

Spline Methods

Tom Lyche and Knut Mørken

Department of Mathematics
University of Oslo

May 6, 2026

Contents

0	Preface	1
1	Why splines and B-splines?	3
1.1	Convex combinations and convex hulls	4
1.1.1	Stable computations	4
1.1.2	The convex hull of a set of points	4
1.2	Parametric curves	6
1.3	Interpolating polynomial curves	9
1.3.1	Quadratic interpolation of three points	9
1.3.2	Interpolation by convex combinations?	9
1.4	Bézier curves	11
1.4.1	Quadratic Bézier curves	11
1.4.2	Composite Bézier curves	13
1.5	A geometric construction of spline curves	14
1.5.1	Linear spline curves	14
1.5.2	Quadratic spline curves	16
1.5.3	Smoothness of spline curves	18
1.5.4	Representing spline curves in terms of basis functions	18
1.6	Conclusion	20
1.7	Exercises	20
2	Basic properties of splines and B-splines	27
2.1	The recurrence relation for B-splines	27
2.2	Some simple consequences of the recurrence relation	28
2.2.1	Basic properties	28
2.2.2	More examples of B-splines	30
2.3	Linear combinations of B-splines	33
2.3.1	Spline functions	33
2.3.2	Spline curves	35
2.4	A matrix representation of B-splines	36

2.5	Algorithms for evaluating a spline	39
2.5.1	High level description	39
2.6	Exercises	42
3	Further properties of splines	49
3.1	Linear independence of B-splines and representation of polynomials	49
3.1.1	Some properties of the B-spline matrices	49
3.1.2	Marsden's identity and representation of polynomials	51
3.1.3	Linear independence of B-splines	53
3.2	Differentiation and smoothness of B-splines	55
3.2.1	Piecewise smooth functions	55
3.2.2	Derivatives of B-splines	56
3.2.3	Computing derivatives of splines and B-splines	59
3.2.4	Smoothness of B-splines	61
3.2.5	The integral of a B-spline	63
3.3	B-splines as a basis for piecewise polynomials	64
3.4	Exercises	66
4	Knot insertion	69
4.1	The control polygon relative to different knot vectors	70
4.2	Knot insertion	72
4.2.1	Basic idea	72
4.2.2	Conversion between B-spline polynomials	74
4.2.3	Formulas and algorithms for knot insertion	76
4.3	B-spline coefficients as functions of the knots	82
4.3.1	The blossom	83
4.3.2	B-spline coefficients as blossoms	86
4.4	Inserting one knot at a time	88
4.5	Bounding the number of sign changes in a spline	90
4.6	Exercises	93
5	Spline Approximation	97
5.1	Local Approximation Methods	98
5.1.1	Piecewise linear interpolation	98
5.1.2	Cubic Hermite interpolation	100
5.1.3	Estimating the derivatives	103
5.2	Cubic Spline Interpolation	104
5.2.1	Interpretations of cubic spline interpolation	108
5.2.2	Numerical solution and examples	109
5.3	General Spline Approximation	110
5.3.1	Spline interpolation	111
5.3.2	Least squares approximation	112
5.4	The Variation Diminishing Spline Approximation	116
5.4.1	Preservation of bounds on a function	119

5.4.2	Preservation of monotonicity	120
5.4.3	Preservation of convexity	122
6	Parametric Spline Curves	127
6.1	Definition of Parametric Curves	127
6.1.1	Regular parametric representations	127
6.1.2	Changes of parameter and parametric curves	129
6.1.3	Arc length parametrisation	130
6.2	Approximation by Parametric Spline Curves	131
6.2.1	Definition of parametric spline curves	131
6.2.2	The parametric variation diminishing spline approximation	133
6.2.3	Parametric spline interpolation	134
6.2.4	Assigning parameter values to discrete data	136
6.2.5	General parametric spline approximation	137
7	Tensor Product Spline Surfaces	139
7.1	Explicit tensor product spline surfaces	139
7.1.1	Definition of the tensor product spline	139
7.1.2	Evaluation of tensor product spline surfaces	143
7.2	Approximation methods for tensor product splines	143
7.2.1	The variation diminishing spline approximation	143
7.2.2	Tensor Product Spline Interpolation	145
7.2.3	Least Squares for Gridded Data	148
7.3	General tensor product methods	152
7.4	Trivariate Tensor Product Methods	154
7.5	Parametric Surfaces	158
7.5.1	Parametric Tensor Product Spline Surfaces	159
8	Quasi-interpolation methods	161
8.1	A general recipe	161
8.1.1	The basic idea	162
8.1.2	A more detailed description	162
8.2	Some quasi-interpolants	164
8.2.1	Piecewise linear interpolation	164
8.2.2	A 3-point quadratic quasi-interpolant	165
8.2.3	A 5-point cubic quasi-interpolant	166
8.2.4	Some remarks on the constructions	167
8.3	Quasi-interpolants are linear operators	168
8.4	Different kinds of linear functionals and their uses	169
8.4.1	Point functionals	169
8.4.2	Derivative functionals	170
8.4.3	Integral functionals	170
8.4.4	Preservation of moments and interpolation of linear functionals . . .	171
8.4.5	Least squares approximation	172

