Expo-Rational B-Splines in Geometric Modeling

Methods for Computer Aided Geometric Design

Rune Dalmo

Preface

This thesis has been submitted as a partial fulfillment of the requirements for the degree of Philosophiae Doctor (Ph.D.) at the University of Oslo. The work has been conducted at Narvik University College in the R&D group Simulations, which is associated with the Centre of Mathematics for Applications at the University of Oslo. The research has been carried out through the years 2011 to 2015 under the supervision of Professor Arne Lakså and co-supervision by Professor Knut Mørken.

The thesis includes a collection of articles which resulted from cooperative work with colleagues in the research group. They can be found as individual chapters in the second and third parts of this text. The first chapter of part one provides background information, objectives of the work and ties together the individual research papers. Then follows a chapter which enumerates the individual contributions of the included articles and provides some remarks and notes for future work.

Acknowledgments

A number of people have supported me during the work which has culminated into this dissertation. First and foremost, I would like to thank my main supervisor Arne Lakså. His deep enthusiasm for the field of research, sharing of knowledge in many aspects, support and guidance during the period of the project have been invaluable. My co-supervisor, Knut Mørken, has taken care of many practical and organizational arrangements at the UiO. He has provided constructive feedback which was very useful in order to compile a final version of this text.

Then I would like to thank Børre Bang for his patience in discussions of theoretical and practical matters, sharing of his ideas and for always taking the time to provide constructive feedback and advice. Lubomir Dechevsky has prepared the ground by initiating and developing the theory of expo-rational B-splines and other topics. He has provided valuable feedback at several occasions. Klas Pettersson has been supportive through providing his expertise in mathematical theory and analysis, and by coordinating a series of interesting seminars on the topic of differential geometry. Peter Zanaty has put his mark by performing efficient and thorough work, which undoubtedly has influenced the way I work.

My dear friend and colleague Jostein Bratlie has spent what seems like an uncountable number of hours together with me. Together we have been learning, researching, programming, writing, traveling and teaching, just to mention a few, throughout the period of the research project.

My friends and colleagues at Narvik University College are always positive and provide a great working atmosphere. They are an interesting group of people with knowledge in many fields. The open door policy is much appreciated.

This thesis would not have been realized without the funding from the Verdikt program (project id 201511) by the Research Council of Norway.

I would like to thank my friends and family for the understanding and patience during long working hours. My parents, Randi and Magne, and my parents in law, Solveig and Ole, have taken care of my family while I have been away traveling to attend research conferences. My daughters, Evine and Alvilde, have been dragging me out of “deep hack mode” to remind me of the most important aspects of life.

Last, but not least, a big warm hug to my wife and best friend Therese for showing patience and for putting up with me during this period.

Abstract

Computer aided geometric design is the design of geometric shapes, such as curves and surfaces, by using computer technology. The applications range from product design and computer aided manufacturing to virtual worlds and computer games. Some of the most common representations of curves and surfaces for this purpose are polynomial patches of Bézier type and piecewise polynomials on B-spline form.

Real world models are sometimes represented by discrete data, scattered points or polygonal meshes. This thesis considers the use of expo-rational B-splines, a blending type spline construction where the coefficients are local functions that are blended together by infinitely smooth basis functions, as an alternative for the representation of discrete and continuous data. Various applied modeling and reconstruction problems are investigated.

A generalization of Bernstein factor matrices, suitable for computation of multivariate Bernstein polynomials, is presented. Results regarding the structure of the factors are given, and the factorization is related to mixed directional derivatives of a B-form, and to the de Casteljau algorithm for evaluating curves and surfaces on Bernstein-Bézier form.

Knot insertion is a fundamental operation for splines. It is used to express a spline in a more refined or flexible way without changing the shape of the spline. A knot insertion rule for ERBS curves, where new local functions are computed according to the inserted knot, is presented.

Contents

Part I

Introduction

1 Background

1.1 A brief history of interpolation

The history of interpolation can be traced back to ancient times. Astronomers in Uruk and Babylon used cuneiform tablets for calendar computations as early as the last three centuries BC [91]. Not only linear interpolation, but more complex schemes with higher order interpolation formulae were most likely developed [94]. The Indian astronomer-mathematician Brahmagupta proposed a method for interpolation, which is a special case of the more general Newton-Stirling interpolation formula, around 628AD. He later described a more general method for interpolation of unequal interval data [64]. Parabolic interpolation schemes are also found in Arabic and Persian sources from the 11th and 15th century [72], possibly influenced by the work of Indian origin [64].

Interpolation theory was developed in the western world after a great revolution in scientific thinking and, apparently, totally unaware of the important work conducted much earlier in other parts of the world [91]. The developments in astronomy and physics, initiated by Copernicus, continued by Kepler and Galileo and culminating with Newton's theories, were of particular importance for further achievements in mathematics towards what is now known as "classical" interpolation theory (until the beginning of the 20th century) [91].

The general formula for finite differences dealing with equidistant data, usually referred to as the Gregory-Newton formula [124], was first written down in 1670 by Gregory and then by Newton in his celebrated Principia, which appeared in 1687. Describing its use other than stating the importance is beyond the scope of this summary, however, we find it appropriate to note that the well-known Taylor series was obtained as a corollary to this formula [75]. Besides, a formula used by the Chinese astronomer Liù Zhuó, who is said to be the first person to use second-order interpolation formulae for computing the positions of the sun and the moon around 600AD, can be re-written in modern algebraic notation so that it equals its first three terms [91].

Newton's formula on divided differences [124] is on the other hand applicable to data at arbitrary intervals. It can be regarded as the most general of all interpolation formulae since variations of it was further studied by many others, including Stirling, Gauss, Waring, Euler, Lagrange, Bessel, Laplace, and Everett, whose formulae easily can be derived from it [91].

There were two parallel developments within interpolation theory during the first half of the 20th century which seem to be of special importance to the subsequent era of digital computers. They are both related to E.T. Whittaker's *cardinal function* [123]. It is represented

by the Newton-Gauss formula of an infinite number of equidistant values:

$$C(x) = \sum_{r=-\infty}^{\infty} \frac{a_r \sin\left\{\frac{\pi}{w}(x-a-rw)\right\}}{\frac{\pi}{w}(x-a-rw)},$$

which takes the values a_r at the points $a + rw$ while possessing the remarkable properties of having no singularities and that "when $C(x)$ is analyzed into periodic constituents by Fourier's integral-theorem, all constituents of period less than $2w$ are absent" [126]. One development began when J.M. Whittaker described a relation between the cardinal series and the truncated Fourier integral representation of a function in the case of convergence [125, 126]. This in turn lead to Shannon's sampling theorem [117, 118], which states that a bandlimited function is completely determined by its samples, and describes how to reconstruct the function from its samples.

The other development emerged from practical applications of classical polynomial interpolation, which implied using only the first few of infinitely many terms since it is computationally difficult to consider all or even many of the known function values when computing an interpolated value. Fixing the number of terms resulted in partitioning of the polynomials. A composite piecewise interpolant will in general not be continuously differentiable at the transition points. The need for smoother interpolants had triggered the development of so-called osculatory interpolation techniques in the late 1800s [114]. Several types of piecewise, differentiable polynomials had appeared (see [114] and the references therein) before Schoenberg [110, 111] described how smooth curves can be obtained by using mechanical splines. Then in the same work he introduced the notation of a mathematical spline and the definition of a spline curve represented by a series of the cardinal type:

$$F(x) = \sum_{n=-\infty}^{\infty} y_n M_k(x-n, t),$$

whose basic functions, which he defined as the inverse Fourier integral

$$M_k(x) = \frac{1}{2\pi} \int_{u=-\infty}^{\infty} \left[\frac{2\sin\left(\frac{u}{2}\right)}{u}\right]^k e^{iux} du,$$

are the so-called B-splines[1] [31, 112]. The spline representation was introduced in [110, 111] for equidistant data (without knots). The abstract for [20] had been available since 1947, but, for some reason, the article was not published before it appeared in [20]. In there, the "fundamental spline functions" associated with a set of arbitrarily spaced knots were defined and presented by using divided differences in terms of $\omega(x) = (x-x_0)\cdots(x-x_n)$:

$$M_n(x; x_0, \cdots, x_n) = \sum_{v=0}^{n} \frac{n(x_v - x)_+^{n-1}}{\omega'(x_v)}.$$

Shannon's paper was recognized within the fields of communication engineering, numerical signal processing and analysis applications, to mention a few (see e.g. [91] for ref-

[1] Schoenberg seems to have used the name "B-splines" in his work since [112].

erences). The spline theory was developed further within mono- and multivariate interpolation and approximation theory, among other fields of mathematics, resulting in several books [33, 35, 113, 76] on the topic. Following the advent of digital computers, splines had a major impact on geometric modeling and computer aided geometric design (CAGD) [51] and computer graphics [53].

1.2 Geometric modeling

A collection of mathematical methods which are used mainly to describe the shape of objects or to express some physical process in terms of geometry constitutes the discipline of geometric modeling. The collection includes solid modeling, algebraic- and computational geometry, and CAGD.

Designing of curves and surfaces is fundamental to the construction of a wide range of products, ranging from tableware and bottles to car bodies, ship hulls and wings of airplanes, just to mention a few. Moreover, they are important components in models representing parts of the real world, such as geological features, terrain or other physical phenomena, as well as objects in virtual worlds.

Computer aided manufacturing (CAM) is the process of machining of 3D shapes out of blocks of wood or steel. The exploration of the use of parametric curves and surfaces from differential geometry [44] in computer aided design (CAD) can be viewed as the origin [51] of CAGD. CAGD is traditionally about the approximation and representation of curves and surfaces for computer processing [5].

Product design problems were dealt with prior to the introduction of computers by means of descriptive geometry stored as physical templates. Most product design is nowadays performed with the aid of computer software where the geometry is stored as mathematical models. The transition from classical descriptions of geometry, such as models or drawing boards, to computer models is usually referred to as “digitizing”, i.e. sampled into a point set which is interpolated or approximated by a mathematical model. According to [5], as a rule of thumb, this CAD philosophy can be associated with Steven A. Coons [16] whereas using CAD for conceptual styling is very much thanks to Pierre Bézier.

Bézier curves [105, 51] are based on Bernstein polynomials [1, 29] and were developed independently by Paul de Faget de Casteljau around 1959 [36] and Bézier around 1962 [84, 2]. Both were working on CAD systems at French car companies; Bézier at Rénault and de Casteljau at Citroën.

Ferguson from the American airplane manufacturer Boeing Co. was the first to introduce piecewise polynomial curves, or parametric spline curves, into CAGD in 1963 [57]. It is worth noting that Gordon and de Boor were studying similar curves at General Motors around the same time, but they were used mainly for interpolation purposes and not in free form design [5]. As noted in section 1.1, the B-splines initiated by de Boor [31, 32] and others were at this time mainly studied within the aspects of approximation theory. Then one major leap was taken when Gordon and Riesenfeld [63] showed the remarkable fact that B-spline curves are the generalization of Bézier curves.

Bivariate polynomial surfaces are usually described either as rectangular tensor products

or in terms of barycentric coordinates with respect to a triangular domain. Ferguson [57] and de Casteljau [36] were among the first to use tensor product surfaces [33, 51] in CAGD. In essence, with tensor product surfaces, one curve is swept through space and possibly deformed by moving its control points along another curve. Tensor product surfaces of Bézier and B-spline type have remained among the most popular surface representations since they were introduced. Their shape is controlled by adjusting coefficients in a net of control points. Bézier surfaces are sometimes referred to as patches [51]. Complex surfaces can be composed from a number of adjacent Bézier patches where the continuity between patches is depending on the continuity of the corresponding control nets. B-spline surfaces are associated with knot vectors in both parametric directions. They are piecewise continuous of nature, thus, large control nets can be used without increasing the order of the basis functions.

De Casteljau [36] was the first to consider the triangular polynomial patches, which are now known as Bézier triangles [51], in 1959 [5]. Triangular patches are, similar to tensor products, also a generalization of curves. They have been studied as piecewise surfaces over regular and arbitrary triangulations [51]. We refer to [50] for a survey of triangular Bézier patches. A mathematical assessment of spline spaces over triangulations can be found in [76].

Triangular and rectangular surface patches are both common in CAGD, however, the latter seems to be more often encountered [53] despite that triangular patches are better suited to describe complex geometries [51]. Some modeling software applications support both types of patches [53]. Geometric continuity between patches of different types has been studied in the literature, see e.g. [51, 105] and the references therein. Conversion between tensor product and triangular Bézier patches is addressed in [105].

The topologies of triangular and rectangular surface patches are limited of nature. Many shapes are too complex to model using only such images of parts of the plane. A simple example from [51] is the sphere. It is not possible to construct it without introducing degenerate patches while using a C^1 map of a part of the plane. There exists several methods that are suited for representation of surfaces with arbitrary topology. We mention surface patches with more than four sides [88, 89], subdivision techniques including Doo-Sabin surfaces [46], Catmull-Clark surfaces [12], Loop subdivision [87] and Hermite subdivision [17, 18], and the (local) refinement methods hierarchical B-splines [60], T-splines [115], PHT-splines [42] and LR B-splines [45].

Some of the early CAD systems were based on conic sections and others on quadric surfaces. A need for compatible formats and exchange of data between CAD systems led to a standardization of data formats [53]. The rational extension of nonuniform B-splines, nonuniform rational B-splines (NURBS), can be viewed as the projection of a surface in a projective space into an affine space, where the control polygon consists of homogeneous points with associated weights [51]. NURBS are today integral parts of the initial graphics exchange specification (IGES) and standard for the exchange of product model data (STEP) industry standards used in CAD. It is worth mentioning that the benefits of local refinement by knot insertion [4, 14], which leads to a non-uniform knot vector, probably encouraged the transition from uniform towards non-uniform spline models [15].

Expo-rational B-splines (ERBS) were introduced to CAGD in [77] as a compound spline function with C^∞-smooth expo-rational basis functions associated with a knot vector. The purpose was not to replace traditional B-splines or NURBS but rather to complement them.

The main difference between ERBS and polynomial B-splines is that the coefficients are local functions instead of ordinary coefficients. Some key features of ERBS are the special Hermite interpolation property, which facilitates approximation of geometric objects, and the minimal local support of the basis functions which is similar to linear polynomial B-splines. Furthermore, ERBS facilitate flexible and intuitive "geometric editing" possibilities and dynamic shapes by simple affine transformations of local functions. A short overview of ERBS is provided in section 1.4.

The scope of this section is limited to methods relevant for this dissertation. We refer to [5, 15, 51, 52, 53, 105] for a more detailed overview of geometric modeling and CAGD, including topics not covered here.

1.3 Algorithms for polynomials on Bernstein form

The Bernstein polynomials [1] of degree d, defined as

$$b_{i,d}(t) = \binom{d}{i} t^i (1-t)^{d-i}, \quad i = 0, \ldots, d,$$

enjoy a number of important properties, including that they are linearly independent and symmetric, they have roots at 0 and 1 only, they form a partition of unity and that they are positive in $(0,1)$. Furthermore, they satisfy the recursion formula

$$b_{i,d}(t) = t\, b_{i-1,d-1}(t) + (1-t) b_{i,d-1}(t),$$

where $b_{-1,d-1} = b_{d,d-1} = 0$ and $b_{0,0} = 1$.

Every polynomial curve $c(t)$ of degree $\leq d$ has a unique Bézier representation of degree d which can be written in Bernstein form as follows:

$$c(t) = \sum_{i=0}^{d} \mathbf{c}_i b_{i,d}(t), \tag{1.1}$$

where $(\mathbf{c}_i)_{i=0}^d$ are its $d+1$ control points and $b_{i,d}(t)$ are the Bernstein basis polynomials.

One of the most common ways to evaluate parametric Bézier curves and surfaces is by using de Casteljau's corner cutting algorithm [36]. By repeatedly applying the recursion formula for Bernstein polynomials to (1.1) for $b_{i,d}(t)$, and letting $\mathbf{c}_{i,0} = \mathbf{c}_i$, one obtains

$$c(t) = \sum_{i=0}^{d} \mathbf{c}_{i,0} b_{i,d}(t) = \sum_{i=0}^{d-1} \mathbf{c}_{i,1} b_{i,d-1}(t) = \cdots = \sum_{i=0}^{0} \mathbf{c}_{i,d} b_{i,0}(t) = \mathbf{c}_{0,d},$$

where the points given by

$$\mathbf{c}_{i,k} = (1-t)\mathbf{c}_{i,k-1} + t\,\mathbf{c}_{i+1,k-1}$$

are the intermediate points of the de Casteljau algorithm.

The ith B-spline of degree d associated with the knots $(t_i, \ldots, t_{i+d+1})$ can be expressed

using the de Boor-Cox recursion formula [19, 31]:

$$M(t;t_i,\ldots,t_{i+d+1}) = M_{i,d}(t) = \frac{t-t_i}{t_{i+d}-t_i}M_{i,d-1}(t) + \frac{t_{i+1+d}-t}{t_{i+1+d}-t_{i+1}}M_{i+1,d-1}(t), \tag{1.2}$$

where the recursion terminates when

$$M(t;t_i,t_{i+1}) = M_{i,0}(t) = \begin{cases} 1, & \text{if } t_i \le t < t_{i+1}, \\ 0, & \text{otherwise.} \end{cases}$$

Let $(t_i)_{i=0}^{n+d+1}$ be a knot vector and the linear combination

$$s(t) = \sum_{i=0}^{n-1} \mathbf{c}_i M_{i,d}(t) \tag{1.3}$$

be a spline curve associated with the n control points $(\mathbf{c}_i)_{i=0}^{n-1}$. Due to the local support property of the B-splines, for $t \in [t_\mu, t_{\mu+1})$, (1.3) can be written as

$$s(t) = \sum_{i=\mu-d}^{\mu} \mathbf{c}_i M_{i,d}(t),$$

thus, $s(t)$ can be evaluated by using only the $d+1$ control points $(\mathbf{c_i})_{i=\mu-d}^{\mu}$. Setting $\mathbf{c}_{i,0} = \mathbf{c}_i$ and applying (1.2) yields

$$s(t) = \sum_{i=\mu-d}^{\mu} \mathbf{c}_{i,0} M_{i,d}(t) = \sum_{i=\mu-d+1}^{\mu} \mathbf{c}_{i,1} M_{i,d-1}(t) = \cdots = \sum_{i=\mu}^{\mu} \mathbf{c}_{i,d} M_{i,0}(t) = \mathbf{c}_{\mu,d},$$

where $\mathbf{c}_{i,r}$ are given by the affine combinations

$$\mathbf{c}_{i,r} = (1-\alpha)\mathbf{c}_{i-1,r-1} + \alpha \mathbf{c}_{i,r-1},$$

where

$$\alpha = \alpha_{i,d-r} = \frac{t-t_i}{t_{i+d+1-r}-t_i}.$$

The discovery of the recurrence relations for B-splines by de Boor [31] in the US and Cox [19] in England pointed at a numerically stable way to evaluate B-splines. The existing methods at that time were based on divided differences and were numerically unstable [19]. The recursive de Boor-Cox algorithm is related to de Casteljau's algorithm. It provides fast and numerically stable computation of B-spline curves and surfaces [15]. We note here what was shown later in [55, 56, 54]; polynomials on Bernstein form is numerically more stable than the monomial form.

Based on the need for increased design flexibility, Lane and Riesenfeld [83] provided methods based on subdivision for Bézier curves and uniform B-splines. Attempts to extend Lane-Riesenfeld subdivision to B-splines with non-uniformly spaced knots resulted in two algorithms for knot vector refinement; Böhm's method [4] for inserting one knot at a time and the Oslo algorithms [14] for inserting a set of knots in parallel. Böhm's method is addressed

briefly in chapter 4. We mention here that the Oslo algorithms are based on the theory of discrete B-splines [14, 61].

1.4 Expo-rational B-splines

Expo-rational B-splines origin from research conducted since the year 2003 [77]. They were presented for the first time in [37] and published in [80, 81, 39]. Since then, several new types of splines derived from ERBS have emerged, including generalized expo-rational B-splines (GERBS) [38], and, most recently, logistic expo-rational B-splines (LERBS) [40, 129]. Together they constitute a family of blending-type spline constructions.

ERBS is the fundament of this dissertation. It is a blending-type spline construction where local functions at the knots are blended together by infinitely smooth basis functions. An ERBS function $f(t)$ is defined in [77] by

$$f(t)=\sum_{k=1}^{n}\ell_k(t)B_k(t),\ t\in(t_1,n_n], \tag{1.4}$$

where $\ell_k(t)$ are scalar-, vector-, or point-valued local functions defined on (t_{k-1},t_{k+1}), $\mathbf{t}=\{t_k\}_{k=0}^{n+1}$ is an increasing knot vector, and the basis function

$$B_k(t)=\begin{cases} S_{k-1}\int_{t_{k-1}}^{t}\varphi_{k-1}(s)ds, & t_{k-1}<t\le t_k,\\ S_k\int_{t}^{t_{k+1}}\varphi_k(s)ds, & t_k<t<t_{k+1},\\ 0, & \text{otherwise,}\end{cases} \tag{1.5}$$

with the ERBS scaling factor defined as

$$S_k=\frac{1}{\int_{t_k}^{t_{k+1}}\varphi(s)ds},\ \text{when } t_k<t_{k+1},$$

and

$$\varphi(t)=\exp\left(-\gamma_k\frac{|t-\big((1-\mu_k)t_k+\mu_k t_{k+1}\big)|^{2\sigma_k}}{\big((t-t_k)(t_{k+1}-t)^{\beta_k}\big)^{\alpha_k}}\right), \tag{1.6}$$

where $\alpha_k>0$, $\beta_k>0$, $\gamma_k>0$, $0\le\mu_k\le1$, and $\sigma_k\ge0$, for $k=1,\ldots,n$, are the intrinsic parameters of the ERBS. Throughout the thesis we shall consider the default values of the intrinsic parameters, $\alpha_k=\beta_k=\gamma_k=\sigma_k=1$, $\mu_k=\frac{1}{2}$, as described in [39], unless specified otherwise. This leaves us with a simpler version of (1.6):

$$\varphi(t)=\exp\left(-\frac{\left(t-\frac{t_k+t_{k+1}}{2}\right)^2}{(t-t_k)(t_{k+1}-t)}\right). \tag{1.7}$$

A plot of the ERBS basis defined in (1.7), using the default set of intrinsic parameters, and its first derivative is provided in Figure 1.1.

The five basic properties of the univariate basis function $B_k(t)$ outlined in [81] are as

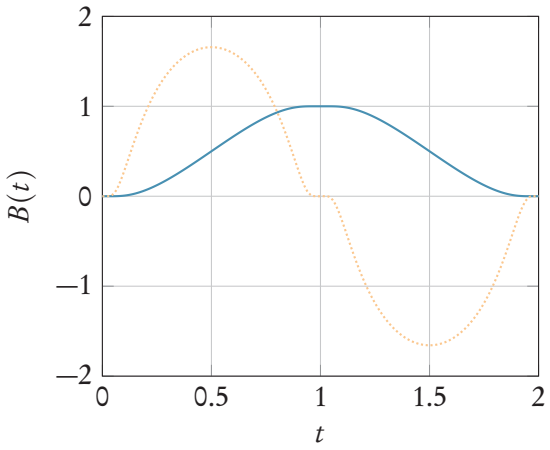

Figure 1.1: The ERBS basis function defined in (1.5) and its first derivative are shown as a solid and a dotted line, respectively, using default values of the intrinsic parameters, and where $t_{k-1}=0$, $t_k=1$ and $t_{k+1}=2$.

follows:

1. $B_k(t)\begin{cases} >0, & \text{if} \quad t_{k-1}<t<t_{k+1} \quad \text{for} \quad k=1,\ldots,n. \\ =0, & \text{otherwise} \end{cases}$

2. $\sum_{k=1}^{n} B_k(t)=1$, i.e. $B_k(t)+B_{k+1}(t)=1$ for $t_k<t\le t_{k+1}$.

3. $B_k(t_k)=1$ if $t_{k-1}<t_k$, and $\lim_{t\to t_k+} B_k(t)=1$ if $t_{k-1}=t_k$, $k=1,\ldots,n$.

4. $D^j B_k(t_i)=0$ for $j=1,2,\ldots$, $i=1,\ldots,n$, and $k=1,\ldots,n$.

5. $B_k(t)$ is C^∞ on (t_{k-1},t_{k+1}).

One consequence of the ERBS minimal support property is that only two functions are blended together in every knot interval (i.e. between two knots). The following formula describes how blending of two functions is performed in a knot interval of (1.4) by applying the basic property 2:

$$\begin{aligned} f(t) &= B_k(t)\ell_k(t)+B_{k+1}(t)\ell_{k+1}(t) \\ &= (1-B_{k+1}(t))\ell_k(t)+B_{k+1}(t)\ell_{k+1}(t) \\ &= \ell_k(t)+B_{k+1}(t)(\ell_{k+1}(t)-\ell_k(t)), \end{aligned} \tag{1.8}$$

where $B(t)$ is a blending function (B-function).

The ERBS construction possesses a Hermite interpolation property which follows from property 4 above, namely that $D^j B_k(t_k)=0$, if $k=1,\ldots,n$ and $j=1,2,\ldots$. A brief summary of the Hermite interpolation property described in [77, theorem 2.4] is as follows. For the general ERBS d-dimensional vector function

$$f(t)=\sum_{k=1}^{n} c_k\circ\omega_k(t)B_k(t),$$

then

$$D^j f(t_k) = \delta_k^j D^j c_k \circ \omega_k(t_k), \tag{1.9}$$

for $j = 0, 1, 2, \ldots$ and $k = 1, \ldots, n$, where $\omega_k(t)$ maps a segment (t_{k-1}, t_{k+1}) in the "global" domain of the ERBS function onto the domain of a local function, and δ_k is the global/local "scaling" factor. The transfinite Hermite interpolation property provides that an ERBS function interpolates the values and all existing derivatives of its local functions in their respective knots.

Blending functions

A B-function [78, 79] is a function designed for blending which possesses the following set of properties:

1. $B : I \to I$ $(I = [0, 1] \subset \mathbb{R})$.
2. $B(0) = 0$.
3. $B(1) = 1$.
4. $B^{'}(t) \geq 0$, $t \in I$.
5. $B(t) + B(1-t) = 1$, $t \in I$.

The last property is optional and specifies point symmetry around the point $(0.5, 0.5)$.

B-functions can take many shapes, including trigonometric, polynomial, rational and expo-rational. One example of a smooth B-function is using the scalable subset [77] of the ERBS basis mapped to the interval $[0, 1]$:

$$B(t) = S_d \int_0^t \varphi(s) ds, \; t \in [0, 1],$$

where

$$\varphi(t) = \exp\left(-\frac{\left(t - \frac{1}{2}\right)^2}{t(1-t)} \right)$$

and

$$S_d = \left[\int_0^1 \varphi(t) dt \right]^{-1}.$$

Another example, which belongs to the family of GERBS [38], is the LERBS presented in [129]:

$$B(t) = \frac{1}{1 + \exp\left(\frac{1}{t} - \frac{1}{1-t}\right)}.$$

The linear function

$$B(t) = t$$

is perhaps the simplest one which meets the criteria of a blending function, as outlined above.

The ERBS blending construction was adapted to an adjusted recursive definition of the B-spline in [78] by considering a B-function and using (1.8). By adding blending functions to the recursive definition of the B-spline, provided in (1.2), we obtain the general B-spline presented in [79]:

$$B_{i,d}(t) = B \circ \omega_{i,d}(t) B_{i,d-1}(t) + (1 - B \circ \omega_{i+1,d}(t)) B_{i+1,d-1}(t), \tag{1.10}$$

where B is a B-function, $\omega_{i,d}(t) = \frac{t-t_i}{t_{i+d}-t_i}$,

$$B_{j,0}(t) = \begin{cases} 1; & \text{if } t_i \leq t < t_{i+1}, \\ 0; & \text{otherwise,} \end{cases}$$

and, in the case of ERBS, the degree $d = 1$. Thus, an alternative way to express (1.4) is

$$f(t) = \sum_{k=1}^{n} \ell_k(t) B_{1,k}(t), \ t \in (t_1, n_n],$$

where $B_{1,k}(t)$ is as defined in (1.10). This specific form of generalization of the ERBS blending construction appeared during the period of this research project. It is a key component in the rendering method proposed in [7] which is included in chapter 7 of this dissertation.

1.5 Objectives and overview

The purpose of the present work is to provide some methods and ideas for practical use of ERBS in CAGD. The work is based on the framework introduced in [77], and the GMlib [82] C++ software library which was extended with an ERBS evaluator and an implementation of the ERBS blending construction for parametric curves and surfaces.

Dreamworld [95] is a research project under the Verdikt program by the Research Council of Norway where the computer game manufacturer Funcom and Narvik University College are stakeholders. Some of the key research and development (R&D) challenges defined in the project include digital distribution of large data sets, optimizing graphics for low-end hardware and online clients, and compression and streaming of large amounts of data. This implies representing detailed and realistic models of synthetic and real terrain, with special details, together with other objects which may be flexible or interactive. Some of the research included in this dissertation has been performed as a partial fulfillment of the contribution from Narvik University College (NUC) to the Dreamworld project.

The main goal of the scientific work was to apply interpolation and smoothing techniques from the ERBS blending construction for use in representation and visualization of geometry. The underlying data used in the experiments were provided by various sources from discrete and continuous origin.

The author has opted for grouping the research topics presented in this thesis into two parts. One part is addressing modeling and representation of data in terms of ERBS. The methods and constructions included there can serve as building blocks for the the other part, whose topics are evaluation and visualization, which covers methods for rendering and

presentation of the so-defined data.

There follows a brief description of the individual objectives and the main contributions of this dissertation. More details are accessible in parts II and III of this thesis.

1.5.1 Modeling and representation

1. Interpolation and approximation. B-splines are typically used to represent discrete or continuous data [51, 105, 33, 53]. The process of determining appropriate knot vectors (and control polygons) to obtain a satisfying approximation has been studied thoroughly in the literature [66, 67, 90, 49, 59]. At present, it is still an active research area (see e.g. [58, 108, 10, 74, 73]). There is probably no single "best" parameterization since the methods depend on the data [53]. As mentioned in section 1.4, with ERBS type splines, the coefficients, which can be scalar-, point- or vector-valued *functions*, such as parametric curves or surfaces, are interpolated at their associated knots. One consequence of the ERBS Hermite interpolation property [77, theorem 2.4] (see (1.9)) is that an ERBS function interpolates positions and all existing derivatives of the local functions at their associated knots.

Using ERBS to represent data involves finding knot vectors, which are interpolation points, and to determine local functions of some type. A method for fitting discrete data with generalized expo-rational B-splines by determining interpolation knots and generating GERBS local curves, in this case as truncated Taylor polynomials, by partitioning the parametric space and solving a least-squares fitting problem is presented in chapter 3. Based on those results, fitting of discrete data using a tensor product GERBS surface, where derivatives for GERBS local Bézier surface patches are obtained from the discrete data by finite differences, is explored in chapter 5. Furthermore, a method to partition the parametric space by feature point detection based on tools from differential geometry is proposed in chapter 5.

The blending construction presented in chapter 6 interpolates vertex positions of a triangulated network. It can be considered as an alternative to global methods for fitting a tensor product surface (of "any" type, including Bézier and NURBS,) to a triangulation. Data fitting via this method may provide less data smoothing than e.g. least squares approximation due to the interpolation properties and the localness within a $\star_2$-neighborhood of the triangulation.

2. Smoothness of the spline approximation. There is a tradeoff between accuracy and computational cost (how much data to store and the approximation error) of a spline representation of a discrete data set [91, 35, 113]. Constraining the spline approximation to satisfy some criteria of measure, such as approximation error, can be in order. Certain feature points can be important to interpolate or approximate with higher precision, while near other areas of the data, which are less important, lower precision may be acceptable.

Flexibility by varying the number of interpolation knots, finding appropriate interpolation points (points of special importance or interest), and setting the polynomial degree and influence of the local functions are investigated in chapters 3 and 5. Local functions with adjustable degree are obtained by varying the multiplicity of the Taylor expansion of polynomials in chapter 3 and by varying the number of derivatives obtained from the discrete data in chapter 5. Local refinement techniques such as knot insertion, which is addressed in chapter 4, can be considered for improved control in this setting through explicit placement of interpolation points.

The construction in chapter 6 provides an ERBS blending approximation of a triangulated surface. Approximation with the proposed method yields flexible smoothness properties over the triangle edges. Smoothness is addressed using two different blending functions. One of them is blending a smooth component together with the triangle edges and is C^0 over the edges. The other one provides C^1 smoothness over the edges under the penalty of pulling the spline surface approximation away from the edges. The results are presented as visual examples.

3. Interactivity. Being able to interact with, or edit, a model is a fundamental feature in many geometric modeling applications [51, 53, 15]. Knot insertion [4, 14] has undoubtedly played an important role [5, 34] in establishing B-splines and NURBS among the most significant representations for CAGD [51]. A consequence of the ERBS blending construction and its minimal support property is that the refined ERBS representation, after inserting a new knot, under the constraint of not changing any of the existing local functions, is on rational form. A proof of this is provided in chapter 4. For practical use, when the local functions are parametric curves or surfaces, it is sometimes required that the new local function associated with the inserted knot should be of the same kind as the existing local functions. A few strategies to generate the new local function are presented and benchmarked in chapter 4.

In contrast to (classic) B-splines and NURBS, whose coefficients are points, it makes sense to apply scaling or rotation to ERBS coefficients due to the "geometric editing"- and dynamic shape possibilities by simple affine transformations of the local functions [77]. Special points of interest can for instance be used to indicate areas where editing is appropriate. Thus, applying strategies for placing the initial interpolation knots and determining suitable local functions can have impact on the interactivity properties of ERBS constructions. The findings in chapters 3 and 5 can be relevant for this purpose.

4. Tensor-product models. One of the most common representations of a surface in terms of blending type splines is by tensor products [53]. In the case of discrete input data, approximated using any kind of spline, there is a need to determine the knot vectors. ERBS-type tensor product spline surfaces interpolate their associated local *surface patches* at their interpolation knots. This is different with classic B-splines where the coefficients are a net of control points which are not necessarily interpolation points. A method for constructing a tensor product ERBS surface from a discrete equidistant point set, or height map, is described in chapter 5. The interpolation knots are determined by feature point detection in terms of curvature sign changes using a method extended from the ones presented in chapter 3, and derivatives for the local surface patches are approximated using finite differences.

5. Triangle-based models. Scattered points and polygonal meshes are often encountered in geometric data representations [53]. Triangulated surfaces, such as Delaunay triangulations [41, 68, 30], are commonly used to represent surface data, in particular terrain models and objects in virtual worlds. The triangulated irregular network (TIN) [103, 104] is one example of a digital data structure, or digital terrain model (DTM), which is used to represent surfaces in geographic information systems (GIS). Chapter 6 proposes an ERBS blending construction, based on triangulated surface input data, constituting a surface representation which can be smooth over the input triangle edges while interpolating the vertex positions.

ERBS surface patches are constructed to cover two triangles sharing one common edge. Up to three surface patches may cover a triangle. They are blended together by the method, which is local within a $\star_2$-neighborhood of every TIN node. The construction in chapter 6 is one example of how a triangulated model can be represented as a blending construction which can be evaluated everywhere on its domain.

6. Mapping and relation between triangle-based models and tensor-product models. As noted in section 1.2, the tensor product NURBS surface constitutes one of the most common representations of geometric shapes in CAD software applications. Utilizing ERBS for interactive editing and dynamic shaping of such existing representations will require some sort of data interchange. Converting a tensor product NURBS surface to a tensor product ERBS representation can be done by constructing appropriate ERBS local surface patches from the NURBS. We note here that the vice versa conversion from ERBS to NURBS yields an approximation, which typically would involve a data fitting procedure. One of the reasons is the fundamental difference in smoothness properties. As an example, it is possible to represent singularities and sharp edges by using a tensor product ERBS construction, thanks to its vanishing derivatives at the knots and the transfinite Hermite interpolation properties. It can be difficult to obtain equivalent shapes with NURBS while maintaining the smoothness properties, at least without applying some type of refinement or subdivision techniques.

Discrete data points can be obtained, for instance by sampling a geometric model or by measuring parts of the real world, imposing constraints on the subsequent interpolation. Triangulations are sometimes used to induce a topology to scattered point data. Edges in triangulations can be regarded as lines between adjacent vertices, and, often as one of the constraints. However, straight line triangle edges may not follow the curvature of the underlying geometry. The construction in chapter 6 can be used to approximate the shape of the underlying geometry over edges in triangulations. Parametric tensor product grids can then be obtained by triangulating scattered data (samples) and applying the method as described in chapter 6. Furthermore, the method may be applicable to a wider class of object in Euclidean space, including embeddings of (partitioned) Riemannian manifolds, via triangulating in the parametric domain.

1.5.2 Evaluation and visualization

7. Tessellation. Tessellations is a well-researched and fundamental problem in mathematics as well as computer graphics [109]. In this work we consider tessellation from the computer graphics- and rendering point of view, as the process of converting scattered points to (solid) surfaces built up by polygon primitives, rather than developing new methods for tessellation. One recent development is the introduction of hardware tessellation in modern graphics processing units (GPUs), where the specific hardware implementations are vendor-dependent. Hardware tessellation has been made available to developers since the Microsoft® DirectX® (DirectX) 11 [92] and open graphics library (OpenGL) 4.0 [116] application programming interfaces (APIs). In the following, we shall use the OpenGL nomenclature to describe the hardware tessellation shaders.

A method for rendering of blending type splines by exploiting the tessellation features available in recent GPU hardware are proposed in chapter 7. The concept provides guide-

lines for data representation in terms of splines on the form of (1.10). Strategies for implementation of an efficient evaluator for visualization using tessellation evaluation shaders are suggested, where ERBS coefficients associated with a knot vector can be represented on the GPU by using tessellation control patches. The discretization is left to the fixed-function tessellator shader step, where the amount of tessellation can be controlled. The tessellator generates sample points and induces a topology, whereas the placement of vertices in some Euclidean space are computed by the tessellation evaluation shader.

8. Pre-evaluation techniques. Pre-evaluation of basis functions connected to tessellations can be useful for speeding up the computations. Bernstein polynomials [1] are commonly used as basis functions in spline constructions [35, 76] and to represent curves and surfaces on Bézier form [51, 105]. Bézier curves and surfaces have in turn been used as local functions within ERBS blending constructions [77]. Evaluators for those are implemented as a part of the GMlib software library [82].

Bernstein polynomials of arbitrary degree and number of variables, represented on matrix form by factorization, are proposed and explored in chapter 8. The multivariate Bernstein factor matrices are defined recursively by using barycentric coordinates and exploiting a decomposition of the matrices into submatrices. Furthermore, in chapter 8, the relevance of the matrix factorization to the de Casteljau algorithm is addressed. This construction can be considered as a part of a specific application in connection with the rendering method proposed in chapter 7.

9. GPU utilization. The rendering method described in chapter 7 (see section 1.5.2 above) is based on using the GPU. In contrast to fixed-function rendering methods, where the geometry representation is sampled into a mesh of vertices and edges, using the central processing unit (CPU), which then needs to be "pushed" to the GPU memory for visualization, the concepts proposed in chapter 7 enables the spline construction to be preserved up to the point where it is discretized by the GPU hardware.

The GPU is composed of several streaming multiprocessors, or vector processing units. They process chunks of data in parallel in a single instructions multiple data (SIMD) fashion [109]. Each streaming multiprocessor typically has many cores. The total number of cores in modern GPUs can be several thousand. This makes them suitable for parallel computations, which are not limited to graphics purposes. General-purpose GPU (GPGPU) is available through different APIs, notably open computing language (OpenCL) [70] and compute unified device architecture (CUDA) [98]. The numerical benchmark experiments described in chapter 10 are performed on a GPU-based implementation of the discrete wavelet transform (DWT) by using the OpenCL 2.0 C language specification [93].

10. Rendering performance and data transmission. Strategies for increasing the rendering performance of a rendering system can be in order, depending on where the bottlenecks reside. Methods for image compression are applicable to captures of the rasterized frame buffer. They can be applied to reduce the required bandwith for transmission in "low-end" hardware, and streaming- or web-based applications, especially when the rendering is performed remotely and then transferred in a sequence of images to a client for display. Compression or data reduction can be applied to the geometry representation (before tessellation)

in cases where CPU-based representation and evaluation of splines is considered. Furthermore, compression or data reduction can be applied to local functions in case of GPU-based representation, such as, for example, the one proposed in chapter 7. This can in both cases reduce the necessary bandwidth for data transmission between CPU and GPU memory.

Chapter 9 considers controlling compression in a JPEG-like set-up by adjusting the shape of the basis function via setting a shape parameter and constructing custom quantization tables based on histograms obtained from a relatively large set of images. The results show increased compression performance against approximation error, when compared to a joint photographic experts group (JPEG) reference, while preserving the perceived change in structural information. For this purpose two signal fidelity measures, peak signal-to-noise ratio (PSNR) and mean structural similarity (MSSIM), respectively, are provided to compare the results.

A benchmark study of a custom method for thresholding of discrete wavelet coefficients for a selection of wavelet bases applied to smooth- and semi-smooth test functions, with and without singularities, are presented in chapter 10. The results indicate that which wavelet that performs best according to selected criteria is depending on the signal's smoothness properties. Furthermore, the performance can increase for signals with varying smoothness properties by partitioning the global signal and selecting the best performing wavelet individually for each local partition. Discretized samples of ERBS, which can be used to represent data with smooth parts and isolated singularities due to its intrinsic partitioning, via selecting the interpolation knots with associated local coefficients, are especially well suited for this method.

1.6 Organization of the thesis

This dissertation is prepared as a collection of articles which resulted from cooperative work with colleagues in the research group Simulations at Narvik University College.

Chapter 1 covers an overview of relevant interpolation techniques and geometric modeling from a historical perspective, and a brief description of ERBS, followed by an outline of the research objectives, which ties together the individual research papers. Chapter 2 presents a list of published results, enumerates the individual contributions of the research papers and provides some remarks and notes for future work.

While chapters 1 and 2 constitutes part I of this thesis, parts II and III present the the included scientific work. Research related to approximation, fitting, refinement and representation of data constitutes part II, whereas part III covers research related to evaluation, rendering and presentation of data.

Some details and figures, which were omitted from the associated research papers due to page limitation policies of the publishers, are included as appendices.

