University of Zagreb
Department of Mathematics

Tina Bosner

Knot insertion algorithms for Chebyshev splines

Doctoral Thesis

Ph.D. Supervisor: dr. sc. Mladen Rogina

Zagreb, 2006.

Contents

Chapter 1

Introduction

Almost everything is said, until now, about algorithms for calculating with polynomial splines, but sometimes we need some properties of a curve that polynomial splines can not satisfy. For example, polynomials are inferior to approximate exponential growth of a curve, or sometimes polynomial splines have some unnecessary inflections that can not be removed. Also, for solving some specific differential [16, 17, 3] or integral [9] equations, certain minimization problems [12, 7], or finding kernel of differential operators [39, 29], it is much better to use curves picewisely spanned by other functions than polynomials. These, and some other problems can be resolved by using splines described by Chebyshev theory [39, 38]. Although, theoretically, we know a lot about them, there are relatively few algorithms which are stable when applied on a computer. More is said about splines associated with extended Chebyshev systems and extended complete Chebyshev (ECC)–systems [39, 21, 24, 23], but we have more freedom with those associated with canonical complete Chebyshev (CCC)–system. The main idea of this thesis is generalization of some algorithms for polynomial splines, based on knot insertion, to CCC–systems.

In Chapter 2 we introduce Chebyshev splines: starting with the definition of CCC–systems, then through determinants, Chebyshev divided differences and Green's function we arrive to B-splines. We gathered here only those parts of the whole theory, that we need, and for some theorems we give slightly updated proofs. We are interested in the properties of B-splines, and the most important property for us is the derivative formula (2.27) for B-splines, which connects the B-splines of certain order with the B-splines of one order less. This derivative formula is the basis for the development of knot insertion, and further algorithms described in Chapter 3. First, we examine the single knot insertion matrices, then Oslo type algorithms, giving the recurrence for elements of knot insertion matrices, with retrospect on special cases for order 3 and 4. We end this chapter by generalizing the well known de Boor algorithm, and again emphasize specially rearranged variant for order 3 and 4.

After this theoretical part, to show the practical value of the algorithms from Chapter 3, we apply them on four kinds of splines. As the first and most simple example, in Chapter 4 we develop algorithms for weighted splines [7, 2, 33]. First, we treat, as a special case, weighted splines of order 4, and then the splines of

order higher than 4. These are the only splines of higher order we calculate with. In particular, we give our consideration to calculation of the integrals of B-splines associated with reduced systems, needed for knot insertion matrices.

The next example, q-splines [12, 3, 1], are also Chebyshev–polynomial splines and we give two simple ways of calculating with them in Chapter 5.

Maybe the most interesting case in this thesis are the tension splines [18, 11, 13, 37, 19, 16, 17, 9], due to their good properties and wide application. In Chapter 6 we elaborate three kinds of tension splines: C^1, C^2, which are mostly used, and Chebyshev tension splines. The algorithms for calculating with these splines, as well as for calculating the first and the second derivatives of the C^1 and C^2 tension splines, are described in details, with special emphasize on numerical stability.

Finally, cycloidal splines [5, 14, 15, 28] from Chapter 7 are our last example. Again, we discuss three cases: equidistant, equidistant Bezier and nonequidistant Bezier cycloidal splines, and explain the reasons for the way we defined them.

In the last Chapter 8, to illustrate the practical computer use of algorithms based on Chapter 6, we list program codes involved in calculating with C^1 and C^2 tension splines.

I sincerely thank my supervisor prof. Mladen Rogina for the support, lot of patience, care, and very motivating guidance in making this thesis.

Chapter 2

Chebyshev splines

In this chapter we will give the preliminaries, *i.e.* the basic facts of Chebyshev theory [39, 38, 32] needed for our algorithms.

2.1 Canonical complete Chebyshev systems

Let $\{u_i\}_1^m$ be a set of functions sufficiently differentiable on the set $I \subseteq \mathbb{R}$, and let $t_1, t_2, \ldots, t_m$ be points in I such that

$$t_1 \leqslant t_2 \leqslant \cdots \leqslant t_m = \underbrace{x_1, \ldots, x_1}_{l_1}, \ldots, \underbrace{x_d, \ldots, x_d}_{l_d},$$

where each x_i is repeated exactly l_i times with $\sum_{i=1}^d l_i = m$. Then we define the matrix associated with $\{u_i\}_1^m$ and $\{t_i\}_1^m$ by

$$\bar{M}\begin{pmatrix} t_1, \ldots, t_m \\ u_1, \ldots, u_m \end{pmatrix} := \begin{bmatrix} u_1(x_1) & u_2(x_1) & \cdots & u_m(x_1) \\ Du_1(x_1) & Du_2(x_1) & \cdots & Du_m(x_1) \\ \cdots & & & \\ D^{l_1-1}u_1(x_1) & D^{l_1-1}u_2(x_1) & \cdots & D^{l_1-1}u_m(x_1) \\ \cdots & & & \\ u_1(x_d) & u_2(x_d) & \cdots & u_m(x_d) \\ Du_1(x_d) & Du_2(x_d) & \cdots & Du_m(x_d) \\ \cdots & & & \\ D^{l_d-1}u_1(x_d) & D^{l_d-1}u_2(x_d) & \cdots & D^{l_d-1}u_m(x_d) \end{bmatrix}.$$

We denote its determinant by

$$\bar{D}\begin{pmatrix} t_1, \ldots, t_m \\ u_1, \ldots, u_m \end{pmatrix} := \det \bar{M}\begin{pmatrix} t_1, \ldots, t_m \\ u_1, \ldots, u_m \end{pmatrix}.$$

Definition 2.1 *Let I be a subinterval of $\mathbb{R}$, suppose $U_k := \{u_i\}_1^k$ is a set of functions in $C^{k-1}(I)$, and let $U_m := \{u_i\}_1^m$ for $m = 1, \ldots, k-1$. We call U_k an Extended Complete Chebyshev (ECC)–system provided that*

$$\bar{D}_{U_m}(t_1, \ldots, t_m) := \bar{D}\begin{pmatrix} t_1, \ldots, t_m \\ u_1, \ldots, u_m \end{pmatrix} > 0 \quad \text{for all } t_1 \leqslant t_2 \leqslant \cdots \leqslant t_m \text{ in } I$$

and all $m = 1, 2, \ldots, k$.

Definition 2.2 *Let* u_1 *be a bounded positive function on the interval* $[a, b]$*, and let there be given a* measure vector $d\boldsymbol{\sigma} := (d\sigma_2, \ldots d\sigma_k)^{\mathrm{T}}$*, where* $\sigma_2, \ldots, \sigma_k$ *are bounded, right continuous and monotone–increasing functions on* $[a, b]$*. Let*

$$\begin{aligned} u_2(x) &= u_1(x) \int_a^x d\sigma_2(\tau_2), \\ &\vdots \\ u_k(x) &= u_1(x) \int_a^x d\sigma_2(\tau_2) \int_a^{\tau_2} \ldots \int_a^{\tau_{k-1}} d\sigma_k(\tau_k), \end{aligned} \tag{2.1}$$

for $x \in [a, b]$*. We call* $U_k = \{u_i\}_1^k$ *a* Canonical Complete Chebyshev (CCC)–system.

We are particularly interested in CCC–systems with $u_1 \equiv 1$ (see Theorem 2.15 and Remark 2.5). We can also define the i^{th} restricted CCC–systems $U_k^{[i]} := \{u_1, \ldots, u_i\}$ associated with the restricted measure vector $d\boldsymbol{\sigma}^{[i]} := (d\sigma_2, \ldots, d\sigma_i)^{\mathrm{T}}$ for $i = 2, \ldots, k-1$.

If all of the measures $d\sigma_i$ possess smooth densities:

$$\sigma_i(t) = \int_a^t w_i(s) ds, \qquad i = 2, \ldots, k$$

with $w_2, \ldots, w_k$ positive functions and $w_i \in C^{k-i+1}([a, b])$, then $\{1, u_2, \ldots u_k\}$ is an ECC–system.

Lemma 2.1 *Suppose* $\{u_i\}_1^k$ *is a CCC–system on an interval* $[a, b]$*. Then each of these functions can be extended to form a CCC–system on any interval* $[c, d]$ *containing* $[a, b]$*.*

Proof: We only need to extend u_1 to remain positive and bounded on $[c, d]$, and then to extend each of the σ_i, $i = 2, \ldots, k$, to be bounded, right continuous, and monotone increasing. ■

Theorem 2.1 *If* $\{u_i\}_1^k$ *is a CCC–system on* $[a, b]$*, then there exists a function* $\bar{u}_{k+1}$ *such that* $\bar{U}_k^{[k+1]} := \{u_i\}_1^k \cup \{\bar{u}_{k+1}\}$ *is a CCC–system of* $k+1$ *functions on* $[a, b]$ *associated with* extended measure vector $d\bar{\boldsymbol{\sigma}}^{[k+1]} := (d\sigma_2, \ldots d\sigma_k, d\bar{\sigma}_{k+1})^{\mathrm{T}}$*.*

Proof: We simply choose, for example, $d\bar{\sigma}_{k+1} = d\lambda$ (or any other measure that satisfies properties from Definition 2.2), where $d\lambda$ is the Lebesgue measure, and define

$$\bar{u}_{k+1} := u_1(x) \int_a^x d\sigma_2(\tau_2) \int_a^{\tau_2} \ldots \int_a^{\tau_k} d\bar{\sigma}_{k+1}(\tau_{k+1}). \qquad \blacksquare$$

Of course, such an extension $\bar{U}_k^{[k+1]}$ is not unique. The bar in the notation of the extended measure vector and CCC–system emphasizes that $d\bar{\sigma}_{k+1}$ is, in fact, arbitrary. It is also obvious that $(\bar{U}_k^{[k+1]})^{[k]} = U_k$.

The i^{th} reduced system is defined to be a Chebyshev system corresponding to the reduced measure vector, that is

$$d\boldsymbol{\sigma}^{(i)}(\delta) := (d\sigma_{i+2}(\delta), \ldots, d\sigma_k(\delta))^{\mathrm{T}} \in \mathbb{R}^{k-(i+1)},\ i = 1, \ldots k-2,$$

for $\delta \subset [a,b]$ measurable with respect to all $d\sigma_j$. We write $U_k^{(i)} := \{u_{i,j}\}_{j=1}^{k-i}$ for $i = 1, 2, \ldots, k-1$, where

$$\begin{aligned} u_{i,1}(x) &= 1, \\ u_{i,2}(x) &= \int_a^x d\sigma_{i+2}(\tau_{i+2}), \\ &\vdots \\ u_{i,k-i}(x) &= \int_a^x d\sigma_{i+2}(\tau_{i+2}) \ldots \int_a^{\tau_{k-1}} d\sigma_k(\tau_k). \end{aligned}$$

We can also define $d\boldsymbol{\sigma}^{(0)} := d\boldsymbol{\sigma}$, $d\boldsymbol{\sigma}^{(k-1)} := \emptyset$ and $u_{0,j} := u_j(x)$ for $j = 1, \ldots, k$, as well as $U_k^{(k-1)} := \{1\}$.

Definition 2.3 *The linear space $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1) := \operatorname{span}\{U_k\}$ is called a CCC–space if U_k is a CCC–system. If $u_1 \equiv 1$ then $\mathcal{S}(k, d\boldsymbol{\sigma}) := \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$.*

Remark 2.1 *We denote the Lebesgue measure as $d\lambda$, the measure vector with all measures equal to the Lebesgue measure as $d\boldsymbol{\lambda} := (d\lambda, \ldots, d\lambda)$, and the space of polynomial splines as $\mathcal{P}_k := \mathcal{S}(k, d\boldsymbol{\lambda})$.*

Definition 2.4 *Assuming that each of the functions u_1, σ_2, …,σ_k have been extended to an interval to the left of a and to the right of b, i.e. to an interval $[c,d] \supset [a,b]$ with $c < a$, $d > b$ according to Lemma 2.1, then for $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ and $x \in [a,b]$ we define generalized derivatives as*

$$L_{(j,d\boldsymbol{\sigma},u_1)} := D_j \cdots D_1 D_0, \quad j = 0, \ldots, k-1,$$

where

$$\begin{aligned} D_0 f(x) &:= \frac{f(x)}{u_1(x)}, \\ D_j f(x) &:= \lim_{\delta \to 0^+} \frac{f(x+\delta) - f(x)}{\sigma_{j+1}(x+\delta) - \sigma_{j+1}(x)},\ j = 1, \ldots, k-1. \end{aligned}$$

If $u_1 \equiv 1$ then $L_{(j,d\boldsymbol{\sigma})} := L_{(j,d\boldsymbol{\sigma},u_1)}$.

According to Theorem 2.1, we can also define

$$D_k f(x) := \lim_{\delta \to 0^+} \frac{f(x+\delta) - f(x)}{\bar{\sigma}_{k+1}(x+\delta) - \bar{\sigma}_{k+1}(x)},$$

for $x \in [a,b]$, and $L_{(k,d\boldsymbol{\sigma},u_1)} := L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} := D_k \cdots D_1 D_0$. Then we have

$$L_{(j,d\boldsymbol{\sigma},u_1)} u_i = \begin{cases} 0, & i = 1, 2, \ldots, j \\ u_{j,i-j}, & i = j+1, \ldots, k, \end{cases} \tag{2.2}$$

for $j = 0, 1, \ldots, k$.

Lemma 2.2 *The functions $u_1, u_2, \ldots, u_k$ are linearly independent.*

Proof: Let $x \in [a, b]$, and assume that

$$a_1 u_1(x) + a_2 u_2(x) + \cdots + a_k u_k(x) = 0 \tag{2.3}$$

for some $a_1, a_2, \ldots, a_k \in \mathbb{R}$. If we apply general derivatives $L_{(j,d\boldsymbol{\sigma},u_1)}$, for $j = 1, \ldots, k-1$, on (2.3), we get a system of k equations

$$a_1 L_{(j,d\boldsymbol{\sigma},u_1)} u_1(x) + a_2 L_{(j,d\boldsymbol{\sigma},u_1)} u_2(x) + \cdots + a_k L_{(j,d\boldsymbol{\sigma},u_1)} u_k(x) = 0, \quad j = 0, 1, \ldots, k-1.$$

Because of (2.2), the determinant of this system is

$$\det \begin{bmatrix} u_1(x) & u_2(x) & u_3(x) & \cdots & u_k(x) \\ 0 & 1 & u_{1,2}(x) & \cdots & u_{1,k-1}(x) \\ 0 & 0 & 1 & \cdots & u_{2,k-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} = u_1(x) > 0$$

for each $x \in [a, b]$, so the only solution is $a_j = 0$ for $j = 1, 2, \ldots, k$. ■
The direct consequence of Lemma 2.2 and the equations (2.2) is:

Theorem 2.2 *CCC–space $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ is k–dimensional linear space and $L_{(j,d\boldsymbol{\sigma},u_1)}$ is liner operator $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1) \to \mathcal{S}(k-j, d\boldsymbol{\sigma}^{(j)}) := \operatorname{span}\{U_k^{(j)}\}$ for $j = 1, \ldots, k-1$.*

Finally, given $u_1, \ldots, u_k$ defined as in (2.1), we define the dual CCC–system $U_k^* = \{u_i^*\}_{i=1}^k$ by

$$\begin{aligned} u_1^*(y) &= 1, \\ u_2^*(y) &= \int_a^y d\sigma_k(\tau_k), \\ &\vdots \\ u_k^*(y) &= \int_a^y d\sigma_k(\tau_k) \int_a^{\tau_k} \cdots \int_a^{\tau_3} d\sigma_2(\tau_2). \end{aligned}$$

Let $d\boldsymbol{\sigma}^* := (d\sigma_k, \ldots d\sigma_2)^{\mathrm{T}}$, then $\mathcal{S}(k, d\boldsymbol{\sigma}^*) := \operatorname{span}\{u_1^*, u_2^*, \ldots u_k^*\}$. For a given dual CCC–system, as for any CCC–system, we define $d\boldsymbol{\sigma}^{(*,(i))} := (d\sigma_{k-i}, \ldots, d\sigma_2)$ and the i^{th} reduced system $U_k^{(*,(i))} := \{u_{i,j}^*\}_{j=1}^{k-i}$ by

$$\begin{aligned} u_{i,1}^*(y) &= 1, \\ u_{i,2}^*(y) &= \int_a^y d\sigma_{k-i}(\tau_{k-i}) \\ &\vdots \\ u_{i,k-i}^*(y) &= \int_a^y d\sigma_{k-i}(\tau_{k-i}) \int_a^{\tau_{k-i}} \cdots \int_a^{\tau_3} d\sigma_2(\tau_2), \end{aligned}$$

for $i = 0, 1, \ldots, k-1$. Associated with this CCC–system, we have the dual derivatives $L_{(j,d\boldsymbol{\sigma}^*)} : \mathcal{S}(k, d\boldsymbol{\sigma}^*) \to \mathcal{S}(k-j, d\boldsymbol{\sigma}^{(*,(j))}) := \operatorname{span}\{U_k^{(*,(j))}\}$:

$$L_{(j,d\boldsymbol{\sigma}^*)} := D_j^* \cdots D_1^*, \qquad j = 1, \ldots, k-1,$$

where by Lemma 2.1, for $f \in \mathcal{S}(k, d\boldsymbol{\sigma}^*)$ and $x \in [a, b]$, we define

$$D_j^* f(y) := \lim_{\delta \to 0^+} \frac{f(y) - f(y-\delta)}{\sigma_{k-j+1}(y) - \sigma_{k-j+1}(y-\delta)}, \qquad j = 1, \ldots, k-1.$$

The dual derivatives act on dual CCC–systems like the general derivatives on classic CCC–systems:

$$L_{(j,d\boldsymbol{\sigma}^*)} u_i^* = \begin{cases} 0, & i = 1, 2, \ldots, j \\ u_{j,i-j}^*, & i = j+1, \ldots, k, \end{cases}$$

for $j = 1, \ldots, k-1$.

Remark 2.2 *Further we will use notation* $U_k^{(s,z)} := (U_k^s)^z$ *and* $d\boldsymbol{\sigma}^{(s,z)} := (d\boldsymbol{\sigma}^s)^z$ *where* s *and* z *are any of* (i), $[i]$ *or* $*$.

2.2 Determinants and Chebyshev divided differences

Further, we need a generalization of divided differences, which play a very important role in definition of the local basis for Chebyshev spline spaces, but first, we have to define determinants D in a CCC–system. Let U_k be the given CCC–system, and $v_1, \ldots, v_k \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$. Then, if

$$M \begin{pmatrix} t_1, \ldots, t_k \\ v_1, \ldots, v_k \end{pmatrix} := \left[L_{(d_i, d\boldsymbol{\sigma}, u_1)} v_j(t_i) \right]_{i,j=1}^k,$$

where $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_k$ and

$$d_i := \max\{j : t_i = \cdots, t_{i-j}\}, \qquad i = 1, 2, \ldots, k,$$

then

$$D \begin{pmatrix} t_1, \ldots, t_k \\ v_1, \ldots, v_k \end{pmatrix} := \det M \begin{pmatrix} t_1, \ldots, t_k \\ v_1, \ldots, v_k \end{pmatrix}.$$

The next lemma is important to prove Theorem 2.3, and it is a variant of the Newton–Leibniz formula for Chebyshev spaces, which is the basis of the whole Chebyshev theory.

Lemma 2.3 (Chebyshev Newton–Leibniz formula) *Let* $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ *be a CCC–space on* $[a, b]$, $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ *and* $a \leqslant c \leqslant d \leqslant b$. *Then*

$$\frac{f(d)}{u_1(d)} - \frac{f(c)}{u_1(c)} = \int_c^d L_{(1,d\boldsymbol{\sigma},u_1)} f(t)\, d\sigma_2(t). \tag{2.4}$$

Proof: We will prove that (2.4) holds for the basis functions in $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$, *i.e.* that

$$\frac{u_j(d)}{u_1(d)} - \frac{u_j(c)}{u_1(c)} = \int_c^d L_{(1,d\boldsymbol{\sigma},u_1)} u_j(t)\, d\sigma_2(t) \tag{2.5}$$

holds for $j = 1, 2, \ldots, k$. For $j = 1$ (2.5) is trivial. Let us prove (2.5) for fixed j, $j = 2, \ldots, k$. We can show a stronger assertion:

$$\psi_i^j(d) - \psi_i^j(c) = \int_c^d D_{i-1}\psi_i^j(t)\, d\sigma_i(t) \tag{2.6}$$

for $i = 2, \ldots, j$, where

$$\psi_i^j(t) = \int_a^t d\sigma_i(\tau_i) \int_a^{\tau_i} d\sigma_{i+1}(\tau_{i+1}) \cdots \int_a^{\tau_j - 1} d\sigma_j(\tau_j).$$

To do that, we proceed by induction on i. The function ψ_i^j has few simple properties:

$$\psi_i^j(t) > 0 \text{ for } x > a, \text{ and } \psi_i^j(a) = 0, \tag{2.7}$$

$$\psi_i^j(t) = \int_a^t \psi_{i+1}^j(\tau_i)\, d\sigma_i(\tau_i),$$

$$D_{i-1}\psi_i^j(t) = \psi_{i+1}^j(t), \tag{2.8}$$

and because of (2.7) and (2.8), ψ_i^j is an increasing function. The basis of induction ($i = j$) is trivial, because $\psi_j^j(t) = \int_a^t d\sigma_j(\tau_j)$. Now, assume that (2.6) holds for $j, j-1, \ldots, i+1$, and let us show it holds for i. Let us start with the right hand side of the equation (2.6):

$$\begin{aligned}
\int_c^d D_{i-1}\psi_i^j(t)\, d\sigma_i(t) &= \int_c^d \lim_{\delta\to 0^+} \frac{\psi_i^j(t+\delta) - \psi_i^j(t)}{\sigma_i(t+\delta) - \sigma_i(t)}\, d\sigma_i(t) \\
&= \int_c^d \lim_{\delta\to 0^+} \frac{\int_t^{t+\delta} \psi_{i+1}^j(\tau_i)\, d\sigma_i(\tau_i)}{\sigma_i(t+\delta) - \sigma_i(t)}\, d\sigma_i(t) \\
&= \int_c^d \lim_{\delta\to 0^+} \frac{\int_t^{t+\delta} d\sigma_i(\tau_i) \int_a^{\tau_i} \psi_{i+2}^j(\tau_{i+1})\, d\sigma_{i+1}(\tau_{i+1})}{\sigma_i(t+\delta) - \sigma_i(t)}\, d\sigma_i(t) \\
&= \int_c^d \lim_{\delta\to 0^+} \frac{\int_t^{t+\delta} d\sigma_i(\tau_i) \int_a^{\tau_i} D_i\psi_{i+1}^j(\tau_{i+1})\, d\sigma_{i+1}(\tau_{i+1})}{\sigma_i(t+\delta) - \sigma_i(t)}\, d\sigma_i(t),
\end{aligned}$$

and if we take into account the induction assumption for ψ_{i+1}^j, we get

$$\begin{aligned}
\int_c^d D_{i-1}\psi_i^j(t) d\sigma_i(t) &= \int_c^d \lim_{\delta\to 0^+} \frac{\int_t^{t+\delta} \left(\psi_{i+1}^j(\tau_i) - \psi_{i+1}^j(a)\right) d\sigma_i(\tau_i)}{\sigma_i(t+\delta) - \sigma_i(t)}\, d\sigma_i(t) \\
&= \int_c^d \lim_{\delta\to 0^+} \left(\frac{1}{\sigma_i(t+\delta) - \sigma_i(t)} \int_t^{t+\delta} \psi_{i+1}^j(\tau_i)\, d\sigma_i(\tau_i) \right. \\
&\qquad \left. -\psi_{i+1}^j(a) \right) d\sigma_i(t).
\end{aligned}$$

Then, we apply $\psi^j_{i+1}(a) = 0$ to the equation above, and use the Mean value theorem for Lebesgue–Stieltjes integrals (see [4]) applied on ψ^j_{i+1}, which says that there exist $\xi_{i+1}(\delta)$, $\psi^j_{i+1}(t) \leqslant \xi_{i+1}(\delta) \leqslant \psi^j_{i+1}(t+\delta)$, ($\psi^j_{i+1}$ is increasing and, in general, only right continuous), such that

$$\int_t^{t+\delta} \psi^j_{i+1}(\tau_i)\, d\sigma_i(\tau_i) = \xi_{i+1}(\delta) \int_t^{t+\delta} d\sigma_i(\tau_i),$$

and we get

$$\begin{aligned}\int_c^d D_{i-1}\psi^j_i(t)\, d\sigma_i(t) &= \int_c^d \lim_{\delta\to 0^+} \left(\frac{\xi_{i+1}(\delta)}{\sigma_i(t+\delta) - \sigma_i(t)} \int_t^{t+\delta} d\sigma_i(\tau_i) \right) d\sigma_i(t) \\ &= \int_c^d \psi^j_{i+1}(t)\, d\sigma_i(t) \\ &= \psi^j_i(d) - \psi^j_i(c),\end{aligned}$$

so, the assertion (2.6) is proved. ■

Lemma 2.3 also produces a generalization of Taylor expansion for polynomials:

Lemma 2.4 (Chebyshev Taylor Expansion) *Let $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ be a CCC–space on $[a,b]$, $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ and $a \leqslant c \leqslant d \leqslant b$. Then for $l = 2, \cdots, k$*

$$\begin{aligned}\frac{f(d)}{u_1(d)} = \frac{f(c)}{u_1(c)} &+ L_{(1,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) + L_{(2,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) \int_c^{\tau_2} d\sigma_3(\tau_3) \\ &+ \cdots + L_{(l-2,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) \cdots \int_c^{\tau_{l-2}} d\sigma_{l-1}(\tau_{l-1}) \\ &+ \int_c^d d\sigma_2(\tau_2) \cdots \int_c^{\tau_{l-2}} d\sigma_{l-1}(\tau_{l-1}) \int_c^{\tau_{l-1}} L_{(l-1,d\boldsymbol{\sigma},u_1)}f(\tau_l)\, d\sigma_l(\tau_l),\end{aligned} \tag{2.9}$$

and there exist $\xi(d)$, $\inf_{x\in[c,d]} L_{(l-1,d\boldsymbol{\sigma},u_1)}f \leqslant \xi(d) \leqslant \sup_{x\in[c,d]} L_{(l-1,d\boldsymbol{\sigma},u_1)}f$ such that

$$\begin{aligned}\frac{f(d)}{u_1(d)} = \frac{f(c)}{u_1(c)} &+ L_{(1,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) + L_{(2,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) \int_c^{\tau_2} d\sigma_3(\tau_3) \\ &+ \cdots + L_{(l-2,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) \cdots \int_c^{\tau_{l-2}} d\sigma_{l-1}(\tau_{l-1}) \\ &+ \xi(d) \int_c^d d\sigma_2(\tau_2) \cdots \int_c^{\tau_{l-2}} d\sigma_{l-1}(\tau_{l-1}) \int_c^{\tau_{l-1}} d\sigma_l(\tau_l).\end{aligned} \tag{2.10}$$

Proof: We will prove (2.9) by induction. The basis is the Chebyshev Newton–Leibniz formula (2.4). Let us assume that (2.9) holds for $l-1$:

$$\begin{aligned}\frac{f(d)}{u_1(d)} = \frac{f(c)}{u_1(c)} &+ L_{(1,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) + L_{(2,d\boldsymbol{\sigma},u_1)}f(c) \int_c^d d\sigma_2(\tau_2) \int_c^{\tau_2} d\sigma_3(\tau_3) \\ &+ \cdots + \int_c^d d\sigma_2(\tau_2) \cdots \int_c^{\tau_{l-2}} L_{(l-2,d\boldsymbol{\sigma},u_1)}f(\tau_{l-1})\, d\sigma_{l-1}(\tau_{l-1})\end{aligned}$$

$$= \frac{f(c)}{u_1(c)} + L_{(1,d\boldsymbol{\sigma},u_1)}f(c)\int_c^d d\sigma_2(\tau_2) + L_{(2,d\boldsymbol{\sigma},u_1)}f(c)\int_c^d d\sigma_2(\tau_2)\int_c^{\tau_2} d\sigma_3(\tau_3)$$
$$+\cdots+\int_c^d d\sigma_2(\tau_2)\cdots\int_c^{\tau_{l-2}}\Big((L_{(l-2,d\boldsymbol{\sigma},u_1)}f(\tau_{l-1}) - L_{(l-2,d\boldsymbol{\sigma},u_1)}f(c))$$
$$+L_{(l-2,d\boldsymbol{\sigma},u_1)}f(c)\Big)d\sigma_l(\tau_{l-1}),$$

then, again, by the Chebyshev Newton–Leibniz formula (2.4) applied on $L_{(l-2,d\boldsymbol{\sigma},u_1)}f$, this becomes (2.9). For proving (2.10) we need the fact that by the Mean value theorem for Lebesgue–Stieltjes integrals we have

$$\int_c^d f(\tau_i)g(\tau_i)\,d\sigma_i(\tau_i) = \xi_{i-1}(d)\int_c^d g(\tau_i)\,d\sigma_i(\tau_i), \tag{2.11}$$

for f and g right continuous, f bounded, $g \geqslant 0$ and

$$m(f,[c,d]) := \inf_{x\in[c,d]} f(x) \leqslant \xi_{i-1}(d) \leqslant M(f,[c,d]) := \sup_{x\in[c,d]} f(x). \tag{2.12}$$

It is obvious that ξ_{i-1} is also right continuous. Let

$$\begin{aligned}
\xi_l &:\equiv L_{(l-1,d\boldsymbol{\sigma},u_1)}f,\\
g_l^l &:\equiv 1,\\
g_i^l(d) &:= \int_c^d d\sigma_{i+1}(\tau_{i+1})\cdots\int_c^{\tau_{l-1}} d\sigma_l(\tau_l), \quad \text{for } l = 2,\ldots,l-1.
\end{aligned}$$

Then by (2.11) and the inductive argument $g_i^l \geqslant 0$, and

$$\int_c^{\tau_{i-1}} \xi_i(\tau_i)g_i^l(\tau_i)\,d\sigma_i(\tau_i) = \xi_{i-1}(\tau_{i-1})\int_c^{\tau_{i-1}} g_i^l(\tau_i)\,d\sigma_i(\tau_i),$$

with $m(L_{(l-1,d\boldsymbol{\sigma},u_1)}f,[c,d]) \leqslant m(\xi_i,[c,\tau_{i-1}]) \leqslant \xi_{i-1}(\tau_{i-1}) \leqslant M(\xi_i,[c,\tau_{i-1}]) \leqslant M(L_{(l-1,d\boldsymbol{\sigma},u_1)}f,[c,d])$ and

$$\lim_{\tau_{i-1}\to c^+} \xi_{i-1}(\tau_{i-1}) = L_{(l-1,d\boldsymbol{\sigma},u_1)}f(c). \tag{2.13}$$

For $\xi :\equiv \xi_1$ (2.10) is proved. ■

Theorem 2.3 *Let U_k be a CCC–system on the interval $[a,b]$. Then*

$$D\begin{pmatrix} t_1,\ldots,t_k \\ u_1,\ldots,u_k \end{pmatrix} > 0$$

for all $a \leqslant t_1 \leqslant t_2 \leqslant \cdots \leqslant t_k \leqslant b$.

Proof: The proof in [38] relies only on Lemma 2.3.

Definition 2.5 *Let $a = t_1 \leqslant t_2 \leqslant \cdots \leqslant t_{k+1} = b$, where each t_i is repeated l_i times. Let U_{k+1} be a CCC–system and f a function for which all $L_{(j-1,d\boldsymbol{\sigma}^{[k]},u_1)}f(t_i)$, $j = 1, \ldots, l_i$ exist.* Chebyshev k^{th} order divided difference with respect to CCC–system *U_{k+1} is a linear functional*

$$[t_1, t_2, \ldots, t_{k+1}]_{U_{k+1}} f = \frac{D\begin{pmatrix} t_1, \ldots, t_{k+1} \\ u_1, \ldots, u_k, f \end{pmatrix}}{D\begin{pmatrix} t_1, \ldots, t_{k+1} \\ u_1, \ldots, u_{k+1} \end{pmatrix}}. \tag{2.14}$$

Some properties of divided differences are listed in the next theorem:

Theorem 2.4

(i) Divided difference is a linear functional of the form:

$$[t_1, \ldots, t_{k+1}]_{U_{k+1}} f = \sum_{i=1}^{d} \sum_{j=1}^{l_i} \alpha_{i,j} L_{(j-1,d\boldsymbol{\sigma}^{[k]},u_1)} f(t_i),$$

where $\alpha_{i,j} \in \mathbb{R}$, $\alpha_{i,l_i} \neq 0$, and the function f satisfies the smoothness conditions from Definition 2.5.

(ii)

$$[t_1, \ldots, t_{k+1}]_{U_{k+1}} u = 0, \qquad for\ each\ u \in \mathcal{S}(k, d\boldsymbol{\sigma}^{[k]}, u_1).$$

(iii) If $t_{1,\delta}, \ldots, t_{k+1,\delta}$ is a sequence of points satisfying $t_{i,\delta} \to t_i^+$ when $\delta \to 0$, then for sufficiently smooth function f (as in Definition 2.5)

$$[t_{1,\delta}, \ldots, t_{k+1,\delta}]_{U_{k+1}} f \longrightarrow [t_1, \ldots, t_{k+1}]_{U_{k+1}} f$$

for $\delta \to 0$.

(iv) If $t_1 \neq t_{k+1}$, divided differences satisfy the recurrence:

$$[t_1, \ldots, t_{k+1}]_{U_{k+1}} f = \frac{[t_2, \ldots, t_{k+1}]_{U_{k+1}^{[k]}} f - [t_1, \ldots, t_k]_{U_{k+1}^{[k]}} f}{[t_2, \ldots, t_{k+1}]_{U_{k+1}^{[k]}} u_{k+1} - [t_1, \ldots, t_k]_{U_{k+1}^{[k]}} u_{k+1}}. \tag{2.15}$$

Proof: The proof of (i) and (ii) is analogous to the proof for ECC–systems [39, 26]; the recurrence for divided differences in (iv) is the result of Mülbach's recurrence [25] and can easily be extended to CCC–systems [30].

