] proposed re-parameterization fitting methods for optimization of parametric u and v grid data. In [162], grid data matrices were decomposed as

$$\mathbf{U} = \mathbf{C}^u + \mathbf{K}^u, \tag{3.21a}$$

$$\mathbf{V} = \mathbf{C}^v + \mathbf{K}^v, \tag{3.21b}$$

Initially, matrix $\mathbf{K}$ is initialized with input parametric grid data (Fig. 3.12a), u and v respectively. Rows and columns of the input grid data represent sections of initial geometrical topology. In first step of optimization, matrix $\mathbf{K}$ is optimized with Levenberg-Marquardt gradient method. Goal of first optimization is to stretch or shorten parametric rows and columns around areas with significant change of geometry (Fig. 3.12c). In second step, matrix $\mathbf{C}$ is initialized with optimal values from matrix $\mathbf{K}$. Matrix $\mathbf{C}$ is optimized with GA using exponential external functional which controls width of the change between pair c_u and c_v. Goal of second optimization is to reduce excess points in the areas outside any significant geometric features (Fig. 3.12e). Global goal of proposed fitting method is to reduce number of parametric values around geometric features. Doing this, number of knot spans is increased around that areas and with them control points which enables better fitting of features. First gradient optimization changes rows and columns of parametric grid and second changes every parametric point. Knots number is fixed and are uniformly distributed. With data re-parameterized and knots defined, surface is trivially obtained by linear LSE. In [163], Ćurković et al. followed same principle of re-parameterization, but instead of GA optimization with external functional, eigenvalues from Principal Component Analysis were applied. Using 2D Gaussian convolution and eigenvalues, points are redistributed around sharp feature to increase number of control points around feature. Similarly, Kragić et al. developed adaptive fitting method based on scalar field for point



(a) Initial **U** and **V** parametric grid data.

(b) Surface from initial parameterization.

(c) **U** and **V** after Levenberg-Marquardt optimization.

(d) Surface after Levenberg-Marquardt optimization.

(e) **U** and **V** after GA optimization.

(f) Surface after GA optimization .

Figure 3.12: Steps of enhanced fitting re-parameterization [162].

cloud re-parameterization which distributes B-Spline coefficients densely around parts with higher complexity. In [90], relaxation field was applied to input point cloud. Relaxation field distributes parametric values in such manner that resulting B-Spline control points distribute densely around complex shapes. Similarly, Kragić and Vučina [164] applied relaxation field based on plane-stress model which gives similar result like in previous work [90]. Additionally, several convergence criterion were investigated. Ebrahimi and Loghmani [165] developed adaptive fitting B-Spline method using scaled BFGS method [166]. Initial B-Spline is fitted with standard LSE system. Afterwards, scaled BFGS is applied on minimizing LSE error with length parameter. Length parameter scales the problem and keeps number of control points and precision of initial B-Spline curve. Also, new control points are inserted to increase curve flexibility and reduce fitting error.

Complex geometries can be fitted as B-Spline patches, enabling more flexibility and preci-

sion with local surfaces. Curve or surface patches can be stitched with three geometric continuity degrees, G^0 called watertight boundary, tangent plane continuity G^1 and second derivative continuity G^2 [167]. Milroy et al. [167] developed B-Spline patches fitting with G^1 continuity. Initial single patch B-Spline is fitted and later refined with constrained linear surface fit with G^0 continuity.

$$\mathbf{Q} = (\mathbf{A}^T\mathbf{A})\mathbf{A}^T(\mathbf{P} - \mathbf{P}_{edge}), \tag{3.22}$$

where $\mathbf{P}_{edge}$ are G^0 constraints in linear form. In further optimization, LSE is solved with Levenberg-Marquardt method where G^1 constraints are defined as penalty function. Zhang et al. [168] presented fitting method for triangle meshes of arbitrary topology. First, data is parameterized and partitioned is quadrilateral patches. Every quadrilateral patch is fitted with B-Spline using linear LSE solution. Patches are connected by imposing C^0 continuity via boundary control points.

Recently, authors implemented neural network for fitting B-Spline model. In most papers, neural networks are trained to predict parametric values and/or knot values. Laube [169] developed deep learning model for parameterization of data points and knot vector. Two deep neural networks are developed, one for data parameterization and another for knot placement. In most cases, authors used different structures of NN for predicting knot positions for achieving best fit of B-Spline model [170, 171, 172].

3.4. NURBS model fitting

NURBS model is extension of B-Spline model with additional degrees of freedom in form of weights w_i (of $w_{i,j}$ for surface). With already mentioned data parameterization and knot evaluation, weights have to be defined to construct NURBS model with weights being positive which makes NURBS fitting nonlinear constrained optimization problem. With that, main focus of currents section will be definition of weights for fitting NURBS model on data points.

Standard NURBS fitting problem would consist of input parameterized data and finding optimal knots and weights for such data. This kind of approach defines global nonlinear optimization problem. Knots have to be in increasing order and weights positive, which makes this constrained optimization. Such approach can be seen in [173], where Lauren-Gengoux and Mekhilef defined global NURBS fitting problem. Control points, knots and weights were variables of optimization with already mentioned constraints accounted. Objective function was standard LSE with knots and weights constraints inserted through penalty method. Problem was solved with several gradient based methods and results discussed. Detailed approach to NURBS fitting with gradient methods can be seen in [174].

Standard NURBS fitting procedure is mostly composed of two parts, first one being linear solution of B-Spline surface and second being optimization of NURBS weights for achieving best fit. Similar variations can be implemented, such as NURBS linear system define by Ma and

Kruth [175]

$$\begin{bmatrix} \mathbf{B}^T\mathbf{B} & 0 & 0 & -\mathbf{B}^T\overline{\mathbf{X}}\mathbf{B} \\ 0 & \mathbf{B}^T\mathbf{B} & 0 & -\mathbf{B}^T\overline{\mathbf{Y}}\mathbf{B} \\ 0 & 0 & \mathbf{B}^T\mathbf{B} & -\mathbf{B}^T\overline{\mathbf{Z}}\mathbf{B} \\ 0 & 0 & 0 & \mathbf{M} \end{bmatrix} \begin{Bmatrix} \mathbf{Q}_x \\ \mathbf{Q}_y \\ \mathbf{Q}_z \\ \mathbf{w} \end{Bmatrix} = [0]\,. \tag{3.23}$$

where $\mathbf{B}$ is matrix of basis functions values and $\overline{\mathbf{X}}$, $\overline{\mathbf{Y}}$, $\overline{\mathbf{Z}}$ are diagonal matrices od data points P_x, P_y and P_z. $\mathbf{M}$ is $n \times n$ (n is total number of control points for surface) non-negative matrix $\mathbf{M} = \mathbf{M}_x + \mathbf{M}_y + \mathbf{M}_z$, where i.e. $\mathbf{M}_x$ is defined as

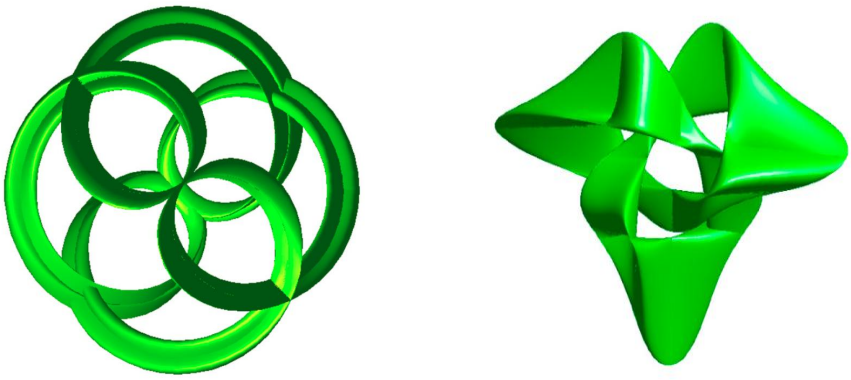
$$\mathbf{M}_x = \mathbf{B}^T\overline{\mathbf{X}}^2\mathbf{B} - (\mathbf{B}^T\overline{\mathbf{X}}\mathbf{B})(\mathbf{B}^T\mathbf{B})^{-1}(\mathbf{B}^T\overline{X}\mathbf{B}, \tag{3.24}$$

$\mathbf{M}_y$ and $\mathbf{M}_y$ are analogously defined, but with P_x and P_z points respectively. This linear system was defined by Ma and Kruth [175], as two-step linear approach. In first step, linear equation from fourth row of Eq. 3.23 is extracted for weights $\mathbf{w}$ and solved with symmetric eigenvalue decomposition technique. In second step, with weights obtained, control points $\mathbf{Q}$ are obtained from first three equation of system 3.23. In order to find best fit, simple constrained minimization problem is solved for finding best positive weights. Similar fitting approach was done by Heidrich et al. [176]. After fitting initial B-Spline curve, weights are optimized based on distance error between current (or initial) curve and data points. Wang et al. [177] defined weight alteration as $\Delta w_i = \delta_j/\alpha_{i,j}$. For some control point Q_i, δ_j is distance between data point P_j and curve value $\mathbf{C}(u_j)$ and $\alpha_{i,j} = \partial\mathbf{C}(u_j)/\partial w_i$. Term for Δw_i was linearized and curve fitting was solved iteratively until some certain tolerance was satisfied. Dimitrov et al. [178] applied similar approach to construction of lofted surfaces such as pipes. For certain number of cross sections, B-Spline curves were fitted. Afterwards, NURBS surface is fitted where weights are set to be equal to the inverse of smallest geometric distance.

Galvez and Iglesias [179] used PSO for finding NURBS parameters. 3D data points, NURBS polynomial degree and fixed number of control points were given as input. Variables of optimization were control points, knots and weights. Knots were constrained in $[0,1]$ interval and weights on $[0,2]$ interval. Linear LSE objective function was defined as

$$\mathbf{R}^T\mathbf{q} = \mathbf{R}^T\mathbf{R}\mathbf{p}, \tag{3.25}$$

where $\mathbf{R}$ is matrix of rational B-spline basis functions values, $\mathbf{q}$ and $\mathbf{p}$ vectors of control points and data points respectively. Linear system in Eq. 3.25 was solved by three different methods: standard LU decomposition, singular value decomposition (SVD) and modified LU decomposition for sparse problems. Proposed method yields very good results, even with problematic features (Fig. 3.13) . Ulker [180] used AIS algorithm for fitting NURBS curve. Knots, weights and control points were variables of optimization. Two step non-deterministic approach was applied. First, knot vector was optimized with respect to standard LSE, from which control points



(a) NURBS fitting Klein cycloid surface. (b) NURBS fitting tranguloid trefoil.