8.4.6	Computation of integral functionals	173
8.5	Alternative ways to construct coefficient functionals	173
8.5.1	Computation via evaluation of linear functionals	173
8.5.2	Computation via explicit representation of the local approximation	175
8.6	Two quasi-interpolants based on point functionals	175
8.6.1	A quasi-interpolant based on the Taylor polynomial	176
8.6.2	Quasi-interpolants based on evaluation	177
8.7	Exercises	178
9	Approximation theory and stability	181
9.1	The distance to polynomials	181
9.2	The distance to splines	183
9.2.1	The constant and linear cases	184
9.2.2	The quadratic case	185
9.2.3	The general case	186
9.3	Stability of the B-spline basis	190
9.3.1	A general definition of stability	190
9.3.2	Stability of the B-spline basis, $p = \infty$	191
9.3.3	Stability of the B-spline basis, $p < \infty$	192
9.4	Convergence of the control polygon for spline functions	194
9.5	Exercises	196
10	Shape Preserving Properties of B-splines	201
10.1	Bounding the number of zeros of a spline	201
10.2	Uniqueness of spline interpolation	204
10.2.1	Lagrange Interpolation	205
10.2.2	Hermite Interpolation	206
10.3	Total positivity	208
11	Computing Zeros of Splines	215
11.1	Counting zeros of the control polygon	215
11.2	Root finding algorithm	217
11.3	Convergence	220
11.4	Rate of convergence	222
11.5	Stability	222
11.6	Implementation and numerical examples	223
A	Some Linear Algebra	227
A.1	Matrices	227
A.1.1	Nonsingular matrices, and inverses.	227
A.1.2	Determinants.	228
A.1.3	Criteria for nonsingularity and singularity.	228
A.2	Vector Norms	229
A.3	Vector spaces of functions	231

A.3.1 Linear independence and bases	232
A.4 Normed Vector Spaces	234

Preface

This manuscript has been developed from a course on splines given at the University of Oslo starting in the 1970's.

Splines are piecewise polynomials, glued together by a certain number of continuous derivatives. B-splines are "basic splines", a very appealing basis for spaces of splines, with a host of both interesting and elegant properties. The "father of splines" is undoubtedly I. J. Schoenberg (1903 -1990). Schoenberg was the first to use splines systematically for the approximation of data, see¹. The use of polynomial splines for the approximation of functions and data is now well established. Splines are also important for representing curves and surfaces in Computer Aided Geometric Design ² and more recently in isogeometric analysis³. We will consider curves as well as tensor product surfaces.

The content is as follows. In Chapter 1 we give a motivation for the study of splines. We ask the question "What is a good mathematical framework for constructing and manipulating smooth curves?

In Chapter 2 we study the most basic properties of B-splines in detail. The starting point is the definition of B-splines via a recurrence relation. We give a matrix representation of splines and B-splines from which it is easy to deduce computational algorithms. The matrix representation is also the basis for our development of much of the theory in later chapters.

In Chapter 3 our focus is on the question "What kind of functions can be represented as linear combinations of B-splines?" The answer is that our spline space contains a large class of piecewise polynomials, and this ensures that splines are reasonably flexible, much more so than polynomials. This is proved by making use of some simple properties of the B-spline matrices.