2 Summary of related results

2.1 List of relevant published research results

The main contribution of this dissertation is based on reprints of a set of scientific papers. In this section follows a list of articles authored or co-authored by the author of the thesis:

1. J. Bratlie, R. Dalmo, and P. Zanaty. Fitting of discrete data with GERBS. In Lirkov et al. [86], pages 569–576

2. R. Dalmo. Local refinement of ERBS curves. In Pasheva and Venkov [100], pages 204–211

3. R. Dalmo and J. Bratlie. Data approximation using a blending type spline construction. In Pasheva and Venkov [101], pages 147–152

4. R. Dalmo, J. Bratlie, B. Bang, and A. Lakså. Smooth spline blending surface approximation over a triangulated irregular network. *International Journal of Applied Mathematics*, 27(1):109–119, 2014

5. J. Bratlie, R. Dalmo, and B. Bang. Evaluation of smooth spline blending surfaces using GPU. In J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curves and surfaces. 8th International Conference*, volume 9213 of *Lecture Notes in Computer Science*, pages 60–69. Springer, 2015

6. R. Dalmo. Matrix factorization of multivariate Bernstein polynomials. *International Journal of Pure and Applied Mathematics*, 103(4):749–780, 2015

7. R. Dalmo, J. Bratlie, and P. Zanaty. Image compression using an adjustable basis function. *Mathematics in Engineering, Science and Aerospace*, 6(1):25–34, 2015

8. R. Dalmo, J. Bratlie, and B. Bang. Performance of a wavelet shrinking method. In Dimov et al. [43], pages 262–270

The articles 1 to 8 above have been selected for inclusion in this dissertation. They can be found in chapters 3 to 10 in the same order as outlined above. In addition, the following list of articles, which are authored or co-authored by the author of the thesis, have appeared during the period of the Ph.D. project:

9. R. Dalmo and J. Bratlie. Discrete wavelet compression of ERBS. In Lirkov et al. [86], pages 577–584

10. J. Bratlie and R. Dalmo. Motion capture data represented using a blending type spline construction. In Pasheva and Venkov [101], pages 153–157

11. R. Dalmo, J. Bratlie, and P. Zanaty. Image processing with LERBS. In Sivasundaram [119], pages 271–278

12. A. Pedersen, R. Dalmo, and J. Bratlie. Modeling terminal ballistics using blending-type spline surfaces. In Sivasundaram [119], pages 796–803

13. B. Haavardsholm, J. Bratlie, and R. Dalmo. Surface deformation over flexible joints using spline blending techniques. In Sivasundaram [119], pages 377–383

14. J. Bratlie, R. Dalmo, and B. Bang. Wavelet compression of spline coefficients. In Dimov et al. [43], pages 246–253

The author has opted for omitting the articles 9–14 as chapters of this dissertation in order to maintain a strong focus on the objectives outlined in section 1.5.

2.2 List of contributions

This section iterates over the contributions from the individual research papers. The presentation of the papers in chapters 3 to 10 follows the same order as the enumeration in section 2.1.

2.2.1 Modeling and representation

Chapter 3: Fitting of discrete data with GERBS

1. A few strategies to determine interpolation knots, generate GERBS local curves by partitioning of the parametric space and solving a least-squares problem for fitting of discrete data were proposed and benchmarked. The purpose was to represent a discrete data set using GERBS type splines with fewer coefficients than the number of data points in the original data.

2. Some feature-based partitioning methods were proposed and compared to uniform partitioning in connection with least squares fitting of the local functions. The method was applied to samples of a smooth signal and of oscillating irregular input data for comparison.

   (a) Feature-based partitioning was able to meet the performance of uniformly distributed interpolation knots when applied to a smooth signal.

   (b) Feature-based partitioning was found to outperform uniform partitioning for oscillating irregular input data.

3. It was noted that the least squares fitting of the individual local functions, in contrast to a "global" fitting problem, can be computed in parallel, as a consequence of the intrinsic minimal support property of the GERBS construction. A similar argument was claimed for the feature detection computations.

Chapter 4: Local refinement of ERBS curves

4. Inserting a new knot into the knot vector of an existing ERBS curve, under the restriction of not allowing to change any of the existing local curves, was proven to yield a rational local curve associated with the new knot. Furthermore, how to map the parameter value within the new knot intervals to $[0,1]$ and how to re-parameterize the affected existing local curves to the new refined knot vector was explored. An analysis of whether the ERBS Hermite interpolation property holds after knot insertion was performed and confirmed.

5. The local functions can be polynomial curves in practical applications, thus, adding one on rational form may not be appropriate. A few strategies to approximate the local curve associated with the new knot were proposed and benchmarked for a parametric test curve. The measures of deviations were provided using four different metrics. The errors were found to be restricted within the support interval of the new local curve.

Chapter 5: Data approximation using a blending type spline construction

6. A method for fitting of a tensor product GERBS surface to equidistant discrete point data and generating interpolation knot vectors by detecting feature points in terms of curvature sign changes was proposed. Approximation of derivatives from the discrete data by finite differences, for use in feature point detection and construction of local surface patches, was proposed.

7. An example was provided where the method was applied to a 2D height map surface. Local Bézier surface patches were constructed for the test case by using Hermite interpolation of the discrete data at the determined feature points. An error measure of the construction was provided.

8. The primary purpose was to provide a smooth, flexible and evaluable spline representation obtained from discrete data. Suggestions for application areas were tessellations and data reduction.

Chapter 6: Smooth spline blending surface approximation over a triangulated irregular network

9. A construction for blending of ERBS surface patches covering pairs of neighboring triangles in a triangulated irregular network was proposed and explored. The four local surface patches for each ERBS patch were constructed as bi-linear Bézier surfaces by interpolating a TIN node position and obtaining directional derivatives from the triangle edges.

10. The purpose was to obtain a surface approximation which was smooth over the triangle edges while interpolating the positions of the TIN nodes. Two functions for blending together overlapping ERBS patches were proposed:

    (a) Blending using a modified version of the ERBS triangles.

(b) Blending based on angle ratios in the triangles.

The main difference between the two blending functions was related to the distribution of patch weights over edges in the underlying triangulation.

11. The blending construction was applied to a synthetic triangular network. The effects of the two blending methods were compared by providing visual examples of tessellations sampled from the constructions.

2.2.2 Evaluation and visualization

Chapter 7: Evaluation of smooth spline blending surfaces using GPU

12. A method for evaluation and rendering of smooth blending type spline constructions, over regular and irregular knot nets, based on the recently introduced tessellation shaders available in graphics hardware was proposed.

13. The concept was based on three recent contributions to the ERBS research:

    (a) GERBS blending functions formulated as a special case of an adjusted recursive definition of the B-splines. This formulation was used to propose how an evaluator could be implemented by utilizing the tessellation shader steps.

    (b) C^∞-smooth B-functions of LERBS type by using elementary functions, where no integration step was required, was considered implemented on the GPU.

    (c) A recently introduced ERBS construction on irregular grids, supporting regular-, T- and star joints, was considered and used in the definitions of grid components of the rendering method.

14. The following grid components were defined:

    (a) *Render lattice* to describe a grid structure based on the net of spline knots.

    (b) *Render locus* to describe loci in the render lattice, closely related to the spline knots and the regular-, T- and star points defined in the above mentioned irregular grid construction.

    (c) *Render block* to describe lines or faces in a render lattice. This concept was related to the description of patch primitives of the tessellation shader.

15. A visual example of a blending type spline surface based on quadratic render blocks with regular render loci was provided.

16. Some strategies for implementations of the proposed concepts were presented and discussed.

Chapter 8: Matrix factorization of multivariate Bernstein polynomials

17. Multivariate Bernstein polynomials of arbitrary degree on matrix form by factorization were presented and explored. A definition of the Bernstein factor matrix based on the recurrence relation for Bernstein polynomials was provided.

18. An alternative, recursive definition of the Bernstein factor matrix based on a particular decomposition of the matrices into submatrices was introduced. A matrix notation for a set of Bernstein polynomials was proposed, and it was shown that the factor matrices yield a factorization of the Bernstein polynomials.

19. Some properties of the factorization, including symmetry, commutativity, and differentiability of the factor matrices, were investigated.

20. The relevance of the matrix factorization to de Casteljau's corner cutting algorithm was addressed. One notable result was the observation that inverting the order of steps of a part of the factorization provides a new, matrix-based, algebraic representation of a multivariate generalization of the de Boor-Cox recursion formula (specialized in Bézier form).

21. A set of representative examples were provided; including a geometric interpretation of the de Casteljau algorithm by matrix factorization, and the representation of a multivariate surface and its directional derivatives in Bézier form by factor matrices.

Chapter 9: Image compression using an adjustable basis function

22. A custom transform related to the discrete cosine transform (DCT) using an adjustable LERBS basis function was proposed. The purpose was to explore image processing with the flexibility provided by setting the shape parameter.

23. Histograms of the distribution of coefficients were computed by applying the custom transform to a collection of images. Quantization tables for specific settings of the shape parameter were generated from the entropy values for the corresponding histograms.

24. The effect of setting the shape parameter of the basis function and using the proposed quantization tables when applied to image compression was investigated. The results were compared to a standard DCT transform with JPEG type quantization. Performance measures for three reference pictures were provided using two different metrics to measure errors:

    (a) PSNR to measure the signal's fidelity without considering its content.
    (b) MSSIM as a picture metric where the data is treated as the visual information it contains.

    The benchmark test indicated better PSNR against compression factor for the custom method than for the reference DCT with JPEG quantization. The performance difference between the two transforms seemed to be much smaller when measured using the MSSIM.

25. A strategy for wavelet shrinkage based on partitioning of signals with varying smoothness properties was introduced. The motivation was to investigate this strategy for compression of discretized geometry, such as parametric curves and surfaces, where varying smoothness measures and singularities can occur.

26. Wavelet shrinkage by Lorentz curve thresholding was benchmarked for a selection of wavelets applied to four test functions with different smoothness measures and compared to the performance on a "global" signal composed by joining together the four test functions. The benchmarked criteria were to specify the accepted error measure and to specify the desired compression rates. The main results were:

    (a) The number of wavelet coefficients which could be discarded while meeting the criteria was found to depend on the signal's characteristics and the choice of wavelet. The best performing wavelet according to the selected criteria was found to be different for the selected types of signals.

    (b) Higher compression rates and less error, respectively, was achieved for the individual partitions than for the composite "global" function.

    A notable increase in PSNR against compression ratio was shown for the "global" signal by transforming the four test functions separately, using their respective "best" performing wavelets, and applying individual shrinking of their coefficients. The compressed signal was generated by joining together the compressed individual partitions. This was summarized as follows:

    (c) Better performance of wavelet shrinkage was possible for signals with varying smoothness properties and isolated singularities by using the partitioning-based approach than using one type of wavelet for the complete signal.

2.3 Remarks and notes for future work

Here we provide our remarks and some topics and ideas for future work based on the results presented in this dissertation.

1. The papers [21, 23, 24, 25, 27] have been presented by the author of this thesis at the conferences corresponding to the conference proceedings which they have appeared in.

2. A new family of basis functions under the umbrella of GERBS have emerged during the period of the research project. The C^∞-smooth logistic ERBS were introduced in [40, 129] as a promising alternative to ERBS by featuring simple explicit computation without requiring an integration step. The ERBS proper has been used in the experiments described in this dissertation unless specified otherwise.

3. The capability of a compression system is sometimes characterized in the literature in terms of *compression ratio* defined as

$$c_r = \frac{\text{source coder input size}}{\text{source coder output size}}.$$

This definition is somewhat ambiguous [3] and is depending on the specific compression method and data type it is employed for. As an example, bits/pixel is a common measure of the size for still pictures and video frames [3].

The measure of compression in chapter 3 is denoting the size of the compressed signal, relative to the size of the original uncompressed signal, in percent. Thus, as an example, a measure of 10% means that the size of the compressed signal is 10% of the size of the original signal.

Compression in chapter 9 has been measured by considering the relative difference in size of the original uncompressed image file and the compressed image file. Real compression including the lossless "packing" step has been performed. As an example, a compression factor of 10 indicates that the size of the compressed signal is 10 times less than the original uncompressed signal.

In chapter 10 "compression" is a measure of the number of discarded wavelet coefficients relative to the total number of wavelet coefficients (including the scaling coefficients).

4. Mean squared error (MSE), PSNR [13] and structural similarity (SSIM) [120, 122] are examples of image quality measures [121]. SSIM aims to be correlated with quality perception of the human visual system (HVS) [122]. However, the SSIM index has faced some criticism, notably by Dosselmann and Yang, who found statistical evidence of a link between the SSIM and MSE in [47] and established a formal connection between the two in [48]. Furthermore, Horé and Ziou presented an analytic relationship between SSIM and PSNR in [69] and performed a series of tests in order to better understand their performance and differences. They concluded that the PSNR and the SSIM quality measures mainly differ on their degree of sensitivity to image degradations. Winkler and Mohandas regarded SSIM as an engineering approach, based on extraction and analysis of certain features in the image, and stated in [127] that such metrics do not necessarily disregard human vision, but image content and distortion analysis is the concept rather than fundamental vision modeling.

5. GPU hardware has evolved remarkably since the advent of smartphones and tablets. Shader processors are not only available in traditional computers. They can be found in products that were considered "low-end" until recently. The method presented in chapter 7 has been relying on the tessellation shaders which are at present only available in the main OpenGL standard. They have not (yet) been included in the OpenGL|ES [85] or WebGL [71] APIs, which are applicable to embedded- and web-based clients, respectively. However, hardware implementations of geometry- and tessellation shaders for mobile devices have been introduced, see e.g. [97, 106, 96, 99, 107],

and are accessible for developers through DirectX and the Android 5.0 API as an extension pack [62].

6. Algorithms for computing the Bernstein factor matrices and its derivatives, described in chapter 8, have to be implemented. Different implementation strategies depending on the application areas should be devised, benchmarked and compared. One idea which should be investigated further is to utilize template metaprogramming to achieve compile time code optimization.

7. The construction in chapter 8 can be used to provide multivariate local patches by means of higher order simplices for use with the ERBS blending construction.

8. Attempting to change the step in the geometric interpretation of the de Casteljau algorithm presented in chapter 8 such that it provides the linear combination of the rational form of the Bernstein polynomials with weights would be interesting. Furthermore, a conversion of the factorization theorem for Bézier curves to projective coordinates, and trying to pass from the iterative procedure in $\mathbb{R}^{n+1}$ to identification of a single point as one element in a matrix describing the iterative process, are ideas which could be explored further.

9. The rendering method proposed in chapter 7 should be further developed and implemented for irregular grids.

10. Rendering of terrain by using hardware tessellation has been explored in previous work, see e.g. [11, 128]. It would be interesting to compare some existing methods with a spline-based representation and rendering approach, such as the one described in chapter 7, applied to terrain models.

11. A comprehensive study of ERBS knot insertion and methods for generating local functions, based on the findings provided in chapter 4, should be conducted in order to provide strategies and alternatives for various applications.

12. Adaptive methods for partitioning of discretized ERBS-type spline spaces could be developed to facilitate the use of adjustable compression methods based on the findings presented in chapter 10.

13. More sophisticated methods to determine quantization tables for the method proposed in chapter 9 could be developed in order to improve the performance of the construction.

Bibliography

[1] S. Bernstein. Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités. *Communications de la Société Mathématique de Kharkow. 2-ée série*, 13(1):1–2, 1912.

[2] P. Bézier. *Numerical control: Mathematics and Applications*. Wiley series in computing. J. Wiley, London, New York, English language edition, 1972.

[3] V. Bhaskaran and K. Konstantinides. *Image and Video Compression Standards. Algorithms and Architectures.* Kluwer Academic Publishers, Norwell, Massachusetts, USA, 2nd edition, 1997.

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

[5] W. Boehm, G. Farin, and J. Kahmann. A survey of curve and surface methods in CAGD. *Computer Aided Geometric Design*, 1(1):1–60, 1984.

[6] J. Bratlie and R. Dalmo. Motion capture data represented using a blending type spline construction. In Pasheva and Venkov [101], pages 153–157.

[7] J. Bratlie, R. Dalmo, and B. Bang. Evaluation of smooth spline blending surfaces using GPU. In J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curves and surfaces. 8th International Conference*, volume 9213 of *Lecture Notes in Computer Science*, pages 60–69. Springer, 2015.

[8] J. Bratlie, R. Dalmo, and B. Bang. Wavelet compression of spline coefficients. In Dimov et al. [43], pages 246–253.

[9] J. Bratlie, R. Dalmo, and P. Zanaty. Fitting of discrete data with GERBS. In Lirkov et al. [86], pages 569–576.

[10] M. Brovka, J. I. López, J. M. Escobar, J. M. Cascón, and R. Montenegro. A new method for T-spline parameterization of complex 2D geometries. *Engineering with Computers*, 30(4):457–473, 2013.

[11] I. Cantlay. DirectX 11 Terrain tessellation, 2011. Nvidia whitepaper.

[12] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

[13] L. C. Chan and P. Whiteman. Hardware-constrained hybrid coding of video imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 19(1):71–84, 1983.

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

[15] E. Cohen, T. Lyche, and R. F. Riesenfeld. MCAD: Key historical developments. *Computer Methods in Applied Mechanics and Engineering*, 199(5):224–228, 2010.

[16] S. A. Coons. Surfaces for Computer-Aided Design of space forms. Project MAC-TR-41, Massachusetts Institute of Technology, Massachusetts, USA, 1967.

[17] P. Costantini and C. Manni. Curve and surface construction using Hermite subdivision schemes. *Journal of Computational and Applied Mathematics*, 233(7):1660–1673, feb 2010. International Conference on Multivariate Approximation and Interpolation with Applications (MAIA 2007).

[18] P. Costantini and C. Manni. A geometric approach for Hermite subdivision. *Numerische Mathematik*, 115(3):333–369, 2010.

[19] M. G. Cox. The numerical evaluation of B-splines. *J. Inst. Maths. Applics.*, 10:134–149, 1972.

[20] H. B. Curry and I. J. Schoenberg. On pólya frequency functions IV: The fundamental spline functions and their limits. *Journal d'Analyse Math.*, 17:71–107, 1966.

[21] R. Dalmo. Local refinement of ERBS curves. In Pasheva and Venkov [100], pages 204–211.

[22] R. Dalmo. Matrix factorization of multivariate Bernstein polynomials. *International Journal of Pure and Applied Mathematics*, 103(4):749–780, 2015.

[23] R. Dalmo and J. Bratlie. Data approximation using a blending type spline construction. In Pasheva and Venkov [101], pages 147–152.

[24] R. Dalmo and J. Bratlie. Discrete wavelet compression of ERBS. In Lirkov et al. [86], pages 577–584.

[25] R. Dalmo, J. Bratlie, and B. Bang. Performance of a wavelet shrinking method. In Dimov et al. [43], pages 262–270.

[26] R. Dalmo, J. Bratlie, B. Bang, and A. Lakså. Smooth spline blending surface approximation over a triangulated irregular network. *International Journal of Applied Mathematics*, 27(1):109–119, 2014.

[27] R. Dalmo, J. Bratlie, and P. Zanaty. Image processing with LERBS. In Sivasundaram [119], pages 271–278.

[28] R. Dalmo, J. Bratlie, and P. Zanaty. Image compression using an adjustable basis function. *Mathematics in Engineering, Science and Aerospace*, 6(1):25–34, 2015.

[29] P. J. Davis. *Interpolation and approximation*. Dover, New York, 1975.

[30] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational Geometry: algorithms and applications*. Springer-Verlag, Berlin Heidelberg, 2nd edition, 2000.

[31] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[32] C. de Boor. Splines as linear combinations of B-splines. a survey. Technical report, Wisconsin Univ Madison Mathematics Research Center, 1976.

[33] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, New York, 1978.

[34] C. de Boor. B(asic)-spline basics, 1993. In Fundamental Developments of Computer-Aided Geometric Modeling, pp. 27–49, Academic Press, London.

[35] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied mathematical sciences*. Springer-Verlag, New York, revised edition, 2001.

[36] P. de Faget de Casteljau. *Shape mathematics and CAD*, volume 2 of *Mathematics and CAD*. Kogan Page, 120 Pentonville Road, London, N1 9JN, English language edition, 1986.

[37] L. T. Dechevsky. Expo-rational B-splines, 2004. Communication at the Sixth International Conference on Mathematical Methods for Curves and Surfaces, Tromsø, Norway.

[38] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[39] L. T. Dechevsky, A. Lakså, and B. Bang. Expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 27(3):319–362, 2006.

[40] L. T. Dechevsky and P. Zanaty. Smooth GERBS, orthogonal systems and energy minimization. In Pasheva and Venkov [100], pages 135–162.

[41] B. Delaunay. Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS*, (6):793–800, 1934.

[42] J. Deng, F. Chen, and Y. Feng. Dimensions of spline spaces over T-meshes. *Journal of Computational and Applied Mathematics*, 194(2):267–283, oct 2006.

[43] I. Dimov, S. Fidanova, and I. Lirkov, editors. *Numerical Methods and Applications 2014*, volume 8962 of *Lecture Notes in Computer Science*. Springer, 2015.

[44] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, New Jersey, 1976.

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

[46] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.

[47] R. Dosselmann and X. D. Yang. An empirical assessment of the structural similarity index. In *Proceedings of Electrical and Computer Engineering, 2009*, pages 112–116. IEEE, 2009.

[48] R. Dosselmann and X. D. Yang. A comprehensive assessment of the structural similarity index. *Signal, Image and Video Processing*, 5(1):81–91, 2011.

[49] M. P. Epstein. On the influence of parametrization in parametric interpolation. *SIAM Journal on Numerical Analysis*, 13(2):261–268, 1976.

[50] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[51] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Computer Science and Scientific Computing. Academic Press, San Diego, CA, USA, 4th edition, 1997.

[52] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Academic Press, 5th edition, 2002.

[53] G. Farin, J. Hoschek, and M.-S. Kim. *Handbook of computer aided geometric design*. Elsevier, Amsterdam, 2002.

[54] R. T. Farouki. On the stability of transformations between power and Bernstein polynomial forms. *Computer Aided Geometric Design*, 8(1):29–36, 1991.

[55] R. T. Farouki and V. T. Rajan. On the numerical condition of polynomials in Bernstein form. *Computer Aided Geometric Design*, 4(3):191–216, 1987.

[56] R. T. Farouki and V. T. Rajan. Algorithms for polynomials in Bernstein form. *Computer Aided Geometric Design*, 5(1):1–26, 1988.

[57] J. Ferguson. Multivariable curve interpolation. *Journal of the Association for Computing Machinery*, 11(2):221–228, 1964.

[58] M. S. Floater and K. Hormann. Surface parameterization: A tutorial and survey. In N. A. Dodgson, M. S. Floater, and M. A. Sabin, editors, *Advances in Multiresolution for Geometric Modelling*, Mathematics and Visualization, Berlin Heidelberg, 2005. Springer-Verlag.

[59] T. A. Foley. Interpolation with interval and point tension controls using cubic weighted ν-splines. *ACM Transactions on Mathematical Software*, 13(1):68–96, 1987.

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

[61] R. N. Goldman and T. Lyche. *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*. Geometric Design Publications. SIAM, 3600 University City Science Center, Philadelphia, PA 19104-2688, 1993.

[62] Google. Android 5.0 API. `https://developer.android.com/about/versions/android-5.0.html\#AndroidExtensionPack`.

[63] W. J. Gordon and R. Riesenfeld. B-spline curves and surfaces. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 95–126. Academic Press, New York, 1974.

[64] R. C. Gupta. Second order interpolation in Indian mathematics up to the fifteenth century. *Ind. J. Hist. Sci.*, 4(1-2):86–98, 1969.

[65] B. Haavardsholm, J. Bratlie, and R. Dalmo. Surface deformation over flexible joints using spline blending techniques. In Sivasundaram [119], pages 377–383.

[66] P. J. Hartley and C. J. Judd. Parametrization of Bézier-type B-spline curves and surfaces. *Computer Aided Design*, 10(2):130–134, 1978.

[67] P. J. Hartley and C. J. Judd. Parametrization and shape of B-spline curves for CAD. *Computer Aided Design*, 12(5):235–238, 1980.

[68] Ø. Hjelle and M. Dæhlen. *Triangulations and Applications*. Mathematics and visualization. Springer-Verlag, Berlin Heidelberg, 2006.

[69] A. Horé and D. Ziou. Is there a relationship between peak-signal-to-noise ratio and structural similarity index measure? *IET Image Processing*, 7(1):12–24, 2013.

[70] L. Howes and A. Munshi. The OpenCL API specification (Version 2.0), 2014.

[71] D. Jackson. WebGL specification (Version 1.0.3), October 2014.

[72] E. S. Kennedy. The Chinese-Uighur calendar as described in the islamic sources. *ISIS*, 55(4):435–443, 1964.

[73] P. Kiciak. Spline surfaces of arbitrary topology with continuous curvature and optimized shape. *Computer Aided Design*, 45(2):154–167, 2013.

[74] Y. Kineri, M. Wang, H. Lin, and T. Maekawa. B-spline surface fitting by iterative geometric interpolation/approximation algorithms. *Computer Aided Design*, 44(7):697–708, 2012.

[75] M. Kline. *Mathematical Thought from Ancient to Modern Times*, volume 2. Oxford University Press, New York, 1972.

[76] M.-J. Lai and L. L. Schumaker. *Spline Functions on Triangulations*, volume 110 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge CB2 8RU, UK, 2007.

[77] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[78] A. Lakså. ERBS-surface construction on irregular grids. In Pasheva and Venkov [100], pages 113–120.

[79] A. Lakså. Construction and properties of non-polynomial spline curves. In Sivasundaram [119], pages 545–554.

[80] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. 2:11, 2004. ISSN 1504-4653.

[81] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[82] A. Lakså, B. Bang, and A. R. Kristoffersen. GM_lib, a C++ library for geometric modeling. Technical report, Narvik University College, Narvik, Norway, 2006.

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

[84] P.-J. Laurent and P. Sablonnière. Pierre Bézier: An engineer and a mathematician. *Computer Aided Geometric Design*, 18(7):609–617, 2001.

[85] J. Leech. OpenGL® ES specification (Version 3.1), January 2015.

[86] I. Lirkov, S. Margenov, and J. Waśniewski, editors. *Large-Scale Scientific Computing 2013*, volume 8353 of *Lecture Notes in Computer Science*. Springer, 2014.

[87] C. T. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, The University of Utah, 1987. Department of Mathematics.

[88] C. T. Loop and T. D. DeRose. A multisided generalization of Bézier surfaces. *ACM Transactions on Graphics*, 8(3):204–234, 1989.

[89] C. T. Loop and T. D. DeRose. Generalized B-spline surfaces of arbitrary topology. *Computer Graphics*, 24(4):347–356, 1990.

[90] D. J. McConalogue. A quasi-intrinsic scheme for passing a smooth curve through a discrete set of points. *The Computer Journal*, 13:392–396, 1970.

[91] E. Meijering. A chronology of interpolation. *Proceedings of the IEEE*, 90(3):319–342, 2002.

[92] Microsoft® corporation. Direct3D 11 features, 2009. `http://msdn.microsoft.com/en-us/library/ff476342(VS.85).aspx`.

[93] A. Munshi. The OpenCL C language specification (Version 2.0), 2014.

[94] O. Neugebauer. *A History of Ancient Mathematical Astronomy*. Springer-Verlag, Berlin, Germany, 1975.

[95] NFR-201511. Dreamworld - sømløs integrering av nettbaserte 3d-spill og sosiale nettverk, 2009. NRF (Norges forskingsråd / The Research Council of Norway) sponsored project, Verdikt program project ID: 201511.

[96] Nvidia. Bringing high-end graphics to handheld devices, 2011. http://www.nvidia.com/content/PDF/tegra_white_papers/Bringing_High-End_Graphics_to_Handheld_Devices.pdf.

[97] Nvidia. Kepler to mobile, 2013. http://blogs.nvidia.com/blog/2013/07/24/kepler-to-mobile/.

[98] Nvidia. CUDA C programming guide v.6.5, 2014.

[99] Nvidia. Nvidia Tegra K1: A new era in mobile computing, 2014. http://www.nvidia.com/content/PDF/tegra_white_papers/tegra-K1-whitepaper.pdf.

[100] V. Pasheva and G. Venkov, editors. *39th International conference applications of mathematics in engineering and economics AMEE13*, volume 1570 of *AIP Conference Proceedings*. AIP Publishing, 2013.

[101] V. Pasheva and G. Venkov, editors. *40th International conference applications of mathematics in engineering and economics AMEE14*, volume 1631 of *AIP Conference Proceedings*. AIP Publishing, 2014.

[102] A. Pedersen, R. Dalmo, and J. Bratlie. Modeling terminal ballistics using blending-type spline surfaces. In Sivasundaram [119], pages 796–803.

[103] T. K. Peucker, R. J. Fowler, J. J. Little, and D. M. Mark. The triangulated irregular network. In *Proceedings of Auto Carto 4*, volume 2, pages 96–103, Reston, Virginia, USA, 1978.

[104] T. K. Peucker, R. J. Fowler, J. J. Little, and D. M. Mark. The triangulated irregular network. In *Proceedings of American Society of Photogrammetry: Digital Terrain Models (DTM) Symposium*, pages 516–540, St. Louis, Missouri, USA, 1978.

[105] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline Techniques*. Mathematics and visualization. Springer-Verlag, Berlin Heidelberg, 2002.

[106] Qualcomm. Qualcomm technologies announces next generation Qualcomm Snapdragon 805, 2013. https://www.qualcomm.com/news/releases/2013/11/20/qualcomm-technologies-announces-next-generation-qualcomm-snapdragon-805.

[107] Qualcomm. Qualcomm Snapdragon 805 processor product brief, February 2014. https://www.qualcomm.com/media/documents/files/snapdragon-805-processor-product-brief.pdf.

[108] P. Sablonnière, D. Sbibih, and M. Tahrichi. Chordal cubic spline quasi interpolation. In J.-D. Boissonnat, P. Chenin, A. Cohen, C. Gout, T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curves and Surfaces*, volume 6920 of *Lecture Notes in Computer Science*, pages 603–611. Springer, 2010.

[109] H. Schäfer, M. Nießner, B. Keinert, M. Stamminger, and C. Loop. State of the art report on real-time rendering with hardware tessellation. *Eurographics 2014 - State of the Art Reports*, pages 93–117, 2014.

[110] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. Part A — On the problem of smoothing or graduation. A first class of analytic approximation formulae. *Quarterly of Applied Mathematics*, IV(1):45–99, 1946.

[111] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. Part B — On the problem of osculatory interpolation. A second class of analytic approximation formulae. *Quarterly of Applied Mathematics*, IV(2):112–141, 1946.

[112] I. J. Schoenberg. On spline functions. In O. Shisha, editor, *Inequalities*, volume I, New York and London, 1967. Academic Press.

[113] L. L. Schumaker. *Spline Functions*. Cambridge University Press, Cambridge CB2 8RU, UK, 3rd edition, 2007.

[114] H. L. Seal. Graduation by piecewise cubic polynomials: A historical review. *Blätter Deutsc. Gesellschaft Versich.math.*, 15:89–114, 1981.

[115] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 477–484, New York, NY, USA, 2003. ACM.

[116] M. Segal and K. Akeley. The OpenGL graphics system: A specification (Version 4.0 (Core Profile)), March 2012.

[117] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[118] C. E. Shannon. Communication in the presence of noise. *Proceedings of the Inst. Radio Eng.*, 37(1):10–21, 1949.

[119] S. Sivasundaram, editor. *ICNPAA 2014 World Congress: 10th International conference on Mathematical Problems in Engineering, Aerospace and Sciences*, volume 1637 of *AIP Conference Proceedings*. AIP Publishing, 2014.

[120] Z. Wang and A. C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002.

[121] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[122] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[123] E. T. Whittaker. On the functions which are represented by the expansions of the interpolation-theory. *Proceedings of the Royal Society of Edinburgh*, 35:181–194, 1915.

[124] E. T. Whittaker and G. Robinson. *A short course in interpolation*. Blackie and Son Limited, 50 old Bailey, London, 1923.

[125] J. M. Whittaker. The "Fourier" theory of the cardinal function. *Proceedings of the Edinburgh Mathematical Society*, 1:169–176, 1927–1929.

[126] J. M. Whittaker. On the cardinal function of interpolation theory. *Proceedings of the Edinburgh Mathematical Society*, 1:41–46, 1927–1929.

[127] S. Winkler and P. Mohandas. The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE Transactions on Broadcasting*, 54(3):660–668, 2008.

[128] E. Yusov and M. Shevtsov. High-performance terrain rendering using hardware tessellation. *Journal of WSCG*, 2011:85–92, 2011.

[129] P. Zanaty. *Application of Generalized Expo-Rational B-splines in Computer Aided Design and Analysis*. PhD thesis, University of Oslo, 2014.

Part II

Modeling and representation

3 Fitting of discrete data with GERBS

Jostein Bratlie, Rune Dalmo, and Peter Zanaty

This chapter is a reprint of [1]

Abstract — In this paper, we present a study of fitting discrete data with generalized expo-rational B-splines (GERBS). We investigate different ways to determine interpolation knots and generate GERBS local curves by partitioning the parametric space and solving a corresponding least-squares fitting problem. We apply our technique to discrete evaluations of continuous synthetic benchmark functions and compare the resulting GERBS to the original data with respect to errors and performance.

3.1 Introduction

In this work we investigate the properties of fitting generalized expo-rational B-splines, introduced in [2], to regular discretized data. GERBS is a family of blending type spline constructions, where local functional coefficients are blended by GERBS basis functions. The choice of basis functions and the local enrichment functions determines the local and hence the global approximation properties of the resulting space.

One of the intrinsic properties of the GERBS bases are the minimal support of the basis functions, which allows for a simple approximation technique; instead of storing the individual data points, and then blending the corresponding local functions together, node by node, we can choose the interpolation knots and the accompanying local functions freely, depending on the data itself.

Using this, we investigate various techniques to partition the parametric space of the GERBS across the discrete data by changing the interpolation knots and simultaneously adjusting the corresponding coefficient functions. In addition, we look at the performance of the different constructions with respect to approximation.

Many papers have been published on the topic of data fitting, data reduction, compression and smoothing with B-splines using various methods. We mention here the knot removal technique presented by Lyche and Mørken in [5] and with a different approach by Eck and Hadenfeld in [3], and the shape-preserving knot removal method by Schumaker and Stanley in [8]. We also mention the work done by Saux and Daniel in [6, 7] on estimating criteria for fitting and data reduction of polygonal curves using B-splines. We leave these topics for now and focus on a few simple methods for constructing GERBS local functions.

In section 3.2 we start by giving a brief introduction to GERBS and its construction, as well as the partitioning and fitting setup we use throughout the article. Then in section 3.3 we describe the different partitioning algorithms and then follows the description of the fitting method in section 3.4. Finally in section 3.5 we give some concluding remarks where we discuss our findings and future work.

3.2 Preliminaries

3.2.1 GERBS basis functions

Consider a strictly increasing knot vector $\vec{t} = \{t_k\}_{k=0}^{n+1}$, $t_0 < t_1 < \cdots < t_{n+1}$, $n \in \mathbb{N}$. The definition of the j-th GERBS is defined in [2] as follows.

$$B_j(t) = \begin{cases} F_j(t), & \text{if} \quad t \in (t_{j-1}, t_j], \\ 1 - F_{j+1}(t), & \text{if} \quad t \in (t_j, t_{j+1}), \\ 0, & \text{if} \quad t \in (-\infty, t_{j-1}] \cup [t_{j+1}, +\infty), \end{cases}$$

$j = 1, \ldots, n,$

where $\{F_i\}_{i=1}^{n+1}$ is a system of cumulative distribution functions such that for F_i, $i = 1, \ldots, n$,

1. the right-hand limit $F_i(t_{i-1}+) = F_i(t_{i-1}) = 0$,

2. the left-hand limit $F_i(t_i-) = F_i(t_i) = 1$,

3. $F_i(t) = 0$ for $t \in (-\infty, t_{i-1}]$,

4. $F_i(t) = 1$ for $t \in [t_i, +\infty)$, and $F(t)$ is monotonously increasing, possibly discontinuous, but left-continuous for $t \in [t_{i-1}, t_i]$.

3.2.2 GERBS curves

Generalized expo-rational B-splines provide a blending type construction, where local functions at each knot are blended together by sufficiently smooth basis functions

$$s(t) = \sum_{i=1}^{n} \ell_i(t - t_i) B_i(t), \tag{3.1}$$

where $\vec{t} = \{t_k\}_{k=0}^{n+1}$ is a strictly increasing knot vector, and each basis function $B_j(t)$ is supported on (t_{j-1}, t_{j+1}) while possessing a Dirac property $B_j(t_i) = \delta_{ij}$.

The local functions ℓ_i throughout this paper shall be Taylor expanding polynomials up to a multiplicity μ_i

$$\ell_i(t - t_i) = \sum_{j=0}^{\mu_i} c_{i,j} \frac{(t - t_i)^j}{j!},$$

and the corresponding GERBS base $B_i(t)$ is required to have vanishing derivatives of order up to, including, μ_i. For the rest of the paper we will use the expo-rational B-spline basis described in [4] which is capable of transfinite Hermite interpolation, i.e. all of its derivatives vanish at all knots.

The knots ($\vec{t}$) and the multiplicities ($\vec{\mu}$) together define a spline space, where the coefficients ($\vec{c}$) have a natural meaning corresponding to a Hermite interpolation problem.

3.2.3 Partitioning and fitting

In digital systems we often have to deal with continuous (analogue) input data. The way it is handled is that the analogue signal is converted to digital by sampling and quantizing (digitizing) and the resulting raw digital data is being used instead of the original data. Often it is a requirement to produce outputs which are either continuous or sampled at a higher rate than that of the input data, this problem translates into interpolation/extrapolation or approximation problems depending on the requirements.

In the current article, we are interested in comparing different strategies for representing uniformly sampled uni-variate functions with the use of GERBS based approximation, see Figure 3.1. The partitioning algorithms work on a sampled data of two benchmark functions,

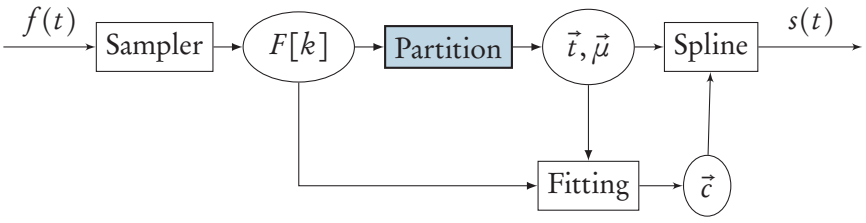

Figure 3.1: Block diagram showing how a continuous signal is discretized and represented as a spline via partitioning and fitting of the data.

given by

$$f_1(t) = \begin{pmatrix} \ln(t+1) \\ -t\sin(2t+1)) \end{pmatrix}, \qquad t \in [0,1],$$

$$f_2(t) = \begin{pmatrix} t \\ t\sin(\frac{1}{t}) \end{pmatrix}, \qquad t \in [0.01, 0.5],$$

and their task is to select the knot configuration and the corresponding local multiplicities of the spline space.

Then a fitting algorithm obtains the coefficients to the spline representation, finally we compare the resulting splines with both the original continuous benchmark functions $f_1(t)$, $f_2(t)$ and their sampled discrete versions $F_1[k]$, $F_2[k]$ and discuss some properties of the resulting transformations.

3.3 Partitioning algorithms

In order to fit a GERBS construction (see section 3.2.2) to discrete data, it is necessary to decide where to place the interpolation knots ($\vec{t}$) and to decide the corresponding multiplicities ($\vec{\mu}$) of the local functions. This can be done in a number of different ways. We describe three different algorithms for constructing local curves.

3.3.1 Uniform partitioning

As a starting point with uniform sampling we define a knot for each discrete data point in $F[k]$. Next, the number of knots is reduced by selecting a subset of F in order to define the spline space. We add as a note here that the data is assumed to be appropriate for selection. (In some cases it is common to smooth the data before selecting to reduce errors or avoid problems related to oscillation.) It is possible to increase the degree by selecting derivatives for each knot. We illustrate uniform partitioning with three different examples:

1. Fixed sample rate
2. Specified number of knots
3. Parametric stride

In the first case, the sample rate simply states how many knots to skip between the selected knots. Hence, a sample rate of two selects every second knot, whereas a sample rate of 10 selects every 10th knot.

The number of knots in the second case defines the size of the resulting knot vector. This implies a computation of the sampling rate depending on the number of elements in F.

We consider parametric stride, where we select knots equidistant in parametric space, in the current article.

3.3.2 Curvature based partitioning

Moving away from uniform partitioning, we describe in brief a naive, curvature extrema based partitioning approach. From the discrete function $F[k]$, $k = 1,\ldots,M$, we compute for each interior knot t_i, $i = 2,\ldots,M-1$ the radius of circumscribed circle of triangle $R[i] = R_{circ}\triangle(F[i-1],F[i],F[i+1])$. These values correspond to the curvature of the curve at the corresponding interior points. Next, we select the extrema of these values as they, together with the two endpoints constitute the points of interest (for more details on feature point selection consult [9, 7]).

To be able to scale the method, the resulting set of feature points is processed further. Feature points that are too close are filtered out and new feature points are introduced uniformly between feature points that were too far away.

3.3.3 Partitioning based on inflexion

We look at two different approaches based on relative angular changes in the discrete data set. In both cases we consider the angle between the two vectors spanning a sample point. Where we in the first approach consider the change in the angle by tracing the curve, we start by sorting the angles into different buckets in the other.

Inline traced partitioning:

In the first variation we look at an approach where we consider the linear interpolation between two neighboring data points to be a vector which provides a first derivative in one point. Given $\vec{a}$ and $\vec{b}$, two vectors, we use the dot product between vectors and the angular difference, γ, of $\vec{a}$ and $\vec{b}$

$$\cos(\gamma) = \frac{<\vec{a},\vec{b}>}{|\vec{a}|\left|\vec{b}\right|} \tag{3.2}$$

Given the discrete data set $F[k]$, and an empty set $\mathbf{q}$ to store the detected feature points. Apply (3.2) to the vectors $\mathbf{a} = p_1 - p_0$ and let $\mathbf{b}$ "run" along the curve, starting with $\mathbf{b} = p_2 - p_1$, then $\mathbf{b} = p_3 - p_2$, and so on. By comparing the results of applying (3.2), whenever the sign of the gradient of the resulting "curve" changes, we find a point of inflection on the curve given by linear interpolation between points in $\mathbf{p}$.

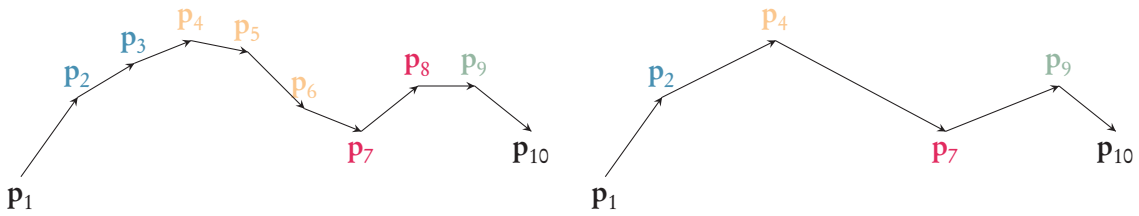

Figure 3.2: Ten points and the linear interpolation in-between drawn as vectors. Left: The ten sample points. Interior sample points sorted into three different buckets. Right: The interior feature points kept after partitioning.

BUCKET BASED PARTITIONING:

The second approach is to do an angular difference based partitioning by segmenting knots into buckets, each bucket corresponding to a range subset of possible angular differences between the forward and the backward edge. From the discrete function $F[k]$, $k = 1,\ldots,M-1$, we compute for each interior knot the angle between the two adjacent vectors, $\vec{a}$ and $\vec{b}$, where $a = F[k+1]-F[k]$ and $b = F[k+2]-F[k+1]$. We sort the angles and divide the knots into equal sized buckets, this can be seen in Figure 3.2. Next we run along the curve selecting feature knots, where if following knots is belonging to the same bucket, only the first is kept as a feature knot.

In addition to the found interior knots the end-point knots are also kept as feature knots. Finally, the resulting set of feature points is processed further, feature points that are too close are filtered out.