Figure 3.13: Surfaces obtained with PSO fitting [179].

were obtained. Secondly, weights were identified using AIS objective function. Exact approach also be applied to surface fitting problems. Costa et al. [181] proposed hybrid optimization procedure for NURBS framework. Number of parameters, i.e. number of control points, knots and polynomial degree were used as discrete (integer) variables. Non-decreasing knot values, control points and weights were defined as continuous parameters, which size is dependent on discrete variables. Data points were parameterized using chord length method. One iteration of hybrid optimization was composed of discrete variable optimization with GA and definition of continuous variables, specifically knots and weights with gradient based method. With parameters defined, control points are obtained form linear LSE. Goal of the hybrid optimization was to define given data points with minimal number of parameters.

Park et al. [182] proposed methods on fitting NURBS patches for triangulated mesh. Data points are segmented using K-means clustering algorithm. Every patch is approximated by polyhedron which is triangulated. NURBS network patch is created and every patch is fitted with G^1 (tangent plane) and C^2 (curvature) continuity.

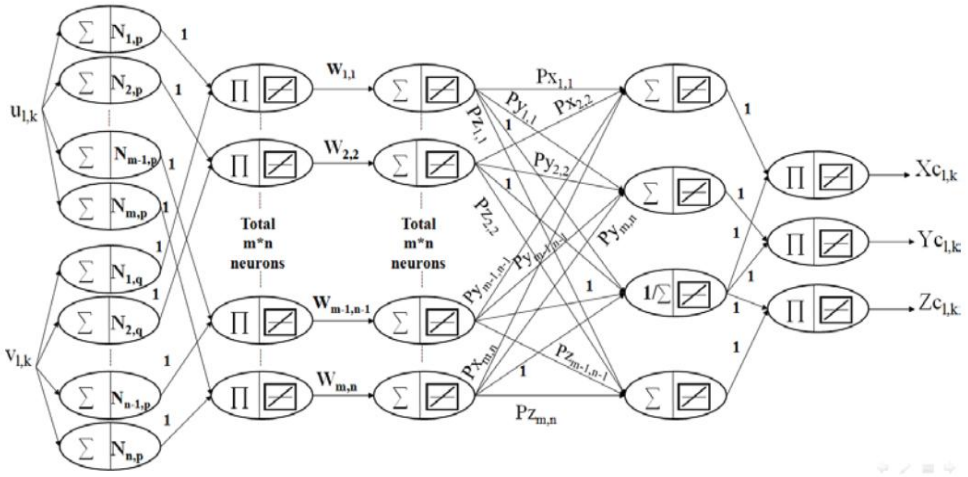


Figure 3.14: RBNN architecture [183].

Neural network can also be applied in NURBS fitting process. They are usually deployed for predicting weights and control points. Elmidany et al. [183] used *Rational B-Spline Neural Network* (RBNN), which contain B-Spline basis functions as one layer of functions inside network due to their higher approximation ability. Parametrized data points are input values and without the need for knot optimization, RBNN approximates surface weights and control points (Fig. 3.14). Network has two inputs: u and v parametric grid points and three outputs: x, y and z grid control points. Network consists of five layers. All activation functions are linear except first one which has B-Spline basis function. Connecting weights between second and third layer represent NURBS weights $w_{i,j}$. Similar approach in construction of NURBS surface using RBNN can be seen in [184]. Tian et al. [185] proposed three layer NN which predicts knot vectors for NURBS surface. Weights are obtained as weights between layers.

4. Methods for Fitting Advanced Parametric Models

For some complex geometries, standard parametric models are not sufficient as fitted parametric model. In order to better approximate local complex features, adaptive parametric models are used lately. Adaptive models as THB-Spline, T-Spline provide local refinement and addition of control parameters in parts of domain where geometric error exceeds given tolerance. This adaptive fitting process is done in several iterations until the tolerance is satisfied or geometric error converges. Adaptive fitting, independent of chosen parametric model, starts with initial simple curve or surface which is iteratively refined in domain where error is exceeded. Fitting is usually done by LSE formulation or usinq *Quasi-Interpolation* (QI) [21, 186]. QI is local approximation scheme for defining polynomial coefficients (control points) in prescribed polynomial space. In general, adaptive fitting results with smaller number of parameters (coefficients) than alternative standard parametric patches or models with dense coefficients. In this chapter, adaptive fitting procedures regarding HB-Spline, THB-Spline, T-Splines and LR-Splines will be discussed.

4.1. Hierachical B-Spline and Truncated Hierachical B-Spline model fitting

After introducing hierarchical B-Spline tensor product structure, Forsey and Bartels [187] presented local LSE approximation of gridded data. New levels are refined with dyadic knots and afterwards LSE linear system is obtained for new level. Following this work, Greiner and Hormann [89] proposed approximating method of scattered 3D data. Initial B-Spline is defined from parameterized data and uniform knots. Linear system is obtained with standard LSE and smoothness functional (Eq. 3.19). Any geometric constraints are solved through Lagrange multipliers λ_i. Afterwards, penalty method is applied and linear system is obtained

$$\left(\mathbf{A} + \omega \mathbf{B}^T \mathbf{B}\right) \mathbf{q} = \mathbf{p}, \tag{4.1}$$

where $\mathbf{A}$ is square matrix $M \times M$ with M being number of basis functions, ω penalty parameter, $\mathbf{B}$ is $N \times M$ with N being number of data points and $\mathbf{P}$ is vector of data points. With every iteration, critical parts of domain are dyadic refined and new system is obtained until the geometric error for every point is below given tolerance. Lee et al. [188] and Zhang et al. [189] proposed similar multilevel cubic B-Spline fitting methods for ensuring C^2 continuity. Subdomains with higher errors are recursively refined using defined through control lattice. Cartesian

z axis is referent for error determination, hence refined subdomain is expressed as bicubic B-Spline surface $z_c = \sum_{i=0}^{3}\sum_{j=0}^{3} B_i(u)B_j(v)Q_{i,j}$, for any point (x_c, y_c). In LSE sense, new control point are evaluated as $Q_{ij} = B_i(u)B_j(v)z_c / \sum_{k=0}^{3}\sum_{l=0}^{3} B_k(u)B_l(v)$. For any point c new $Q_{i,j}$ or Q_c will be obtained. Goal is to minimize error between new hierarchical B-Splines surface and z coordinates, hence new hierarchical level is expressed as

$$Q_{i,j} = \frac{\sum_c B_c(u)B_c(v)Q_c}{\sum_c B_c(u)B_c(v)}. \tag{4.2}$$

C^2 continuity is insured by expanding new subdomain by three rows or three columns which are called constrained strips (Fig. 4.1).

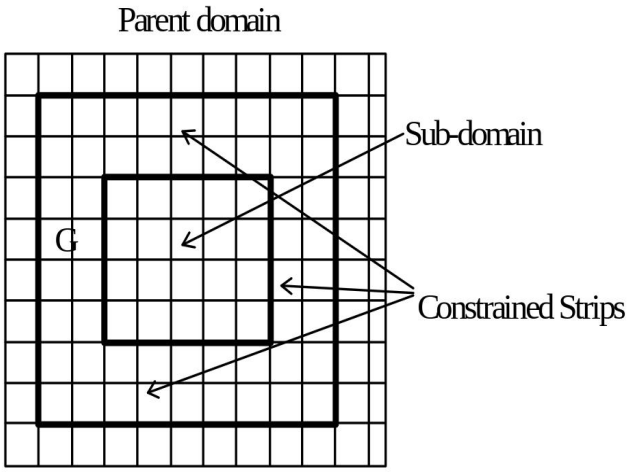
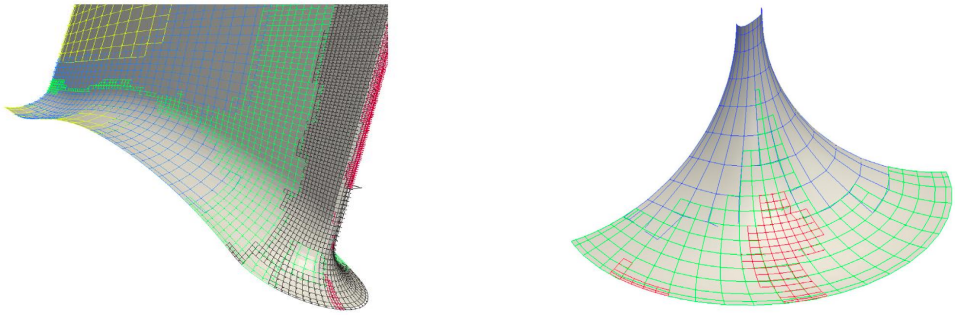


Figure 4.1: Overlapping between subdomain and parent domain for ensuring C^2 continuity [189].

Standard LSE fitting using adaptive hierarchical refinement was proposed Giannelli, Kiss et al. [40, 190, 49, 22]. Giannelli et al. [40] defined standard THB fitting with LSE function $\sum_{i=1}^{N}(f_i - \mathbf{p}_i)^2$ where f_i is THB-Spline model $\sum_{\tau\in\mathcal{T}} c_\tau \tau$ (c_τ are THB-Spline coefficients, i.e. control points denoted in [40]). Initial tensor product $\mathcal{B}^0$ is defined and LSE error is evaluated. Parts od domain where error is greater than some arbitrary tolerance is refined and new THB-Spline basis is obtained. Procedure is repeated for some number of iterations or until the tolerance is satisfied. Kiss et al. [190, 22, 49] followed same procedure, but with additional smoothness term in LSE formulation resulting with sparse linear system

$$\left(\mathbf{A}^T\mathbf{A} + \lambda E\right)\mathbf{q} = \mathbf{A}^T\mathbf{p}, \tag{4.3}$$

where $\mathbf{A}$ is matrix of THB-Spline basis functions, E is smoothness matrix. Result of [49] can be seen on Fig. 4.2. QI schemes have been highly used and developed in local adaptive fitting of hierarchical spaces. Kraft [33] discussed QI for computing B-Spline coefficients as weighted



(a) Turbine blade fillet represented by THB-Spline.