In chapter 4 we develop algorithms for knot insertion, expressing the B-spline coefficients relative to a refined knot vector in terms of the B-spline coefficients relative to the original

¹Curry, H. B. and I.J. Schoenberg, On Pólya frequency functions IV: The fundamental spline functions and their limits, J. Analyse Math. 17 (1966), 71-107

²Barnhill, B. and Riesenfeld, R., Computer Aided Geometric Design, Academic Press, 1974

³Cottrell, J. A., Hughes, T. J. R. and Bazilevs, Y., Isogeometric Analysis: Toward Integration of CAD and FEA, Wiley Publishing, 2009

knot vector. Associated with a spline curve is a piecewise linear function interpolating the B-spline coefficients. This control polygon converges to the spline curve when more knots are inserted, and gives an approximate, very simple, and useful representation of the curve.

Chapter 5 introduces a number of methods for obtaining spline approximations to given functions, or more precisely, to data obtained by sampling a function.

Parametric spline curves is the topic of Chapter 6. Curves require a parametrization, and we consider various choices of parametrizations for spline curve approximations.

In Chapter 7 we study tensor product spline surfaces, both in the explicit and parametric case. Methods based on interpolation and least squares are considered.

A general recipe for constructing a class of spline approximations known as Quasi-interpolation methods is considered in chapter 8.

The quasi-interpolants also allow us, in Chapter 9, to establish two important properties of B-splines, stability and convergence of the control polygon.

In Chapter 10 we consider shape preserving properties of B-splines. We give bounds for the number of zeros of a spline and show that the B-spline collocation matrix is nonsingular and totally positive (This means that all minors are nonnegative).

In Chapter 11 we use properties of the control polygon to give algorithms for computing zeros of splines.

The presentation makes use of some basic linear algebra, and this is reviewed in an appendix.

Chapter 1

Why splines and B-splines?

Why should we study splines and B-splines? Perhaps the answer seems obvious to you, and then you may just jump to Chapter 2. If not, it is worthwhile to ask some basic questions. As we attempt to answer those questions we are going to touch upon topics like construction of smooth curves from finite point sets, polynomial interpolation, and Bézier curves. And we are also going to see that our study in this book is just one particular way to answer the questions.

Basic question 1.1. *What is a good mathematical framework for constructing and manipulating smooth curves?*

This is an open question that leaves us with many options. Do we want to represent the curve with a simple formula or a computational method? What tools may we use for calculations? What do we mean by smooth curves?

In order to address this question, we are going to gradually become more specific by introducing and exploring constraints that appear natural, or lead to simplifications, or in other ways make it possible to specify the mathematical framework.

The first two constraints are quite simple.

1. The mathematical framework should lead to methods that are suitable for computer implementation.
2. It should be easy to both approximate a given set of points and design new curves from scratch.

The first constraint can be made more precise by requiring that the methods should be efficient in their use of time and memory and they should not be overly sensitive to round-off errors in the computations. One way to control the sensitivity of the methods to round-off errors is by insisting that all the operations involved should amount to forming weighted averages, or convex combinations, of the given points. This has the added advantage that the constructions are geometrical in nature and easy to visualise.

In this chapter we explore some first consequences of the two constraints above. In Section 1.1, we discuss affine and convex combinations and the convex hull of a set of points, and relate these concepts to numerical stability (sensitivity to rounding errors), while in Section 1.2 we give a brief and very informal introduction to parametric curves. The first method for curve construction, namely polynomial interpolation, is introduced in Section 1.3. In Section 1.4 we show how to construct Bézier curves, and in Section 1.5 we generalise this construction to spline curves. At the outset, our construction of spline curves is geometrical in nature, but in Section 1.5.4 we show that spline curves can be written conveniently in terms of certain basis functions, namely B-splines. In the final section, we relate the material in this chapter to the rest of the book.

1.1 Convex combinations and convex hulls

An important constraint on our study is that it should result in numerical methods that will ultimately be implemented in floating point arithmetic on a computer. We should therefore make sure that these methods are reasonably insensitive to the primary source of problems, namely round-off errors and other numerical uncertainties that occur in numerical computations. This requirement is often referred to by saying that the methods should be *numerically stable*.