3.4 FITTING

Once the interpolation space is set up, by defining the knot vectors ($\vec{t}$) and the multiplicities ($\vec{\mu}$) the problem of finding the coefficients to best match the given discrete data set $F[k], k = \{k_1, k_2, \ldots, k_M\}$ of M points remains. For this purpose we will use the best L_2 approximant, given by the coefficient minimizing the mean squared errors

$$||S-F||^2 = \sum_{i=1}^{M} |s(k_i)-F[k_i]|^2,$$

the coefficients are obtained by the usual technique of solving least squares problems. We restrict our investigation to cases where the fitting problem is not ill-posed.

Figure 3.3 shows the performance of the considered methods introduced in section 3.3. The x-axis shows the percentage of the original data that is being used while the y-axis displays the signal-to-noise ratio (SNR) measured in dB, defined as

$$\text{SNR} = 10\log_{10}\left(\frac{||F||^2}{||F-S||^2}\right),$$

where F stands for the original discrete data and S represent the reconstructed data and $||\cdot||^2$

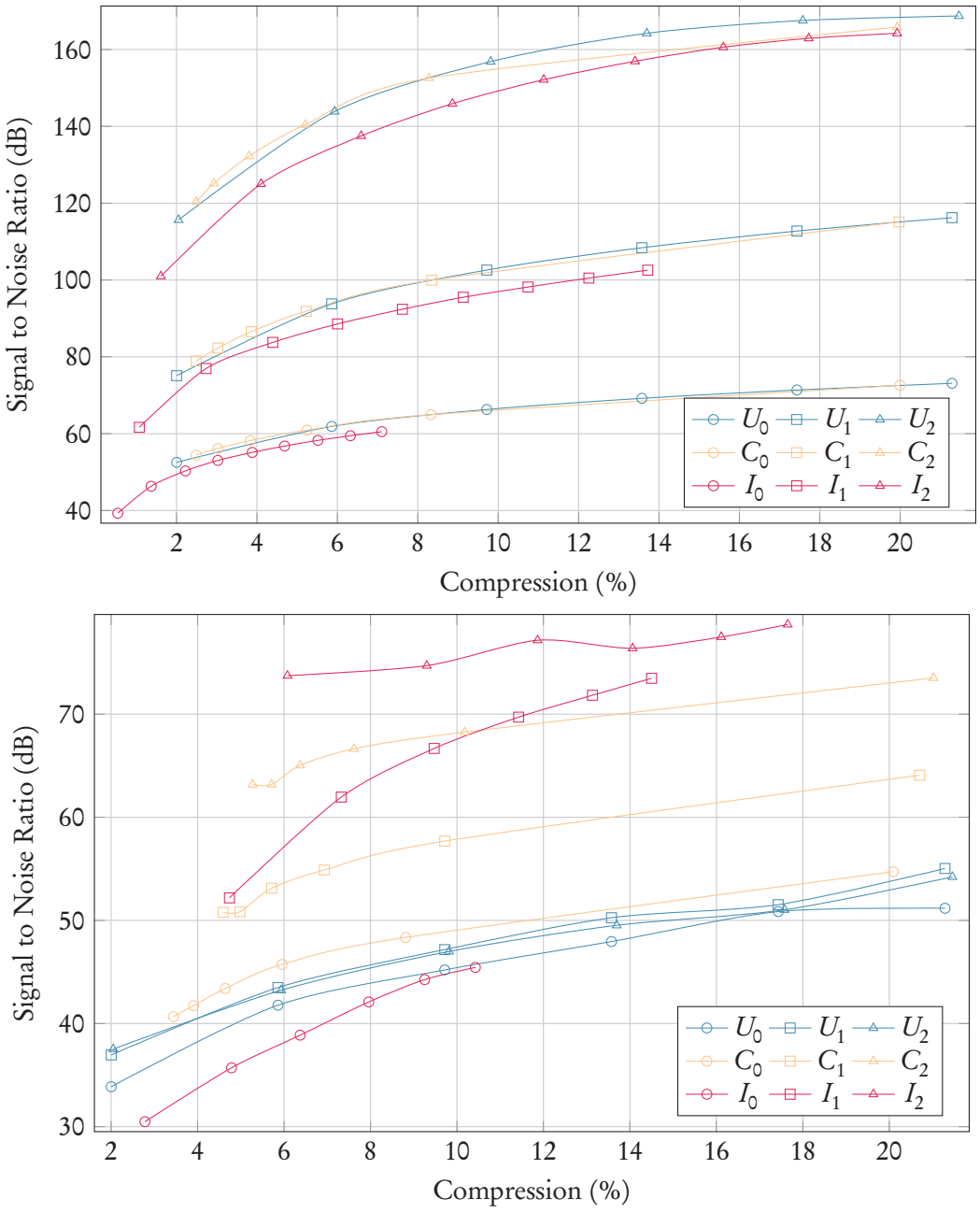

Figure 3.3: Error rates for the smooth (top) and oscillating (bottom) synthetic benchmarks. The blue lines U_0,U_1,U_2 represent the uniform algorithm, the orange lines C_0,C_1,C_2, stand for the curvature based refining method, while the purple lines I_0,I_1,I_2, show the performance of the bucketing algorithm based on angles. The lower indices correspond the applied uniform multiplicity, μ_i in (3.1).

stands for the square of the usual L_2 norm.

3.5 Concluding remarks

Our technique is local, hence there is a small bandwidth in the resulting matrix in the least squares fitting, which in turn gives a computational advantage over global methods, i.e. classical polynomial B-splines.

The computational complexity of each feature detection method is linear and readily parallelizable, similarly since the splines are local the reconstruction and evaluation can also be done parallel.

We note that there is a trade-off between smoothing and interpolation which can be adjusted depending on how confident we are in the data and the condition number of the fitting. Furthermore, the smoothness of the resulting curve can be easily adjusted e.g. to fulfill a smoothness criteria of the underlying physics of the discrete sample points.

The primary utilization is to reduce an original data set and use GERBS type splines to represent the final data. From the two synthetic benchmarks we can conclude that the two types of feature extraction coupled with the coloring or the refining extension allowed for a construction of a series of tune-able spline spaces which performed at least as good as the least squares fitting for smooth inputs and proved to be much more stable for oscillating irregular input data.

3.5.1 Future work

Future work related to applications includes the extension of the current study to applications in cartography and animation data, including the adaptation of the presented ideas to industry standard representations used there, i.e., Catmull-Rom splines in animations and Bézier curves in cartography.

The locality of the method makes it a suitable candidate for streaming data, one particular potential area of use can be in massive multiplayer online (MMO) games within the computer games industry, where large amounts of data of similar structure has to be handled real time. Transferring data over a limited bandwidth, especially for relatively large discrete data sets, translates to simply transferring coefficients via the networks, since the coefficient alone are enough to reconstruct data from a sender on the receiver's end.

Finally, to put a last note for future work, we believe more sophisticated methods for partitioning of the parametric space could enhance the results much further. It could be interesting to apply well studied principles for data reduction, such as (shape-preserving) knot removal or those based on features and criteria of the original data.

References

[1] J. Bratlie, R. Dalmo, and P. Zanaty. Fitting of discrete data with GERBS. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Large-Scale Scientific Computing 2013*, volume 8353 of *Lecture Notes in Computer Science*, pages 569–576. Springer, 2014.

[2] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[3] M. Eck and J. Hadenfeld. Knot removal for B-spline curves. *Computer Aided Geometric Design*, 12(3):259–282, 1994.

[4] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[5] T. Lyche and K. Mørken. Knot removal for parametric B-spline curves and surfaces. *Computer Aided Geometric Design*, 4(3):217–230, 1987.

[6] E. Saux and M. Daniel. Estimating criteria for fitting B-spline curves: Application to data compression. In S. Klimenko and E. Shikin, editors, *Proceedings of GraphiCon '98*. Moscow State University, 1998.

[7] E. Saux and M. Daniel. Data reduction of polygonal curves using B-splines. *Computer Aided Design*, 31(8):507–515, 1999.

[8] L. L. Schumaker and S. S. Stanley. Shape-preserving knot removal. *Computer Aided Geometric Design*, 13(9):851–872, 1996.

[9] K. Thapa. Data compression and critical points detection using normalized symmetric scattered matrix. In *Proceedings of Auto Carto 9*, pages 78–89, Baltimore, Maryland, USA, 1989.

4 Local refinement of ERBS curves

Rune Dalmo

This chapter is a reprint of [4]

Abstract — Expo-rational B-splines (ERBS) provide a blending type construction where local functions at each knot are blended together by infinitely smooth basis functions. In this work we consider some specific ERBS curves that are approximations of parametric curves. We study local refinement to increase flexibility by inserting local control curves at points of interest on the ERBS curve.

Inserting knots into an existing B-spline knot vector results in a new spline space which contains the original spline space as a sub-space. In contrast to B-splines, knot insertion with ERBS results in a rational local function. We investigate methods to generate local curves of different brands. Using this, we blend extra local curves with the original ERBS curve, by knot insertion, and compare the differences with respect to geometric shape and approximation errors.

4.1 Introduction

A number of different approaches to local refinement of B-splines has been explored in previous works. We mention here knot insertion, approached differently by Boehm [1] and Cohen, Lyche and Riesenfeld [3], T-splines [12], LR-splines [7], subdivision of various kinds (e.g. the recursive method by Catmull and Clark in [2]) and macro element spaces [11]. Most of these techniques are applicable to ERBS[10, 6]. However, the ERBS blending construction, using local functions as coefficients, are different from the well-known B-spline type blending construction. To the best of our knowledge, applying local refinement to ERBS has not been studied in detail yet.

Since knot insertion seems to have played an important role for establishing B-splines and NURBS as common tools in applications for CAGD, we will focus on this particular topic in this preliminary study.

4.2 Preliminaries

4.2.1 Expo-rational B-splines

Let $\mathbf{t} = \{t_k\}_{k=0}^{n+1}$ be a strictly increasing knot vector. The expo-rational B-spline associated with t_{k-1}, t_k and t_{k+1} is defined in [10, 6]. We consider here the scalable subset, proposed in [9], where the integrals are independent of the knot vector:

$$B_k(t) = \begin{cases} S_{k-1}\int_0^{\omega_{k-1}(t)} \psi_{k-1}(s)ds, & t_{k-1} < t \le t_k, \\ S_k \int_{\omega_k(t)}^1 \psi_k(s)ds, & t_k < t < t_{k+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{4.1}$$

where $\omega_k(t) = \frac{t-t_k}{t_{k+1}-t_k}$, $\psi(s) = e^{-\beta \frac{|s-\lambda|^{(1+\gamma)\alpha}}{(s(1-s)^\gamma)^\alpha}}$, and the scaling factor is $S_k = \left(\int_0^1 \psi(s)ds\right)^{-1}$, where $\alpha > 0, \beta > 0, \gamma > 0, 0 \le \lambda \le 1$. The ERBS shares three of its five basic properties with the linear B-spline; partition of unity, minimal support and that it interpolates its coefficient in its central knot. In addition, all of its derivatives are zero in every knot, and it is C^∞-smooth on $\mathbb{R}$.

We note here that the ERBS index number indicates the central knot, where it interpolates its local function (the peak of the basis function), contrary to the B-spline index number which indicates at which knot the B-spline "starts".

An ERBS function $f(t)$ is a blending type construction where local functions are interpolated at each knot. It is defined on $(t_1, t_n]$ by

$$f(t) = \sum_{k=1}^{n} \ell_k(t) B_k(t), \ t \in (t_1, t_n], \tag{4.2}$$

where the local functions $\ell_k(t)$, defined on (t_{k-1}, t_{k+1}), are scalar-, vector- or point-valued. The ERBS Hermite interpolation properties (see Theorem 2.4 in [9]) states that an ERBS function interpolates the values and all existing derivatives of its local functions in their

associated knots.

4.2.2 Knot insertion

Knot insertion is the process of inserting new knots into an existing knot vector and compute new coefficients for the splines which are non-zero in the affected knot intervals. This way the new and finer spline space contains the original coarser spline space. We mention here the Oslo algorithm [3] by Cohen, Lyche and Riesenfeld, which is appropriate when inserting more than a few knots at a time, and provide Boehm's method [1], to insert one knot at a time, in the following lemma:

Lemma 4.1. (Boehm's method). *Let* $\hat{\mathbf{t}} = (\hat{t}_j)_{j=1}^{n+d+1}$ *be a given knot vector and let* $\mathbf{t} = (t_i)_{i=1}^{n+d+2}$ *be the knot vector obtained by inserting a knot* z *in* $\hat{\mathbf{t}}$ *in the interval* $[\hat{t}_k, \hat{t}_{k+1}]$. *If*

$$f = \sum_{j=1}^{n} \hat{c}_j B_{j,d,\hat{\mathbf{t}}}(t) = \sum_{i=1}^{n+1} c_i B_{i,d,\mathbf{t}}(t),$$

where $B_{j,d,\hat{\mathbf{t}}}(t)$ *is the* jth *B-spline of degree* d *on the knot vector* $\hat{\mathbf{t}}$, *defined as*

$$B_{j,d,\hat{\mathbf{t}}} = \tfrac{t-\hat{t}_j}{\hat{t}_{j+d}-\hat{t}_j} B_{j,d-1,\hat{\mathbf{t}}}(t) + \tfrac{\hat{t}_{j+d+1}-t}{\hat{t}_{j+d+1}-\hat{t}_{j+1}} B_{j+1,d-1,\hat{\mathbf{t}}}(t),$$

for all real numbers t, *with*

$$B_{j,0,\hat{\mathbf{t}}}(t) = \begin{cases} 1, & \text{if } \ \hat{t}_j \le t \le \hat{t}_{j+1}; \\ 0, & \text{otherwise}, \end{cases}$$

then $(c_i)_{i=1}^{n+d+1}$ *can be expressed in terms of* $(\hat{c}_j)_{j=1}^{n}$ *through the formulas*

$$c_i = \begin{cases} \hat{c}_i, & \text{if } \ 1 \le i \le k-d; \\ \frac{z-\hat{t}_i}{\hat{t}_{i+d}-\hat{t}_i}\hat{c}_i + \frac{\hat{t}_{i+d}-z}{\hat{t}_{i+d}-\hat{t}_i}\hat{c}_{i-1}, & \text{if } \ k-d+1 \le i \le k; \\ \hat{c}_{i-1}, & \text{if } \ k+1 \le i \le n+1. \end{cases}$$

For details on the topic of knot insertion we refer to [8].

4.3 Local refinement of ERBS curves

Given an existing ERBS curve, as defined in (4.2), where the local functions $\ell_k(t)$ are vector-valued curves. Suppose we generate extra local curves to increase the flexibility whilst shaping the ERBS curve. The extra local curves can be blended with the original ERBS curve. We discuss in brief the blending construction in (4.2) before we change the topic to knot insertion.

4.3.1 The ERBS blending construction

Let us consider (4.2) on the interval (t_k, t_{k+1}). It follows from the ERBS minimal support property that the only non-zero ERBS on that interval are $B_k(t)$ and $B_{k+1}(t)$. But since $B_k(t) + B_{k+1}(t) = 1$, due to the partition of unity property, $B_{k+1}(t)$ can be expressed as $1 - B_k(t)$. We can clearly see that the ERBS function is a blending of two local functions inside the knot interval:

$$\begin{aligned} f(t) &= \ell_k(t)B_k(t) + \ell_{k+1}(t)(1 - B_k(t)) \\ &= \ell_{k+1}(t) + \big(\ell_k(t) - \ell_{k+1}(t)\big)B_k(t), \end{aligned} \tag{4.3}$$

when $t_k < t < t_{k+1}$. By utilizing the fact that $B_k(t_k) = 1$, we write

$$f(t) = \begin{cases} \ell_k(t_k), & \text{if} \quad t = t_k \\ \ell_{k+1}(t) + \big(\ell_k(t) - \ell_{k+1}(t)\big)B_k(t), & \text{if} \quad t_k < t < t_{k+1} \end{cases} \tag{4.4}$$

Deriving (4.4) at $t = t_k, k = 1, \ldots, n$ yields the ERBS Hermite interpolation properties:

$$D^j f(t_k) = D^j \ell_k(t_k), \quad \text{for} \quad k = 1, \ldots, n \quad \text{and} \quad j = 0, 1, 2, \ldots \tag{4.5}$$

Formula (4.5) shows that at $t = t_k, k = 1, \ldots, n$, all derivatives of the ERBS curve are equal to the respective derivatives of the local curve.

4.3.2 Knot insertion on ERBS curves

We mention briefly that the local curve for a new knot, inserted at an existing inner knot, can be obtained by constructing a copy of the existing knot's local curve. This is using the property that ERBS curves interpolate their local curves in their central knots. We will not discuss multiple knots further here, but focus on inserting new knots between existing knots.

Since each local curve $\ell_k(t)$ is supported on (t_{k-1}, t_{k+1}) (see Figure 4.1), it follows that if we modify $\ell_k(t)$ to compensate for a new knot inserted in (t_k, t_{k+1}), the change will influence the ERBS blending on the whole interval (t_{k-1}, t_{k+1}). It is not hard to imagine that such a change would cause a chain reaction invoked by adjusting $\ell_{k-1}(t)$, to compensate for the change of $\ell_k(t)$, which in turn triggers a need to adjust $\ell_{k-2}(t)$, and so on. We therefore apply the restriction of not changing any of the existing local curves in order to achieve local refinement.

In the following theorem, constrained by the mentioned restriction, we use Boehm's method (see Lemma 4.1) to express the new local functions after knot insertion in terms of the existing:

Theorem 4.1. (Knot insertion on ERBS) *Let* $\hat{\mathbf{t}} = (\hat{t}_j)_{j=0}^{n+1}$ *be a given knot vector and let* $\mathbf{t} = (t_i)_{i=0}^{i=n+2}$ *be the knot vector obtained by inserting a knot* z *in* $\hat{\mathbf{t}}$ *in the interval* $(\hat{t}_k, \hat{t}_{k+1})$. *If*

$$f(t) = \sum_{j=1}^{n} \hat{\ell}_j(t) B_{j,\hat{\mathbf{t}}}(t) = \sum_{i=1}^{n+1} \ell_i(t) B_{i,\mathbf{t}}(t),$$

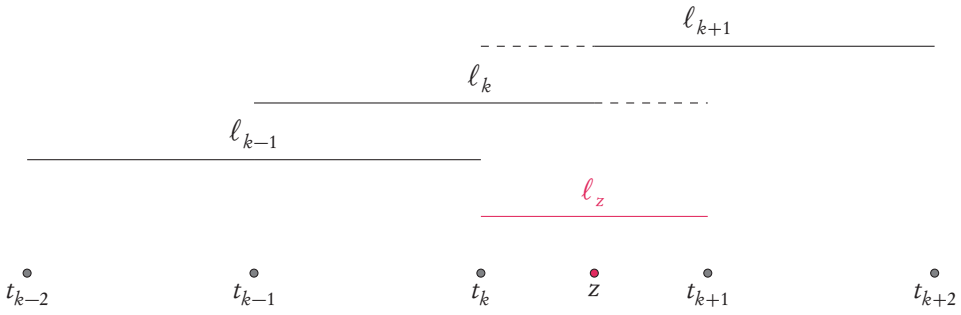

Figure 4.1: Support of the local curves $(\ell_i)_{i=k-1}^{k+1}$ on the knot interval (t_{k-1}, t_{k+1}). Dashed segments indicate those parts of the local curves ℓ_k and ℓ_{k+1} which are not in use after inserting the knot z, with associated local curve ℓ_z, between the knots t_k and t_{k+1}.

where $B_{j,\hat{\mathbf{t}}}(t)$ is the jth ERBS on the knot vector $\hat{\mathbf{t}}$, defined as in (4.1)*, then $(\ell_i(t))_{i=1}^{n+2}$ can be expressed in terms of $(\hat{\ell}_j(t))_{j=1}^{n+1}$ through the formulas*

$$\ell_i(t) = \begin{cases} \hat{\ell}_i(t), & \text{if } 1 \le i < k, \\ \hat{\ell}_i(t) \circ w_k(t) & \text{if } i = k, \\ \hat{\ell}_k(t) + F_{k,\hat{\mathbf{t}}}(t)\left(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\right) & \text{if } i = k+1, \\ \hat{\ell}_{i-1}(t) \circ w_{k+1}(t) & \text{if } i = k+2, \\ \hat{\ell}_{i-1}(t), & \text{if } k+2 < i \le n+1, \end{cases} \tag{4.6}$$

where

$$F_{k,\hat{\mathbf{t}}}(t) = \begin{cases} \frac{B \circ \hat{\omega}_k(t)}{B \circ \omega_k(t)} & \text{if } \hat{t}_k < t \le z, \\ \frac{B \circ \hat{\omega}_k(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)} & \text{if } z < t < \hat{t}_{k+1}, \end{cases} \tag{4.7}$$

with B(t) equal to the first half of the ERBS basis, as defined in the first part of (4.1)*, on the interval* $[0,1]$ *and* $\hat{\omega}_k(t) = \frac{t-\hat{t}_k}{\hat{t}_{k+1}-\hat{t}_k}$, $\omega_k(t) = \frac{t-\hat{t}_k}{z-\hat{t}_k}$, $\omega_{k+1}(t) = \frac{t-z}{\hat{t}_{k+1}-z}$ *are affine functions mapping the parameter value t within the new knot intervals to* $[0,1]$*, and the affine functions $w_k(t) = \frac{t-\hat{t}_{k-1}}{z-\hat{t}_{k-1}}$ and $w_{k+1}(t) = \frac{t-z}{\hat{t}_{k+2}-z}$ are re-parameterizing the local functions, $\hat{\ell}_k(t)$ and $\hat{\ell}_{k+1}(t)$ respectively, to the new knot vector* $\mathbf{t}$*.*

Proof. We look at the first two and the last two formulas in (4.6). Observe that for $j <= k$ we have $\hat{t}_j = t_j$. For $i < k$ it follows that $\ell_i(t) = \hat{\ell}_i(t)$. When $i = k$, $\ell_i(t) = \hat{\ell}_i(t) \circ w_k(t)$, since $\hat{\ell}_i(t)$ must be re-parameterized to the domain $(\hat{t}_{k-1}, z)$ on the new knot vector $\mathbf{t}$. Similarly, we have $t_i = \hat{t}_{i-1}$ for $i > k+1$. So $\ell_i(t) = \hat{\ell}_{i-1}(t)$ for such values of i, except when $i = k+2$, then $\ell_i(t)$ must be re-parameterized to the domain $(z, \hat{t}_{k+2})$, hence, $\ell_i(t) = \hat{\ell}_{i-1}(t) \circ w_{k+1}(t)$.

Next, we discuss the case when $i = k+1$. As it can be seen in (4.3), the only non-zero ERBS on the interval $(\hat{t}_k, \hat{t}_{k+1})$ are $B_{k,\hat{\mathbf{t}}}$ and $B_{k+1,\hat{\mathbf{t}}}$. Using the symmetry property of the ERBS

basis function we rephrase (4.3) to express the ERBS curve on the interval $(\hat{t}_k, \hat{t}_{k+1})$:

$$\begin{aligned} f(t) &= \hat{\ell}_k(t) B \circ (1 - \hat{\omega}_k(t)) + \hat{\ell}_{k+1}(t) B \circ \hat{\omega}_k(t) \\ &= \hat{\ell}_k(t) + B \circ \hat{\omega}_k(t) \Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t) \Big) \end{aligned} \tag{4.8}$$

After knot insertion, the interval $(\hat{t}_k, \hat{t}_{k+1})$ is divided into two new intervals; $(\hat{t}_k, z)$ and $(z, \hat{t}_{k+1})$. The ERBS function in (4.8) expressed on the new knot vector $\mathbf{t}$ is then

$$f(t) = \begin{cases} \hat{\ell}_k(t) + B \circ \omega_k(t) \Big(\ell_z(t) - \hat{\ell}_k(t) \Big) & \text{if} \quad \hat{t}_k < t \leq z, \\ \ell_z(t) + B \circ \omega_{k+1}(t) \Big(\hat{\ell}_{k+1}(t) - \ell_z(t) \Big) & \text{if} \quad z < t < \hat{t}_{k+1}. \end{cases} \tag{4.9}$$

The local function $\ell_z(t)$ associated with z is found by solving the equation where the left-hand side (LHS) is given by (4.8) and (4.9) constitutes the right-hand side (RHS) (see section A.1 for details on the computation):

$$\ell_z(t) = \begin{cases} \hat{\ell}_k(t) + \frac{B \circ \hat{\omega}_k(t)}{B \circ \omega_k(t)} \Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t) \Big) & \text{if} \quad \hat{t}_k < t \leq z, \\ \hat{\ell}_k(t) + \frac{B \circ \hat{\omega}_k(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)} \Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t) \Big) & \text{if} \quad z < t < \hat{t}_{k+1}. \end{cases} \tag{4.10}$$

A slight re-formulation of (4.10) and exploiting (4.7) leads to the middle formula in (4.6). □

We observe (4.10) and conclude that the local function $\ell_z(t)$, associated with the new knot z, is represented through knot insertion in terms of the (original) local functions $\hat{\ell}_k(t)$ and $\hat{\ell}_{k+1}(t)$ and the rational function $F_{k,\hat{\mathbf{t}}}(t)$. The rational form is different from the original construction. $F_{k,\hat{\mathbf{t}}}(t)$ is different in the intervals $(\hat{t}_k, z)$ and $(z, \hat{t}_{k+1})$.

We can see from Theorem 4.1 that $\omega_k(z) = 1$. The first part of (4.9) shows that $f(z) = \ell_z(z)$. For reference, see the Hermite interpolation properties in (4.3) and (4.4). Since $\lim_{t \to z^+} \omega_{k+1}(t) = 0$, $\lim_{t \to z^-} f(t) = \ell_z(t)$. This shows that (4.7) is continuous on $(\hat{t}_k, \hat{t}_{k+1})$.

Both $\hat{\omega}_k(t)$ and $\omega_k(t)$ tend to zero as t approaches $\hat{t}_k$ from above. But $\hat{\omega}_k(t)$ tends to zero faster than $\omega_k(t)$, since $z < t_{k+1}$, hence,

$$\lim_{t \to \hat{t}_k{}^+} F_{k,\hat{\mathbf{t}}}(t) = \lim_{t \to \hat{t}_k{}^+} \frac{0}{B \circ \omega_k(t)} = 0.$$

On the other hand, when t is approaching $\hat{t}_{k+1}$ from below, both $\hat{\omega}_k(t)$ and $\omega_{k+1}(t)$ tend to one. But, since $\hat{t}_k < z$, $\hat{\omega}_k(t)$ tends to one faster than $\omega_{k+1}(t)$. Thus,

$$\lim_{t \to \hat{t}_{k+1}{}^-} F_{k,\hat{\mathbf{t}}}(t) = \lim_{t \to \hat{t}_{k+1}{}^-} \frac{1 - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)} = 1.$$

A plot of $F_{k,\hat{\mathbf{t}}}(t)$, as defined in (4.7), is provided in Figure 4.2.

Our final note here is that the observations on the limits of $F_{k,\hat{\mathbf{t}}}(t)$ as t goes towards $\hat{t}_k$ and $\hat{t}_{k+1}$ shows that the middle formula in (4.6), on the form of (4.8), satisfies the ERBS Hermite interpolation properties (4.5).

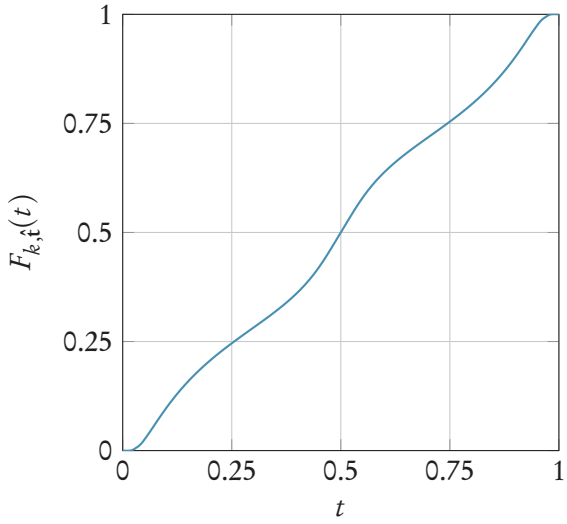

Figure 4.2: A plot of $F_{k,\hat{\mathbf{t}}}(t)$ in (4.7) with $\hat{t}_k = 0, z = 0.5$ and $\hat{t}_{k+1} = 1$.

4.3.3 Approximation of local functions

As we have seen in the previous section, generating a local curve for a new knot inserted into an existing knot vector has a solution which involves a rational function. In practical applications the local functions are usually polynomials, such as Bézier curves, Hermite curves or circle arcs. It is therefore interesting to investigate methods to generate local curves of different brands and how well they approximate the original ERBS. We propose the following strategies to generate local curves:

I Use a truncated Taylor expansion of the original ERBS curve in the new knot. This is, due to the ERBS Hermite interpolation properties (see (4.3), (4.4) and (4.5)), equivalent to evaluating the original ERBS curve given the new knot as its parametric value.

II Construct a new local curve, of the same brand as the existing local curves, which Hermite interpolates the value and all existing derivatives of the ERBS in the new knot.

III Construct a new local curve, of the same brand as the existing local curves, by performing a least squares approximation of the local curve specified by the middle formula in (4.6) of Theorem 4.1.

IV Adjust the coefficients of the local curve, obtained by any of the two previous methods, to obtain a better geometric approximation of the global ERBS curve.

As an example we consider a closed parametric ERBS curve, which approximates a circle with radius $r = 10$ and parametric value $t \in [0, 2\pi)$, using four quadratic Bézier local curves on the knot vector $\mathbf{t} = \{\frac{3\pi}{2}, 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 0\}$. Table 4.1 shows error measurements for the different methods[1] to generate local curves for the new knot $z = 3.5$.

[1]Plots of the parametric ERBS curve and its local Bézier curves before knot insertion and after knot insertion, for the scenarios I, II, and II + IV, are provided in section B.1. These plots are an extension of the published version of this article.

	I	II	II + IV	III	III + IV
$L^2(f-g)$	0.075	0.080	0.12	0.071	0.032
$L^\infty(f-g)$	0.29	0.26	0.45	0.25	0.14
$L^2_G(f-g)$	0.049	0.046	0.0078	0.026	0.0069
$L^\infty_G(f-g)$	0.22	0.17	0.033	0.097	0.032

Table 4.1: Error measurements for different kinds of local curves when inserting one new knot on an ERBS approximation of a circle using four quadratic Bézier local curves.

We measure the deviation using norms presented in [9]; a max norm

$$L^\infty(f-g) = \max_{t\in[t_1,t_{n+1}]} |f(t)-g(t)|,$$

to measure the guaranteed maximum deviation, and an L^2 norm

$$L^2(f-g) = \sqrt{\frac{1}{t_{n+1}-t_1}\int_{t_1}^{t_{n+1}} |f(t)-g(t)|^2\,dt},$$

to investigate the "quadratic" mean deviation. These two norms take the parameterization into consideration, thus, they are used to measure the "mathematical" deviation.

We use another error measure, proposed in [9], which only refers to the geometric shape and not to the speed of the parameterization, using a non-symmetric version of the Hausdorff distance, for measuring the result, using a geometric version of a metric related to a max norm,

$$L^\infty_G(f-g) = \max_{t\in[t_1,t_{n+1}]} |f(t)-C_g(f(t))|,$$

where $C_g(\mathbf{p})$ refers to the closest point on a curve f from a point $\mathbf{p}$. A metric related to the L^2 norm is constructed in [9] by the following:

$$L^2_G(f-g) = \sqrt{\frac{\int_{t_1}^{t_{n+1}} |f(t)-C_g(f(t))|^2 |Df(t)|\,dt}{\int_{t_1}^{t_{n+1}} |Df(t)|\,dt}}.$$

The last method (IV) above is achieved here by a rather naïve, iterative algorithm which translates or dilates the coefficients of the Bézier curve in small steps; first along the line which passes through the first and the last coefficients, then along a line which passes through one of the "inner" coefficients and its closest point on the first line, until it reaches a stop criterion when the approximation error does not improve.

4.4 Concluding remarks

Local refinement of ERBS curves, in terms of knot insertion, constrained by not altering any of the existing local curves, provides a rational function as the new coefficient.

We propose a few methods to approximate the local curves associated with the inserted knots. This is of interest in applications where it is desirable to use a homogeneous set of

local curves. The error is restricted within the support intervals of the new local curves.

Methods I and II shows nearly similar performance but the geometric shape of the global curve is different in the two cases. We note a significant increase in geometric deviation with method III. Invoking IV shows that it is possible to improve the global geometric approximation error by altering the new local curve. But, since it entails a change in the parameterization speed, the mathematical deviation may increase. Furthermore, the refined ERBS curve may not interpolate the original in the new knot.

As topics for future work we suggest investigating least squares approximation with respect to the global approximation error, knot insertion on the generalized set of ERBS basis functions (GERBS [5]), or even Sigmoid functions. It would also be interesting to investigate convergence when the value of the inserted knot becomes arbitrarily close to an existing knot.

A different approach for investigating local refinement in terms of knot insertion could be to consider Taylor expansions of functions as local coefficients rather than the polynomial functions considered here.

References

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

[2] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.

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

[4] R. Dalmo. Local refinement of ERBS curves. In V. Pasheva and G. Venkov, editors, *39th International conference applications of mathematics in engineering and economics AMEE13*, volume 1570 of *AIP Conference Proceedings*, pages 204–211. AIP Publishing, 2013.

[5] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[6] L. T. Dechevsky, A. Lakså, and B. Bang. Expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 27(3):319–362, 2006.

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

[8] R. N. Goldman and T. Lyche. *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*. Geometric Design Publications. SIAM, 3600 University City Science Center, Philadelphia, PA 19104-2688, 1993.

[9] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[10] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[11] L. L. Schumaker and T. Sorokina. Smooth macro-elements on Powell-Sabin-12 splits. *Mathematics of Computation*, 75(254):711–726, 2006.

[12] T. W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. T-splines and T-NURCCs. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 477–484, New York, NY, USA, 2003. ACM.

5 Data approximation using a blending type spline construction

Rune Dalmo and Jostein Bratlie

This chapter is a reprint of [2]

Abstract — Generalized expo-rational B-splines (GERBS) is a blending type spline construction where local functions at each knot are blended together by C^k-smooth basis functions. One way of approximating discrete regular data using GERBS is by partitioning the data set into subsets and fit a local function to each subset. Partitioning and fitting strategies can be devised such that important or interesting data points are interpolated in order to preserve certain features.

We present a method for fitting discrete data using a tensor product GERBS construction. The method is based on detection of feature points using differential geometry. Derivatives, which are necessary for feature point detection and used to construct local surface patches, are approximated from the discrete data using finite differences.

5.1 Motivation and background

Discrete regular data is used to represent data in a variety of application areas, such as the video-game industry or cartography systems, where they are used e.g. to represent surface elevation data in terms of height maps.

Fitting of discrete curve data with generalized expo-rational B-splines (GERBS) [3] was addressed in [1]. In that work several strategies for partitioning of the discrete data set was proposed. Furthermore, examples of curve fitting based on least squares approximation were provided.

In the present paper we investigate fitting of discrete surface data using spline blending functions. We are interested in a spline representation, which is evaluable everywhere on its domain, as a substitute for the discrete data. The methodology is based on expansion from and synthesizing the discrete data utilizing constrained blending- and measure techniques. We propose a two-step process of determining interpolation knots at selected points of interest and choosing appropriate local functions as coefficients.

In the following sections we explain briefly the considered blending type spline construction followed by strategies for partitioning and data fitting. Then we present an example of partitioning and fitting applied to a synthetic test data and provide our concluding remarks.

5.2 Blending type spline constructions

The blending functions of GERBS [7, 3] is presented in [6] as an adjusted recursive definition of the B-spline associated with the knots $(t_i)_{i=0}^{k+d}$:

$$B_{d,k}(t) = B \circ \omega_{d,k}(t) B_{d-1,k}(t) + (1 - B \circ \omega_{d,k+1}(t)) B_{d-1,k+1}(t),$$

where $\omega_{d,i}(t) = \frac{t-t_i}{t_{i+d}-t_i}$, $B_{0,i}(t) = \begin{cases} 1; & \text{if } t_i \leq t < t_{i+1} \\ 0; & \text{otherwise,} \end{cases}$ and, in the case of GERBS, the degree $d = 1$, and B is a C^k-smooth blending function possessing the following set of properties:

1. $B : I \to I$ $(I = [0,1] \subset \mathbb{R})$,
2. $B(0) = 0$,
3. $B(1) = 1$,
4. $B'(t) >= 0$, $t \in I$.
5. $B(t) + B(1-t) = 1$, $t \in I$.

The last property is optional and specifies point symmetry around the point $(0.5, 0.5)$, however, we assume this property in the present study.

B-functions come in a wide range of flavors including trigonometric, polynomial, rational and expo-rational. The perhaps most simple example of a B-function is $B(t) = t$. One

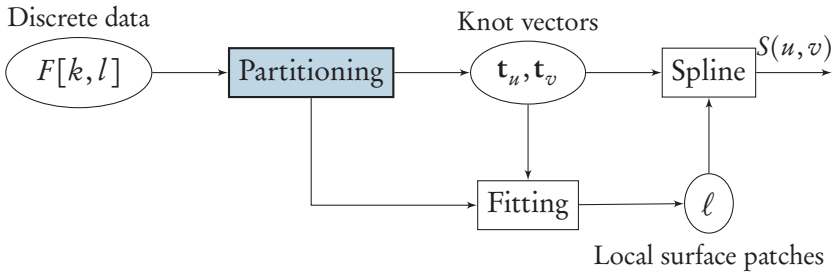

Figure 5.1: The steps involved in partitioning and fitting of discrete data using a blending type spline surface.

example of a C^∞-smooth B-function, which belongs to the family of logistic expo-rational B-splines (LERBS) presented in [8], is the following:

$$B(t) = \frac{1}{1+e^{\left(\frac{1}{t}-\frac{1}{1-t}\right)}}.$$

The type of surfaces we consider here are tensor products. A tensor product B-function spline surface is defined in [6, 5] as follows:

$$S(u,v) = \sum_{i=1}^{n}\sum_{j=1}^{m} \ell_{i,j}(u,v) B_{1,i}(u) B_{1,j}(v),$$

where $\ell_{i,j}(u,v)$ are *local surface patches* which are blended together by the C^k-smooth basis functions B. We note that using local surface patches as coefficients facilitates blending of points (Bézier and B-spline surfaces), points and vectors (Hermite interpolation surfaces) or even scalar- point- or vector valued functions (GERBS). Furthermore, we note the ERBS Hermite interpolation property [5] which states that ERBS type spline blending constructions interpolate the position and all existing derivatives of the local functions at every knot.

5.3 Partitioning and data fitting

As outlined in [1], the process of fitting a blending type spline to discrete data can be done in a sequence of a few steps, which are illustrated in Figure 5.1.

A fundamental part of parametric spline approximation in general is to determine appropriate knot vectors. In the case of GERBS, the knots are interpolation points. In order to fit a GERBS construction to discrete data it is therefore necessary to decide where to place the interpolation knots (t_u, t_v) and to decide the degree of the local surface patches. This can be done in a number of different ways. We suggest exploiting the Hermite interpolation property by placing local patches at points of interest, such as singularities, extrema or inflection points.

In the present work we have chosen the four corners of the tensor product grid as initial interpolation points. Next, we proceed by finding inflection points as follows. From

differential geometry [4] we have that the curvature of the curve α at $t \in I$ is

$$k(t) = \frac{|\alpha' \wedge \alpha''|}{|\alpha'|^3},$$

where $\alpha : I \to \mathbb{R}^3$ is a regular parameterized curve, $\alpha' = \frac{d\alpha}{dt}$ and $\alpha'' = \frac{d^2\alpha}{dt^2}$. (This is obtained by a re-parameterization of $\alpha(I)$ by the arc length $s = s(t)$, measured from $t_0 \in I$, where $t = t(s)$ is the inverse function of s). In the case of a plane curve, $\alpha : I \to \mathbb{R}^2$, the numerator is the wedge product (scalar value), which means that the curvature is signed. We use the curvature sign changes to detect inflection points in both parametric directions u and v of the tensor product surface. Finally we consolidate the corresponding parameter values together with start- and end parameter values, or corners, and assemble two knot vectors $\mathbf{t}_u$ and $\mathbf{t}_v$.

The method above utilizes first- and second derivatives of parametric lines to determine the knot vectors. The common definition of the derivative of a function f at a point x is

$$f'(x) = \lim_{h \to \infty} \frac{f(x+h) - f(x)}{h}.$$

Since we are considering discrete data where h holds a fixed (non-zero) value instead of approaching zero, derivatives can be approximated using finite differences:

$$f'(x) = \frac{\delta_h[f](x)}{h}.$$

For this purpose we use the central differences given by

$$\delta_h[f](x) = f(x + \tfrac{1}{2}h) - f(x - \tfrac{1}{2}h),$$

whose error is proportional to the square of the spacing:

$$\frac{\delta_h[f](x)}{h} - f'(x) = O(h^2).$$

It is appropriate to remark here that oscillating functions can yield zero derivative with central differences. Furthermore, we note that derivatives on the boundaries can be approximated using higher order finite differences.

5.4 Results

In this section we provide an example of fitting a spline surface to a synthetic 2D height map surface. The discrete height map $F[k,l]$ is obtained by sampling the function given by

$$F(x,y) = \sin(\tfrac{1}{2}x)\cos(\tfrac{y}{5}), \quad x \in [0,30], \quad y \in [0,30]$$

into a grid with $M = 30 \times N = 30$ equidistant points. In the example, partitioning of the data set is based on inflection point detection, and the local patches for the spline surface are constructed by Hermite interpolation of order six. A plot of the discrete point set $F[k,l]$

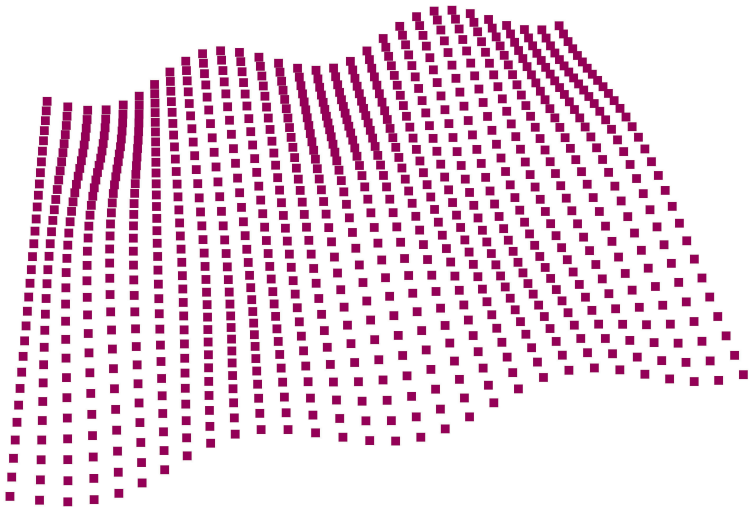

Figure 5.2: An example of a discrete height map surface obtained from sampling a bi-variate periodic test function into a uniform regular grid with 30×30 points.