Let us prove (iii). It is sufficient to consider the case where just one t moves with δ. Now suppose $t_1 \leqslant \cdots \leqslant t_i < t_{i+1} = \cdots = t_{i+l} < t_{i+l+1} \leqslant \cdots \leqslant t_{k+1}$, and

that $t_{i+l,\delta} \to t_{i+l}^+$. Then

$$[t_1,\dots,t_{i+l-1},t_{i+l,\delta},t_{i+l+1},\dots,t_{k+1}]_{U_{k+1}}f$$
$$= \frac{\det\begin{bmatrix} \cdots & & \cdots \\ u_1(t_{i+1}) & \cdots & u_k(t_{i+1}) & f(t_{i+1}) \\ \cdots & & \\ L_{(l-2,d\boldsymbol{\sigma},u_1)}u_1(t_{i+1}) & \cdots & L_{(l-2,d\boldsymbol{\sigma},u_1)}u_k(t_{i+1}) & L_{(l-2,d\boldsymbol{\sigma},u_1)}f(t_{i+1}) \\ u_1(t_{i+l,\delta}) & \cdots & u_k(t_{i+l,\delta}) & f(t_{i+l,\delta}) \\ \cdots & & & \cdots \end{bmatrix}}{\det\begin{bmatrix} \cdots & & \cdots \\ u_1(t_{i+1}) & \cdots & u_{k+1}(t_{i+1}) \\ \cdots & & \\ L_{(l-2,d\boldsymbol{\sigma},u_1)}u_1(t_{i+1}) & \cdots & L_{(l-2,d\boldsymbol{\sigma},u_1)}u_{k+1}(t_{i+1}) \\ u_1(t_{i+l,\delta}) & \cdots & u_{k+1}(t_{i+l,\delta}) \\ \cdots & & \cdots \end{bmatrix}}.$$

After applying Chebyshev Taylor expansion (2.10) on $u_1(t_{i+l,\delta}),\dots,u_{k+1}(t_{i+l,\delta})$, $f(t_{i+l,\delta})$ and some simplifications, we get

$$[t_1,\dots,t_{i+l-1},t_{i+l,\delta},t_{i+l+1},\dots,t_{k+1}]_{U_{k+1}}f$$
$$= \frac{\det\begin{bmatrix} \cdots & & \cdots \\ u_1(t_{i+1}) & \cdots & u_k(t_{i+1}) & f(t_{i+1}) \\ \cdots & & \\ L_{(l-2,d\boldsymbol{\sigma},u_1)}u_1(t_{i+1}) & \cdots & L_{(l-2,d\boldsymbol{\sigma},u_1)}u_k(t_{i+1}) & L_{(l-2,d\boldsymbol{\sigma},u_1)}f(t_{i+1}) \\ \xi_1(t_{i+l,\delta}) & \cdots & \xi_k(t_{i+l,\delta}) & \xi(t_{i+l,\delta}) \\ \cdots & & & \cdots \end{bmatrix}}{\det\begin{bmatrix} \cdots & & \cdots \\ u_1(t_{i+1}) & \cdots & u_{k+1}(t_{i+1}) \\ \cdots & & \\ L_{(l-2,d\boldsymbol{\sigma},u_1)}u_1(t_{i+1}) & \cdots & L_{(l-2,d\boldsymbol{\sigma},u_1)}u_{k+1}(t_{i+1}) \\ \xi_1(t_{i+l,\delta}) & \cdots & \xi_{k+1}(t_{i+l,\delta}) \\ \cdots & & \cdots \end{bmatrix}},$$

for $L_{(l-1,d\boldsymbol{\sigma},u_1)}u_i(t_{i+l}) \leqslant \xi_i(t_{i+l,\delta}) \leqslant L_{(l-1,d\boldsymbol{\sigma},u_1)}u_i(t_{i+l,\delta})$, $i = 1,\dots,k+1$, because $L_{(l-1,d\boldsymbol{\sigma},u_1)}u_i$ are increasing, and (see (2.12))

$$m(L_{(l-1,d\boldsymbol{\sigma},u_1)}f,[t_{i+l},t_{i+l,\delta}]) \leqslant \xi(t_{i+l,\delta}) \leqslant M(L_{(l-1,d\boldsymbol{\sigma},u_1)}f,[t_{i+l},t_{i+l,\delta}]).$$

By (2.13) $\xi_i(t_{i+l,\delta}) \to L_{(l-1,d\boldsymbol{\sigma},u_1)}u_i(t_{i+l})$ and $\xi(t_{i+l,\delta}) \to L_{(l-1,d\boldsymbol{\sigma},u_1)}f(t_{i+l})$ as $t_{i+l,\delta} \to t_{i+l}^+$, so by taking the limit, we obtain (iii). ■

Remark 2.3 *As we usually observe CCC–system U_k, then for U_{k+1}, from Definition 2.5, we take $\bar{U}_k^{[k+1]}$. In this case $(\bar{U}_k^{[k+1]})^{[k]} = U_k$.*

2.3 Green's function

Definition 2.6 *For a CCC–space $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ the function $G_{(k,d\boldsymbol{\sigma},u_1)}(x,y) : [a,b] \times [a,b] \to \mathbb{R}$ defined by*

$$G_{(k,d\boldsymbol{\sigma},u_1)}(x,y) = \begin{cases} H_{(k,d\boldsymbol{\sigma},u_1)}(x,y), & x \geqslant y, \\ 0, & \text{otherwise}, \end{cases}$$

where

$$H_{(k,d\boldsymbol{\sigma},u_1)}(x,y) = u_1(x) \int_y^x d\sigma_2(\tau_2) \int_y^{\tau_2} \cdots \int_y^{\tau_{k-1}} d\sigma_k(\tau_k),$$

is called a Green's function with respect to $d\boldsymbol{\sigma}$. If $u_1 \equiv 1$ then $G_{(k,d\boldsymbol{\sigma})} := G_{(k,d\boldsymbol{\sigma},u_1)}$ and $H_{(k,d\boldsymbol{\sigma})} := H_{(k,d\boldsymbol{\sigma},u_1)}$.

The following theorems show some characteristic properties of the Green's function:

Theorem 2.5 *For $a \leqslant y \leqslant x \leqslant b$ and $i = 0, 1 \ldots k-2$ holds*

$$G_{(k-i,d\boldsymbol{\sigma}^{(i)},u_{i,1})}(x,y) = \sum_{l=i+1}^{k} (-1)^{k-l} u_{i,l-i}(x) u^*_{k-l+1}(y). \tag{2.16}$$

Proof: To prove (2.16), let us define:

$$\psi_i(x,y) := \delta_i(x) \int_y^x d\sigma_{i+2}(\tau_{i+2}) \int_y^{\tau_{i+2}} d\sigma_{i+3}(\tau_{i+3}) \cdots \int_y^{\tau_{k-1}} d\sigma_k(\tau_k)$$

for $i = 0, 1, \ldots, k-2$, with

$$\delta_i(x) = \begin{cases} u_1(x) & \text{for } i = 0, \\ 1 & \text{for } i = 1, \ldots, k-2. \end{cases}$$

So, we want to establish even more, that

$$\psi_i(x,y) = \sum_{l=i+1}^{k} (-1)^{k-l} u_{i,l-i}(x) u^*_{k-l+1}(y), \tag{2.17}$$

holds for $i = 0, 1, \ldots, k-2$ and all $x, y \in [a,b]$. We prove this by induction on i. For $i = k-2$ we have

$$\begin{aligned} \delta_{k-2}(x) \int_y^x d\sigma_k(\tau_k) &= \delta_{k-2}(x) \int_a^x d\sigma_k(\tau_k) - \delta_{k-2}(x) \int_a^y d\sigma_k(\tau_k) \\ &= u_{k-2,2}(x) u^*_1(y) - u_{k-2,1}(x) u^*_2(y), \end{aligned}$$

which is (2.17) in this case. Now assume that (2.17) holds for $i+1, \ldots, k-2$, and we prove it for i. We have

$$\psi_i(x,y) = \delta_i(x) \int_a^x \psi_{i+1}(\tau_{i+2}, y)\, d\sigma_{i+2}(\tau_{i+2}) - \delta_i(x) \int_a^y \psi_{i+1}(\tau_{i+2}, y)\, d\sigma_{i+2}(\tau_{i+2}). \tag{2.18}$$

After applying the induction assumption to the first term of the right hand side of (2.18), we see that it reduces to

$$\sum_{l=i+2}^{k} (-1)^{k-l} u_{i,l-i}(x) u^*_{k-l+1}(y).$$

Now, a simple induction argument shows that

$$\int_a^y d\sigma_{i+2}(\tau_{i+2}) \int_y^{\tau_{i+2}} d\sigma_{i+3}(\tau_{i+3}) \cdots \int_y^{\tau_{k-1}} d\sigma_k(\tau_k)$$
$$= (-1)^{k-i} \int_a^y d\sigma_k(\tau_k) \int_a^{\tau_k} d\sigma_{k-1}(\tau_{k-1}) \cdots \int_a^{\tau_{i+3}} d\sigma_{i+2}(\tau_{i+2}).$$

Since $u_{i,1}(x) = 1$ for $i > 0$, the second term of (2.18) is

$$(-1)^{k-i-1} u_{i,1}(x) u^*_{k-i}(y),$$

and (2.17) is proved for i. ■

Theorem 2.6

$$L_{(i,d\boldsymbol{\sigma},u_1)} G_{(k,d\boldsymbol{\sigma},u_1)}(\cdot, y) = G_{(k-i,d\boldsymbol{\sigma}^{(i)})}(\cdot, y)$$

for $i = 1, \ldots, k-1$.

Proof: If $x > y$, according to Theorem 2.5:

$$G_{(k,d\boldsymbol{\sigma},u_1)}(x, y) = \sum_{l=1}^{k} (-1)^{k-l} u_l(x) u^*_{k+1-l}(y).$$

From this, and (2.2), it follows that

$$L_{(i,d\boldsymbol{\sigma},u_1)} G_{(k,d\boldsymbol{\sigma},u_1)}(x, y) = \sum_{l=i+1}^{k} (-1)^{k-l} u_{i,l-i}(x) u^*_{k+1-l}(y).$$

The right hand side of the equation can be recognized as $G_{(k-i,d\boldsymbol{\sigma}^{(i)})}$. ■

Theorem 2.7 *Let* $r \in \mathrm{L}_1[a, b]$, *and suppose* $f_0, \ldots, f_{k-1}$ *are given real numbers. Let* u *be the unique member of* $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ *such that*

$$L_{(i,d\boldsymbol{\sigma},u_1)} u(a) = f_i, \qquad i = 0, 1, \ldots, k-1,$$

and let the measure $d\bar{\sigma}_{k+1}$ *be defined as in Theorem 2.1. Then*

$$f(x) = u(x) + \int_a^b G_{(k,d\boldsymbol{\sigma},u_1)}(x, \tau_{k+1}) r(\tau_{k+1})\, d\bar{\sigma}_{k+1}(\tau_{k+1})$$

is the unique solution of the initial–value problem

$$L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} f(x) = r(x), \qquad x \in [a, b], \tag{2.19}$$
$$L_{(i,d\boldsymbol{\sigma},u_1)} f(a) = f_i, \qquad i = 0, 1, \ldots, k-1. \tag{2.20}$$

Proof: It is easily checked from Definition 2.6 and Theorem 2.6 that

$$L_{(i,d\boldsymbol{\sigma},u_1)} \int_a^b G_{(k,d\boldsymbol{\sigma},u_1)}(x,\tau_{k+1}) r(\tau_{k+1})\, d\bar{\sigma}_{k+1}(\tau_{k+1})|_{x=a} = 0, \qquad i = 0, 1, \ldots, k-1,$$

and thus f satisfies the initial conditions (2.20). On the other hand, since

$$L_{(k-1,d\boldsymbol{\sigma},u_1)} G_{(k,d\boldsymbol{\sigma},u_1)}(x,y) = \begin{cases} 1, & x \geqslant y, \\ 0, & \text{otherwise}, \end{cases}$$

we have

$$L_{(k-1,d\boldsymbol{\sigma},u_1)} f(x) = L_{(k-1,d\boldsymbol{\sigma},u_1)} u(x) + \int_a^x r(\tau_{k+1})\, d\bar{\sigma}_{k+1}(\tau_{k+1}).$$

After applying the differential operator D_k, with $L_{(k-1,d\boldsymbol{\sigma},u_1)} u(x) = \text{const}$, we obtain (2.19). ■

As in the polynomial case, the function $G_{(k,d\boldsymbol{\sigma},u_1)}$ also plays an important role in generalization of the Taylor expansion, and gives an alternative to the Lemma 2.4:

Theorem 2.8 (Generalized Taylor Expansion) *Suppose* $L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} f \in \mathrm{L}_1[a,b]$. *Then for all* $a \leqslant x \leqslant b$,

$$f(x) = u_f(x) + \int_a^b G_{(k,d\boldsymbol{\sigma},u_1)}(x,\tau_{k+1}) L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} f(\tau_{k+1})\, d\bar{\sigma}_{k+1}(\tau_{k+1}),$$

where u_f *is the function in* $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ *such that*

$$L_{(i,d\boldsymbol{\sigma},u_1)} u_f(a) = L_{(i,d\boldsymbol{\sigma},u_1)} f(a), \qquad i = 0, 1, \ldots, k-1.$$

Proof: Let

$$g(x) = u_f(x) + \int_a^b G_{(k,d\boldsymbol{\sigma},u_1)}(x,\tau_{k+1}) L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} f(\tau_{k+1})\, d\bar{\sigma}_{k+1}(\tau_{k+1}).$$

Theorem 2.7 implies that $L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} f(x) = L_{(k,d\bar{\boldsymbol{\sigma}}^{[k+1]},u_1)} g(x)$, for all $a \leqslant x \leqslant b$, and thus $f - g \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$. But, as Theorem 2.7 also asserts that $L_{(i,d\boldsymbol{\sigma},u_1)}(f - g)(a) = 0$, $i = 0, 1, \ldots, k-1$, we conclude that $f = g$. ■

Analogously, we can prove the dual version of previous theorems:

Theorem 2.9 *If* $d\boldsymbol{\sigma}^{[k-i]} = (d\sigma_2, \ldots, d\sigma_{k-i})^{\mathrm{T}}$ *is the restricted measure vector, then*

$$L_{(i,d\boldsymbol{\sigma}^*)} G_{(k,d\boldsymbol{\sigma},u_1)}(x,\cdot) = (-1)^i G_{(k-i,d\boldsymbol{\sigma}^{[k-i]},u_1)}(x,\cdot)$$

for $i = 1, \ldots, k-1$.

Theorem 2.10 *Let $r \in \mathrm{L}_1[a,b]$, and suppose $f_0, \ldots, f_{k-1}$ are given real numbers. Let u^* be the unique member of $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ such that*

$$L_{(i,d\boldsymbol{\sigma}^*)}u^*(b) = f_i, \qquad i = 0, 1, \ldots, k-1.$$

Then

$$f(y) = u^*(y) + \int_a^b (-1)^k r(\tau_{k+1}) G_{(k,d\boldsymbol{\sigma},u_1)}(\tau_{k+1}, y)\, d\bar{\sigma}^*_{k+1}(\tau_{k+1}),$$

where $d\bar{\sigma}^*_{k+1}$ is the extension of $d\boldsymbol{\sigma}^*$ according to Theorem 2.1, $d\bar{\boldsymbol{\sigma}}^{(*,[k+1])} := (d\sigma_k, \ldots d\sigma_2, \bar{\sigma}^*_{k+1})^{\mathrm{T}}$, is the unique solution of the initial–value problem

$$\begin{aligned} L_{(k,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])})}f(y) &= r(y), \qquad y \in [a,b], \\ L_{(i,d\boldsymbol{\sigma}^*)}f(b) &= f_i, \qquad i = 0, 1, \ldots, k-1. \end{aligned}$$

Theorem 2.11 (Dual Taylor Expansion) *Let $L_{(k,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])})}f \in \mathrm{L}_1[a,b]$. Then for all $a \leqslant y \leqslant b$,*

$$f(y) = u^*_f(y) + \int_a^b G_{(k,d\boldsymbol{\sigma},u_1)}(\tau_{k+1}, y)(-1)^k L_{(k,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])})}f(\tau_{k+1})\, d\bar{\sigma}^*_{k+1}(\tau_{k+1}), \quad (2.21)$$

where $d\bar{\sigma}^*_{k+1}$ is as in Theorem 2.10, and u^*_f is an element of $\mathcal{S}(k, d\boldsymbol{\sigma}^*)$ such that

$$L_{(i,d\boldsymbol{\sigma}^*)}u^*_f(b) = L_{(i,d\boldsymbol{\sigma}^*)}f(b), \qquad i = 0, 1, \ldots, k-1.$$

2.4 Chebyshev B-splines

For a partition $\Delta = \{x_i\}_{i=0}^{l+1}$ of an interval $[a,b]$, given multiplicity vector $\boldsymbol{m} = (m_1, \ldots, m_l)$, $(0 < m_i \leqslant k)$, and $M := \sum_{i=1}^{l} m_i$, we shall denote by $\boldsymbol{T}_{(\Delta,\boldsymbol{m})} := \{t_1 \ldots t_{2k+M}\}$, an extended partition in the usual way:

$$\begin{gathered} t_1 \leqslant \ldots \leqslant t_k = x_0 = a, \\ t_{k+1} \leqslant \ldots \leqslant t_{k+M} = \underbrace{x_1, \ldots, x_1}_{m_1}, \ldots, \underbrace{x_l, \ldots, x_l}_{m_l}, \\ b = x_{l+1} = t_{k+M+1} \leqslant \ldots \leqslant t_{2k+M}. \end{gathered} \quad (2.22)$$

When it is obvious which partition and multiplicity vector the extended partition is associated with, we will use shorter notation: $\boldsymbol{T} := \boldsymbol{T}_{(\Delta,\boldsymbol{m})}$. Most often, to avoid that the first $k-1$ or the last $k-1$ knots lie outside the interval $[a,b]$, we take $t_1 = \ldots = t_k = x_0 = a$ and $b = x_{l+1} = t_{k+M+1} = \ldots = t_{2k+M}$.

Definition 2.7 *Let $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ be a CCC–space. A set $\mathcal{S}(k, d\boldsymbol{\sigma}, u_1, \boldsymbol{T})$ of bounded functions on $[a,b]$ such that:*

(i) for each $s \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1, \boldsymbol{T})$ and each $i = 0, \ldots, l$, there exists $s_i \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1)$ such that $s|_{[x_i, x_{i+1}]} = s_i|_{[x_i, x_{i+1}]}$,

(ii) $L_{(j,d\boldsymbol{\sigma},u_1)}s_{i-1}(x_i) = L_{(j,d\boldsymbol{\sigma},u_1)}s_i(x_i)$ *for* $j = 0,\dots,k-m_i-1$, $i = 1,\dots,l$

is called the space of Chebyshev splines on extended partition $\boldsymbol{T}$. *If* $u_1 \equiv 1$, *then* $\mathcal{S}(k,d\boldsymbol{\sigma},\boldsymbol{T}) := \mathcal{S}(k,d\boldsymbol{\sigma},u_1,\boldsymbol{T})$.

It is clear that $\mathcal{S}(k,d\boldsymbol{\sigma},u_1,\boldsymbol{T})$ is a linear space. It can be easy proved (see [38]) that

$$n := \dim \mathcal{S}(k,d\boldsymbol{\sigma},u_1,\boldsymbol{T}) = k + M.$$

For our purpose, the most convenient basis for $\mathcal{S}(k,d\boldsymbol{\sigma},u_1,\boldsymbol{T})$ are B-splines:

Definition 2.8 *Given an extended partition* $\boldsymbol{T} = \{t_i\}_1^{n+k}$ *as in (2.22), with* $t_i < t_{i+k}$ *for* $i = k,\dots,n$, *and CCC–system* U_k, *we define the associated normalized B-spline, or just B-spline* $T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}$ *as*

$$\begin{aligned} T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) &:= \alpha^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})} Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}(x), \\ Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}(x) &:= (-1)^k [t_i,\dots,t_{i+k}]_{\bar{U}_k^{(*,[k+1])}} G_{(k,d\boldsymbol{\sigma},u_1)}(x,\cdot), \\ \alpha^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})} &:= \frac{D\begin{pmatrix} t_i,\ \dots,\ t_{i+k} \\ u_1^*,\ \dots,\ \bar{u}_{k+1}^* \end{pmatrix} D\begin{pmatrix} t_{i+1},\ \dots,\ t_{i+k-1} \\ u_1^*,\ \dots,\ u_{k-1}^* \end{pmatrix}}{D\begin{pmatrix} t_{i+1},\ \dots,\ t_{i+k} \\ u_1^*,\ \dots,\ u_k^* \end{pmatrix} D\begin{pmatrix} t_i,\ \dots,\ t_{i+k-1} \\ u_1^*,\ \dots,\ u_k^* \end{pmatrix}} \end{aligned} \tag{2.23}$$

for $i = 1,\dots,n+k$, *with the extension* $\bar{u}^*_{k+1}$ *of* U_k^* *as in Theorem 2.1. The spline* $Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}$ *is called unnormalized B-spline. If* $u_1 \equiv 1$ *then* $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})} := T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}$, $Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},\boldsymbol{T})} := Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}$ *and* $\alpha^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},\boldsymbol{T})} := \alpha^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}$.

Theorem 2.12 *B-splines* $T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}$, $i = 1,\dots,n$ *from Definition 2.8 make a basis for* $\mathcal{S}(k,d\boldsymbol{\sigma},u_1,\boldsymbol{T})$, *and* $\alpha^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})} > 0$.

Proof: See [38, 39]. ■

Although unnormalized B-splines $Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}$ in (2.23) depend on extension by measure $d\bar{\sigma}^*_{k+1}$ in the extended CCC–system $\bar{U}_k^{(*,k+1)}$, which can be arbitrarily chosen, for normalized B-splines this is not true:

Lemma 2.5 *The B-spline* $T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}$ *does not depend on the function* $\bar{u}^*_{k+1}$, *i.e. on the measure* $d\bar{\sigma}^*_{k+1}$ *by which we extend the CCC–system.*

Proof: From Definition 2.8 follows that

$$\begin{aligned} T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) = (-1)^k\, & \frac{D\begin{pmatrix} t_i,\ \dots,\ t_{i+k} \\ u_1^*,\ \dots,\ \bar{u}_{k+1}^* \end{pmatrix} D\begin{pmatrix} t_{i+1},\ \dots,\ t_{i+k-1} \\ u_1^*,\ \dots,\ u_{k-1}^* \end{pmatrix}}{D\begin{pmatrix} t_{i+1},\ \dots,\ t_{i+k} \\ u_1^*,\ \dots,\ u_k^* \end{pmatrix} D\begin{pmatrix} t_i,\ \dots,\ t_{i+k-1} \\ u_1^*,\ \dots,\ u_k^* \end{pmatrix}} \\ & \cdot \frac{D\begin{pmatrix} t_i,\ \dots,\ t_{i+k-1},\ t_{i+k} \\ u_1^*,\ \dots,\ u_k^*,\ G_{(k,d\boldsymbol{\sigma},u_1)} \end{pmatrix}}{D\begin{pmatrix} t_i,\ \dots,\ t_{i+k} \\ u_1^*,\ \dots,\ \bar{u}_{k+1}^* \end{pmatrix}}, \end{aligned} \tag{2.24}$$

so determinants which depend on $\bar{u}^*_{k+1}$ cancel. ■

We now continue with some properties of the B-splines:

Theorem 2.13 *Let $\boldsymbol{T} = \{t_i\}_1^{n+k}$ be an extended partition of $[a, b]$, as in (2.22). Let $T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x)$ be the B-spline associated with $t_i, \ldots, t_{i+k}$, $t_i < t_{i+k}$, $i = 1, 2, \ldots, n$. Then*

$$\begin{aligned} T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) &= 0, \qquad x < t_i \text{ and } x > t_{i+k}, \\ T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) &> 0, \qquad t_i < x < t_{i+k}, \end{aligned}$$

$i = 1, 2, \ldots, n$, and

$$\sum_{i=1}^{n} T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) = u_1(x)$$

for all $a \leqslant x \leqslant b$.

Proof: See [38, 39] ■

Remark 2.4 *Direct consequence of Theorem 2.13 is that for $u_1 \equiv 1$ B-splines make a partition of unity:*

$$\sum_{i=1}^{n} T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = 1 \tag{2.25}$$

for all $x \in [a, b]$.

Theorem 2.14 (Peano representation of Chebyshev divided differences) *Let $L_{(k,d\bar{\boldsymbol{\sigma}}_k^{(*,[k+1])})} f \in \mathrm{L}_1[t_i, t_{i+k}]$. Then*

$$[t_i, \ldots, t_{i+k}]_{\bar{U}_k^{(*,[k+1])}} f = \int_{t_i}^{t_{i+k}} Q^k_{(i,d\bar{\boldsymbol{\sigma}}^{(*,[k+1])},u_1,\boldsymbol{T})}(\tau_{k+1}) L_{(k,d\bar{\boldsymbol{\sigma}}_k^{(*,[k+1])})} f(\tau_{k+1}) d\bar{\sigma}^*_{k+1}(\tau_{k+1}),$$

with $d\bar{\sigma}^*_{k+1}$ as in Theorem 2.10.

Proof: We simply apply the divided difference to the dual Taylor expansion (2.21) and use Definition 2.8. ■

Lemma 2.6 *B-spline is equal to the difference of divided differences of Green's function:*

$$\begin{aligned} T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) = (-1)^k \Big(& [t_{i+1}, \ldots, t_{i+k}]_{U_k^*} G_{(k,d\boldsymbol{\sigma},u_1)}(x, \cdot) - \\ & [t_i, \ldots, t_{i+k-1}]_{U_k^*} G_{(k,d\boldsymbol{\sigma},u_1)}(x, \cdot) \Big). \end{aligned}$$

Proof: The well known Sylvester's identity, which is very important in total positivity theory (see page 8 in [10]), gives us

$$D\begin{pmatrix} t_{i+1}, \ldots, t_{i+k-1} \\ u_1^*, \ \ldots, \ u_{k-1}^* \end{pmatrix} D\begin{pmatrix} t_i, \ \ldots, t_{i+k-1}, & t_{i+k} \\ u_1^*, \ \ldots, & u_k^*, & G_{(k,d\boldsymbol{\sigma},u_1)} \end{pmatrix} =$$
$$D\begin{pmatrix} t_{i+1}, \ldots, t_{i+k-1}, & t_{i+k} \\ u_1^*, \ \ldots, & u_{k-1}^*, & G_{(k,d\boldsymbol{\sigma},u_1)} \end{pmatrix} D\begin{pmatrix} t_i, \ \ldots, t_{i+k-1} \\ u_1^*, \ldots, \ u_k^* \end{pmatrix} -$$
$$D\begin{pmatrix} t_i, \ \ldots, t_{i+k-2}, & t_{i+k-1} \\ u_1^*, \ldots, & u_{k-1}^*, & G_{(k,d\boldsymbol{\sigma},u_1)} \end{pmatrix} D\begin{pmatrix} t_{i+1}, \ldots, t_{i+k} \\ u_1^*, \ \ldots, \ u_k^* \end{pmatrix},$$

and the proof follows from (2.24) and (2.23) when we divide both sides of the equation above with the product of positive determinants

$$D\begin{pmatrix} t_{i+1}, \ldots, t_{i+k} \\ u_1^*, \ \ldots, \ u_k^* \end{pmatrix} \cdot D\begin{pmatrix} t_i, \ \ldots, t_{i+k-1} \\ u_1^*, \ldots, \ u_k^* \end{pmatrix},$$

and use the definition of divided difference (2.14). ■

Theorem 2.15 *The B-splines $T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})} \in \mathcal{S}(k, d\boldsymbol{\sigma}, u_1, \boldsymbol{T})$, $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})} \in \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$ satisfy:*

$$T^k_{(i,d\boldsymbol{\sigma},u_1,\boldsymbol{T})}(x) = u_1(x) \cdot T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x), \qquad i = 1, \ldots, n,$$

for $x \in [a, b]$.

Proof: The proof follows directly from (2.24) and Theorem 2.9. ■

Remark 2.5 *Because of Theorem 2.15 from now on we can assume, without loss of generality, that $u_1 \equiv 1$.*

With $u_1 \equiv 1$, B-splines have one more important property: the continuity of B-splines as a function of their knots, or to be precise:

Theorem 2.16 *Let*

$$\mathsf{T}^k_{i,d\boldsymbol{\sigma}}(x; t_i, t_{i+1}, \ldots, t_{i+k}) := T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x), \tag{2.26}$$

for $x \in [a, b]$ and $i = 1, \ldots, n$. If $t_{i,\delta}, \ldots, t_{i+k,\delta}$ is a sequence of points satisfying $t_{j,\delta} \to t_j^+$, $j = i, \ldots, i + k$, when $\delta \to 0$, then

$$\mathsf{T}^k_{i,d\boldsymbol{\sigma}}(x; t_{i,\delta}, \ldots, t_{i+k,\delta}) \longrightarrow \mathsf{T}^k_{i,d\boldsymbol{\sigma}}(x; t_i, \ldots, t_{i+k})$$

when $\delta \to 0$.

Proof: The proof of Theorem 2.16 follows from Lemma 2.6, and Theorem 2.4 (iii). ■

Remark 2.6 *The use of different font in the notation of $\mathsf{T}^k_{i,d\boldsymbol{\sigma}}$ emphasizes that functions $\mathsf{T}^k_{i,d\boldsymbol{\sigma}}$ and $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}$ have different arguments: $\mathsf{T}^k_{i,d\boldsymbol{\sigma}}$ is a function of $x, t_i, t_{i+1}, \ldots, t_{i+k}$, while $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}$ is a function of x, i.e. $t_i, t_{i+1}, \ldots, t_{i+k}$ are fixed.*

It is known that the de Boor–Cox type recurrence does not exist for the general Chebyshev splines [40], so we will use the derivative formula (2.27) [31, 32] when applicable.

Theorem 2.17 (Derivative formula) *Let $L_{(1,d\boldsymbol{\sigma})}$ be the first generalized derivative with respect to CCC–space $\mathcal{S}(k, d\boldsymbol{\sigma})$, and let the multiplicity vector $\boldsymbol{m} = (m_1, \ldots, m_l)$ satisfy $m_i \leqslant k$ for $i = 1, \ldots, l$. Then for $x \in [a, b]$ and $i = 1, \ldots, n$, the following derivative formula holds:*

$$L_{(1,d\boldsymbol{\sigma})} T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = \frac{T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(x)}{C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}} - \frac{T^{k-1}_{(i+1,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(x)}{C^{k-1}_{(i+1,d\boldsymbol{\sigma},\boldsymbol{T})}}, \tag{2.27}$$

where

$$C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})} := \int_{t_i}^{t_{i+k-1}} T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2)\, d\sigma_2(\tau_2). \tag{2.28}$$

Proof: According to Lemma 2.6

$$T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = (-1)^k \Big([t_{i+1}, \ldots, t_{i+k}]_{U_k^*} G_{(k,d\boldsymbol{\sigma})}(x, \cdot) - [t_i, \ldots, t_{i+k-1}]_{U_k^*} G_{(k,d\boldsymbol{\sigma})}(x, \cdot) \Big). \tag{2.29}$$

After applying derivative $L_{(1,d\boldsymbol{\sigma})}$ on (2.29), we conclude that

$$L_{(1,d\boldsymbol{\sigma})} T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = -(\omega_{i+1} - \omega_i),$$

with (see Theorem 2.6)

$$\omega_i := (-1)^{k-1} [t_i, \ldots, t_{i+k-1}]_{U_k^*} G_{(k-1,d\boldsymbol{\sigma}^{(1)})}(x, \cdot).$$

The order of the divided difference can be reduced by Theorem 2.4 (iv):

$$\omega_i = \frac{(-1)^{k-1}}{\gamma_i} \Big([t_{i+1}, \ldots, t_{i+k-1}]_{U_k^{(*,[k-1])}} G_{(k-1,d\boldsymbol{\sigma}^{(1)})}(x, \cdot) - [t_i, \ldots, t_{i+k-2}]_{U_k^{(*,[k-1])}} G_{(k-1,d\boldsymbol{\sigma}^{(1)})}(x, \cdot) \Big),$$

$$\gamma_i := [t_{i+1}, \ldots, t_{i+k-1}]_{U_k^{(*,[k-1])}} u_k^* - [t_i, \ldots, t_{i+k-2}]_{U_k^{(*,[k-1])}} u_k^*.$$

It is trivial to see that $U_k^{(*,[k-1])} = U_k^{((1),*)}$, which implies

$$\omega_i = \frac{(-1)^{k-1}}{\gamma_i} \Big([t_{i+1}, \ldots, t_{i+k-1}]_{U_k^{((1),*)}} G_{(k-1,d\boldsymbol{\sigma}^{(1)})}(x, \cdot) - [t_i, \ldots, t_{i+k-2}]_{U_k^{((1),*)}} G_{(k-1,d\boldsymbol{\sigma}^{(1)})}(x, \cdot) \Big),$$

and

$$\gamma_i := [t_{i+1}, \ldots, t_{i+k-1}]_{U_k^{((1),*)}} u_k^* - [t_i, \ldots, t_{i+k-2}]_{U_k^{((1),*)}} u_k^*. \tag{2.30}$$

As in (2.29), we conclude that

$$\omega_i = \frac{T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}}{\gamma_i},$$

so it only remains to prove that $\gamma_i = C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}$. The recurrence for divided differences (2.15) in $\mathcal{S}(k, d\bar{\boldsymbol{\sigma}}^{((1),*,[k])})$, with $d\bar{\boldsymbol{\sigma}}^{((1),*,[k])} = (d\sigma_k, \ldots, d\sigma_3, d\bar{\sigma}^*_k)^{\mathrm{T}}$ (see Remark 2.3), applied to u^*_k gives

$$[t_i, \ldots, t_{i+k-1}]_{\bar{U}^{((1),*,[k])}_k} u^*_k = \frac{[t_{i+1}, \ldots, t_{i+k-1}]_{U^{((1),*)}_k} u^*_k - [t_i, \ldots, t_{i+k-2}]_{U^{((1),*)}_k} u^*_k}{[t_{i+1}, \ldots, t_{i+k-1}]_{U^{((1),*)}_k} \bar{u}^*_k - [t_i, \ldots, t_{i+k-2}]_{U^{((1),*)}_k} \bar{u}^*_k}, \tag{2.31}$$

where $\bar{u}^*_k$ is the element from the extended reduced system (see Theorem 2.1):

$$\bar{u}^*_k(y) = \int_a^y d\sigma_k(\tau_k) \int_a^{\tau_k} \cdots \int_a^{\tau_4} d\sigma_3(\tau_3) \int_a^{\tau_3} d\bar{\sigma}^*_k(\tau_2).$$

From equations (2.30) and (2.31) follows:

$$\gamma_i = [t_i, \ldots, t_{i+k-1}]_{\bar{U}^{((1),*,[k])}_k} u^*_k \cdot \Big([t_{i+1}, \ldots, t_{i+k-1}]_{U^{((1),*)}_k} \bar{u}^*_k - [t_i, \ldots, t_{i+k-2}]_{U^{((1),*)}_k} \bar{u}^*_k \Big).$$

By using the Peano representation of Chebyshev divided differences (Theorem 2.14), we conclude that

$$[t_i, \ldots, t_{i+k-1}]_{\bar{U}^{((1),*,[k])}_k} u^*_k = \int_{t_i}^{t_{i+k-1}} Q^{k-1}_{(i,d\bar{\boldsymbol{\sigma}}^{((1),*,[k])},\boldsymbol{T})}(\tau_2) L_{(k-1,d\bar{\boldsymbol{\sigma}}^{((1),*,[k])})} u^*_k(\tau_2)\, d\bar{\sigma}^*_k(\tau_2), \tag{2.32}$$

and by trivial calculation we get

$$L_{(k-1,d\bar{\boldsymbol{\sigma}}^{((1),*,[k])})} u^*_k(y) = \bar{D}^*_{k-1} \int_a^y d\sigma_2(\tau_2),$$

with

$$\bar{D}^*_{k-1} f(y) := \lim_{\delta \to 0^+} \frac{f(y) - f(y-\delta)}{d\bar{\sigma}^*_k(y) - d\bar{\sigma}^*_k(y-\delta)}$$

for $f \in \mathcal{S}(k, d\boldsymbol{\sigma}^*, \boldsymbol{T})$. If we choose $d\bar{\sigma}^*_k = d\sigma_2$, then $d\boldsymbol{\sigma}^* = d\bar{\boldsymbol{\sigma}}^{((1),*,[k])}$, and once more, the Sylvester identity gives

$$a^{k-1}_{(i,d\boldsymbol{\sigma}^*,\boldsymbol{T})} = \Big([t_{i+1}, \ldots, t_{i+k-1}]_{U^{((1),*)}_k} \bar{u}^*_k - [t_i, \ldots, t_{i+k-2}]_{U^{((1),*)}_k} \bar{u}^*_k \Big),$$

where $a^{k-1}_{(i,d\boldsymbol{\sigma}^*,\boldsymbol{T})}$ is the normalizing constant from Definition 2.8. Also

$$L_{(k-1,d\bar{\boldsymbol{\sigma}}^{((1),*,[k])})} u^*_k(y) = L_{(k-1,d\boldsymbol{\sigma}^*)} u^*_k(y) = 1,$$

then from (2.32) follows

$$[t_i, \ldots, t_{i+k-1}]_{\bar{U}^{((1),*,[k])}_k} u^*_k = \int_{t_i}^{t_{i+k-1}} Q^{k-1}_{(i,d\boldsymbol{\sigma}^*,\boldsymbol{T})}(\tau_2)\, d\sigma_2(\tau_2).$$

With this we accomplished (2.27). If some $m_i = k-1$ or $m_i = k$, then in (2.27) we just have the right continuity. ■

Corollary 2.1 *Let $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$, $f(x) = \sum_{j=1}^{n} a_j T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x)$ for $x \in [a,b]$, with $\mathcal{S}(k, d\boldsymbol{\sigma})$ and $\boldsymbol{m}$ the multiplicity vector as in Theorem 2.17. Then*

$$L_{(1,d\boldsymbol{\sigma})} f(x) = \sum_{j=2}^{n} \bar{a}_j T^{k-1}_{(j,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(x), \tag{2.33}$$

where

$$\bar{a}_j = \frac{a_j - a_{j-1}}{C^{k-1}_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}} \tag{2.34}$$

for $j = 2, \ldots, n$.