(b) Detail of turbine blade filler from left figure.

Figure 4.2: Turbine blade fillet THB-Spline representation. Different mesh colors represent different hierarchy level [49].

average of a function values. For any (sub)domain Ω^l, general QI has the form

$$P^P f = \sum_k (P_k^p f) B_k^p, \tag{4.4}$$

where P_k^p are linear functionals. Krafts proposed QI results only with few B-Spline coefficients for any given level. Speleers and Manni [191] presented QI for THB-Spline projectors for any polynomial degree p. In order to construct new hierarchical space $\mathbb{S}_\Omega$, corresponding coefficients $\lambda_{i,l}$ (control points) have to be determined using QI definition

$$\mathfrak{Q}(f) = \sum_{l=0}^{N-1} \sum_{i \in \mathbf{I}_l} \lambda_{i,l}(f) B_{i,l,\Omega_N}^{\tau}, \tag{4.5}$$

where N is number of hierarchical levels in this context, $\mathbf{I}_l$ is sed of active basis functions at level l and Ω is hierarchical level domain. Afterwards, Speleers [186] presented new QI schemes for uniform hierarchical meshes with fewer evaluations, $O(1)$ complexity per degree of freedom. Buffa and Garau [193] extended QI projector for hierarchical spaces by enabling construction of open knot vectors with higher multiplicity of internal knots. QI can be also combined with LSE evaluation for fitting hierarchical surfaces. Bracco et al. [194] used LSE approximations and combined it with QI for fitting scattered 3D data. Critical subdomains are extracted via LSE and local refinement is done by QI scheme. Bracco et al. [192] presented similar combination of LSE combined with QI. In first stage, linear LSE solution is obtained. In second stage, QI is used for obtaining THB-Spline definition. Results of LSE+QI combined fitting can be seen on Fig. 4.3. More detailed analysis of given approach can be seen in [195]. Giust et al. [196] similarly proposed QI scheme based on LSE fitting for two new spline projectors, one for HB and other for THB. Local LSE fitting solution was obtained using spline projectors.

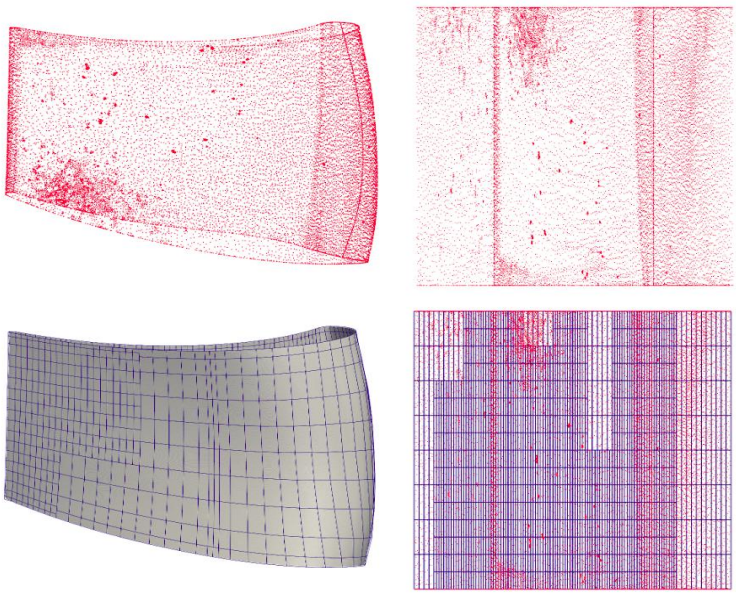


Figure 4.3: Scattered data of critical turbine blade airfoil part (top), reconstructed surface and corresponding hierarchical mesh [192].

4.2. T-Spline model fitting

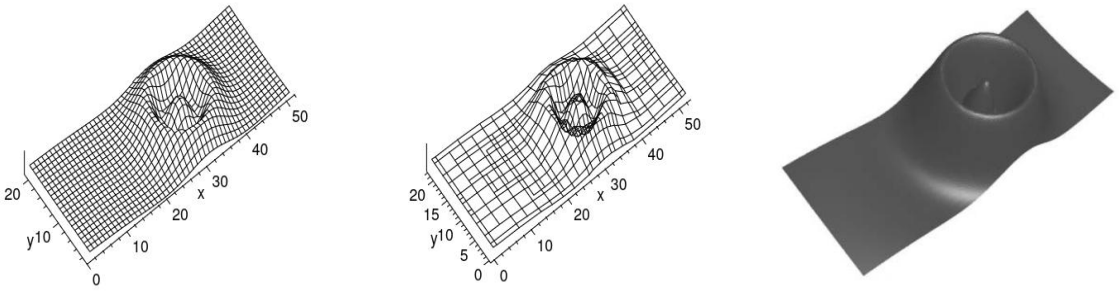
In the first paper regarding T-Spline fitting, Zheng and et al. [197] proposed LSE formulation for grid data, where initial B-Spline surface is adaptively refined, thus creating T-mesh. Fitting model was semi-standard T-Spline (Sec. 2.3.3) which equations is

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n-1} w_i B_{i,j}(u,v)\mathbf{Q}_i}{\sum_{i=0}^{n-1} w_i B_{i,j}(u,v)}, \tag{4.6}$$

LSE is formulated with thin plate energy

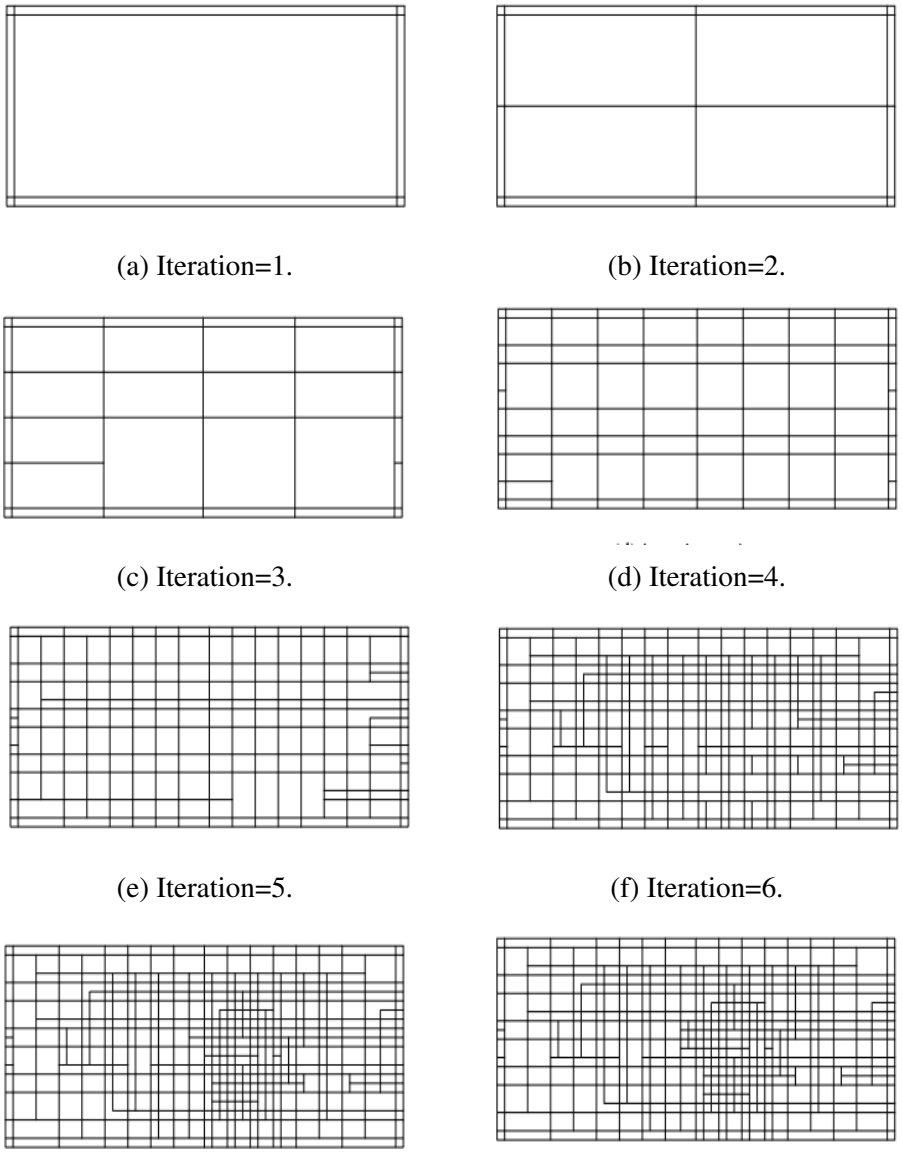
$$E = \sum_{i=1}\sum_{j=1} (\mathbf{S}(u,v) - z_{i,j})^2 + \lambda f_s. \tag{4.7}$$

Starting surface is cubic B-Spline, which mesh is later adaptively refined into T-mesh. T-mesh is achieved by splitting regions with higher geometric error via local knot insertion algorithm. Results of proposed optimization are on Fig. 4.4 and region splitting per iterations can be seen on Fig. 4.5. Same principle was applied by Wang and Zheng [198] for PC data. Input PC is parameterized on unit square using mean value coordinates method [86]. Starting from simple T-Spline grid, points with highest error are refined and T-mesh is created for semi-standard T-Spline (Fig. 4.6). Same authors proposed new adaptive T-Spline fitting algorithm [199] based on their previous work. Novelty of this paper in comparison to previous one ([198]) is error estimation. For every parameterized vertex of data point p_i discrete mean curvature h_i is



(a) Input data. (b) T-mesh. (c) T-Spline surface.

Figure 4.4: Input data, T-mesh and T-Spline surface obtained by fitting method in [197].



(a) Iteration=1. (b) Iteration=2.

(c) Iteration=3. (d) Iteration=4.

(e) Iteration=5. (f) Iteration=6.

(g) Iteration=7. (h) Iteration=8.