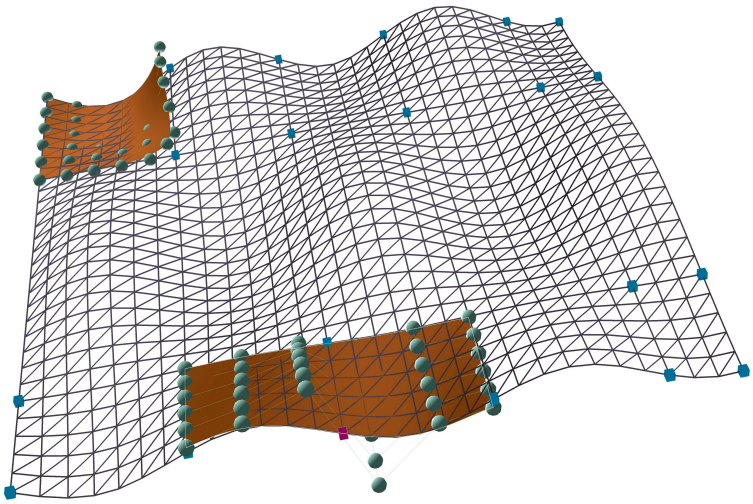

Figure 5.3: A tessellation of a blending type spline surface is shown together with two local surface patches in Bézier representation. The green spheres and lines outline the control nets of the local patches. The small cubes on the spline surface indicate the positions of the interpolation knots. The interpolation points of the local patches are the upper left corner and the purple cube on the bottom boundary.

is shown in Figure 5.2. Figure 5.3 shows a tessellation of the spline surface $S(u,v)$ at a relatively high resolution together with a couple of local surface patches. The mean squared error (MSE) obtained by

$$\text{MSE} = \frac{1}{NM}\sum_{i=1}^{N}\sum_{j=1}^{M}(F[i,j]-S(i,j))^2$$

is approximately equal to 1.079×10^{-4} for the test case.

5.5 Concluding remarks

In this paper we have presented strategies for partitioning and fitting of discrete data using a blending type spline construction. We assume that the discrete data have a smooth origin or underlying smoothness properties. The methods are depending on the data, feature points and the desired results; e.g. interpolation properties and order of approximation.

The spline construction is smooth and can be evaluated everywhere on the domain. This is of interest in tessellation applications since the resolution of the tessellation can be specified independent of the discrete data. One particular case is data reduction using the tessellation shader steps of modern graphics processing units (GPUs). Traditionally, lots of polygon data had to be "pushed" through the graphics pipeline for rendering. The alternative, facilitated by the suggested approach, is to evaluate or generate data from control nets.

Local surface patches interpolate feature points and control the blending spline approximation. One immediate consequence is that flexibility and control of the spline can be achieved by specifying the degree of local patches, number of knots and layout of the knot vectors.

When it comes to applications, the method is applicable to cases where replacing discrete data with a construction which can be evaluated is desirable, such as, for example, tessellation of surface data at arbitrary resolution. Furthermore, the partitioning and fitting strategies can be parts in adaptive compression methods. The methods are applicable to other kinds of discrete data than geometry, as considered here, thus, application within trend analysis or combining geometry modeling with data generated from partial differential equations (PDEs) are other possibilities. A far-going extension could be to utilize multiresolution analysis (MRA) wavelets for partitioning.

We mention optimizing for numerical performance or approximation error as topics for future work. We believe this would be relevant to developing adaptive strategies. As a final note for future work we suggest extending the methods to non-regular grids.

References

[1] J. Bratlie, R. Dalmo, and P. Zanaty. Fitting of discrete data with GERBS. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Large-Scale Scientific Computing 2013*, volume 8353 of *Lecture Notes in Computer Science*, pages 569–576. Springer, 2014.

[2] R. Dalmo and J. Bratlie. Data approximation using a blending type spline construction. In V. Pasheva and G. Venkov, editors, *40th International conference applications of mathematics in engineering and economics AMEE14*, volume 1631 of *AIP Conference Proceedings*, pages 147–152. AIP Publishing, 2014.

[3] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[4] M. P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, New Jersey, 1976.

[5] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[6] A. Lakså. ERBS-surface construction on irregular grids. In V. Pasheva and G. Venkov, editors, *39th International conference applications of mathematics in engineering and economics AMEE13*, volume 1570 of *AIP Conference Proceedings*, pages 113–120. AIP Publishing, 2013.

[7] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[8] P. Zanaty. *Application of Generalized Expo-Rational B-splines in Computer Aided Design and Analysis*. PhD thesis, University of Oslo, 2014.

6 Smooth spline blending surface approximation over a triangulated irregular network

Rune Dalmo, Jostein Bratlie, Børre Bang, and Arne Lakså

This chapter is a reprint of [2]

Abstract — A triangulated irregular network (TIN) is a data structure commonly used to represent a geometric surface in computer software systems, such as geographic information systems and terrain modeling systems. Expo-rational B-splines (ERBS), a blending type spline construction in the family of generalized expo-rational B-splines (GERBS), can be used to create an approximation surface which interpolates the vertex positions of the TIN nodes. Utilizing the properties for local support of this blending spline construction, one can construct an approximation surface which is local within the second neighbourhood with respect to the infliction nodes of the underlying TIN. We present two variations of blending functions used with the construction. The first one blends the TIN edges with a smooth component, thus, the surface approximation is only C^0 over the TIN edges. The second blending method provides a surface approximation which is smooth over the TIN edges.

6.1 Introduction

Various methods for approximation and interpolation to obtain visually nice, or smooth, surfaces have been explored to the present. We provide here a brief overview of some blending methods applicable to arbitrary triangulations of non-regular data.

Triangular Bernstein-Bézier patches, described e.g. in [4], can be joined together in a smooth way by constructing pairs of co-planar triangles. However, the conditions ensuring a sufficient continuity across the common boundary lock triangles together and the construction becomes stiff. Besides, the number of co-planar triangles required grows with increasing degree of continuity, making the construction less local.

Schumaker et al. explored methods in [13, 7] where they utilize spline functions on macro-element spaces. Macro-elements of required smoothness are constructed on splits of triangles. They are used to construct super-spline spaces with local, stable bases, to overcome the above mentioned problems.

The parameterization method of Floater [5], although it is not a blending type method, is applicable to smooth surface approximation of triangulations, notably using the shape-preserving parameterization [5] or mean value coordinates [6].

Techniques based on radial basis functions, introduced by Broomhead and Lowe in the neural network community [1], are now common tools for geometric data analysis. We note here that triangular data structures, since they define connectivity between vertices, can be used to determine constraints for radial basis functions. Thus, blending of radial basis functions can be considered to construct smooth surface approximations over triangulations.

In this paper we describe, using expo-rational B-splines (ERBS) surfaces [9], blending over a triangulation based on non-regular data. Our motivation is to generate a surface which is smooth over the TIN edges and at the same time interpolates the vertex positions.

The following sections explain the kind of triangulations we consider, ERBS patches and how they are constructed from pairs of triangles, how we approximate, do interpolation and blend the results, followed by some examples.

6.2 Preliminaries

6.2.1 Triangulated Irregular Network

The triangulated irregular network (TIN), proposed and explored by Peucker et al. in a series of papers, notably [11, 12], is a digital terrain model (DTM) consisting of irregularly distributed nodes and lines. It constitutes a network of non-overlapping triangles representing a tessellation of a surface, usually in Euclidean space $\mathbb{R}^3$. TINs are typically vector-based representations of terrain data. They are commonly used in geographic information systems (GIS). The TIN surface model is oriented to line features as well as points. Using triangles to represent terrain facilitates a realistic representation if the spatial data units recognize natural surface changes in slope, at peaks, pits, passes, ridge lines, saddle points and course lines or discontinuities. Triangular facets can be created to meet these conditions by having their corners located at control points with exact known coordinates, and having triangle edges fall

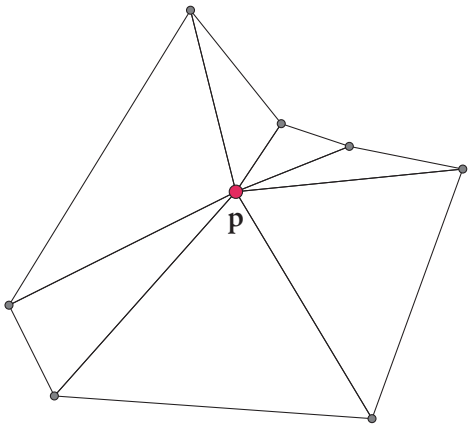

Figure 6.1: A point **p** and its first neighborhood (points connected to this point through an edge) in a TIN

along approximations of ridges. More information regarding terrain representation based on TINs can be found in [10].

A part of a TIN is displayed in Figure 6.1. Due to the nature of TIN data we conclude that it is possible to "see" the first neighborhood from any point in a TIN and measure the distances along the edges. Using this it follows that an edge does not intersect the physical terrain.

6.2.2 Expo-rational B-splines

Expo-rational B-splines, as defined in [9], provide a blending type construction where local functions at each knot are blended together by C^∞-smooth basis functions:

$$f(t) = \sum_{k=1}^{n} l_k(t) B_k(t)$$

where $\mathbf{t} = \{t_k\}_{k=0}^{k=n+1}$ is an increasing knot vector, and each basis function $B_j(t)$ is C^∞ on its support (t_{j-1}, t_{j+1}) with $B_k(t_k) = 1$, and $D^j B_k(t_k) = 0$ for $j = 1, 2, \ldots$.

In this paper we consider the scalable subset of the ERBS basis presented in [3] with the default set of intrinsic parameters proposed by Lakså in [8]:

$$B_k(t) = \begin{cases} S_{k-1} \int_0^{w_{k-1}(t)} \psi_{k-1}(s) ds, & t_{k-1} < t \leq t_k \\ S_k \int_{w_k(t)}^1 \psi_k(s) ds, & t_k < t < t_{k+1} \\ 0, & \text{otherwise,} \end{cases} \tag{6.1}$$

where $w_k(t) = \frac{t - t_k}{t_{k+1} - t_k}$, $\psi(t) = e^{-\frac{\left(t-\frac{1}{2}\right)^2}{t(1-t)}}$, and $S_k = \left(\int_0^1 \psi_k(t) dt \right)^{-1}$.

We have the following general formula for parametric tensor product surfaces using

ERBS,

$$S(u,v) = \sum_{i=1}^{n_u} \sum_{j=1}^{n_v} s_{ij}(u,v) B_i(u) B_j(v), \tag{6.2}$$

where $s_{ij}(u,v),\ i = 1,\ldots,n_u,\ j = 1,\ldots,n_v$ are $n_u \times n_v$ local Bézier surface patches, and $B_i(u), B_j(v)$ are the respective ERBS basis functions.

6.3 ERBS over a TIN

Given a TIN, $\Delta(x,y,z)$, which is a tessellation of a surface in $\mathbb{R}^3$ where the z axis represents data, for instance physical terrain heights. We construct ERBS surface patches from pairs of neighboring triangles by considering inner edges in the TIN structure. Each ERBS surface patch $L(u,v)$ is a differentiable map, $L : \Omega_L \subset \mathbb{R}^2 \to \mathbb{R}^3$. Figure 6.2 illustrates an ERBS surface patch construction, based on four bi-linear Bézier local surfaces, as defined in (6.2). Its positions $\mathbf{p}_i$, $i = 1,\ldots,4$ and derivatives $\mathbf{u}_1, \mathbf{u}_2, \mathbf{v}_1$ and $\mathbf{v}_2$ are retrieved from the triangulation, where $\mathbf{p}_1, \mathbf{p}_2$ and $\mathbf{p}_3$ are vertices in one triangle, and $\mathbf{p}_1, \mathbf{p}_3$ and $\mathbf{p}_4$ outline a neighbor triangle. The dotted line between $\mathbf{p}_1$ and $\mathbf{p}_3$ constitutes the common edge shared by the two triangles. The right part of Figure 6.2 illustrates an ERBS surface patch on a TIN.

Directional derivatives for the bi-linear Bézier local surface $S(u,v)$ in position $\mathbf{p}_1$ are defined as follows:

$$\begin{aligned} \frac{\partial S}{\partial u} &= \mathbf{p}_2 - \mathbf{p}_1, \\ \frac{\partial S}{\partial v} &= \mathbf{p}_4 - \mathbf{p}_1, \\ \frac{\partial^2 S}{\partial u \partial v} &= (\mathbf{p}_3 + \mathbf{p}_1) - (\mathbf{p}_4 + \mathbf{p}_2). \end{aligned} \tag{6.3}$$

Directional derivatives for the three remaining Bézier surfaces local to one ERBS surface patch, one in each of the vertices $\mathbf{p}_2$, $\mathbf{p}_3$ and $\mathbf{p}_4$, are defined in a manner similar to (6.3).

A consequence of the ERBS Hermite interpolation properties explored in [8, theorem 2.4] is that the derivatives of an ERBS surface patch, in a given knot, are equal to all existing derivatives of the Bézier local surface in that knot. It follows that each ERBS surface patch interpolates TIN positions in four vertices: $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ and $\mathbf{p}_4$ in Figure 6.2.

6.4 Smooth surface construction

We define a parametric regular grid surface $\Theta(u,v)$, $\Theta : \Omega_\Theta \subset \mathbb{R}^2 \to \mathbb{R}^3$, which covers the triangulation $\Delta(x,y,z)$. The positions in $\mathbb{R}^3$ for the points $\mathbf{p}_\Theta(u,v)$ on the surface Θ are computed as follows. First we find in which triangle $t \in \Delta$ the point $\mathbf{p}_\Delta$ is inside, where $\mathbf{p}_\Delta$ is a 2D projection of $\mathbf{p}_\Theta$ mapped to Δ in Cartesian coordinates. Next, we evaluate each ERBS surface patch L_i in $\mathbf{p}_{L_i}$, where L_i covers t and $\mathbf{p}_{L_i}$ is $\mathbf{p}_\Theta$ mapped to L_i in its local coordinates, and blend the results. We denote the mapping from $\mathbf{p}_\Theta$ to $\mathbf{p}_{L_i}$ by $\omega_i(u,v)$.

The blending distribution is computed from the barycentric coordinates of the triangles.

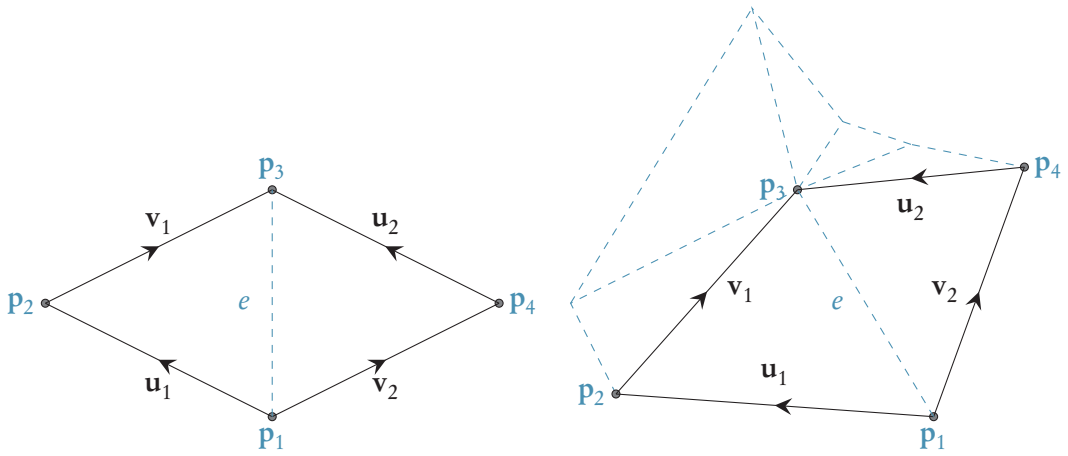

Figure 6.2: Left: A bi-linear ERBS surface patch $L(u,v)$ constructed from vertices and edges in two neighboring triangles. Right: An ERBS surface patch $L(u,v)$ on a TIN $\Delta(x,y,z)$.

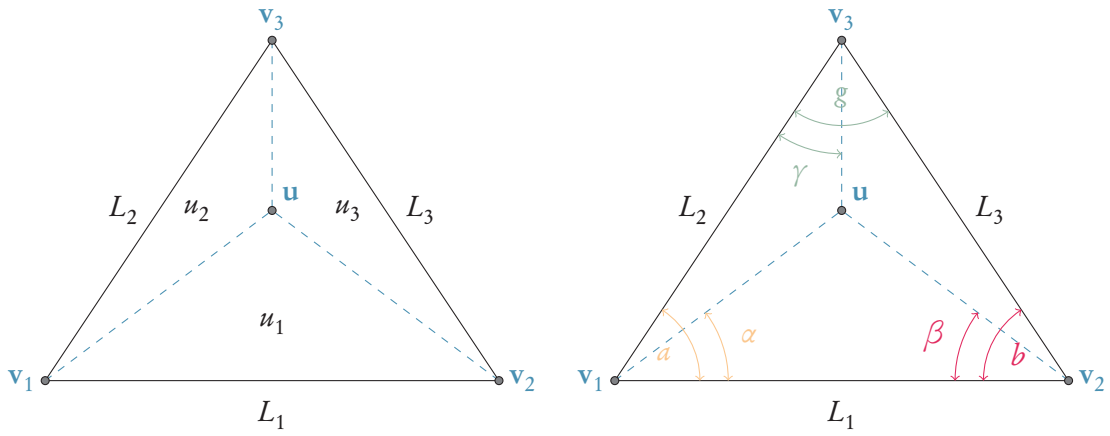

Figure 6.3: Criteria for blending of bi-linear ERBS surface patches using homogeneous barycentric coordinates u_1, u_2 and u_3 (left) and angle ratios $\frac{\alpha}{a}$, $\frac{\beta}{b}$ and $\frac{\gamma}{g}$ (right), respectively.

We present two blending functions, which are both invariant under affine maps, utilizing the scalable subset of the ERBS basis function.

6.4.1 Blending using custom ERBS triangles

A set of ERBS basis functions in homogeneous barycentric coordinates is defined in [8] as

$$\mathscr{B}_{k,i}(\mathbf{u}) = \frac{B(u_i)}{\sum_{j=1}^{k} B(u_j)} \text{ for } k > 1 \text{ and } i = 1,2,\ldots,k, \tag{6.4}$$

and where $B(u_i)$, $i = 1,2,\ldots,k$ is defined in (6.1). In the case of triangles, each $\mathscr{B}_{3,i}(\mathbf{u})$ evaluates to 1 in the vertex u_i and 0 in the two other vertices. Everywhere else the value is between 0 and 1.

ERBS surface patches can be blended using a slightly modified version of the ERBS trian-

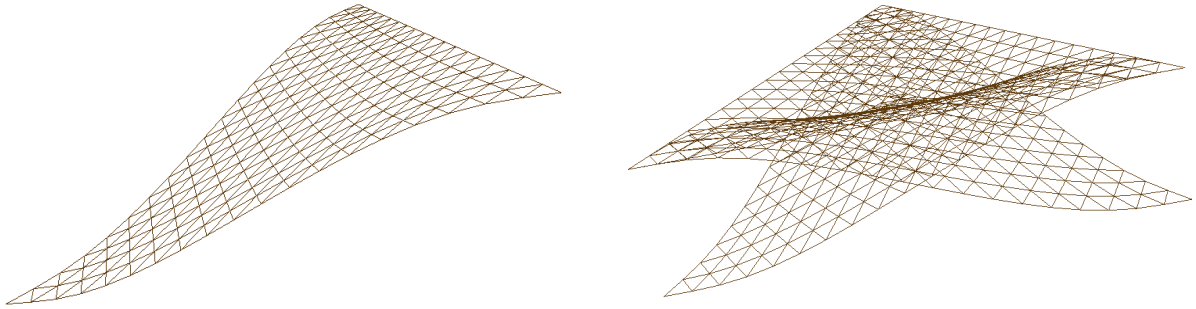

Figure 6.4: Modified ERBS triangles in homogeneous barycentric coordinates, rendered with triangle strips. Left: One ERBS basis function $\hat{\mathscr{B}}_i(\mathbf{u})$. It evaluates to 0 in one vertex, $\frac{1}{2}$ along the opposing edge and between 0 and $\frac{1}{2}$ elsewhere. Right: Three modified ERBS triangles $\hat{\mathscr{B}}_i(\mathbf{u})$, $i = 1,2,3$ in homogeneous barycentric coordinates. The sum of the basis functions is 1 everywhere. For the sake of clarity we mention that the plots do not show parameter lines.

gle (6.4). Given a point $\mathbf{u} = (u_1, u_2, u_3)$, in homogeneous barycentric coordinates, as shown in the left part of Figure 6.3. Then

$$\Theta(u,v) = \sum_{i=1}^{3} \hat{\mathscr{B}}_i(\mathbf{u})\, L_i \circ \omega_i(u,v), \tag{6.5}$$

where $L_i \circ \omega_i(u,v)$ is the ith ERBS surface patch evaluated in its local coordinates, and the set of ERBS basis functions $\hat{\mathscr{B}}_i(\mathbf{u})$ is defined as

$$\hat{\mathscr{B}}_i(\mathbf{u}) = \frac{B(1-u_i)}{\sum_{j=1}^{3} B(1-u_j)},\ i = 1,2,3, \tag{6.6}$$

where we clearly can see that $\sum_{i=1}^{3} \hat{\mathscr{B}}_i(\mathbf{u}) = 1$.

A plot of the modified ERBS triangle (6.6) is shown in Figure 6.4. It evaluates to 0 in one vertex, is positive on the edges, is $\frac{1}{2}$ along the opposing edge between the two other vertices and is between 0 and $\frac{1}{2}$ elsewhere. As a consequence, when it comes to blending of ERBS surface patches on a specific edge, we conclude that one ERBS triangle is $\frac{1}{2}$ and the two other ERBS triangles sum up to $\frac{1}{2}$. Figure 6.4 illustrates this by showing three modified ERBS triangles in homogeneous barycentric coordinates.

6.4.2 Blending using angle ratios in triangles

Given a point $\mathbf{u} = (u_1, u_2, u_3)$, in homogeneous barycentric coordinates, as shown in the right part of Figure 6.3. Then

$$\begin{aligned}\Theta(u,v) = &\Big(\Big(1-B\left(\tfrac{\alpha}{a}\right)\Big)\, L_1\circ\omega_1(u,v) + \\ &\qquad B\left(\tfrac{\alpha}{a}\right) L_2\circ\omega_2(u,v)\Big)\,\Big(1-B\left(\tfrac{\gamma}{g}\right)\Big) \\ +&\Big(\Big(1-B\left(\tfrac{\beta}{b}\right)\Big)\, L_1\circ\omega_1(u,v) + \\ &\qquad B\left(\tfrac{\beta}{b}\right) L_3\circ\omega_3(u,v)\Big)\, B\left(\tfrac{\gamma}{g}\right),\end{aligned} \tag{6.7}$$

where a, b, g, α, β and γ are the angles in Figure 6.3, so that α, β and γ depend on $\mathbf{u}$, and $B(u)$ is the ERBS basis function in (6.1) and $L_i \circ \omega_i(u,v)$ is the ith ERBS surface patch evaluated in its local coordinates. By re-arranging (6.7) and using (6.5) we obtain

$$\begin{aligned}\tilde{\mathscr{B}}_1(\mathbf{u}) &= \Big(1-B(\tfrac{\alpha}{a})\Big)\Big(1-B(\tfrac{\gamma}{g})\Big)+\Big(1-B(\tfrac{\beta}{b})\Big)B(\tfrac{\gamma}{g}),\\ \tilde{\mathscr{B}}_2(\mathbf{u}) &= B(\tfrac{\alpha}{a})\Big(1-B(\tfrac{\gamma}{g})\Big),\\ \tilde{\mathscr{B}}_3(\mathbf{u}) &= B(\tfrac{\beta}{b})B(\tfrac{\gamma}{g}).\end{aligned} \tag{6.8}$$

It follows that $\sum_{i=1}^{3} \tilde{\mathscr{B}}_i(\mathbf{u}) = 1$ since

$$\begin{aligned}\sum_{i=1}^{3} \tilde{\mathscr{B}}_i(\mathbf{u}) = 1 &- B(\tfrac{\gamma}{g}) - B(\tfrac{\alpha}{a}) + B(\tfrac{\alpha}{a})B(\tfrac{\gamma}{g}) + B(\tfrac{\gamma}{g}) - B(\tfrac{\beta}{b})B(\tfrac{\gamma}{g}) \\ &+ B(\tfrac{\alpha}{a}) - B(\tfrac{\alpha}{a})B(\tfrac{\gamma}{g}) \\ &+ B(\tfrac{\beta}{b})B(\tfrac{\gamma}{g}) \\ = 1.&\end{aligned} \tag{6.9}$$

The left part of Figure 6.5 shows a plot of one of the basis functions in (6.8). It evaluates to 0 along two edges and 1 along the third edge, but is 1 at one vertex only. Hence, there is a "jump" between basis functions in the vertices.

Using this and investigating (6.7)-(6.9) we conclude that, in this case, one single ERBS surface patch $L_i \circ \omega_i(u,v)$ (see Figure 6.2) will be evaluated on each edge. The right part of Figure 6.5 shows a plot of the three basis functions in (6.8) in homogeneous barycentric coordinates.

6.5 Concluding remarks

The method presented in this article is considered to be data-driven. We believe it makes sense to use such methods in cases where we must rely on topology information to generate derivatives. The method is local within the second neighborhood of a TIN node as a consequence of the ERBS surface patch construction.

Whether the construction provides smoothness in the vertices or on the edges depends on the blending functions. The provided blending functions interpolate the position in every

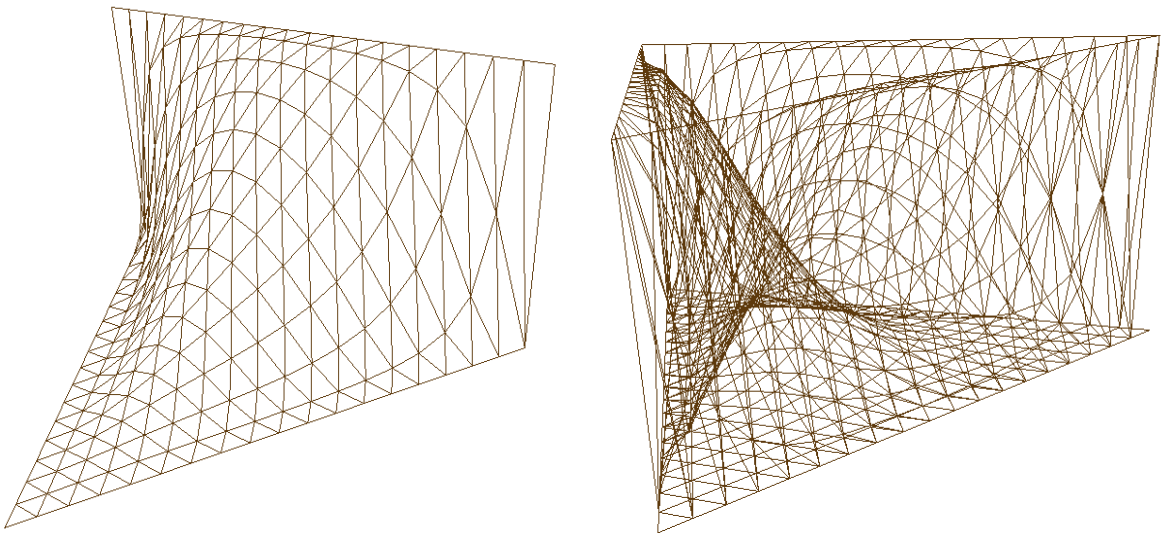

Figure 6.5: ERBS blending functions using angle ratios, in homogeneous barycentric coordinates, rendered with triangle strips. Left: One angle ratio ERBS basis function $\tilde{\mathscr{B}}_i(\mathbf{u})$. It evaluates to 0 along two edges and 1 along one edge. Right: Three angle ratio ERBS basis functions $\tilde{\mathscr{B}}_i(\mathbf{u}), i = 1,2,3$. The sum of basis functions is 1 everywhere. On each edge, one basis function evaluates to 1, whereas the two remaining basis functions evaluate to 0. For the sake of clarity we mention that the plots do not show parameter lines.

vertex. First order directional derivatives, in a given vertex, exist but are not continuous since the limits from the different ERBS surface patches do not converge towards the same. We consider the construction to be C^0 in the vertices.

Blending using the original ERBS triangles proposed in [8] (see (6.4)) provides C^0 approximation over the edges, but is C^∞-smooth in the vertices. In the case of modified ERBS triangles, defined in (6.6), the resulting surface $\Theta(u,v)$ is C^0 in the vertices and across the edges. However, when compared to the original TIN, the visual result is slightly improved in terms of smoothness, since the ERBS surface patch which is smooth over the considered edge is given the weight $\frac{1}{2}$. "Remains" of edges are still visible due to the jump in first order directional derivatives between the two other patches.

The blending method based on angle ratios, described in (6.7), ensures that only one ERBS surface patch, which is smooth over the considered edge, is evaluated on that edge. Notably, the surface $\Theta(u,v)$ inherits smoothness properties from the ERBS surface patch over an edge, as a consequence of the ERBS Hermite interpolation properties, since the derivatives of (6.7) are 0 along the edges (6.8). As expected, the discontinuities in the first order directional derivatives are not longer visible.

We note that there is a trade-off between approximation error and smoothness over edges. The bi-linear Bézier local patches in (6.3) approximate the outer edges well, but pull the surface away from the inner edge. In contrast to the angle ratio method of (6.7), which considers only a single ERBS surface patch, the modified ERBS triangle method in (6.5) and (6.6) approximates edges better since evaluations from outer patch edges are blended in. Visual examples showing the difference between the two blending functions applied to a synthetic TIN are provided in Figure 6.6.

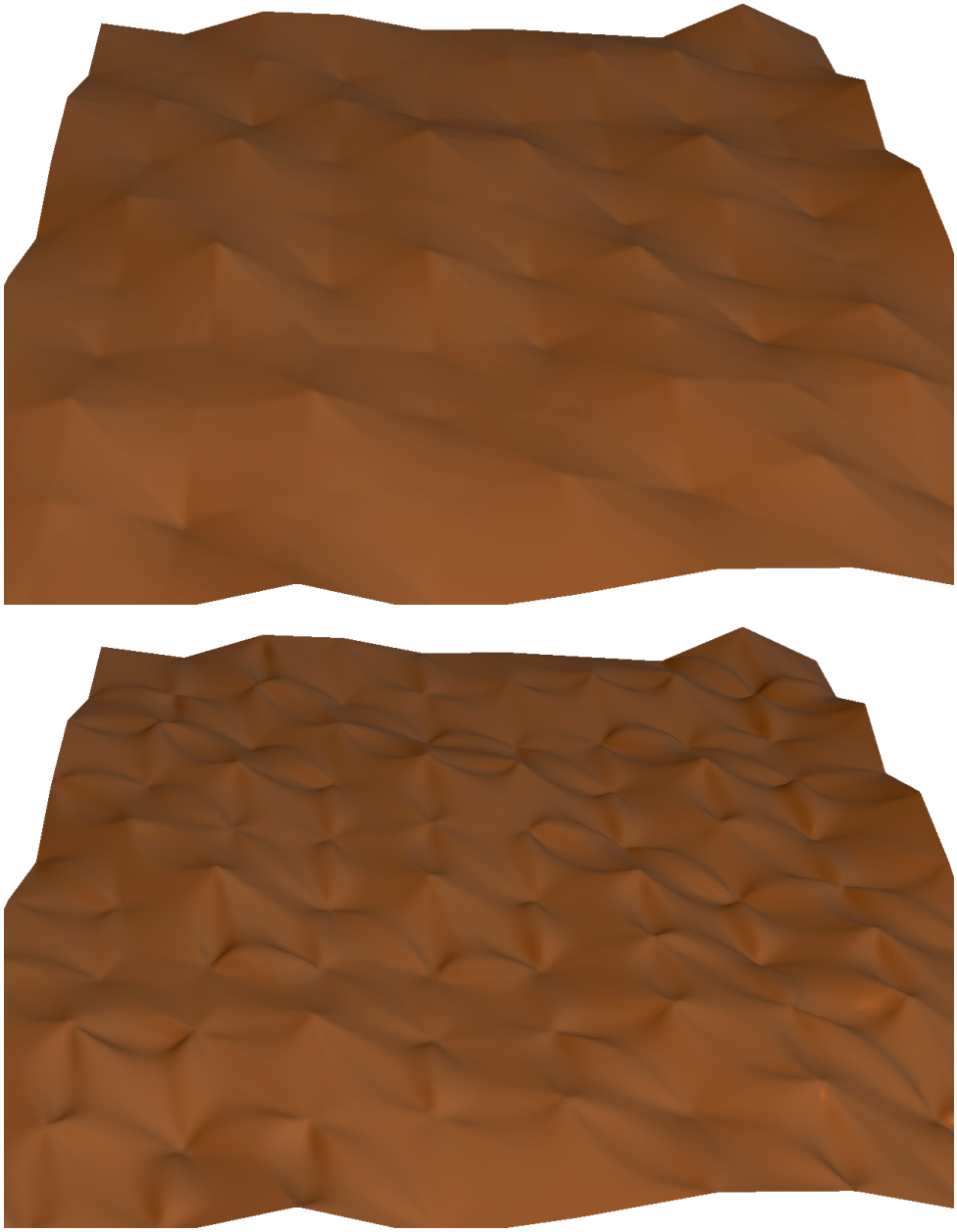

Figure 6.6: Regular grid approximations of a synthetic TIN using different blending methods. TIN vertex positions are interpolated in both cases. The surfaces are C^0 in the vertices and smooth outside vertices and edges. Top: Custom ERBS triangle blending functions are used. The surface is C^0 across the edges, thus, traces of edges appear as discontinuities in first order directional derivatives. Bottom: The surface is smooth over the edges as a consequence of using the ERBS angle ratio blending functions.

Another consequence of the ERBS Hermite interpolation properties is that the construction will work even for higher order derivatives than the bi-linear Bézier case considered here, since all existing derivatives from the underlying local patches are propagated to the ERBS surface patches. The method will still be local within the second neighborhood, given that information regarding derivatives of higher order is provided with the TIN.

References

[1] D. S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2(3):321–355, 1988.

[2] R. Dalmo, J. Bratlie, B. Bang, and A. Lakså. Smooth spline blending surface approximation over a triangulated irregular network. *International Journal of Applied Mathematics*, 27(1):109–119, 2014.

[3] L. T. Dechevsky, A. Lakså, and B. Bang. Expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 27(3):319–362, 2006.

[4] G. Farin. Triangular Bernstein-Bézier patches. *Computer Aided Geometric Design*, 3(2):83–127, 1986.

[5] M. S. Floater. Parametrization and smooth approximation of surface triangulations. *Computer Aided Geometric Design*, 14(3):231–250, 1997.

[6] M. S. Floater. Mean value coordinates. *Computer Aided Geometric Design*, 20(1):19–27, 2003.

[7] M.-J. Lai and L. L. Schumaker. *Spline Functions on Triangulations*, volume 110 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge CB2 8RU, UK, 2007.

[8] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[9] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[10] R. Laurini and D. Thompson. *Fundamentals of Spatial Information Systems*. Number 37 in A.P.I.C. Academic Press, 24-28 Oval Road, London, NW1 7DX, 1992.

[11] T. K. Peucker, R. J. Fowler, J. J. Little, and D. M. Mark. The triangulated irregular network. In *Proceedings of Auto Carto 4*, volume 2, pages 96–103, Reston, Virginia, USA, 1978.

[12] T. K. Peucker, R. J. Fowler, J. J. Little, and D. M. Mark. The triangulated irregular network. In *Proceedings of American Society of Photogrammetry: Digital Terrain Models (DTM) Symposium*, pages 516–540, St. Louis, Missouri, USA, 1978.

[13] L. L. Schumaker and T. Sorokina. Smooth macro-elements on Powell-Sabin-12 splits. *Mathematics of Computation*, 75(254):711–726, 2006.

Part III

Evaluation and visualization

7 Evaluation of smooth spline blending surfaces using GPU

Jostein Bratlie, Rune Dalmo, and Børre Bang

This chapter is a reprint of [1]

Abstract — Recent development in several aspects of research on blending type spline constructions has opened up new application areas. We propose a method for evaluation and rendering of smooth blending type spline constructions using the tessellation shader steps of modern graphics hardware. In this preliminary study we focus on concepts and terminology rather than implementation details. Our approach could lead to more efficient, dynamic and stable blending-type spline based applications in fields such as interactive modeling, computer games and more.

7.1 Introduction

The purpose of this article is to introduce a concept for evaluation and rendering of smooth blending type spline surfaces using features available in recent versions of modern rendering pipelines [10], most notably the OpenGL [11], maintained by the Khronos group, and DirectX [9]. Since the year 2000, a family of blending-type spline constructions named expo-rational B-splines (ERBS) [8] and, later, generalized expo-rational B-splines (GERBS) [3, 2] has been introduced and explored by the R&D group Simulations at NUC. GERBS type splines enjoy some properties, including Hermite interpolation at the knots [6] and minimal support combined with C^k-smooth basis functions, which makes them attractive to interactive geometric modeling and smooth representations of parametric curves and surfaces.

Despite the flexibility of the construction, there is a price to pay, in particular with respect to interactive geometric modeling, due to the cost of the spline basis function evaluator. The performance is constrained by the following limitations:

1. Evaluation of the expo-rational basis function (ERB) requires an integration step.

2. The graphics rendering hardware is designed to support triangle constructions and simple cases of classic splines on Bézier form, i.e. mapped to the interval $[0,1]$.

The first issue was addressed by Zanaty in [13], where a relation between the ERBS basis function and Sigmoidal functions was explored. The second issue is addressed in this article.

Recently, in [7], Lakså expressed generic blending functions, including GERBS, in terms of classic B-splines. This is interesting since the rendering pipelines mentioned above were designed to be used with ordinary B-splines.

In this work we consider tessellation techniques which are now standardized across vendor specific application programming interfaces (APIs). Therefore, it is possible to adapt and use such tessellation steps to obtain rendering methods applicable to B-spline type constructions. We seek to describe the relevant technology; blending-type splines and rendering pipelines, as well as the concepts necessary for evaluation and rendering.

In the following sections we describe GERBS as an adjusted recursive definition of classic B-splines, similar to [7], followed by an overview of the relevant steps of the graphics pipeline present in modern GPU hardware. Next, we introduce and define the critical components of the proposed rendering- and evaluation method followed by a description of the method itself. Finally, we give our concluding remarks, where we discuss some theoretical performance results, and suggest topics for future work.

7.2 Spline blending functions

The blending functions of GERBS [8, 2] is presented in [7] as an adjusted recursive definition of the B-spline associated with the knots $(t_i)_{i=0}^{k+d}$:

$$B_{d,k}(t) = B \circ \omega_{d,k}(t) B_{d-1,k}(t) + (1 - B \circ \omega_{d,k+1}(t)) B_{d-1,k+1}(t), \tag{7.1}$$

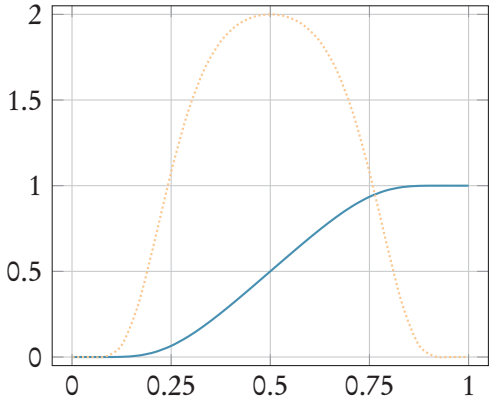

Figure 7.1: Plots of a logistic ERB and its first derivative, as solid and dotted lines, respectively.

where $\omega_{d,i}(t) = \frac{t-t_i}{t_{i+d}-t_i}$, $B_{0,i}(t) = \begin{cases} 1; & \text{if } t_i \leq t < t_{i+1}, \\ 0; & \text{otherwise,} \end{cases}$ and, in the case of GERBS, the degree $d = 1$, and B is a C^k-smooth blending function possessing the following set of properties:

1. $B : I \to I$ $(I = [0,1] \subset \mathbb{R})$,
2. $B(0) = 0$,
3. $B(1) = 1$,
4. $B'(t) >= 0$, $t \in I$.
5. $B(t) + B(1-t) = 1$, $t \in I$.

The last property is optional and specifies point symmetry around the point $(0.5, 0.5)$, however, we assume this property in the present study.

B-functions come in a wide range of flavors including trigonometric, polynomial, rational and expo-rational. The perhaps most simple example of a B-function is $B(t) = t$. One example of a C^∞-smooth B-function, which belongs to the family of LERBS was presented in [13] and can be expressed, as a logistic expo-rational B-function, as follows:

$$B(t) = \frac{1}{1 + e^{\left(\frac{1}{t} - \frac{1}{1-t}\right)}}. \tag{7.2}$$

In contrast to the classic ERBS basis function, it follows from (7.2) that evaluation of this particular $B(t)$ does not require an integration step. A plot of $B(t)$ in (7.2), where $t \in [0,1]$, is shown in Figure 7.1. An essential remark is that the exponential function is implemented in GPU hardware.

A tensor product B-function spline surface is defined in [7, 6] as follows:

$$S(u,v) = \sum_{i=1}^{n} \sum_{j=1}^{m} \ell_{i,j}(u,v) B_{1,i}(u) B_{1,j}(v),$$

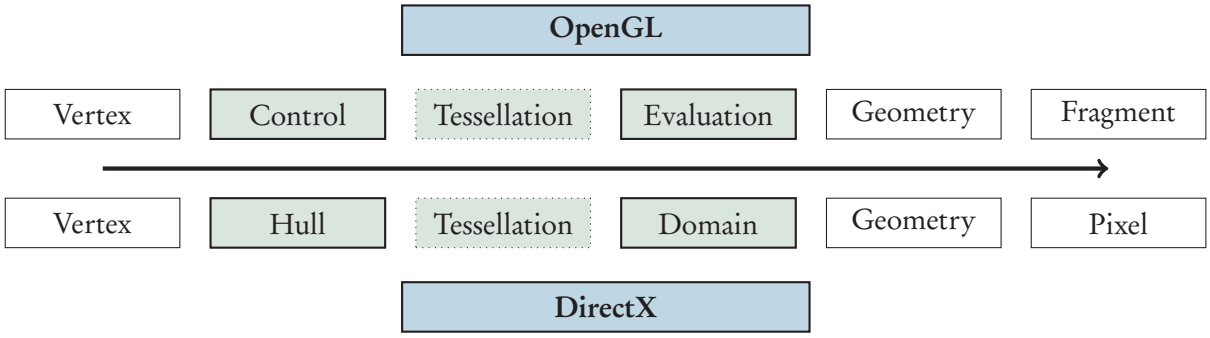

Figure 7.2: The three tessellation shader steps shown as parts of the OpenGL and DirectX rendering pipelines. The control- and evaluation shaders, using OpenGLs terminology, which are illustrated as shaded gray boxes, are programmable. The primitive generator, or tessellation step, is fixed.

where $\ell_{i,j}(u,v)$ are *local surface patches* which are blended together by the C^k-smooth basis functions B. We note that using local surface patches as coefficients facilitates blending of points (Bézier and B-spline surfaces), points and vectors (Hermite interpolation surfaces) or even scalar- point- or vector valued functions (GERBS). Furthermore, we note the ERBS Hermite interpolation property [6] which states that ERBS type spline blending constructions interpolate the position and all existing derivatives of the local functions at every knot.