Proof: When we apply $L_{(1,d\boldsymbol{\sigma})}$ on $f(x)$, use Theorem 2.17, and the fact that for $x \in [a,b]$, $T^{k-1}_{(1,d\boldsymbol{\sigma}^{(1)})}(x) = T^{k-1}_{(n+1,d\boldsymbol{\sigma}^{(1)})}(x) = 0$, we get (2.33) with coefficients (2.34). ■

Corollary 2.2 *Let the CCC–space $\mathcal{S}(k, d\boldsymbol{\sigma})$ and the multiplicity vector $\boldsymbol{m}$ be as in Theorem 2.17, and $x \in [a,b]$. Then*

$$T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) - T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(t_i) = \frac{\int_{t_i}^{x} T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2) d\sigma_2(\tau_2)}{C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}} - \frac{\int_{t_{i+1}}^{x} T^{k-1}_{(i+1,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2) d\sigma_2(\tau_2)}{C^{k-1}_{(i+1,d\boldsymbol{\sigma},\boldsymbol{T})}}. \tag{2.35}$$

Proof: If the multiplicity of every knot contained in the support $[t_i, t_{i+k}]$ of the B-spline $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}$ is less then k, or the multiplicity of the t_{i+k} is equal to k, then by integrating the derivative formula (2.27) from t_i to x, we get our assertion (2.35). In this case $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(t_i) = 0$. When the multiplicity of t_i is equal to k, the derivative formula (2.27) gives

$$L_{(1,d\boldsymbol{\sigma})} T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = -\frac{T^{k-1}_{(i+1,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(x)}{C^{k-1}_{(i+1,d\boldsymbol{\sigma},\boldsymbol{T})}}.$$

By integrating this equation, together with $T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(t_i) = 1$, we get

$$T^k_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}(x) - 1 = -\frac{\int_{t_{i+1}}^{x} T^{k-1}_{(i+1,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2)\, d\sigma_2(\tau_2)}{C^{k-1}_{(i+1,d\boldsymbol{\sigma},\boldsymbol{T})}},$$

what is again equal to the assertion (2.35), upon which the de Boor's maxim is applied: anything multiplied by zero is zero, in fact, in our case, zero divided by anything is zero. ■

Corollary 2.3 *Let the CCC–space $\mathcal{S}(k, d\boldsymbol{\sigma})$ and the multiplicity vector $\boldsymbol{m}$ be as in Theorem 2.17, $t_i < t_{i+k-1}$ and $x \in [a,b]$. Then*

$$\int_a^x T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2)\, d\sigma_2(\tau_2) = C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})} \sum_{j=i}^{n} T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x). \tag{2.36}$$

Proof: For $x \in [t_i, t_{i+k-1}]$ by consecutive use of Corollary 2.2 on $T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}$, $j = i, \ldots, i+k-2$, and because $T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = 0$, $j = i+k-1, \ldots, n$ for $x \in [t_i, t_{i+k-1}]$, we have

$$\int_a^x T^{k-1}_{(i,d\boldsymbol{\sigma}^{(1)},\boldsymbol{T})}(\tau_2)\, d\sigma_2(\tau_2) = C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})} \sum_{j=i}^{i+k-2} \left(T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x) - T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(t_i)\right),$$

but as $\{t_j\}_{j=i}^{i+k-1}$ contains only the points of multiplicity less than k, because of the assumption $t_i < t_{i+k-1}$, the previous equation gives (2.36). For $x < t_i$, both sides of the equation (2.36) are equal to zero, while for $x > t_{i+k-1}$

$$\sum_{j=i}^{n} T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = 1,$$

so the assertion holds for this case also. ■

2.5 Integrals of B-splines

Theorem 2.16 assures that B-splines are continuous functions of their knots. To prove the same for the integrals of B-splines, we need a theorem from the measure theory:

Theorem 2.18 (Lebesgue dominated convergence theorem) *If (f_l) is a sequence of measurable functions on a measure space $(X, \mathcal{M}, \mu)$ such that $\lim_l f_l = f$ (a.e.), and if $|f_l| \leqslant g$ for all l, where g is integrable on X, then*

$$\lim_l \int_X f_l\, d\mu = \int_X f\, d\mu.$$

Proof: See [4]. ■

Theorem 2.19 *Let*

$$\mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}(t_i, \ldots, t_{i+k-1}) := C^{k-1}_{(i,d\boldsymbol{\sigma},\boldsymbol{T})},$$

for $i = 2, \ldots, n$. If $t_{i,\delta}, \ldots, t_{i+k-1,\delta}$ is a sequence of points satisfying $t_{j,\delta} \to t_j^+$, $j = i, \ldots, i+k-1$, when $\delta \to 0$, then

$$\mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}(t_{i,\delta}, \ldots, t_{i+k-1,\delta}) \longrightarrow \mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}(t_i, \ldots, t_{i+k-1})$$

when $\delta \to 0$.

Proof: The proof is a direct consequence of Theorem 2.18. First, let us fix $i = 2, \ldots, n$, let $a \leqslant t_i^l \leqslant \cdots \leqslant t_{i+k-1}^l \leqslant b$ and $a \leqslant t_i \leqslant \cdots \leqslant t_{i+k-1} \leqslant b$ such that

$$\lim_l (t_i^l, \ldots, t_{i+k-1}^l) = (t_i^+, \ldots, t_{i+k-1}^+).$$

We have to check that

$$\int_{t_i}^{t_{i+k-1}} \mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(\tau_2; t_i^l, \ldots, t_{i+k-1}^l)\, d\sigma_2(\tau_2) \to \int_{t_i}^{t_{i+k-1}} \mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(\tau_2; t_i, \ldots, t_{i+k-1})\, d\sigma_2(\tau_2), \tag{2.37}$$

when $(t_i^l, \ldots, t_{i+k-1}^l) \to (t_i^+, \ldots, t_{i+k-1}^+)$ for $\mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}$ defined in (2.26), because

$$\mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}(t_i^l, \ldots, t_{i+k-1}^l) = \int_{t_i}^{t_{i+k-1}} \mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(\tau_2; t_i^l, \ldots, t_{i+k-1}^l)\, d\sigma_2(\tau_2).$$

Let

$$f_l(x) := \mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(x; t_i^l, \ldots, t_{i+k-1}^l)$$

for $x \in [a, b]$. We have to check if the conditions of Theorem 2.18 are fulfilled. Because of Theorem 2.13 we have

$$|f_l(x)| \leqslant 1 =: g(x)$$

for $x \in [a, b]$ and each l, and from Theorem 2.16 we get

$$\lim_l f_l(x) = f(x) := \mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(x; t_i, \ldots, t_{i+k-1}),$$

for each $x \in [a, b]$, so (2.37) holds. ■

Here, again, we apply the same arguments for using the notation $\mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}$ as in Remark 2.6.

Remark 2.7 *It can be easily shown that* $\mathsf{T}^{k-1}_{i,d\boldsymbol{\sigma}^{(1)}}(x; t_i, \ldots, t_{i+k-1})$ *and* $\mathsf{C}^{k-1}_{i,d\boldsymbol{\sigma}}(t_i, \ldots, t_{i+k-1})$ *are continuous functions of their knots, if each* σ_j *is a continuous function on* $[a, b]$ *for* $j = 2, \ldots, k$.

Because of Theorem 2.19 and Remark 2.7, it will be sufficient to calculate only the integrals of B-splines associated with the extended partition with all interior knots of multiplicity one for which we will usually have explicit formulæ. For all other cases of extended partitions, we just take the limits of the knots.

Chapter 3

Knot insertion

There are a lot of algorithms for calculating with polynomial splines based on knot insertion [8]. Our attempt is to generalize some of these algorithms to a general class of Chebyshev splines. In this chapter we introduce the knot insertion for Chebyshev splines, and algorithms based on it [34, 36]. This topic has already been elaborated in [21, 22, 23, 27, 24], but from a different point of view. Although geometrical approach they offer is very nice and more theoretical, practical realization, as well as numerical stability, are not discussed. As we will show in this chapter for the general case, and for the special cases of CCC–systems in chapters 4, 5, 6 and 7, we manage to avoid the problems with numerical stability in our algorithms.

In the previous chapter we have introduced a very important equation: the derivative formula (2.27), so the first idea that occurs is to build an algorithm which calculates a spline by recursive use of integral version of the derivative formula (2.35). It is very easy to see that (2.35) can lead to an error caused by catastrophic cancellation for some values of x. To confirm this assertion, we use such an algorithm to calculate the most simplest splines: the polynomial splines.

Example 3.1 *Let us observe Bezier splines of order $k = 8$, with extended partition $\boldsymbol{T}$ where $t_1 = \cdots = t_8 = 0$ and $t_9 = \cdots = t_{16} = 1$. The integrals in (2.35) are calculated by Gauss–Legendre quadrature formula. The Table 3.1 shows relative errors of the values of B-splines $T^8_{(i,d\boldsymbol{\lambda},\boldsymbol{T})}(x)$ for $i = 1, \ldots, 8$, at the points x given at the top of the table. B-splines are calculated with the recursive use of derivative formula, and compared to the de Boor algorithm. According to the table, the biggest error is for $T^8_{(1,d\boldsymbol{\lambda},\boldsymbol{T})}(x)$ when x approaches 1.*

3.1 Knot insertion matrices

Definition 3.1 *Let Δ and $\widetilde{\Delta}$ be arbitrary partitions of $[a, b]$, $\boldsymbol{m}$ and $\widetilde{\boldsymbol{m}}$ their multiplicity vectors, $\boldsymbol{T} := \boldsymbol{T}_{(\Delta,\boldsymbol{m})}$ and $\widetilde{\boldsymbol{T}} := \boldsymbol{T}_{(\widetilde{\Delta},\widetilde{\boldsymbol{m}})}$ associated extended partitions, such that $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$. Let $\boldsymbol{T} = \{t_j\}_{j=1}^{n+k}$ and $\widetilde{\boldsymbol{T}} = \{\tilde{t}_i\}_{i=1}^{m+k}$, with $n \leqslant m$, and let $T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}$, $T^k_{(i,d\boldsymbol{\sigma},\widetilde{\boldsymbol{T}})}$ be B-splines in $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$, $\mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$ respectively. If*

$i\backslash x$	0.75000000	0.87500000	0.93750000	0.96875000	0.98437500
1	-0.7731D-11	0.3783D-08	-0.2012D-06	-0.1621D-04	0.2808D-02
2	0.1516D-12	-0.3409D-09	0.1490D-08	-0.3076D-06	-0.3681D-04
3	0.1853D-12	0.2495D-10	0.3311D-10	0.1323D-07	0.1640D-06
4	-0.1684D-13	-0.7127D-12	0.8389D-11	-0.7683D-10	0.4101D-08
5	-0.7860D-14	-0.1358D-12	-0.1119D-11	-0.9500D-11	-0.1193D-09
6	0.5347D-14	0.3463D-13	0.9157D-13	0.5219D-12	0.2009D-11
7	-0.2673D-14	-0.3393D-14	-0.1682D-14	-0.8596D-14	-0.3905D-14
8	-0.2287D-14	-0.2544D-14	-0.2616D-14	-0.1941D-14	-0.2603D-14

$i\backslash x$	0.99218750	0.99609375	0.99804688	0.99902344
1	0.1250D+00	0.4300D+02	-0.1791D+04	-0.2294D+06
2	-0.2215D-02	-0.9020D-01	-0.1957D-02	0.2893D+03
3	0.1098D-04	-0.6664D-04	-0.4245D-02	-0.5551D+00
4	0.2899D-07	0.1484D-05	0.2532D-04	0.7497D-03
5	-0.9730D-09	-0.7426D-08	-0.7057D-07	-0.7397D-06
6	0.9616D-11	0.2937D-10	0.1420D-09	0.6090D-09
7	-0.7448D-14	-0.4001D-13	-0.8627D-13	-0.1197D-12
8	-0.2815D-14	-0.1940D-14	-0.1913D-14	-0.2236D-14

Table 3.1: Relative error of Bezier splines calculated with recursive use of the derivative formula

$f \in \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$ is given with

$$f(x) = \sum_{j=1}^{n} c_j T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = \sum_{i=1}^{m} d_i T^k_{(i,d\boldsymbol{\sigma},\widetilde{\boldsymbol{T}})}(x),$$

then $m \times n$ matrix $\Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T},\widetilde{\boldsymbol{T}})} = [\gamma^k_{i,j}]^{m,n}_{i=1,j=1}$, such that

$$\mathbf{d} = \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T},\widetilde{\boldsymbol{T}})}\mathbf{c},$$

with $\mathbf{c} := (c_1, \ldots, c_n)^{\mathrm{T}}$ and $\mathbf{d} := (d_1, \ldots, d_m)^{\mathrm{T}}$, is called the knot insertion matrix of order k from $\boldsymbol{T}$ to $\widetilde{\boldsymbol{T}}$.

The next theorem is an alternative definition of the knot insertion matrix:

Theorem 3.1 *Let $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$ as in Definition 3.1, and let $T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}$, $T^k_{(i,d\boldsymbol{\sigma},\widetilde{\boldsymbol{T}})}$ be B-splines from these spline spaces respectively. Then*

$$T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T})}(x) = \sum_{i=1}^{m} \gamma^k_{i,j} T^k_{(i,d\boldsymbol{\sigma},\widetilde{\boldsymbol{T}})}(x), \qquad j = 1, \ldots, n. \tag{3.1}$$

Proof: Follows directly from Definition 3.1 and the linear independence of B-splines (see Theorem 2.12). ■

Theorem 3.2 *The knot insertion matrix is unique.*

Proof: The uniqueness follows from Theorem 3.1 and the linear independence of B-splines. ■

For each row of knot insertion matrix we have:

Theorem 3.3 *The elements of knot insertion matrix* $\Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T},\tilde{\boldsymbol{T}})}$ *satisfy:*

$$\sum_{j=1}^{n} \gamma^k_{i,j} = 1, \qquad i = 1, \dots m.$$

Proof: From Definition 3.1 we have

$$d_i = \sum_{j=1}^{n} c_j \gamma^k_{i,j}, \qquad i = 1, \dots m,$$

and the rest follows from partition of unity (2.25).

3.2 Single knot insertion matrices

Our goal is to calculate explicitly the nontrivial elements of the knot insertion matrix. We start by considering the knot insertion matrices for inserting just one knot.

Let $\boldsymbol{T} = \boldsymbol{T}_{(\Delta,\boldsymbol{m})} = \{t_j\}_{j=1}^{n+k}$ be the extended partition as defined in (2.22), and let $\bar{t} \in (t_i, t_{i+1}) \subset [a, b]$. Let us denote $\bar{\boldsymbol{T}} := \boldsymbol{T} \cup \{\bar{t}\} = \{\bar{t}_j\}_{j=1}^{n+k+1}$ with

$$\bar{t}_j = t_j \text{ for } j = 1, \dots, i, \quad \bar{t}_{i+1} = \bar{t}, \quad \bar{t}_j = t_{j-1} \text{ for } j = i+2, \dots, n+k+1,$$

and let $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$ and $\mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}})$ be the spline spaces associated with $\boldsymbol{T}$ and $\bar{\boldsymbol{T}}$. Then, we will denote the B-splines in these spline spaces, as well as in the reduced spaces, with:

$$T^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}} := T^{k-l}_{(j,d\boldsymbol{\sigma}^{(l)},\boldsymbol{T})}, \qquad \bar{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}} := T^{k-l}_{(j,d\boldsymbol{\sigma}^{(l)},\bar{\boldsymbol{T}})},$$

for $l = 0, \dots, k-1$, and their integrals:

$$C_{k-l}(j) := C^{k-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\boldsymbol{T})} := \int_{t_j}^{t_{j+k-l}} T^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}), \tag{3.2}$$

$$\bar{C}_{k-l}(j) := C^{k-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\bar{\boldsymbol{T}})} := \int_{\bar{t}_j}^{\bar{t}_{j+k-l}} \bar{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}), \tag{3.3}$$

for $l = 1, \dots, k-1$. We will also use new notation for the single knot insertion matrix $\Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\bar{t})} := \Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\bar{\boldsymbol{T}})}$ and for the multiplicity vector

$$\boldsymbol{m}^{(1)} := (1, \dots, 1). \tag{3.4}$$

Theorem 3.4 *Let $\boldsymbol{T} := \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(1)})} = \{t_j\}_{j=1}^{n+k}$ be an extended partition of $[a,b]$ with all interior knots of multiplicity one. Let $U_k = \{1, u_2, \ldots, u_k\}$ be the CCC–system associated with the measure vector $d\boldsymbol{\sigma} := (d\sigma_2, \ldots, d\sigma_k)^{\mathrm{T}}$. For $\bar{t} \in (a,b)$, and i such that $\bar{t} \in (t_i, t_{i+1})$, let $\bar{\boldsymbol{T}} = \boldsymbol{T} \cup \{\bar{t}\}$. Then the nontrivial elements of the knot insertion matrix $\Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\bar{t})} = [\gamma^l_{i,j}]$ of order l from $\boldsymbol{T}$ to $\bar{\boldsymbol{T}}$ are:*

$$\begin{aligned} \gamma^1_{j,j} &= 1 \quad \text{for } j \le i, \\ \gamma^1_{j,j-1} &= 1 \quad \text{for } j \ge i+1, \end{aligned}$$

for $l = 1$, and

$$\begin{aligned} \gamma^l_{j,j} &= 1 && \text{for } j \le i-l+1, \\ \gamma^l_{j,j} &= \gamma^{l-1}_{j,j}\frac{\bar{C}_{l-1}(j)}{C_{l-1}(j)} && \text{for } i-l+2 \le j \le i, \\ \gamma^l_{j,j-1} &= \gamma^{l-1}_{j+1,j}\frac{\bar{C}_{l-1}(j+1)}{C_{l-1}(j)} && \text{for } i-l+2 \le j \le i, \\ \gamma^l_{j,j-1} &= 1 && \text{for } j \ge i+1, \end{aligned} \tag{3.5}$$

for $l = 2, \ldots, k$, where $C_{l-1}(j)$ and $\bar{C}_{l-1}(j)$, for $l = 2, \ldots, k$, are defined as in (3.2) and (3.3). B-splines $T^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}$ and $\bar{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}$ are associated with the extended partitions $\boldsymbol{T}$ and $\bar{\boldsymbol{T}}$, respectively.

Proof: We will prove Theorem 3.4 by induction and Theorem 3.1. The knot insertion matrix of order 1, $\Gamma^1_{(d\boldsymbol{\sigma}^{(k-1)},\boldsymbol{T},\bar{\boldsymbol{T}})}$, is trivial. Let us assume that the theorem holds for $l-1$. By (3.1), let us define

$$\begin{aligned} f(x) &:= T^l_{j,d\boldsymbol{\sigma}^{(k-l)}}(x), \\ g(x) &:= \gamma^{l-1}_{j,j}\frac{\bar{C}_{l-1}(j)}{C_{l-1}(j)}\bar{T}^l_{j,d\boldsymbol{\sigma}^{(k-l)}}(x) + \gamma^{l-1}_{j+2,j+1}\frac{\bar{C}_{l-1}(j+2)}{C_{l-1}(j+1)}\bar{T}^l_{j+1,d\boldsymbol{\sigma}^{(k-l)}}(x). \end{aligned}$$

We need to prove that $f = g$, because from that we directly get (3.5). We will do that by using the fact that two splines are equal if and only if they have equal generalized derivatives and the same value at the same point, what follows from Chebyshev Newton–Leibniz formula (2.4). Obviously, $f(t_j) = g(t_j)$, then by derivative formula (2.27) and the use of $\Gamma^{l-1}_{(d\boldsymbol{\sigma}^{(k-l+1)},\boldsymbol{T},\bar{t})}$, we have

$$\begin{aligned} L_{(1,d\boldsymbol{\sigma}^{(k-l)})}f &= \frac{T^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}}{C_{l-1}(j)} - \frac{T^{l-1}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}}{C_{l-1}(j+1)} \\ &= \frac{1}{C_{l-1}(j)}\left(\gamma^{l-1}_{j,j}\bar{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}} + \gamma^{l-1}_{j+1,j}\bar{T}^{l-1}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}\right) \\ &\quad -\frac{1}{C_{l-1}(j+1)}\left(\gamma^{l-1}_{j+1,j+1}\bar{T}^{l-1}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}} + \gamma^{l-1}_{j+2,j+1}\bar{T}^{l-1}_{j+2,d\boldsymbol{\sigma}^{(k-l+1)}}\right) \\ &= \frac{\gamma^{l-1}_{j,j}}{C_{l-1}(j)}\bar{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}} + \left(\frac{\gamma^{l-1}_{j+1,j}}{C_{l-1}(j)} - \frac{\gamma^{l-1}_{j+1,j+1}}{C_{l-1}(j+1)}\right)\bar{T}^{l-1}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}} \end{aligned}$$

$$
\begin{aligned}
&-\frac{\gamma_{j+2,j+1}^{l-1}}{C_{l-1}(j+1)}\bar{T}_{j+2,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1},\\
L_{(1,d\boldsymbol{\sigma}^{(k-l)})}g &= \gamma_{j,j}^{l-1}\frac{\bar{C}_{l-1}(j)}{C_{l-1}(j)}\left(\frac{\bar{T}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}}{\bar{C}_{l-1}(j)}-\frac{\bar{T}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}}{\bar{C}_{l-1}(j+1)}\right)\\
&\quad+\gamma_{j+2,j+1}^{l-1}\frac{\bar{C}_{l-1}(j+2)}{C_{l-1}(j+1)}\left(\frac{\bar{T}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}}{\bar{C}_{l-1}(j+1)}-\frac{\bar{T}_{j+2,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}}{\bar{C}_{l-1}(j+2)}\right)\\
&= \frac{\gamma_{j,j}^{l-1}}{C_{l-1}(j)}\bar{T}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}+\frac{1}{\bar{C}_{l-1}(j+1)}\left(-\frac{\gamma_{j,j}^{l-1}\bar{C}_{l-1}(j)}{C_{l-1}(j)}\right.\\
&\quad\left.+\frac{\gamma_{j+2,j+1}^{l-1}\bar{C}_{l-1}(j+2)}{C_{l-1}(j+1)}\right)\bar{T}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}-\frac{\gamma_{j+2,j+1}^{l-1}}{C_{l-1}(j+1)}\bar{T}_{j+2,d\boldsymbol{\sigma}^{(k-l+1)}}^{l-1}.
\end{aligned}
$$

If we integrate equation (3.1) and apply it to the knot insertion matrix $\Gamma_{(d\boldsymbol{\sigma}^{(k-l+1)},\boldsymbol{T},\bar{t})}^{l-1}$, we get

$$C_{l-1}(r)=\gamma_{r,r}^{l-1}\bar{C}_{l-1}(r)+\gamma_{r+1,r}^{l-1}\bar{C}_{l-1}(r+1),\qquad r=j,j+1,$$

and therefore $L_{(1,d\boldsymbol{\sigma}^{(k-l)})}f=L_{(1,d\boldsymbol{\sigma}^{(k-l)})}g$. ■

Corollary 3.1 *Let $\boldsymbol{T}=\boldsymbol{T}_{(\Delta,\boldsymbol{m})}$ be an extended partition of $[a,b]$ associated with an arbitrary multiplicity vector $\boldsymbol{m}=(m_1,\ldots,m_l)$, $\bar{t}\in(x_i,x_{i+1})$, and let $\widetilde{\Delta}=\{y_j\}_{j=0}^{M+1}$ where $M:=\sum_{j=1}^{l}m_j$, and*

$$
\begin{aligned}
y_0 &= x_0,\\
y_{s_{j-1}+1} &= x_j,\quad s_0:=0,\ s_j:=\sum_{r=1}^{j}m_r,\\
y_{s_{j-1}+1} &< y_{s_{j-1}+2}<\cdots<y_{s_j}<y_{s_j+1},\\
y_{s_i} &< \bar{t},\\
y_{M+1} &= x_{l+1},
\end{aligned}
$$

for $j=1,\ldots,l$. If $\widetilde{\boldsymbol{T}}:=\boldsymbol{T}_{(\widetilde{\Delta},\boldsymbol{m}^{(1)})}$, then for $l=1,\ldots,k$

$$\Gamma_{(d\boldsymbol{\sigma}^{(k-l)},\widetilde{\boldsymbol{T}},\bar{t})}^{l}\longrightarrow\Gamma_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\bar{t})}^{l},$$

when $y_{s_{j-1}+r}\to x_j^+$ for $j=1,\ldots,l$ and $r=2,\ldots,m_i$.

Proof: This Corollary is a direct consequence of the application of Theorem 2.19 on Theorem 3.4. ■

Remark 3.1 *According to Remark 2.7, single knot insertion matrices are continuous functions of $(t_1,\ldots,t_{n+k})$ if each σ_j is continuous function on $[a,b]$ for $j=2,\ldots,k$.*

Corollary 3.2 *For $\boldsymbol{T}$ as in Corollary 3.1 and $l = 1, \ldots, k$, it is valid that*

$$\lim_{\bar{t} \to t_i^+} \Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)}, \boldsymbol{T}, \bar{t})} = \Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)}, \boldsymbol{T}, t_i)}.$$

Proof: Again, Theorem 2.19 applied to $\Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)}, \boldsymbol{T}, \bar{t})}$ in Corollary 3.1 proves the assertion of Corollary 3.2. ■

Corollaries 3.1, 3.2 and Remark 3.1 show how to calculate the single knot insertion matrix on arbitrary extended partitions.

3.3 Oslo type algorithms

3.3.1 General knot insertion matrices

Next, we are interested in the more general knot insertion matrices, when more than one knot is inserted. Let, again, $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$, with $\boldsymbol{T} = \{t_j\}_{j=1}^{n+k}$ and $\widetilde{\boldsymbol{T}} = \{\tilde{t}_l\}_{l=1}^{m+k}$, where $m = n + r$. Let

$$\boldsymbol{T} = \boldsymbol{T}^{[0]} \subset \boldsymbol{T}^{[1]} \subset \cdots \subset \boldsymbol{T}^{[r-1]} \subset \boldsymbol{T}^{[r]} = \widetilde{\boldsymbol{T}},$$

where $\boldsymbol{T}^{[l]} = \boldsymbol{T}^{[l-1]} \cup \{\bar{t}_l\}$ for some $\bar{t}_l \in \widetilde{\boldsymbol{T}}$. Then, also,

$$\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}^{[1]}) \subset \cdots \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}^{[r-1]}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}}),$$

and we can observe the algorithm in r steps, where in the i^{th} step we insert one knot, forming new extended partition $\boldsymbol{T}^{[i]}$, which stops when we finally get $\widetilde{\boldsymbol{T}}$. Let for $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$ and $T^{k,[l]}_{j,d\boldsymbol{\sigma}} := T^k_{(j,d\boldsymbol{\sigma},\boldsymbol{T}^{[l]})}$

$$f = \sum_{j=1}^{n} c_j^{[0]} T^{k,[0]}_{j,d\boldsymbol{\sigma}} = \sum_{j=1}^{n+1} c_j^{[1]} T^{k,[1]}_{j,d\boldsymbol{\sigma}} = \cdots = \sum_{j=1}^{m} c_j^{[r]} T^{k,[r]}_{j,d\boldsymbol{\sigma}}.$$

Then for $\mathbf{c^{[l]}} := (c_1^{[l]}, \ldots, c_{n+l}^{[l]})^{\mathrm{T}}$ and $\Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[l-1]},\boldsymbol{T}^{[l]})} = [\gamma^{k,[l]}_{i,j}]^{n+l,n+l-1}_{i=1,j=1}$ the results in the previous section imply that

$$\mathbf{c^{[l]}} = \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[l-1]},\boldsymbol{T}^{[l]})} \mathbf{c^{[l-1]}}. \tag{3.6}$$

Applying (3.6) for $l = 1, \ldots, r$ we obtain

$$\mathbf{c^{[r]}} = \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[r-1]},\boldsymbol{T}^{[r]})} \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[r-2]},\boldsymbol{T}^{[r-1]})} \cdots \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[0]},\boldsymbol{T}^{[1]})} \mathbf{c^{[0]}}.$$

The uniqueness of the knot insertion matrices (see Theorem 3.2) finally implies that:

$$\Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T},\widetilde{\boldsymbol{T}})} = \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[r-1]},\boldsymbol{T}^{[r]})} \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[r-2]},\boldsymbol{T}^{[r-1]})} \cdots \Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T}^{[0]},\boldsymbol{T}^{[1]})}. \tag{3.7}$$

Theorem 3.5 *The elements of the knot insertion matrix* $\Gamma^k_{(d\boldsymbol{\sigma},\boldsymbol{T},\widetilde{\boldsymbol{T}})} = [\gamma^k_{i,j}]^{m,n}_{i=1,j=1}$ *satisfy*

$$\gamma^k_{i,j} \geqslant 0 \qquad i = 1,\ldots,m,\ j = 1,\ldots,n.$$

Proof: From Theorem 3.4 and accompanying two Corollaries, every single knot insertion matrix has nonnegative elements, and from (3.7) the same is true for every general knot insertion matrix. ■

Remark 3.2 *Theorem 3.5 and Theorem 3.3 say that elements of a knot insertion matrix in each row make a partition of unity, similarly to B-splines (2.25).*

3.3.2 Recurrence for elements of knot insertion matrices

In the general case, we can also find a different algorithm from the one in the previous subsection, based on multiplying single knot insertion matrices (3.7). Let, again $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$, and let

$$T^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}} := T^{k-l}_{(j,d\boldsymbol{\sigma}^{(l)},\boldsymbol{T})}, \qquad \widetilde{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}} := T^{k-l}_{(j,d\boldsymbol{\sigma}^{(l)},\widetilde{\boldsymbol{T}})},$$

for $l = 0,\ldots,k-1$ be the B-splines associated with $\boldsymbol{T}$ and $\widetilde{\boldsymbol{T}}$,

$$\widetilde{C}_{k-l}(j) := C^{k-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\widetilde{\boldsymbol{T}})} := \int_{\tilde{t}_j}^{\tilde{t}_{j+k-l}} \widetilde{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}).$$

Then, according to Theorem 3.1

$$T^l_{i,d\boldsymbol{\sigma}^{(k-l)}}(x) = \sum_j \gamma^l_{j,i} \widetilde{T}^l_{j,d\boldsymbol{\sigma}^{(k-l)}}(x), \tag{3.8}$$

for $l = 1,\ldots,k$ and $x \in [a,b]$, where $\Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\widetilde{\boldsymbol{T}})} = [\gamma^l_{i,j}]$.

Theorem 3.6 *Let* $U_k = \{1, u_2, \ldots, u_k\}$ *be the CCC–system associated with the measure vector* $d\boldsymbol{\sigma} := (d\sigma_2,\ldots,d\sigma_k)^{\mathrm{T}}$. *Let* $T^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}$, $\widetilde{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}$ *be the B-splines and* $C_{k-l}(j)$, $\widetilde{C}_{k-l}(j)$ *the integrals of B-splines associated with extended partitions* $\boldsymbol{T}$, $\widetilde{\boldsymbol{T}}$, *respectively, where* $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$. *Then*

$$\gamma^l_{j,i} = \gamma^l_{j-1,i} + \widetilde{C}_{l-1}(j)\left(\frac{\gamma^{l-1}_{j,i}}{C_{l-1}(i)} - \frac{\gamma^{l-1}_{j,i+1}}{C_{l-1}(i+1)}\right), \tag{3.9}$$

and

$$\gamma^l_{j,i} = \frac{\sum\limits_{r\leqslant j} \gamma^{l-1}_{r,i}\widetilde{C}_{l-1}(r)}{C_{l-1}(i)} - \frac{\sum\limits_{r\leqslant j} \gamma^{l-1}_{r,i+1}\widetilde{C}_{l-1}(r)}{C_{l-1}(i+1)}, \tag{3.10}$$

with $\Gamma^l_{(d\boldsymbol{\sigma}^{(k-l)},\boldsymbol{T},\widetilde{\boldsymbol{T}})} = [\gamma^l_{i,j}]$ *for* $l = 2,\ldots,k$.

Proof: The proof is based on (3.8), the derivative formula (2.27) and induction. The knot insertion matrix $\Gamma^1_{(d\boldsymbol{\sigma}^{(k-1)},\boldsymbol{T},\widetilde{\boldsymbol{T}})}$ is trivial to calculate since

$$\gamma^1_{i,j} = \begin{cases} 1 & \text{if } \tilde{t}_i < \tilde{t}_{i+1} \text{ and } [\tilde{t}_i, \tilde{t}_{i+1}) \subset [t_j, t_{j+1}), \\ 0 & \text{else.} \end{cases}$$

By using the first general derivative on (3.8), we get

$$\frac{T^{l-1}_{i,d\boldsymbol{\sigma}^{(k-l+1)}}(x)}{C_{l-1}(i)} - \frac{T^{l-1}_{i+1,d\boldsymbol{\sigma}^{(k-l+1)}}(x)}{C_{l-1}(i+1)} = \sum_j \gamma^l_{j,i} \left(\frac{\widetilde{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}(x)}{\widetilde{C}_{l-1}(j)} - \frac{\widetilde{T}^{l-1}_{j+1,d\boldsymbol{\sigma}^{(k-l+1)}}(x)}{\widetilde{C}_{l-1}(j+1)} \right). \tag{3.11}$$

After applying the induction assumption, *i.e.* $\Gamma^{l-1}_{(d\boldsymbol{\sigma}^{(k-l+1)},\boldsymbol{T},\widetilde{\boldsymbol{T}})}$, the left hand side of (3.11) becomes

$$\sum_j \left(\frac{1}{C_{l-1}(i)} \gamma^{l-1}_{j,i} - \frac{1}{C_{l-1}(i+1)} \gamma^{l-1}_{j,i+1} \right) \widetilde{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}(x),$$

while the right hand side of (3.11), after rearranging, equals

$$\sum_j \frac{1}{\widetilde{C}_{l-1}(j)} \left(\gamma^l_{j,i} - \gamma^l_{j-1,i} \right) \widetilde{T}^{l-1}_{j,d\boldsymbol{\sigma}^{(k-l+1)}}(x).$$

Because of the linear independence of the B-splines, we get (3.9), and just by iterating (3.9), we get (3.10). ■

The recursive formula (3.10) for $\gamma^l_{j,i}$ is like a discrete integral version of derivative formula (2.35) for B-splines. As it is shown in this chapter, $\gamma^l_{j,i}$ has some analogous properties as the discrete polynomial splines [6, 39, 8]. One of this properties is the Oslo type algorithm from Theorem 3.6 which is the generalized Oslo algorithm for polynomial splines. The recurrence (3.9) for polynomial splines equals

$$\gamma^l_{j,i} = \gamma^l_{j-1,i} + (\tilde{t}_{j+l-1} - \tilde{t}_j) \left(\frac{\gamma^{l-1}_{j,i}}{t_{i+l-1} - t_i} - \frac{\gamma^{l-1}_{j,i+1}}{t_{i+l} - t_{i+1}} \right),$$

what is a recurrence which follows from the column oriented approach to the Oslo algorithm (see the equation (4.6) on the page 114 in [8]).

Although (3.9) and (3.10), generally, do not have to be numerically stable, there are special cases: for $k = 4$, $l = 3, 4$, $\boldsymbol{T} := \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(1)})}$ and $\widetilde{\boldsymbol{T}} := \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(2)})}$ with

$$\boldsymbol{m}^{(2)} = (2, \ldots, 2), \tag{3.12}$$

$$\begin{array}{ccccccc} \cdots & & t_{i-1} & t_i & t_{i+1} & t_{i+2} & \cdots \\ \hline \cdots & \tilde{t}_{r-3} & & \tilde{t}_{r-1} & \tilde{t}_{r+1} & \tilde{t}_{r+3} & \cdots \\ \cdots & \tilde{t}_{r-2} & & \tilde{t}_r & \tilde{t}_{r+2} & \tilde{t}_{r+4} & \cdots \end{array},$$

where (3.10) can be rearranged to avoid the subtractions.

The following lemma and theorem connect Chebyshev B-splines of orders 3 and 4 with the less smooth ones, which are, usually, easier to calculate.