Figure 4.5: Fitting iterations of problem on Fig. 4.4 [197].

evaluated as

$$h_i = ||1/(4A)\sum_{j=1}(cot\alpha_j + cot\beta_j)(p_j - p_i)||, \tag{4.8}$$

where A is the sum of triangle areas adjacent to the vertex p_i, α_i and β_j are two angles opposite

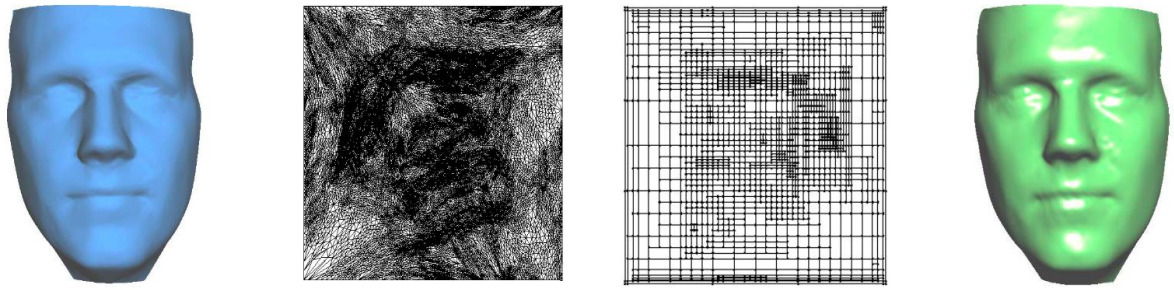


Figure 4.6: PC Triangulation, mean value coordinate projection, T-mesh, T-Sline surface [198].

to the edge connecting p_i and p_j (Fig. 4.7). Curvature guidance factor k_i is evaluated as

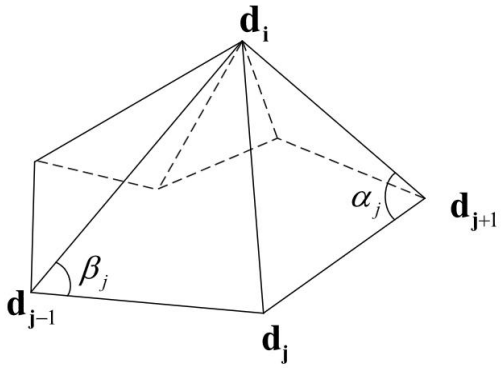


Figure 4.7: Neighborhood of vertex p_i [199].

$$k_i = max(\frac{\overline{h}_{max} - \overline{h}_i}{\overline{h}_{max} - \overline{h}_{min}}, \eta), \tag{4.9}$$

where $\overline{h}_i = log(h_i + 1)$ is logarithm of the mean curvature, $\overline{h}_{max}$ and $\overline{h}_{min}$ are maximum and minimum value of $\overline{h}_i$ and η is small threshold filter value. Points with higher k_i are extracted and those regions are refined by splitting them with new knots. Lin and Zhang [200] proposed progressive fitting method for large data sets. Fitting of data starts with initial cubic B-Spline patch. Then progressive fitting algorithm is applied for finding regions with higher error where new knots are inserted. Progressive fitting algorithm evaluates difference between T-Spline and data points. Lu et al. [201] proposed fast T-Spline fitting procedure base on work by Wang and Zheng. With every iteration, domain is divided in active and nonactive parts. Nonactive parts have satisfying error while active parts need to be refined. Therefore, algorithm only refines active parts with every iteration which reduces computational time. Three layer grid is defined for active parts which encompasses active mesh and parts of mesh which are influenced by

active mesh. This structure preserves refinement changes only in active parts of domain.
Wang et al. [202] proposed method for converting any quadrilateral mesh into T-Spline surface with C^2 continuity. In topology stage, T-mesh is generated from input mesh. In geometry stage, T-Spline surface is fitted from defined T-mesh.

4.3. LR B-Splines fitting

LR B-Splines fitting can be implemented in same manner as (T)HB-Splines or T-Splines, where initial mesh is iteratively refined until certain tolerance is achieved. Skytt et al. [203] proposed similar method, where initial LR B-Spline surface is iteratively refined in regions where distance between the points and the surface is not within prescribed tolerance. Approximation of PC with LR B-Spline is done in two ways: LSE with smoothing and multilevel B-Spline.
LSE global function is defined as

$$E = \alpha_1 \sum_{i=1}^{N} (F_i(u,v) - z_i)^2 + \alpha_2 f_s(F). \tag{4.10}$$

Smoothing term is given as

$$f_s(F) = \int\int_{\Omega} \int_0^{\pi} \sum_{i=1}^{3} w_i \left(\frac{\partial^i F(u_0 + r cos\phi, v_0 + r sin\phi)}{\partial r^i} \bigg|_{r=0} \right) d\phi du_0 dv_0. \tag{4.11}$$

Smoothing term approximates minimization of surface area, curvature and variation in curvature. Integration of given smoothing term is calculated by Gauss quadrature. Linear system is obtained and solved with conjugate gradient method. Let $p_c = (c_x, y_c, z_c)$, $c = 1, ..., C$ be points in the support of any given B-Spline basis function. Using multilevel B-Spline local approximation, control point of given B-Spline basis function is evaluated as

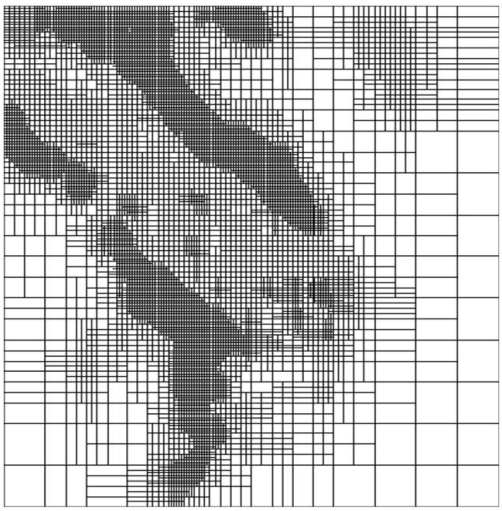
$$Q_i = \frac{\sum_c (s_i N_i(x_c, y_c))^2 \phi_c}{\sum_c (s_i N_i(x_c, y_c))^2}, \tag{4.12}$$

where s_i is scaling factor, $N_i(x_c, y_c)$ is basis function value for every c point and ϕ_c is evaluated for every data point as

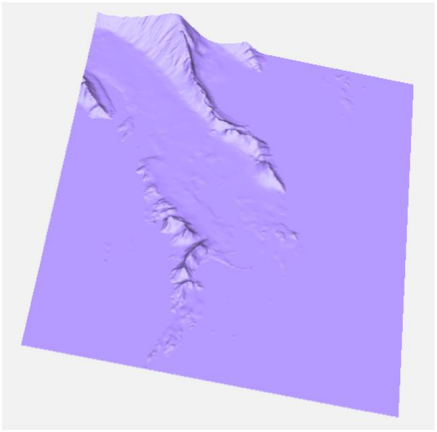
$$\phi_c = \frac{s_i N_i(x_c, y_c) z_c}{\sum_l s_l N_l(x_c, y_c)^2}. \tag{4.13}$$

The LR multilevel B-Spline approximation is local and has no effect outside support of given basis function. During fitting iterations, regions with higher error are split by inserting knot lines in selected direction where elements are split in the middle. Results of their work can be seen on Fig. 4.8. Skytt et al. [204] applied same fitting principle as above, but in this case for bathymetry data of obtained by sonar technologies. In this paper they added *deconfliction* filter, for cleaning generated LR B-Spline surface in case of overfitting. Kermarrec et al. [205] also applied LSE and LR multilevel B-Spline approximation method presented in this section

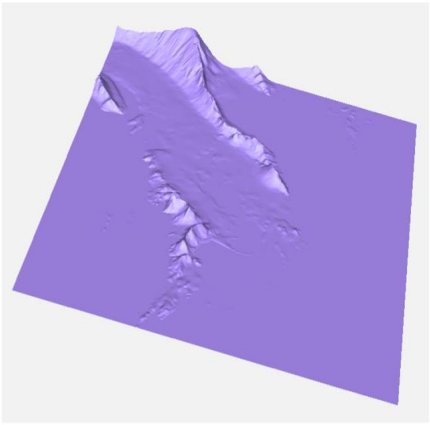
for fitting coastal regions obtained by *Terrestrial Laser Scanner*. More detailed description of fitting geographical data with LR B-Splines can be seen in [206].



(a) LR B-Spline mesh of Værøy island PC.



(b) LR B-Spline approximation of Værøy island using LR multilevel B-Spline approximation.



(c) LR B-Spline approximation of Værøy island using LSE approximation.

Figure 4.8: Results of fitting Værøy island PC (Værøy is an island in the Lofoten archipelago in Norway) [203].

5. Conclusion

The scope of this study deals with geometric parameterization and shape synthesis of data for CAD, CAM or CAE applications. Initial data is usually obtained by scanning technologies and are in need of exact shape paremeterization. These sets of data, notably in form of a PC can be enormous (depending on geometry complexity) and contain coordinate points without any relevant functionality. In order to gain any "sense" from these data sets, geometric parameterization is needed. When parameterized model is established, further modifications, numerical analysis or enhancements can be relatively easily applied in comparison to initial PC data set. This study presents models that can be used for parameterization along with numerical and optimization procedures required to achieve appropriate parametric model from initial PC.

Most notable CAD parametric curves and surfaces or models are described in this study. Standard models like Bèzier, B-Spline and NURBS are provide along with their properties. Starting from initial, global Bèziers model, B-Spline is general improvement of Bèzier in the sense of more local manipulation and the same goes for NURBS with their weights which can highlight desired areas. Fundamental geometric algorithms are also defined which are used for manipulating named models. Advanced, adaptive parametric models like (T)HB-Splines, T-Spline and LR B-Spline are also presented. They are derived from standard models like B-Spline and NURBS and they enable additional manipulation of given geometric topology through adaptive refinement of model shape parameters. Dozens more models can be found in literature, but these present CAD standards are most used ones for geometric parameterization or numerical analysis. Methods of obtaining representative parametric model from initial PC are defined and explained through fitting process. Before conducting steps of fitting, PC data have to be parameterized, usually on unit line $[0, 1]$ or unit square $[0, 1] \times [0, 1]$. This study presents methods for parameterization of input data as the first step of fitting process. Fitting is usually obtained with linear LSE system, but before establishing linear LSE system, certain optimization steps are required. This paper describes optimization methods required for evaluating parameters of every model and achieving solution with LSE. In some cases fitting is done without solving linear system, which is also mentioned throughout this paper.