7.3 GPU Tessellation

The most recently added shader component in the GPU rendering pipeline is the tessellation shader. It consists of three sub components, steps two through four, between the vertex shader and the geometry shader, as shown in Figure 7.2.

The tessellation shader steps operate on a type of primitive called *patch*. The tessellation patch primitive can be of three different types; isoline, triangle or quad.

The tessellation step is controlled through three different substeps; control, tessellation and evaluation. The control and evaluation substeps are programmable while the tessellation substep is hardware implementation specific. In the following we provide a brief description of each substep of the tessellator.

I *Control* specifies which type of patch primitive to be considered and the amount of tessellation applied to each patch. It provides control of tessellation inside the patch and on the boundary of the patch independently.

II *Tessellation*, which is not programmable, only controllable, performs the actual tessellation. It generates primitives. We can think of this step as where the topology of the tessellation is induced.

III *Evaluation* determines the position of the new tessellated vertex. It is based on affine transformations and performs in a manner similar to what the vertex shader does for a vertex, when combined with a basic primitive, such as point, line, triangle-strip and more.

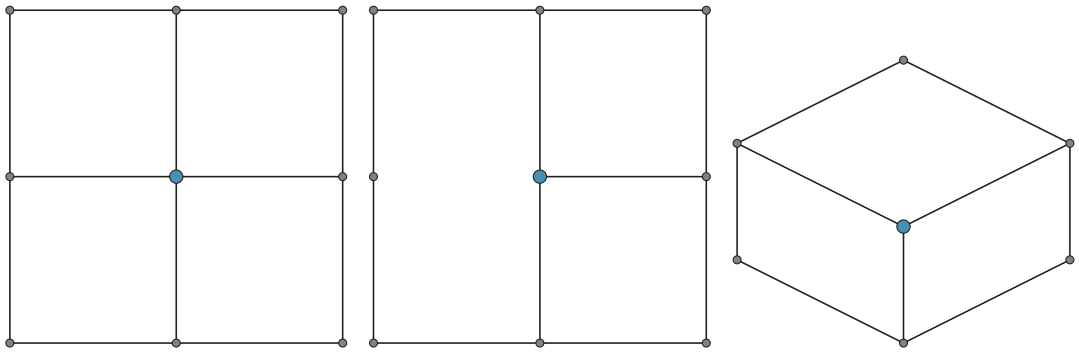

Figure 7.3: Three different types of render loci. From left to right: regular, T and star render loci.

A vertex generated by the tessellation step has a normalized position within the tessellation patch. This means that the evaluation step works on coordinates on the range $[0, 1]$ and, in the case of triangle type patch primitives, barycentric coordinates.

By considering the blending construction in (7.1) we propose building an evaluator based on the tessellation steps. Then, the tessellation patches of type line, triangle and quad, provided by the control step, take the roles as a render block on a blending type spline curve, -triangle surface and -surface, respectively. As we shall see in the following section, this is one layer in a hierarchical blending construction.

We conclude this section by mentioning briefly the roles of the two following steps. The tessellator determines parameter values stating where in the parametric domain a blending type curve or surface is to be evaluated. Finally, the tessellation evaluator is a shader implementation of a blending-type spline evaluator.

7.4 Render-lattice, -blocks and -loci

In [7] a concept for an ERBS-construction on irregular grids was presented. The concept is to divide spline knot nets into regular and irregular grids, leaving us with three different types of points at the knots; regular-, T- and star points, as shown in Figure 7.3. In [7] T- and star-points are defined as follows:

- A *T*-point is defined as a grid (parameter) line ending in an orthogonal grid line.
- A *Star*-point is defined as a point where several grid lines meets in a non-orthogonal way.

We propose the following descriptive names for a few of the grid components, when they are used for a rendering purpose:

- *Render lattice* to describe a grid structure arising from the net of spline knots.
- *Render locus* to describe loci in the render lattice, closely related to spline knots and regular-, T- and star-points.

- *Render block* to describe each line or face in a render lattice, i.e. the subset of the lattice that will be handled by a patch-type primitive.

The render block concept is closely connected to the description of the patch primitives of the GPU hardware. This provides some basic properties shared by all render blocks:

1. The domain of a render block is a parametric domain.
2. The parameter variables take values in the range $[0,1]$.
3. In order to meet the requirements associated with the ERBS Hermite interpolation properties, the outer tessellation levels of two adjacent patch primitives must be the same.

Below follows a set of definitions which describe the concepts behind the names introduced above.

Definition 7.1. *A* render locus *is an extension to the points defined in [7]. A* render locus *is defined as a locus in a render lattice associated with a spline knot on a regular or irregular spline net. A render locus can be one of three basic types, depending on the point type of the spline knot, as described in [7]. The three types are regular, T and star.*

Definition 7.2. *A* render block *is an extension to the patch-type primitive of modern tessellation based GPU architecture. The parametric domain of the render block is limited by the boundary given by two, three or four render loci. The number of render loci is decided by the type of patch-type primitive, which for line, triangle, or quadratic patch-type primitives are two, three or four, respectively. A point in the domain of a line, triangle or quadratic render locus has a normalized position ((u), (u,v,w), (u,v), respectively) in the parametric domain* $\Omega \in [0,1]$.

Definition 7.3. *A* render lattice *is defined in such a way that it coincides with the spline knot nets of the blending-type spline construction. Each locus on the render lattice is called a render locus. In a valid render lattice, render loci is divided and partitioned such that the render lattice consists of adjacently connected render blocks.*

Examples of regular and irregular render lattices with quadratic render blocks are shown in Figure 7.4. Throughout this article we shall consider regular render lattices with quadratic render blocks and regular render loci. T- and star-type render loci are subjects for future work.

7.4.1 Quadratic render block with regular render loci

A render block on a regular render lattice where all render loci are regular is defined in the following way:

$$P(u,v) = \sum_{i=1}^{2}\sum_{i=1}^{2} \ell_{i,j} \circ \omega_{i,j}(u,v) B_j(v) B_i(u)\,,$$

where $u,v \in [0,1]$ are parameters of the render block surface, and $\omega_{i,j}(u,v)$ are "map-to-local" functions mapping the parametric domain of the render block to the domain of the local surface patch associated with the given render locus.

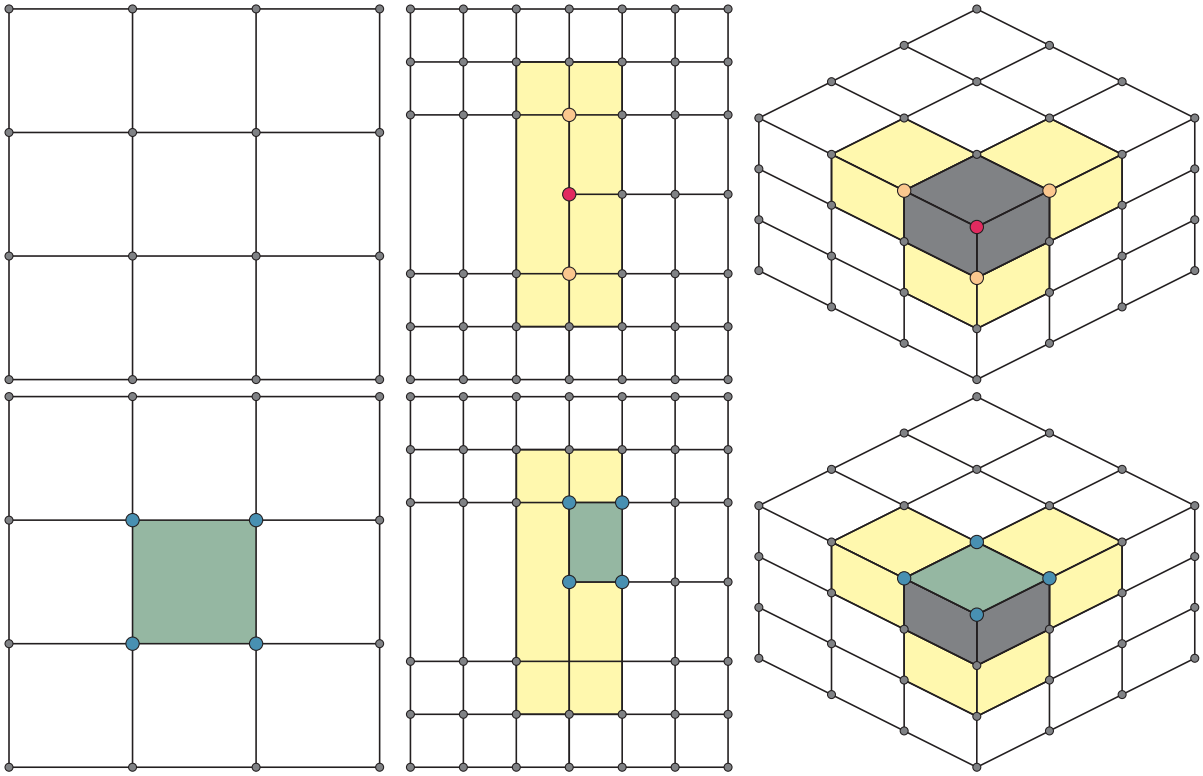

Figure 7.4: Three different render lattices. A render block is highlighted in each render lattice on the bottom row. The left column shows regular render loci. Regular and T-type render loci are shown in the middle column, whereas the right column contains regular and star-type render loci. The illustrations are provided to show the equivalence to the illustrations in [7, Figure 4].

Figure 7.5: The "Tower" surface: a blending spline tensor-product surface made up by 3×3 knots with an associated local plane surface patches. The local surfaces in the corners are locally rotated 90 degrees in the xy-plane and 45 degrees in xz-plane. The rendering lattice is here made up by 3×3 render loci (one for each spline knot locus) and four render blocks. On the left is a shaded version of the resulting surface, while on the right is a wireframe version showing the tessellation of the four render blocks.

The parametric domain of the patch-type primitive is normalized and the knot interval is always local, therefore, the "map-to-local" functions can be simplified from how they are described in [6]. For each parametric direction, u, v, the local mapping function, shown here for s, is defined as

$$\omega(s) = \gamma + \varkappa \times s,$$

where γ is the parametric offset of the local patch and $\varkappa$ is the scaling factor to the parametric domain of the local patch.

Figure 7.5 shows an example of a blending spline made up of 3×3 render loci, associated with one local patch each, and rendered over a render lattice consisting of four render blocks.

7.4.2 Implementation strategies

The method proposed above provides a rendering strategy for "global" geometric objects which are depending on evaluation of "local" geometry, possibly in several layers. When it comes to implementation, several issues related to optimization and performance, including the following, could be considered:

1. Evaluation of blending functions
2. Evaluation of local geometry
3. Data organization
4. Shader roles

In this article we shall not focus om implementation details, however, we find it appropriate to comment some of the above mentioned issues. Efficient evaluation of the original ERB was one major issue which, after it was addressed in [13], lead to the idea behind this article.

The second and third issues are tightly coupled. For this reason we prefer to see them in connection. We propose two valid, but different, strategies here. The first is based on using a general evaluation scheme of pre-sampled data. It is unnecessary to customize the shaders for each type of local patch, such as, for example, a cap of a sphere vs. a Bézier patch, as each local patch has to be pre-evaluated. One downside of this is that the pre-evaluated data must be stored and managed, another is that the precision of the local surface data is given by the resolution of the pre-sampling approximation step. On the positive side we note that, as a consequence of pre-evaluation, the evaluation time of a local patch would not depend on the patch type. Additionally, since the nature of the blending spline smoothly blends the local geometry, the resolution of the data of the local patches does not need to be as high.

The second strategy for dealing with the second and third issues is using custom built shaders for a given configuration of local render loci of a render block. We mention the following drawbacks of this approach; on-the-fly evaluation of local patches could be more expensive than a look-up in a pre-evaluated table. Furthermore, each shader on a render block must be regenerated whenever there is a change in the configuration of its render loci. Some major upsides with this approach are that it facilitates code modularization and the possibility of pixel-accurate [12, 5] resolution. Code modularization is specific to individual GPU architectures and APIs, but as an example, using OpenGL, one could divide each of the tessellation evaluation shader parts (BS evaluator, B-function evaluator, local patch evaluator and others) into shader objects and choose the appropriate ones when used.

The last issue is achieving efficiency by shader design. Using the features provided by the tessellation shader it is possible to generate patch geometry on the fly by circumventing the vertex shader altogether, providing only the coefficients of the local geometry to the shader. The final geometry is generated by the tessellation shader and the local geometry only exists as an evaluation result. A change in position, orientation or coefficient data of the local geometry would directly cause deformation to the rendered geometry. This change would not cause any additional computational costs as far as shader evaluation is concerned and a stable framerate would be maintained. This literally means keeping the spline representation all the way to the graphics hardware. Furthermore, it facilitates affine spatial transformations of the spline coefficients before they are provided to the rendering pipeline.

7.5 Concluding remarks

We have introduced a method for smooth rendering of blending-type splines where standard features of the tessellation shader architecture are exploited. The presented work has described a method for smooth rendering of blending spline constructions using tessellation based GPU architecture. In addition, fundamental render block types and its terminology has been proposed and described as well as strategies for implementation.

Some notable features of this method include, but are not limited to:

- Predictable and constant rendering time (in the sense of modification of the spline construction)
- The rendering method is local with respect to the render block

The method could be suitable for visualization in computer games or computer-generated imagery (CGI). Furthermore, since modification of the underlying spline construction adds little strain to the rendering method, it supports animation, simulations and interactivity, with little computational overhead.

The rendering method preserves the geometry description and topology until it is discretized in the hardware. Independent of implementation, the sampling is performed by the hardware instead of a defined procedure followed by pushing to the GPU.

Variable levels of detail (LODs) per render block could be achieved by using well known methods for setting the inner tessellation levels through the control step of the tessellation shader [12, 5]. The outer tessellation levels could be used to adjust and minimize artifacts over the boundary between two adjacent render blocks. This is of interest when the blending spline construction is used in application areas which results in large render patches, such as terrain representation.

The facts that the rendering method is strictly local (limited to a render block), and that it operates directly on the underlying spline construction, makes applications within interactive geometric modeling and sculpting interesting topics for future work.

A preliminary implementation supporting rendering of regular tensor-product surface render latices was created. The surfaces shown in Figure 7.5 were rendered using this preliminary implementation, which proves the concept. We propose focusing on efficient design for render blocks containing T- and star-type render loci on irregular render lattices in the next stages of research and development.

Inter-operable features between GPGPU specialized architecture APIs, such as OpenCL or CUDA, and graphics architecture APIs, such as OpenGL or DirectX, are available. For this reason, developing appropriate data structures and strategies for efficient data sharing and communication, is desirable.

A more comprehensible study on the efficiency of the solution should be conducted, as part of any specific implementation.

References

[1] J. Bratlie, R. Dalmo, and B. Bang. Evaluation of smooth spline blending surfaces using GPU. In J.-D. Boissonnat, A. Cohen, O. Gibaru, C. Gout, T. Lyche, M.-L. Mazure, and L. L. Schumaker, editors, *Curves and surfaces. 8th International Conference*, volume 9213 of *Lecture Notes in Computer Science*, pages 60–69. Springer, 2015.

[2] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[3] L. T. Dechevsky, A. Lakså, and B. Bang. Expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 27(3):319–362, 2006.

[4] M. Floater, T. Lyche, M.-L. Mazure, K. Mørken, and L. L. Schumaker, editors. *Mathematical Methods for Curves and Surfaces. 8th International Conference*, volume 8177 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2014.

[5] J. Hjelmervik. Direct pixel-accurate rendering of smooth surfaces. In Floater et al. [4], pages 238–247.

[6] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[7] A. Lakså. ERBS-surface construction on irregular grids. In V. Pasheva and G. Venkov, editors, *39th International conference applications of mathematics in engineering and economics AMEE13*, volume 1570 of *AIP Conference Proceedings*, pages 113–120. AIP Publishing, 2013.

[8] A. Lakså, B. Bang, and L. T. Dechevsky. Exploring expo-rational B-splines for curves and surfaces. In M. Dæhlen, K. Mørken, and L. L. Schumaker, editors, *Mathematical methods for Curves and Surfaces*, pages 253–262. Nashboro Press, 2005.

[9] Microsoft® corporation. Direct3D 11 features, 2009. `http://msdn.microsoft.com/en-us/library/ff476342(VS.85).aspx`.

[10] H. Schäfer, M. Nießner, B. Keinert, M. Stamminger, and C. Loop. State of the art report on real-time rendering with hardware tessellation. *Eurographics 2014 - State of the Art Reports*, pages 93–117, 2014.

[11] M. Segal and K. Akeley. The OpenGL graphics system: A specification (Version 4.0 (Core Profile)), March 2012.

[12] Y. Yeo, S. Bhandare, and J. Peters. Efficient pixel-accurate rendering of animated curved surfaces. In Floater et al. [4], pages 491–509.

[13] P. Zanaty. *Application of Generalized Expo-Rational B-splines in Computer Aided Design and Analysis*. PhD thesis, University of Oslo, 2014.

8 Matrix factorization of multivariate Bernstein polynomials

Rune Dalmo

This chapter is a reprint of [5]

Abstract — Ordinary univariate Bernstein polynomials can be represented in matrix form using factor matrices. In this paper we present the definition and basic properties of such factor matrices extended from the univariate case to the general case of arbitrary number of variables by using barycentric coordinates in the hyper-simplices of respective dimension. The main results in the paper are related to the design of an iterative algorithm for fast convex computation of multivariate Bernstein polynomials based on sparse-matrix factorization. In the process of derivation of this algorithm, we investigate some properties of the factorization, including symmetry, commutativity and differentiability of the factor matrices, and address the relevance of this factorization to the de Casteljau algorithm for evaluating curves and surfaces on Bézier form. A set of representative examples is provided, including a geometric interpretation of the de Casteljau algorithm, and representation by factor matrices of multivariate surfaces and their derivatives in Bézier form. Another new result is the observation that inverting the order of steps of a part of the new factorization algorithm provides a new, matrix-based, algebraic representation of a multivariate generalization of a special case of the de Boor-Cox computational algorithm.

8.1 Introduction

Univariate Bernstein polynomials were introduced by Sergei Bernstein [1] in his proof of the Weierstrass approximation theorem. They have been studied extensively in the literature since then, see, e.g. [17], or a textbook on approximation theory, e.g., [7].

Bernstein polynomials consitute the basis of Bézier curves and surfaces [12], and they are commonly used as basis functions in spline constructions [10, 20]. One way of representing Bernstein polynomials is by matrix form as factor matrices. Such matrices are related to de Casteljau's corner cutting algorithm [11]. The univariate case of Bernstein factor matrices was addressed in [16], however, univariate Bernstein factor matrices seem to have been exposed and used prior to that, see e.g. [3], where the de Casteljau algorithm is expressed on matrix form, or the method presented in [15, algorithm 12]. In addition, we mention that Tom Lyche and Knut Mørken have been using a slightly different matrix representation of B-splines in their lecture notes [18] at the university of Oslo. A particularly attractive property of the de Casteljau algorithm and its matrix-based version is the convexity of computation, whereby the total accumulated error of computation stays within the convex hull of the errors made in the process of the algorithm's iterations.

The definition of ordinary Bernstein polynomials can be generalized with preservation of the convexity-of-computation property by using barycentric coordinates, see e.g. [14, 19]. In this paper we investigate symmetric and recursive properties of the factor matrices for higher dimensions of the polynomials' argument. Furthermore, we present an equivalent recursive definition which facilitates the construction of Bernstein factor matrices for arbitrary dimensions of the barycentric argument, and investigate some properties which follow from the layout and construction of the matrices. We are interested in these findings mainly because the factorization can be seen to correspond to the de Casteljau algorithm and to a special case of the de Boor-Cox recursion formula for B-splines. Parts of the results, including an outline of the recursive definition of the Bernstein factor matrices, were announced in [6].

The organization of the paper is, as follows. In section 8.1.1 we provide a brief description of Bernstein polynomials, the Bézier representation of polynomial curves, and univariate Bernstein factor matrices. Section 8.1.2 is concerning some properties of the algebra of square matrices. We generalize the factor matrices to the multivariate setting in section 8.2, via first considering the case of $\mathbb{R}^2$, and then the d-variate case. Then in section 8.3 we address the directional derivatives of Bernstein polynomials and Bernstein factor matrices in the multivariate setting. Section 8.4 covers the relevance of the factorization to the de Casteljau algorithm, followed by some examples of how the construction can be used. Proofs of some of the lemmas and theorems are provided in section 8.5. Finally, in section 8.6, we give our concluding remarks where we suggest some topics for future work.

8.1.1 Univariate Bernstein polynomials and curves on Bézier form

Computing the binomial expansion

$$1 = (t + (1-t))^n = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i},$$

where $t \in \mathbb{R}$, leads to the Bernstein polynomials [1] of degree n,

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0, \ldots, n \,, \tag{8.1}$$

which satisfy a number of important properties, including linear independence, symmetry, roots at 0 and 1 only, partition of unity and positivity (i.e., convex partition of unity) in $(0,1)$. They can be expressed through the following (convex) recursion formula:

$$B_i^n(t) = tB_{i-1}^{n-1}(t) + (1-t)B_i^{n-1}(t) \,, \tag{8.2}$$

where $B_{-1}^{n-1} = B_n^{n-1} = 0$ and $B_0^0 = 1$.

Since the $n+1$ linearly independent Bernstein polynomials B_i^n form a basis for all polynomials of degree $\leq n$, any polynomial curve $c(t)$ of degree $\leq n$ has a unique n-th degree Bernstein-Bézier representation

$$c(t) = \sum_{i=0}^{n} \mathbf{c}_i B_i^n(t), \tag{8.3}$$

where the coefficients $\mathbf{c}_i \in \mathbb{R}^d$, called Bézier points [19], are elements of an affine space, i.e., the elements of $\mathbb{R}^d$ are points with coordinates defined with respect to a frame, consisting of origin with coordinates $(\underbrace{0, \ldots, 0}_{d \text{ times}})$ and the canonical orthonormal basis with coordinates $(\underbrace{0, \ldots, 0}_{i-1 \text{ times}}, 1, \underbrace{0, \ldots, 0}_{n-i-1 \text{ times}})$. We note that the Bernstein-Bézier representation is sometimes referred to as the B-form, as suggested by Carl de Boor in [9].

The curve in (8.3) can be evaluated using de Casteljau's corner cutting algorithm [11] by letting $\mathbf{c}_i^0 = \mathbf{c}_i$ and repeatedly applying the recursion formula for Bernstein polynomials in (8.2):

$$c(t) = \sum_{i=0}^{n} \mathbf{c}_i^0 B_i^n(t) = \sum_{i=0}^{n-1} \mathbf{c}_i^1 B_i^{n-1}(t) = \cdots = \sum_{i=0}^{0} \mathbf{c}_i^n B_i^0(t) = \mathbf{c}_0^n \,,$$

where $\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\mathbf{c}_{i+1}^{k-1}$ are intermediate points of de Casteljau's algorithm. This recursive convex linear interpolation between two points can be expressed in matrix form as outlined in [16]. As an example, computing from right to left, for $n = 3$:

$$c(t) = \begin{pmatrix} 1-t & t \end{pmatrix} \begin{pmatrix} 1-t & t & 0 \\ 0 & 1-t & t \end{pmatrix} \begin{pmatrix} 1-t & t & 0 & 0 \\ 0 & 1-t & t & 0 \\ 0 & 0 & 1-t & t \end{pmatrix} \begin{pmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{pmatrix}. \tag{8.4}$$

By consecutively multiplying the matrices in (8.4) from right to left, the dimension of the vector on the RHS is reduced by 1 for each computation. The factor matrices are of dimension $n \times (n+1)$ given by their number of rows and columns respectively. It follows that (8.3) can be rewritten as

$$c(t) = \mathbf{T}_1(t)\mathbf{T}_2(t)\mathbf{T}_3(t)\mathbf{c}, \tag{8.5}$$

where $\mathbf{c} = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)^T$ and the factor matrices are denoted by $\mathbf{T}_n(t)$. As noted in [15],

computing the three matrices from the left results in a vector containing the four Bernstein polynomials of degree three:

$$\begin{aligned}\mathbf{T}_1(t)\mathbf{T}_2(t)\mathbf{T}_3(t) &= \begin{pmatrix}(1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3\end{pmatrix}\\ &= \begin{pmatrix}B_0^3(t) & B_1^3(t) & B_2^3(t) & B_3^3(t)\end{pmatrix}.\end{aligned}$$

This method can be seen as a special case of the de Boor-Cox recursion formula [8, 4] for B-splines, since B-splines are the proper generalization of Bézier curves [13].

The curve $c(t)$ can be evaluated by multiplying the vector of Bernstein polynomials together with the vector of coefficients $\mathbf{c}$. We introduce the following matrix notation for the set of Bernstein polynomials of degree n:

$$\mathbf{B}^n(t) = \mathbf{T}_1(t)\mathbf{T}_2(t)\cdots\mathbf{T}_n(t). \tag{8.6}$$

Thus, by applying (8.6) to (8.5), the cubic Bézier curve in (8.4) can be expressed as

$$c(t) = \mathbf{B}^3(t)\mathbf{c}.$$

The binomial in (8.4) can be expressed using barycentric coordinates simply by substituting $1-t$ with v and t with w, so that (v,w) is a barycentric coordinate with respect to a line segment $L = \begin{pmatrix}1_0 & 1_1\end{pmatrix} \subset \mathbb{R}^1$, where $v+w=1$. The first three matrices of (8.4) would then become

$$\mathbf{T}_1\mathbf{T}_2\mathbf{T}_3 = \begin{pmatrix} v & w \end{pmatrix}\begin{pmatrix} v & w & 0 \\ 0 & v & w \end{pmatrix}\begin{pmatrix} v & w & 0 & 0 \\ 0 & v & w & 0 \\ 0 & 0 & v & w \end{pmatrix}. \tag{8.7}$$

Prior to introducing multivariate Bernstein polynomials, we provide a brief description of square matrices and some of their properties, which we shall use in some of the proofs and examples in the sequel of this paper.

8.1.2 The algebra of square matrices

N-th order square matrices form a vector space with respect to the ordinary summation of matrices and multiplication with a scalar [21, §93 (Example 1), §94]. Its dimension is N^2, and one basis in it is provided by the canonical matrices $E_{ij} = (\delta_{ik}\delta_{\ell j})_{k,\ell=1}^N$, $i,j = 1,\ldots,N$, where

$$\delta_{ij} = \begin{cases} 1, & \text{if} \quad i=j, \\ 0, & \text{otherwise,} \end{cases}$$

is the Kronecker delta. For the linear span of the so-defined canonical basis we shall use the notation $\mathbf{E} = \mathbf{E}_{N\times N} = \mathbf{E}_{N\times N}(\mathbb{F})$ where $\mathbb{F}$ denotes the field of scalars of the linear span and where, in our case, we shall always consider only the case $\mathbb{F} = \mathbb{R}$ (real scalars). It is well-known, that when endowed with matrix multiplication, $\mathbf{E}_{N\times N}$ becomes a non-commutative algebra [21, §93]. Since in the sequel we shall not be using all the properties of $\mathbf{E}_{N\times N}$ as an algebra (for example, we shall not be discussing inverse elements in $\mathbf{E}_{N\times N}$), our choice here is to provide a self-consistent presentation of only those of the properties of the algebra $\mathbf{E}_{N\times N}$

which are of relevance to this article through the next few lemmas.

Lemma 8.1. *The dimension of the space* $\mathbf{E}_{N\times N}$ *is determined by the largest dimension of the matrices involved in a given multiplication. For instance, let*

$$\mathbf{AB}=\mathbf{C},$$

where $\mathbf{A}$, $\mathbf{B}$ *and* $\mathbf{C}$ *are matrices with dimensions* $K\times L$, $L\times M$ *and* $K\times M$, *respectively. Then* $N=\max\{K,L,M\}$, *and we can, for instance, express the matrix* $\mathbf{A}$ *as*

$$\mathbf{E}_{N\times N}=\sum_{k=1}^{N}\sum_{l=1}^{N}a_{kl}E_{kl},$$

where a_{kl} *are the entries in* $\mathbf{A}$, *which we define to be zero if* $k>K$ *or if* $l>L$. *The result is a space* $\mathbf{E}_{N\times N}$ *which contains* $\mathbf{A}$ *as element of the linear span of* E_{ij} *for the first* K *values of* i *and the first* L *values of* j, *i.e., corresponding to a* $K\times L$*-rectangular submatrix situated in the upper left corner of the* $N\times N$ *square matrix. Entries from* $\mathbf{B}$ *and* $\mathbf{C}$ *can be obtained analogously.*

Lemma 8.1 allows us to simplify the considerations for multiplication of several algebrae [21, §94 (item 2)], to a consideration within one single algebra $\mathbf{E}_{N\times N}$ of sufficiently high order N [21, §93 (item 1)].

Lemma 8.2. *The product of two basis matrices is*

$$E_{ij}E_{k\ell}=\delta_{jk}E_{i\ell}, \tag{8.8}$$

where $i,j,k,\ell=1,\ldots,N$.

Formula (8.8) is a concise form of the respective formula in [21, §93 (item 1)] and a simplification of the respective formulae in [21, §94 (item 2)]. This will prove essential in the proofs of our main results, because there (8.8) will be used iteratively.

Lemma 8.3. *Let* $\mathbf{A}=(a_{ij})_{i=1,j=1}^{N,N}\in\mathbf{E}_{N\times N}$, $\mathbf{B}=(b_{kl})_{k=1,l=1}^{N,N}\in\mathbf{E}_{N\times N}$, $\mathbf{C}=(c_{mn})_{m=1,n=1}^{N,N}\in\mathbf{E}_{N\times N}$ *and* $\mathbf{D}=(d_{pq})_{p=1,q=1}^{N,N}\in\mathbf{E}_{N\times N}$ *be four matrices. Then* $\mathbf{AB}-\mathbf{CD}$ *is an element of* $\mathbf{E}_{N\times N}$, *defined by*

$$\mathbf{AB}-\mathbf{CD}=\sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}\left[\sum_{\lambda=1}^{N}\left(a_{\alpha\lambda}b_{\lambda\beta}-c_{\alpha\lambda}d_{\lambda\beta}\right)\right]E_{\alpha\beta}. \tag{8.9}$$

For the particular case when $\mathbf{C}=\mathbf{B}$, $\mathbf{D}=\mathbf{A}$ in formula (8.9) we obtain the following:

Corollary 8.1. *The commutator of* $\mathbf{A}=(a_{ij})_{i=1,j=1}^{N,N}\in\mathbf{E}_{N\times N}$ *and* $\mathbf{B}=(b_{kl})_{k=1,l=1}^{N,N}\in\mathbf{E}_{N\times N}$ *is an element of* $\mathbf{E}_{N\times N}$, *defined by*

$$[\mathbf{A},\mathbf{B}]=\sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}\left[\sum_{\lambda=1}^{N}\left(a_{\alpha\lambda}b_{\lambda\beta}-a_{\lambda\beta}b_{\alpha\lambda}\right)\right]E_{\alpha\beta}.$$

8.2 Multivariate Bernstein polynomials

The symmetric properties of the Bernstein polynomials suggest that barycentric coordinates are a natural choice to represent multivariate scenarios. We proceed by considering factorization of the Bernstein polynomials in two variables expressed in barycentric coordinates with respect to a triangle in $\mathbb{R}^2$. Then we shall generalize the resulting factor matrices to the multivariate setting, with a recursive definition based on a particular decomposition of the matrices into submatrices.

8.2.1 Bivariate Bernstein polynomials

Proceeding as in (8.1), we compute the trinomial expansion

$$1 = (u+v+w)^n = \sum_{i,j,k} \binom{n}{i,j,k} u^i v^j w^k, \tag{8.10}$$

where $i,j,k \geq 0$ and $i+j+k=n$. This leads to the Bernstein polynomials of degree n,

$$B^n_{ijk}(u,v,w) = \binom{n}{i,j,k} u^i v^j w^k, \tag{8.11}$$

where (u,v,w) is the barycentric coordinate of a point $\mathbf{p}$ with respect to a triangle $A = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2) \subset \mathbb{R}^2$, $(i,j,k) \in \{0,1,\ldots,n\}^3$, and $i+j+k=n$. We note that (u,v,w) represents a local parameter with respect to A and $\mathbf{p}$ a global parameter, since $\mathbf{p} = A\mathbf{u} = \mathbf{a}_0 u + \mathbf{a}_1 v + \mathbf{a}_2 w$, with $u+v+w=1$.

The recurrence relation for Bernstein polynomials associated with a triangle $A \subset \mathbb{R}^2$ in homogeneous barycentric coordinates is defined (see e.g. [14, 19]) as

$$B^n_{ijk} = uB^{n-1}_{i-1,j,k} + vB^{n-1}_{i,j-1,k} + wB^{n-1}_{i,j,k-1}, \tag{8.12}$$

where we consider expressions with negative subscripts to be zero. It is well known [14] that the Bernstein polynomials

$$\mathbf{B}^n_2 = \{B^n_{ijk}\}_{i+j+k=n} \tag{8.13}$$

form a basis for the space of polynomials of degree n. Thus, given any triangle A, every polynomial p of degree n has a unique Bernstein-Bézier representation

$$p = \sum_{i+j+k=n} c_{ijk} B^n_{ijk}, \tag{8.14}$$

where B^n_{ijk} are the Bernstein polynomials associated with A.

In the sequel we shall assume a lexicographic order of the $\binom{n+2}{2}$ Bernstein polynomials, as provided in (8.13) and (8.14), such that $B^n_{rst} > B^n_{ijk}$ when $r > i$, or if $r = i$, then $s > j$, or if $r = i$ and $s = j$, then $t > k$. As an example, the order of B^n_{ijk} for $i+j+k=n$ and $n=3$ is

the following:

$$B^3_{300} > B^3_{210} > B^3_{201} > B^3_{120} > B^3_{111} > B^3_{102} > B^3_{030} > B^3_{021} > B^3_{012} > B^3_{003}.$$

We use the recurrence relation in (8.12) to define the triangular (bivariate) Bernstein factor matrix in homogeneous barycentric coordinates, as follows:

Definition 8.1. *The Bernstein factor matrix* $\mathbf{T}_{2,n}(u,v,w)$ *is a* $\binom{n+1}{2} \times \binom{n+2}{2}$ *band-limited matrix with three non-zero elements on each row. The columns of* $\mathbf{T}_{2,n}(u,v,w)$ *correspond to the* $\binom{n+2}{2}$ *vectors of terms of the recurrence relation in* (8.12) *in lexicographic order, such that the non-zero elements in every column are*

1. *positioned according to the lexicographic index numbers of the* B^{n-1}_{ijk} *on the RHS, and*
2. *taking the values of their associated variables;* u*,* v*, or* w*.*

It follows immediately from Definition 8.1 that the first three triangular (bivariate) factor matrices in homogeneous barycentric coordinates are as follows:

$$\mathbf{T}_{2,1} = \begin{pmatrix} u & v & w \end{pmatrix}, \tag{8.15}$$

$$\mathbf{T}_{2,2} = \left(\begin{array}{ccc|ccc} u & v & w & 0 & 0 & 0 \\ \hline 0 & u & 0 & v & w & 0 \\ 0 & 0 & u & 0 & v & w \end{array}\right), \tag{8.16}$$

$$\mathbf{T}_{2,3} = \left(\begin{array}{ccccc|cccc} u & v & w & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & u & 0 & v & w & 0 & 0 & 0 & 0 \\ 0 & 0 & u & 0 & v & w & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & u & 0 & 0 & v & w & 0 \\ 0 & 0 & 0 & 0 & u & 0 & 0 & v & w \\ 0 & 0 & 0 & 0 & 0 & u & 0 & 0 & v \end{array}\right), \tag{8.17}$$

where the horizontal and vertical lines are used to annotate sub-matrices, which we shall describe later.

We find it appropriate to include the following lemma, which states that the proposed factor matrices are a factorization of the set of Bernstein polynomials in three variables in barycentric coordinates.

Lemma 8.4. *The Bernstein factor matrices* $\{\mathbf{T}_{2,i}\}_{i=1}^{n}$ *factor the Bernstein polynomials* $\mathbf{B}_2^n$*:*

$$\mathbf{T}_{2,1}\mathbf{T}_{2,2}\cdots\mathbf{T}_{2,n} = \mathbf{B}_2^n. \tag{8.18}$$

By investigating the matrices in (8.7) and (8.15)-(8.17) we observe that a specific Bernstein factor matrix consists of four sub-matrices: The upper left sub-matrix is the factor matrix of the same dimension but of one degree lower. The lower right sub-matrix is the factor matrix of one dimension lower but of the same degree. The upper right sub-matrix is a zero matrix. In the lower left sub-matrix the only non-zero elements are on the diagonal from one of the entries on the top to the lower right element. We shall use the term right diagonal, since

it can be seen as a main diagonal which is shifted as far as possible to the right in a square matrix.

It turns out that this relation holds for arbitrary degree of the factor matrices. We, therefore, summarize the findings in the following lemma.

Lemma 8.5. *The factor matrix* $\mathbf{T}_{2,n}(u,v,w)$*, for degree* $n>1$*, consists of four sub-matrices, such that*

$$\mathbf{T}_{2,n}(u,v,w)=\left(\begin{array}{c|c}\mathbf{T}_{2,n-1}(u,v,w) & \mathbf{0}\\ \hline \mathbf{T}^{\text{diag}}_{2,n}(u) & \mathbf{T}_{1,n}(v,w)\end{array}\right),$$

where $\mathbf{T}^{\text{diag}}_{2,n}(u)$ *is a matrix where the only non-zero elements are on the right diagonal and equal to* u*.*

8.2.2 The d-variate case, $d=2,3,\ldots$

Multivariate Bernstein polynomials over a d-dimensional simplex $A\subset\mathbb{R}^d$ are defined in [19] as follows:

$$B^n_{\mathbf{i}}(\mathbf{u})=\binom{n}{\mathbf{i}}\mathbf{u}^{\mathbf{i}}=\binom{n}{i_0,\ldots,i_d}u_0^{i_0}\ldots u_d^{i_d}, \tag{8.19}$$

where $\mathbf{i}=(i_0,\ldots,i_d)\in\{0,1,\ldots,n\}^{d+1}$, $|\mathbf{i}|=i_0+\cdots+i_d=n$, $\mathbf{u}=(u_0,\ldots,u_d)$ is the barycentric coordinate of a point $\mathbf{p}$ with respect to A, $\mathbf{u}^{\mathbf{i}}=(u_0^{i_0}\ldots u_d^{i_d})$, and $u_0+\cdots+u_d=1$. There are $\binom{n+d}{d}$ Bernstein polynomials of degree n. They form a basis for all d-variate polynomials of total degree $\leq n$, they form a partition of unity and are positive for $\mathbf{u}>0$, which is one reason to consider Bernstein polynomials over the simplex A.

The following recurrence relation holds for the Bernstein polynomials in (8.19):

$$B^n_{\mathbf{i}}(\mathbf{u})=u_0B^{n-1}_{\mathbf{i}-\mathbf{e}_0}+\cdots+u_dB^{n-1}_{\mathbf{i}-\mathbf{e}_d}, \tag{8.20}$$

where $\mathbf{e}_0,\ldots,\mathbf{e}_d$ represents the columns of the $(d+1)\times(d+1)$ identity matrix, $B^0_{\mathbf{0}}=1$, $B^0_{\mathbf{i}}=0$ if $\mathbf{i}$ has negative components, and $|\mathbf{i}-\mathbf{e}_j|=n-1$. We shall assume a lexicographic order of the multivariate Bernstein polynomials, similar to the $\mathbb{R}^2$ case, based on the index numbers $i_0,\ldots,i_d$. The Bernstein factor matrices for arbitrary number of variables are defined as follows:

Definition 8.2. *The Bernstein factor matrix* $\mathbf{T}_{d,n}(u_0,\ldots,u_d)$ *is a* $\binom{n+d-1}{d}\times\binom{n+d}{d}$ *band-limited matrix with* d *non-zero elements on each row. The columns of* $\mathbf{T}_{d,n}(u_0,\ldots,u_d)$ *correspond to the* $\binom{n+d}{d}$ *vectors of terms in the recurrence relation* (8.20) *in lexicographic order, such that the non-zero elements in every column are*

1. *positioned according to the lexicographic index numbers of the* $B^{n-1}_{i-\mathbf{e}_j}$ *on the RHS, and*
2. *taking the values of their associated variables;* $u_0,u_1,\cdots,$ *or* u_d*.*

We recall Lemma 8.5, which can be extended to the multivariate case. An outline of a proof is to use the recurrence relation provided in (8.20) and Definition 8.2, which is similar to the proof of Lemma 8.5. We use this to propose an alternative equivalent definition of

the multivariate Bernstein factor matrices of arbitrary degree, based on recursion of the sub-matrices, as summarized in the following theorem, which is a main result:

Theorem 8.1. *The Bernstein factor matrix* $\mathbf{T}_{d,n}(\mathbf{u})$ *is a* $\binom{n+d-1}{d} \times \binom{n+d}{d}$ *band-limited matrix with* $d+1$ *nonzero elements on each row. The matrix is defined recursively as follows:*

$$\mathbf{T}_{d,n}(u_0,\ldots,u_d) = \left(\begin{array}{c|c} \mathbf{T}_{d,n-1}(u_0,\ldots,u_d) & 0 \\ \hline \mathbf{T}^{\text{diag}}_{d,n}(u_0) & \mathbf{T}_{d-1,n}(u_1,\ldots,u_d) \end{array}\right), \tag{8.21}$$

where $d \geq 0$ *is the dimension,* $n \geq 1$ *is the degree,* $\mathbf{T}_{0,q} = (u_d)$ *for* $q \leq n$, $\mathbf{T}_{p,1} = (u_{d-p},\ldots,u_d)$ *for* $p \leq d$, *and* $\mathbf{T}^{\text{diag}}_{q,n}(u_{d-q})$ *is a matrix where the only non-zero elements are on the right diagonal and equal to* u_{d-q}.

In the following we will in some cases omit specifying the parameters of the matrix $\mathbf{T}$ and its sub-matrices for simplification. We note that the sub-matrices in (8.21) are such that $\mathbf{T}_{d,n-1}$ is of size $\binom{n+d-2}{d} \times \binom{n+d-1}{d}$, $\mathbf{T}_{d-1,n}$ is $\binom{n+d-2}{d-1} \times \binom{n+d-1}{d-1}$ and $\mathbf{T}^{\text{diag}}_{d,n}$ is $\binom{d+n-2}{d} \times \binom{n+d-1}{d-1}$. The upper right sub-matrix is a $\binom{d+n-2}{d} \times \binom{n+d-1}{d-1}$ matrix where all the entries are zero.

Lemma 8.6. *The entries in every row in* $\mathbf{T}_{d,n}(\mathbf{u})$ *are either* $0, u_0, \ldots,$ *or* u_d, *such that each of the elements* $u_0,\ldots,u_d$ *occurs once, in increasing order, with optional zeros in-between.*

For the sake of clarity, we formalize that the proposed factor matrices are a factorization of the set of Bernstein polynomials in d variables, similar to Lemma 8.4 for the $\mathbb{R}^2$ case, in the following lemma.