Lemma 3.1 *Let $T^3_{i,d\boldsymbol{\sigma}^{(1)}} \in \mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \boldsymbol{T})$ be the Chebyshev 3^{rd} order B-spline associated with the multiplicity vector $\boldsymbol{m}^{(1)} = (1, \ldots, 1)^{\mathrm{T}}$, and let us assume that $\widetilde{T}^3_{i,d\boldsymbol{\sigma}^{(1)}} \in \mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \widetilde{\boldsymbol{T}})$ are B-splines associated with multiplicity vector $\boldsymbol{m}^{(2)} = (2, \ldots, 2)$ on the same knot sequence. If $\boldsymbol{T} = \{t_j\}_{j=1}^{n+4}$ and $\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\}_{j=1}^{2n}$, and r is an index such that $t_i = \tilde{t}_r < \tilde{t}_{r+1}$, then for $i = 2, \ldots, n$:*

$$T^3_{i,d\boldsymbol{\sigma}^{(1)}} = \frac{\widetilde{C}_2(r)}{C_2(i)} \widetilde{T}^3_{r,d\boldsymbol{\sigma}^{(1)}} + \widetilde{T}^3_{r+1,d\boldsymbol{\sigma}^{(1)}} + \frac{\widetilde{C}_2(r+3)}{C_2(i+1)} \widetilde{T}^3_{r+2,d\boldsymbol{\sigma}^{(1)}}.$$

Proof: In this case, the elements of the knot insertion matrix $\Gamma^2_{(d\boldsymbol{\sigma}^{(2)}, \boldsymbol{T}, \widetilde{\boldsymbol{T}})}$ can simply be calculated by

$$\gamma^2_{j,i} = \begin{cases} 1 & \text{for } \tilde{t}_{j-1} = \tilde{t}_j = t_i \text{ or } \tilde{t}_j = \tilde{t}_{j+1} = t_{i+1}, \\ 0 & \text{else,} \end{cases}$$

what implies that

$$\begin{aligned} C_2(i) &= \gamma^2_{r,i} \widetilde{C}_2(r) + \gamma^2_{r+1,i} \widetilde{C}_2(r+1) = \widetilde{C}_2(r) + \widetilde{C}_2(r+1), && (3.13) \\ C_2(i+1) &= \gamma^2_{r+2,i+1} \widetilde{C}_2(r+2) + \gamma^2_{r+3,i+1} \widetilde{C}_2(r+3) \\ &= \widetilde{C}_2(r+2) + \widetilde{C}_2(r+3). && (3.14) \end{aligned}$$

The equations (3.13) and (3.14), together with the recurrence (3.10) applied to $k = 4$, $l = 3$ and $\boldsymbol{m} = \boldsymbol{m}^{(2)}$, give

$$\begin{aligned} \gamma^3_{r,i} &= \frac{\gamma^2_{r,i} \widetilde{C}_2(r)}{C_2(i)} = \frac{\widetilde{C}_2(r)}{C_2(i)}, \\ \gamma^3_{r+1,i} &= \frac{\gamma^2_{r,i} \widetilde{C}_2(r) + \gamma^2_{r+1,i} \widetilde{C}_2(r+1)}{C_2(i)} = 1, \\ \gamma^3_{r+2,i} &= \frac{\gamma^2_{r,i} \widetilde{C}_2(r) + \gamma^2_{r+1,i} \widetilde{C}_2(r+1)}{C_2(i)} - \frac{\gamma^2_{r+2,i+1} \widetilde{C}_2(r+2)}{C_2(i+1)} = \\ &= 1 - \frac{\gamma^2_{r+2,i+1} \widetilde{C}_2(r+2)}{C_2(i+1)} = \frac{\widetilde{C}_2(r+3)}{C_2(i+1)}. \end{aligned}$$

■

Theorem 3.7 *Let $T^4_{i,d\boldsymbol{\sigma}} \in \mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T})$, $\widetilde{T}^4_{i,d\boldsymbol{\sigma}} \in \mathcal{S}(4, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$ be associated with the multiplicity vectors $\boldsymbol{m}^{(1)}$, $\boldsymbol{m}^{(2)}$ as in Lemma 3.1. Then there exist positive $\gamma^4_{j,i}$, such that*

$$T^4_{i,d\boldsymbol{\sigma}} = \sum_{j=r}^{r+3} \gamma^4_{j,i} \widetilde{T}^4_{j,d\boldsymbol{\sigma}},$$

where $r = r_i$ satisfies

$$t_i = \tilde{t}_{r_i} < \tilde{t}_{r_i+1},$$

and $\gamma^4_{j,i}$, $j = r, \dots, r+3$ *are determined by the formulæ:*

$$\begin{aligned}
\gamma^4_{r,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r)}{C_3(i)},\\
\gamma^4_{r+1,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1)}{C_3(i)},\\
\gamma^4_{r+2,i} &= \frac{\gamma^3_{r+3,i+1}\widetilde{C}_3(r+3) + \gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{C_3(i+1)},\\
\gamma^4_{r+3,i} &= \frac{\gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{C_3(i+1)}.
\end{aligned}$$

Proof: Again, (3.10) for $k = 4$, $l = 4$ and $\boldsymbol{m} = \boldsymbol{m}^{(2)}$ gives

$$\begin{aligned}
\gamma^4_{r,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r)}{C_3(i)},\\
\gamma^4_{r+1,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1)}{C_3(i)},\\
\gamma^4_{r+2,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1) + \gamma^3_{r+2,i}\widetilde{C}_3(r+2)}{C_3(i)} - \frac{\gamma^3_{r+2,i+1}\widetilde{C}_3(r+2)}{C_3(i+1)}\\
&= 1 - \frac{\gamma^3_{r+2,i+1}\widetilde{C}_3(r+2)}{C_3(i+1)} = \frac{\gamma^3_{r+3,i+1}\widetilde{C}_3(r+3) + \gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{C_3(i+1)},\\
\gamma^4_{r+3,i} &= \frac{\gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1) + \gamma^3_{r+2,i}\widetilde{C}_3(r+2)}{C_3(i)}\\
&\quad - \frac{\gamma^3_{r+2,i+1}\widetilde{C}_3(r+2)\gamma^3_{r+3,i+1}\widetilde{C}_3(r+3)}{C_3(i+1)}\\
&= 1 - \frac{\gamma^3_{r+2,i+1}\widetilde{C}_3(r+2)\gamma^3_{r+3,i+1}\widetilde{C}_3(r+3)}{C_3(i+1)} = \frac{\gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{C_3(i+1)},
\end{aligned}$$

with the help of $\Gamma^3_{(d\boldsymbol{\sigma}^{(1)},\boldsymbol{T},\widetilde{\boldsymbol{T}})}$ and

$$\begin{aligned}
C_3(i) &= \gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1) + \gamma^3_{r+2,i}\widetilde{C}_3(r+2),\\
C_3(i+1) &= \gamma^3_{r+2,i+1}\widetilde{C}_3(r+2) + \gamma^3_{r+3,i+1}\widetilde{C}_3(r+3) + \gamma^3_{r+4,i+1}\widetilde{C}_3(r+4). \qquad \blacksquare
\end{aligned}$$

3.4 Generalized de Boor algorithm

For calculating with polynomial splines, an alternative way to the de Boor–Cox recurrence is the well known de Boor algorithm. It can be shown that the de Boor–Cox type recurrence exists only for polynomial, trigonometric and hyperbolic splines [40], but on the other hand, the de Boor algorithm can be generalized to general

Chebyshev splines. The main idea, as in the polynomial case, is to insert the point at which we want to calculate the value until maximum multiplicity, when the value of the spline is equal to a single de Boor point. As in Section 3.2, we will observe the extended partition with all interior knots of multiplicity one, and the general case can be deduced from Theorem 2.19 and Remark 2.7.

3.4.1 Splines of general order

Let, again, $\boldsymbol{T} = \{t_j\}_{j=1}^{n+k}$ be an extended partition of $[a, b]$ with all interior knots of multiplicity one. Let $\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$ be the spline space of order k associated with $\boldsymbol{T}$, and $f \in \mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T})$. Then

$$f(x) = \sum_{j=1}^{n} c_j T_{j,d\boldsymbol{\sigma}}^k(x), \qquad x \in [a, b]$$

for some de Boor points $c_j \in \mathbb{R}$. We want to calculate the value $f(\bar{t})$ for some $\bar{t} \in [a, b]$, and let index i be such that $\bar{t} \in [t_i, t_{i+1}) \subset [a, b]$. First, we assume that $\bar{t} \neq t_i$, *i.e.* $\bar{\Delta} := \Delta \cup \{\bar{t}\}$. Next, let us define

$$\bar{\boldsymbol{m}}^{[l]} := (\underbrace{1, \ldots, 1}_{i-k}, l, \underbrace{1, \ldots, 1}_{n-i})$$

and $\bar{\boldsymbol{T}}^{[l]} := \boldsymbol{T}_{(\bar{\Delta}, \bar{\boldsymbol{m}}^{[l]})}$ for $l = 1, \ldots, k-1$. The B-splines associated with these extended partitions we denote as

$$\bar{T}_{j,d\boldsymbol{\sigma}}^{k,[l]} := T_{(j,d\boldsymbol{\sigma},\bar{\boldsymbol{T}}^{[l]})}^k, \qquad l = 1, \ldots, k-1.$$

At l^{th} step of the algorithm, we insert $\bar{t}$ into $\bar{\boldsymbol{T}}^{[l-1]}$, with $\bar{\boldsymbol{T}}^{[0]} := \boldsymbol{T}$, for $l = 1, \ldots, k-1$, and use the fact that

$$\mathcal{S}(k, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[1]}) \subset \cdots \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[l-1]}) \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[l]}) \subset \cdots \subset \mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[k-1]}).$$

Then

$$f(\bar{t}) = \sum_{j=1}^{n} c_j T_{j,d\boldsymbol{\sigma}}^k(\bar{t}) = \sum_{j=1}^{n+1} c_j^{[1]} \bar{T}_{j,d\boldsymbol{\sigma}}^{k,[1]}(\bar{t}) = \cdots = \sum_{j=1}^{n+l} c_j^{[l]} \bar{T}_{j,d\boldsymbol{\sigma}}^{k,[l]}(\bar{t})$$
$$= \cdots = \sum_{j=1}^{n+k-1} c_j^{[k-1]} \bar{T}_{j,d\boldsymbol{\sigma}}^{k,[k-1]}(\bar{t}) = c_i^{[k-1]},$$

because $\bar{T}_{i,d\boldsymbol{\sigma}}^{k,[k-1]}$ is the only nontrivial B-spline in $\mathcal{S}(k, d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[k-1]})$ at the point $\bar{t}$. The de Boor points $c_j^{[l]}$ are calculated by using the single knot insertion matrices from Corollary 3.1:

$$c_j^{[l]} = \gamma_{j,j-1}^{k,[l]} c_{j-1}^{[l-1]} + \gamma_{j,j}^{k,[l]} c_j^{[l-1]}, \qquad j = i-k+l+1, \ldots, i, \tag{3.15}$$

for $l = 1, \ldots, k-1$, where $\Gamma^k_{(d\boldsymbol{\sigma}, \bar{\boldsymbol{T}}^{[l-1]}, \bar{t})} = [\gamma^{k,[l]}_{i,j}]^{n+l, n+l-1}_{i=1,j=1}$, and $c^{[0]}_j := c_j$ for $j = 1, \ldots, n$. As in the polynomial case, the recurrence (3.15) can be represented with a triangle scheme, as for order $k = 4$ on Figure 3.1.

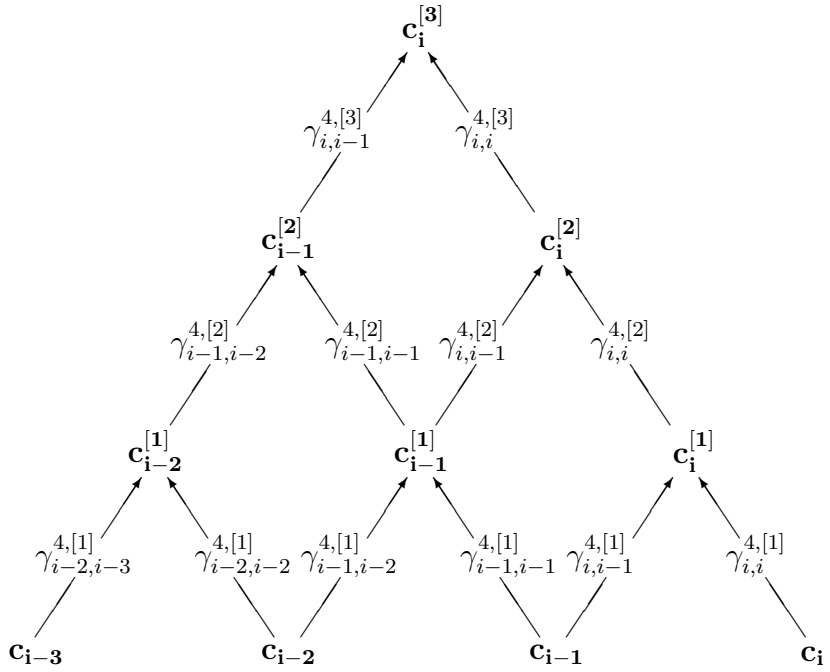

Figure 3.1: The de Boor algorithm for splines of order 4 and $\bar{t} \neq t_i$

Now, if $\bar{t} = t_i$, then $\boldsymbol{T} = \bar{\boldsymbol{T}}^{[1]}$, only this time $\bar{\Delta} = \Delta$ and

$$\bar{\boldsymbol{m}}^{[l]} := (\underbrace{1, \ldots, 1}_{i-k-1}, l, \underbrace{1, \ldots, 1}_{n-i}),$$

for $l = 1, \ldots, k-1$, so, we actually can skip the first step and shift the indices one place to the left:

$$\begin{aligned} f(t_i) &= \sum_{j=1}^{n} c^{[1]}_j \bar{T}^{k,[1]}_{j,d\boldsymbol{\sigma}}(t_i) = \cdots = \sum_{j=1}^{n+l-1} c^{[l]}_j \bar{T}^{k,[l]}_{j,d\boldsymbol{\sigma}}(t_i) = \cdots \\ &= \sum_{j=1}^{n+k-2} c^{[k-1]}_j \bar{T}^{k,[k-1]}_{j,d\boldsymbol{\sigma}}(t_i) = c^{[k-1]}_{i-1}, \end{aligned}$$

where

$$c^{[l]}_j = \gamma^{k,[l]}_{j,j-1} c^{[l-1]}_{j-1} + \gamma^{k,[l]}_{j,j} c^{[l-1]}_j, \qquad j = i-k+l, \ldots, i-1,$$

for $l = 2, \ldots, k-1$ and $\Gamma^k_{(d\boldsymbol{\sigma},\bar{\boldsymbol{T}}^{[l-1]},\bar{t})} := [\gamma^{k,[l]}_{i,j}]^{n+l-1,n+l-2}_{i=1,j=1}$.

We can proceed in the same way even if $\bar{t} = t_i$, and t_i is of multiplicity m_i. In that case we can skip the first m_i steps.

3.4.2 Splines of order 3 and 4

When the order of splines is relatively small, 3 or 4, the generalized de Boor algorithm can be rearranged. If we multiply the single knot insertion matrices (as we have done in the general case in previous section), some of the integrals of B-splines cancel.

First, let us introduce some new notation. Let $\boldsymbol{T}$ be an extended partition with all interior knots of multiplicity one, then, for $\bar{t} \in (t_i, t_{i+1})$, $\bar{\boldsymbol{T}} = \{\bar{t}_j\} := \bar{\boldsymbol{T}}^{[1]}$, $\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\} := \bar{\boldsymbol{T}}^{[2]}$ and $\widehat{\boldsymbol{T}} = \{\hat{t}_j\} := \bar{\boldsymbol{T}}^{[3]}$:

$$
\begin{array}{ccccccc}
\cdots & \bar{t}_{i-1} & \bar{t}_i & \bar{t}_{i+1} & \bar{t}_{i+2} & \bar{t}_{i+3} & \cdots \\
\cdots & t_{i-1} & t_i & \bar{t} & t_{i+1} & t_{i+2} & \cdots \\
\hline
\cdots & \tilde{t}_{i-1} & \tilde{t}_i & \tilde{t}_{i+1} & \tilde{t}_{i+3} & \tilde{t}_{i+4} & \cdots \\
 & & & \tilde{t}_{i+2} & & & \\
\cdots & \hat{t}_{i-1} & \hat{t}_i & \hat{t}_{i+1} & \hat{t}_{i+4} & \hat{t}_{i+5} & \cdots \\
 & & & \hat{t}_{i+2} & & & \\
 & & & \hat{t}_{i+3} & & &
\end{array}
\quad .
$$

The B-splines associated with these extended partitions are denoted by:

$$
\begin{aligned}
T^{4-l}_{j,d\boldsymbol{\sigma}^{(l)}} &:= T^{4-l}_{(j,d\boldsymbol{\sigma}^{(l)},\boldsymbol{T})}, \\
\bar{T}^{4-l}_{j,d\boldsymbol{\sigma}^{(l)}} &:= T^{4-l}_{(j,d\boldsymbol{\sigma}^{(l)},\bar{\boldsymbol{T}})}, \\
\widetilde{T}^{4-l}_{j,d\boldsymbol{\sigma}^{(l)}} &:= T^{4-l}_{(j,d\boldsymbol{\sigma}^{(l)},\widetilde{\boldsymbol{T}})}, \\
\widehat{T}^{4-l}_{j,d\boldsymbol{\sigma}^{(l)}} &:= T^{4-l}_{(j,d\boldsymbol{\sigma}^{(l)},\widehat{\boldsymbol{T}})},
\end{aligned}
$$

for $l = 0, \ldots, 3$. We use the same notation for the integrals of these B-splines:

$$
\begin{aligned}
C_{4-l}(j) := C^{4-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\boldsymbol{T})} &:= \int_{t_j}^{t_{j+k-l}} T^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}), \\
\bar{C}_{4-l}(j) := C^{4-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\bar{\boldsymbol{T}})} &:= \int_{\bar{t}_j}^{\bar{t}_{j+k-l}} \bar{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}), \\
\widetilde{C}_{4-l}(j) := C^{4-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\widetilde{\boldsymbol{T}})} &:= \int_{\tilde{t}_j}^{\tilde{t}_{j+k-l}} \widetilde{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}), \\
\widehat{C}_{4-l}(j) := C^{4-l}_{(j,d\boldsymbol{\sigma}^{(l-1)},\widehat{\boldsymbol{T}})} &:= \int_{\hat{t}_j}^{\hat{t}_{j+k-l}} \widehat{T}^{k-l}_{j,d\boldsymbol{\sigma}^{(l)}}(\tau_{l+1})\, d\sigma_{l+1}(\tau_{l+1}),
\end{aligned}
$$

for $l = 1, \ldots, 3$. Then, the algorithm for $f \in \mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \boldsymbol{T})$ looks like

$$f(\bar{t}) = \sum_{j=2}^{n} c_j T^3_{j,d\boldsymbol{\sigma}^{(1)}}(\bar{t}) = \sum_{j=2}^{n+2} \tilde{c}_j \widetilde{T}^3_{j,d\boldsymbol{\sigma}^{(1)}}(\bar{t}) = \tilde{c}_i,$$

with

$$\begin{aligned}\tilde{c}_i = \frac{\bar{C}_1(i+1)\,\widetilde{C}_2(i+1)}{C_1(i)\,C_2(i-1)}\,c_{i-2} + \left(\frac{\widetilde{C}_2(i+1)\,\bar{C}_2(i-1)}{\bar{C}_2(i)\,C_2(i-1)} + \frac{\widetilde{C}_2(i)\,\bar{C}_2(i+1)}{\bar{C}_2(i)\,C_2(i)}\right) c_{i-1} \\ + \frac{\bar{C}_1(i)\,\widetilde{C}_2(i)}{C_1(i)\,C_2(i)}\,c_i. \qquad (3.16)\end{aligned}$$

The equation above follows from (3.15) for $k = 3$, and

$$\Gamma^2_{(d\boldsymbol{\sigma},\boldsymbol{T},\bar{t})}(i-1:i+1, i-1:i) = \begin{array}{c} \\ i-1 \\ i \\ i+1 \end{array} \begin{array}{c} \begin{array}{cc} i-1 & i \end{array} \\ \begin{pmatrix} 1 & 0 \\ \frac{\bar{C}_1(i+1)}{C_1(i)} & \frac{\bar{C}_1(i)}{C_1(i)} \\ 0 & 1 \end{pmatrix} \end{array}, \qquad (3.17)$$

$$\Gamma^3_{(d\boldsymbol{\sigma},\boldsymbol{T},\bar{t})}(i-2:i+1, i-2:i) = \begin{array}{c} \\ i-2 \\ i-1 \\ i \\ i+1 \end{array} \begin{array}{c} \begin{array}{ccc} i-2 & i-1 & i \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ \gamma^2_{i,i-1}\frac{\bar{C}_2(i)}{C_2(i-1)} & \frac{\bar{C}_2(i-1)}{C_2(i-1)} & 0 \\ 0 & \frac{\bar{C}_2(i+1)}{C_2(i)} & \gamma^2_{i,i}\frac{\bar{C}_2(i)}{C_2(i)} \\ 0 & 0 & 1 \end{pmatrix} \end{array}, \qquad (3.18)$$

and

$$\Gamma^3_{(d\boldsymbol{\sigma},\bar{\boldsymbol{T}},\bar{t})}(i-1:i+1, i-1:i) = \begin{array}{c} \\ i-1 \\ i \\ i+1 \end{array} \begin{array}{c} \begin{array}{cc} i-1 & i \end{array} \\ \begin{pmatrix} 1 & 0 \\ \frac{\widetilde{C}_2(i+1)}{\bar{C}_2(i)} & \frac{\widetilde{C}_2(i)}{\bar{C}_2(i)} \\ 0 & 1 \end{pmatrix} \end{array}. \qquad (3.19)$$

For $\bar{t} = t_i$ we just take $\bar{t} \to t_i^+$, by Corollary 3.2, wherever it appears in the algorithm.

For $f \in \mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T})$ and $\bar{t} \in (t_i, t_{i+1})$, the algorithm becomes:

$$f(\bar{t}) = \sum_{j=1}^{n} c_j T^4_j(\bar{t}) = \sum_{j=1}^{n+3} \hat{c}_j \widehat{T}^4_j(\bar{t}) = \hat{c}_i,$$

with

$$\hat{c}_i = c_{i-3}\frac{\widehat{C}_3(i+1)\widetilde{C}_2(i+1)\bar{C}_1(i+1)}{C_1(i)C_2(i-1)C_3(i-2)} + c_{i-2}\left(\frac{\widehat{C}_3(i+1)\widetilde{C}_2(i+1)\bar{C}_3(i-2)}{\bar{C}_2(i)\bar{C}_3(i-1)C_3(i-2)}\right.$$

$$+\ \frac{\widehat{C}_3(i+1)\widetilde{C}_3(i-1)\bar{C}_2(i+1)\bar{C}_3(i)}{\widetilde{C}_3(i)\bar{C}_3(i-1)C_2(i)C_3(i-1)}+\frac{\widehat{C}_3(i)\widetilde{C}_3(i+1)\bar{C}_2(i+1)}{\widetilde{C}_3(i)C_2(i)C_3(i-1)}\Bigg)$$
$$+\ c_{i-1}\Bigg(\frac{\widehat{C}_3(i+1)\widetilde{C}_3(i-1)\bar{C}_2(i-1)}{\widetilde{C}_3(i)C_2(i-1)C_3(i-1)}+\frac{\widehat{C}_3(i)\widetilde{C}_3(i+1)\bar{C}_2(i-1)\bar{C}_3(i-1)}{\widetilde{C}_3(i)\bar{C}_3(i)C_2(i-1)C_3(i-1)}$$
$$+\ \frac{\widehat{C}_3(i)\widetilde{C}_2(i)\bar{C}_3(i+1)}{\bar{C}_2(i)\bar{C}_3(i)C_3(i)}\Bigg)+c_i\frac{\widehat{C}_3(i)\widetilde{C}_2(i)\bar{C}_1(i)}{C_1(i)C_2(i)C_3(i)}. \tag{3.20}$$

This follows from (3.15), (3.17), (3.18), (3.19) and

$$\Gamma^4=\begin{array}{c} \\ i-3\\ i-2\\ i-1\\ i\\ i+1\end{array}\begin{array}{c}\begin{array}{cccc} i-3 & i-2 & i-1 & i\end{array}\\ \begin{pmatrix} 1 & 0 & 0 & 0\\ \gamma^3_{i-1,i-2}\frac{\bar{C}_3(i-1)}{C_3(i-2)} & \frac{\bar{C}_3(i-2)}{C_3(i-2)} & 0 & 0\\ 0 & \gamma^3_{i,i-1}\frac{\bar{C}_3(i)}{C_3(i-1)} & \gamma^3_{i-1,i-1}\frac{\bar{C}_3(i-1)}{C_3(i-1)} & 0\\ 0 & 0 & \frac{\bar{C}_3(i+1)}{C_3(i)} & \gamma^3_{i,i}\frac{\bar{C}_3(i)}{C_3(i)}\\ 0 & 0 & 0 & 1\end{pmatrix}\end{array},$$

with $\Gamma^4:=\Gamma^4_{(d\boldsymbol{\sigma},\boldsymbol{T},\bar{t})}(i-3:i+1,i-3:i)$,

$$\Gamma^4_{(d\boldsymbol{\sigma},\bar{\boldsymbol{T}},\bar{t})}(i-2:i+1,i-2:i)=\begin{array}{c} \\ i-2\\ i-1\\ i\\ i+1\end{array}\begin{array}{c}\begin{array}{ccc} i-2 & i-1 & i\end{array}\\ \begin{pmatrix} 1 & 0 & 0\\ \bar{\gamma}^3_{i,i-1}\frac{\bar{C}_3(i)}{\bar{C}_3(i-1)} & \frac{\bar{C}_3(i-1)}{\bar{C}_3(i-1)} & 0\\ 0 & \frac{\bar{C}_3(i+1)}{\bar{C}_3(i)} & \bar{\gamma}^3_{i,i}\frac{\bar{C}_3(i)}{\bar{C}_3(i)}\\ 0 & 0 & 1\end{pmatrix}\end{array},$$

and

$$\Gamma^4_{(d\boldsymbol{\sigma},\widetilde{\boldsymbol{T}},\bar{t})}(i-1:i+1,i-1,i)=\begin{array}{c} \\ i-1\\ i\\ i+1\end{array}\begin{array}{c}\begin{array}{cc} i-1 & i\end{array}\\ \begin{pmatrix} 1 & 0\\ \frac{\widehat{C}_3(i+1)}{\widetilde{C}_3(i)} & \frac{\widehat{C}_3(i)}{\widetilde{C}_3(i)}\\ 0 & 1\end{pmatrix}\end{array}.$$

For $\bar{t}=t_i$ we do the same as for $k=3$. If $\boldsymbol{T}$ is a general extended partition, we can use Corollary 3.1 to coalesce the knots in the same way.

Chapter 4

Weighted splines

Weighted splines [7, 2, 33] serve as the first and relatively simple step in generalization of polynomial splines, because they are also piecewise polynomial. We will use the algorithms developed in the previous chapter for calculating with weighted splines. First, we will observe splines of order 4 as the special case, and then weighted splines of general order. Weighted splines are as yet the only nontrivial case of arbitrary order splines that we can evaluate by the knot insertion algorithms.

Let $\boldsymbol{T}$ be, again, the extended partition with all interior knots of multiplicity one: $\boldsymbol{T} = \{t_j\}_{j=1}^{n+k+2}$ and $t_1 \leqslant t_2 \leqslant \cdots \leqslant a = t_{k+2} < t_{k+3} < \cdots < t_n < t_{n+1} = b \leqslant \cdots \leqslant t_{n+k+1} \leqslant t_{n+k+2}$, associated with partition $\Delta = \{t_i\}_{i=k+2}^{n+1}$. If we define a positive function w as $w|_{[t_i,t_{i+1})} :\equiv w_i$, $w_i = \text{const}$ for $i = k+2, \ldots, n-1$ and $w|_{[t_n,t_{n+1}]} :\equiv w_n$, $w_n = \text{const}$, then let $d\boldsymbol{\sigma} := (d\lambda(\tau_2), \frac{d\lambda(\tau_3)}{w(\tau_3)}, \ldots, d\lambda(\tau_{k+2}))^{\mathrm{T}}$, where $d\lambda$ is the Lebesgue measure. Weighted powers of order $k+2$, with $k \geqslant 2$, are:

$$
\begin{aligned}
u_1(x) &= 1, \\
u_2(x) &= \int_a^x d\tau_2, \\
u_3(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \frac{d\tau_3}{w(\tau_3)}, \\
&\vdots \\
u_{k+2}(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \frac{d\tau_3}{w(\tau_3)} \int_a^{\tau_3} d\tau_4 \cdots \int_a^{\tau_{k+1}} d\tau_{k+2},
\end{aligned}
$$

while the first reduced CCC–system is:

$$
\begin{aligned}
u_{1,1}(x) &= 1, \\
u_{1,2}(x) &= \int_a^x \frac{d\tau_3}{w(\tau_3)}, \\
&\vdots \\
u_{1,k+1}(x) &= \int_a^x \frac{d\tau_3}{w(\tau_3)} \int_a^{\tau_3} d\tau_4 \cdots \int_a^{\tau_{k+1}} d\tau_{k+2}.
\end{aligned}
$$

Further, we need generalized derivatives:

$$\begin{aligned} L_{(0,d\boldsymbol{\sigma})}f(x) &= f(x),\\ L_{(1,d\boldsymbol{\sigma})}f(x) &= Df(x),\\ L_{(2,d\boldsymbol{\sigma})}f(x) &= w(x)D^2f(x),\\ L_{(3,d\boldsymbol{\sigma})}f(x) &= Dw(x)D^2f(x),\\ &\vdots\\ L_{(k+2,d\boldsymbol{\sigma})}f(x) &= D^kw(x)D^2f(x), \end{aligned}$$

where for $L_{(k+2,d\boldsymbol{\sigma})}$, the measure by which we extend the CCC–system is again the Lebesgue measure, and also generalized derivative with respect to the first reduced system:

$$\begin{aligned} L_{(0,d\boldsymbol{\sigma}^{(1)})}f(x) &= f(x),\\ L_{(1,d\boldsymbol{\sigma}^{(1)})}f(x) &= w(x)Df(x),\\ L_{(2,d\boldsymbol{\sigma}^{(1)})}f(x) &= Dw(x)Df(x),\\ &\vdots\\ L_{(k+1,d\boldsymbol{\sigma}^{(1)})}f(x) &= D^kw(x)Df(x). \end{aligned}$$

This produces few important properties:

$$\begin{aligned} \mathcal{S}(k+2,d\boldsymbol{\sigma}) \subseteq \operatorname{Ker} L_{(k+2,d\boldsymbol{\sigma})} = \operatorname{Ker} D^kwD^2,\\ \mathcal{S}(k+1,d\boldsymbol{\sigma}^{(1)}) \subseteq \operatorname{Ker} L_{(k+1,d\boldsymbol{\sigma}^{(1)})} = \operatorname{Ker} D^kwD, \end{aligned}$$

weighted splines are C^1 polynomial splines of order $k+2$, splines associated with the first reduced system are C^0 polynomial splines of order $k+1$, and splines associated with the j^{th} reduced system are ordinary polynomial splines of order $k+2-j$ for $j = 2,\dots,k+1$. Therefore $L_{(j,d\boldsymbol{\sigma})} : \mathcal{S}(k+2,d\boldsymbol{\sigma}) \to \mathcal{P}_{k+2-j}$ (see Remark 2.1) for $j \geqslant 2$.

4.1 Weighted splines of order 4 ($k = 2$)

For $k = 2$, $d\boldsymbol{\sigma} = (d\lambda(\tau_2), \frac{d\lambda(\tau_3)}{w(\tau_3)}, d\lambda(\tau_4))^{\mathrm{T}}$. Let $\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\}_{l=1}^{2n} = \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(2)})}$ with $m^{(2)} = (2,\dots,2)$, as in section 3.3.2. Let $d\boldsymbol{\lambda} := (d\lambda(\tau_2), d\lambda(\tau_3), d\lambda(\tau_4))^{\mathrm{T}}$ and

$$\begin{aligned} T_j^l &:= T^l_{(j,d\boldsymbol{\sigma}^{(4-l)},\boldsymbol{T})}, \qquad & \widetilde{T}_j^l &:= T^l_{(j,d\boldsymbol{\sigma}^{(4-l)},\widetilde{\boldsymbol{T}})},\\ B_j^l &:= T^l_{(j,d\boldsymbol{\lambda}^{(4-l)},\boldsymbol{T})}, \qquad & \widetilde{B}_j^l &:= T^l_{(j,d\boldsymbol{\lambda}^{(4-l)},\widetilde{\boldsymbol{T}})}, \end{aligned}$$

for $l = 1,\dots,4$, then obviously $\widetilde{T}_j^4 = \widetilde{B}_j^4$, $\widetilde{T}_j^3 = \widetilde{B}_j^3$, $\widetilde{T}_j^2 = \widetilde{B}_j^2$, and $T_j^2 = B_j^2$. The integrals of B-splines are as follows:

$$\widetilde{C}_2(r-1) = \int_{\tilde{t}_{r-1}}^{\tilde{t}_{r+1}} \widetilde{T}_{r-1}^2(\tau_3)\frac{d\tau_3}{w(\tau_3)} = \frac{h_i}{2\,w_i},$$

$$\widetilde{C}_2(r) = \int_{\tilde{t}_r}^{\tilde{t}_{r+2}} \widetilde{T}_r^2(\tau_3)\frac{d\tau_3}{w(\tau_3)} = \frac{h_i}{2\,w_i},$$
$$C_2(i) = \int_{t_i}^{t_{i+2}} T_i^2(\tau_3)\frac{d\tau_3}{w(\tau_3)} = \widetilde{C}_2(r) + \widetilde{C}_2(r+1) = \frac{1}{2}\left(\frac{h_i}{w_i} + \frac{h_{i+1}}{w_{i+1}}\right),$$
$$\widetilde{C}_3(r-1) = \int_{\tilde{t}_{r-1}}^{\tilde{t}_{r+2}} \widetilde{T}_{r-1}^3(\tau_2)\,d\tau_2 = \frac{h_i}{3},$$
$$\widetilde{C}_3(r) = \int_{\tilde{t}_r}^{\tilde{t}_{r+3}} \widetilde{T}_r^3(\tau_2)\,d\tau_2 = \frac{h_i + h_{i+1}}{3},$$

and by Lemma 3.1

$$\begin{aligned} C_3(i) &= \int_{t_i}^{t_{i+3}} T_i^3(\tau_2)\,d\tau_2 \\ &= \frac{\widetilde{C}_2(r)}{C_2(i)}\,\widetilde{C}_3(r) + \widetilde{C}_3(r+1) + \frac{\widetilde{C}_2(r+3)}{C_2(i+1)}\,\widetilde{C}_3(r+2) \\ &= \frac{1}{3}\left(\frac{\dfrac{h_i}{w_i}(h_i + h_{i+1})}{\dfrac{h_i}{w_i} + \dfrac{h_{i+1}}{w_{i+1}}} + h_{i+1} + \frac{\dfrac{h_{i+2}}{w_{i+2}}(h_{i+1} + h_{i+2})}{\dfrac{h_{i+1}}{w_{i+1}} + \dfrac{h_{i+2}}{w_{i+2}}}\right), \end{aligned} \tag{4.1}$$

with $h_i := t_{i+1} - t_i$ and $t_i = \tilde{t}_r < \tilde{t}_{r+1}$. Now, from (4.1) we can calculate $C_3(i)$ and $C_3(i+1)$, and further, we can get all four coefficients $\gamma_{j,i}^4$ from Theorem 3.7. Finally, to obtain B-splines T_j^4 according to Theorem 3.7, we only need $\widetilde{T}_j^4$. As we mentioned before, $\mathcal{S}(4, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}}) = \mathcal{S}(4, d\boldsymbol{\lambda}, \widetilde{\boldsymbol{T}})$, *i.e.* $\widetilde{T}_j^4 = \widetilde{B}_j^4$, and the polynomial splines of order 4 are easy to calculate.

4.2 Weighted splines of order $k+2$ $(k > 2)$

For the general case we use the generalized de Boor algorithm from Section 3.4. As the single knot insertion matrices have the main role in this algorithm, with nontrivial elements consisting of quotients of integrals of B-splines, the problem of calculating the value of a spline becomes the problem of calculating the integrals of B-splines associated with the reduced systems. The simplest idea is to calculate these integrals by Gauss–Legendre integration, but the complexity of such an algorithm would be too large, and if, for example, we would like to make a interactive program in which we change parameters and check the results, the algorithm would be too slow. Fortunately, there are two ways of calculating these integrals that perform better.