This paper presents parametric curves and surfaces and optimization methods required to define CAD model via reverse engineering process. CAD representation of given topology is mostly first step of any CAD, CAM or CAE software, but as it can be seen it requires large number of complex and computationally expensive numerical and optimization algorithm "just" to obtain proper physical model.

REFERENCES

[1] A. Nguyen i B. Le, 3d point cloud segmentation: A survey, *IEEE Conference on Robotics, Automation and Mechatronics, RAM - Proceedings*, 225–230, 2013.

[2] X. F. Han, J. S. Jin, M. J. Wang, W. Jiang, L. Gao i L. Xiao, A review of algorithms for filtering the 3d point cloud, *Signal Processing: Image Communication*, 57, 103–112, 9 2017.

[3] M. Ćurković i D. Vučina, Adaptive representation of large 3d point clouds for shape optimization, 547–553, 2017.

[4] D. Vučina, M. Ćurković i T. Novković, Classification of 3d shape deviation using feature recognition operating on parameterization control points, *Computers in Industry*, 65, 1018–1031, 8 2014.

[5] P.-J. Laurent i P. Sablonnière, Pierre bézier: An engineer and a mathematician, *Computer Aided Geometric Design*, 18, 609–617, 9 2001.

[6] L. Piegl i W. Tiller, *The NURBS Book*, Springer Berlin Heidelberg, 1995.

[7] S. Bernstein, Démonstration du théorème de weierstrass fondée sur le calcul des probabilités, *Communications of the Kharkov Mathematical Societ*, Volume 13, 1–2, 1912.

[8] G. E. Farin i G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, Elsevier Science, 2002.

[9] L. Piegl, Key developments in computer-aided geometric design, *Computer-Aided Design*, 21, 262–273, 6 1989.

[10] W. Boehm i A. Müller, On de casteljau's algorithm, *Computer Aided Geometric Design*, 16, 587–605, 8 1999.

[11] M. G. Cox, The numerical evaluation of b-splines, 1972.

[12] C. de Boor, On calculating with b-splines, *Journal of Approximation Theory*, 6, 50–62, 7 1972.

[13] W. J. Gordon i R. F. Riesenfeld, *B-SPLINE CURVES AND SURFACES*, 95–126, Elsevier, 1974.

[14] C. de Boor, *A Practical Guide to Splines*, 27, Springer New York, 1978.

[15] W. Tiller, Rational b-splines for curve and surface representation, *IEEE Computer Graphics and Applications*, 3, 61–69, 9 1983.

[16] L. Piegl i W. Tiller, Curve and surface constructions using rational b-splines, *Computer-Aided Design*, 19, 485–498, 11 1987.

[17] K. J. Versprille, Computer-aided design applications of the rational b-spline approximation form, 1975.

[18] E. Cohen, T. Lyche i R. Riesenfeld, Discrete b-splines and subdivision techniques in computer-aided geometric design and computer graphics, *Computer Graphics and Image Processing*, 14, 87–111, 10 1980.

[19] W. Boehm, Inserting new knots into b-spline curves, *Computer-Aided Design*, 12, 199–201, 7 1980.

[20] T. Lyche, E. Cohen i K. Mørken, Knot line refinement algorithms for tensor product b-spline surfaces, *Computer Aided Geometric Design*, 2, 133–139, 9 1985.

[21] T. Lyche, C. Manni i H. Speleers, *Foundations of spline theory: B-splines, spline approximation, and hierarchical refinement*, 2219, 1–76, Springer Verlag, 2018.

[22] G. Kiss, Theory and algorithms for truncated hierarchical b-splines, 2015.

[23] T. Lyche i K. Morken, A data-reduction strategy for splines with applications to the approximation of functions and data, 1988.

[24] T. Lyche i K. Morken, Knot removal for parametric b-spline curves and surfaces, 1987.

[25] W. Tiller, Knot-removal algorithms for nurbs curves and surfaces, *Computer-Aided Design*, 24, 445–453, 8 1992.

[26] H. Prautzsch, Degree elevation of b-spline curves, *Computer Aided Geometric Design*, 1, 193–198, 11 1984.

[27] H. Prautzsch i B. Piper, A fast algorithm to raise the degree of spline curves, *Computer Aided Geometric Design*, 8, 253–265, 10 1991.

[28] L. Piegl i W. Tiller, Software-engineering approach to degree elevation of b-spline curves, *Computer-Aided Design*, 26, 17–28, 1 1994.

[29] L. Piegl i W. Tiller, Algorithm for degree reduction of b-spline curves, *Computer-Aided Design*, 27, 101–110, 2 1995.

[30] H. J. Wolters, G. Wu i G. Farin, Degree reduction of b-spline curves, *Geometric Modelling*, 235–241, 1998.

[31] J.-H. Yong, S.-M. Hu, J.-G. Sun i X.-Y. Tan, Degree reduction of b-spline curves, *Computer Aided Geometric Design*, 18, 117–127, 3 2001.

[32] D. R. Forsey i R. H. Bartels, Hierarchical b-spline refinement, *ACM SIGGRAPH Computer Graphics*, 22, 205–212, 8 1988.

[33] R. Kraft, *Hierarchical B-splines*, Citeseer, 1994.

[34] R. Kraft, Adaptive and linearly independent multilevel b- splines, *Surface Fitting and Multiresolution Methods 1St Ed.*, 209–218, 1997.

[35] C. Giannelli, B. Jüttler i H. Speleers, Strongly stable bases for adaptively refined multilevel spline spaces, *Advances in Computational Mathematics*, 40, 459–490, 4 2014.

[36] C. Giannelli i B. Jüttler, Bases and dimensions of bivariate hierarchical tensor-product splines, *Journal of Computational and Applied Mathematics*, 239, 162–178, 2 2013.

[37] A.-V. Vuong, C. Giannelli, B. Jüttler i B. Simeon, A hierarchical approach to adaptive local refinement in isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering*, 200, 3554–3567, 12 2011.

[38] D. Mokriš, B. Jüttler i C. Giannelli, On the completeness of hierarchical tensor-product b-splines, *Journal of Computational and Applied Mathematics*, 271, 53–70, 12 2014.

[39] D. Schillinger, L. Dedè, M. A. Scott, J. A. Evans, M. J. Borden, E. Rank i 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, 116–150, 12 2012.

[40] C. Giannelli, B. Jüttler i H. Speleers, Thb-splines: The truncated basis for hierarchical splines, *Computer Aided Geometric Design*, 29, 485–498, 10 2012.

[41] H. Speleers, P. Dierckx i S. Vandewalle, Quasi-hierarchical powell–sabin b-splines, *Computer Aided Geometric Design*, 26, 174–191, 2 2009.

[42] T. Hughes, J. Cottrell i Y. Bazilevs, Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement, *Computer Methods in Applied Mechanics and Engineering*, 194, 4135–4195, 10 2005.

[43] P. Bornemann i F. Cirak, A subdivision-based implementation of the hierarchical b-spline finite element method, *Computer Methods in Applied Mechanics and Engineering*, 253, 584–598, 1 2013.

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

[45] W. Liao, H. Liu i T. Li, *Splines and Subdivision*, 15–63, Springer Singapore, 2017.

[46] H. R. Atri i S. Shojaee, Meshfree truncated hierarchical refinement for isogeometric analysis, *Computational Mechanics*, 62, 1583–1597, 12 2018.

[47] A. Buffa i C. Giannelli, Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence, *Mathematical Models and Methods in Applied Sciences*, 26, 1–25, 1 2016.

[48] A. Buffa, C. Giannelli, P. Morgenstern i D. Peterseim, Complexity of hierarchical refinement for a class of admissible mesh configurations, *Computer Aided Geometric Design*, 47, 83–92, 10 2016.

[49] G. Kiss, C. Giannelli i B. Jüttler, Algorithms and data structures for truncated hierarchical b–splines, 304–323, Springer Berlin Heidelberg, 2014.

[50] T. Song, H. Liao i G. Subbarayan, Efficient local refinement near parametric boundaries using kd-tree data structure and algebraic level sets, *Algorithms*, 15, 245, 7 2022.

[51] T. W. Sederberg, J. Zheng, A. Bakenov i A. Nasri, T-splines and t-nurccs, *ACM Transactions on Graphics*, 22, 477–484, 7 2003.

[52] T. W. Sederberg, D. L. Cardon, G. T. Finnigan, N. S. North, J. Zheng i T. Lyche, T-spline simplification and local refinement, *ACM Transactions on Graphics*, 23, 276–283, 8 2004.

[53] T. W. Sederberg, G. T. Finnigan, X. Li, H. Lin i H. Ipson, Watertight trimmed nurbs, *ACM Transactions on Graphics*, 27, 1–8, 8 2008.

[54] A. Buffa, D. Cho i G. Sangalli, Linear independence of the t-spline blending functions associated with some particular t-meshes, *Computer Methods in Applied Mechanics and Engineering*, 199, 1437–1445, 4 2010.

[55] X. Li, J. Zheng, T. W. Sederberg, T. J. Hughes i M. A. Scott, On linear independence of t-spline blending functions, *Computer Aided Geometric Design*, 29, 63–76, 1 2012.

[56] M. Scott, X. Li, T. Sederberg i T. Hughes, Local refinement of analysis-suitable t-splines, *Computer Methods in Applied Mechanics and Engineering*, 213-216, 206–222, 3 2012.

[57] X. Li i M. A. Scott, On the nesting behavior of t-splines, 5 2011.

[58] C. Asche i V. Berkhahn, Efficient data structures for t-spline modeling, EG-ICE 2012 international workshop, 2012.

[59] W. Xiao, Y. Liu, R. Li, W. Wang, J. Zheng i G. Zhao, Reconsideration of t-spline data models and their exchanges using step, *Computer-Aided Design*, 79, 36–47, 10 2016.

[60] W. Wang, Y. Zhang, X. Du i G. Zhao, An efficient data structure for calculation of unstructured t-spline surfaces, *Visual Computing for Industry, Biomedicine, and Art*, 2, 2, 12 2019.

[61] H. Kang, F. Chen i J. Deng, Modified t-splines, *Computer Aided Geometric Design*, 30, 827–843, 12 2013.

[62] E. Evans, M. Scott, X. Li i D. Thomas, Hierarchical t-splines: Analysis-suitability, bézier extraction, and application as an adaptive basis for isogeometric analysis, *Computer Methods in Applied Mechanics and Engineering*, 284, 1–20, 2 2015.