Lemma 8.7. *The Bernstein factor matrices* $\{\mathbf{T}_{d,i}\}_{i=1}^{n}$ *factor the Bernstein polynomials* $\mathbf{B}^n_d$:

$$\mathbf{T}_{d,1}\mathbf{T}_{d,2}\cdots\mathbf{T}_{d,n} = \mathbf{B}^n_d. \tag{8.22}$$

We include the following lemma concerning a symmetry property of the Bernstein factor matrices:

Lemma 8.8. *Given the barycentric coordinates* $\mathbf{u} = (u_0,\ldots,u_d)$ *and* $\mathbf{z} = (z_0,\ldots,z_d)$ *with respect to a* d*-dimensional simplex A. Then*

$$\mathbf{T}_{d,k-1}(\mathbf{z})\mathbf{T}_{d,k}(\mathbf{u}) = \mathbf{T}_{d,k-1}(\mathbf{u})\mathbf{T}_{d,k}(\mathbf{z}). \tag{8.23}$$

8.3 Directional derivatives

Let us consider the line given by

$$l(t) = \mathbf{p} + t\mathbf{v},$$

where $\mathbf{p} = \mathbf{a}_0 u_0 + \cdots + \mathbf{a}_d u_d$ where $u_0 + \cdots + u_d = 1$ is a point in $\mathbb{R}^d$ with barycentric coordinates

$$\mathbf{u} = (u_0, u_1, \ldots, u_d), \tag{8.24}$$

and $\mathbf{v} = \mathbf{a}_0 v_0 + \cdots + \mathbf{a}_d v_d$ where $v_0 + \cdots + v_d = 0$ is a vector with barycentric coordinates

$$\mathbf{v} = (v_0, v_1, \ldots, v_d), \tag{8.25}$$

and a polynomial $s(\mathbf{u})$ in Bernstein-Bézier form:

$$s(\mathbf{p}) = \sum_{|\mathbf{i}|=n} \mathbf{c}_\mathbf{i} B_\mathbf{i}^n(\mathbf{u}). \tag{8.26}$$

Consider $\mathbb{R}^{d+1}$ with Cartesian coordinates $u_0, u_1, \ldots, u_d$, a $(d+1)$-variate sufficiently smooth function $\sigma(u_0, u_1, \ldots, u_d)$ and a unit vector in $\mathbb{R}^{d+1}$ with Cartesian coordinates $(v_0, v_1, \ldots, v_d)$:

$$\sum_{k=0}^{d} v_k^2 = 1. \tag{8.27}$$

The 1-st directional derivative of σ at $\mathbf{u}^0 = (u_0^0, u_1^0, \ldots, u_d^0) \in \mathbb{R}^{d+1}$ in the direction of the vector $\mathbf{v} = (v_0, v_1, \ldots, v_d)$ satisfying (8.27) is, as habitual,

$$\lim_{t \to 0+} \frac{\mathrm{d}}{\mathrm{d}t} \sigma(t), \tag{8.28}$$

where $\sigma(t) = \sigma(\mathbf{u}^0 + t\mathbf{v})$, $t \in (0, t_0)$, $0 < t_0 < \infty$.

Definition 8.3. *The 1-st directional derivative of $s(\mathbf{u}) = s(u_0, u_1, \ldots, u_d)$ at the point $\mathbf{p} = (u_0^0, u_1^0, \ldots, u_d^0) \in \mathbb{R}^d$ in the direction of the vector $\mathbf{v} = (v_0^0, v_1^0, \ldots, v_d^0) \in \mathbb{R}^d$ where u_k^0, v_k^0, $k = 0, 1, \ldots, d$ are barycentric coordinates in $\mathbb{R}^d$ satisfying* (8.24) *and* (8.25)*, respectively, is the directional derivative in $\mathbb{R}^{d+1}$ in the particular case when $\mathbf{u}^0 = \mathbf{p}$ belongs to the affine manifold (hyperplane in $\mathbb{R}^{d+1}$) $\left\{(u_0, u_1, \ldots, u_d) \in \mathbb{R}^{d+1} : \sum_{k=0}^{d} u_k = 1\right\}$ and the directional vector $\mathbf{v}$ belongs to the intersection of the unit sphere $\left\{(v_0, v_1, \ldots, v_d) \in \mathbb{R}^{d+1} : \sum_{k=0}^{d} v_k^2 = 1\right\}$ and the d-dimensional subspace (hyperplane in $\mathbb{R}^{d+1}$) $\left\{(v_0, v_1, \ldots, v_d) \in \mathbb{R}^{d+1} : \sum_{k=0}^{d} v_k^2 = 0\right\}$.*

Directional derivatives in barycentric coordinates of order higher than first are defined analogously.

From the chain rule, using also the properties (8.24) and (8.25) of the barycentric coordinates and Definition 8.3, we have that

$$D_\mathbf{v} s(\mathbf{p}) = v_0 D_{u_0} s(\mathbf{p}) + \cdots + v_d D_{u_d} s(\mathbf{p}).$$

Partial derivatives of a Bernstein polynomial with respect to each of its parameters can be obtained as

$$\frac{\partial}{\partial u_j} B_\mathbf{i}^n = n B_{\mathbf{i}-\mathbf{e}_j}^{n-1},$$

where $\mathbf{i}$ is a multiindex, $|\mathbf{i}| = n$, $|\mathbf{i}-\mathbf{e}_j| = n-1$, and $B_\mathbf{j} = 0$ if $\mathbf{j}$ has a negative component. This gives rise to the following lemma, which can be found in a trivariate setting in [14]:

Lemma 8.9. *For arbitrary $i_0 + \cdots + i_d = n$, the directional derivative of a Bernstein polynomial*

$B_{\mathbf{i}}^n(\mathbf{p})$ *in the direction of* $\mathbf{v}$ *is*

$$D_{\mathbf{v}}B_{\mathbf{i}}^n(\mathbf{p}) = n\left[v_0 B_{\mathbf{i}-\mathbf{e}_0}^{n-1}(\mathbf{u}) + \cdots + v_d B_{\mathbf{i}-\mathbf{e}_d}^{n-1}(\mathbf{u})\right]. \tag{8.29}$$

Recall from Lemma 8.7 that

$$\mathbf{B}_d^n(\mathbf{u}) = \mathbf{T}_{d,1}(\mathbf{u})\mathbf{T}_{d,2}(\mathbf{u})\cdots\mathbf{T}_{d,n}(\mathbf{u}). \tag{8.30}$$

This product of matrices can be differentiated by the same formulae as if the factors were numbers. If $\mathbf{T}(\mathbf{u})$ is a matrix whose entries are functions of $\mathbf{u}$, then the derivative in the direction of $\mathbf{v}$, $D_{\mathbf{v}}\mathbf{T}$ of $\mathbf{T}$, is defined as the matrix obtained by differentiating each entry of $\mathbf{T}$ with respect to $\mathbf{v}$. The entries are obtained by differentiation of linear convex combinations, which yields constants independent of $\mathbf{u}$.

We formalize the definition of the derivative of the Bernstein factor matrix $\mathbf{T}_{d,n}(\mathbf{u})$:

Lemma 8.10. *The derivative of a Bernstein factorial matrix in the direction of* $\mathbf{v} = (v_0, \ldots, v_d)$,

$$D_{\mathbf{v}}\mathbf{T}_{d,n} = \mathbf{T}_{d,n}(\mathbf{v}),$$

is a $\binom{d+n-1}{n-1} \times \binom{d+n}{n}$ *band-limited matrix with* $d+1$ *nonzero constant elements on each row (independent of* $\mathbf{u}$*).*

We note here that the submatrix $D_{\mathbf{v}}\mathbf{T}_{d,n}^{\text{diag}}$ is zero if $v_0 = 0$, since the only non-zero elements in $\mathbf{T}_{d,n}^{\text{diag}}$ are equal to v_0. Furthermore, since $\mathbf{T}_{d-1,n}(\mathbf{v})$ does not contain v_0, the submatrix $D_{\mathbf{v}}\mathbf{T}_{d-1,n}$ is zero if $v_1 = \cdots = v_d = 0$, i.e. if $D_{\mathbf{v}}$ corresponds to the partial derivative D_{u_0}.

The remainder of this section is partially based on an analogous treatment of univariate B-splines on matrix form in [18], adjusted here to the setting of multivariate Bernstein polynomials. The following well-known rule for differentiating a product of two matrices will be used in order to find the derivative of the factorization.

Lemma 8.11. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be two matrices with compatible dimensions for the matrix product* $\mathbf{AB}$ *and with entries that are functions of* $(u_0, \ldots, u_d)$*. Then*

$$D(\mathbf{AB}) = (D\mathbf{A})\mathbf{B} + \mathbf{A}(D\mathbf{B}).$$

By applying the rule from Lemma 8.11 to the product of (8.30), we get

$$D_{\mathbf{v}}\mathbf{B}_d^n(\mathbf{u}) = \sum_{k=1}^{n} \mathbf{T}_{d,1}(\mathbf{u})\cdots\mathbf{T}_{d,k-1}(\mathbf{u})D_{\mathbf{v}}\mathbf{T}_{d,k}\mathbf{T}_{d,k+1}(\mathbf{u})\cdots\mathbf{T}_{d,n}(\mathbf{u}). \tag{8.31}$$

Formula (8.31) can be simplified by applying the following lemma:

Lemma 8.12. *The matrices* $\mathbf{T}_{d,k-1}$ *and* $\mathbf{T}_{d,k}$ *satisfy the relation*

$$D_{\mathbf{v}}\mathbf{T}_{d,k-1}\mathbf{T}_{d,k}(\mathbf{u}) = \mathbf{T}_{d,k-1}(\mathbf{u})D_{\mathbf{v}}\mathbf{T}_{d,k}, \tag{8.32}$$

for $k \geq 2$, *any* $\mathbf{u} = (u_0, \ldots, u_d)$, $u_k \in \mathbb{R}$, $k = 0, 1, \ldots, d$, *and any given direction* $\mathbf{v} = (v_0, \ldots, v_d)$ *with* $\sum_{k=0}^{d} v_k^2 > 0$.

The differentiation operator $D_{\mathbf{v}}$ in (8.31) can be shifted between the matrices by using (8.32), and by making use of Lemma 8.10 we obtain

$$D_{\mathbf{v}} \mathbf{B}_d^n(\mathbf{u}) = n \mathbf{T}_{d,1}(\mathbf{u}) \cdots \mathbf{T}_{d,n-1}(\mathbf{u}) D_{\mathbf{v}} \mathbf{T}_{d,n} = n \mathbf{B}_d^{n-1} \mathbf{T}_{d,n}(\mathbf{v}). \tag{8.33}$$

We are now ready to provide one of the main results, a theorem which states that the r-th directional derivative of the set of Bernstein polynomials $\mathbf{B}_d^n$ can be obtained by differentiating r of the factor matrices.

Theorem 8.2. *Given a set of Bernstein polynomials* $\mathbf{B}_d^n(\mathbf{u})$ *and the directions* $\mathbf{v}_1, \ldots, \mathbf{v}_r$, *for* $1 \leq r \leq n$. *Then*

$$D_{\mathbf{v}_1} \cdots D_{\mathbf{v}_r} \mathbf{B}_d^n(\mathbf{u}) = \frac{n!}{(n-r)!} \mathbf{B}_d^{n-r}(\mathbf{u}) \mathbf{T}_{d,n-r+1}(\mathbf{v}_1) \cdots \mathbf{T}_{d,n}(\mathbf{v}_r). \tag{8.34}$$

We recall from (8.26) that any d-variate polynomial of degree n has a Bernstein-Bézier representation,

$$s(\mathbf{u}) = \mathbf{B}_d^n(\mathbf{u}) \mathbf{c},$$

where $\mathbf{c} = \begin{pmatrix} \mathbf{c}_0 & \cdots & \mathbf{c}_d \end{pmatrix}^T$. Next, we obtain the following via applying Theorem 8.2:

Corollary 8.2. *Given a polynomial* $s(\mathbf{u})$ *on Bernstein-Bézier form and the directions* $\mathbf{v}_1, \ldots, \mathbf{v}_r$. *Then*

$$D_{\mathbf{v}_1} \cdots D_{\mathbf{v}_r} s(\mathbf{p}) = \frac{n!}{(n-r)!} \mathbf{B}_d^{n-r}(\mathbf{u}) \mathbf{T}_{d,n-r+1}(\mathbf{v}_1) \cdots \mathbf{T}_{d,n}(\mathbf{v}_r) \mathbf{c}. \tag{8.35}$$

8.4 Relevance to the de Casteljau algorithm

The Bernstein polynomials form a basis; thus, every d-variate polynomial $s(\mathbf{u})$ has a Bernstein-Bézier representation with respect to a reference simplex A:

$$s(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{c}_{\mathbf{i}}^0 B_{\mathbf{i}}^n(\mathbf{u}), \tag{8.36}$$

which can be evaluated with de Casteljau's algorithm by applying the recurrence relation in (8.20). Similar to the case of curves, we obtain

$$s(\mathbf{u}) = \sum_{|\mathbf{i}|=n-1} \mathbf{c}_{\mathbf{i}}^1 B_{\mathbf{i}}^{n-1}(\mathbf{u}),$$

and, after $n-2$ steps,

$$s(\mathbf{u}) = \sum_{|\mathbf{i}|=0} \mathbf{c}_{\mathbf{i}}^n B_{\mathbf{i}}^0(\mathbf{u}) = \mathbf{c}_{0\ldots0},$$

where $\mathbf{c}_{\mathbf{i}}^{k} = u_0\mathbf{c}_{\mathbf{i}+\mathbf{e}_0}^{k-1} + \cdots + u_d\mathbf{c}_{\mathbf{i}+\mathbf{e}_d}^{k-1}$, with $|\mathbf{i}| = n, \ldots, 0$, are intermediate coefficients. These intermediate coefficients are polynomials of degree k.

The directional derivative of a polynomial in Bernstein-Bézier form is formulated, based on a description in [19], as follows.

Lemma 8.13. *The directional derivative of a d-variate polynomial $s(\mathbf{u})$ on Bernstein-Bézier form, at the point $\mathbf{p}$ in the direction of $\mathbf{v}$, is given by*

$$D_{\mathbf{v}}s(\mathbf{p}) = n\sum_{\mathbf{j}}\mathbf{d}_{\mathbf{j}}B_{\mathbf{j}}^{n-1}(\mathbf{u}),$$

where

$$\mathbf{d}_{\mathbf{j}} = v_0\mathbf{c}_{\mathbf{j}+\mathbf{e}_0} + \cdots + v_d\mathbf{c}_{\mathbf{j}+\mathbf{e}_d}$$

are the intermediate points resulting from the first step of the de Casteljau algorithm based on the barycentric coordinates of $\mathbf{v}$.

We observe that the coefficients of the first derivative of a polynomial on Bernstein-Bézier form are as provided by Lemma 8.13 and verify that $D_{\mathbf{v}}s$ at the point $\mathbf{p}$ with barycentric coordinates $\mathbf{u}$ can be evaluated using the de Casteljau algorithm by applying one step of the algorithm using $\mathbf{v}$ and then $n-1$ steps using $\mathbf{u}$.

We abbreviate the intermediate points $\mathbf{d}_{\mathbf{j}}$, using the notation from [19], by $\mathbf{d}_{\mathbf{j}} = \Delta_{\mathbf{v}}\mathbf{c}_{\mathbf{j}}$. Higher-order derivatives are obtained in the same way, e.g., an r-th directional derivative $D_{\mathbf{v}_1}\cdots D_{\mathbf{v}_r}s$ has the Bézier coefficients $\Delta_{\mathbf{v}_1}\cdots\Delta_{\mathbf{v}_r}\mathbf{c}_{\mathbf{j}}$, where $|\mathbf{j}| = n-r$. The following result is obtained by applying Lemma 8.13 repeatedly.

Lemma 8.14. *Given the directions $\mathbf{v}_1, \ldots, \mathbf{v}_r$, for $1 \le r \le n$. Then,*

$$D_{\mathbf{v}_1}\cdots D_{\mathbf{v}_r}s(\mathbf{p}) = \frac{n!}{(n-r)!}\sum_{\mathbf{j}=n-r}\mathbf{d}_{\mathbf{j}}^{(r)}B_{\mathbf{j}}^{n-r}(\mathbf{u}),$$

where $\mathbf{d}_{\mathbf{j}}^{(r)}$ are the intermediate points obtained after performing r steps of the de Casteljau algorithm using $\mathbf{v}_1, \ldots, \mathbf{v}_r$ in this order.

One consequence of Lemma 8.14 is that $D_{\mathbf{v}_1}\cdots D_{\mathbf{v}_r}s$ can be evaluated at the point $\mathbf{p}$ with barycentric coordinates $\mathbf{u}$ by first applying r steps of the de Casteljau algorithm using $\mathbf{v}_1\cdots\mathbf{v}_r$ in this order, and then by using $\mathbf{u}$ in the following $n-r$ steps.

The forward difference operator $\Delta_{\mathbf{v}}$ commutes with the steps of the de Casteljau algorithm [19] since the computation of affine combinations of affine combinations is commutative:

$$\sum_i \alpha_i \sum_j \beta_j \mathbf{p}_{ij} = \sum_j \beta_j \sum_i \alpha_i \mathbf{p}_{ij}.$$

Thus, an r-th directional derivative can also be obtained by first computing $n-r$ steps of the de Casteljau algorithm followed by r differentiation steps.

We obtain the following result; by using Theorem 8.1, Lemma 8.7, and (8.20), formula (8.36) can be expressed in matrix form:

Corollary 8.3. *Any d-variate polynomial $s(\mathbf{u})$ in Bernstein-Bézier form,*

$$s(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{c}_{\mathbf{i}}^0 B_{\mathbf{i}}^n(\mathbf{u}),$$

can be expressed by using Bernstein factor matrices as:

$$s(\mathbf{u}) = \mathbf{T}_{d,1}(\mathbf{u})\mathbf{T}_{d,2}(\mathbf{u})\cdots\mathbf{T}_{d,n}(\mathbf{u})\mathbf{c}, \tag{8.37}$$

where $\mathbf{c} = \begin{pmatrix}\mathbf{c}_0 & \cdots & \mathbf{c}_d\end{pmatrix}^T$. *Computing the RHS in* (8.37) *from right to left corresponds to the steps of the de Casteljau algorithm.*

We note that computing only the matrices $\mathbf{T}_{d,1}\cdots\mathbf{T}_{d,n}$, from left to right, yields a vector containing the $\binom{d+n}{d}$ Bernstein polynomials of degree n in $d+1$ variables with respect to the simplex $A \subset \mathbb{R}^d$:

$$\mathbf{B}_d^n(\mathbf{u}) = \mathbf{T}_{d,1}(\mathbf{u})\cdots\mathbf{T}_{d,n}(\mathbf{u}).$$

One important new observation here is that, in a manner similar to the case of curves, this method can be seen to correspond to a special case of the de Boor-Cox recursion formula for B-splines in a multivariate setting.

We conclude this section by recalling Corollary 8.2 and Lemma 8.14 to make another important new observation: namely Corollary 8.3 is applicable also for computations of directional derivatives.

8.4.1 Examples

The de Casteljau algorithm as an iterative procedure

The de Casteljau algorithm can be described as an iterative process by using projection and a linear step. Let us recall the expression in (8.5) for a cubic parametric curve on Bézier form by factorization using matrices, where in general for degree n we obtain:

$$c(t) = \mathbf{T}_1(t)\mathbf{T}_2(t)\cdots\mathbf{T}_n(t)\mathbf{v}_0,$$

where $\mathbf{v}_0 = \begin{pmatrix}\mathbf{c}_0 & \cdots & \mathbf{c}_0\end{pmatrix}^T = \begin{pmatrix}\mathbf{c}_0^0 & \mathbf{c}_1^0 & \cdots & \mathbf{c}_{n-1}^0 & \mathbf{c}_n^0\end{pmatrix}^T$ is a vector containing the coefficients, or Bézier points, for the curve. Then we extend the matrices $\mathbf{T}_k(t)$ for $1 \le k \le n$, as described in Lemma 8.1, to obtain a product of n square matrices of size $(n+1)^2$, multiplied with $\mathbf{v}_0$. After performing one step of the de Casteljau algorithm, which corresponds to multiplying together the two rightmost factors in the product, we find

$$c(t) = \underbrace{\mathbf{T}_1(t)\cdots\mathbf{T}_{n-1}(t)}_{\substack{\text{n-1 times matrices}\\ \text{of size } (n+1)^2}}\mathbf{v}_1,$$

where $\mathbf{v}_1 = \begin{pmatrix} \mathbf{c}_0^1(t) & \mathbf{c}_1^1(t) & \cdots & \mathbf{c}_{n-1}^1(t) & 0 \end{pmatrix}^T$. We investigate the difference between the two steps by considering the difference between the vectors of coefficients:

$$\mathbf{v}_1 - \mathbf{v}_0 = \begin{pmatrix} \mathbf{c}_0^1(t) - \mathbf{c}_0^0 \\ \mathbf{c}_1^1(t) - \mathbf{c}_1^0 \\ \vdots \\ \mathbf{c}_{n-1}^1(t) - \mathbf{c}_{n-1}^0 \\ 0 - \mathbf{c}_n^0 \end{pmatrix} = \mathbf{d}_1.$$

Recall that the intermediate points $\mathbf{c}_i^k = (1-t)\mathbf{c}_i^{k-1} + t\,\mathbf{c}_{i+1}^{k-1}$. Then the first $n-1$ entries in $\mathbf{d}_1$ are

$$\mathbf{c}_i^1(t) - \mathbf{c}_i^0 = (1-t)\mathbf{c}_i^0 + t\,\mathbf{c}_{i+1}^0 - \mathbf{c}_i^0 = t(\mathbf{c}_{i+1}^0 - \mathbf{c}_i^0).$$

It follows that the difference vector $\mathbf{d}_1$ can be decomposed into an orthogonal projection from $\mathbb{R}^{n+1}$ onto $\mathbb{R}^n$, and a linear step:

$$\mathbf{d}_1 = \mathbf{p}_1 + \mathbf{l}_1,$$

where

$$\mathbf{p}_1 = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -\mathbf{c}_n^0 \end{pmatrix}, \quad \text{and} \quad \mathbf{l}_1 = t \begin{pmatrix} \mathbf{c}_1^0 - \mathbf{c}_0^0 \\ \mathbf{c}_2^0 - \mathbf{c}_1^0 \\ \vdots \\ \mathbf{c}_n^0 - \mathbf{c}_{n-1}^0 \\ 0 \end{pmatrix}.$$

In general, we find that the k-th step of the iterative procedure, which yields

$$c(t) = \underbrace{\mathbf{T}_1(t) \cdots \mathbf{T}_{n-k}(t)}_{\substack{\text{n-k times matrices} \\ \text{of size } (n+1)^2}} \mathbf{v}_k,$$

consists of a similar projection from $\mathbb{R}^{n+2-k}$ onto $\mathbb{R}^{n+1-k}$ and a linear step, such that

$$\mathbf{d}_k = \mathbf{v}_k - \mathbf{v}_{k-1} = \mathbf{p}_k + \mathbf{l}_k,$$

where

$$\mathbf{p}_k = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ -\mathbf{c}_{n-k}^{k-1}(t) \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \quad \text{and} \quad \mathbf{l}_k = t \begin{pmatrix} \mathbf{c}_1^{k-1}(t) - \mathbf{c}_0^{k-1}(t) \\ \vdots \\ \mathbf{c}_{n-k}^{k-1}(t) - \mathbf{c}_{n-k-1}^{k-1}(t) \\ 0 \\ \vdots \\ 0 \end{pmatrix}.$$

The process terminates when

$$c(t) = \mathbf{v}_n = \left(\mathbf{c}_0^n(t)\right) = \mathbf{c}_0^n(t).$$

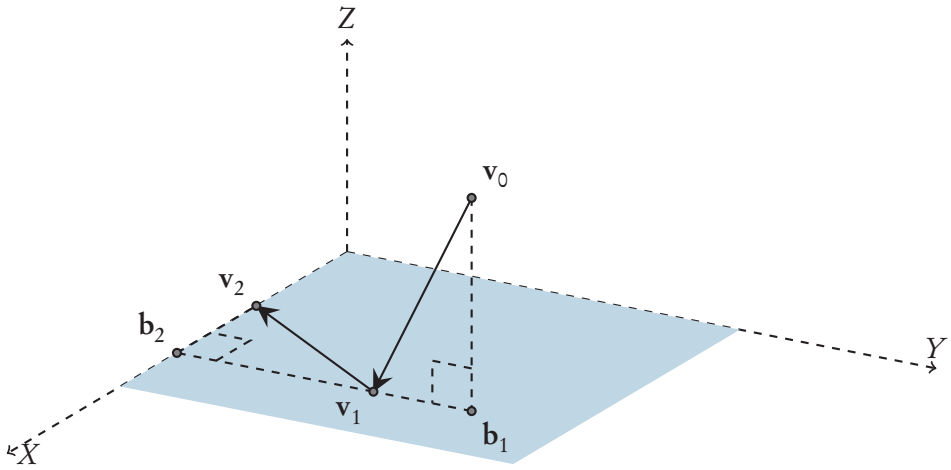

Figure 8.1: An illustration of the steps of the de Casteljau algorithm by iteratively performing a orthonormal projection followed by a linear step, for a parametric curve, with $n = 2$. The initial coefficients, or points, are contained in the vector $\mathbf{v}_0 \in \mathbb{R}^3$. $\mathbf{b}_1$ is the orthogonal projection of $\mathbf{v}_0$ onto $\mathbb{R}^2 \subset \mathbb{R}^3$, $\mathbf{v}_1$ constitutes the vector of intermediate points after the first step of the algorithm, and $\mathbf{b}_2$ is the orthogonal projection of $\mathbf{v}_1$ onto $\mathbb{R}^1 \subset \mathbb{R}^2$. The final step yields the vector $\mathbf{v}_2 \in \mathbb{R}^1$. It contains one point which constitutes the value of the curve at the parameter value t.

Figure 8.1 shows how the vectors of coefficients are repeatedly projected onto a subspace and translated by a linear step depending on t for the case of a parametric curve with $n = 2$. The number of coefficients, or points, is reduced by one for each projection until one single point remains.

In this example, $\mathbb{R}^{n+1-k} = W_k$ is a subset of the Euclidean space $\mathbb{R}^{n+2-k}$, and $\mathbf{b}_k = \mathbf{v}_{k-1} + \mathbf{p}_k$ is the orthogonal projection of $\mathbf{v}_{k-1}$ onto W_k. Then $\mathbf{b}_k$ is the closest point in W_k to $\mathbf{v}_{k-1}$ in the sense that $||\mathbf{v}_{k-1} - \mathbf{b}_k|| < ||\mathbf{v}_{k-1} - \mathbf{x}||$ for all $\mathbf{x}$ in W_k distinct from $\mathbf{b}_k$. It follows that $\mathbf{b}_k$ is the best approximation to $\mathbf{v}_{k-1}$ by elements of W_k.

We note that the space of square matrices used in the considered example is a complete inner product space (i.e., a Hilbert space) with the Euclidean norm $||\cdot||_{\ell^2}$. The method can be generalized to uniformly convex Banach spaces with ℓ^p-norm, with $1 < p < \infty$, since there will still exist a unique best approximation in such cases [7]. However, that will most likely yield non-polynomial special functions, in general, but they may share some properties with the polynomial case, where $p = 2$.

Representing a parametric surface

In this example we show how directional derivatives of a multivariate cubic Bézier representation of a parametric surface $s(\mathbf{u})$ can be expressed in matrix form by factorization. First of all, we consider the surface

$$s(\mathbf{u}) = \mathbf{B}_d^3(\mathbf{u})\mathbf{c},$$

where $\mathbf{B}_d^3(\mathbf{u})$ are the d-variate Bernstein polynomials of degree 3, and $\mathbf{c}$ are its Bézier coefficients. The derivative of $s(\mathbf{u})$ at the point $\mathbf{p}$ in the direction of $\mathbf{v}_0$,

$$D_{\mathbf{v}_0} s(\mathbf{p}) = D_{\mathbf{v}_0} \mathbf{B}_d^3(\mathbf{u})\mathbf{c},$$

can be expressed as a factorization of the Bernstein polynomials by matrices:

$$D_{\mathbf{v}_0} s(\mathbf{p}) = D_{\mathbf{v}_0} [\mathbf{T}_1(\mathbf{u})\mathbf{T}_2(\mathbf{u})\mathbf{T}_3(\mathbf{u})]\mathbf{c},$$

which computes to

$$D_{\mathbf{v}_0} s(\mathbf{p}) = \Big[D_{\mathbf{v}_0}\mathbf{T}_1\mathbf{T}_2(\mathbf{u})\mathbf{T}_3(\mathbf{u}) + \mathbf{T}_1(\mathbf{u})D_{\mathbf{v}_0}\mathbf{T}_2\mathbf{T}_3(\mathbf{u}) + \mathbf{T}_1(\mathbf{u})\mathbf{T}_2(\mathbf{u})D_{\mathbf{v}_0}\mathbf{T}_3\Big]\mathbf{c},$$

and by applying Lemma 8.12 we obtain

$$D_{\mathbf{v}_0} s(\mathbf{p}) = 3\mathbf{B}_d^2(\mathbf{u})D_{\mathbf{v}_0}\mathbf{T}_3\mathbf{c}.$$

We proceed by applying one more differentiation step, this time in the direction of $\mathbf{v}_1$:

$$D_{\mathbf{v}_1} D_{\mathbf{v}_0} s(\mathbf{p}) = D_{\mathbf{v}_1}\Big[3\mathbf{B}_d^2(\mathbf{u})D_{\mathbf{v}_0}\mathbf{T}_3\mathbf{c}\Big] = 3D_{\mathbf{v}_1}\mathbf{B}_d^2(\mathbf{u})D_{\mathbf{v}_0}\mathbf{T}_3\mathbf{c},$$

which computes to

$$D_{\mathbf{v}_1} D_{\mathbf{v}_0} s(\mathbf{p}) = 3\Big[D_{\mathbf{v}_1}\mathbf{T}_1\mathbf{T}_2(\mathbf{u}) + \mathbf{T}_1(\mathbf{u})D_{\mathbf{v}_1}\mathbf{T}_2\Big]D_{\mathbf{v}_0}\mathbf{T}_3\mathbf{c},$$

and by applying Lemma 8.12 again and re-ordering the terms we obtain

$$D_{\mathbf{v}_1} D_{\mathbf{v}_0} s(\mathbf{p}) = 6\mathbf{B}_d^1(\mathbf{u})D_{\mathbf{v}_1}\mathbf{T}_2 D_{\mathbf{v}_0}\mathbf{T}_3\mathbf{c}.$$

With the use of Lemma 8.10 we write

$$D_{\mathbf{v}_1} D_{\mathbf{v}_0} s(\mathbf{p}) = 6\mathbf{B}_d^1(\mathbf{u})\mathbf{T}_2(\mathbf{v}_1)\mathbf{T}_3(\mathbf{v}_0)\mathbf{c}.$$

Commutativity of multiplication

We shall now look at how a special property of the multivariate Bernstein factor matrix $\mathbf{T}_{d,n}(\mathbf{u})$ can be used to provide an alternative proof of Lemma 8.8 by using induction.

Let us recall from (8.23) that:

$$\mathbf{T}_{d,k-1}(\mathbf{z})\mathbf{T}_{d,k}(\mathbf{u}) = \mathbf{T}_{d,k-1}(\mathbf{u})\mathbf{T}_{d,k}(\mathbf{z}),$$

for $d \geq 0$ and $n \geq 1$.

Proof. The proof is based on induction, on the variables d and n, of (8.23) of the fully-ordered (well-ordered) set of all (d, n). We equip $\mathbb{N} \times \mathbb{N}$ with the lexicographic order relation $\leq$ defined by $(d, n) \leq (s, t)$ if ($(d < s)$ or $(d = s$ and $n \leq t)$). This is a full-order relation on $\mathbb{N} \times \mathbb{N}$, which means that we may use the induction theorem. Details on full-ordered sets and

induction can be found in [2]. We proceed as follows to prove a property P for all (d,n) in $\mathbb{N}\times\mathbb{N}$:

1. Show that P holds for $(d,n)=(0,1)$.

2. Suppose that P holds for all elements of $\mathbb{N}\times\mathbb{N}$ which are less than some arbitrary (d,n). Show that P holds for (d,n).

3. Conclude that P holds for all (d,n) in $\mathbb{N}\times\mathbb{N}$ by the induction theorem for the well-ordered sets.

1. Using $d=0,n=1$ in (8.23) gives the following relation:

$$\mathbf{T}_{0,1}(u)\mathbf{T}_{0,2}(z)=\mathbf{T}_{0,1}(z)\mathbf{T}_{0,2}(u),$$

which yields

$$\begin{pmatrix} u \end{pmatrix}\begin{pmatrix} z \end{pmatrix}=\begin{pmatrix} z \end{pmatrix}\begin{pmatrix} u \end{pmatrix}, \tag{8.38}$$

and it follows that both the LHS and the RHS of (8.38) compute to uz, since the product of two numbers commutes.

2. We want to prove $P(d,n)$ using any $(p,q)<(d,n)$. When applying (8.21) and its derivative from the definitions above, the left-hand side of (8.23) becomes

$$\left(\begin{array}{c|c} \mathbf{T}_{d,n-1}(\mathbf{u}) & 0 \\ \hline \mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u}) & \mathbf{T}_{d-1,n}(\mathbf{u}) \end{array}\right)\left(\begin{array}{c|c} \mathbf{T}_{d,n}(\mathbf{z}) & 0 \\ \hline \mathbf{T}^{\mathrm{diag}}_{d,n+1}(\mathbf{z}) & \mathbf{T}_{d-1,n+1}(\mathbf{z}) \end{array}\right),$$

which can be written as

$$\left(\begin{array}{c|c} \mathbf{T}_{d,n-1}(\mathbf{u})\mathbf{T}_{d,n}(\mathbf{z}) & 0 \\ \hline \mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})\mathbf{T}_{d,n}(\mathbf{z})+\mathbf{T}_{d-1,n}(\mathbf{u})\mathbf{T}^{\mathrm{diag}}_{d,n+1}(\mathbf{z}) & \mathbf{T}_{d-1,n}(\mathbf{u})\mathbf{T}_{d-1,n+1}(\mathbf{z}) \end{array}\right). \tag{8.39}$$

A similar computation for the RHS of (8.23) yields

$$\left(\begin{array}{c|c} \mathbf{T}_{d,n-1}(\mathbf{z})\mathbf{T}_{d,n}(\mathbf{u}) & 0 \\ \hline \mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z})\mathbf{T}_{d,n}(\mathbf{u})+\mathbf{T}_{d-1,n}(\mathbf{z})\mathbf{T}^{\mathrm{diag}}_{d,n+1}(\mathbf{u}) & \mathbf{T}_{d-1,n}(\mathbf{z})\mathbf{T}_{d-1,n+1}(\mathbf{u}) \end{array}\right). \tag{8.40}$$

It follows that both the upper left and the lower right sub-matrices of (8.39) are equivalent to the corresponding sub-matrices of (8.40) by definition of the induction step.

Next, we need to check that the lower left sub-matrices of (8.39) and (8.40) are equivalent:

$$\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})\mathbf{T}_{d,n}(\mathbf{z})+\mathbf{T}_{d-1,n}(\mathbf{u})\mathbf{T}^{\mathrm{diag}}_{d,n+1}(\mathbf{z})=\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z})\mathbf{T}_{d,n}(\mathbf{u})+\mathbf{T}_{d-1,n}(\mathbf{z})\mathbf{T}^{\mathrm{diag}}_{d,n+1}(\mathbf{u}). \tag{8.41}$$

The non-zero elements of $\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})$ and $\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z})$ are equal to u_0 and v_0, respectively. Thus, the LHS of (8.41) becomes

$$u_0\begin{pmatrix} \mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z}) & \mathbf{T}_{d-1,n}(\mathbf{z}) \end{pmatrix}+z_0\begin{pmatrix} 0 & \mathbf{T}_{d-1,n}(\mathbf{u}) \end{pmatrix}.$$

Similarly, we obtain for the RHS of (8.41):

$$z_0\Big(\ \mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})\ \Big|\ \mathbf{T}_{d-1,n}(\mathbf{u})\ \Big)+u_0\Big(\ 0\ \Big|\ \mathbf{T}_{d-1,n}(\mathbf{z})\ \Big).$$

Re-ordering the terms and inserting them into (8.41) yields

$$\Big(\ u_0\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z})\ \Big|\ u_0\mathbf{T}_{d-1,n}(\mathbf{z})+z_0\mathbf{T}_{d-1,n}(\mathbf{u})\ \Big)=\Big(\ z_0\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})\ \Big|\ u_0\mathbf{T}_{d-1,n}(\mathbf{z})+z_0\mathbf{T}_{d-1,n}(\mathbf{u})\ \Big),$$

and it follows that LHS = RHS in (8.41), since $\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{u})$ contains u_0s and $\mathbf{T}^{\mathrm{diag}}_{d,n}(\mathbf{z})$ contains v_0s, and since the product of two real scalars is commutative.

3. Finally, we conclude that (8.23) holds for all (d,n) in $\mathbb{N}\times\mathbb{N}$ provided $d\geq 0, n\geq 1$. □

8.5 Proofs

This section contains proofs of some of the lemmas and theorems presented earlier in this article.

Proof of Lemma 8.2

Proof. We consider the product

$$E_{ij}E_{k\ell}=E_{\alpha\beta},$$

where the entries in E_{ij} are

$$c_{\mu\nu}=\begin{cases}1, & \text{if}\quad \mu=i,\nu=j,\\ 0, & \text{otherwise,}\end{cases}$$

and the elements of $E_{k\ell}$ are

$$d_{\lambda\eta}=\begin{cases}1, & \text{if}\quad \lambda=k,\eta=\ell,\\ 0, & \text{otherwise.}\end{cases}$$

Since

$$E_{ij}=\left(\delta_{i\mu}\delta_{\nu j}\right)_{\mu=1,\nu=1}^{N,N},$$

and

$$E_{k\ell}=\left(\delta_{k\lambda}\delta_{\eta\ell}\right)_{\lambda=1,\eta=1}^{N,N},$$

the elements of $E_{\alpha\beta}$ can be expressed as

$$e_{\alpha\beta}=\sum_{a=1}^{N}c_{\alpha a}d_{a\beta}=\sum_{a=1}^{N}\delta_{i\alpha}\delta_{aj}\delta_{ka}\delta_{\beta\ell}.$$

Fixing $\alpha=i$ and $\beta=\ell$ yields

$$e_{\alpha\beta}=\sum_{a=1}^{N}\delta_{aj}\delta_{ka},$$

and for $a = j$ and $k = a$ we obtain

$$e_{\alpha\beta} = \sum_{a=j=k}^{N} 1 = 1.$$

Thus,

$$e_{\alpha\beta} = \delta_{kj}\delta_{i\alpha}\delta_{\ell\beta},$$

which means that

$$e_{i\ell} = \delta_{kj}.$$

Finally, we obtain

$$E_{ij}E_{kl} = \delta_{kj}E_{i\ell},$$

and the result follows. □

Proof of Lemma 8.3

Proof.

$$\mathbf{AB}-\mathbf{CD} = \left(\sum_{i=1}^{N}\sum_{j=1}^{N} a_{ij}E_{ij}\right)\left(\sum_{k=1}^{N}\sum_{\ell=1}^{N} b_{k\ell}E_{k\ell}\right) - \left(\sum_{m=1}^{N}\sum_{n=1}^{N} c_{mn}E_{mn}\right)\left(\sum_{p=1}^{N}\sum_{q=1}^{N} d_{pq}E_{pq}\right),$$

where the RHS computes to

$$\mathbf{AB}-\mathbf{CD} = \sum_{i}\sum_{j}\sum_{k}\sum_{\ell} a_{ij}b_{k\ell}E_{ij}E_{k\ell} - \sum_{m}\sum_{n}\sum_{p}\sum_{q} c_{mn}d_{pq}E_{mn}E_{pq}.$$

By using the multiplier in (8.8) for $E_{ij}E_{k\ell}$ and $E_{mn}E_{pq}$ and re-ordering the sums we obtain

$$\mathbf{AB}-\mathbf{CD} = \sum_{i}\sum_{j}\sum_{k}\sum_{\ell} a_{ij}b_{k\ell}\delta_{jk}E_{i\ell} - \sum_{m}\sum_{n}\sum_{p}\sum_{q} c_{mn}d_{pq}\delta_{np}E_{mq},$$

where setting $j = k = \mu$ and $n = p = \nu$ yields

$$\mathbf{AB}-\mathbf{CD} = \sum_{i}\sum_{\ell}\left(\sum_{\mu} a_{i\mu}b_{\mu\ell}\right)E_{i\ell} - \sum_{m}\sum_{q}\left(\sum_{\nu} c_{m\nu}d_{\nu q}\right)E_{mq}.$$

Then we set $i = \alpha$, $\ell = \beta$, $m = \alpha$ and $q = \beta$, and we get

$$\begin{aligned}\mathbf{AB}-\mathbf{CD} &= \sum_{\alpha}\sum_{\beta}\left(\sum_{\mu} a_{\alpha\mu}b_{\mu\beta}\right)E_{\alpha\beta} - \sum_{\alpha}\sum_{\beta}\left(\sum_{\nu} c_{\alpha\nu}d_{\nu\beta}\right)E_{\alpha\beta} \\ &= \sum_{\alpha}\sum_{\beta}\left(\sum_{\mu} a_{\alpha\mu}b_{\mu\beta} - \sum_{\nu} c_{\alpha\nu}d_{\nu\beta}\right)E_{\alpha\beta}.\end{aligned}$$

Finally, we fix $\nu = \mu = \lambda$, and the result follows. □

Proof of Corollary 8.1

Proof. The result follows by setting $\mathbf{C} = \mathbf{B}$ and $\mathbf{D} = \mathbf{A}$ in Lemma 8.3 and re-ordering the terms, since $[\mathbf{A}, \mathbf{B}] = \mathbf{AB} - \mathbf{BA}$. □

Proof of Lemma 8.4

Proof. We use induction on the variable n of (8.18). It is trivial that $\mathbf{T}_{2,1} = \mathbf{B}_2^1 = (u, v, w)$. Assuming that it holds for $n = r - 1$ and setting $n = r$ in (8.18) yields

$$\mathbf{B}_2^{r-1}\mathbf{T}_{2,r} = \mathbf{B}_2^r.$$

But then the columns of $\mathbf{T}_{2,r}$ must correspond to the variables u, v and w in the RHS of the recurrence relation (8.12) for $n = r$, where the values u, v and w are indexed within every column according to the position of the indexes of the B_{ijk}^{r-1} in lexicographic order, which is exactly $\mathbf{T}_{2,r}$ by Definition 8.1. We conclude that (8.18) holds for arbitrary n by induction. □

Proof of Lemma 8.5

Proof. The proof is based on using the recurrence relation (8.12) and Definition 8.1. $\mathbf{T}_{2,n-1}(u, v, w)$ corresponds by Definition 8.1 to the set of all $\binom{n+1}{2}$ RHS terms of B_{ijk}^{n-1} in (8.12) for $i + j + k = n - 1$. By Definition 8.1, a similar collection of terms from the first $\binom{n+1}{2}$ vectors obtained from (8.12), for B_{ijk}^n, which corresponds to the elements where $i > 0$ since $i + j + k = n$, results in a matrix whose upper part contains $\mathbf{T}_{2,n-1}(u, v, w)$ and its lower part contains $\mathbf{T}_{2,n}^{\text{diag}}(u)$. Here, the upper part consists of the first $\binom{n}{2}$ rows, whereas the remaining $\binom{n+1}{2} - \binom{n}{2}$ rows consitute the lower part. Similarly, collecting the remaining $\binom{n+2}{2} - \binom{n+1}{2}$ vectors results in a matrix where the upper part is a zero matrix and the non-zero elements in the lower part are either v or w, since $i = 0$, and they follow the layout of $\mathbf{T}_{1,n}(v, w)$. □

Proof of Lemma 8.6

Proof. According to Definition 8.2, the columns in $\mathbf{T}_{d,n}(\mathbf{u})$ are arranged such that

1. The order of the columns are based on the lexicographic order of the permutations of $\mathbf{i}$ for $|\mathbf{i}| = n$, and

2. u_i is placed on the $(\mathbf{i} - \mathbf{e}_i)$-th row within the column, if $0 \leq (\mathbf{i} - \mathbf{e}_i) < \binom{d+n-1}{d}$.