1.1.1 Stable computations

One characteristic of numerical instabilities is that a chain of computations contain numbers of large magnitude even though the numbers that form the input to the computations, and the final result, are not particularly large numbers. A simple way to avoid this is to base the computations on computing weighted averages as in

$$c = (1 - \lambda)c_1 + \lambda c_2. \quad (1.1)$$

Here c_1 and c_2 are two given numbers and λ a given weight, a real number in the range $[0, 1]$. The result of the computation is the number c which must lie between c_1 and c_2 as averages always do. A special example is of course computation of the mean between two numbers, $c = (c_1 + c_2)/2$. A computation on the form (1.1) is often referred to as a *convex combination*, and c is often said to be a convex combination of c_1 and c_2 . If all our computations are convex combinations, all intermediate results as well as the final result must be within the numerical range of the input data, thereby indicating that the computations are reasonably stable. It is overly optimistic to hope that we can do all our computations by forming convex combinations, but convex combinations will certainly be a guiding principle.

1.1.2 The convex hull of a set of points

Convex combinations make sense for vectors as well as for real numbers. If $\mathbf{c}_1 = (x_1, y_1)$ and $\mathbf{c}_2 = (x_2, y_2)$ are two vectors, or equivalently points (we always denote points and vectors by bold type), then a convex combination of $\mathbf{c}_1$ and $\mathbf{c}_2$ is an expression on the form

$$\mathbf{c} = (1 - \lambda)\mathbf{c}_1 + \lambda\mathbf{c}_2, \quad (1.2)$$

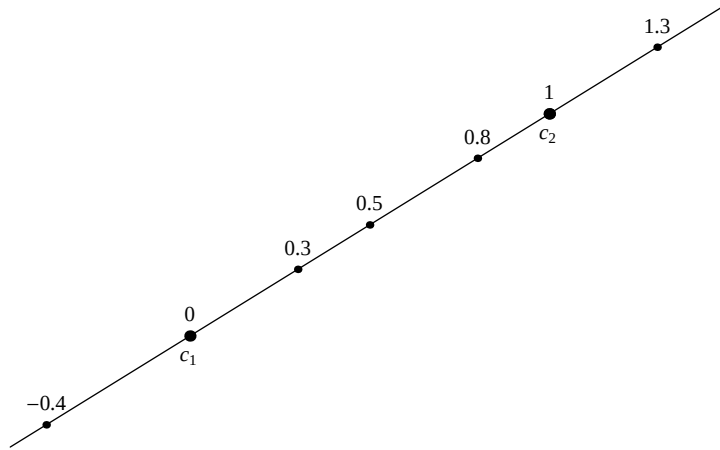


Figure 1.1. Some points on the line $(1 - \lambda)c_1 + \lambda c_2$ and the corresponding values of λ .

where the weight λ is some number in the range $0 \leq \lambda \leq 1$. This expression is usually implemented on a computer by expressing it in terms of convex combinations of real numbers,

$$(x, y) = ((1 - \lambda)x_1 + \lambda x_2, (1 - \lambda)y_1 + \lambda y_2),$$

where $(x, y) = c$.

A combination as in (1.2) with no restriction on λ other than $\lambda \in \mathbb{R}$ is called an *affine combination* of c_1 and c_2 . As λ takes on all real numbers, the point c in (1.2) will trace out the whole straight line that passes through c_1 and c_2 . If we restrict λ to lie in the interval $[0, 1]$, we only get the part of the line that lies between c_1 and c_2 . This is the *convex hull*, or the set of all weighted averages, of the two points. Figure 1.1 shows two points c_1 and c_2 and the line they define, together with some points on the line and their corresponding values of λ .

Just as we can take the average of more than two numbers, it is possible to form convex combinations of more than two points. A convex combination of n points $(c_i)_{i=1}^n$ is an expression on the form

$$c = \lambda_1 c_1 + \lambda_2 c_2 + \cdots + \lambda_n c_n$$

where the n numbers λ_i sum to one and also satisfy $0 \leq \lambda_i \leq 1$ for $i = 1, 2, \dots, n$. As for two points, the convex hull of the points $(c_i)_{i=1}^n$ is the set of all possible convex combinations of the points.