[63] M. Scott, D. Thomas i E. Evans, Isogeometric spline forests, *Computer Methods in Applied Mechanics and Engineering*, 269, 222–264, 2 2014.

[64] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott i T. Sederberg, Isogeometric analysis using t-splines, *Computer Methods in Applied Mechanics and Engineering*, 199, 229–263, 1 2010.

[65] L. B. da Veiga, A. Buffa, G. Sangalli i R. Vazquez, Analysis-suitable t-splines of arbitrary degree: Definition, linear independence and approximation properties, *Mathematical Models and Methods in Applied Sciences*, 23, 1979–2003, 10 2013.

[66] T. Dokken, T. Lyche i K. F. Pettersen, Polynomial splines over locally refined box-partitions, *Computer Aided Geometric Design*, 30, 331–356, 3 2013.

[67] K. A. Johannessen, T. Kvamsdal i T. Dokken, Isogeometric analysis using lr b-splines, *Computer Methods in Applied Mechanics and Engineering*, 269, 471–514, 2 2014.

[68] A. Bressan, Some properties of lr-splines, *Computer Aided Geometric Design*, 30, 778–794, 11 2013.

[69] C. Bracco, T. Lyche, C. Manni, F. Roman i H. Speleers, Generalized spline spaces over t-meshes: Dimension formula and locally refined generalized b-splines, *Applied Mathematics and Computation*, 272, 187–198, 1 2016.

[70] J. Gu, T. Yu, L. V. Lich, T.-T. Nguyen i T. Q. Bui, Adaptive multi-patch isogeometric analysis based on locally refined b-splines, *Computer Methods in Applied Mechanics and Engineering*, 339, 704–738, 9 2018.

[71] C. Zimmermann i R. A. Sauer, Adaptive local surface refinement based on lr nurbs and its application to contact, *Computational Mechanics*, 60, 1011–1031, 12 2017.

[72] L. Chen i R. de Borst, Locally refined t-splines, *International Journal for Numerical Methods in Engineering*, 114, 637–659, 5 2018.

[73] K. A. Johannessen, F. Remonato i T. Kvamsdal, On the similarities and differences between classical hierarchical, truncated hierarchical and lr b-splines, *Computer Methods in Applied Mechanics and Engineering*, 291, 64–101, 7 2015.

[74] S.-J. Ahn, Geometric fitting of parametric curves and surfaces, *Journal of Information Processing Systems*, 4, 153–158, 12 2008.

[75] P. Dierckx, An algorithm for smoothing, differentiation and integration of experimental data using spline functions, *Journal of Computational and Applied Mathematics*, 1, 165–184, 9 1975.

[76] P. Dierckx, An algorithm for least-squares fitting of cubic spline surfaces to functions on a rectilinear mesh over a rectangle, *Journal of Computational and Applied Mathematics*, 3, 113–129, 1977.

[77] H. Engels, A least squares method for estimation of bezier curves and surfaces and its applicability to multivariate analysis, *Mathematical Biosciences*, 79, 155–170, 6 1986.

[78] Åke Björck, *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics, 1 1996.

[79] T. A. Pastva, C. F. Borges i R. Franke, Bezier curve fitting, 1998.

[80] J. Hoschek, Intrinsic parametrization for approximation, *Computer Aided Geometric Design*, 5, 27–31, 6 1988.

[81] Z. C. Li, C. Y. Suen, T. D. Bui i Q. L. Gu, Harmonic models of shape transformations in digital images and patterns, *CVGIP: Graphical Models and Image Processing*, 54, 198–209, 5 1992.

[82] J. Maillot, H. Yahiaz i A. Verroustz, Interactive texture mapping, *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1993*, 27–34, 9 1993.

[83] J. E. Marsden i T. J. R. Hughes, *Mathematical foundations of elasticity*, Dover, 1994.

[84] M. S. Floater, Parametrization and smooth approximation of surface triangulations, *Computer Aided Geometric Design*, 14, 231–250, 4 1997.

[85] M. S. Floater i M. Reimers, Meshless parameterization and surface reconstruction, *Computer Aided Geometric Design*, 18, 77–92, 3 2001.

[86] M. S. Floater, Mean value coordinates, *Computer Aided Geometric Design*, 20, 19–27, 3 2003.

[87] W. Ma i J. P. Kruth, Parameterization of randomly measured points for least squares fitting of b-spline curves and surfaces, *Computer-Aided Design*, 27, 663–675, 9 1995.

[88] J. F. Remacle, C. Geuzaine, G. Compère i E. Marchandise, High-quality surface remeshing using harmonic maps, *International Journal for Numerical Methods in Engineering*, 83, 403–425, 7 2010.

[89] G. Greiner i K. Hormann, Interpolating and approximating scattered 3d-data with hierarchical tensor product b-splines, 1997.

[90] I. Marinić-Kragić, M. Ćurković i D. Vučina, Adaptive re-parameterization based on arbitrary scalar fields for shape optimization and surface fitting, *Engineering Applications of Artificial Intelligence*, 67, 39–51, 1 2018.

[91] Y. K. Lai, S. M. Hu i H. Pottmann, Surface fitting based on a feature sensitive parametrization, *Computer-Aided Design*, 38, 800–807, 7 2006.

[92] M. P. do. Carmo, *Differential geometry of curves and surfaces*, Prentice-Hall, 1st edn., 1976.

[93] F. S. Cohen, W. Ibrahim i C. Pintavirooj, Ordering and parameterizing scattered 3d data for b-spline surface approximation, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22, 642–648, 6 2000.

[94] R. Menikoff i C. Zemach, Methods for numerical conformal mapping, *Journal of Computational Physics*, 36, 366–410, 7 1980.

[95] P. J. Olver, Complex analysis and conformal mapping, *University of Minnesota*, 2017.

[96] M. Jin, J. Kim, F. Luo, S. Lee i X. Gu, Conformal surface parameterization using euclidean ricci flow, *Technique report, May*, 2006.

[97] R. S. Hamilton, *The Ricci flow on surfaces*, 237–262, 1988.

[98] K. Su, L. Cui, K. Qian, N. Lei, J. Zhang, M. Zhang i X. D. Gu, Area-preserving mesh parameterization for poly-annulus surfaces based on optimal mass transportation, *Computer Aided Geometric Design*, 46, 76–91, 8 2016.

[99] K. Su, W. Chen, N. Lei, J. Zhang, K. Qian i X. Gu, Volume preserving mesh parameterization based on optimal mass transportation, *Computer-Aided Design*, 82, 42–56, 1 2017.

[100] M. Ćurković, I. Marinić-Kragić i D. Vučina, A novel projection of open geometry into rectangular domain for 3d shape parameterization, *Integrated Computer-Aided Engineering*, 25, 1–14, 1 2018.

[101] Z. Zhu, A. Iglesias, L. You i J. J. Zhang, A review of 3d point clouds parameterization methods, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13352 LNCS, 690–703, 2022.

[102] K. Gurney, *An Introduction to Neural Networks*, CRC Press, 10 2018.

[103] F. Scholz i B. Jüttler, Parameterization for polynomial curve approximation via residual deep neural networks, *Computer Aided Geometric Design*, 85, 101977, 2 2021.

[104] C. Giannelli, S. Imperatore, A. Mantzaflaris i F. Scholz, Learning meshless parameterization with graph convolutional neural networks, *Lecture Notes in Networks and Systems*, 817, 375–387, 2024.

[105] D. Ríos, F. Scholz i B. Jüttler, Quadratic surface preserving parameterization of unorganized point data, *Computer Aided Geometric Design*, 110, 102287, 5 2024.

[106] C. F. Borges i T. Pastva, Total least squares fitting of bézier and b-spline curves to ordered data, *Computer Aided Geometric Design*, 19, 275–289, 4 2002.

[107] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Longman Publishing Co., Inc., 1st edn., 1989.

[108] R. Eberhart i J. Kennedy, New optimizer using particle swarm theory, *Proceedings of the International Symposium on Micro Machine and Human Science*, 39–43, 1995.

[109] L. N. D. Castro i F. J. V. Zuben, Learning and optimization using the clonal selection principle, *IEEE Transactions on Evolutionary Computation*, 6, 239–251, 6 2002.

[110] X. S. Yang i S. Deb, Cuckoo search via lévy flights, *2009 World Congress on Nature and Biologically Inspired Computing, NABIC 2009 - Proceedings*, 210–214, 2009.

[111] X. S. Yang, Firefly algorithm, stochastic test functions and design optimization, *International Journal of Bio-Inspired Computation*, 2, 78–84, 2010.

[112] X. S. Yang, A new metaheuristic bat-inspired algorithm, *Studies in Computational Intelligence*, 284, 65–74, 2010.

[113] X. S. Yang i A. H. Gandomi, Bat algorithm: A novel approach for global engineering optimization, *Engineering Computations (Swansea, Wales)*, 29, 464–483, 2012.

[114] E. Castillo, A. Cobo, R. Gómez-Nesterkin i A. Hadi, A general framework for functional networks, *Networks*, 35, 70–82, 5 2000.

[115] A. Gálvez, A. Iglesias, A. Cobo, J. Puig-Pey i J. Espinola, *Bézier Curve and Surface Fitting of 3D Point Clouds Through Genetic Algorithms, Functional Networks and Least-Squares Approximation*, 680–693, Springer Berlin Heidelberg, 2007.

[116] A. Gálvez, A. Cobo, J. Puig-Pey i A. Iglesias, *Particle Swarm Optimization for Bézier Surface Reconstruction*, 116–125, 2008.

[117] A. Iglesias, A. Gálvez i A. Avila, Discrete bézier curve fitting with artificial immune systems, *Studies in Computational Intelligence*, 441, 59–75, 2013.

[118] A. Gálvez i A. Iglesias, Firefly algorithm for polynomial bézier surface parameterization, *Journal of Applied Mathematics*, 2013, 2013.

[119] A. Gálvez, A. Iglesias i L. Cabellos, Cuckoo search with lévy flights for weighted bayesian energy functional optimization in global-support curve data fitting, *Scientific World Journal*, 2014, 2014.

[120] A. Iglesias, A. Galvez i M. Collantes, Bat algorithm for curve parameterization in data fitting with polynomial bézier curves, 107–114, IEEE, 10 2015.

[121] P. Pandunata i S. M. H. Shamsuddin, Differential evolution optimization for bezier curve fitting, 68–72, IEEE, 8 2010.