The possible results of computing $\mathbf{i} - \mathbf{e}_i$, for $i = 0, \ldots, d$ leads to the $\binom{n+d-1}{d}$ permutations of $\mathbf{j} = (j_0, \ldots, j_d)$, where $|\mathbf{j}| = n - 1$. There are $d + 1$ possible ways to obtain the position j_k, for $k = 1, \ldots, \binom{n+d-1}{d}$ in the lexicographic order of the values of $\mathbf{j}$, by taking $\mathbf{i} - \mathbf{e}_i$. Since the values of $\mathbf{i}$ are ordered, the values of j_k will appear in order such that the associated values of u_i are in the order $u_0, \ldots, u_d$. Since j_k corresponds to a row in $\mathbf{T}_{d,n}(\mathbf{u})$, the result follows. □

Proof of Lemma 8.7

Proof. We use use induction in the variable n of (8.22) for arbitrary degree d.
1. By setting $n=2$ in the LHS of (8.22) and applying Theorem 8.1 we obtain

$$\mathbf{T}_{d,1}\mathbf{T}_{d,2}=\begin{pmatrix}u_0 & u_1 & \cdots & u_d\end{pmatrix}\left(\begin{array}{c|c}\mathbf{T}_{d,1}(u_0,\ldots,u_d) & 0\\ \hline \mathbf{T}^{\text{diag}}_{d,n}(u_0) & \mathbf{T}_{1,n}(u_1,\ldots,u_d)\end{array}\right),$$

which computes to

$$\begin{pmatrix}u_0^2 & 2u_1u_0 & 2u_2u_0 & \cdots & u_1^2 & 2u_2u_1 & \cdots & 2u_du_1 & u_2^2 & \cdots & u_d^2\end{pmatrix}=\mathbf{B}_d^2,$$

thus, (8.22) holds for $n=2$.

2. Setting $n=r$ in (8.22) gives

$$\mathbf{T}_{d,1}\mathbf{T}_{d,2}\cdots\mathbf{T}_{d,r}=\mathbf{B}_d^r. \tag{8.42}$$

Assuming that $\mathbf{T}_{d,1}\mathbf{T}_{d,2}\cdots\mathbf{T}_{d,r-1}=\mathbf{B}_d^{r-1}$, we obtain

$$\mathbf{B}_d^{r-1}\mathbf{T}_{d,r}=\mathbf{B}_d^r. \tag{8.43}$$

Using the recurrence formula in (8.20), formula (8.43) holds if, and only if, the columns of $\mathbf{T}_{d,n}$ correspond to the variables $u_0,\ldots,u_d$ in the RHS of the recurrence relation in (8.20) for $n=r$, where the values $u_0,\ldots,u_d$ are indexed within every column according to the position of the indices of the associated $B^{r-1}_{i-\mathbf{e}_j}$ in lexicographic order. But, this is true according to Definition 8.2; hence, we conclude that (8.42) holds and that $\mathbf{T}_{d,1}\mathbf{T}_{d,2}\cdots\mathbf{T}_{d,n}$ is a factorization of the set of Bernstein polynomials $\mathbf{B}_d^n$. □

Proof of Lemma 8.8

Proof. Consider the following relation:

$$\mathbf{T}_{d,k-1}(\mathbf{z})\mathbf{T}_{d,k}(\mathbf{u})-\mathbf{T}_{d,k-1}(\mathbf{u})\mathbf{T}_{d,k}(\mathbf{z}),$$

where $\mathbf{u}=(u_0,\ldots,u_d)$ and $\mathbf{z}=(z_0,\ldots,z_d)$ are barycentric coordinates with respect to a d-dimensional simplex A. Using Lemma 8.3, this translates to

$$\sum_{\alpha=1}^{N}\sum_{\beta=1}^{N}\left[\sum_{\lambda=1}^{N}\left(a_{\alpha\lambda}b_{\lambda\beta}-c_{\alpha\lambda}d_{\lambda\beta}\right)\right]E_{\alpha\beta}, \tag{8.44}$$

where $a_{\alpha\lambda}, b_{\lambda\beta}, c_{\alpha\lambda}$ and $d_{\lambda\beta}$ are elements of $\mathbf{T}_{d,k-1}(\mathbf{z})$, $\mathbf{T}_{d,k}(\mathbf{u})$, $\mathbf{T}_{d,k-1}(\mathbf{u})$ and $\mathbf{T}_{d,k}(\mathbf{z})$, respectively. Next, we change the order of the sums to

$$\sum_{\alpha=1}^{N}\sum_{\lambda=1}^{N}\sum_{\beta=1}^{N}\left(a_{\alpha\lambda}b_{\lambda\beta}-d_{\lambda\beta}c_{\alpha\lambda}\right)E_{\alpha\beta}.$$

Then the terms $a_{\alpha\lambda}b_{\lambda\beta}$ are obtained in an order such that, for each row α, every entry in the λ-th column of $\mathbf{T}_{d,k-1}(\mathbf{z})$ is multiplied with all entries on the λ-th row of $\mathbf{T}_{d,k}(\mathbf{u})$. Furthermore, the order of the terms $d_{\lambda\beta}c_{\alpha\lambda}$ yields that all entries on the λ-th row of $\mathbf{T}_{d,k}(\mathbf{z})$ are multiplied with every entry in the λ-th column of $\mathbf{T}_{d,k-1}(\mathbf{u})$. We recall Lemma 8.6, and conclude that for each α we obtain

$$\sum_{\lambda}\sum_{\beta}\Big(\Big[a_{\alpha\lambda}b_{\lambda\beta}\Big]-\Big[d_{\lambda\beta}c_{\alpha\lambda}\Big]\Big)=[z_0(u_0+\cdots+u_d)+\cdots+z_d(u_0+\cdots+u_d)] \\ -[(z_0+\cdots+z_d)u_0+\cdots+(z_0+\cdots+z_d)u_d],$$

where the RHS can be re-arranged to

$$(z_0+\cdots+z_d)(u_0+\cdots+u_d)-(u_0+\cdots+u_d)(z_0+\cdots+z_d),$$

which clearly is equal to zero since scalar multiplication is commutative. But then (8.44) is equal to zero, since changing the order of the sums commutes. Finally, since two matrices $\mathbf{A}$ and $\mathbf{B}$ commute if $\mathbf{AB}-\mathbf{BA}=0$ (see Corollary 8.1), it follows that

$$\mathbf{T}_{d,k-1}(\mathbf{z})\mathbf{T}_{d,k}(\mathbf{u})-\mathbf{T}_{d,k-1}(\mathbf{u})\mathbf{T}_{d,k}(\mathbf{z})=0.$$

□

Proof of Lemma 8.9

Proof. The barycentric coordinates of the point $\mathbf{p}+t\mathbf{v}$ are

$$(u_0+tv_0,\ldots,u_d+tv_d).$$

Using this with (8.19) yields

$$B_{\mathbf{i}}^n(\mathbf{p}+t\mathbf{v})=\binom{n}{\mathbf{i}}(u_0+tv_0)^{i_0}\cdots(u_d+tv_d)^{i_d}.$$

By differentiating with respect to t and evaluating at $t=0$ we obtain

$$D_{\mathbf{v}}B_{\mathbf{i}}^n(\mathbf{p})=\binom{n}{\mathbf{i}}\Big[i_0u_0^{i_0-1}v_0u_1^{i_1}\cdots u_d^{i_d}+u_0^{i_0}i_1u_1^{i_1-1}v_1\cdots u_d^{i_d}+\cdots+u_0^{i_0}\cdots u_{d-1}^{i_{d-1}}i_du_d^{i_d-1}v_d\Big],$$

where the RHS computes to (8.29). □

Proof of Lemma 8.10

Proof. By applying the chain rule we obtain

$$D_{\mathbf{v}}\mathbf{T}_{d,n}=v_0D_{u_0}\mathbf{T}_{d,n}+\cdots+v_dD_{u_d}\mathbf{T}_{d,n}.$$

The RHS corresponds to setting $\mathbf{T}_{d,n}(\mathbf{v})$, and the result follows. □

Proof of Lemma 8.12

Proof. The result follows by differentiating both sides of (8.23) with respect to $\mathbf{z}$ in the direction of $\mathbf{v}$. □

Proof of Theorem 8.2

Proof. We differentiate (8.33) to find the second derivative. Since $D_{\mathbf{v}_k}\left(\mathbf{T}_{d,n}(\mathbf{v}_1)\right) = 0$, we get

$$D_{\mathbf{v}_2} D_{\mathbf{v}_1} \mathbf{B}_d^n(\mathbf{u}) = n D_{\mathbf{v}_2} \mathbf{B}_d^{n-1}(\mathbf{u}) \mathbf{T}_{d,n}(\mathbf{v}_1),$$

and by applying (8.33) to $D_{\mathbf{v}_2}\mathbf{B}_d^{n-1}$, we obtain

$$D_{\mathbf{v}_2} D_{\mathbf{v}_1} \mathbf{B}_d^n(\mathbf{u}) = n(n-1)\mathbf{B}_d^{n-2}(\mathbf{u}) \mathbf{T}_{d,n-1}(\mathbf{v}_2) \mathbf{T}_{d,n}(\mathbf{v}_1).$$

Similarly, for the r-th derivative we find

$$D_{\mathbf{v}_r} \cdots D_{\mathbf{v}_1} \mathbf{B}_d^n(\mathbf{u}) = \frac{n!}{(n-r)!} \mathbf{B}_d^{n-r}(\mathbf{u}) \mathbf{T}_{d,n-r+1}(\mathbf{v}_r) \cdots \mathbf{T}_{d,n}(\mathbf{v}_1).$$

Since, in addition, $\mathbf{B}_d^{n-r}(\mathbf{u}) = \mathbf{T}_{d,1}(\mathbf{u}), \cdots, \mathbf{T}_{d,n-r}(\mathbf{u})$ holds, and since it does not matter which of the n matrices $\mathbf{T}_{d,k}$ is being differentiated (it only matters that we differentiate r of them), due to the symmetry property of (8.32), we conclude that (8.34) is true. □

Proof of Lemma 8.13

Proof. Since $l(t)$ is a line in the domain, it can be seen to correspond to a curve $s \circ l(t)$ on a given parametric object,

$$s \circ l(t) = s(\mathbf{p}) = \sum_{\mathbf{i}} \mathbf{c}_{\mathbf{i}} B_{\mathbf{i}}^n(\mathbf{u}).$$

Then the derivative of $s(\mathbf{u})$ at the point $\mathbf{p}$ in the direction of $\mathbf{v}$ is given by its derivative with respect to t at $t = 0+$:

$$\begin{aligned} D_{\mathbf{v}} s(\mathbf{p}) &= \frac{\mathrm{d}}{\mathrm{d}t}(s \circ l(t))\Big|_{t=0+} \\ &= v_0 \frac{\partial}{\partial u_0} s + \cdots + v_d \frac{\partial}{\partial u_d} s \\ &= n \sum_{\mathbf{j}} \mathbf{d}_{\mathbf{j}} B_{\mathbf{j}}^{n-1}(\mathbf{u}). \end{aligned}$$

□

8.6 Concluding remarks

We have proposed a matrix representation of multivariate Bernstein polynomials of arbitrary dimension by factorization. The factor matrix is defined recursively via a particular

decomposition of the matrix into sub-matrices.

The multiplication of the factor matrices from right to left was shown to correspond to the steps of the de Casteljau algorithm. A similar relation to the de Boor-Cox recursion formula, specialized for Bézier representation, can be found by computing the matrices from left to right, which yields a vector containing the Bernstein polynomials for the given degree in the specified number of variables. These polynomials can then be multiplied together with the associated vector of coefficients in order to evaluate the parametric object in question.

The matrix representation illustrates clearly one of the differences between the two methods; in the case of de Casteljau, every step yields intermediate points, whereas the intermediate values with the de Boor-Cox method are numbers. The total number of arithmetic operations when multiplying the matrices from right to left (de Casteljau) are higher than in the case of multiplication from left to right (de Boor-Cox), provided that the coefficients are points in $\mathbb{R}^d, d > 1$.

References

[1] S. Bernstein. Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités. *Communications de la Société Mathématique de Kharkow. 2-ée série*, 13(1):1–2, 1912.

[2] K. Ciesielski. *Set Theory for the Working Mathematician*. London Mathematical Society Student Texts (Book 39). Cambridge University Press, Cambridge, UK, 1997.

[3] P. Costantini and C. Manni. Geometric construction of generalized cubic splines. *Rendiconti di Matematica e delle sue Applicazioni*, 26:327–338, 2006.

[4] M. G. Cox. The numerical evaluation of B-splines. *J. Inst. Maths. Applics.*, 10:134–149, 1972.

[5] R. Dalmo. Matrix factorization of multivariate Bernstein polynomials. *International Journal of Pure and Applied Mathematics*, 103(4):749–780, 2015.

[6] R. Dalmo, J. Bratlie, and A. Lakså. Recursive construction of Bernstein factor matrices, 2014. Communication at the Eight International Conference on Curves and Surfaces, Paris, France.

[7] P. J. Davis. *Interpolation and approximation*. Dover, New York, 1975.

[8] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[9] C. de Boor. B-form basics. In G. E. Farin, editor, *Geometric Modeling: Algorithms and New Trends*, pages 131–148, Philadelphia, 1987. SIAM.

[10] C. de Boor. *A Practical Guide to Splines*, volume 27 of *Applied mathematical sciences*. Springer-Verlag, New York, revised edition, 2001.

[11] P. de Faget de Casteljau. *Shape mathematics and CAD*, volume 2 of *Mathematics and CAD*. Kogan Page, 120 Pentonville Road, London, N1 9JN, English language edition, 1986.

[12] G. Farin. *Curves and surfaces for CAGD: a practical guide*. Computer Science and Scientific Computing. Academic Press, San Diego, CA, USA, 4th edition, 1997.

[13] W. J. Gordon and R. Riesenfeld. B-spline curves and surfaces. In R. Barnhill and R. Riesenfeld, editors, *Computer Aided Geometric Design*, pages 95–126. Academic Press, New York, 1974.

[14] M.-J. Lai and L. L. Schumaker. *Spline Functions on Triangulations*, volume 110 of *Encyclopedia of Mathematics and Its Applications*. Cambridge University Press, Cambridge CB2 8RU, UK, 2007.

[15] A. Lakså. *Basic properties of Expo-Rational B-splines and practical use in Computer Aided Geometric Design*. PhD thesis, University of Oslo, 2007. (Dr.philos.).

[16] A. Lakså. A method for sparse-matrix computation of B-spline curves and surfaces. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Large-Scale Scientific Computing 2009*, volume 5910 of *Lecture Notes in Computer Science*, pages 796–804. Springer, 2010.

[17] G. G. Lorentz. *Bernstein Polynomials*. AMS Chelsea Publishing, Providence, Rhode Island, 2nd edition, 1986. Reprinted, 2012.

[18] T. Lyche and K. Mørken. Spline methods. Lecture notes at the University of Oslo (Draft), 2011.

[19] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline Techniques*. Mathematics and visualization. Springer-Verlag, Berlin Heidelberg, 2002.

[20] L. L. Schumaker. *Spline Functions*. Cambridge University Press, Cambridge CB2 8RU, UK, 3rd edition, 2007.

[21] B. L. van der Waerden. *Algebra II*. Springer-Verlag, Berlin Heidelberg New York, Fünfte Auflage der Modernen Algebra edition, 1967.

9 Image compression using an adjustable basis function

Rune Dalmo, Jostein Bratlie, and Peter Zanaty

This chapter is a reprint of [4]

Abstract — We investigate the performance of image compression using a custom transform, related to the discrete cosine transform, where the shape of the waveform basis function can be adjusted via setting a shape parameter. A strategy for generating quantization tables for various shapes of the basis function, including the cosine function, is proposed. Two signal fidelity measures, peak signal-to-noise ratio and mean structural similarity index, respectively, are computed for a few selected photos to benchmark the results.

9.1 Introduction

The discrete cosine transform (DCT) [1] is commonly used in a wide range of applications within image-, video- and audio compression, such as the JPEG image format, MPEG, MJPEG, DV, Daala, Theora type video compression and AAC, Vorbis, WMA and Mp3 audio compression.

Logistic expo-rational B-splines (LERBS) [19] is a recent sub-type of generalized expo-rational B-splines (GERBS) [5]. In this article we introduce a family of LERBS-based wave functions sharing some of the properties of the cosine function. In contrast to the cosine function, when using the proposed LERBS-based construction, the shape of the waveform basis function can be adjusted via setting a shape-parameter. We investigate the effect of the shape-parameter when applied to image compression and compare the results with a DCT-based JPEG-type encoding.

Peak signal-to-noise ratio (PSNR) [3], which is a logarithmic representation of mean squared error (MSE), is perhaps the most widely recognized fidelity metric within the image- and video processing community [18]. PSNR has only an approximate relationship with the image quality perceived by human observers, and is completely ignorant to spatial relationship between pixels and the interpretation of images by the human visual system (HVS) simply because it compares data byte-by-byte without considering what they actually represent [18]. The structural similarity (SSIM) [13, 15] is based on computing variance, covariance and mean within blocks of images to generate a distortion map. According to [18], one can distinguish between *data metrics*, which are appropriate for measuring the signal's fidelity without considering its content, and *picture metrics*, where the data is treated as the visual information it contains. For this reason we benchmark our results using both PSNR, which can be regarded as a data metric, and SSIM, considered as a picture metric.

In the remaining sections we explain in brief the DCT and the kind of LERBS we consider followed by a narrow description of the JPEG [12, 2] image format with emphasis on quantization. Next we explain the experiment this paper is about, present our results, and finally we give our concluding remarks.

9.2 Image transforms

In this section we will focus on the relevant components of JPEG. It is one of the most common image formats and is being used by many vendors to store pictures from digital cameras.

9.2.1 Discrete cosine transform

The DCT, introduced in [1], is a linear, invertible function $f : \mathbb{R}^N \to \mathbb{R}^N$ which expresses a finite sequence of data points in terms of a sum of cosine functions oscillating at different frequencies. In the two-dimensional case it is often represented as an invertible $N \times N$ square matrix. The DCT is related to the Fourier transform; it is similar to the discrete Fourier transform (DFT) when using real numbers only.

There are eight "standard" DCT variants of which four are common. The type-II DCT, which transforms N real numbers $x_0,\ldots,x_{N-1}$ into the N real numbers $X_0,\ldots,X_{N-1}$ according to

$$X_k = \sum_{n=0}^{N-1} x_n \cos\left[\tfrac{\pi}{N}\left(n+\tfrac{1}{2}\right)k\right], \quad \text{where} \quad k = 0,\ldots,N-1,$$

is the most common and is simply called "the DCT". Its inverse, the type-III DCT, transforms N real numbers $x_0,\ldots,x_{N-1}$ into the N real numbers $X_0,\ldots,X_{N-1}$ according to

$$X_k = \frac{1}{2}x_0 + \sum_{n=1}^{N-1} x_n \cos\left[\tfrac{\pi}{N}\left(k+\tfrac{1}{2}\right)n\right], \quad \text{where} \quad k = 0,\ldots,N-1,$$

and is simply called "the iDCT".

DCT-type encodings are typically used within application areas such as signal- and image processing due to the strong "energy compaction". This means that most of the signal information is concentrated in a few low-frequency components, thus, they are present in lossy compression of audio-, image- and video data. We note in addition that solving partial differential equations (PDEs) by spectral methods is another application area.

9.2.2 Logistic expo-rational basis

The default logistic expo-rational basis function (LERB) was introduced in [6, 19] and is defined as follows:

$$B(t) = \frac{1}{1+\exp\left(\frac{2}{(1+t)}-\frac{2}{(1-t)}\right)}. \tag{9.1}$$

In this work we introduce a "trigonometric" version of the LERB, derived from a version of (9.1) mapped to the interval $[0,1]$:

$$B(t) = \tanh\left[\alpha\left(\frac{1}{t}-\frac{1}{1-t}\right)\right]. \tag{9.2}$$

Formula (9.2) is designed to share some properties with the cosine function. The following properties for the first half-period (i.e., mapped to $[0,1]$) of $\cos(\pi t)$ are shared with $B(t)$ as defined in (9.2):

- $B(0) = 1$
- $B(1) = -1$
- $B(\frac{1}{2}) = 0$
- $B(t)$ is positive on the interval $[0,\frac{1}{2})$
- $B(t)$ is negative on the interval $(\frac{1}{2},1]$
- $B(t)$ is monotonically decreasing on the interval $[0,1]$

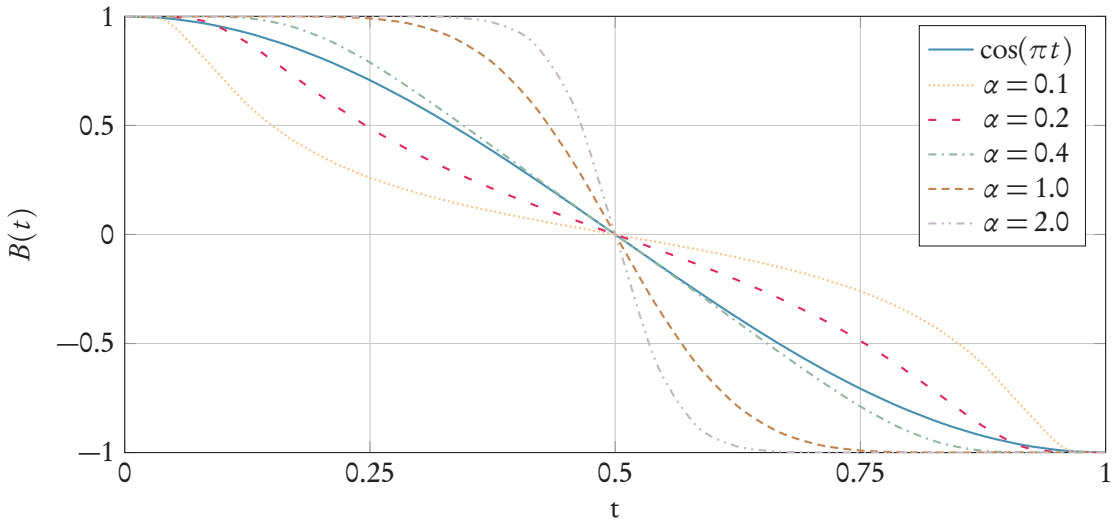

Figure 9.1: Plots of one half-period of (9.2), where the dashed and dotted lines illustrate a few values of the shape parameter α, are shown together with the cosine function mapped to the same interval (solid line).

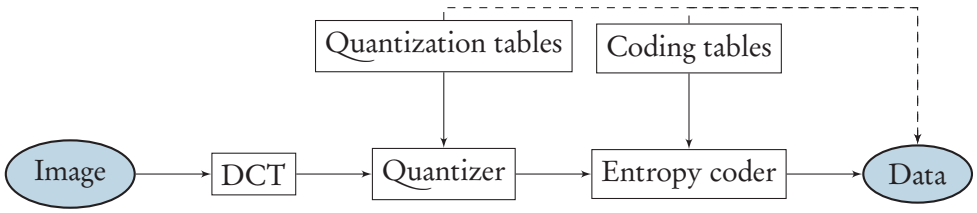

Figure 9.2: Block diagram of a JPEG encoder. An input image is transformed by a DCT step, then quantized and encoded in a compression step using fewer bits. Quantization- and coding tables are stored together with the encoded data.

For simplicity we show the properties for the first half-period here. Similar sharing of properties between $B(t)$ and the cosine function follows for the second half-period due to symmetry. A plot of the first half-period of $B(t)$, where the shape parameter α takes the values 0.1, 0.2, 0.4, 1.0 and 2.0, is shown in figure Figure 9.1 together with a plot of the function $\cos(\pi t)$ for reference.

9.2.3 Joint Photographic Experts Group

The JPEG image format [12, 2] provides lossy data compression where the amount of compression loss can be controlled through a quality parameter. A block diagram of the JPEG encoding step is shown in Figure 9.2. The pixels of an input image is transformed to the frequency domain by a 2D DCT working on blocks of size 8×8 pixels. Then the transformed signal is quantized, where the loss occurs, before it is packed in a lossless step and stored. Quantization- and coding tables are stored together with the encoded data. Thus, the decoding process is simply an inverse process which takes the encoded data as input and re-produces an approximation of the original image.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Table 9.1: Standard JPEG luminance quantization table with quality factor 50.

9.3 Quantization

JPEG quantization is performed block-wise by applying the following function to each of the DCT transformed blocks:

$$B_{j,k} = \text{round}\left(\frac{G_{j,k}}{Q_{j,k}}\right) \text{ for } j = 0,1,\ldots,N-1; k = 0,1,\ldots,N-1, \tag{9.3}$$

where $N \times N$ is the block size, G is the unquantized DCT coefficients, Q is the quantization matrix and B is the quantized DCT coefficients. It follows from (9.3) that larger values of Q produce greater compression since the coefficients become small, or even zeros, and require fewer bits to be stored. Furthermore, since the value for each pixel is divided by a table value as an integer operation, decimals are thrown away, which leads to image quality loss. Higher numbers correspond to lower quality image whereas lower numbers correspond to higher quality image. The "best" images have tables of all ones which means no compression.

Most software applications, including the widely used free software library "libjpeg" by the Independent JPEG Group (IJG) [8], use quantization tables as specified in [2, appendix K]. The construction of these tables are based on psycho-visual experiments where a group of people have responded to when the effects of various settings can be seen in images. We note that this is somewhat related to what is known as *Rose's criterion* [9]; that the SNR needs to be better than around 5 for the human eye to reliably identify an object. The IJG standard luminance quantization table is listed in Table 9.1. The JPEG quality setting is simply a scaling of the numbers in the quantization tables. We note that many digital cameras compute quantization tables on the fly when the image is being processed and that some camera vendors have patents on quantization table construction.

9.4 Set-up of the experiment

The JPEG quantization tables are applicable to coefficients of the DCT. We replace the cosine function with the LERB defined in (9.2), thus, it is necessary to generate custom tables to meet the considered shapes of the LERB. For this purpose we use the benchmark data set provided by the Uncompressed Colour Image Database (UCID) [10] (pronounced "use-it"). The current version (v2) contains 1338 images.

7.8758	4.8929	4.3133	4.0897	3.6322	3.3311	3.0317	2.8411
5.0400	4.3951	4.1513	4.0383	3.6805	3.3094	3.1354	2.9326
4.4159	4.2438	4.1520	3.7998	3.4620	3.0715	2.9731	2.8971
4.1739	4.1009	4.0513	3.7251	3.4449	3.1286	3.0658	2.9295
3.8889	3.8881	3.9116	3.7795	3.6793	3.1126	2.9421	2.8422
3.7094	3.8880	3.7726	3.5155	3.6822	3.1693	3.2196	2.6978
3.3184	3.7429	3.7172	3.6245	3.5663	3.6892	3.2651	2.6444
3.4164	3.6871	3.4591	3.5164	3.6689	3.5667	3.5849	3.2275

Table 9.2: 8-bit entropy table for the DC coefficients of $\alpha = 0.4$ computed from the 1338 UCID images.

We proceed as follows to benchmark the LERB-based transform by considering gray-scale images, which correspond to the luminance part of colour images. For a specific value of the parameter α in (9.2) we transform all the UCID images, analyze the results and generate histograms. Then we calculate entropy tables which we use to generate quantization matrices. The values of the quantization tables can be scaled to vary the compression- and error rates. Finally, we invoke the XZ utils [11] to perform lossless compression through a Lempel-Ziv-Markov chain algorithm (LZMA).

Figures 9.3 to 9.5 show histograms for the DC- , first "horizontal"- and first "vertical" coefficients, respectively.

An example of an 8-bit entropy table generated for the DC coefficients using the value $\alpha = 0.4$ is provided in Table 9.2. Such entropy tables are used to generate quantization matrices by applying the formula

$$Q[i][j] = b - E[i][j], \quad 0 \le i < N, \quad 0 \le j < N,$$

where $N \times N$ is the block size, b is the number of bits, E is the entropy table and Q is the resulting quantization matrix.

9.5 Results

Performance graphs for some test pictures selected from the Kodak Lossless True Color Image Suite [7], which can be seen in Figure 9.6, are shown in Figure 9.7. Two commonly used measures for image quality are plotted against a horizontal axis denoting compressed file size relative to original file size. The vertical axes of the left graphs show PSNR measured in dB:

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right),$$

where MAX_I is the maximum possible pixel value of the image and MSE is the mean squared error defined as

$$\text{MSE} = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i,j) - K(i,j)]^2,$$

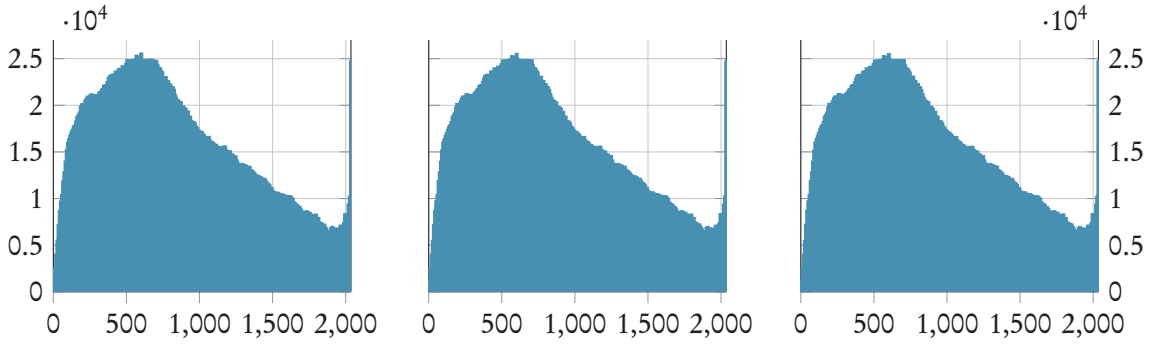

Figure 9.3: Histograms for the sum of DC coefficients for the 1338 UCID images using LERB with $\alpha = 0.2$ (left), $\alpha = 0.4$ (middle) and $\alpha = 1.0$ (right).

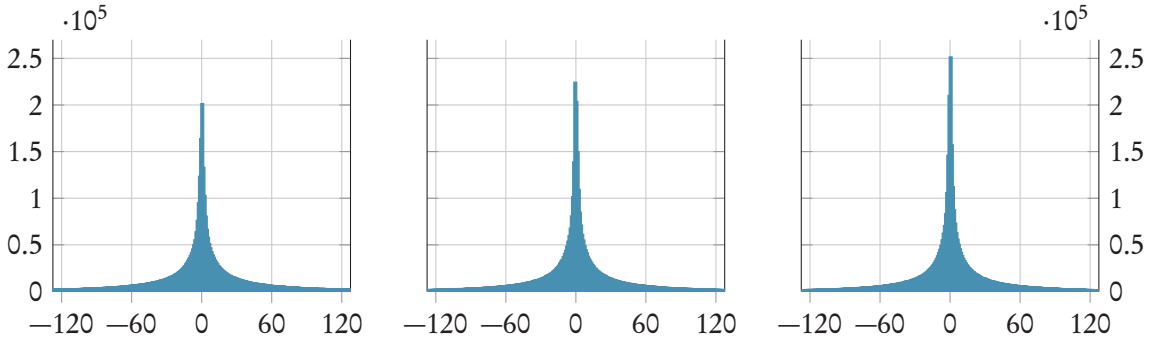

Figure 9.4: Histograms for the sum of 01 coefficients for the 1338 UCID images using LERB with $\alpha = 0.2$ (left), $\alpha = 0.4$ (middle) and $\alpha = 1.0$ (right).

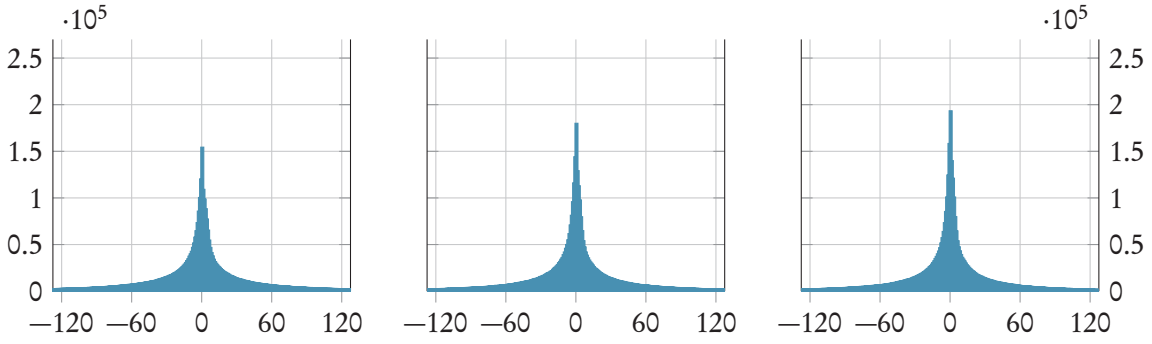

Figure 9.5: Histograms for the sum of 10 coefficients for the 1338 UCID images using LERB with $\alpha = 0.2$ (left), $\alpha = 0.4$ (middle) and $\alpha = 1.0$ (right).

Figure 9.6: Test image “kodim8” (top) by photographer Alfons Rudolph, “kodim21” (middle) by photographer Alan Fink, and “kodim23” (bottom) by photographer Steve Kelly. The top image shows several buildings with steep roofs in a Germanic town in Essligen, Germany, including the Seifenfabrikation Alfred. It is a picture with details such as edges, contrast and depth. The middle image shows a remote view of a lighthouse and dwelling on a rocky peninsula in Maine, USA. It is a picture with details combined with a soft background. The bottom image shows a close-up of two brightly colored makaw birds with out of focus greenery in the background taken at Maui, Hawaii, USA.

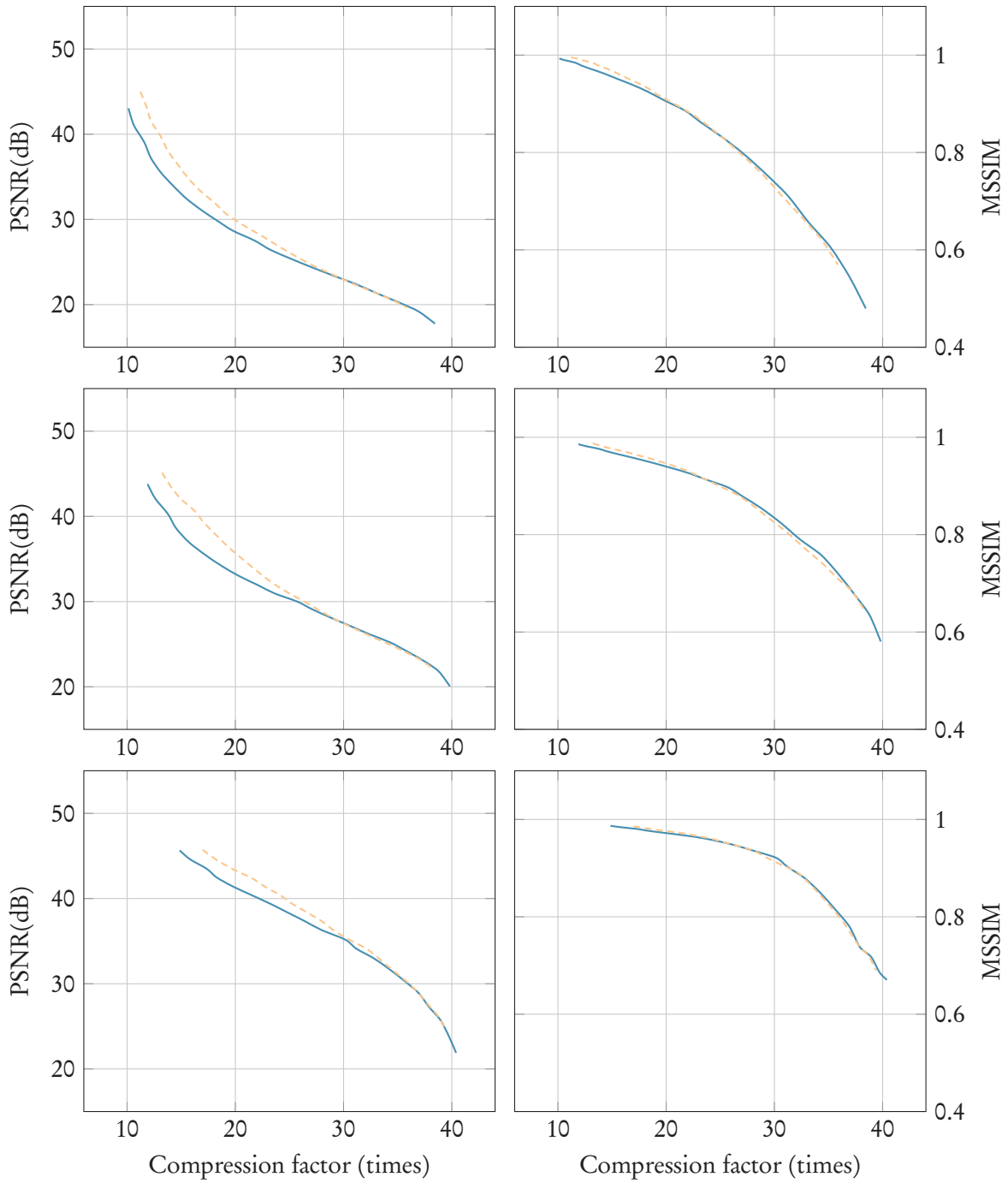

Figure 9.7: Performance graphs for the "kodim8", "kodim21" and "kodim23" test images in Figure 9.6 are shown in the top-, middle- and bottom rows, respectively. The solid line corresponds to a reference cosine transform with JPEG quantization whereas the LERB (with $\alpha = 0.4$) transform is shown as a dashed line.

where I is the uncompressed $m \times n$ monochrome image and K is its compressed approximation. In the right graphs the vertical axes show the mean structural similarity (MSSIM) [15] index, which is a method providing a single overall quality measure of the entire image, defined as

$$\mathrm{MSSIM}(\mathbf{X},\mathbf{Y}) = \tfrac{1}{M}\sum_{j=1}^{M}\mathrm{SSIM}(\mathbf{x}_j,\mathbf{y}_j),$$

where $\mathbf{X}$ and $\mathbf{Y}$ are the reference and distorted images, respectively, $\mathbf{x}_j$ and $\mathbf{y}_j$ are the image content at the jth local window, M is the number of local windows of the image. SSIM is the structural similarity index described in [15] based on the ideas introduced in [13]:

$$\mathrm{SSIM}(\mathbf{x},\mathbf{y}) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}, \tag{9.4}$$

which considers image degradation as perceived change in structural information between two windows x and y, of size 8×8 in this case, with μ_x and μ_y as the average of $\mathbf{x}$ and $\mathbf{y}$, respectively, σ_x^2 and σ_y^2 the variance of $\mathbf{x}$ and $\mathbf{y}$, σ_{xy} the covariance of $\mathbf{x}$ and $\mathbf{y}$, $C_1 = (k_1 L)^2$ and $C_2 = (k_2 L)^2$ are variables to stabilize in the case of weak denominator, L the dynamic range of the pixel-values, and $k_1 = 0.01$, $k_2 = 0.03$ (default values). The formula (9.4) is applied on luma only. The resultant SSIM index decimal value is on the range $[-1, 1]$, where the value 1 corresponds to two identical data sets. The local statistics μ_x, μ_y, σ_x, σ_y and σ_{xy} are computed within a local 8×8 square window which slides pixel-by-pixel over the entire image. At each step, the local statistics and SSIM index are calculated within the local window.

The first picture, "kodim8", has many details and is sharp nearly everywhere while the second one, "kodim21", is a mixture between details and a soft background with a color gradient. The third picture, "kodim23", is a portrait of two parrots where the birds are in focus against a blurry background.

9.6 Concluding remarks

Setting the shape parameter affects the coefficients of the discrete transform. We conclude by investigating the histograms in Figures 9.4 and 9.5 that more compact distributions can be achieved, thus, "better" quantization and higher compression rates is possible.

The performance graphs in Figure 9.7 indicate that our method provides higher compression rates against PSNR than the reference DCT using JPEG type quantization. However, as mentioned in section 9.1, optimizing for the best PSNR does not necessarily provide the most pleasing images. But the MSSIM graphs show that we are able to meet JPEG in terms of perceived change in structural information.

We note that in contrast to the DCT, where the inverse transform is orthogonal, the inverse LERB transform is not. For this reason we need to invert a small matrix. However, in many applications, such matrices can be pre-evaluated to improve the performance.

Our method is applicable within image processing, including user guided compression, or adaptive image processing methods where values of the parameter α can be computed to

satisfy some criteria.

The information stored in a picture does not necessarily need to be photos. In future work it could be interesting to experiment with other types of data since the adjustable waveform basis provides improved flexibility over the cosine function.

In this first approach we have for simplicity computed the local statistics for the SSIM index within a local 8×8 square window moving over the image. Another note for future work is to consider more recent or sophisticated methods, such as the circular-symmetric Gaussian weighting functions for local statistics computations as described in [15], or proceeding from the single scale implementation to multiple scales [17] or the wavelet domain [16]. More information related to SSIM for signal fidelity measures can be found in [14].

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 23(1):90–93, 1974.

[2] CCITT. Information technology – digital compression and coding of continuous-tone still images – requirements and guidelines. ISO/IEC JTC1/SC29 T.81, The International Telegraph and Telephone Consultative Committee, September 1992.

[3] L. C. Chan and P. Whiteman. Hardware-constrained hybrid coding of video imagery. *IEEE Transactions on Aerospace and Electronic Systems*, 19(1):71–84, 1983.

[4] R. Dalmo, J. Bratlie, and P. Zanaty. Image compression using an adjustable basis function. *Mathematics in Engineering, Science and Aerospace*, 6(1):25–34, 2015.

[5] L. T. Dechevsky, B. Bang, and A. Lakså. Generalized expo-rational B-splines. *International Journal of Pure and Applied Mathematics*, 57(6):833–872, 2009.

[6] L. T. Dechevsky and P. Zanaty. Smooth GERBS, orthogonal systems and energy minimization. In V. Pasheva and G. Venkov, editors, *39th International conference applications of mathematics in engineering and economics AMEE13*, volume 1570 of *AIP Conference Proceedings*, pages 135–162. AIP Publishing, 2013.

[7] R. Franzen. Kodak lossless true color image suite, 1993. PhotoCD PCD0992. Lossless, true color images released by the Eastman Kodak Company.