It can be shown that the convex hull of a set of points is the smallest convex set that contains all the points (recall that a set is convex if the straight line connecting any two points in the set is always completely contained in the set). This provides a simple geometric interpretation of the convex hull. As we have already seen, the convex hull of two points can be identified with the straight line segment that connects the points, whereas the convex hull of three points coincides with the triangle spanned by the points, see Figure 1.2. In general, the convex hull of n points is the n -sided polygon with the points as corners. However, if some of

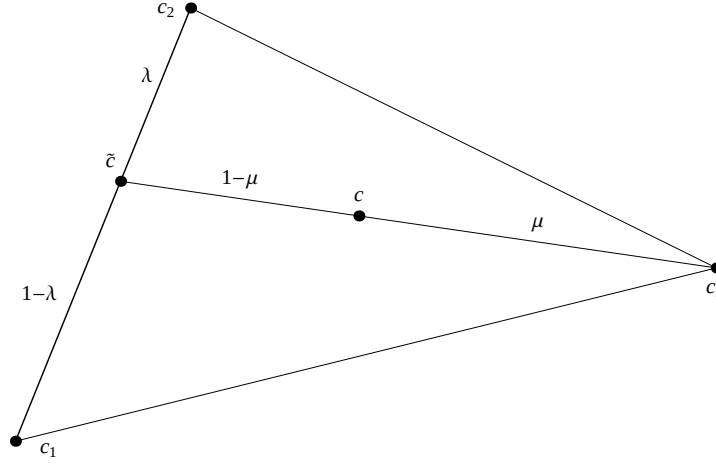


Figure 1.2. Determining the convex hull of three points.

the points are contained in the convex hull of the others, then the number of edges is reduced correspondingly, see the examples in Figure 1.3.

We can form convex and affine combinations in any space dimension, we just let $\mathbf{c}_1$ and $\mathbf{c}_2$ be points in the appropriate space. If we are working in $\mathbb{R}^n$ for instance, then $\mathbf{c}_1$ and $\mathbf{c}_2$ have n components. In our examples we will mostly use $n = 2$, as this makes the visualisation simpler.

1.2 Parametric curves

Before we consider some concrete methods for constructing smooth curves, we need to discuss a few basic concepts. For this we revisit the case of the straight line segment between two points $\mathbf{c}_0 = (x_0, y_0)$ and $\mathbf{c}_1 = (x_1, y_1)$. We have already seen that the line segment coincides with the convex hull of the two points, see (1.2). More generally, we can express the line segment as

$$\mathbf{q}(t | \mathbf{c}_0, \mathbf{c}_1; t_0, t_1) = \frac{t_1 - t}{t_1 - t_0} \mathbf{c}_0 + \frac{t - t_0}{t_1 - t_0} \mathbf{c}_1 \quad \text{for } t \in [t_0, t_1], \quad (1.3)$$

where t_0 and t_1 are two arbitrary real numbers with $t_0 < t_1$. Note that this is a convex combination since the two coefficients add to one,

$$\frac{t_1 - t}{t_1 - t_0} + \frac{t - t_0}{t_1 - t_0} = 1,$$

and each of them is nonnegative as long as t is in the interval $[t_0, t_1]$. In fact, if we set $\lambda = (t - t_0)/(t_1 - t_0)$, then (1.3) becomes (1.2).

A representation as in (1.3) that maps each real number to a point in $\mathbb{R}^2$, is an example of a *parametric representation*. The line can also be expressed as the linear function