[122] R. Storn i K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *Journal of Global Optimization*, 11, 341–359, 1997.

[123] S. Kirkpatrick, C. D. Gelatt i M. P. Vecchi, Optimization by simulated annealing, *Science*, 220, 671–680, 1983.

[124] E. K. Ueda, M. de Sales Guerra Tsuzuki, R. Y. Takimoto, A. K. Sato, T. de Castro Martins, P. E. Miyagi i R. S. U. Rosso, Piecewise bézier curve fitting by multiobjective simulated annealing, *IFAC-PapersOnLine*, 49, 49–54, 2016.

[125] J. Lifton, T. Liu i J. McBride, Non-linear least squares fitting of bézier surfaces to unstructured point clouds, *AIMS Mathematics*, 6, 3142–3159, 2021.

[126] A. Iglesias, A. Gálvez i C. Loucera, Two simulated annealing optimization schemas for rational bézier curve fitting in the presence of noise, *Mathematical Problems in Engineering*, 2016, 2016.

[127] S. G, "estimating the dimension of a model.", *The Annals of Statistics*, 6, 461 – 464, 1978.

[128] M. A. Zaman i S. Chowdhury, Modified bézier curves with shape-preserving characteristics using differential evolution optimization algorithm, *Advances in Numerical Analysis*, 2013, 1–8, 3 2013.

[129] L. Yang i X. M. Zeng, Bezier curves and surfaces with shape parameters, *International Journal of Computer Mathematics*, 86, 1253–1263, 7 2009.

[130] E. K. Ueda, A. K. Sato, T. C. Martins, R. Y. Takimoto, R. S. U. Rosso i M. S. G. Tsuzuki, Curve approximation by adaptive neighborhood simulated annealing and piecewise bézier curves, *Soft Computing*, 24, 18821–18839, 12 2020.

[131] J. Lu, X. Su, J. Zhong i G. Hu, Multi-objective shape optimization of developable bézier-like surfaces using non-dominated sorting genetic algorithm, *Mechanics and Industry*, 24, 11 2023.

[132] S. Zain, Y. Misro i K. Miura, Curve fitting using generalized fractional bézier curve, 77–82, CAD Solutions LLC, 6 2022.

[133] D. Vucina, Z. Lozina i I. Pehnec, Computational procedure for optimum shape design based on chained bezier surfaces parameterization, *Engineering Applications of Artificial Intelligence*, 25, 648–667, 4 2012.

[134] M. Fortes i E. Medina, Fitting missing data by means of adaptive meshes of bézier curves, *Mathematics and Computers in Simulation*, 191, 33–48, 1 2022.

[135] X. Cui, Y. Li i L. Xu, Adaptive extension fitting scheme: An effective curve approximation method using piecewise bézier technology, *IEEE Access*, 11, 58422–58435, 2023.

[136] M. B. Loknar, G. Klančar i S. Blažič, Minimum-time trajectory generation for wheeled mobile systems using bézier curves with constraints on velocity, acceleration and jerk, *Sensors*, 23, 2 2023.

[137] P. Dierckx, *Curve and Surface Fitting with Splines*, Oxford University Press, 1995.

[138] Z. Huang i F. S. Cohen, Affine-invariant b-spline moments for curve matching, *IEEE Transactions on Image Processing*, 5, 1473–1480, 1996.

[139] H. Park i J. H. Lee, B-spline curve fitting based on adaptive curve refinement using dominant points, *Computer-Aided Design*, 39, 439–451, 6 2007.

[140] L. A. Piegl i W. Tiller, Surface approximation to scanned data, *Visual Computer*, 16, 386–395, 2000.

[141] W. Li, S. Xu, G. Zhao i L. P. Goh, Adaptive knot placement in b-spline curve approximation, *Computer-Aided Design*, 37, 791–797, 7 2005.

[142] V. Weiss, L. Andor, G. Renner i T. Várady, Advanced surface fitting techniques, *Computer Aided Geometric Design*, 19, 19–42, 1 2002.

[143] D. L. B. Jupp, Approximation to data by splines with free knots, *SIAM Journal on Numerical Analysis*, 15, 328–343, 1978.

[144] P. D. Loach i A. J. Wathen, On the best least squares approximation of continuous functions using linear splines with free knots, *IMA Journal of Numerical Analysis*, 11, 393–409, 7 1991.

[145] H. Kang, F. Chen, Y. Li, J. Deng i Z. Yang, Knot calculation for spline fitting via sparse optimization, *Computer-Aided Design*, 58, 179–188, 1 2015.

[146] W. V. Loock, G. Pipeleers, J. D. Schutter i J. Swevers, A convex optimization approach to curve fitting with b-splines, *IFAC Proceedings Volumes*, 44, 2290–2295, 1 2011.

[147] Y. Yuan, N. Chen i S. Zhou, Adaptive b-spline knot selection using multi-resolution basis set, *IIE Transactions*, 45, 1263–1277, 12 2013.

[148] F. Yoshimoto, T. Harada i Y. Yoshimoto, Data fitting with a spline using a real-coded genetic algorithm, *Computer-Aided Design*, 35, 751–760, 7 2003.

[149] H. Akaike, A new look at the statistical model identification, *IEEE Transactions on Automatic Control*, 19, 716–723, 1974.

[150] T.-H. Le, D.-J. Kim, K.-C. Min i S.-W. Pyo, B-spline surface fitting using genetic algorithm, *Journal of the Society of Naval Architects of Korea, Volume 46, No. 1*, 2009.

[151] G. S. Kumar, P. K. Kalra i S. G. Dhande, Parameter optimization for b-spline curve fitting using genetic algorithms, *2003 Congress on Evolutionary Computation, CEC 2003 - Proceedings*, 3, 1871–1878, 2003.

[152] C. H. Garcia-Capulin, F. J. Cuevas, G. Trejo-Caballero i H. Rostro-Gonzalez, A hierarchical genetic algorithm approach for curve fitting with b-splines, *Genetic Programming and Evolvable Machines*, 16, 151–166, 4 2015.

[153] A. Iglesias i A. Gálvez, Applying functional networks to fit data points from b-spline surfaces, *Proceedings of Computer Graphics International Conference, CGI*, 329–332, 2001.

[154] A. Iglesias, G. Echevarría i A. Gálvez, Functional networks for b-spline surface reconstruction, *Future Generation Computer Systems*, 20, 1337–1353, 11 2004.

[155] A. Gálvez i A. Iglesias, Efficient particle swarm optimization approach for data fitting with free knot b-splines, *Computer-Aided Design*, 43, 1683–1692, 12 2011.

[156] A. Gálvez, A. Iglesias i J. Puig-Pey, Iterative two-step genetic-algorithm-based method for efficient polynomial b-spline surface reconstruction, *Information Sciences*, 182, 56–76, 1 2012.

[157] A. Gálvez i A. Iglesias, Firefly algorithm for explicit b-spline curve fitting to data points, *Mathematical Problems in Engineering*, 2013, 2013.

[158] W. Wang, H. Pottmann i Y. Liu, Fitting b-spline curves to point clouds by curvature-based squared distance minimization, *ACM Transactions on Graphics*, 25, 214–238, 4 2006.

[159] I. Kovács i T. Várady, Constrained fitting with free-form curves and surfaces, *Computer-Aided Design*, 122, 102816, 5 2020.

[160] H. P. Moreton i C. H. Séquin, Functional optimization for fair surface design, 167–176, 7 1992.

[161] G. Greiner, J. Loos i W. Wesselink, Data dependent thin plate energy and its use in interactive surface modeling, *Computer Graphics Forum*, 15, 175–185, 8 1996.

[162] M. Cúrkovic, D. Vučina i A. Cúrkovic, Enhanced 3d parameterization for integrated shape synthesis by fitting parameter values to point sets, *Integrated Computer-Aided Engineering*, 24, 241–260, 1 2017.

[163] M. Ćurković, A. Ćurković i D. Vučina, Novel re-parameterization for shape optimization and comparison with knot-based gradient fitting method, *Computer Methods in Applied Mechanics and Engineering*, 336, 304–332, 7 2018.

[164] I. Marinić-Kragić i D. Vučina, Convergence and remeshing criteria for fitting method based on iterative reparameterization via plane-stress model, *Computer-Aided Design and Applications*, 18, 1000–1017, 2021.

[165] A. Ebrahimi i G. B. Loghmani, Shape modeling based on specifying the initial b-spline curve and scaled bfgs optimization method, *Multimedia Tools and Applications*, 77, 30331–30351, 12 2018.

[166] Y. X. Yuan, A modified bfgs algorithm for unconstrained optimization, *IMA Journal of Numerical Analysis*, 11, 325–332, 7 1991.

[167] M. J. Milroy, C. Bradley, G. W. Vickers i D. J. Weir, G1 continuity of b-spline surface patches in reverse engineering, *Computer-Aided Design*, 27, 471–478, 6 1995.

[168] L. Y. Zhang, R. R. Zhou, J. Y. Zhu i X. Wu, Piecewise b-spline surfaces fitting to arbitrary triangle meshes, *CIRP Annals*, 51, 131–134, 1 2002.

[169] P. Laube, M. O. Franz i G. Umlauf, Deep learning parametrization for b-spline curve approximation, *Proceedings - 2018 International Conference on 3D Vision, 3DV 2018*, 691–699, 10 2018.

[170] P. Laube, M. O. Franz i G. Umlauf, Learnt knot placement in b-spline curve approximation using support vector machines, *Computer Aided Geometric Design*, 62, 104–116, 5 2018.

[171] Z. Wen, J. Luo i H. Kang, The deep neural network solver for b-spline approximation, *Computer-Aided Design*, 169, 103668, 4 2024.

[172] M. Saillot, D. Michel i A. Zidna, B-spline curve approximation with transformer neural networks, *Mathematics and Computers in Simulation*, 223, 275–287, 9 2024.

[173] P. Laurent-Gengoux i M. Mekhilef, Optimization of a nurbs representation, *Computer-Aided Design*, 25, 699–710, 11 1993.

[174] N. Carlson, Nurbs surface fitting with gauss-newton, 2009.

[175] W. Ma i J. P. Kruth, Nurbs curve and surface fitting for reverse engineering, *International Journal of Advanced Manufacturing Technology*, 14, 918–927, 1998.

[176] W. Heidrich, R. Bartels i G. Labahn, Fitting uncertain data with nurbs, 1997.