[8] IJG. Independent JPEG group, 1991. The Independent JPEG Group (IJG) is responsible for the reference implementation of the original JPEG standard.

[9] A. Rose. The sensitivity performance of the human eye on an absolute scale. *Journal of the Optical Society of America*, 38(2):196–208, 1948.

[10] G. Schaefer and M. Stich. UCID - An Uncompressed Colour Image Database. In M. M. Yeung, R. W. Lienhart, and C.-S. Li, editors, *Storage and Retrieval Methods and Applications for Multimedia 2004*, volume 5307 of *Proceedings of SPIE*, pages 472–480, San Jose, USA, 2004.

[11] Tukaani Project. XZ utils, 2005. Free general-purpose data compression software with high compression ratio. XZ utils are the successor to LZMA utils.

[12] G. K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, 1991.

[13] Z. Wang and A. C. Bovik. A universal image quality index. *IEEE Signal Processing Letters*, 9(3):81–84, 2002.

[14] Z. Wang and A. C. Bovik. Mean squared error: Love it or leave it? A new look at signal fidelity measures. *IEEE Signal Processing Magazine*, 26(1):98–117, 2009.

[15] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[16] Z. Wang and E. P. Simoncelli. Translation insensitive image similarity in complex wavelet domain. In *IEEE International Conference on Acoustic, Speech and Signal Processing*, volume 2, pages 573–576, 2005.

[17] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multi-scale structural similarity for image quality assessment. In *Conference Record of the 37th Asilomar Conference on Signals, Systems and Computers*, volume 2, pages 1398–1402, Asilomar, CA, USA, 2004.

[18] S. Winkler and P. Mohandas. The evolution of video quality measurement: From PSNR to hybrid metrics. *IEEE Transactions on Broadcasting*, 54(3):660–668, 2008.

[19] P. Zanaty. *Application of Generalized Expo-Rational B-splines in Computer Aided Design and Analysis*. PhD thesis, University of Oslo, 2014.

10 Performance of a Wavelet Shrinking Method

Rune Dalmo, Jostein Bratlie, and Børre Bang

This chapter is a reprint of [1]

Abstract — A concept for shrinking of wavelet coefficients has been presented and explored in a series of articles [5, 3, 4]. The theory and experiments so far suggest a strategy where the shrinking adapts to local smoothness properties of the original signal. From this strategy we employ partitioning of the global signal and local shrinking under smoothness constraints. Furthermore, we benchmark shrinking of local partitions' wavelet coefficients utilizing a selection of wavelet basis functions. Then we present and benchmark an adaptive partition-based shrinking strategy where the best performing shrinkage is applied to individual partitions, one at a time. Finally, we compare the local and global benchmark results.

10.1 Introduction

Shrinking of wavelet coefficients was first introduced by Donoho and Johnstone [6, 7] and has been an active area of research since then. Wavelet shrinkage are most notably applied to perform signal de-noising and within the quantification step of data compression.

Several strategies for thresholding- and non-thresholding type wavelet shrinkage of signals that belong to the general scale of Besov spaces were addressed in [5]. Later, in [3, 4], adaptive strategies based on composition of Lorentz-type thresholding and Besov-type non-threshold shrinkage, suitable for spatially in-homogeneous signals which exhibit both smooth regions and regions with (isolated) singularities, were introduced and explored.

Motivated by some typical characteristics of curves and surfaces in computer aided geometric design (CAGD), notably that they can have smooth parts, singularities and varying smoothness measures, and that they can be piecewise polynomials, we are interested in developing partition-based adaptive strategies for wavelet shrinkage.

In the following sections we describe the wavelet transform, coefficient shrinkage, compression, the test data we use and how we apply wavelet compression. Finally we give our concluding remarks where we comment on the results and suggest some topics for future work.

10.2 Curve fitting by wavelet shrinkage

The wavelet transform [2] is a technique to represent any arbitrary function f as wavelets, generated by scaling and translations from one single mother function ψ:

$$\psi_{a,b}(t) = |a|^{-1/2}\psi\left(\frac{t-b}{a}\right), \tag{10.1}$$

where a and b are constants defining scaling and translation, respectively. It is required that the mother function has mean zero:

$$\int \psi(t)dt = 0,$$

which typically implies at least one oscillation of $\psi(t)$ across the t-axis. Following from the dilations of a single function, compared with the mother function, low frequency wavelets ($a > 1$) are wider in the t-direction, whereas high frequency wavelets ($a < 1$) are narrower.

For application within signal analysis, the parameters a and b in (10.1) are usually restricted through discretization. A dilation step ($a_0 > 1$) and a translation step ($b_0 \neq 0$) are fixed, leading to the wavelets for $j, k \in \mathbb{Z}$:

$$\psi_{jk}(t) = a_0^{-j/2}\psi(a_0^{-j}t - kb_0). \tag{10.2}$$

The discrete wavelet transform (DWT), T, associated with the discrete wavelets in (10.2),

maps functions f to sequences indexed by $\mathbb{Z}^2$:

$$(Tf)_{jk} = < \psi_{jk}, f > = a_0^{-j/2} \int \psi(a_o^{-j} t - k b_0) f(t) dt.$$

Following the principle of decomposition, f can be reconstructed from its wavelet coefficients $\beta_{jk} = < \psi_{jk}, f >$:

$$f = \sum_{j,k} \beta_{jk} \psi_{jk}(t).$$

We address estimating f in the following non-parametric regression problem, as outlined in [3], by using orthonormal wavelets:

$$Y_i = f(X_i) + \epsilon_i, \quad i = 1, 2, \ldots, n, \tag{10.3}$$

where X_i are independent random variables uniformly distributed on $[0,1]^n$. We assume n independent identically distributed observations $Z_i, i = 1, 2, \ldots, n$, with unknown density $f(x)$, $x \in \mathbb{R}^n$, and errors $E\epsilon_1 = 0$, $E\epsilon_1^2 = \delta^2$.

As proposed in [3], let $B^s_{pq}(\mathbb{R}^n)$ be the Besov space with metric indices p, q and smoothness index s. For $f \in B^s_{pq}$, $0 < p \leq \infty$, $0 < q \leq \infty$, $n(\frac{1}{p} - 1)_+ < s < r$,

$$f(x) = \sum_{k \in \mathbb{Z}^d} \alpha_{0k} \phi^{[0]}_{0k}(x) + \sum_{j=0}^{\infty} \sum_{k \in \mathbb{Z}^d} \sum_{l=1}^{2^d - 1} \beta^{[l]}_{jk} \psi^{[l]}_{jk}(x), \quad a.e. \quad x \in \mathbb{R}^d \tag{10.4}$$

holds, where $\alpha_{0k} = \langle \phi^{[0]}_{0k}, f \rangle$ and $\beta_{jk}[l] = \langle \psi^{[l]}_{jk}, f \rangle$. The empirical wavelet estimator $\hat{f}(x)$ is defined via:

$$\hat{f}(x) = \sum_{k \in \mathbb{Z}^d} \hat{\alpha}_{0k} \phi^{[0]}_{0k}(x) + \sum_{j=0}^{\infty} \sum_{k \in \mathbb{Z}^d} \sum_{l=1}^{2^d - 1} \hat{\beta}^{[l]}_{jk} \psi^{[l]}_{jk}(x), \quad a.e. \quad x \in \mathbb{R}^d, \tag{10.5}$$

where in this case of non-parametric regression as defined in (10.3):

$$\hat{\alpha}_{j_0 k} = \frac{1}{n} \sum_{i=1}^{n} \phi^{[0]}_{j_0 k}(X_i) Y_i, \quad \hat{\beta}^{[l]}_{jk} = \frac{1}{n} \sum_{i=1}^{n} \psi^{[l]}_{jk}(X_i) Y_i.$$

We note here that the estimator $\hat{f}$ can be obtained simply by replacing the coefficients in (10.4) by their empirical version, however, this procedure is not as fast as the DWT.

10.2.1 Coefficient selection

Coefficient shrinkage can be obtained through the DWT in the following way:

1. Wavelet transform of the input data
2. Manipulation of the empirical wavelet coefficients
3. Inverse wavelet transform of the modified coefficients

The second step is where the shrinking occurs by adjusting the empirical wavelet coefficients towards zero. This manipulation usually depends on a classification of the coefficients where the goal is to obtain noise reduction without loosing too much information about the signal itself. One such classification is the binary case where a coefficient is either noisy and unimportant or relatively noise-free and important. Some sort of threshold on a measure of regularity, or criterion, is required in order to distinguish between such classes. For this purpose we will utilize the general Lorentz thresholding which was introduced in [5] and further explored in [3, 4]. Utilizing the *Sobolev-type* embedding within the Besov-space scale: $B^{\sigma}_{\eta\eta} \hookrightarrow B^{s}_{pp}$ if $\sigma - \frac{1}{\eta} = s - \frac{1}{p} =: \tau \in \mathbb{R}$ and $0 < p \leq \eta < \infty$, the method is in brief, for compactly supported f and ψ and assuming that $j_0 = 0$ for any sample size n, as follows.

1. Consider all $(j,k) : \mathrm{supp}(\psi_{jk}) \cap \mathrm{supp}(f) \neq \emptyset$. Denote the set of all such (j,k) by $I(f,\psi)$.

2. Consider the decreasing rearrangement $\{b_\nu \ \nu = 1,\ldots,M\}$ of the finite set $\{2^{j(\tau+1/2)}\}|\hat{\beta}_{jk}| : (j,k) \in I(f,\psi)$. The wavelet estimator f^*_v is defined by

$$f^*_v(x) = \sum_k \hat{\alpha}_{0k}\phi_{0k}(x) + \sum_{\nu=1}^{[v^{-\frac{\eta p}{\eta - p}}]} \hat{\beta}_{j_\nu k_\nu}\psi_{j_\nu k_\nu}(x), \quad x \in \mathbb{R}, \tag{10.6}$$

   where v is a smoothing factor.

3. Ensure uniqueness of f^*_v if the sequence $\{b_\nu\}$ is not strictly decreasing by removing redundant terms that are equal to the limit $b_{[v^{-\frac{\eta p}{\eta - p}}]}$.

The method outlined above is, according to [4], a far-going generalization of the concept of Lorentz-curve thresholding based on computability of the Peetre K-functional between Lebesgue spaces, in terms of non-increasing rearrangement of a measurable function, and isometricity of Besov spaces to vector-valued sequence spaces of Lebesques type. The significance of every $|\hat{\beta}_{jk}|$ is being assessed with respect to its level j so that the most significant features appear first and less significant details emerge only for sufficiently small ν. The criterion for significance of the coefficients is the regularity assumption expressed in the value of τ. Throughout this paper we shall fix τ to the boundary value $\tau = -\frac{1}{2}$ which corresponds to the critical regularity $\epsilon = -2(\eta - p)(\tau + \frac{1}{2}) = 0$.

10.2.2 Wavelet compression

Compressing a wavelet-transformed signal is essentially a two-step process:

1. Quantify the wavelet coefficients

2. Code-word assignment for the quantified coefficients

Errors are introduced when wavelet coefficients are being manipulated, eg. through shrinkage. We note that in order to achieve loss-less compression, that step has to be omitted.

The wavelet transformed and possibly quantified signal can be "packed" using error-free compression of the coefficients β, for example by applying Huffman code [8] or run-length encoding (RLE).

We investigate compression in this case with respect to quantification only by counting the relative number of coefficients which are discarded.

10.3 Wavelet shrinkage applied to a partitioned signal

In this preliminary experiment we benchmark the wavelet shrinking method, described in section 10.2.1, on a synthetic test function $F(x) : x \in [0,1]$ with four pre-determined partitions which are re-parameterized to meet the intervals $[0, \frac{1}{4})$, $[\frac{1}{4}, \frac{1}{2})$, $[\frac{1}{2}, \frac{3}{4})$ and $[\frac{3}{4}, 1]$ of $F(x)$, respectively. The partitions are defined by the following "local" functions[1]:

I The "λ-tear":

$$f_1(x) = x_+^{\lambda} \exp\left(-\tfrac{x^2}{1-x^2}\right), \quad \lambda > 0, \quad x \in [-0.2, 1],$$

where $x_+ = \max(x, 0)$ and we choose $\lambda = 0.25$.

II Double chirp:

$$f_2(x) = \sqrt{x} \exp\left(-\tfrac{x^2}{1-x^2}\right) \sin(64\pi x(1-x)), \quad x \in [0,1].$$

III Sinusoidal density, for $x \in [-3, 2]$:

$$f_3(x) = \begin{cases} \frac{1}{2}|\sin x| & \text{for } x \in [-2\pi/3, \pi/3], \\ 0 & \text{elsewhere.} \end{cases}$$

IV Triangle wave:

$$f_4(x) = \left|2\left(x - \left\lfloor x + \tfrac{1}{2} \right\rfloor\right)\right|, \quad x \in [0,1].$$

Each partition is sampled into 1024 uniformly distributed samples in our benchmark tests. 10 levels of DWT are applied such that only two scaling coefficients α are present together with 1022 wavelet coefficients β. We consider the following selection of wavelets:

1. Orthogonal Daubechies: $d_1, d_2, d_4, d_8, d_{12}, d_{16}, d_{20}$
2. Bi-orthogonal: $b_{1.1}, b_{1.3}, b_{1.5}, b_{2.2}, b_{2.4}, b_{2.6}, b_{2.8}, b_{3.1}, b_{3.3}, b_{3.5}, b_{3.7}, b_{3.9}, b_{4.4}, b_{5.5}, b_{6.8}$
3. Coiflets: c_1, c_2, c_3, c_4, c_5
4. Symlets: $s_2, s_3, s_4, s_6, s_8, s_{10}, s_{12}, s_{14}, s_{16}, s_{18}, s_{20}$
5. The discrete Meyer filter

and measure the signal-to-noise ratio (SNR) in dB, defined as

$$\text{SNR} = 10 \log_{10}\left(\frac{||F||_2^2}{||F - \hat{F}||_2^2}\right),$$

[1]Plots of each of the functions defined in items I to IV above are provided in section B.2. These plots are an extension of the published version of this article.

Table 10.1: The best performing wavelets according to selected criteria on the individual partitions and on the global signal. The first four criteria are maximum compression rate while maintaining a SNR of 120dB, 100dB, 80dB and 60dB, respectively. The last criterion is fixing the compression rate to 99% while obtaining maximum SNR. For each function, the compression rates are provided in percent, and the SNR is shown in dB.

Criterion	Partition								Global function	
	I		**II**		**III**		**IV**			
120dB	b.5.5	93%	s.18	87%	b.3.3	93%	d.2	96%	s.12	85%
100dB	b.3.3	95%	s.20	90%	b.3.1	94%	d.2	96.5%	s.10	88%
80dB	b.3.1	97%	s.18	93%	b.3.1	95%	b.3.1	97.5%	b.6.8	91%
60dB	b.3.1	98%	s.12	95%	b.2.2	97%	b.2.2	98.5%	b.6.8	94%
99%	b.3.1	45 dB	s.12	21 dB	b.2.8	28 dB	b.2.2	55 dB	s.16	22 dB

where F is the original signal, $\hat{F}$ denotes the compressed (shrunk) signal and $\|\cdot\|_2^2$ stands for the square of the L_2 norm. Compression is measured in terms of how many of the wavelet coefficients β which are set to zero relative to the size of the signal. The scaling coefficients α_{0k} in (10.5) are not affected by the shrinking procedure.

It was noted in [3] that wavelets with relatively large support tend to oscillate near singularities. This, combined with standard wavelet theory, leads us to the following conjectures:

Conjecture 10.1. *The best performing wavelet, while shrinking according to some specified criteria of measure, for different kind of signals, are not the same.*

Conjecture 10.2. *The wavelet shrinking performance for signals of varying smoothness or shape can increase if the signal is partitioned and shrunk using the best wavelet for each partition.*

Based on Conjectures 10.1 and 10.2 we propose the following method to compress and uncompress the global partitioned signal:

1. Select for each partition the wavelet which performs best according to some criterion, such as
   a. SNR threshold value, or
   b. performs best at high compression.
2. For each individual partition, perform DWT and apply coefficient shrinkage according to
   a. fixed SNR, or
   b. fixed compression ratio.
3. Reconstruct the individual partitions and use them to compose the global signal.

10.4 Results

Table 10.1 shows the "best" wavelets for the four individual partitions as well as the global composite function. The criteria used in the benchmark tests include measuring compression

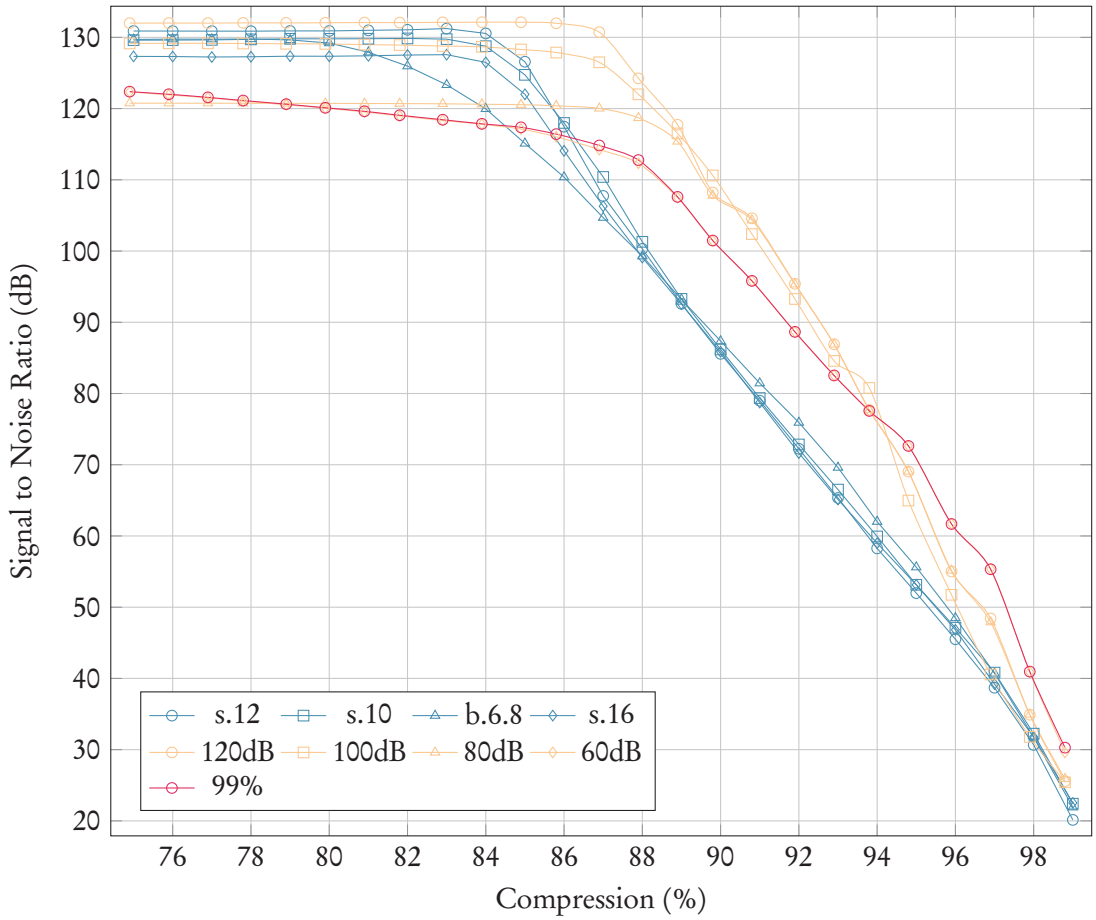

Figure 10.1: Performance charts for the best performing wavelets and the adaptive method, for selected criteria as provided in Table 10.1, on the global composite test function F, are shown. The Symlets s.12 and s.10 are those which provide highest compression ratio while maintaining a SNR of 120dB and 100dB, respectively, while the bi-orthogonal wavelet b.6.8 performs best at 80dB and 60dB. The Symlet s.16 generates yields the highest SNR at 99% compression. Charts for the adaptive method for fixed compression ratio based on selecting the best performing wavelets for each partition, according to the same criteria, are shown as well.

rates, where we select the wavelet providing the highest compression while maintaining SNR of 120dB, 100dB, 80dB and 60dB, respectively. In addition, we include the wavelets providing the best SNR at approximately 99% compression.

Figure 10.1 shows the performance on the global function of the best wavelets, as presented in Table 10.1, together with the performance of the adaptive strategy applied for fixed compression ratio. The horizontal axis displays the compression rate while the vertical axis shows the SNR measured in dB.

10.5 Concluding remarks

As expected in Conjecture 10.1, since the four partitions are of different nature, they have different "best performing" wavelets. The smooth Double chirp function benefits from symlet

type wavelets with relatively long support. Relatively short bi-orthogonal wavelets perform better on the Delta tear and the Sinusoidal density functions since they have one and two singularities, respectively, but are smooth otherwise. Very short filters seem to suit the Triangle wave function well, which is no surprise, because it is linear and has one singularity. When shrinking the global function without invoking the adaptive strategy, fairly long wavelet filters provide the best results.

We conclude that the experiments presented in this paper support both Conjectures 10.1 and 10.2. The performance of different wavelets is depending on the signal's smoothness and singularity properties. Thus, for signals with varying smoothness properties, the partitioning based adaptive approach can provide better coefficient shrinking performance than when using one wavelet type on the complete signal.

The adaptive method provides increased compression rates when compared to the non-adaptive approach. Furthermore, it provides a significant increase in SNR for fixed compression rates. When applied to the test function, even for 99% compression, the SNR is increased by approximately 5dB.

The method is suitable for signals with varying smoothness properties, such as parametric representations of curves and surfaces.

Wavelet transform and shrinking of individual partitions are candidates for parallel computation since the partitions do not depend on each other.

As an anecdote, we note what is known as *Rose's criterion* [9]; that the SNR needs to be better than around 5 for the human eye to reliably identify an object.

10.5.1 Future work

First of all we suggest using the findings presented in this article to construct an adaptive method. It would then be interesting to apply strategies for partitioning of the global signal. Local feature detection, such as identifying singularities or extreme values of functions, could be considered for this purpose.

For partitions containing isolated singularities, one possible improvement can be to invoke Besov type non-thresholding shrinkage as outlined in [3]. According to [5], this tends to produce better fitting of signals near singularities under the penalty of undersmoothing smooth regions. However, such undersmoothing could be compensated for in the partitioning process by selecting relatively small partitions near singularities.

The individual parts of the global signal could be classified by some smoothness measure, which could be seen in connection with the value of the parameter τ in (10.6). Lorentz type thresholding is performed here using a fixed value of $\tau = -\frac{1}{2}$ which is appropriate in cases when the signal's smoothness properties are unknown. We note that the method could be improved by considering the qualitative differences between strategies for "less regular" functions ($\tau < -\frac{1}{2}$) and "more regular" functions ($\tau > \frac{1}{2}$). Furthermore, this classification could also be considered to determine the best type of wavelet for each partition, possibly based on correlation.

References

[1] R. Dalmo, J. Bratlie, and B. Bang. Performance of a wavelet shrinking method. In I. Dimov, S. Fidanova, and I. Lirkov, editors, *Numerical Methods and Applications 2014*, volume 8962 of *Lecture Notes in Computer Science*, pages 262–270. Springer, 2015.

[2] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Communications on Pure and Applied Mathematics*, XLI:909–966, 1988.

[3] L. T. Dechevsky, N. Grip, and J. Gundersen. A new generation of wavelet shrinkage: Adaptive strategies based on composition of Lorentz-type thresholding and Besov-type non-threshold shrinkage. In F. Truchetet and O. Laligant, editors, *Wavelet Applications in Industrial Processing V*, volume 6763 of *Proceedings of SPIE*, pages 1–14, Boston, MA, USA, 2007. Paper 676304.

[4] L. T. Dechevsky, J. Gundersen, and N. Grip. Wavelet compression, data fitting and approximation based on adaptive composition of Lorentz-type thresholding and Besov-type non-threshold shrinkage. In I. Lirkov, S. Margenov, and J. Waśniewski, editors, *Large-Scale Scientific Computing 2009*, volume 5910 of *Lecture Notes in Computer Science*, pages 738–746. Springer, 2010.

[5] L. T. Dechevsky, J. O. Ramsay, and S. I. Penev. Penalized wavelet estimation with besov regularity constraints. *Mathematica Balkanica (N. S.)*, 13(3-4):257–376, 1999.

[6] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 8(3):425–455, 1994.

[7] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *Journal of the Royal Statistical Society Series*, B57(2):301–369, 1995.

[8] D. A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the I.R.E.*, volume 40, pages 1098–1110, 1952.

[9] A. Rose. The sensitivity performance of the human eye on an absolute scale. *Journal of the Optical Society of America*, 38(2):196–208, 1948.

Appendices

A Detailed Proofs

In this chapter we provide calculations which were omitted from the published versions of the articles due to page limitation policies of the publication channels.

A.1 Proof of Theorem 4.1

Formula (4.10) is obtained by solving the equation where the left hand side is given by (4.8) and the right hand side is (4.9) for $\ell_z(t)$. There are two parts:

1. The case when $\hat{t}_k < t \leq z$ yields

$$\hat{\ell}_k(t) + B \circ \hat{\omega}_k(t)\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) = \hat{\ell}_k(t) + B \circ \omega_k(t)\Big(\ell_z(t) - \hat{\ell}_k(t)\Big),$$

which can be simplifed to

$$B \circ \hat{\omega}_k(t)\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) = B \circ \omega_k(t)\Big(\ell_z(t) - \hat{\ell}_k(t)\Big).$$

It follows that

$$\frac{B \circ \hat{\omega}_k(t)}{B \circ \omega_k(t)}\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) = \ell_z(t) - \hat{\ell}_k(t)$$

which can be re-arranged to the corresponding part of (4.10):

$$\ell_z(t) = \hat{\ell}_k(t) + \frac{B \circ \hat{\omega}_k(t)}{B \circ \omega_k(t)}\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big).$$

2. For $z < t < \hat{t}_{k+1}$, we have

$$\hat{\ell}_k(t) + B \circ \hat{\omega}_k(t)\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) = \ell_z(t) + B \circ \omega_{k+1}(t)\Big(\hat{\ell}_{k+1}(t) - \ell_z(t)\Big).$$

We re-arrange so that the terms containing $\ell_z(t)$ appear on one side:

$$\ell_z(t) - B \circ \omega_{k+1}(t)\ell_z(t) = \hat{\ell}_k(t) + B \circ \hat{\omega}_k(t)\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) - B \circ \omega_{k+1}(t)\hat{\ell}_{k+1}(t),$$

The left hand side can be re-arranged:

$$\ell_z(t)\big(1 - B \circ \omega_{k+1}(t)\big) = \hat{\ell}_k(t) + B \circ \hat{\omega}_k(t)\Big(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\Big) - B \circ \omega_{k+1}(t)\hat{\ell}_{k+1}(t),$$

and it follows that

$$\ell_z(t) = \frac{\hat{\ell}_k(t) + B \circ \hat{\omega}_k(t)\left(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\right) - B \circ \omega_{k+1}(t)\hat{\ell}_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)}.$$

Re-arranging the terms yields

$$\ell_z(t) = \hat{\ell}_k(t)\frac{1 - B \circ \hat{\omega}_k(t)}{1 - B \circ \omega_{k+1}(t)} + \hat{\ell}_{k+1}(t)\frac{B \circ \hat{\omega}_k(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)}. \tag{A.1}$$

By writing

$$\hat{\ell}_k(t)\frac{1 - B \circ \hat{\omega}_k(t) + B \circ \omega_{k+1}(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)}$$

for the first term on the right hand side of (A.1) and re-arranging we obtain

$$\begin{aligned}\ell_z(t) = \hat{\ell}_k(t) + \hat{\ell}_k(t)\frac{-B \circ \hat{\omega}_k(t) + B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)} \\ + \hat{\ell}_{k+1}(t)\frac{B \circ \hat{\omega}_k(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)}\end{aligned}$$

which can be written on the form of the corresponding part of (4.10):

$$\ell_z(t) = \hat{\ell}_k(t) + \frac{B \circ \hat{\omega}_k(t) - B \circ \omega_{k+1}(t)}{1 - B \circ \omega_{k+1}(t)}\left(\hat{\ell}_{k+1}(t) - \hat{\ell}_k(t)\right).$$

B Figures

Here we include some figures which were taken out of the previously published articles due to space limitations.

B.1 The curves in section 4.3.3

Figure B.1 shows a plot of the ERBS curve (before knot insertion) and its local curves.The new knot $z = 3.5$ is indicated on the curve with a green mark in Figure B.2. Furthermore, Figure B.2 illustrates the new support of the existing local curves relative to the refined knot vector. This corresponds to the cases when $i = k$ and $i = k + 2$ in (4.6). Figures B.3 to B.6 are related to Table 4.1 as follows:

- Scenario **I**: corresponding curves are shown in Figure B.3.
- Scenario **II**: corresponding curves are shown in Figures B.4 and B.5.
- Scenario **II+IV**: corresponding curves are shown in Figure B.6.

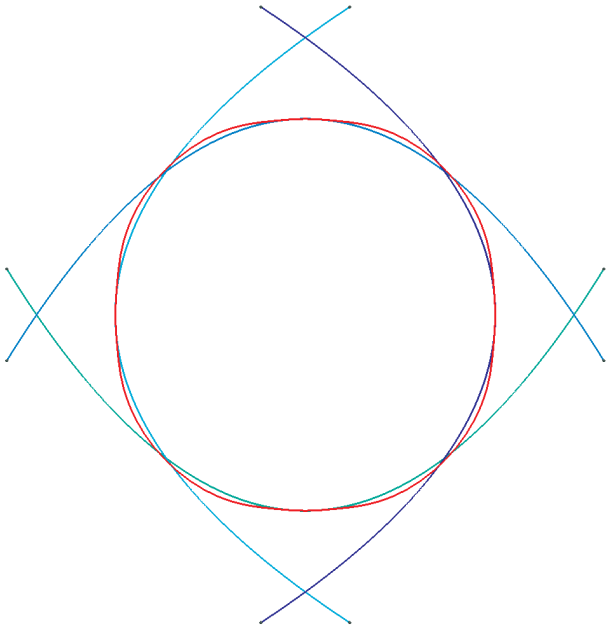

Figure B.1: An ERBS approximation (red) of a circle using four quadratic Bézier local curves.

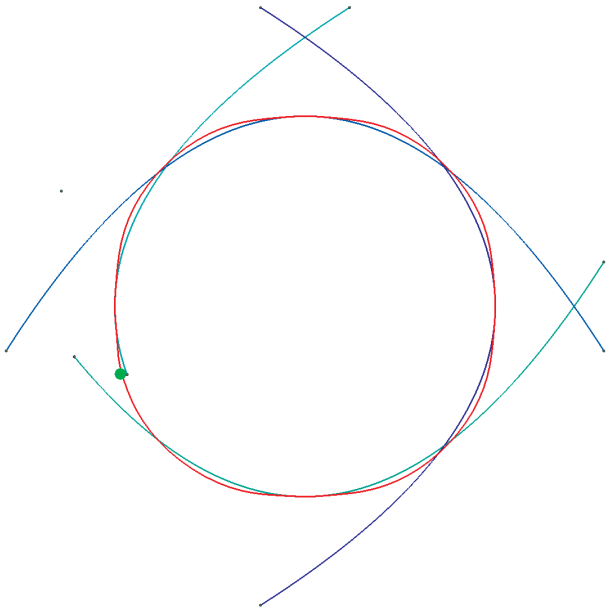

Figure B.2: A new knot (green mark) inserted into an existing knot vector associated with an ERBS approximation (red) of a circle. The new support intervals for the affected existing local curves are illustrated. They can be seen by comparing them with the ones in Figure B.1.

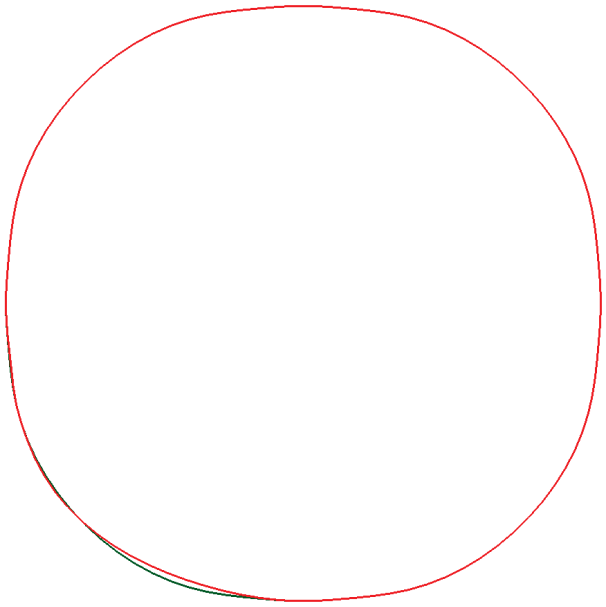

Figure B.3: ERBS approximation (red) of an existing ERBS curve (green) by knot insertion using a sub-curve of the original ERBS curve as local curve associated with the new knot.

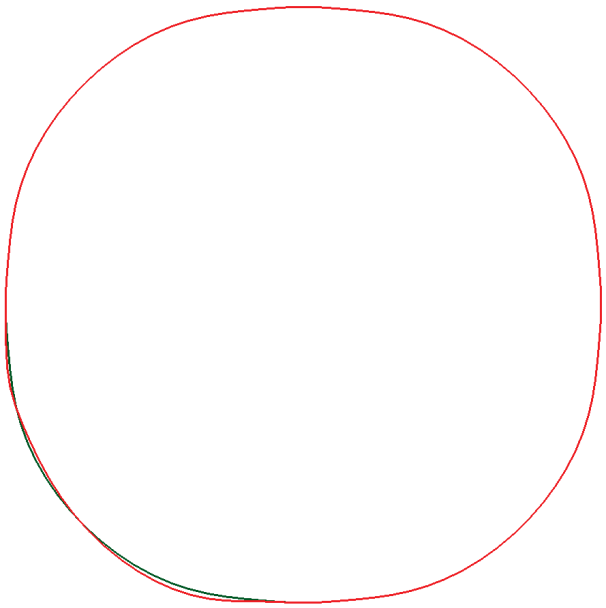

Figure B.4: ERBS approximation (red) of an existing ERBS curve (green) by knot insertion using a Bézier curve obtained by Hermite interpolation of the original ERBS curve as local curve associated with the new knot.

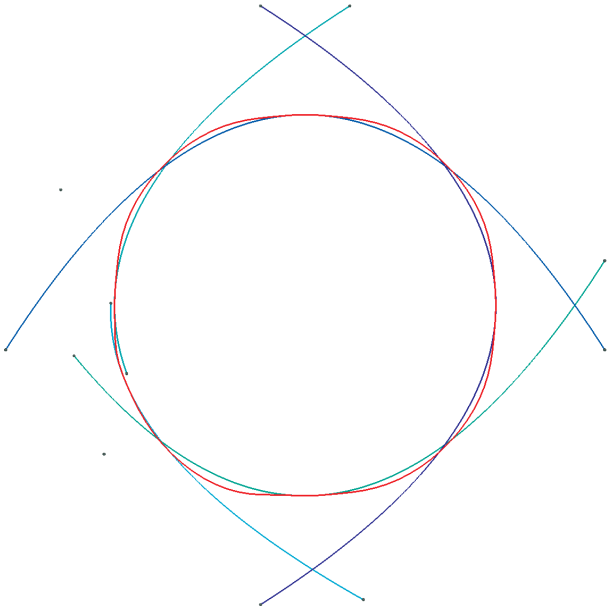

Figure B.5: ERBS approximation (red) and its Bézier local curves, where the new local curve is obtained by Hermite interpolation of the original ERBS curve.

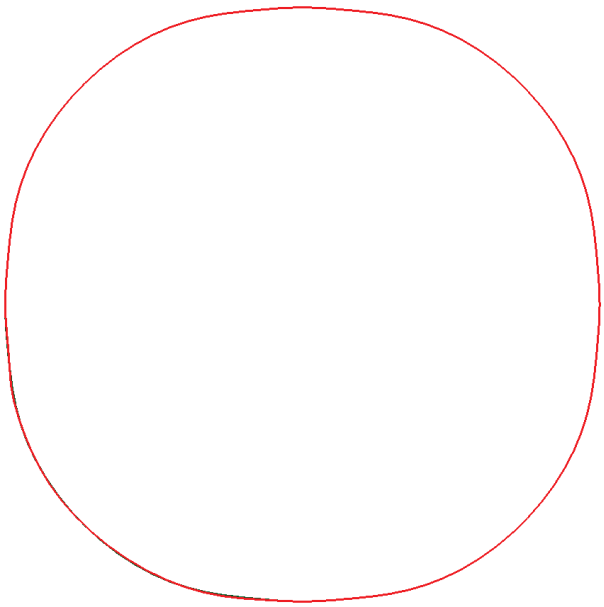

Figure B.6: ERBS approximation (red) of an existing ERBS curve (green) by knot insertion using a Bézier curve with adjusted coefficients as local curve associated with the new knot. The difference is barely visible on the plot since the red curve efficiently covers the green curve as a consequence of the relatively small approximation error.

B.2 The functions in section 10.3

The functions **I-IV** were selected because they possess a diverse combination of smoothness properties and singularities. They are shown in Figures B.7 to B.10 as follows:

I The "λ-tear":

$$f_1(x) = x_+^\lambda \exp\left(-\tfrac{x^2}{1-x^2}\right), \quad x \in [-0.2, 1],$$

where $x_+ = \max(x, 0)$, and $\lambda = 0.25$, is shown in Figure B.7.

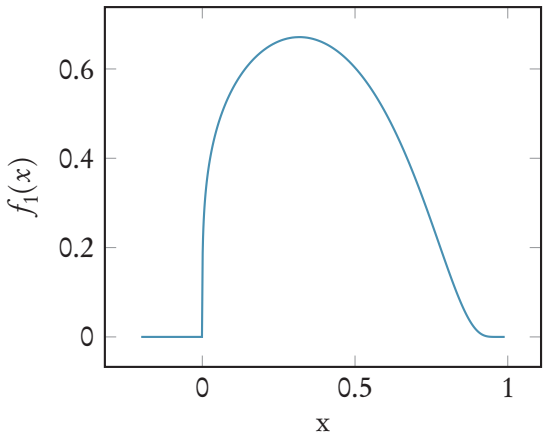

Figure B.7: The "λ-tear" is a continuous function with smooth regions and one singluarity.

II Double chirp:

$$f_2(x) = \sqrt{x} \exp\left(-\tfrac{x^2}{1-x^2}\right) \sin(64\pi x(1-x)), \quad x \in [0, 1],$$

is shown in Figure B.8.

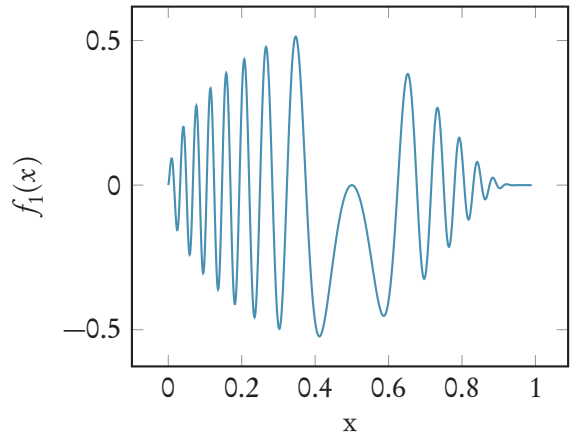

Figure B.8: The double chirp is a smooth function without singularities.

III Sinusoidal density, for $x \in [-3, 2]$:

$$f_3(x) = \begin{cases} \frac{1}{2}|\sin x| & \text{for } x \in [-2\pi/3, \pi/3], \\ 0 & \text{elsewhere.} \end{cases},$$

is shown in Figure B.9.

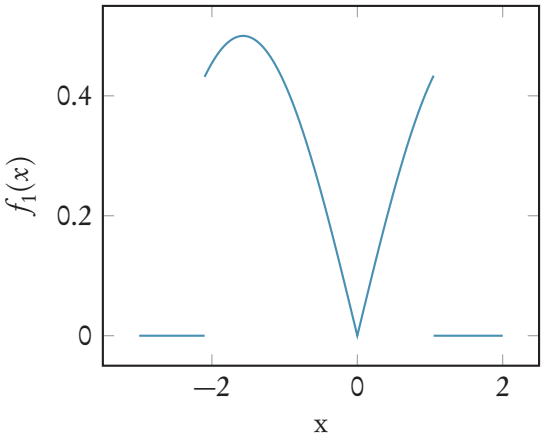

Figure B.9: Sinusoidal density. A function with smooth parts, isolated singularities and jumps.

IV Triangle wave:

$$f_4(x) = \left|2\left(x - \left\lfloor x + \tfrac{1}{2}\right\rfloor\right)\right|, \quad x \in [0, 1],$$

is shown in Figure B.10.

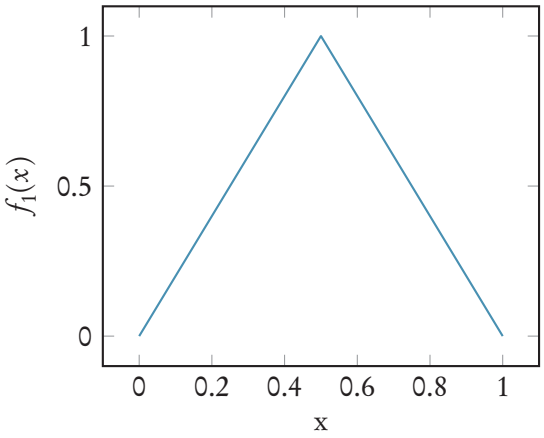

Figure B.10: Triangle wave. A continuous piecewise linear function with one singularity.

List of Acronyms and Abbreviations

API	Application programming interface
B-function	Blending function
CAD	Computer aided design
CAGD	Computer aided geometric design
CAM	Computer aided manufacturing
CGI	Computer-generated imagery
CPU	Central processing unit
CUDA	Compute unified device architecture
DCT	Discrete cosine transform
DFT	Discrete Fourier transform
DirectX	Microsoft® DirectX®
DTM	Digital terrain model
DWT	Discrete wavelet transform
ERB	Expo-rational basis function
ERBS	Expo-rational B-spline
GERBS	Generalized expo-rational B-spline
GIS	Geographic information system
GPGPU	General-purpose GPU
GPU	Graphics processing unit
HVS	Human visual system
IGES	Initial graphics exchange specification
IJG	Independent JPEG Group
JPEG	Joint photographic experts group
LERB	Logistic expo-rational basis function
LERBS	Logistic expo-rational B-spline

LHS Left-hand side
LOD Level of detail
LZMA Lempel-Ziv-Markov chain algorithm

MMO Massive multiplayer online
MRA Multiresolution analysis
MSE Mean squared error
MSSIM Mean structural similarity

NUC Narvik University College
NURBS Nonuniform rational B-spline

OpenCL Open computing language
OpenGL Open graphics library

PDE Partial differential equation
PSNR Peak signal-to-noise ratio

R&D Research and development
RHS Right-hand side
RLE Run-length encoding

SIMD Single instruction multiple data
SNR Signal-to-noise ratio
SSIM Structural similarity
STEP Standard for the exchange of product model data

TIN Triangulated irregular network

UCID Uncompressed Colour Image Database

List of Figures

List of Tables