$$y = f(x) = \frac{x_1 - x}{x_1 - x_0} y_0 + \frac{x - x_0}{x_1 - x_0} y_1,$$

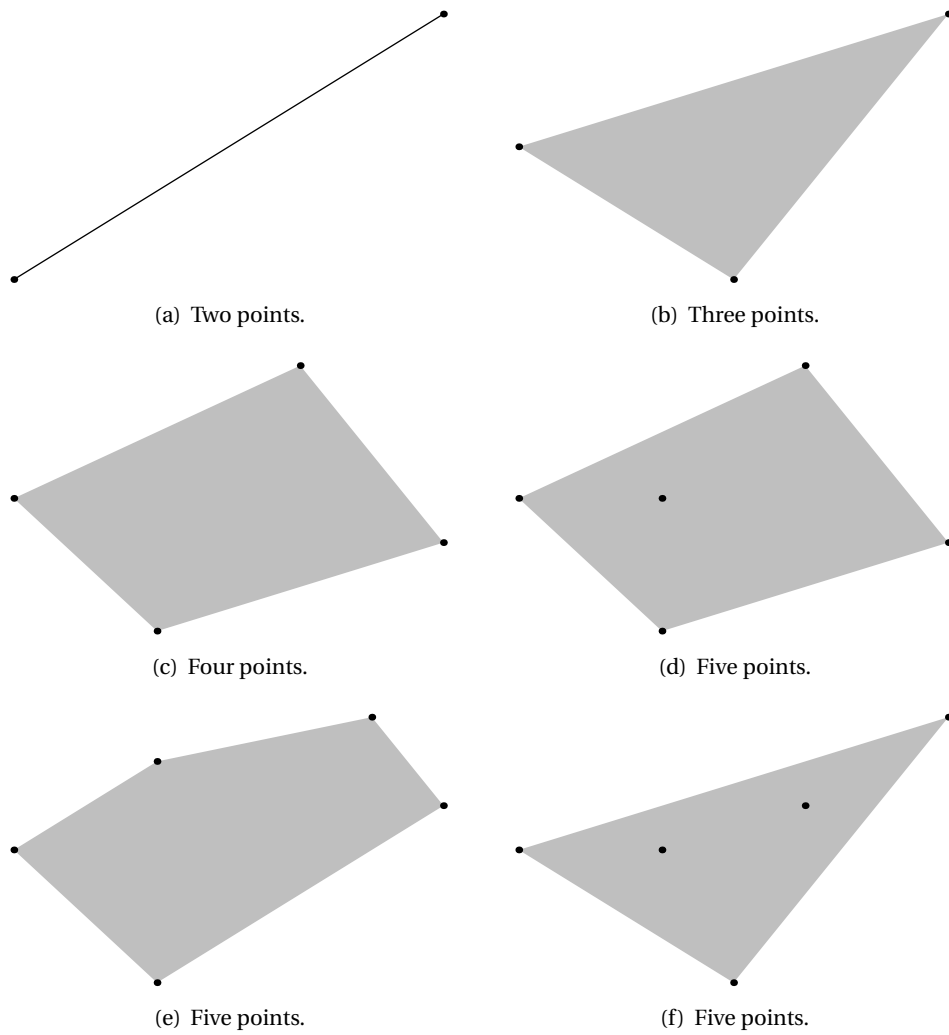


Figure 1.3. Examples of convex hulls (shaded area) of points (black dots).

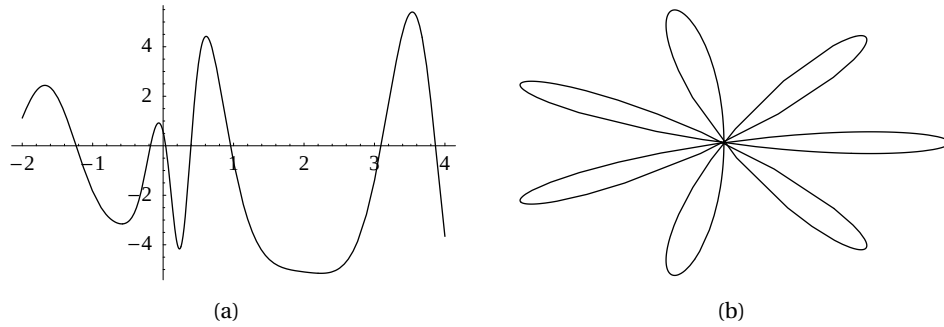


Figure 1.4. A function (a) and a parametric curve (b).

but then we run into problems if $x_0 = x_1$, i.e., if the line is vertical. In general, a parametric representation has the flexibility that it can cross itself, be vertical, or return to its starting point. Functions lack this flexibility; they always map a real number to a real number, see the two examples in Figure 1.4.

In this chapter we only work with parametric representations in the plane, and we will refer to these simply as (parametric) curves. All our constructions start with a set of points, from which we generate new points, preferably by forming convex combinations as in (1.2). In later chapters we will work mainly with functions (the points are simply real numbers) since the core of spline theory is independent of the space dimension. The reason for working with planar curves in this chapter is that the constructions are geometric in nature and particularly easy to visualise in the plane.