[177] T. R. Wang, N. Liu, L. Yuan, K. X. Wang i X. J. Sheng, Iterative least square optimization for the weights of nurbs curve, *Mathematical Problems in Engineering*, 2022, 2022.

[178] A. Dimitrov, R. Gu i M. Golparvar-Fard, Non-uniform b-spline surface fitting from unordered 3d point clouds for as-built modeling, *Computer-Aided Civil and Infrastructure Engineering*, 31, 483–498, 7 2016.

[179] A. Gálvez i A. Iglesias, Particle swarm optimization for non-uniform rational b-spline surface reconstruction from clouds of 3d data points, *Information Sciences*, 192, 174–192, 6 2012.

[180] E. Ülker i E. Ulker, Nurbs curve fitting using artificial immune system, *International Journal of Innovative Computing, Information and Control ICIC International c*, 8, 2875–2887, 2012.

[181] G. Costa, M. Montemurro i J. Pailhès, A general hybrid optimization strategy for curve fitting in the non-uniform rational basis spline framework, *Journal of Optimization Theory and Applications*, 176, 225–251, 1 2018.

[182] I. K. Park, I. D. Yun i S. U. Lee, Constructing nurbs surface model from scattered and unorganized range data, *Proceedings - 2nd International Conference on 3-D Digital Imaging and Modeling, 3DIM 1999*, 312–320, 1999.

[183] T. Elmidany, A. Elkeran, A. Galal i M. Elkhateeb, Nurbs surface reconstruction using rational b-spline neural networks, *Journal of Control Engineering and Technology (JCET) JCET*, 1, 34–38, 2011.

[184] T. To, Using rational b-spline neural networks for curve approximation, *Proceedings of the 7th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering, World Scientific and Engineering Academy and Society (WSEAS)*, 2009.

[185] X. Tian, L. Kong, D. Kong, L. Yuan i D. Kong, An improved method for nurbs surface based on particle swarm optimization bp neural network, *IEEE Access*, 8, 184656–184663, 2020.

[186] H. Speleers, Hierarchical spline spaces: quasi-interpolants and local approximation estimates, *Advances in Computational Mathematics*, 43, 235–255, 4 2017.

[187] D. R. Forsey i R. H. Bartels, Surface fitting with hierarchical splines, *ACM Transactions on Graphics (TOG)*, 14, 134–161, 1 1995.

[188] S. Lee, G. Wolberg i S. Y. Shin, Scattered data interpolation with multilevel b-splines, *IEEE Transactions on Visualization and Computer Graphics*, 3, 228–244, 1997.

[189] W. Zhang, Z. Tang i J. Li, Adaptive hierarchical b-spline surface approximation of large-scale scattered data, *Proceedings - Pacific Conference on Computer Graphics and Applications*, 8–16, 1998.

[190] G. Kiss, C. Giannelli, U. Zore, B. Jüttler, D. Großmann i J. Barner, Adaptive cad model (re-)construction with thb-splines, *Graphical Models*, 76, 273–288, 9 2014.

[191] H. Speleers i C. Manni, Effortless quasi-interpolation in hierarchical spaces, *Numerische Mathematik*, 132, 155–184, 1 2016.

[192] C. Bracco, C. Giannelli, D. Großmann, S. Imperatore, D. Mokriš i A. Sestini, *THB-Spline Approximations for Turbine Blade Design with Local B-Spline Approximations*, 63–82, 2022.

[193] A. Buffa i E. M. Garau, Refinable spaces and local approximation estimates for hierarchical splines, *IMA Journal of Numerical Analysis*, 37, 1125–1149, 7 2017.

[194] C. Bracco, C. Giannelli i A. Sestini, Adaptive scattered data fitting by extension of local approximations to hierarchical splines, *Computer Aided Geometric Design*, 52-53, 90–105, 3 2017.

[195] C. Bracco, C. Giannelli, D. Großmann i A. Sestini, Adaptive fitting with thb-splines: Error analysis and industrial applications, *Computer Aided Geometric Design*, 62, 239–252, 5 2018.

[196] A. Giust, B. Jüttler i A. Mantzaflaris, Local (t)hb-spline projectors via restricted hierarchical spline fitting, *Computer Aided Geometric Design*, 80, 101865, 6 2020.

[197] J. Zheng, Y. Wang i H. S. Seah, Adaptive t-spline surface fitting to z-map models, *Proceedings - GRAPHITE 2005 - 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia*, 405–411, 2005.

[198] Y. Wang i J. Zheng, Adaptive t-spline surface approximation of triangular meshes, *2007 6th International Conference on Information, Communications and Signal Processing, ICICS*, 2007.

[199] Y. Wang i J. Zheng, Curvature-guided adaptive t-spline surface fitting, *Computer-Aided Design*, 45, 1095–1107, 8 2013.

[200] H. Lin i Z. Zhang, An efficient method for fitting large data sets using t-splines, *https://doi.org/10.1137/120888569*, 35, 12 2013.

[201] Z. Lu, X. Jiang, G. Huo, D. Ye, B. Wang i Z. Zheng, A fast t-spline fitting method based on efficient region segmentation, *Computational and Applied Mathematics*, 39, 1–19, 5 2020.

[202] W. Wang, Y. Zhang, M. A. Scott i T. J. R. Hughes, Converting an unstructured quadrilateral mesh to a standard t-spline surface, *Computational Mechanics*, 48, 477–498, 10 2011.

[203] V. Skytt, O. Barrowclough i T. Dokken, Locally refined spline surfaces for representation of terrain data, *Computers Graphics*, 49, 58–68, 6 2015.

[204] V. Skytt, Q. Harpham, T. Dokken i H. E. Dahl, Deconfliction and surface generation from bathymetry data using lr b-splines, *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 10521 LNCS, 270–295, 2017.

[205] G. Kermarrec, V. Skytt i T. Dokken, Surface approximation of coastal regions: Lr b-spline for detection of deformation pattern, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, V-2-2022, 119–126, 5 2022.

[206] G. Kermarrec, V. Skytt i T. Dokken, *Optimal Surface Fitting of Point Clouds Using Local Refinement*, Springer International Publishing, 2023.

Abbrevations

CAD	Computer-Aided Design
CAM	Computer-Aided Manufacturing
CAE	Computer-Aided Engineering
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional
PC	Point Cloud
FEA	Finite Element Analysis
CFD	Computational Fluid Dynamics
MBD	Multi-Body Dynamics
PC	Computational Fluid Dynamics
PoU	Partition of Unity
HB	Hierarchical Basis
THB	Truncated Hierarchical Basis
LR	Locally Refined
LR B-Splines	Locally Refined B-Splines
PB-Splines	Point Based Splines
IGA	Isogeometric Analysis
HASTS	Hierarchical Analysis Suitable T-Splines
LSE	Least-Squares Error
RMSE	Root-Mean Squared Error
NN	Neural Network
GA	Genetic Algorithm
FA	Firefly Algorithm
BA	Bat Algorithm
PSO	Particle Swarm Optimization
AIS	Artificial Immune Systems
CSA	Clone Selection Algorithm
DE	Differential Evolution
SA	Simulated Annealing
BIC	Bayesian information criterion
NSGA-II	elitist Non-dominated Sorting Genetic Algorithm
ICL	Inverse Chord Length
AIC	Akaike information criterion
RBNN	Rational B-Spline Neural Network
QI	Quasi-Interpolation

Abstract

Geometric parameterization and shape synthesis is a crucial process in establishing an appropriate physical model for applications in CAD, CAM, or CAE tools. With increasingly complex engineering objects, the process of reverse engineering itself becomes more demanding and complex. The point cloud of a given object can contain a very large set of points that need to be parameterized with an appropriate mathematical model. Due to their robustness and wide applicability, parametric curves and surfaces i.e., parametric models such as B-Spline and NURBS present themselves as very useful tools in geometry parameterization. This paper presents and mathematically describes the most commonly used standard parametric curves and surfaces applied in engineering analyses, and their role in the process of geometric parameterization and shape synthesis. Appropriate optimization algorithms and numerical procedures used in establishing a suitable model are also described. With growing demands for fast and precise geometry approximation, adaptive parametric models have been developed that achieve local improvements of the selected model and accelerate the parameterization process itself. Such parametric models as T-Spline and THB-Spline are also presented in this paper, as well as optimization procedures used in establishing such parametric model. The aim of this paper is to present complex mathematical parametric curves and surfaces and the corresponding parameterization and optimization procedures with the goal of establishing an appropriate CAD model, which is as exact as possible to the original topology and thus can find its application in further numerical tools and analyses.

Sažetak

Geometrijska parametrizacija i sinteza oblika ključan je postupak prilikom uspostave primjerenog fizičkog modela za primjene u CAD, CAM ili CAE alatima. Sa sve kompleksnijim inženjerskim objektima, sam postupak reverzibilnog inženjerstva postaje zahtjevniji i kompleksniji. Oblak točaka danog objekta može sadržavati jako veliki skup točaka koje je potrebno parametrizirati s primjerenim matematičkim modelom. Zbog svoje robusnosti i velike primjenjivosti, parametarske krivulje i plohe tj., parametarski modeli kao što su B-Spline i NURBS predstavljaju se kao vrlo koristan alat prilikom parametrizacije geometrije. U radu su predstavljene i matematički opisane najkorištenije, standardne parametarske krivulje i plohe primijenjene u inženjerskim analizama, te njihova uloga prilikom postupka geometrijske parametrizacije i sinteze oblika. Odgovarajući algoritmi optimizacije i numerički postupci korišteni prilikom uspostave odgovarajućeg modela također su opisani. S rastućim zahtjevima za brzu i preciznu aproksimaciju geometrije razvijeni su i adaptivni parametarski modeli koji postižu lokalna poboljšanja odabranog modela i ubrzavaju sam postupak parametrizacije. Takvi parametarski modeli kao što su T-Spline i THB-Spline također su predstavljeni u ovom radu, kao i optimizacijski postupci koji se koriste prilikom uspostave takvog parametarskog modela. Cilj ovoga rada je prikazati složene matematičke parametarske krivulje i plohe te odgovarajuće postupke parametrizacije i optimizacije s ciljem uspostave primjerenog CAD modela, koji je što više egzaktan izvornoj topologiji i time može naći svoju primjenu u daljnjim numeričkim alatima i analizama.