Let us introduce some new notation. Like in the previous section, let $d\boldsymbol{\lambda} := (\underbrace{d\lambda, \ldots, d\lambda}_{k+1})^{\mathrm{T}}$,

$$T_j^l := T^l_{(j, d\boldsymbol{\sigma}^{(k+2-l)}, \boldsymbol{T})}, \qquad B_j^l := T^l_{(j, d\boldsymbol{\lambda}^{(k+2-l)}, \boldsymbol{T})},$$

for $l = 1, \ldots, k+2$, and

$$\begin{aligned} C_{k+1}(j) &= \int_{t_j}^{t_{j+k+1}} T_j^{k+1}(\tau_2)\, d\tau_2, \quad C_k(j) = \int_{t_j}^{t_{j+k}} T_j^k(\tau_3) \frac{d\tau_3}{w(\tau_3)}, \\ C_l(j) &= \int_{t_j}^{t_{j+l}} T_j^l(\tau_{k+3-l})\, d\tau_{k+3-l} \quad \text{for } l = 1, \ldots, k-1. \end{aligned} \tag{4.2}$$

4.2.1 Recurrence for integrals of polynomial B-splines

For weighted splines $T_j^k = B_j^k$, in fact

$$C_k(j) = \int_{t_j}^{t_{j+k}} B_j^k(\tau_3) \frac{d\tau_3}{w(\tau_3)}.$$

We want to find a recurrence for

$$\int_{t_i}^{x} B_i^k(\tau) \frac{d\tau}{w(\tau)}, \qquad x \in [t_i, t_{i+k}],$$

and

$$\int_{t_j}^{t_{j+1}} B_i^k(\tau) \frac{d\tau}{w(\tau)}, \qquad j = i, \ldots, i+k-1,$$

(see [35]). Let $x \in [t_j, t_{j+1})$, then

$$\begin{aligned} \int_{t_i}^{x} B_i^k(\tau) \frac{d\tau}{w(\tau)} &= \sum_{s=i}^{j-1} \int_{t_s}^{t_{s+1}} B_i^k(\tau) \frac{1}{w_s}\, d\tau + \frac{1}{w_j} \int_{t_j}^{x} B_i^k(\tau)\, d\tau \\ &= \sum_{s=i}^{j-1} \frac{1}{w_s} \left(\int_{t_i}^{t_{s+1}} B_i^k(\tau)\, d\tau - \int_{t_i}^{t_s} B_i^k(\tau)\, d\tau \right) \\ &\quad + \frac{1}{w_j} \left(\int_{t_i}^{x} B_i^k(\tau)\, d\tau - \int_{t_i}^{t_j} B_i^k(\tau)\, d\tau \right) \\ &= \sum_{s=i}^{j-1} \frac{1}{w_s} \frac{t_{i+k} - t_i}{k} \left(\sum_{r=i}^{s} B_r^{k+1}(t_{s+1}) - \sum_{r=i}^{s-1} B_r^{k+1}(t_s) \right) \\ &\quad + \frac{1}{w_j} \frac{t_{i+k} - t_i}{k} \left(\sum_{r=i}^{j} B_r^{k+1}(x) - \sum_{r=i}^{j-1} B_r^{k+1}(t_j) \right), \end{aligned} \tag{4.3}$$

from the well known formula for integrals of polynomial splines. Let us define

$$\bar{\alpha}_{i,j+1}^{k+1}(x) := \sum_{r=i}^{j} B_r^{k+1}(x) \quad \text{and} \quad \alpha_{i,j+1}^{k+1} := \bar{\alpha}_{i,j+1}^{k+1}(t_{j+1}). \tag{4.4}$$

Then (4.3) becomes

$$\int_{t_i}^{x} B_i^k(\tau) \frac{d\tau}{w(\tau)} = \frac{t_{i+k} - t_i}{k} \left(\sum_{s=i}^{j-1} \frac{1}{w_s} \Big(\alpha_{i,s+1}^{k+1} - \alpha_{i,s}^{k+1} \Big) + \frac{1}{w_j} \Big(\bar{\alpha}_{i,j+1}^{k+1}(x) - \alpha_{i,j}^{k+1} \Big) \right). \tag{4.5}$$

Next, we derive a recurrence for $\bar{\alpha}_{i,j+1}^{k+1}(x)$. By the de Boor–Cox recurrence

$$\begin{aligned}
\sum_{r=i}^{j} B_r^{k+1}(x) &= \sum_{r=i}^{j} \left(\frac{x-t_r}{t_{r+k}-t_r} B_r^k(x) + \frac{t_{r+k+1}-x}{t_{r+k+1}-t_{r+1}} B_{r+1}^k(x) \right) \\
&= \sum_{r=i}^{j} \frac{x-t_r}{t_{r+k}-t_r} B_r^k(x) + \sum_{r=i}^{j} B_{r+1}^k(x) - \sum_{r=i}^{j} \frac{x-t_{r+1}}{t_{r+k+1}-t_{r+1}} B_{r+1}^k(x) \\
&= \sum_{r=i+1}^{j} \left(\frac{x-t_r}{t_{r+k}-t_r} - \frac{x-t_r}{t_{r+k}-t_r} \right) B_r^k(x) + \frac{x-t_i}{t_{i+k}-t_i} B_i^k(x) + \sum_{r=i}^{j-1} B_{r+1}^k(x) \\
&= \frac{x-t_i}{t_{i+k}-t_i} B_i^k(x) + \sum_{r=i+1}^{j} B_r^k(x) \\
&= \frac{x-t_i}{t_{i+k}-t_i} B_i^k(x) + \bar{\alpha}_{i+1,j+1}^k(x),
\end{aligned}$$

because $B_{j+1}^k(x) = 0$, so we get the recurrence:

$$\bar{\alpha}_{i,j+1}^{k+1}(x) = \frac{x-t_i}{t_{i+k}-t_i} B_i^k(x) + \bar{\alpha}_{i+1,j+1}^k(x), \tag{4.6}$$

for $x \in [t_j, t_{j+1})$ and $j = i, \ldots, i+k-1$. Obviously

$$\begin{aligned}
\bar{\alpha}_{i,j+1}^k(x) &= \sum_{r=i}^{j} B_r^k(x) = B_i^k(x) + \sum_{r=i+1}^{j} B_r^k(x) \\
&= B_i^k(x) + \bar{\alpha}_{i+1,j+1}^k(x),
\end{aligned}$$

and from here $B_i^k(x) = \bar{\alpha}_{i,j+1}^k(x) - \bar{\alpha}_{i+1,j+1}^k(x)$, which applied to (4.6) gives

$$\begin{aligned}
\bar{\alpha}_{i,j+1}^{k+1}(x) &= \frac{x-t_i}{t_{i+k}-t_i} \Big(\bar{\alpha}_{i,j+1}^k(x) - \bar{\alpha}_{i+1,j+1}^k(x) \Big) + \bar{\alpha}_{i+1,j+1}^k(x) \\
&= \frac{x-t_i}{t_{i+k}-t_i} \bar{\alpha}_{i,j+1}^k(x) + \bar{\alpha}_{i+1,j+1}^k(x) \left(1 - \frac{x-t_i}{t_{i+k}-t_i} \right).
\end{aligned}$$

Finally, we have the recurrence

$$\bar{\alpha}_{i,j+1}^{k+1}(x) = \frac{x-t_i}{t_{i+k}-t_i}\, \bar{\alpha}_{i,j+1}^k(x) + \frac{t_{i+k}-x}{t_{i+k}-t_i}\, \bar{\alpha}_{i+1,j+1}^k(x), \tag{4.7}$$

for $x \in [t_j, t_{j+1})$ and $j = i, \ldots, i+k-1$.

Let us first calculate

$$\frac{1}{w_j} \frac{t_{i+k}-t_i}{k} \left(\sum_{r=i}^{j} B_r^{k+1}(x) - \sum_{r=i}^{j-1} B_r^{k+1}(t_j) \right) = \frac{t_{i+k}-t_i}{k\, w_j} \Big(\bar{\alpha}_{i,j+1}^{k+1}(x) - \alpha_{i,j}^{k+1} \Big).$$

If we define

$$\bar{\delta}_{i,j}^{k+1}(x) := \bar{\alpha}_{i,j+1}^{k+1}(x) - \alpha_{i,j}^{k+1},$$

then from (4.7), we have

$$\begin{aligned}\bar{\delta}_{i,j}^{k+1}(x) &= \frac{x-t_i}{t_{i+k}-t_i}\,\bar{\alpha}_{i,j+1}^{k}(x) + \frac{t_{i+k}-x}{t_{i+k}-t_i}\,\bar{\alpha}_{i+1,j+1}^{k}(x) - \frac{t_j-t_i}{t_{i+k}-t_i}\,\alpha_{i,j}^{k} - \frac{t_{i+k}-t_j}{t_{i+k}-t_i}\,\alpha_{i+1,j}^{k} \\ &= \frac{t_j-t_i}{t_{i+k}-t_i}\,\bar{\delta}_{i,j}^{k}(x) + \frac{t_{i+k}-x}{t_{i+k}-t_i}\,\bar{\delta}_{i+1,j}^{k}(x) + \frac{x-t_j}{t_{i+k}-t_i}\Big(\bar{\alpha}_{i,j+1}^{k}(x) - \alpha_{i+1,j}^{k}\Big). \quad (4.8)\end{aligned}$$

Further,

$$\begin{aligned}\bar{\alpha}_{i,j+1}^{k}(x) - \alpha_{i+1,j}^{k} &= \bar{\alpha}_{i,j+1}^{k}(x) - \bar{\alpha}_{i+1,j+1}^{k}(x) + \bar{\alpha}_{i+1,j+1}^{k}(x) - \alpha_{i+1,j}^{k} \\ &= \bar{\alpha}_{i,j+1}^{k}(x) - \bar{\alpha}_{i+1,j+1}^{k}(x) + \bar{\delta}_{i+1,j}^{k}(x) \\ &= \sum_{r=i}^{j} B_r^k(x) - \sum_{r=i+1}^{j} B_r^k(x) + \bar{\delta}_{i+1,j}^{k}(x) \\ &= B_i^k(x) + \bar{\delta}_{i+1,j}^{k}(x), \quad (4.9)\end{aligned}$$

which follows from the definition (4.4). Applying (4.9) to (4.8), we get

$$\bar{\delta}_{i,j}^{k+1}(x) = \frac{t_j-t_i}{t_{i+k}-t_i}\,\bar{\delta}_{i,j}^{k}(x) + \frac{t_{i+k}-t_j}{t_{i+k}-t_i}\,\bar{\delta}_{i+1,j}^{k}(x) + \frac{x-t_j}{t_{i+k}-t_i}\,B_i^k(x), \quad (4.10)$$

for $x \in [t_j, t_{j+1})$ and $j = i, \ldots, i+k-1$. Finally, from (4.5) we have

$$\frac{k}{t_{i+k}-t_i}\int_{t_i}^{x} B_i^k(\tau)\frac{d\tau}{w(\tau)} = \sum_{s=i}^{j-1}\frac{\delta_{i,s}^{k+1}}{w_s} + \frac{1}{w_j}\,\bar{\delta}_{i,j}^{k+1}(x), \quad (4.11)$$

with

$$\delta_{i,s}^{k+1} := \bar{\delta}_{i,s}^{k+1}(t_{s+1}),$$

$x \in [t_j, t_{j+1})$ and $j = i, \ldots, i+k-1$. Specially,

$$\frac{k}{t_{i+k}-t_i}\int_{t_i}^{t_{i+k}} B_i^k(\tau)\frac{d\tau}{w(\tau)} = \sum_{s=i}^{i+k-1}\frac{\delta_{i,s}^{k+1}}{w_s},$$

and by (4.11)

$$\begin{aligned}\frac{k}{t_{i+k}-t_i}\int_{t_j}^{t_{j+1}} B_i^k(\tau)d\tau &= \frac{k}{t_{i+k}-t_i}\,w_j\left(\int_{t_i}^{t_{j+1}} B_i^k(\tau)\frac{d\tau}{w(\tau)}\right. \\ &\quad \left. - \int_{t_i}^{t_j} B_i^k(\tau)\frac{d\tau}{w(\tau)}\right) = \delta_{i,j}^{k+1},\end{aligned}$$

where $\delta_{i,j}^{k+1}$ is calculated, because of (4.10), by the recurrence

$$\begin{aligned}\delta_{i,j}^{2} &= \begin{cases} 1 \text{ for } j = i, \\ 0 \text{ for } j \neq i, \end{cases} \\ \delta_{i,j}^{k+1} &= \frac{t_j-t_i}{t_{i+k}-t_i}\,\delta_{i,j}^{k} + \frac{t_{i+k}-t_j}{t_{i+k}-t_i}\,\delta_{i+1,j}^{k} + \frac{t_{j+1}-t_j}{t_{i+k}-t_i}\,B_i^k(t_{j+1}),\end{aligned}$$

for $j = i, \ldots, i+k-1$ (see also [41]). This is one way how to calculate $C_k(j)$ from (4.2).

4.2.2 Generalized Oslo algorithm for calculating integrals of B-splines

As we emphasized before, the splines associated with the second reduced system are polynomial splines of order k. But, for $C_k(j)$ in (4.2), we need to calculate the integral with respect to the Lebesgue–Stieltjes measure, which has piecewise constant density. One possibility is to represent T_j^k as a linear combination of B-splines from the space of weighted splines which are not even continuous. These B-splines are one–interval supported. The integrals of such B-splines with given measure are trivial to calculate. It is also important to mention that all knot insertion matrices up to order k are polynomial knot insertion matrices. This fact makes the calculation of the single knot insertion matrices, by the recurrence in Theorem 3.4, much easier.

To be precise, for this case we observe $\boldsymbol{T}$ with $t_1 = t_2 = \cdots = t_{k+2} = a$ and $b = t_{n+1} = \cdots = t_{n+k+1} = t_{n+k+2}$, then, let $\boldsymbol{m}^{(k)} := (k, \ldots, k)$, $\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\}_{j=1}^{k(n-k)+4} := \boldsymbol{T}_{(\Delta, \boldsymbol{m}^{(k)})}$, and let $\tilde{t}_j = t_r < t_{r+1} = \tilde{t}_{j+k}$. Thereupon

$$\widetilde{C}_k(j) := \int_{\tilde{t}_j}^{\tilde{t}_{j+k}} \widetilde{T}_j^k(\tau) \frac{d\tau}{w(\tau)} = \frac{t_{r+1} - t_r}{w_r\, k},$$

where $\widetilde{T}_j^k := T^k_{(j, d\boldsymbol{\sigma}^{(2)}, \widetilde{\boldsymbol{T}})}$. Let

$$T_j^k(x) = \sum_{i=k(j-k-1)+2}^{k(j-3)+3} \gamma_{i,j}^k \widetilde{T}_i^k(x),$$

then the knot insertion matrix $\Gamma^k := \Gamma^k_{(d\boldsymbol{\sigma}^{(2)}, \boldsymbol{T}, \widetilde{\boldsymbol{T}})} = \Gamma^k_{(d\boldsymbol{\lambda}^{(2)}, \boldsymbol{T}, \widetilde{\boldsymbol{T}})} = [\gamma_{i,j}^k]$, which is a polynomial knot insertion matrix, can be obtained explicitly (see pages 159, 160 in [8]). So, from previous

$$C_k(j) = \sum_{i=k(j-k-1)+2}^{k(j-3)+3} \gamma_{i,j}^k \, \frac{t_{r_i+1} - t_{r_i}}{w_{r_i}\, k},$$

with $\tilde{t}_i = t_{r_i} < t_{r_i+1} = \tilde{t}_{i+k}$.

Now, when we know how to calculate the integrals of B-spline associated with the second reduced system, we can also calculate the knot insertion matrices of order $k+1$. Let us remember that the splines from $\mathcal{S}(k+1, d\boldsymbol{\sigma}^{(1)}, \boldsymbol{T})$ are C^0 polynomial splines of order $k+1$, and it is obvious that $\mathcal{S}(k+1, d\boldsymbol{\sigma}^{(1)}, \widetilde{\boldsymbol{T}}) = \mathcal{S}(k+1, d\boldsymbol{\lambda}^{(1)}, \widetilde{\boldsymbol{T}})$. For B-spline from $\mathcal{S}(k+1, d\boldsymbol{\sigma}^{(1)}, \widetilde{\boldsymbol{T}})$ we have

$$\widetilde{C}_{k+1}(j) := \int_{\tilde{t}_j}^{\tilde{t}_{j+k+1}} \widetilde{T}_j^{k+1} d\tau = \frac{\tilde{t}_{j+k+1} - \tilde{t}_j}{k+1},$$

then

$$C_{k+1}(j) = \sum_{i=k(j-k-1)+2}^{k(j-2)+2} \gamma_{i,j}^{k+1} \, \frac{\tilde{t}_{i+k+1} - \tilde{t}_i}{k+1},$$

with $\Gamma^{k+1} := \Gamma^{k+1}_{(d\boldsymbol{\sigma}^{(1)},\boldsymbol{T},\bar{\boldsymbol{T}})} = [\gamma^{k+1}_{i,j}]$ being again calculated as the product of the single knot insertion matrices of order $k+1$.

This algorithm is a generalization of the Oslo algorithm for polynomial splines. As in polynomial case, the generalized Oslo algorithm can be represented with a triangle scheme, and it can also be accelerated by avoiding operations where multiplication by zero or addition of zero occurs.

This generalized Oslo algorithm for calculating $C_{k+1}(j)$ can be combined either with the recurrence for calculating $C_k(j)$ from the previous subsection, or with the generalized Oslo algorithm for $C_k(j)$. The rest of the integrals of B-splines associated with the partition with multiplicities higher than one, which we need for the de Boor algorithm, can then be achieved by Theorem 2.19 or Remark 2.7.

Let now $\bar{t} \in [t_i, t_{i+1}) \subset [a,b]$ and $f \in \mathcal{S}(k+2, d\boldsymbol{\sigma}, \boldsymbol{T})$ with $f = \sum_{i=1}^{n} c_j T_j^{k+2}$. Our goal is to calculate $f(\bar{t})$. Having $C_k(j)$ and $C_{k+1}(j)$ calculated, the single knot insertion matrices $\Gamma^{k+2}_{(d\boldsymbol{\sigma},\bar{\boldsymbol{T}}^{[l-1]},\bar{t})}$ from Subsection 3.4.1 needed for the de Boor algorithm, can be obtained in two steps. As we mentioned before, $\Gamma^{k}_{(d\boldsymbol{\sigma}^{(2)},\bar{\boldsymbol{T}}^{[l-1]},\bar{t})}$ is polynomial knot insertion matrix, which we can get explicitly, so the first step is to calculate $\Gamma^{k+1}_{(d\boldsymbol{\sigma}^{(1)},\bar{\boldsymbol{T}}^{[l-1]},\bar{t})}$ by the recurrence in Theorem 3.4 and by using $C_k(j)$. The second step is to get $\Gamma^{k+2}_{(d\boldsymbol{\sigma},\bar{\boldsymbol{T}}^{[l-1]},\bar{t})}$, again from Theorem 3.4 and with $C_{k+1}(j)$. Now, we have all elements to proceed with the generalized de Boor algorithm, and calculate the value of the given weighted spline.

Chapter 5

q-Splines

The q-splines [12, 3, 1] are also, like weighted splines, Chebyshev–polynomial splines, but only of order 4. They have the following analogue in the beam theory. Consider a simply supported elastic beam with supports $\{(x_i, f_i)\}_{i=0}^{l+1}$. Then the deflection of the beam between successive supports is the solution $s(x)$ of the differential equation $[E \cdot I \cdot D^2]s = M$. Here E denotes the Young's modulus of elasticity, I is the cross-sectional moment of inertia, and M is the bending moment. We suppose that $E \cdot I = \frac{1}{q}$, $q > 0$, where q and, under assumption of weightlessness, M, are piecewise linear continuous functions with break points at the supports. Differentiating the above equation twice, we arrive at the two-point boundary value problem on $[x_i, x_{i+1}]$ for $i = 0, \ldots, l$

$$D^2 \frac{1}{q} D^2 s = 0, \quad s(x_i) = f_i,\ s(x_{i+1}) = f_{i+1},\ s''(x_i) = s_i'',\ s''(x_{i+1}) = s_{i+1}'', \tag{5.1}$$

where s_i'' and s_{i+1}'' are so chosen to ensure $s \in C^2[x_0, x_{l+1}]$ when $s'(x_0)$ and $s'(x_{l+1})$ are given. Such a function s is called a q-spline. We will deal with q-splines through Chebyshev theory, and by using algorithms from Chapter 3.

Let $\boldsymbol{T}$ be, again, the extended partition of $[a, b]$ with all interior knots of multiplicity one: $\boldsymbol{T} = \{t_j\}_{j=1}^{n+4}$ and $t_1 \leqslant t_2 \leqslant t_3 \leqslant a = x_0 = t_4 < t_5 < \cdots < t_n < t_{n+1} = b = x_{l+1} \leqslant t_{n+2} \leqslant t_{n+3} \leqslant t_{n+4}$, with $n = l + 4$, associated with partition $\Delta = \{x_i\}_{i=0}^{l+1}$. Let q be a positive continuous piecewise linear function defined by

$$q|_{[t_i, t_{i+1}]}(x) := \frac{q_{i+1} - q_i}{h_i}(x - t_i) + q_i,$$

for $t_i < t_{i+1}$, where $h_i := t_{i+1} - t_i$. The positivity assumption on q is equivalent to $q_i > 0$ for each i. Then for $d\boldsymbol{\sigma} := (d\lambda(\tau_2), q(\tau_3)d\lambda(\tau_3), d\lambda(\tau_4))^{\mathrm{T}}$, with $d\lambda$ being the Lebesgue measure as in the previous chapter, CCC–system associated with $d\boldsymbol{\sigma}$ is

$$\begin{aligned} u_1(x) &= 1 \\ u_2(x) &= \int_a^x d\tau_2 \\ u_3(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} q(\tau_3)\, d\tau_3 \end{aligned} \tag{5.2}$$

$$u_4(x) = \int_a^x d\tau_2 \int_a^{\tau_2} q(\tau_3)\, d\tau_3 \int_a^{\tau_3} d\tau_4.$$

The first reduced system is

$$\begin{aligned} u_{1,1}(x) &= 1 \\ u_{1,2}(x) &= \int_a^x q(\tau_3)\, d\tau_3 \\ u_{1,3}(x) &= \int_a^x q(\tau_3)\, d\tau_3 \int_a^{\tau_3} d\tau_4, \end{aligned}$$

while the generalized derivatives are

$$\begin{aligned} L_{(1,d\boldsymbol{\sigma})} &= D, & L_{(1,d\boldsymbol{\sigma}^{(1)})} &= \tfrac{1}{q}D, \\ L_{(2,d\boldsymbol{\sigma})} &= \tfrac{1}{q}D^2, & L_{(2,d\boldsymbol{\sigma}^{(1)})} &= D\tfrac{1}{q}D, \\ L_{(3,d\boldsymbol{\sigma})} &= D\tfrac{1}{q}D^2, & L_{(3,d\boldsymbol{\sigma}^{(1)})} &= D^2\tfrac{1}{q}D. \\ L_{(4,d\boldsymbol{\sigma})} &= D^2\tfrac{1}{q}D^2, \end{aligned}$$

For $L_{(4,d\boldsymbol{\sigma})}$, the measure by which we extend the CCC–system is also the Lebesgue measure. Then $\mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T}) \subseteq \operatorname{Ker} L_{(4,d\boldsymbol{\sigma})} \cap C^2[a,b]$, *i.e.* $s \in \mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T})$ fulfils (5.1) for some f_i and s_i'', $i = 0, \ldots, l+1$, so $\mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T})$ is the space of q-splines. From (5.2) it is obvious that $\mathcal{S}(4, d\boldsymbol{\sigma}, \boldsymbol{T}) \subset \mathcal{S}(5, d\boldsymbol{\lambda}, \widetilde{\boldsymbol{T}})$ where $d\boldsymbol{\lambda} := (d\lambda, d\lambda, d\lambda, d\lambda)$ and $\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\} := \boldsymbol{T}_{(\Delta, \boldsymbol{m}^{(2)})}$:

$$\begin{array}{cccccc} \cdots & t_{i-1} & t_i & t_{i+1} & t_{i+2} & \cdots \\ \hline \cdots & \tilde{t}_{r-3} & \tilde{t}_{r-1} & \tilde{t}_{r+1} & \tilde{t}_{r+3} & \cdots \\ \cdots & \tilde{t}_{r-2} & \tilde{t}_r & \tilde{t}_{r+2} & \tilde{t}_{r+4} & \cdots \end{array},$$

with $\boldsymbol{m}^{(2)}$ defined as in (3.12). Also, $\mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \boldsymbol{T}) \subseteq \operatorname{Ker} L_{(3,d\boldsymbol{\sigma}^{(1)})} \cap C^1[a,b]$, and $\mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \boldsymbol{T}) \subset \mathcal{S}(4, d\boldsymbol{\lambda}^{(1)}, \widetilde{\boldsymbol{T}})$. The spline space associated with the second reduced system is the space of ordinary polynomial splines of order 2.

To calculate with q-splines, we use the generalized de Boor algorithm described in Subsection 3.4.2, or the Oslo type recurrence from Subsection 3.3.2. It remains to derive the integrals of B-splines needed for both of algorithms. Lemma 3.1 is suitable for calculating $C_3(j)$.

5.1 q-Splines by the generalized de Boor algorithm

Let us consider less smooth B-splines $\widetilde{T}_j^l := T^l_{(j, d\boldsymbol{\sigma}^{(4-l)}, \widetilde{\boldsymbol{T}})}$, for $l = 2, 3, 4$, $\widehat{\boldsymbol{T}} = \{\hat{t}_j\} := \boldsymbol{T}_{(\Delta, \boldsymbol{m}^{(3)})}$, $\boldsymbol{m}^{(3)} := (3, \ldots, 3)$:

$\cdots$	t_{i-1}	t_i	t_{i+1}	t_{i+2}	$\cdots$
$\cdots$	$\hat{t}_{s-5}$	$\hat{t}_{s-2}$	$\hat{t}_{s+1}$	$\hat{t}_{s+4}$	$\cdots$
$\cdots$	$\hat{t}_{s-4}$	$\hat{t}_{s-1}$	$\hat{t}_{s+2}$	$\hat{t}_{s+5}$	$\cdots$
$\cdots$	$\hat{t}_{s-3}$	$\hat{t}_s$	$\hat{t}_{s+3}$	$\hat{t}_{s+6}$	$\cdots$

and $\widehat{B}_j^l := \boldsymbol{T}^l_{(j,d\boldsymbol{\lambda}^{(5-l)},\widehat{\boldsymbol{T}})}$ for $l = 4, 5$. As $\mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \widetilde{\boldsymbol{T}}) \subset \mathcal{S}(4, d\boldsymbol{\lambda}^{(1)}, \widehat{\boldsymbol{T}})$, we can write

$$\widetilde{T}^3_{r-1}(x) = \sum_{j=s-2}^{s-1} a_{r-1,j} \widehat{B}^4_j(x),$$
$$\widetilde{T}^3_{r}(x) = \sum_{j=s-1}^{s+1} a_{r,j} \widehat{B}^4_j(x),$$

for $t_i = \tilde{t}_{r-1} = \tilde{t}_r = \hat{t}_{s-2} = \hat{t}_{s-1} = \hat{t}_s < \tilde{t}_{r+1} = \hat{t}_{s+1}$. To determine $a_{r-1,j}$ and $a_{r,j}$, we need

$$L_{(1,d\boldsymbol{\sigma}^{(1)})} \widetilde{T}^3_{r-1}(x) = \frac{\widetilde{B}^2_{r-1}(x)}{\widetilde{C}_2(r-1)} - \frac{\widetilde{B}^2_{r}(x)}{\widetilde{C}_2(r)}, \tag{5.3}$$

by the derivative formula (2.27), with $\widetilde{T}^2_j = \widetilde{B}^2_j := T^2_{(2,d\boldsymbol{\lambda}^{(3)},\widetilde{\boldsymbol{T}})}$ and

$$\widetilde{C}_2(j) := \int_{\tilde{t}_j}^{\tilde{t}_{j+2}} \widetilde{B}^2_j(\tau_3)\, q(\tau_3)\, d\tau_3.$$

Specially,

$$\widetilde{C}_2(r-1) = \frac{(2q_i + q_{i+1})h_i}{6}, \quad \widetilde{C}_2(r) = \frac{(q_i + 2q_{i+1})h_i}{6}.$$

By (5.3) we have

$$L_{(1,d\boldsymbol{\sigma}^{(1)})} \widetilde{T}^3_{r-1}(t_i^+) = \frac{1}{\widetilde{C}_2(r-1)}, \qquad L_{(1,d\boldsymbol{\sigma}^{(1)})} \widetilde{T}^3_{r-1}(t_{i+1}^-) = -\frac{1}{\widetilde{C}_2(r)},$$

and further

$$a_{r-1,s-2} = \frac{2q_i}{2q_i + q_{i+1}}, \qquad a_{r-1,s-1} = \frac{2q_{i+1}}{q_i + 2q_{i+1}},$$

so

$$\widetilde{T}^3_{r-1}(x) = \frac{2q_i}{2q_i + q_{i+1}} \widehat{B}^4_{s-2}(x) + \frac{2q_{i+1}}{q_i + 2q_{i+1}} \widehat{B}^4_{s-1}(x). \tag{5.4}$$

For $\widetilde{T}^3_r$ we use

$$\widetilde{T}^3_r(t_{i+1}) = 1, \quad L_{(1,d\boldsymbol{\sigma}^{(1)})} \widetilde{T}^3_r(t_{i+1}^-) = \frac{1}{\widetilde{C}_2(r)}, \quad L_{(1,d\boldsymbol{\sigma}^{(1)})} \widetilde{T}^3_r(t_{i+1}^+) = -\frac{1}{\widetilde{C}_2(r+1)},$$

to get

$$a_{r,s-1} = \frac{q_i}{2q_{i+1} + q_i}, \quad a_{r,s} = 1, \quad a_{r,s+1} = \frac{q_{i+2}}{2q_{i+1} + q_{i+2}},$$

and finally

$$\widetilde{T}_r^3(x) = \frac{q_i}{q_i + 2q_{i+1}} \widehat{B}_{s-1}^4(x) + \widehat{B}_s^4(x) + \frac{q_{i+2}}{2q_{i+1} + q_{i+2}} \widehat{B}_{s+1}^4(x). \tag{5.5}$$

If we denote

$$\widetilde{C}_3(j) := \int_{\tilde{t}_j}^{\tilde{t}_{j+3}} \widetilde{T}_j^3(\tau_2)\, d\tau_2,$$

then it is easy to see from (5.4) and (5.5) that

$$\widetilde{C}_3(r-1) = \frac{h_i}{4} \left(\frac{2q_i}{2q_i + q_{i+1}} + \frac{2q_{i+1}}{q_i + 2q_{i+1}} \right),$$
$$\widetilde{C}_3(r) = \frac{1}{4} \left(\frac{q_i h_i}{q_i + 2q_{i+1}} + h_i + h_{i+1} + \frac{q_{i+2} h_{i+1}}{2q_{i+1} + q_{i+2}} \right),$$

and it is obvious that $C_2(i) = \widetilde{C}_2(r) + \widetilde{C}_2(r+1)$. Now, according to Lemma 3.1

$$\begin{aligned} C_3(i) &= \frac{\widetilde{C}_2(r)}{C_2(i)} \widetilde{C}_3(r) + \widetilde{C}_3(r+1) + \frac{\widetilde{C}_2(r+3)}{C_2(i+1)} \widetilde{C}_3(r+2) \\ &= \frac{(q_i + 2q_{i+1})h_i}{(q_i + 2q_{i+1})h_i + (2q_{i+1} + q_{i+2})h_{i+2}} \widetilde{C}_3(r) + \widetilde{C}_3(r+1) \\ &\quad + \frac{(2q_{i+2} + q_{i+3})h_{i+2}}{(q_{i+1} + 2q_{i+2})h_{i+1} + (2q_{i+2} + q_{i+3})h_{i+2}} \widetilde{C}_3(r+2). \end{aligned}$$

The rest of the integrals of B-splines needed for the generalized de Boor algorithm, with $C_1(j) = h_j$, follow from Theorem 2.19 or Remark 2.7, just by taking the limits of the knots.

5.2 q-Splines by the Oslo type algorithm

In this case, by (2.35), we can get the B-splines of higher order in a stable manner. Let us start with

$$\widetilde{T}_{r-1}^4(x) = \frac{1}{\widetilde{C}_3(r-1)} \int_{\tilde{t}_{r-1}}^{x} \widetilde{T}_{r-1}^3(\tau_2)\, d\tau_2 - \frac{1}{\widetilde{C}_3(r)} \int_{\tilde{t}_r}^{x} \widetilde{T}_r^3(\tau_2)\, d\tau_2. \tag{5.6}$$

From (5.6), by using (5.4), (5.5) and the well known equation for integrals of polynomial splines (which is a generalization of (2.36))

$$\int_{-\infty}^{x} T_{(i,d\boldsymbol{\lambda}^{(1)},\boldsymbol{T})}^k(t)\, dt = \frac{t_{i+k} - t_i}{k} \sum_{j=i}^{i+k-1} T_{(i,d\boldsymbol{\lambda},\boldsymbol{T})}^{k+1}(x),$$

for any extended partition $\boldsymbol{T} = \{t_j\}$ and $x \in [t_i, t_{i+k}]$, we get

$$\begin{aligned}\widetilde{T}^4_{r-1}(x) = &\frac{1}{\widetilde{C}_3(r-1)}\frac{2q_i}{2q_i+q_{i+1}}\frac{h_i}{4}\widehat{B}^5_{s-2}(x)\\ &+\frac{1}{\widetilde{C}_3(r)}\left(\frac{h_i+h_{i+1}}{4}+\frac{q_{i+2}}{2q_{i+1}+q_{i+2}}\frac{h_{i+1}}{4}\right)\widehat{B}^5_{s-1}(x)\\ &+\frac{1}{\widetilde{C}_3(r)}\frac{q_{i+2}}{2q_{i+1}+q_{i+2}}\frac{h_{i+1}}{4}\widehat{B}^5_{s}(x).\end{aligned}$$

In the same way

$$\begin{aligned}\widetilde{T}^4_{r}(x) = &\frac{1}{\widetilde{C}_3(r)}\frac{q_i}{q_i+2q_{i+1}}\frac{h_i}{4}\widehat{B}^5_{s-1}(x)\\ &+\frac{1}{\widetilde{C}_3(r)}\left(\frac{q_i}{q_i+2q_{i+1}}\frac{h_i}{4}+\frac{h_i+h_{i+1}}{4}\right)\widehat{B}^5_{s}(x)\\ &+\frac{1}{\widetilde{C}_3(r+1)}\frac{2q_{i+2}}{q_{i+1}+2q_{i+2}}\frac{h_{i+1}}{4}\widehat{B}^5_{s+1}(x).\end{aligned}$$

Again Lemma 3.1 and Theorem 3.7 give $T^4_i := T^4_{(i,d\boldsymbol{\sigma},\boldsymbol{T})}$.

Chapter 6

Tension splines

As the first example of non–polynomial Chebyshev splines we will observe tension splines [18, 11, 13, 37]. It is well known that they have very good shape preserving and approximation properties widely used in removing extraneous inflections [13], and solution of singularly perturbed differential [19] and integral equations [9].

Let $a = x_0 < x_1 < \cdots < x_l < x_{l+1} = b$ be a partition of $[a, b]$, and tension parameters $p_i \in \mathbb{R}$, $p_i > 0$ for $i = 0, \ldots, l$. Tension spline is function s such that

$$s^{(4)}(x) - p_i^2 s^{(2)}(x) = (D^2 - p_i^2) D^2 s(x) = 0$$

for every $x \in [x_i, x_{i+1})$, $i = 0, \ldots, l$. It immediately follows that

$$s|_{[x_i, x_{i+1})} \in \operatorname{span}\{1, x, \sinh(p_i x), \cosh(p_i x)\}$$

for $i = 0, \ldots, l$. Most often, tension splines of class C^1 or C^2 are used, but C^2 ones are not covered by the Chebyshev theory, so we will treat them as a subspace of C^1 tension splines. In fact, C^2 tension splines belong to the class of generalized tension B-splines examined in [13] (see also [16]). Next to this two spline spaces, we will also observe tension splines that are, like C^1, Chebyshev splines, and have continuous second generalized derivative at the joining points x_i instead of the second ordinary derivative. This kind of splines we will call Chebyshev tension splines.

To derive CCC–system for tension splines, we have to represent the differential operator $L_4 := (D^2 - p^2)D^2$, with $p|_{[x_i, x_{i+1})} :\equiv p_i$, for $i = 0, \ldots, l-1$ and $p|_{[x_l, x_{l+1}]} :\equiv p_l$, more conveniently:

Theorem 6.1 *Let $a = x_0 < x_1 < \cdots < x_l < x_{l+1} = b$ be a partition of $[a, b]$,*

$$q \in \bigcap_{i=0}^{l-1} C^1([x_i, x_{i+1})) \cap C^1([x_l, x_{l+1}]), \quad r \in \bigcap_{i=0}^{l-1} C([x_i, x_{i+1})) \cap C([x_l, x_{l+1}]),$$

and let $L_2[y] := D(q\,Dy) + r\,y = 0$ have a solution u without zeros on the interval $[a, b]$. Then for every $y \in \bigcap_{i=0}^{l-1} C^2([x_i, x_{i+1})) \cap C^2([x_l, x_{l+1}])$

$$L_2[y] = \frac{1}{u} D\left[q\,u^2\,D\left(\frac{y}{u}\right)\right].$$

Proof: Let u be a solution without zeros on the interval $[a, b]$, *i.e.*

$$L_2[u] = q\,u'' + q'u' + r\,u = 0. \tag{6.1}$$

Then on each $[x_i, x_{i+1})$ and $[x_l, x_{l+1}]$:

$$\begin{aligned}
\frac{1}{u}D\left[q\,u^2\,D\left(\frac{y}{u}\right)\right] &= \frac{1}{u}D\left[q\,u^2\,\frac{y'u - y\,u'}{u^2}\right] \\
&= \frac{1}{u}D\left[q(y'u - y\,u')\right] \\
&= \frac{1}{u}[q'(y'u - y\,u') + q(y''u + y'u' - y'u' - y\,u'')] \\
&= \frac{1}{u}[(q\,y'' + q'y')u - (q\,u'' + q'u')y] \\
&= \frac{1}{u}[(q\,y'' + q'y')u + r\,u\,y] \\
&= \frac{1}{u}[(q\,y'' + q'y' + r\,y)u] = L_2[y],
\end{aligned}$$

because of (6.1). ■

If we apply Theorem 6.1 on $L_2[y] = (D^2 - p^2)y = y'' - p^2 y$, with $u(x) = \cosh(p(x)\,x)$, then $q \equiv 1$ and $r \equiv -p^2$, and we get

$$L_2[y] = \frac{1}{\cosh(p\,x)}D\left[\cosh^2(p\,x)D\left(\frac{y}{\cosh(p\,x)}\right)\right].$$

Because $L_4 = L_2 D^2$,

$$L_4 = \left(\frac{1}{\cosh(p\,x)}D\right)\left(\cosh^2(p\,x)D\right)\left(\frac{1}{\cosh(p\,x)}D\right)D.$$

Therefore, the associated measure vector for tension splines is

$$d\boldsymbol{\sigma} := (d\tau_2, \cosh(p(\tau_3)\,\tau_3)\,d\tau_3, \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)})^{\mathrm{T}},$$

with the measure by which we extend the CCC–system (see Theorem 2.1) $d\sigma_5 := \cosh(p(\tau_5)\,d\tau_5)$, the associated CCC–system:

$$\begin{aligned}
u_1(x) &= 1 \\
u_2(x) &= \int_a^x d\tau_2 \\
u_3(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \cosh(p(\tau_3)\,\tau_3)\,d\tau_3 \\
u_4(x) &= \int_a^x d\tau_2 \int_a^{\tau_2} \cosh(p(\tau_3)\,\tau_3)\,d\tau_3 \int_a^{\tau_3} \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)},
\end{aligned}$$

and the generalized derivatives are given by

$$\begin{aligned}
L_{(1,d\boldsymbol{\sigma})} &= D \\
L_{(2,d\boldsymbol{\sigma})} &= \left(\frac{1}{\cosh(p\,x)}D\right)D \\
L_{(3,d\boldsymbol{\sigma})} &= \left(\cosh^2(p\,x)D\right)\left(\frac{1}{\cosh(p\,x)}D\right)D \\
L_{(4,d\boldsymbol{\sigma})} &= L_4 = \left(\frac{1}{\cosh(p\,x)}D\right)\left(\cosh^2(p\,x)D\right)\left(\frac{1}{\cosh(p\,x)}D\right)D.
\end{aligned}$$

So, we have $\mathcal{S}(4, d\boldsymbol{\sigma}) = \text{span}\{1, x, \sinh(p(x)\,x), \cosh(p(x)\,x)\} \subseteq \text{Ker}L_{(4,d\boldsymbol{\sigma})}$. For the first reduced system

$$\begin{aligned}
u_{1,1}(x) &= 1 \\
u_{1,2}(x) &= \int_a^x \cosh(p(\tau_3)\,\tau_3)\,d\tau_3 \\
u_{1,3}(x) &= \int_a^x \cosh(p(\tau_3)\,\tau_3)\,d\tau_3 \int_a^{\tau_3} \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)}
\end{aligned}$$

holds $\mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}) = \text{span}\{1, \sinh(p(x)\,x), \cosh(p(x)\,x)\} \subseteq \text{Ker}L_{(3,d\boldsymbol{\sigma}^{(1)})}$, with

$$L_{(3,d\boldsymbol{\sigma}^{(1)})} = \left(\frac{1}{\cosh(p\,x)}D\right)\left(\cosh^2(p\,x)D\right)\left(\frac{1}{\cosh(p\,x)}D\right),$$

while the second reduced system

$$\begin{aligned}
u_{2,1}(x) &= 1 \\
u_{2,2}(x) &= \int_a^x \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)}
\end{aligned}$$

satisfies $\mathcal{S}(2, d\boldsymbol{\sigma}^{(2)}) = \text{span}\{1, \tanh(p(x)\,x)\} \subseteq \text{Ker}L_{(2,d\boldsymbol{\sigma}^{(2)})}$ with

$$L_{(2,d\boldsymbol{\sigma}^{(2)})} = \left(\frac{1}{\cosh(p\,x)}D\right)\left(\cosh^2(p\,x)D\right).$$

6.1 C^1 tension splines

C^1 tension splines will also be a starting point for calculating with Chebyshev and C^2 tension splines. Let $\Delta = \{x_i\}_{i=0}^{l+1}$, $\boldsymbol{T} = \{t_i\} := \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(1)})}$ with $\boldsymbol{m}^{(1)}$ as in (3.4) and $\widetilde{\boldsymbol{T}} = \{\tilde{t}_r\} := \boldsymbol{T}_{(\Delta,\boldsymbol{m}^{(2)})}$ with $\boldsymbol{m}^{(2)}$ as in (3.12). With the notation from the previous chapter, for $t_i = \tilde{t}_{r-1} = \tilde{t}_r < \tilde{t}_{r+1}$

$$\widetilde{C}_1(r) := \int_{\tilde{t}_r}^{\tilde{t}_{r+1}} \frac{d\tau_4}{\cosh^2(p(\tau_4)\,\tau_4)} = \frac{\sinh(p_i h_i)}{p_i \cosh(p_i t_i)\cosh(p_i t_{i+1})},$$

where $h_i := t_{i+1} - t_i$. By Corollary 2.2

$$\widetilde{T}^2_{r-1}(x) = \frac{\cosh(p_i t_i)\sinh(p_i(t_{i+1}-x))}{\cosh(p_i x)\sinh(p_i h_i)},$$
$$\widetilde{T}^2_{r}(x) = \frac{\cosh(p_i t_{i+1})\sinh(p_i(x-t_i))}{\cosh(p_i x)\sinh(p_i h_i)},$$

for $x \in [t_i, t_{i+1})$ (and hence in what follows, if $i = l$ we assume the subinterval $[x_l, x_{l+1}]$ instead), and from here

$$\widetilde{C}_2(r-1) := \int_{\tilde{t}_{r-1}}^{\tilde{t}_{r+1}} \widetilde{T}^2_{r-1}(\tau_3)\cosh(p(\tau_3)\,\tau_3)\,d\tau_3 = \frac{\cosh(p_i t_i)}{p_i}\tanh\left(\frac{p_i h_i}{2}\right),$$
$$\widetilde{C}_2(r) := \int_{\tilde{t}_{r}}^{\tilde{t}_{r+2}} \widetilde{T}^2_{r}(\tau_3)\cosh(p(\tau_3)\,\tau_3)\,d\tau_3 = \frac{\cosh(p_i t_{i+1})}{p_i}\tanh\left(\frac{p_i h_i}{2}\right).$$

Again, because of Corollary 2.2:

$$\widetilde{T}^3_{r-1}(x) = 2\cosh\left(\frac{p_i h_i}{2}\right)\frac{\sinh\left(\frac{p_i(x-t_i)}{2}\right)\sinh\left(\frac{p_i(t_{i+1}-x)}{2}\right)}{\sinh^2\left(\frac{p_i h_i}{2}\right)}, \quad \text{for } x \in [t_i, t_{i+1})$$

$$\widetilde{T}^3_{r}(x) = \begin{cases} \dfrac{\sinh^2\left(\frac{p_i(x-t_i)}{2}\right)}{\sinh^2\left(\frac{p_i h_i}{2}\right)} & \text{for } x \in [t_i, t_{i+1}), \\ \dfrac{\sinh^2\left(\frac{p_{i+1}(t_{i+2}-x)}{2}\right)}{\sinh^2\left(\frac{p_{i+1} h_{i+1}}{2}\right)} & \text{for } x \in [t_{i+1}, t_{i+2}), \end{cases}$$

and

$$\begin{aligned} \widetilde{C}_3(r-1) &:= \int_{\tilde{t}_{r-1}}^{\tilde{t}_{r+2}} \widetilde{T}^3_{r-1}(\tau_2)\,d\tau_2 \\ &= \frac{2\cosh\left(\frac{p_i h_i}{2}\right)}{p_i \sinh^2\left(\frac{p_i h_i}{2}\right)}\left(\frac{p_i h_i}{2}\cosh\left(\frac{p_i h_i}{2}\right) - \sinh\left(\frac{p_i h_i}{2}\right)\right), \qquad (6.2) \\ \widetilde{C}_3(r) &:= \int_{\tilde{t}_{r}}^{\tilde{t}_{r+3}} \widetilde{T}^3_{r}(\tau_2)\,d\tau_2 \\ &= \frac{\sinh(p_i h_i) - p_i h_i}{2p_i \sinh^2\left(\frac{p_i h_i}{2}\right)} + \frac{\sinh(p_{i+1} h_{i+1}) - p_{i+1} h_{i+1}}{2p_{i+1}\sinh^2\left(\frac{p_{i+1} h_{i+1}}{2}\right)}. \qquad (6.3) \end{aligned}$$

6.1.1 Generalized de Boor algorithm for C^1 tension splines

Now we can apply the generalized de Boor algorithm. Let $s \in \mathcal{S}(4, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$, $s = \sum_j c_j \widetilde{T}^4_j$, and $\bar{t} \in (\tilde{t}_r, \tilde{t}_{r+1}) = (t_i, t_{i+1})$. In this case, the knot insertion matrices

involved in the generalized de Boor algorithm can be explicitly calculated by using Theorem 3.4. Let $\Gamma^2_{(d\boldsymbol{\sigma},\tilde{\boldsymbol{T}},\bar{t})} = [\gamma^2_{i,j}]$. Then

$$\gamma^2_{r,r-1} = \frac{\sinh\left(p_i(t_{i+1}-\bar{t})\right)\cosh\left(p_i t_i\right)}{\sinh\left(p_i h_i\right)\cosh\left(p_i \bar{t}\right)},$$
$$\gamma^2_{r,r} = \frac{\sinh\left(p_i(\bar{t}-t_i)\right)\cosh\left(p_i t_{i+1}\right)}{\sinh\left(p_i h_i\right)\cosh\left(p_i \bar{t}\right)}.$$

From here, we can get $\Gamma^3_{(d\boldsymbol{\sigma},\tilde{\boldsymbol{T}},\bar{t})} = [\gamma^3_{i,j}]$:

$$\gamma^3_{r-1,r-2} = \frac{\sinh\left(p_i(t_{i+1}-\bar{t})\right)\left(\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)+\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)\right)}{\sinh\left(p_i h_i\right)\tanh\left(\frac{p_i h_i}{2}\right)},$$
$$\gamma^3_{r-1,r-1} = \frac{\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)}{\tanh\left(\frac{p_i h_i}{2}\right)},$$
$$\gamma^3_{r,r-1} = \frac{\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)}{\tanh\left(\frac{p_i h_i}{2}\right)},$$
$$\gamma^3_{r,r} = \frac{\sinh\left(p_i(\bar{t}-t_i)\right)\left(\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)+\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)\right)}{\sinh\left(p_i h_i\right)\tanh\left(\frac{p_i h_i}{2}\right)}.$$

If we denote

$$\tilde{f}(x) := \frac{\sinh x - x}{2\sinh^2\frac{x}{2}}, \quad \tilde{g}(x) := \frac{x\cosh x - \sinh x}{\sinh^2 x}, \quad \tilde{u}(x,y) := \frac{\sinh x}{\sinh(x+y)} \tag{6.4}$$

then the elements of $\Gamma^4_{(d\boldsymbol{\sigma},\tilde{\boldsymbol{T}},\bar{t})} = [\gamma^4_{i,j}]$ are:

$$\gamma^4_{r-2,r-3} = \tilde{u}\left(\tfrac{p_i(t_{i+1}-\bar{t})}{2}, \tfrac{p_i(\bar{t}-t_i)}{2}\right)\cdot \frac{\frac{1}{p_i}\left[2\tilde{g}\left(\frac{p_i(\bar{t}-t_i)}{2}\right)+\tilde{u}\left(\frac{p_i(t_{i+1}-\bar{t})}{2}, \frac{p_i(\bar{t}-t_i)}{2}\right)\left(\tilde{f}(p_i(\bar{t}-t_i))+\tilde{f}(p_i(t_{i+1}-\bar{t}))\right)\right]}{\frac{1}{p_{i-1}}\tilde{f}(p_{i-1}h_{i-1})+\frac{1}{p_i}\tilde{f}(p_i h_i)},$$
$$\gamma^4_{r-2,r-2} = \frac{\frac{1}{p_{i-1}}\tilde{f}(p_{i-1}h_{i-1})+\frac{1}{p_i}\tilde{f}(p_i(\bar{t}-t_i))}{\frac{1}{p_{i-1}}\tilde{f}(p_{i-1}h_{i-1})+\frac{1}{p_i}\tilde{f}(p_i h_i)},$$
$$\gamma^4_{r-1,r-2} = \tilde{u}\left(\tfrac{p_i(t_{i+1}-\bar{t})}{2}, \tfrac{p_i(\bar{t}-t_i)}{2}\right)\cdot \frac{\tilde{u}\left(\frac{p_i(\bar{t}-t_i)}{2}, \frac{p_i(t_{i+1}-\bar{t})}{2}\right)\left(\tilde{f}(p_i(\bar{t}-t_i))+\tilde{f}(p_i(t_{i+1}-\bar{t}))\right)+2\tilde{g}\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)}{2\tilde{g}\left(\frac{p_i h_i}{2}\right)},$$
$$\gamma^4_{r-1,r-1} = \tilde{u}\left(\tfrac{p_i(\bar{t}-t_i)}{2}, \tfrac{p_i(t_{i+1}-\bar{t})}{2}\right)\cdot \frac{2\tilde{g}\left(\frac{p_i(\bar{t}-t_i)}{2}\right)+\tilde{u}\left(\frac{p_i(t_{i+1}-\bar{t})}{2}, \frac{p_i(\bar{t}-t_i)}{2}\right)\left(\tilde{f}(p_i(\bar{t}-t_i))+\tilde{f}(p_i(t_{i+1}-\bar{t}))\right)}{2\tilde{g}\left(\frac{p_i h_i}{2}\right)},$$

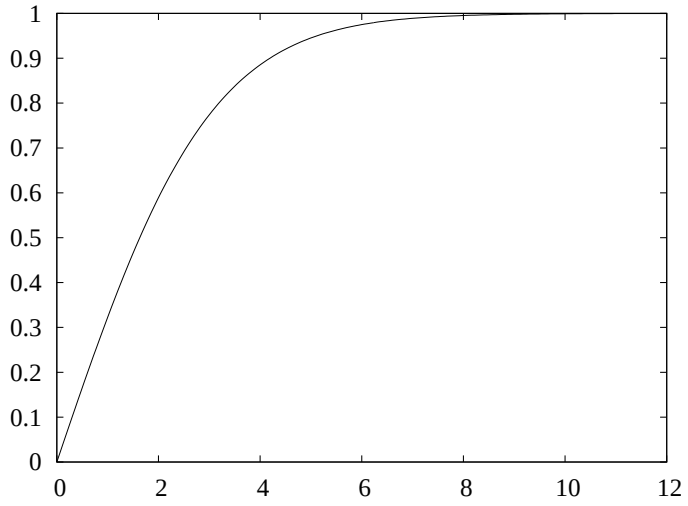

Figure 6.1: Function $\tilde{f}$

$$\gamma^4_{r,r-1} = \frac{\frac{1}{p_i}\tilde{f}(p_i(t_{i+1}-\bar{t})) + \frac{1}{p_{i+1}}\tilde{f}(p_{i+1}h_{i+1})}{\frac{1}{p_i}\tilde{f}(p_i h_i) + \frac{1}{p_{i+1}}\tilde{f}(p_{i+1}h_{i+1})},$$

$$\gamma^4_{r,r} = \tilde{u}\left(\frac{p_i(\bar{t}-t_i)}{2}, \frac{p_i(t_{i+1}-\bar{t})}{2}\right) \cdot \frac{\frac{1}{p_i}\left[\tilde{u}\left(\frac{p_i(\bar{t}-t_i)}{2}, \frac{p_i(t_{i+1}-\bar{t})}{2}\right)\left(\tilde{f}(p_i(\bar{t}-t_i)) + \tilde{f}(p_i(t_{i+1}-\bar{t}))\right) + 2\tilde{g}\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)\right]}{\frac{1}{p_i}\tilde{f}(p_i h_i) + \frac{1}{p_{i+1}}\tilde{f}(p_{i+1}h_{i+1})}.$$

To calculate the rest of the knot insertion matrices needed for the de Boor algo-

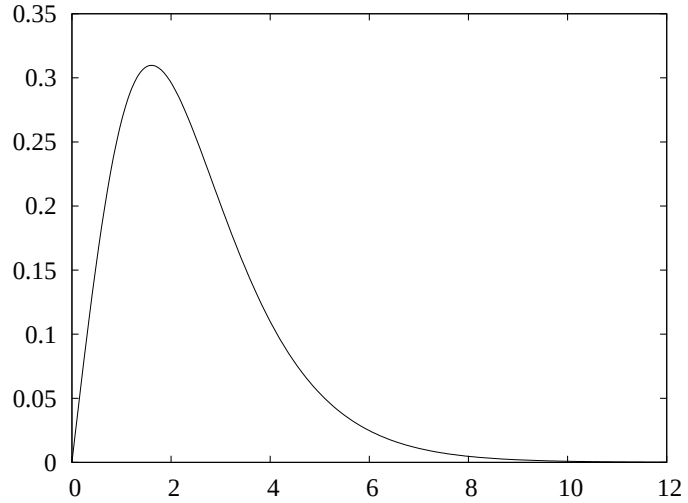

Figure 6.2: Function $\tilde{g}$

rithm by Corollaries 3.1 and 3.2 we have to say more about the functions $\tilde{f}$, $\tilde{g}$ and $\tilde{u}$ (6.4) (see Figure 6.1 and 6.2). We only need arguments for all three functions to be nonnegative. It can be easily shown that $\tilde{g}(x) > 0$ for $x > 0$, and

$$\lim_{x\to 0} \tilde{g}(x) = 0, \qquad \lim_{x\to\infty} \tilde{g}(x) = 0.$$

Because $\tilde{f}'(x) = \frac{\tilde{g}\left(\frac{x}{2}\right)}{\sinh\left(\frac{x}{2}\right)}$, $\tilde{f}$ is increasing nonnegative function with

$$\lim_{x\to 0} \tilde{f}(x) = 0, \qquad \lim_{x\to\infty} \tilde{f}(x) = 1.$$

The function $\tilde{u}$ is clearly positive for positive arguments, and

$$\lim_{x\to 0} \tilde{u}(x,y) = 0, \quad \lim_{y\to 0} \tilde{u}(x,y) = 1, \quad \lim_{x\to\infty} \tilde{u}(x,y) = e^{-y}, \quad \lim_{y\to\infty} \tilde{u}(x,y) = 0.$$

To achieve numerical stability we have to do some rearranging. First, let us define new functions

$$f(x) := \frac{\tilde{f}(x)}{x} = \frac{\sinh x - x}{2x\sinh^2\frac{x}{2}}, \tag{6.5}$$

$$g(x) := e^x\frac{\tilde{g}(x)}{x} = e^x\frac{x\cosh x - \sinh x}{x\sinh^2 x}, \tag{6.6}$$

$$u(x,y) := e^y\tilde{u}(x,y) = e^y\frac{\sinh x}{\sinh(x+y)} \tag{6.7}$$

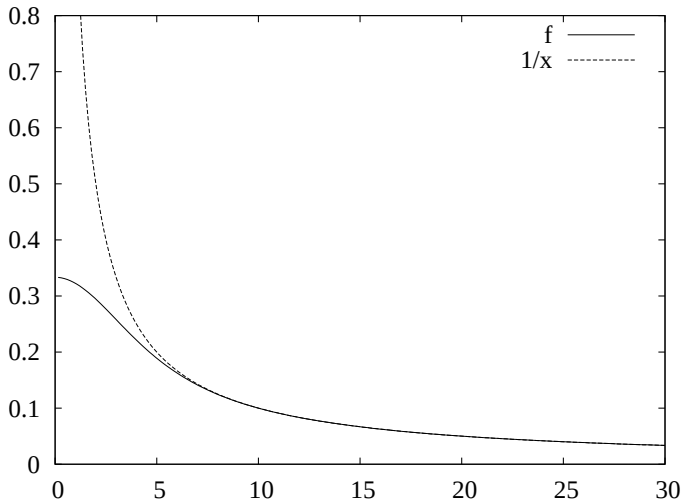

Figure 6.3: Functions f and $\frac{1}{x}$

from (6.4). The functions $\tilde{f}$ and $\tilde{g}$ are divided by x to achieve that algorithm can deal even with some of the tension parameters equal to zero, and by multiplying $\tilde{g}$

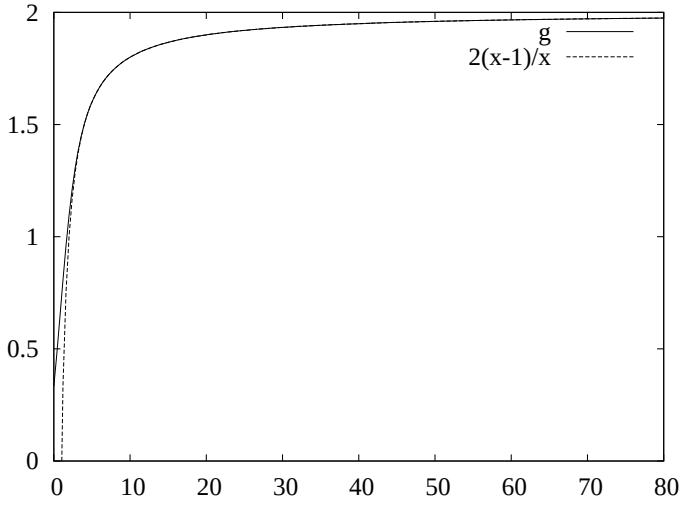

Figure 6.4: Functions g and $2\frac{x-1}{x}$

and $\tilde{u}$ by exponential function, the algorithm avoids possible underflows, which can lead to division of zero by zero. The function f is positive, decreasing, $f(x) < \frac{1}{x}$ for $x \geqslant 0$, in fact $f(x) \approx \frac{1}{x}$ for large x (see Figure 6.3), and

$$\lim_{x\to 0} f(x) = \frac{1}{3}, \qquad \lim_{x\to\infty} f(x) = 0,$$

while g is positive, increasing, $g(x) \approx 2\frac{x-1}{x}$ for large x (see Figure 6.4), and

$$\lim_{x\to 0} g(x) = \frac{1}{3}, \qquad \lim_{x\to\infty} g(x) = 2.$$

It is again trivial to see that u is positive, with $0 \leqslant u(x,y) \leqslant 1$ and

$$\lim_{x\to 0} u(x,y) = 0,\ \lim_{y\to 0} u(x,y) = 1,\ \lim_{x\to\infty} u(x,y) = 1,\ \lim_{y\to\infty} u(x,y) = 1 - e^{-2x}.$$

u is not continuous at $(0,0)$, but this is not a problem, since u is always calculated as $u(p\,h, p\,k)$. If $h = k = 0$, then u is multiplied by zero and the de Boor's maxim gives zero. If $p = 0$ and at least $h \neq 0$, then

$$\lim_{p\to 0} u(p\,h, p\,k) = \frac{h}{h+k},$$

and in this case u is multiplied by nonzero expression. If we set all $p_i = 0$, we get the cubic polynomial spline, as expected. All three functions can be calculated numerically stable up to the computer accuracy by Padé or Chebyshev polynomial approximation, combined with the asymptotic formulæ.

We use the function CC_3 instead of $\widetilde{C}_3(j)$:

$$\begin{aligned} CC_3(h_1,h_2,h_3;p_1,p_2,p_3) := {} & u\left(\frac{\tilde{p}_1 h_1}{2}, \frac{\tilde{p}_1 h_2}{2}\right)\Big(h_1 f(p_1 h_1) + h_2 f(p_2 h_2)\Big) \\ & + h_2 g\left(\frac{p_2 h_2}{2}\right) + u\left(\frac{\tilde{p}_2 h_3}{2}, \frac{\tilde{p}_2 h_2}{2}\right)\Big(h_2 f(p_2 h_2) + h_3 f(p_3 h_3)\Big), \quad (6.8) \end{aligned}$$

which unites both $\widetilde{C}_3(r-1)$ and $\widetilde{C}_3(r)$ from (6.2) and (6.3), up to some multiplicative factor, with $\tilde{p}_1 = \max\{p_1, p_2\}$ and $\tilde{p}_2 = \max\{p_2, p_3\}$. Then the de Boor algorithm for order 4 (3.20) can be represented as

$$s(\bar{t}) = c_{r-3}B_{r-3} + c_{r-2}B_{r-2} + c_{r-1}B_{r-1} + c_rB_r,$$

where $B_j := \widetilde{T}_j^4(\bar{t})$, $j = r-3, \ldots, r$ are given by

$$\begin{aligned}
B_{r-3} &= u_1^2 e^{-p_i(\bar{t}-t_i)}\frac{C_1}{C_2},\\
B_{r-2} &= u_1\left(\frac{C_1C_{11}}{C_3C_2} + \frac{1}{C_6C_7}\left(\frac{C_1C_4C_5}{C_3} + C_8C_9\right)\right),\\
B_{r-1} &= u_2\left(\frac{1}{C_6C_7}\left(C_1C_4 + \frac{C_8C_9C_3}{C_5}\right) + \frac{C_8C_{12}}{C_5C_{10}}\right),\\
B_r &= u_2^2 e^{-p_i(t_{i+1}-\bar{t})}\frac{C_8}{C_{10}},
\end{aligned}$$

with

$$\begin{aligned}
u_1 &:= u(p_i(t_{i+1}-\bar{t}), p_i(\bar{t}-t_i)),\\
u_2 &:= u(p_i(\bar{t}-t_i), p_i(t_{i+1}-\bar{t})),\\
C_1 &:= CC_3(0,0,t_{i+1}-\bar{t};0,0,p_i),\\
C_2 &:= CC_3(h_{i-1},0,h_i;p_{i-1},0,p_i),\\
C_3 &:= CC_3(0,\bar{t}-t_i,t_{i+1}-\bar{t};0,p_i,p_i),\\
C_4 &:= CC_3(0,\bar{t}-t_i,0;0,p_i,0),\\
C_5 &:= CC_3(\bar{t}-t_i,t_{i+1}-\bar{t},0;p_i,p_i,0),\\
C_6 &:= CC_3(\bar{t}-t_i,0,t_{i+1}-\bar{t};p_i,0,p_i),\\
C_7 &:= CC_3(0,h_i,0;0,p_i,0),\\
C_8 &:= CC_3(\bar{t}-t_i,0,0;p_i,0,0),\\
C_9 &:= CC_3(0,t_{i+1}-\bar{t},0;0,p_i,0),\\
C_{10} &:= CC_3(h_i,0,h_{i+1};p_i,0,p_{i+1}),\\
C_{11} &:= CC_3(h_{i-1},0,\bar{t}-t_i;p_{i-1},0,p_i),\\
C_{12} &:= CC_3(t_{i+1}-\bar{t},0,h_{i+1};p_i,0,p_{i+1}).
\end{aligned}$$

For $\bar{t} = t_i$, we just have to take the limit $\bar{t} \to t_i^+$ wherever it appears in the algorithm. Therefore, C^1 tension spline can be calculated numerically stable, avoiding dangerous subtractions.

6.1.2 Splines associated with the reduced systems

We can also stably calculate splines associated with reduced systems. Let $s \in \mathcal{S}(3, d\boldsymbol{\sigma}^{(1)}, \widetilde{\boldsymbol{T}})$, $s = \sum_j c_j^1 \widetilde{T}_j^3$, and $\bar{t} \in [\tilde{t}_r, \tilde{t}_{r+1}) = [t_i, t_{i+1})$, then the generalized de Boor

algorithm expressed by (3.16) looks like

$$s(\bar{t}) = c^1_{r-2}B_{r-2} + c^1_{r-1}B_{r-1} + c^1_r B_r.$$

In the expression above $B_j := \widetilde{T}^3_j(\bar{t})$ are given by

$$B_{r-2} = e^{-p_i(\bar{t}-t_i)}u(p_i(t_{i+1}-\bar{t}), p_i(\bar{t}-t_i))\frac{\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)}{\tanh\left(\frac{p_i h_i}{2}\right)}, \tag{6.9}$$

$$B_{r-1} = \frac{2\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)}{\left(\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)+\tanh\left(\frac{p_i(t_{i+1}-\bar{t})}{2}\right)\right)\tanh\left(\frac{p_i h_i}{2}\right)}, \tag{6.10}$$

$$B_r = e^{-p_i(t_{i+1}-\bar{t})}u(p_i(\bar{t}-t_i), p_i(t_{i+1}-\bar{t}))\frac{\tanh\left(\frac{p_i(\bar{t}-t_i)}{2}\right)}{\tanh\left(\frac{p_i h_i}{2}\right)}. \tag{6.11}$$

Again, taking the limit when $p_i \to 0$ is not a problem, since

$$\lim_{p\to 0}\frac{\tanh p\,x}{\tanh p\,y} = \frac{x}{y}, \qquad \lim_{p\to 0}\frac{\tanh p\,x}{\tanh p\,x + \tanh p\,y} = \frac{x}{x+y}. \tag{6.12}$$

Let now $s \in \mathcal{S}(2, d\boldsymbol{\sigma}^{(2)}, \widetilde{\boldsymbol{T}})$, $s = \sum_j c_j^2 \widetilde{T}^2_j$. Then for $\bar{t} \in [\tilde{t}_r, \tilde{t}_{r+1}) = [t_i, t_{i+1})$

$$s(\bar{t}) = c^2_{r-1}\widetilde{T}^2_{r-1}(\bar{t}) + c^2_r \widetilde{T}^2_r(\bar{t}),$$

and $\widetilde{T}^2_{r-1}(\bar{t})$, $\widetilde{T}^2_r(\bar{t})$ can be stably calculated as

$$\begin{aligned}
\widetilde{T}^2_{r-1}(\bar{t}) &= \frac{\cosh(p_i t_i)\sinh(p_i(t_{i+1}-\bar{t}))}{\cosh(p_i\bar{t})\sinh(p_i h_i)} \\
&= \frac{1+e^{-2|p_i t_i|}}{1+e^{-2|p_i\bar{t}|}}e^{-2p_i(\max\{0,\bar{t}\}-\max\{0,t_i\})}u(p_i(t_{i+1}-\bar{t}), p_i(\bar{t}-t_i)), \\
\widetilde{T}^2_r(\bar{t}) &= \frac{\cosh(p_i t_{i+1})\sinh(p_i(\bar{t}-t_i))}{\cosh(p_i\bar{t})\sinh(p_i h_i)} \\
&= \frac{1+e^{-2|p_i t_{i+1}|}}{1+e^{-2|p_i\bar{t}|}}e^{-2p_i(\min\{0,t_{i+1}\}-\min\{0,\bar{t}\})}u(p_i(\bar{t}-t_i), p_i(t_{i+1}-\bar{t})),
\end{aligned}$$

for t_i, t_{i+1} and $\bar{t}$ being either positive or negative.

6.1.3 Generalized and ordinary derivatives of C^1 tension spline

By the previous subsection and Corollary 2.1, we can get the first and the second generalized derivative of any C^1 tension spline. To be precise, for $\bar{t} \in [\tilde{t}_r, \tilde{t}_{r+1}) = [t_i, t_{i+1})$

$$s(\bar{t}) = \sum_{j=r-3}^{r} c_j \widetilde{T}^4_j(\bar{t}),$$

and the first generalized derivative is

$$L_{(1,d\boldsymbol{\sigma})}s(\bar{t}) = \sum_{j=r-2}^{r} c_j^1 \widetilde{T}_j^3(\bar{t}),$$

with

$$c_j^1 = \frac{c_j - c_{j-1}}{\widetilde{C}_3(j)},$$

for $j = r-2, r-1, r$. The second generalized derivative of this spline is

$$L_{(2,d\boldsymbol{\sigma})}s(\bar{t}) = \sum_{j=r-1}^{r} c_j^2 \widetilde{T}_j^2(\bar{t}), \tag{6.13}$$

with

$$c_j^2 = \frac{c_j^1 - c_{j-1}^1}{\widetilde{C}_2(j)}, \tag{6.14}$$

for $j = r-1, r$. But, if we want to get ordinary derivatives of tension spline, it is also possible to do it in a stable manner. The first derivative is equal to the first generalized derivative, what we already have, and for the second is $D^2 = \cosh(p\,x) L_{(2,d\boldsymbol{\sigma})}$. By multiplying (6.13) with $\cosh(p_i\bar{t})$ and inserting (6.14) we get that

$$\begin{aligned} D^2 s(\bar{t}) = {} & \frac{c_{r-1}^1 - c_{r-2}^1}{\widetilde{\mathfrak{C}}_2(r-1)}\, e^{-p_i(\bar{t}-t_i)} u(p_i(t_{i+1}-\bar{t}), p_i(\bar{t}-t_i)) \\ & + \frac{c_r^1 - c_{r-1}^1}{\widetilde{\mathfrak{C}}_2(r)}\, e^{-p_i(t_{i+1}-\bar{t})} u(p_i(\bar{t}-t_i), p_i(t_{i+1}-\bar{t})), \end{aligned} \tag{6.15}$$

where $\widetilde{\mathfrak{C}}_2(r-1)$ and $\widetilde{\mathfrak{C}}_2(r)$ are defined in (6.19) and (6.20), which can be approximated to the machine precision. It is now not a problem, for example, to apply collocation methods with tension splines in a numerically stable way (see [17]), since for these methods, except the spline value, we need the first and the second ordinary derivative of the spline to form a collocation matrix.

6.2 Chebyshev tension splines

As we have already calculated tension splines associated to the multiplicity vector $\boldsymbol{m}^{(2)}$, we will apply Lemma 3.1 and Theorem 3.7 to get tension splines associated to $\boldsymbol{m}^{(1)}$. Lemma 3.1 gives us

$$C_3(i) = \frac{\widetilde{C}_2(r)}{C_2(i)}\,\widetilde{C}_3(r) + \widetilde{C}_3(r+1) + \frac{\widetilde{C}_2(r+3)}{C_2(i+1)}\,\widetilde{C}_3(r+2), \tag{6.16}$$

with

$$\frac{\widetilde{C}_2(r)}{C_2(i)} = \frac{\dfrac{\cosh(p_i t_{i+1})}{p_i}\tanh\left(\dfrac{p_i h_i}{2}\right)}{\dfrac{\cosh(p_i t_{i+1})}{p_i}\tanh\left(\dfrac{p_i h_i}{2}\right)+\dfrac{\cosh(p_{i+1} t_{i+1})}{p_{i+1}}\tanh\left(\dfrac{p_{i+1} h_{i+1}}{2}\right)},$$

$$\frac{\widetilde{C}_2(r+3)}{C_2(i+1)} = \frac{\dfrac{\cosh(p_{i+2} t_{i+2})}{p_{i+2}}\tanh\left(\dfrac{p_{i+2} h_{i+2}}{2}\right)}{\dfrac{\cosh(p_{i+1} t_{i+2})}{p_{i+1}}\tanh\left(\dfrac{p_{i+1} h_{i+1}}{2}\right)+\dfrac{\cosh(p_{i+2} t_{i+2})}{p_{i+2}}\tanh\left(\dfrac{p_{i+2} h_{i+2}}{2}\right)},$$

which can be calculated by means of the functions

$$s(x,y) := \frac{x\tanh y}{y\tanh x}, \qquad z(x,y,a) := \frac{1}{1+a\frac{\cosh y}{\cosh x}}, \tag{6.17}$$

with arguments x and y nonnegative and a positive. The function s is positive, and

$$\lim_{(x,y)\to(0,0)} s(x,y) = 1, \quad \lim_{x\to 0} s(x,y) = \frac{\tanh y}{y}, \quad \lim_{y\to 0} s(x,y) = \frac{x}{\tanh x}, \tag{6.18}$$

and $s(x,y) \approx \frac{x}{y}$ for large x and y. The function z satisfies: $0 < z(x,y,a) \leqslant 1$,

$$z(0,0,0) = 1, \quad \lim_{x\to\infty} z(x,y,a) = 1, \quad \lim_{y\to\infty} z(x,y,a) = 0, \quad \lim_{a\to\infty} z(x,y,a) = 0,$$

and $z \approx e^{-b-y+x}\frac{1+e^{-2x}}{1+e^{-2y}}$ for $b+y-x$ large, where $b := \ln a$. Now,

$$\frac{\widetilde{C}_2(r)}{C_2(i)} = z\left(p_i t_{i+1}, p_{i+1} t_{i+1}, \frac{h_{i+1}}{h_i}\, s\left(\frac{p_i h_i}{2}, \frac{p_{i+1} h_{i+1}}{2}\right)\right),$$

$$\frac{\widetilde{C}_2(r+3)}{C_2(i+1)} = z\left(p_{i+2} t_{i+2}, p_{i+1} t_{i+2}, \frac{h_{i+1}}{h_{i+2}}\, s\left(\frac{p_{i+2} h_{i+2}}{2}, \frac{p_{i+1} h_{i+1}}{2}\right)\right),$$

and (6.16) can be calculated numerically stable. Further, Theorem 3.7 gives T_i^4, the B-spline of Chebyshev tension spline space.

6.3 C^2 tension splines

As we mentioned before, we will treat the C^2 tension splines space as the subspace of $\mathcal{S}(4, d\boldsymbol{\sigma}, \widetilde{\boldsymbol{T}})$.

6.3.1 Quasi–Oslo type algorithm for C^2 tension splines

Let us define

$$\widetilde{\mathfrak{C}}_2(r-1) := \frac{\tanh\left(\frac{p_i h_i}{2}\right)}{p_i}, \tag{6.19}$$

$$\widetilde{\mathfrak{C}}_2(r) := \frac{\tanh\left(\frac{p_i h_i}{2}\right)}{p_i}, \tag{6.20}$$

$$\mathfrak{C}_2(i) := \widetilde{\mathfrak{C}}_2(r) + \widetilde{\mathfrak{C}}_2(r+1).$$

Then, we can also define a variation of Lemma 3.1 and Theorem 3.7 for C^2 tension splines. If

$$\begin{aligned}\gamma^3_{r,i} &:= \frac{\widetilde{\mathfrak{C}}_2(r)}{\mathfrak{C}_2(i)},\\ \gamma^3_{r+1,i} &:= 1,\\ \gamma^3_{r+2,i} &:= \frac{\widetilde{\mathfrak{C}}_2(r+3)}{\mathfrak{C}_2(i+1)},\end{aligned}$$

and

$$\gamma^4_{r,i} := \frac{\gamma^3_{r,i}\widetilde{C}_3(r)}{\mathfrak{C}_3(i)}, \tag{6.21}$$

$$\gamma^4_{r+1,i} := \frac{\gamma^3_{r,i}\widetilde{C}_3(r) + \gamma^3_{r+1,i}\widetilde{C}_3(r+1)}{\mathfrak{C}_3(i)}, \tag{6.22}$$

$$\gamma^4_{r+2,i} := \frac{\gamma^3_{r+3,i+1}\widetilde{C}_3(r+3) + \gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{\mathfrak{C}_3(i+1)}, \tag{6.23}$$

$$\gamma^4_{r+3,i} := \frac{\gamma^3_{r+4,i+1}\widetilde{C}_3(r+4)}{\mathfrak{C}_3(i+1)}, \tag{6.24}$$

with

$$\mathfrak{C}_3(i) := \gamma^3_{r,i}\widetilde{C}_3(r) + \widetilde{C}_3(r+1) + \gamma^3_{r+2,i}\widetilde{C}_3(r+2), \tag{6.25}$$

then

$$\mathfrak{T}^4_i := \sum_{j=r}^{r+3} \gamma^4_{j,i}\widetilde{T}^4_j$$

make B-splines for C^2 tension splines space according to [16]. Straightforward calculation can prove that $\mathfrak{T}^4_i$ is of class C^2.

6.3.2 Quasi–derivative formula for C^2 tension splines

Also, the variation of derivative formula (2.27) holds. Let

$$\mathfrak{T}^3_i := \frac{\widetilde{\mathfrak{C}}_2(r)}{\mathfrak{C}_2(i)}\,\widetilde{T}^3_r + \widetilde{T}^3_{r+1} + \frac{\widetilde{\mathfrak{C}}_2(r+3)}{\mathfrak{C}_2(i+1)}\,\widetilde{T}^3_{r+2}, \tag{6.26}$$

then obviously

$$\mathfrak{C}_3(i) = \int_{t_i}^{t_{i+3}} \mathfrak{T}^3_i(\tau_2)\,d\tau_2,$$

and easy calculation shows that $\mathfrak{T}^3_i$ is of class C^1 and

$$D\mathfrak{T}^4_i = \frac{\mathfrak{T}^3_i}{\mathfrak{C}_3(i)} - \frac{\mathfrak{T}^3_{i+1}}{\mathfrak{C}_3(i+1)}. \tag{6.27}$$

Now, let

$$\widetilde{\mathfrak{T}}_r^2(x) := \begin{cases} \dfrac{\sinh\left(p_i(x-t_i)\right)}{\sinh\left(p_i h_i\right)}, & \text{for } x \in [t_i, t_{i+1}), \\ 0 & \text{otherwise,} \end{cases} \tag{6.28}$$

$$\widetilde{\mathfrak{T}}_{r+1}^2(x) := \begin{cases} \dfrac{\sinh\left(p_{i+1}(t_{i+2}-x)\right)}{\sinh\left(p_{i+1} h_{i+1}\right)}, & \text{for } x \in [t_{i+1}, t_{i+2}), \\ 0 & \text{otherwise,} \end{cases} \tag{6.29}$$

and

$$\mathfrak{T}_i^2 = \widetilde{\mathfrak{T}}_r^2 + \widetilde{\mathfrak{T}}_{r+1}^2. \tag{6.30}$$

Then it is easy to see that $\mathfrak{T}_i^2$ is continuous,

$$\widetilde{\mathfrak{C}}_2(r) = \int_{\tilde{t}_r}^{\tilde{t}_{r+2}} \widetilde{\mathfrak{T}}_r^2(\tau_3)\, d\tau_3, \qquad \widetilde{\mathfrak{C}}_2(r+1) = \int_{\tilde{t}_{r+1}}^{\tilde{t}_{r+3}} \widetilde{\mathfrak{T}}_{r+1}^2(\tau_3)\, d\tau_3,$$

and

$$D\mathfrak{T}_i^3 = \frac{\mathfrak{T}_i^2}{\mathfrak{C}_2(i)} - \frac{\mathfrak{T}_{i+1}^2}{\mathfrak{C}_2(i+1)}. \tag{6.31}$$

For the simplicity, the linear combination of $\mathfrak{T}_i^3$ ($\mathfrak{T}_i^2$) we shall call just a tension spline of order 3 (2).

For numerical stability, next to the functions defined in section 6.1, we need

$$w(x,y,p,q) := \frac{1}{1 + \frac{y}{x}\, s\left(\frac{px}{2}, \frac{qy}{2}\right)}, \tag{6.32}$$

with s defined in (6.17). Then $0 < w(x,y,p,q) \leqslant 1$,

$$\lim_{y\to 0} w(x,y,p,q) = 1, \quad \lim_{x\to 0} w(x,y,p,q) = 0, \quad \lim_{p,q\to 0} w(x,y,p,q) = \frac{x}{x+y}, \tag{6.33}$$

for $x \neq 0$, so w is not continuous at $(0,0,0,0)$, but as with the function u, this is not a problem, since in this case w is multiplied by zero. Now

$$\frac{\widetilde{\mathfrak{C}}_2(r)}{\mathfrak{C}_2(i)} = w(h_i, h_{i+1}, p_i, p_{i+1}), \qquad \frac{\widetilde{\mathfrak{C}}_2(r+3)}{\mathfrak{C}_2(i+1)} = w(h_{i+2}, h_{i+1}, p_{i+2}, p_{i+1}),$$

and the rest can be stably calculated with previous functions.

6.3.3 Derivatives of C^2 tension spline

As in the case of C^1 tension splines, we can also calculate derivatives of C^2 tension spline. Let $s = \sum_j c_j \mathfrak{T}_j^4$ and $\bar{t} \in [t_i, t_{i+1})$. Then

$$s(\bar{t}) = \sum_{j=i-3}^{i} c_j \mathfrak{T}_j^4(\bar{t}).$$

The first derivative is equal to

$$Ds(\bar{t}) = \sum_{j=i-2}^{i} c_j^1 \mathfrak{T}_j^3(\bar{t}),$$

where

$$c_j^1 = \frac{c_j - c_{j-1}}{\mathfrak{C}_3(j)},$$

for $j = i-2, i-1, i$, by (6.27). The second derivative is

$$D^2 s(\bar{t}) = \sum_{j=i-1}^{i} c_j^2 \mathfrak{T}_j^2(\bar{t}),$$

with

$$c_j^2 = \frac{c_j^1 - c_{j-1}^1}{\mathfrak{C}_2(j)},$$

for $j = i-1, i$ by (6.31).

The numerical stability in these algorithms is achieved by using only positive linear (most often convex) combinations of positive values, and special functions calculated up to the computer accuracy.

Chapter 7

Cycloidal splines

The last example of applications of knot insertion algorithms are cycloidal splines (also called helix splines). We can notice that a lot of shapes of the things are made from straight lines or circle arcs. Also, in the industry, sometimes the computer guided machines have to cut shapes like circles or helices. Therefore, it is useful to have curves piecewisely spanned by linear polynomials, sine and cosine, *i.e.* cycloidal curves [5, 14, 15, 28], and of course a stable algorithm for calculating with them.

7.1 Equidistant cycloidal splines

Let $\Delta = \{x_i\}_{i=0}^{l+1}$ be a partition of $[0, (l+1)\frac{\pi}{2}]$, such that $x_i = i\frac{\pi}{2}$, and let $\mathrm{Cs} : \mathbb{R} \to \mathbb{R}$ be defined by

$$\mathrm{Cs(x)} := \cos\left(\mathrm{x} - \frac{\pi}{4} - \mathrm{i}\frac{\pi}{2}\right)$$

for $x \in [i\frac{\pi}{2}, (i+1)\frac{\pi}{2}]$, $i \in \mathbb{Z}$. Cs is continuous, periodical extension of $\cos(x - \frac{\pi}{4})|_{[0,\frac{\pi}{2}]}$. Consider a CCC–system on $\mathbb{R}$:

$$\begin{aligned}
u_1(x) &= 1, \\
u_2(x) &= \int_0^x d\tau_2, \\
u_3(x) &= \int_0^x d\tau_2 \int_0^{\tau_2} \mathrm{Cs}(\tau_3)\,\mathrm{d}\tau_3, \\
u_4(x) &= \int_0^x d\tau_2 \int_0^{\tau_2} \mathrm{Cs}(\tau_3)\,\mathrm{d}\tau_3 \int_0^{\tau_3} \frac{1}{\mathrm{Cs}^2(\tau_4)}\,\mathrm{d}\tau_4,
\end{aligned}$$

and the associated generalized derivatives:

$$\begin{aligned}
L_{(1,d\boldsymbol{\sigma})} &= D, \\
L_{(2,d\boldsymbol{\sigma})} &= \frac{1}{\mathrm{Cs}} D^2, \\
L_{(3,d\boldsymbol{\sigma})} &= \mathrm{Cs}^2 \mathrm{D} \frac{1}{\mathrm{Cs}} \mathrm{D}^2, \\
L_{(4,d\boldsymbol{\sigma})} &= \frac{1}{\mathrm{Cs}} D\, \mathrm{Cs}^2 \mathrm{D} \frac{1}{\mathrm{Cs}} \mathrm{D}^2.
\end{aligned}$$

One easily verifies that $\mathrm{span}\{u_1, u_2, u_3, u_4\} = \mathrm{span}\{1, x, \sin x, \cos x\}$ on each interval $[i\frac{\pi}{2}, (i+1)\frac{\pi}{2}]$.

Let $\bar{t} \in (t_i, t_{i+1})$ where $\boldsymbol{T}$, $\bar{\boldsymbol{T}}$, $\widetilde{\boldsymbol{T}}$ and $\widehat{\boldsymbol{T}}$, and the rest of the notation as in subsection 3.4.2 ($\boldsymbol{T}$ has all interior knots of multiplicity one). Then by Lemma 3.1

$$C_1(j) = \frac{\sin h_j}{\mathrm{Cs(t_j)Cs(t_{j+1})}}, \tag{7.1}$$

$$C_2(j) = \mathrm{Cs(t_{j+1})}\frac{\sin\frac{\mathrm{h_j}+\mathrm{h_{j+1}}}{2}}{\cos\frac{\mathrm{h_j}}{2}\cos\frac{\mathrm{h_{j+1}}}{2}}, \tag{7.2}$$

$$\begin{aligned} C_3(j) = \cos\frac{h_{j+1}}{2}\Bigg[&\frac{\sin\frac{h_j}{2}}{\sin\frac{h_j+h_{j+1}}{2}}\left(\frac{h_j-\sin h_j}{2\sin^2\frac{h_j}{2}}+\frac{h_{j+1}-\sin h_{j+1}}{2\sin^2\frac{h_{j+1}}{2}}\right)\\ &+\frac{2}{\sin^2\frac{h_{j+1}}{2}}\left(\sin\frac{h_{j+1}}{2}-\frac{h_{j+1}}{2}\cos\frac{h_{j+1}}{2}\right)\\ &+\frac{\sin\frac{h_{j+2}}{2}}{\sin\frac{h_{j+1}+h_{j+2}}{2}}\left(\frac{h_{j+1}-\sin h_{j+1}}{2\sin^2\frac{h_{j+1}}{2}}+\frac{h_{j+2}-\sin h_{j+2}}{2\sin^2\frac{h_{j+2}}{2}}\right)\Bigg]. \end{aligned} \tag{7.3}$$

The cycloidal splines can now be calculated with the de Boor algorithm (3.20). As in the case of the tension splines, in order to avoid redundant operations and for numerical stability, instead of C_1, C_2, C_3 we define functions $CC1$, $CC2$ and $CC3$:

$$\begin{aligned} CC1(x) &:= \sin\frac{x}{2}, \qquad CC2(x,y) := \sin\frac{x+y}{2}, \\ CC3(x,y,z) &:= \frac{1}{2}u\left(\frac{x}{2},\frac{y}{2}\right)(f(x)+f(y)) + 2g(\frac{y}{2}) + \frac{1}{2}u\left(\frac{z}{2},\frac{y}{2}\right)(f(y)+f(z)), \end{aligned} \tag{7.4}$$

with

$$f(x) := \frac{x-\sin x}{\sin^2\frac{x}{2}}, \quad g(x) := \frac{\sin x - x\cos x}{\sin^2 x}, \quad u(x,y) := \frac{\sin x}{\sin(x+y)}. \tag{7.5}$$

The function f has to be evaluated only on interval $[0, \frac{\pi}{2}]$, with $\lim_{x\to 0} f(x) = 0$, so it can be nicely approximated by Padé approximant on the whole interval. The same is true for g, which we only need on interval $[0, \frac{\pi}{4}]$, also with $\lim_{x\to 0} g(x) = 0$. Because

$$\lim_{x\to 0} u(x,y) = 0, \qquad \lim_{y\to 0} u(x,y) = 1,$$

the function u is not continuous at $(0,0)$, like the function u from tensions splines, but from (7.4) it is obvious that $u(0,0)$ is then multiplied by zero. Further the algorithm proceeds in the same manner as the one for C^1 tension splines.

To give an example of application of the algorithm, we will use elementary functions x, $\sin x$ and $\cos x$. For that, we need their de Boor points, on interval $[0, (l+1)\frac{\pi}{2}]$ ($t_1 = -\frac{3\pi}{2}$, $t_2 = -\pi$, $t_3 = -\frac{\pi}{2}$, $t_4 = 0$ and $t_{l+5} = (l+1)\frac{\pi}{2}$, $t_{l+6} = (l+2)\frac{\pi}{2}$, $t_{l+7} = (l+3)\frac{\pi}{2}$, $t_{l+8} = (l+4)\frac{\pi}{2}$):

$$\begin{aligned}
\mathbf{x}: &\quad c_i = (i-2)\frac{\pi}{2}, \qquad i = 1, 2, \ldots, l+4, \\
\mathbf{\sin x}: &\quad c_{4i+1} = -\frac{\pi}{2}, \quad c_{4i+2} = c_{4(i+1)} = 0, \quad c_{4i+3} = \frac{\pi}{2}, \qquad i = 0, 1, \ldots, \\
\mathbf{\cos x}: &\quad c_{4i+1} = c_{4i+3} = 0, \quad c_{4i+2} = \frac{\pi}{2}, \quad c_{4(i+1)} = -\frac{\pi}{2}, \qquad i = 0, 1, \ldots .
\end{aligned}$$

Because some of de Boor points alternate in sign, catastrophic cancellation can occur in some cases. Also, the continuity of the second generalized derivative is probably unnecessary condition. There is no such difficulty if we use the Bezier variant of cycloidal splines.

This algorithm can also be carried out for $\bar{t} = t_i$ and even with extended partition with knots of arbitrary multiplicities, just by applying Theorem 2.19 or Remark 2.7 in (7.1), (7.2) and (7.3).

7.2 Equidistant Bezier cycloidal splines

We will, nevertheless, proceed with the Bezier case. Let $\bar{t} \in (x_i, x_{i+1}) \subset [0, (l+1)\frac{\pi}{2}]$, and let $\boldsymbol{T}$, $\bar{\boldsymbol{T}}$, $\widetilde{\boldsymbol{T}}$ and $\widehat{\boldsymbol{T}}$ be extended partitions of the interval $[0, (l+1)\frac{\pi}{2}]$ whose all interior knots, except maybe one, have multiplicity 4: $\boldsymbol{T} = \{t_j\}_{j=1}^{4(l+2)}$ where

$$t_{4j+1} = t_{4j+2} = t_{4j+3} = t_{4(j+1)} = x_j \quad \text{for } j = 0, \ldots, l+1;$$

$\bar{\boldsymbol{T}} = \{\bar{t}_j\}_{j=1}^{4(l+2)+1}$ where

$$\begin{aligned}
\bar{t}_j &= t_j \quad \text{for } j = 1, \ldots, 4(i+1), \\
\bar{t}_{4(i+1)+1} &= \bar{t}, \\
\bar{t}_j &= t_{j-1} \quad \text{for } j = 4(i+1)+2, \ldots, 4(l+2)+1;
\end{aligned}$$

$\widetilde{\boldsymbol{T}} = \{\tilde{t}_j\}_{j=1}^{4(l+2)+2}$ where

$$\begin{aligned}
\tilde{t}_j &= t_j \quad \text{for } j = 1, \ldots, 4(i+1), \\
\tilde{t}_{4(i+1)+1} &= \tilde{t}_{4(i+1)+2} = \bar{t}, \\
\tilde{t}_j &= t_{j-2} \quad \text{for } j = 4(i+1)+3, \ldots, 4(l+2)+2;
\end{aligned}$$

and finally, $\widehat{\boldsymbol{T}} = \{\hat{t}_j\}_{j=1}^{4(l+2)+3}$, where

$$\begin{aligned}
\hat{t}_j &= t_j \quad \text{for } j = 1, \ldots, 4(i+1), \\
\hat{t}_{4(i+1)+1} &= \hat{t}_{4(i+1)+2} = \hat{t}_{4(i+1)+3} = \bar{t}, \\
\hat{t}_j &= t_{j-3} \quad \text{for } j = 4(i+1)+4, \ldots, 4(l+2)+3.
\end{aligned}$$

We use the notation $\bar{T}_j^k$, $\widetilde{T}_j^k$, $\widehat{T}_j^k$, $\bar{C}_{k-1}(j)$, $\widetilde{C}_{k-1}(j)$, $\widehat{C}_{k-1}(j)$ for B-splines and their integrals accordingly. These integrals are then calculated from (7.1), (7.2) and (7.3) according to Theorem 2.19 or Remark 2.7 by coalescing the knots. For computer implementation, we again use the de Boor algorithm together with functions defined in (7.4), only now more arguments are equal to zero.

Given splines are generalized Bezier splines, and analogous properties are valid: if we observe an interval $[x_i, x_{i+1}] = [i\frac{\pi}{2}, (i+1)\frac{\pi}{2}]$, and a parameterized curve in $\mathbb{R}^2$:

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} a_x T^4_{4i+1}(t) + b_x T^4_{4i+2}(t) + c_x T^4_{4i+3} + d_x T^4_{4(i+1)}(t) \\ a_y T^4_{4i+1}(t) + b_y T^4_{4i+2}(t) + c_y T^4_{4i+3} + d_y T^4_{4(i+1)}(t) \end{bmatrix}$$

on a given interval, for some index i and de Boor points

$$A = \begin{bmatrix} a_x \\ a_y \end{bmatrix}, \quad B = \begin{bmatrix} b_x \\ b_y \end{bmatrix}, \quad C = \begin{bmatrix} c_x \\ c_y \end{bmatrix}, \quad D = \begin{bmatrix} d_x \\ d_y \end{bmatrix},$$

then the curve passes through points A and D, *i.e.*

$$\begin{bmatrix} x(x_i) \\ y(x_i) \end{bmatrix} = A, \qquad \begin{bmatrix} x(x_{i+1}) \\ y(x_{i+1}) \end{bmatrix} = D,$$

and derivatives at the end points, by the derivative formula (2.27), since $L_{(1,d\boldsymbol{\sigma})} = D$, are

$$\begin{bmatrix} \dot{x}(x_i) \\ \dot{y}(x_i) \end{bmatrix} = \frac{1}{C_3(4i+1)}(B - A), \qquad \begin{bmatrix} \dot{x}(x_{i+1}) \\ \dot{y}(x_{i+1}) \end{bmatrix} = \frac{1}{C_3(4i+3)}(D - C).$$

When we, for example, want to derive de Boor points for x, $\sin x$ and $\cos x$, we get:

$$\begin{aligned} \mathbf{x}: &\quad d_0 = 0,\ d_1 = \tfrac{\pi}{2} - 1,\ d_2 = 1,\ d_3 = \tfrac{\pi}{2}, \\ \mathbf{\sin x}: &\quad d_0 = 0,\ d_1 = \tfrac{\pi}{2} - 1,\ d_2 = 1,\ d_3 = 1, \\ \mathbf{\cos x}: &\quad d_0 = 1,\ d_1 = 1,\ d_2 = \tfrac{\pi}{2} - 1,\ d_3 = 0. \end{aligned}$$

This time we do not have any problems with catastrophic cancellations, because all of the de Boor points are nonnegative.

7.3 Nonequidistant Bezier cycloidal splines

Further, a generalization can be made in sense that, instead of taking an equidistant partition, we take an arbitrary partition $\{x_i\}_{i=0}^{l+1}$ of $[a, b]$, such that $0 < h_i = x_{i+1} - x_i \leqslant \frac{\pi}{2}$, and multiplicity vector $\boldsymbol{m}^{(4)} = (4, \ldots, 4)$. Now, Cs : [a, b] $\to \mathbb{R}$ is defined by Cs(x) := $\cos(\mathrm{x} - \frac{\pi}{4} - \mathrm{x_i})$ for $x \in [x_i, x_{i+1})$, $i = 0, \ldots, l-1$ or $x \in [x_l, x_{l+1}]$ for $i = l$. Generally, Cs is not continuous.

The algorithm from previous section can be easily modified for this case: we only need to replace the knots $i\frac{\pi}{2}$ with given x_i in (7.1),(7.2) and (7.3). We can again apply this algorithm to express functions x, $\sin x$, $\cos x$, $x - \sin x$ and $1 - \cos x$ on interval $[0, z]$, $z \leqslant \frac{\pi}{2}$ (with $t_1 = 0$), where their de Boor points are:

$$
\begin{aligned}
\mathbf{x}: &\quad d_0 = 0,\ d_1 = \frac{z - \sin z}{2\sin^2 \frac{z}{2}},\ d_2 = \frac{\sin z - z\cos z}{2\sin^2 \frac{z}{2}},\ d_3 = z,\\
\mathbf{\sin x}: &\quad d_0 = 0,\ d_1 = \frac{z - \sin z}{2\sin^2 \frac{z}{2}},\ d_2 = \frac{\sin z - z\cos z}{2\sin^2 \frac{z}{2}},\ d_3 = \sin z,\\
\mathbf{\cos x}: &\quad d_0 = 1,\ d_1 = 1,\ d_2 = z\cot\frac{z}{2} - 1,\ d_3 = \cos z,\\
\mathbf{x - \sin x}: &\quad d_0 = 0,\ d_1 = 0,\ d_2 = 0,\ d_3 = z - \sin z,\\
\mathbf{1 - \cos x}: &\quad d_0 = 0,\ d_1 = 0,\ d_2 = 2 - z\cot\frac{z}{2},\ d_3 = 1 - \cos z.
\end{aligned}
$$

All of these expressions can be stably calculated as in (7.5).

7.4 The choice of CCC–systems

In this section we just want to comment the choice of the CCC–systems in Sections 7.1 and 7.3. We know that ECC–system for $\mathrm{span}(1, x, \sin x, \cos x)$ exists on interval only of length less than 2π (although there exists an ECC–space on interval $[a, a{+}2\pi]$ which contains 1, x, $\sin x$ and $\cos x$ but, its dimension is higher than 4, (see [5])). We want to overcome this restriction by substituting ECC–system with the CCC–system. The choice of the length of the basic subinterval $h_i = \frac{\pi}{2}$ and of the function Cs in Sections 7.1 and 7.2, is purely for obtaining a nice CCC–system and reduced systems in the sense of easy calculation and numerical stability, and in Section 7.3 we put the restriction $h_i \leqslant \frac{\pi}{2}$ because the algorithm from Section 7.2 can be very easily generalized for that case.

The problem of calculating cycloidal splines has already attracted a lot of attention, and although in [15, 21, 28] algorithms for calculating with cycloidal splines have been developed, the explicit numerically stable algorithms for practical use have not yet been discussed, what we have, in our case, presented in this chapter.

Chapter 8

Program codes

In this thesis we have given the algorithms for calculating with several kinds of splines. For all of them we have made the program codes, but for brevity, we will list here only the program codes of subroutines involved in calculating the tension splines, as well as their first and second derivatives. The subroutines are written in Fortran 90, and the algorithms for them are described in Chapter 6. All the program codes of the routines, as well as the examples, can be found on the enclosed CD.

The routines are divided in two groups: the first is for calculating with C^1 tension splines, and the second with C^2. To avoid confusion in notations in the comments of the subroutines, when we refer to C^1 tension splines, we call them Chebyshev tension splines, while when we refer to C^2 tension splines, we simply call them tension splines. What both groups have in common is the use of the module `functions`, which contains all auxiliary and elementary functions needed in calculations:

`f`, `g` and `u` correspond to the functions having the same names defined in (6.5), (6.6) and (6.7). To be precise, `f(x)` $= f($`x`$)$, `g(x)` $= g($`x`$)$ and `u(h,k,p)`$=$ $u($`p`$\cdot$`h`,`p`$\cdot$`k`$)$. We calculate `f` and `g` by Padé approximation for small `x`, and for large `x`, *i.e.* large `p`$\cdot$`h` and `p`$\cdot$`k`, we use limits and asymptotic behavior of all three functions, as described in Subsection 6.1.1;

`CC3` is needed for calculating the integrals of B-splines associated with the first reduced system of C^1 tension splines, *i.e.* `CC3` calculates the function CC_3 from (6.8);

`divth1` and `divth2` calculate `divth1(h,k,p)`$= \dfrac{\tanh(\mathtt{p}\cdot\mathtt{h})}{\tanh(\mathtt{p}\cdot\mathtt{k})}$ and `divth2(h,k,l,p)` $= \dfrac{\tanh(\mathtt{p}\cdot\mathtt{h})}{\tanh(\mathtt{p}\cdot\mathtt{h})+\tanh(\mathtt{p}\cdot\mathtt{k})}$, with `l`=`h`+`k`, needed for (6.9), (6.10) and (6.11). In these functions, limits (6.12) are used;

`s` corresponds to the function having the same name in (6.17), again, its limits (6.18) are used;

`KC2div` calculates the function w defined in (6.32) and uses (6.33);

`KC2` is needed for (6.19) and (6.20), `KC2(x)`= $\dfrac{\tanh(x)}{x}$;

`KC3` derives $\mathfrak{C}_3(i)$ from (6.25), where the call of the function has the arguments `KC3(h,p)`, with `h` and `p` being arrays of length 3: `h(1)`= h_i, `h(2)`= h_{i+1}, `h(3)`= h_{i+2}, and `p(1)`= p_i, `p(2)`= p_{i+1}, `p(3)`= p_{i+2}.

`gamma4, gamma4d1` **and** `gamma4d4` calculate the coefficients of the quasi–Oslo type algorithm which connects C^2 tension B-splines to C^1 tension B-splines: the subroutine `gamma4` produces all four coefficients (6.21)–(6.24), while the function `gamma4d1` only the first one (6.21) and `gamma4d4` only the last one (6.24). A call of the subroutine is `gamma4(h,p,d)`, where all three arguments are arrays of 4 elements: `h(1)`= h_i, `h(2)`= h_{i+1}, `h(3)`= h_{i+2}, `h(4)`= h_{i+3}; `p(1)`= p_i, `p(2)`= p_{i+1}, `p(3)`= p_{i+2}, `p(4)`= p_{i+3} and the output argument `d` contains `d(1)`= $\gamma^4_{r,i}$, `d(2)`= $\gamma^4_{r+1,i}$, `d(3)`=$\gamma^4_{r+2,i}$, `d(4)`= $\gamma^4_{r+3,i}$. In the call of the function `gamma4d1(h,p)` the arrays `h` and `p` contain only h_i, h_{i+1}, h_{i+2}, p_i, p_{i+1}, p_{i+2}, while in `gamma4d4(h,p)` h_{i+1}, h_{i+2}, h_{i+3}, p_{i+1}, p_{i+2}, p_{i+3}.

The notation above is the same as in Chapter 6, and the functions that do not have their arguments listed here have the same arguments as in Chapter 6. Each of the following subroutines uses the module `functions`. Some of the subroutines use some other, as represented in Table 8.1:

subroutine	...calls the subroutine:
`tension4C1`	
`tension3C1`	
`tension2`	
`tension1der4C1`	`tension3C1`
`tension2der4C1`	`tension2`
`tension1der4C1coeff`	
`tension1der3C1coeff`	
`tension4C2`	`tension4C1`
`tension3C2`	`tension3C1`
`tension1der4C2`	`tension3C2`
`tension2der4C2`	`tension2`
`tension1der4C2coeff`	
`tension1der3C2coeff`	

Table 8.1: The subroutines for calculating C^1 and C^2 tension splines and their derivatives

In the comments of the subroutines, we also mention an auxiliary term: `dimx`, which we do not use explicitly in most of the subroutines. It is put here just to help the user to define the dimensions of the actual parameters in the call of the subroutines.

8.1 C^1 tension splines

Although the comments in subroutines describe the input and output variables, few things need detailed explanations. C^1 tension splines are associated with an extended partition with all interior knots of multiplicity two, and the boundary points of multiplicity four, but the algorithms are developed in such a way that it is not necessary to remember each interior knot twice. Therefore the array `xx` in the following routines has each interior knot saved only once, and for consistence, both boundary points are saved twice. The points in the array `xx` represent modified partition of the given interval (not the extended one). The array `px` contains the tension parameters of each subinterval defined by `xx`, but as the first and the last subinterval are trivial, the first and the last elements of `px` do not matter, and can therefore be arbitrary. Further, the routines are designed so that they can calculate the value of a parameterized spline curve whose de Boor points have dimension `dim`. It is also important to emphasize that the de Boor points given in `cx` have numeration that corresponds to the extended partition, and not to the `xx`.

The subroutine `tension4C1` calculates the value of the spline curve defined with de Boor points `cx` at the point `x`, by the generalized de Boor algorithm described in Subsection 6.1.1:

```
    subroutine tension4C1 (x,xx,px,ileft,cx,dim,spl)

!
!   Generalized de Boor algorithm for calculating Chebyshev tension
!   spline curve
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               spline curve is evaluated
!       xx    = rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
```

```
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4, the array of de Boor points associated
!               with the extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the Chebyshev tension spline curve
!               at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision, parameter :: zero=0.0d0
    double precision :: t(-1:2),p(-1:1),c(dim,-3:0) ! t(0)<=x<=t(1)
    double precision :: temp1,temp2,temp3,temp4,h(-1:1),xa,xb,u1,u2
    double precision :: C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12
    integer :: ileftc,i

!
!   Storing only relevant data
!
    t(-1:2)=xx(ileft-1:ileft+2)
    p(-1:1)=px(ileft-1:ileft+1)
    ileftc=2*ileft
    c(1:dim,-3:0)=cx(1:dim,ileftc-3:ileftc)
!
!   Calculating the length of the subintervals involved
!
    do i=-1,1
        h(i)=t(i+1)-t(i)
    enddo
```

```
!
!   If x is equal to the left end point of subinterval
!   [xx(ileft),xx(ileft+1)], then the point x must be inserted only
!   once...
!
    if (x==t(0)) then

        if (t(-1)==t(0)) then
            spl=c(1:dim,-3)
            return
        endif

        temp1=CC3(zero,zero,h(0),zero,zero,p(0))
        temp2=CC3(h(-1),zero,zero,p(-1),zero,zero)
        spl=(c(1:dim,-3)*temp1+c(1:dim,-2)*temp2)/ &
            CC3(h(-1),zero,h(0),p(-1),zero,p(0))
        return
!
!   ... and also if x is equal to the right end point
!
    elseif (x==t(1)) then

        if (t(1)==t(2)) then
            spl=c(1:dim,0)
            return
        endif

        temp1=CC3(zero,zero,h(1),zero,zero,p(1))
        temp2=CC3(h(0),zero,zero,p(0),zero,zero)
        spl=(c(1:dim,-1)*temp1+c(1:dim,0)*temp2)/ &
            CC3(h(0),zero,h(1),p(0),zero,p(1))
        return

    endif
!
!   If x is in the interior of [xx(ileft),xx(ileft+1)], then the
!   generalized de Boor algorithm of order 4 is performed, where the
!   point x is inserted 3 times, but first, 12 integrals of
!   B-splines of order 3 are calculated
!
    xa=x-t(0)
    xb=t(1)-x
    u1=u(xb/2,xa/2,p(0))
    u2=u(xa/2,xb/2,p(0))
```

```
    C1=CC3(zero,zero,xb,zero,zero,p(0))
    C2=CC3(h(-1),zero,h(0),p(-1),zero,p(0))
    C3=CC3(zero,xa,xb,zero,p(0),p(0))
    C4=CC3(zero,xa,zero,zero,p(0),zero)
    C5=CC3(xa,xb,zero,p(0),p(0),zero)
    C6=CC3(xa,zero,xb,p(0),zero,p(0))
    C7=CC3(zero,h(0),zero,zero,p(0),zero)
    C8=CC3(xa,zero,zero,p(0),zero,zero)
    C9=CC3(zero,xb,zero,zero,p(0),zero)
    C10=CC3(h(0),zero,h(1),p(0),zero,p(1))
    C11=CC3(h(-1),zero,xa,p(-1),zero,p(0))
    C12=CC3(xb,zero,h(1),p(0),zero,p(1))
!
!   Calculation of 4 nontrivial B-splines at the point x
!
    temp1=u1*C1/C2
    temp2=C1*C11/(C3*C2) + (C1*C4*C5/C3 + C8*C9)/(C6*C7)
    temp3=(C1*C4 + C8*C9*C3/C5)/(C6*C7) + C8*C12/(C5*C10)
    temp4=u2*C8/C10
!
!   The final value of the Chebyshev tension spline curve
!   represented as the linear combination of B-splines
!
    spl=u1*(exp(-p(0)*xa)*c(1:dim,-3)*temp1+c(1:dim,-2)*temp2)+ &
        u2*(c(1:dim,-1)*temp3+exp(-p(0)*xb)*c(1:dim,0)*temp4)


    end subroutine tension4C1
```

If we want to calculate the value of the first derivative of a spline curve, first we need to be able to calculate the splines associated with the first reduced system as in Subsection 6.1.2. The subroutine `tension3C1` performs this task. The extended partition associated with these splines is the same as for the splines of order 4:

```
    subroutine tension3C1 (x,xx,px,ileft,cx,dim,spl)

!
!   Generalized de Boor algorithm for calculating the third order
!   Chebyshev tension spline curve, i.e. a spline curve associated
!   with the first reduced system of Chebyshev tension splines
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               spline curve is evaluated
!       xx    = rank-one double precision array of size dimx, the
```

```
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4, the array of de Boor points of the third
!               order Chebyshev tension spline, associated with the
!               extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+3)], so cx(1:dim,1) can be
!               arbitrary
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the spline curve associated with
!               the first reduced system at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision :: t(0:1),p,c(dim,-2:0)    !t(0)<=x<=t(1)
    double precision :: temp1,temp2,temp3,h,xa,xb,temp
```

```
    integer :: ileftc

!
!   Storing only relevant data
!
    t(0:1)=xx(ileft:ileft+1)
    p=px(ileft)
    ileftc=2*ileft
    c(1:dim,-2:0)=cx(1:dim,ileftc-2:ileftc)
!
!   Calculating the length of the subintervals involved
!
    h=t(1)-t(0)
    xa=x-t(0)
    xb=t(1)-x
!
!   If x is equal to the left end point of subinterval
!   [xx(ileft),xx(ileft+1)], then the value of the spline at x is
!   equal to the de Boor point cx(ileft-2)...
!
    if (x==t(0)) then
        spl=c(1:dim,-2)
        return
!
!   ... and also if x is equal to the right end point, then the
!   spline is equal to the cx(ileft)
!
    elseif (x==t(1)) then
        spl=c(1:dim,0)
        return
    endif
!
!   If x is in the interior of [xx(ileft),xx(ileft+1)], then the
!   generalized de Boor algorithm of order 3 is performed, where the
!   point x is inserted 2 times
!
    temp=divth1(xb/2,h/2,p)
    temp1=exp(-p*xa)*u(xb,xa,p)*temp
    temp2=2*temp*divth2(xa/2,xb/2,h/2,p)
    temp3=exp(-p*xb)*u(xa,xb,p)*divth1(xa/2,h/2,p)
!
!   The final value of the spline curve
!
    spl=c(1:dim,-2)*temp1+c(1:dim,-1)*temp2+c(1:dim,0)*temp3
```

```
    end subroutine tension3C1
```

Now we can calculate the first derivative of a tension spline curve like in Subsection 6.1.3:

```
    subroutine tension1der4C1 (x,xx,px,ileft,cx,dim,spl)

!
!   Calculating the first derivative of the Chebyshev tension spline
!   curve
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               first derivative of the spline curve is evaluated
!       xx    = rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4, the array of de Boor points of the
!               Chebyshev tension spline, associated with the
!               extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)
!
!   OUTPUT:
```

```
!       spl   = rank-one double precision array of size dim, the
!               output value of the first derivative of the
!               Chebyshev tension spline curve at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision, parameter :: zero=0.0d0
    double precision :: t(-1:2),p(-1:1),c(dim,-3:0) ! t(0)<=x<=t(1)
    double precision :: h(-1:1)
    integer :: ileftc,i

!
!   Storing only relevant data
!
    t(-1:2)=xx(ileft-1:ileft+2)
    p(-1:1)=px(ileft-1:ileft+1)
    ileftc=2*ileft
    c(1:dim,-3:0)=cx(1:dim,ileftc-3:ileftc)
!
!   Calculating the length of the subintervals involved
!
    do i=-1,1
        h(i)=t(i+1)-t(i)
    enddo
!
!   Calculating the de Boor points of the first derivative
!
    c(1:dim,0)=(c(1:dim,0)-c(1:dim,-1))/ &
               CC3(h(0),zero,h(1),p(0),zero ,p(1))
    c(1:dim,-1)=(c(1:dim,-1)-c(1:dim,-2))*2/ &
                CC3(zero,h(0),zero,zero,p(0),zero)/ &
                (1.0d0+exp(-p(0)*h(0)))
    c(1:dim,-2)=(c(1:dim,-2)-c(1:dim,-3))/ &
                CC3(h(-1),zero,h(0),p(-1),zero,p(0))
!
!   Calculating the value of the first derivative of the Chebyshev
!   tension spline curve as the linear combination of the B-splines
```

```
!   associated with the first reduced system
!
    call tension3C1 (x,t,p,2,c,dim,spl)


    end subroutine tension1der4C1
```

Sometimes, the user wants to calculate the first derivative of the given spline at many points, for example for drawing the graph of the first derivative, which can be done with the subroutine `tension1der4C1`. In that case, the most of the de Boor points of the first derivative would be calculated several times. To avoid the repetition of the same calculation, we suggest the subroutine `tension1der4C1coeff` which calculates all de Boor points of the first derivative:

```
    subroutine tension1der4C1coeff (xx,px,cx,dimx,dim)

!
!   Calculating all de Boor points of the first derivative of the
!   Chebyshev tension spline curve
!
!   INPUT:
!       xx    = rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       dimx  = integer variable, the size of the array xx
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   INPUT and OUTPUT:
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4;
!               ON INPUT: the array of de Boor points of the
!               Chebyshev tension spline, associated with the
!               extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
```

```
!               support [T(i),T(i+4)]
!               ON OUTPUT: the array of de Boor points of the
!               first derivative of the Chebyshev tension spline;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+3)] (cx(1:dim,1) will not be used)
!

    use functions

    implicit none

    integer, intent(in) :: dimx,dim
    double precision, intent(in) :: xx(dimx),px(dimx-1)
    double precision, intent(inout) :: cx(dim,2*dimx-4)

    double precision, parameter :: zero=0.0d0
    double precision :: h(2*dimx-1),p(2*dimx-1)
    integer :: i

!
!   Calculating the length of all subintervals involved and creating
!   the array of tension parameters corresponding to the extended
!   partition T
!
    do i=1,dimx-1
        h(2*i-1)=zero
        h(2*i)=xx(i+1)-xx(i)
        p(2*i-1)=zero
        p(2*i)=px(i)
    enddo
    h(2*dimx-1)=zero
    p(2*dimx-1)=zero
!
!   Calculating the de Boor points of the first derivative
!
    do i=2*dimx-4,2,-1
        cx(1:dim,i)=(cx(1:dim,i)-cx(1:dim,i-1))*2/ &
                    CC3(h(i),h(i+1),h(i+2),p(i),p(i+1),p(i+2))/ &
                    (1.0d0+exp(-p(i+1)*h(i+1)))
    enddo


    end subroutine tension1der4C1coeff
```

The second derivatives of the C^1 tension splines form a space of splines piece-

wisely spanned by $\sinh(p\,x)$ and $\cosh(p\,x)$, and possibly discontinuous at the knots of the partition. The spline s from such a spline space is calculated (see (6.15)) by

$$\begin{aligned} s(\bar{t}) = {} & c_{r-1}\, e^{-p_i(\bar{t}-t_i)} u(p_i(t_{i+1}-\bar{t}), p_i(\bar{t}-t_i)) \\ & + c_r\, e^{-p_i(t_{i+1}-\bar{t})} u(p_i(\bar{t}-t_i), p_i(t_{i+1}-\bar{t})), \end{aligned}$$

what performs the subroutine `tension2`. We also use the same subroutine for calculating the second order C^2 tension splines, therefore the comments referring to C^1 tension splines will follow after `a)`, and for C^2 tension splines after `b)`.

```
    subroutine tension2 (x,xx,px,ileft,cx,ileftc,dim,spl)

!
!   Calculating the second order a) Chebyshev tension spline or
!                                b) tension spline curves
!
!   INPUT:
!      x      = double precision variable, the point at which the
!               spline curve is evaluated
!      xx     = a) rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4;
!               b) rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!      px     = rank-one double precision array of size
!               a) dimx-1, b) dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               a) px(1) and px(dimx-1) can be arbitrary
!               b) px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!      ileft  = integer variable, the index of the array xx such
!               that x is contained in the subinterval
```

```
!                [xx(ileft),xx(ileft+1)]
!      cx      = a) rank-two double precision array of shape dim by
!                2*dimx-4, the array of de Boor pints of the second
!                order Chebyshev tension spline associated with the
!                extended partition T;
!                cx(1:dim,i) corresponds to the B-spline with the
!                support [T(i),T(i+2)], so cx(1:dim,1) and
!                cx(2:dim,2) can be arbitrary
!                b) rank-two double precision array of shape dim by
!                dimx, the array of de Boor points of the second
!                order tension spline associated with the extended
!                partition xx;
!                cx(1:dim,i) corresponds to the B-spline with the
!                support [xx(i),xx(i+2)], so cx(1:dim,1) and
!                cx(2:dim,2) can be arbitrary
!      ileftc  = a) integer variable, the index in the second
!                dimension of the array cx such that cx(1:dim,ileftc)
!                corresponds to the B-spline with the support
!                [T(ileftc),T(ileftc+2)] where
!                T(ileftc-1)=T(ileftc)=xx(ileft)
!                b) ileftc=ileft
!      dim     = integer variable, the extent of the first dimension
!                in the array cx, assumed to be positive
!
!    AUXILIARY TERM:
!      dimx    = a) size(xx)
!                b) size(xx)-4
!
!    OUTPUT:
!       spl    = rank-one double precision array of size dim, the
!                output value of the second order (Chebyshev)
!                tension spline curve at the point x
!

     use functions

     implicit none

     integer, intent(in) :: ileft,ileftc,dim
     double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
     double precision, intent(out) :: spl(dim)

     double precision :: t(0:1),p,c(dim,-1:0) ! t(0)<=x<=t(1)
     double precision :: h1,h2,temp1,temp2
```

```
!
!   Storing only relevant data
!
    t(0:1)=xx(ileft:ileft+1)
    p=px(ileft)
    c(1:dim,-1:0)=cx(1:dim,ileftc-1:ileftc)
!
!   Calculating the length of the subintervals involved
!
    h1=x-t(0)
    h2=t(1)-x
!
!   Direct calculation of 2 nontrivial B-splines at the point x
!
    temp1=dexp(-p*h1)*u(h2,h1,p)
    temp2=dexp(-p*h2)*u(h1,h2,p)
!
!   The final value of the spline curve
!
    spl=c(1:dim,-1)*temp1+c(1:dim,0)*temp2


    end subroutine tension2
```

The second derivative (see Subsection 6.1.3) of a given C^1 tension spline now can be obtained by `tension2der4C1`:

```
    subroutine tension2der4C1 (x,xx,px,ileft,cx,dim,spl)

!
!   Calculating the second derivative of the Chebyshev tension
!   spline curve
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               second derivative of the spline curve is evaluated
!       xx    = rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
```

```
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4, the array of de Boor points of the
!               Chebyshev tension spline, associated with the
!               extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the second derivative of the
!               Chebyshev tension spline curve at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision, parameter :: zero=0.0d0
    double precision :: t(-1:2),p(-1:1),c(dim,-3:0) ! t(0)<=x<=t(1)
    double precision :: h(-1:1),kc2ph
    integer :: ileftc,i

!
!   Storing only relevant data
!
    t(-1:2)=xx(ileft-1:ileft+2)
    p(-1:1)=px(ileft-1:ileft+1)
```

```
    ileftc=2*ileft
    c(1:dim,-3:0)=cx(1:dim,ileftc-3:ileftc)
!
!   Calculating the length of the subintervals involved
!
    do i=-1,1
        h(i)=t(i+1)-t(i)
    enddo
!
!   Calculating the de Boor points of the first derivative
!
    c(1:dim,0)=(c(1:dim,0)-c(1:dim,-1))/ &
               CC3(h(0),zero,h(1),p(0),zero ,p(1))
    c(1:dim,-1)=(c(1:dim,-1)-c(1:dim,-2))*2/ &
                CC3(zero,h(0),zero,zero,p(0),zero)/ &
                (1.0d0+exp(-p(0)*h(0)))
    c(1:dim,-2)=(c(1:dim,-2)-c(1:dim,-3))/ &
                CC3(h(-1),zero,h(0),p(-1),zero,p(0))
!
!   Calculating the de Boor points of the second derivative
!
    kc2ph=h(0)*KC2(p(0)*h(0)/2)/2
    do i=0,-1,-1
        c(1:dim,i)=(c(1:dim,i)-c(1:dim,i-1))/kc2ph
    enddo
!
!   Calculating the value of the second derivative of the Chebyshev
!   tension spline curve as the linear combination of the second
!   order B-splines
!
    call tension2(x,t,p,2,c(1:dim,-1:0),2,dim,spl)

    end subroutine tension2der4C1
```

Again, if we want to calculate the second derivative of the given spline at many points, to avoid repetitions, as we explained for `tension1der4C1coeff`, we need the subroutine `tension1der3C1coeff` which calculates all de Boor points of the first derivative of the third order Chebyshev tension spline. The subroutine `tension1der3C1coeff` in combination with `tension1der4C1coeff`, gives de Boor points of the second derivative of C^1 tension spline.

```
    subroutine tension1der3C1coeff (xx,px,cx,dimx,dim)

!
!   Calculating all de Boor points of the first derivative of the
```

```
!   third order Chebyshev tension spline curve
!
!   INPUT:
!       xx    = rank-one double precision array of size dimx, the
!               increasing array of knots, assumed to fulfill the
!               conditions:
!               xx(1)=xx(2)<xx(3)<...<xx(i)<xx(i+1)<...<xx(dimx-1)=
!               xx(dimx); xx(3),...,xx(dimx-2) is the partition of
!               [xx(1),xx(dimx)]; the associated extended
!               partition T has all interior knots of multiplicity
!               2, and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx-1, the
!               array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1) and px(dimx-1) can be arbitrary
!       dimx  = integer variable, the size of the array xx
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   INPUT and OUTPUT:
!       cx    = rank-two double precision array of shape dim by
!               2*dimx-4;
!               ON INPUT: the array of de Boor points of the
!               third order Chebyshev tension spline, associated
!               with the extended partition T;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+3)], so cx(1:dim,1) can be
!               arbitrary
!               ON OUTPUT: the array of de Boor points of the
!               first derivative of the third order Chebyshev
!               tension spline;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [T(i),T(i+2)] (cx(1:dim,1) and cx(1:dim,2)
!               will not be used)
!

    use functions

    implicit none

    integer, intent(in) :: dimx,dim
    double precision, intent(in) :: xx(dimx),px(dimx-1)
    double precision, intent(inout) :: cx(dim,2*dimx-4)
```

```
    double precision :: h,kc2ph
    integer :: i

!
!   Calculating the de Boor points of the first derivative
!
    do i=dimx-2,2,-1
        h=xx(i+1)-xx(i)
        kc2ph=h*KC2(px(i)*h/2)/2
        cx(1:dim,2*i)=(cx(1:dim,2*i)-cx(1:dim,2*i-1))/kc2ph
        cx(1:dim,2*i-1)=(cx(1:dim,2*i-1)-cx(1:dim,2*i-2))/kc2ph
    enddo


    end subroutine tension1der3C1coeff
```

To illustrate the use of the subroutines above, we suggest you the following example, which can be also found on the enclosed CD:

Example 8.1 *We solve the singularly perturbed differential two-point boundary value problem*

$$\varepsilon y'' + y' - (1+\varepsilon)y = 0, \tag{8.1}$$

$$y(0) = 1 + e^{-1}, \qquad y(1) = 1 + e^{-\frac{1+\varepsilon}{\varepsilon}}, \tag{8.2}$$

by the collocation method by C^1 tension spline at the generalized Gaussian points (for all details see [17] *and* [19]*), where the choice of tension parameters is done by (6.4) on the page 152 of* [19]*, and the generalized Gaussian points are calculated by equations (3.3) and (3.4) from the same source. Important remark is that the tension parameters p_i from* [17] *and* [19] *are equal to the $p_i h_i$ from this thesis.*

The main program `exampleC1` *calculates the approximation of the given problem, with $\varepsilon = 2^{-6}$, by C^1 tension spline where $[0,1]$ is divided on $n = 64$ subintervals, then plots the approximation together with the exact solution*

$$y(x) = e^{-x\frac{1+\varepsilon}{\varepsilon}} + e^{x-1}, \tag{8.3}$$

the absolute error, the first and the second derivative of the approximating C^1 tension spline, and derive maximal absolute error. `exampleC1` *calls the subroutine* `linsystemC1` *which forms the matrix and the vector of the system of linear equations, produced by the collocation method, and then solves the system. To do that we also need the function* `gausspoints`*, for calculating the generalized Gaussian points, and the subroutine* `diffopC1` *which calculates the value of the differential operator from (8.1) applied on C^1 tension spline. The module* `exampldiffeqC1` *contains all functions that define the boundary value problem (8.1)–(8.2).*

8.2 C^2 tension splines

The extended partition associated with C^2 tension splines has all interior knots of multiplicity one and the both boundary points of multiplicity four, and, this time, it is placed in the array `xx`. Because of that, the numeration of the elements of array `cx` with de Boor points corresponds to the array `xx`. The array of tension parameters `px` is again associated with the `xx`. Similarly to the previous case, the first and the last three elements of the array `px` are arbitrary.

The subroutine `tension4C2` calculates the value of a C^2 spline by the quasi–Oslo type algorithm described in Subsection 6.3.1:

```
    subroutine tension4C2 (x,xx,px,ileft,cx,dim,spl)

!
!   Quasi-Oslo type algorithm for tension spline curves
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               spline curve is evaluated
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               dimx, the array of de Boor points associated with
!               the extended partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
```

```
!      dimx   = size(xx)-4
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the tension spline curve at the
!               point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision :: t(-2:3),p(-2:2),c(dim,-3:0) ! t(0)<=x<=t(1)
    integer :: i,j
    double precision :: h(-2:2),cc(4,4),tspl(4),Bspl1,Bspl2,Bspl3, &
                        Bspl4,d(4)

!
!   Storing only relevant data
!
    t(-2:3)=xx(ileft-2:ileft+3)
    p(-2:2)=px(ileft-2:ileft+2)
    c(1:dim,-3:0)=cx(1:dim,ileft-3:ileft)
!
!   Calculating the length of the subintervals involved
!
    do i=-2,2
        h(i)=t(i+1)-t(i)
    enddo
!
!   Calculating nontrivial Chebyshev tension B-splines at the
!   point x
!
    cc=0.0d0
    do i=1,4
        cc(i,i)=1.0d0
    enddo

    call tension4C1 (x,t(-1:2),p(-1:1),2,cc,4,tspl)
!
```

```
!   Calculating tension B-splines as the linear combination of the
!   Chebyshev tension B-splines
!
    Bspl1=gamma4d4(h(-2:0),p(-2:0))*tspl(1)
    call gamma4(h(-2:1),p(-2:1),d)
    Bspl2=d(2)*tspl(1)+d(3)*tspl(2)+d(4)*tspl(3)
    call gamma4(h(-1:2),p(-1:2),d)
    Bspl3=d(1)*tspl(2)+d(2)*tspl(3)+d(3)*tspl(4)
    Bspl4=gamma4d1(h(0:2),p(0:2))*tspl(4)
!
!   The final value of the spline curve
!
    spl=c(1:dim,-3)*Bspl1+c(1:dim,-2)*Bspl2+c(1:dim,-1)*Bspl3+ &
        c(1:dim,0)*Bspl4


    end subroutine tension4C2
```

The equation (6.26) gives us the formula for calculating the third order B-spline, and subroutine `tension3C2` produces the value of such a spline:

```
    subroutine tension3C2 (x,xx,px,ileft,cx,dim,spl)

!
!   Quasi-Oslo type algorithm for the third order tension spline
!   curves
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               spline curve is evaluated
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       ileft = integer variable, the index of the array xx such
```

```
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               dimx, the array of de Boor points of the third order
!               tension spline, associated with the extended
!               partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+3)], so cx(1:dim,1) can be
!               arbitrary
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)-4
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the third order tension spline curve
!               at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision :: t(-1:2),p(-1:1),c(dim,-2:0) ! t(0)<=x<=t(1)
    double precision :: h(-1:1),cc(3,4),tspl(3),Bspl1,Bspl2,Bspl3
    integer :: i

!
!   Storing only relevant data
!
    t(-1:2)=xx(ileft-1:ileft+2)
    p(-1:1)=px(ileft-1:ileft+1)
    c(1:dim,-2:0)=cx(1:dim,ileft-2:ileft)
!
!   Calculating the length of the subintervals involved
!
    do i=-1,1
        h(i)=t(i+1)-t(i)
```

```
    enddo
!
!   Calculating the nontrivial third order Chebyshev tension
!   B-splines at the point x
!
    cc=0.0d0
    do i=1,3
        cc(i,i+1)=1.0d0
    enddo

    call tension3C1 (x,t(-1:1),p(-1:0),2,cc,3,tspl)
!
!   Calculating the third order tension B-splines as the linear
!   combination of the third order Chebyshev tension B-splines
!
    Bspl1=KC2div(h(0),h(-1),p(0),p(-1))*tspl(1)
    Bspl2=KC2div(h(-1),h(0),p(-1),p(0))*tspl(1)+tspl(2)+ &
          KC2div(h(1),h(0),p(1),p(0))*tspl(3)
    Bspl3=KC2div(h(0),h(1),p(0),p(1))*tspl(3)
!
!   The final value of the spline curve
!
    spl=c(1:dim,-2)*Bspl1+c(1:dim,-1)*Bspl2+c(1:dim,0)*Bspl3


    end subroutine tension3C2
```

Then, the first derivative of a C^2 tension spline is calculated as in Subsection 6.3.3:

```
    subroutine tension1der4C2 (x,xx,px,ileft,cx,dim,spl)

!
!   Calculating the first derivative of the tension spline curve
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               the first derivative of the spline curve is
!               evaluated
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
```

```
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               dimx, the array of de Boor points of the tension
!               spline, associated with the extended partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)-4
!
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the first derivative of the tension
!               spline curve at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision :: t(-2:3),p(-2:2),c(dim,-3:0) ! t(0)<=x<=t(1)
    double precision :: h(-2:2)
    integer :: i

!
!   Storing only relevant data
!
    t(-2:3)=xx(ileft-2:ileft+3)
```

```
    p(-2:2)=px(ileft-2:ileft+2)
    c(1:dim,-3:0)=cx(1:dim,ileft-3:ileft)
!
!   Calculating the length of the subintervals involved
!
    do i=-2,2
        h(i)=t(i+1)-t(i)
    enddo
!
!   Calculating the de Boor points of the first derivative
!
    do i=0,-2,-1
        c(1:dim,i)=(c(1:dim,i)-c(1:dim,i-1))/KC3(h(i:i+2),p(i:i+2))
    enddo
!
!   Calculating the value of the first derivative of the tension
!   spline curve as the linear combination of the third order
!   tension B-splines
!
    call tension3C2 (x,t,p,3,c(1:dim,-2:0),dim,spl)


    end subroutine tension1der4C2
```

As in previous section, to speed up the calculation of the first derivative of C^2 tension spline at many points, it is useful to calculate all de Boor points of the first derivative in advance:

```
    subroutine tension1der4C2coeff (xx,px,cx,dimx,dim)

!
!   Calculating all de Boor points of the first derivative of the
!   tension spline curve
!
!   INPUT:
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
```

```
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       dimx  = integer variable, dimx+4 is the size of the array xx
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   INPUT and OUTPUT:
!       cx    = rank-two double precision array of shape dim by
!               dimx;
!               ON INPUT: the array of de Boor points of the tension
!               spline, associated with the extended partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+4)]
!               ON OUTPUT: the array of the de Boor points of the
!               first derivative of the tension spline;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+3)] (cx(1:dim,1) will not be
!               used)
!

    use functions

    implicit none

    integer, intent(in) :: dimx,dim
    double precision, intent(in) :: xx(dimx+4),px(dimx+3)
    double precision, intent(inout) :: cx(dim,dimx)

    double precision :: h(3)
    integer :: i

!
!   Calculating the de Boor points of the first derivative
!
    h=0.0d0
    do i=dimx,2,-1
        h(1)=xx(i+1)-xx(i)
        cx(1:dim,i)=(cx(1:dim,i)-cx(1:dim,i-1))/KC3(h,px(i:i+2))
        h(2:3)=h(1:2)
    enddo
```

```
    end subroutine tension1der4C2coeff
```

We get the value of the second order B-spline by (6.28), (6.29) and (6.30), which leads us to the the algorithm for calculating the value of a spline from the space of the second derivatives of the C^2 tension splines, and this can be also achieved by `tension2`, listed in the previous section. So, the second derivative of the C^2 tension spline is calculated, again as in Subsection 6.3.3, by:

```
    subroutine tension2der4C2 (x,xx,px,ileft,cx,dim,spl)

!
!   Calculating the second derivative of the tension spline curve
!
!   INPUT:
!       x     = double precision variable, the point at which the
!               the second derivative of the spline curve is
!               evaluated
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       ileft = integer variable, the index of the array xx such
!               that x is contained in the subinterval
!               [xx(ileft),xx(ileft+1)]
!       cx    = rank-two double precision array of shape dim by
!               dimx, the array of de Boor points of the tension
!               spline, associated with the extended partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+4)]
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   AUXILIARY TERM:
!      dimx   = size(xx)-4
!
```

```
!   OUTPUT:
!       spl   = rank-one double precision array of size dim, the
!               output value of the second derivative of the tension
!               spline curve at the point x
!

    use functions

    implicit none

    integer, intent(in) :: ileft,dim
    double precision, intent(in) :: x,xx(*),px(*),cx(dim,*)
    double precision, intent(out) :: spl(dim)

    double precision :: t(-2:3),p(-2:2),c(dim,-3:0) ! t(0)<=x<=t(1)
    double precision :: h(-2:2),kc2ph(-1:1)
    integer :: i

!
!   Storing only relevant data
!
    t(-2:3)=xx(ileft-2:ileft+3)
    p(-2:2)=px(ileft-2:ileft+2)
    c(1:dim,-3:0)=cx(1:dim,ileft-3:ileft)
!
!   Calculating the length of the subintervals involved and relevant
!   integrals
!
    do i=-2,2
        h(i)=t(i+1)-t(i)
        if (i>-2 .and. i<2) kc2ph(i)=h(i)*KC2(p(i)*h(i)/2)/2
    enddo
!
!   Calculating the de Boor points of the first derivative
!
    do i=0,-2,-1
        c(1:dim,i)=(c(1:dim,i)-c(1:dim,i-1))/KC3(h(i:i+2),p(i:i+2))
    enddo
!
!   Calculating the de Boor points of the second derivative
!
    do i=0,-1,-1
        c(1:dim,i)=(c(1:dim,i)-c(1:dim,i-1))/(kc2ph(i)+kc2ph(i+1))
    enddo
```

```
!
!   Calculating the value of the second derivative of the tension
!   spline curve as the linear combination of the second order
!   tension B-splines
!
    call tension2 (x,t,p,3,c(1:dim,-1:0),2,dim,spl)


    end subroutine tension2der4C2
```

At the end, the subroutine `tension1der3C2coeff` calculates all de Boor points of the first derivative of the third order tension spline. Again, with `tension1der4C2coeff` and `tension1der3C2coeff` we can get all de Boor points of the second derivative of the given C^2 tension spline.

```
    subroutine tension1der3C2coeff (xx,px,cx,dimx,dim)

!
!   Calculating de Boor points of the first derivative of the third
!   order tension spline curve
!
!   INPUT:
!       xx    = rank-one double precision array of size dimx+4,
!               the increasing array of knots, assumed to fulfill
!               the conditions:
!               xx(1)=xx(2)=xx(3)=xx(4)<xx(5)<...<xx(i)<xx(i+1)<...
!               <xx(dimx)<xx(dimx+1)=xx(dimx+2)=xx(dimx+3)=
!               xx(dimx+4); xx(1),...,xx(dimx+4) is the extended
!               partition with all interior knots of multiplicity 1,
!               and the boundary points of multiplicity 4
!       px    = rank-one double precision array of size dimx+3,
!               the array of tension parameters, assumed to be
!               nonnegative: px(i)>=0 is the tension parameter on
!               the subinterval [xx(i),xx(i+1));
!               px(1),px(2),px(3),px(dimx+1),px(dimx+2) and
!               px(dimx+3) can be arbitrary
!       dimx  = integer variable, dimx+4 is the size of the array xx
!       dim   = integer variable, the extent of the first dimension
!               in the array cx, assumed to be positive
!
!   INPUT and OUTPUT:
!       cx    = rank-two double precision array of shape dim by
!               dim;
!               ON INPUT: the array of de Boor points of the
!               third order tension spline, associated with the
```

```
!               extended partition xx;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+3)], so cx(1:dim,1) can be
!               arbitrary
!               ON OUTPUT: the array of the de Boor points of the
!               first derivative of the third order tension spline;
!               cx(1:dim,i) corresponds to the B-spline with the
!               support [xx(i),xx(i+2)] (cx(1:dim,1) and cx(1:dim,2)
!               will not be used)
!

    use functions

    implicit none

    integer, intent(in) :: dimx,dim
    double precision, intent(in) :: xx(dimx+4),px(dimx+3)
    double precision, intent(inout) :: cx(dim,dimx)

    double precision :: h(2),kc2ph(2)
    integer :: i

!
!   Calculating the de Boor points of the first derivative
!
    h=0.0d0
    kc2ph=0.0d0
    do i=dimx,3,-1
        h(1)=xx(i+1)-xx(i)
        kc2ph(1)=h(1)*KC2(px(i)*h(1)/2)/2
        cx(1:dim,i)=(cx(1:dim,i)-cx(1:dim,i-1))/(kc2ph(1)+kc2ph(2))
        h(2)=h(1)
        kc2ph(2)=kc2ph(1)
    enddo


    end subroutine tension1der3C2coeff
```

As in the case of the C^1 tension splines, we apply the C^2 tension spline subroutines for solving the same problem as in Example 8.1, which is also enclosed on the CD:

Example 8.2 *We solve the singularly perturbed differential two-point boundary value problem (8.1)–(8.2) by the collocation method by C^2 tension spline (for all details see* [16] *and* [20]*). The tension parameters are defined as in* [17] *and* [19]*, so*

the remark from Example 8.1 about their relationship with those defined here, holds still. The choice of the tension parameters are made by (18)-(20) from [20].

The main program `exampleC2` *solves the given problem, where it is again* $\varepsilon = 2^{-6}$*, by the collocation method by* C^2 *tension spline, and with* $[0,1]$ *being divided on* $n = 64$ *subintervals.* `exampleC2` *produces the same kind of results as* `exampleC1`*, and for forming and solving the collocation linear system uses the subroutine* `linsystemC2`*. For collocation points, in this case, we use the knots of the partition. Finally, the subroutine* `diffopC1` *calculates the value of the the differential operator from (8.1) applied on* C^2 *tension spline, and the module* `exampldiffeqC2` *has the same role as* `exampldiffeqC1` *from Example 8.1.*

Appendix A

Summary

In this thesis our point of interest are canonical complete Chebyshev (CCC)–systems and splines associated with them. We are interested in finding numerically stable algorithms for calculating with such splines, and we do that by generalizing the knot insertion based algorithms for polynomial splines to CCC–systems. To be able to construct these algorithms, we introduce knot insertion matrices, and then develop Oslo type algorithms and the generalized de Boor algorithm. To show the practical value of these algorithms, we apply them on four kinds of splines: weighted, q-splines, tension and cycloidal splines. Weighted and tension splines are particulary interesting, since weighted splines are the only splines of order higher than 4 which can be stably evaluated, and tension splines because of their wide application. For each of these splines, algorithms are developed with all the details specific for the spline in question. Finally to illustrate the practical computer use of given algorithms, we list program codes involved in calculating with C^1 and C^2 tension splines.

Appendix B

Sažetak

Središte zanimanja ove disertacije su kanonski kompletni Čebiševljevi (CCC)–sistemi i splajnovi izvedeni iz njih, a također i numerički stabilni algoritmi za računanje takvih splajnova. Njih pronalazimo u generalizaciji algoritama baziranih na ubacivanju čvorova za polinomne splajnove na CCC-sisteme. Za konstrukciju takvih algoritama uvode se matrice za ubacivanje čvorova, a iz njih algoritmi tipa Oslo i generalizirani de Boor-ov algoritam. U svrhu pokazivanja praktičnosti ovih algoritama, oni se primjenjuju za računanje četiri vrste splajnova: za težinske, q-splajnove, napete i cikloidne splajnove. Težinski i napeti splajnovi su posebno zanimljivi, pošto su težinski, u ovoj disertaciji, jedini splajnovi reda višeg od 4 koji se računaju, a napeti zbog njihove široke primjene. Za svaki od tih splajnova razvijeni su algoritmi sa svim detaljima specifičnim za dotični splajn. Na kraju, kao ilustracija praktične upotrebe danih algoritama na računalu, popisani su programski kodovi za računanje s C^1 i C^2 napetim splajnovima.

Appendix C

Curriculum vitæ

I was born on July 8, 1973 in Zagreb, where I have finished elementary school. I continued my education at the Mathematics high school in Zagreb, and in 1992 I have entered upon University of Zagreb, Department of Mathematics. I graduated applied mathematics in 1997, and the same year started postgraduate studies in mathematics. Next year I took a job as an assistant on the same faculty. Until now, my work field is numerical mathematics, more precisely, I am involved in the development of Chebyshev spline theory, with an emphasis on numerically stable algorithms for calculating with spline curves. My motivation was the growing need for such algorithms, especially in CAGD, solving ordinary differential equations, and some minimization problems. In 2002 I got masters degree in mathematics by defending master's thesis on the subject "Polar Forms of Splines and Knot Insertion Algorithms". Most of my results are published in 4 reviewed articles, and presented in scientific meetings and seminars. I also participated in several conferences in Croatia and abroad. Next to my scientific work, I am also engaged in lectures for students, mostly in subjects connected to the numerical mathematics and computer science.

Bibliography

[1] T. Bosner, *Polarne forme i ubacivanje čvorova splajnova*, master's thesis, Dept. of Mathematics, University of Zagreb, 2002.

[2] ——, *Knot insertion algorithms for weighted splines*, in Proceedings of the Conference on Applied Mathematics and Scientific Computing, Z. Drmač, M. Marušić, and Z. Tutek, eds., Springer, 2005, pp. 151–160.

[3] T. Bosner and M. Rogina, *A stable algorithms for calculating with q-splines*, in Proceedings of Computational and Applied Mathematics, M. Rogina, V. Hari, Z. Tutek, and N. Limić, eds., 2001, pp. 99–104.

[4] C. W. Burrill, *Measure, Integration, and Probability*, McGraw–Hill Book Company, 1972.

[5] J. M. Carnicer, E. Mainar, and J. M. Peña, *A unified framework for cubics and cycloids*, in Curve and Surface Design, T. Lyche, M.-L. Mazure, and L. L. Schumaker, eds., Brentwood, 2003, Nashboro Press, pp. 31–40.

[6] E. Cohen, T. Lyche, and R. Riesenfeld, *Discrete B-splines and subdivision techniques in computer–aided geometric and computer graphic*, Computer Graphic and Image Processing, 14 (1980), pp. 87–111.

[7] T. A. Foley, *Interpolation with interval and point tension controls using cubic weighted ν-splines*, ACM Transactions on Mathematical Software, Vol. 13, No. 1. (1987), pp. 68–96.

[8] R. N. Goldman and T. Lyche, eds., *Knot Insertion and Deletion Algorithms for B-Spline Curves and Surfaces*, SIAM, 1993.

[9] V. Horvat and M. Rogina, *Tension spline collocation methods for singularly perturbed Volterra integro-differential and Volterra integral equations*, J. Comput. Appl. Math., 140 (2002), pp. 381–402.

[10] S. Karlin, *Total Positivity*, Stanford Univ. Press, California, 1968.

[11] P. E. Koch and T. Lyche, *Construction of exponential tension B-splines of arbitrary order*, in Curves and Surfaces, P. J. Laurent, A. Le Méhauté, and L. L. Schumaker, eds., Boston, 1991, Academic Press, pp. 255–258.

[12] R. Kulkarni and P.-J. Laurent, *Q-splines*, Numer. Algorithms, 1 (1991), pp. 45–73.

[13] B. I. Kvasov, *Shape–Preserving Spline Approximation*, World Scientific, Singapore, 2000.

[14] E. Mainar and J. M. Peña, *Quadratic–cycloidal curves*, Advances in Computational Mathematics, 20 (2004), pp. 161–175.

[15] E. Mainar, J. M. Peña, and J. Sánchez-Reyes, *Shape preserving alternatives to the rational Bézier model*, Computer Aided Geometric Design, 18 (2001), pp. 37–60.

[16] M. Marušić, *Kolokacija napetim splajnom*, master's thesis, Dept. of Mathematics, University of Zagreb, 1992.

[17] ——, *Napeti B-spline i kolokacija u generaliziranim Gaussovim točkama*, PhD thesis, Dept. of Mathematics, University of Zagreb, 1995.

[18] ——, *Stable calculation by splines in tension*, Grazer Mathematische Berichte, 328 (1996), pp. 65–76.

[19] ——, *A fourth/second order accurate collocation method for singularly perturbed two-point boundary value problems using tension splines*, Numer. Math., 88 (2001), pp. 135–158.

[20] M. Marušić and M. Rogina, *A collocation method for singularly perturbed two-point boundary value problems with splines in tension*, Adv. Comput. Math., Vol. 6, No. 1. (1996), pp. 65–76.

[21] M.-L. Mazure, *Chebyshev blossoming.* RR 953M IMAG, Université Joseph Fourier, Grenoble, January 1996.

[22] ——, *Blossoming: a geometrical approach*, Constr. Approx., 15 (1999), pp. 33–68.

[23] ——, *Blossoms and optimal bases*, Advances in Computational Mathematics, 20 (2004), pp. 177–203.

[24] M.-L. Mazure and H. Pottmann, *Tchebycheff curves*, in Total Positivity and its Applications, M. Gasca and C. A. Micchelli, eds., Kluwer Academic Pub., 1996, pp. 187–218.

[25] G. Mülbach, *A recurrence formula for generalized divided differences and some applications*, J. of Approx. Theory, 9 (1973), pp. 165–172.

[26] G. Nürnberger, *Approximation by Spline Functions*, Springer–Verlag, Berlin, 1989.

[27] H. Pottmann, *The geometry of Tchebycheffian splines*, Computer Aided Geometric Design, 10 (1993), pp. 181–210.

[28] H. Pottmann and M. G. Wagner, *Helix splines as an example of affine Tchebycheffian splines*, Advances on Computational Math., 2 (1994), pp. 123–142.

[29] P. M. Prenter, *Piecewise L-splines*, Numer. Math., 18 (1971), pp. 243–253.

[30] M. Rogina, *TB-spline funkcije i primjene*, master's thesis, Dept. of Mathematics, University of Zagreb, 1984.

[31] ——, *Basis of splines associated with some singular differential operators*, BIT, 32 (1992), pp. 496–505.

[32] ——, *Nove rekurentne relacije za Čebiševljeve spline funkcije i njihove primjene*, PhD thesis, Dept. of Mathematics, University of Zagreb, 1994.

[33] ——, *A knot insertion algorithm for weighted cubic splines*, in Curves and Surfaces with Applications in CAGD, A. Le Méhauté, C. Rabut, and L. L. Schumaker, eds., Nashville & London, 1997, Vanderbilt University Press, pp. 387–395.

[34] ——, *On construction of fourth order Chebyshev splines*, Math. Commun., 4 (1999), pp. 83–92.

[35] ——, *Weighted integrals of polynomial splines*, in Proceedings of A4A5 (to appear), Chester, 2006.

[36] M. Rogina and T. Bosner, *On calculating with lower order Chebyshev splines*, in Curves and Surfaces Design, P. J. Laurent, P. Sabloniere, and L. L. Schumaker, eds., Nashville, 2000, Vanderbilt Univ. Press, pp. 343–353.

[37] ——, *A de Boor type algorithm for tension splines*, in Curve and Surface Fitting, A. Cohen, J.-L. Merrien, and L. L. Schumaker, eds., Brentwood, 2003, Nashboro Press, pp. 343–352.

[38] L. L. Schumaker, *On Tchebycheffian spline functions*, J. of Approx. Theory, 18 (1976), pp. 278–303.

[39] ——, *Spline Functions: Basic Theory*, John Wiley & Sons, New York, 1981.

[40] ——, *On recursions for generalized splines*, J. of Approx. Theory, 36 (1982), pp. 16–31.

[41] A. H. Vermeulen, R. H. Bartels, and G. R. Heppler, *Integrating products of B-splines*, Siam J. Sci. Stat. Comput., Vol. 13, No. 4 (1992), pp. 1025–1038.