In (1.3) the two parameters t_0 and t_1 were arbitrary except that we assumed $t_0 < t_1$. If we consider the variable t to denote time, the parametric representation $\mathbf{q}(t \mid \mathbf{c}_0, \mathbf{c}_1; t_0, t_1)$ gives a way to travel from $\mathbf{c}_0$ to $\mathbf{c}_1$: The parameter t_0 gives the time at which we start at $\mathbf{c}_0$, and t_1 the time at which we arrive at $\mathbf{c}_1$. With this interpretation, different choices of t_0 and t_1 correspond to different ways of travelling along the line.

In the case of (1.3), the geometric result is always the same line segment. However, when we introduce more points, the effect of changing the time parameters can dramatically change the geometry of the curve. In practise, a curve is usually given by a particular parametric representation, but the same curve can be represented by many different parametric representations.

The distinction between a curve and a particular parametric representation is not just of theoretical significance. When only the geometric shape is significant we are discussing curves and their properties. Some examples are the outlines of the characters in a font and the level curves on a map. When it is also significant how we travel along the curve (how it is represented) then we are talking about a particular parametric representation of the underlying geometric curve, which in mathematical terms is simply a vector valued function. An example is the path of a camera in a computer based system for animation.

1.3 Interpolating polynomial curves

A natural way to construct a curve from a set of given points is to force the curve to pass through the points, or *interpolate* the points, a generalisation of the line segment defined by two points. Interpolating, polynomial curves can be defined for any finite number of points, but we only consider quadratic interpolation of three points here.

1.3.1 Quadratic interpolation of three points

How can we construct a curve that interpolates three points? In addition to the three given interpolation points $\mathbf{c}_0$, $\mathbf{c}_1$ and $\mathbf{c}_2$ we also need three parameters $(t_i)_{i=0}^2$. We first construct the two straight lines $\mathbf{q}_{0,1}(t) = \mathbf{q}(t \mid \mathbf{c}_0, \mathbf{c}_1; t_0, t_1)$ and $\mathbf{q}_{1,1}(t) = \mathbf{q}(t \mid \mathbf{c}_1, \mathbf{c}_2; t_1, t_2)$. If we now form the weighted average

$$\mathbf{q}_{0,2}(t) = \mathbf{q}(t \mid \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2; t_0, t_1, t_2) = \frac{t_2 - t}{t_2 - t_0} \mathbf{q}_{0,1}(t) + \frac{t - t_0}{t_2 - t_0} \mathbf{q}_{1,1}(t),$$

we obtain a curve that is quadratic in t , and it is easy to check that it passes through the given points as required,

$$\begin{aligned} \mathbf{q}_{0,2}(t_0) &= \mathbf{q}_{0,1}(t_0) = \mathbf{c}_0, \\ \mathbf{q}_{0,2}(t_1) &= \frac{t_2 - t_1}{t_2 - t_0} \mathbf{q}_{0,1}(t_1) + \frac{t_1 - t_0}{t_2 - t_0} \mathbf{q}_{1,1}(t_1) = \frac{t_2 - t_1}{t_2 - t_0} \mathbf{c}_1 + \frac{t_1 - t_0}{t_2 - t_0} \mathbf{c}_1 = \mathbf{c}_1, \\ \mathbf{q}_{0,2}(t_2) &= \mathbf{q}_{1,1}(t_2) = \mathbf{c}_2. \end{aligned}$$

Four examples are shown in Figure 1.5, with the interpolation points $(\mathbf{c}_i)_{i=0}^2$ given as black dots and the values of the three parameters $\mathbf{t} = (t_i)_{i=0}^2$ shown below each plot. The tangent vector at the end of the curve (at $t = t_2$) is also displayed in each case. Note that the interpolation points are the same in plots (a) and (b), and also in plots (c) and (d). When we only had two points, the linear interpolant between the points was independent of the values of the parameters t_0 and t_1 ; in the case of three points and quadratic interpolation the result is clearly highly dependent on the choice of parameters.

It is possible to give qualitative explanations of the results if we view $\mathbf{q}_{0,2}(t)$ as the position at time t of someone travelling along the curve. For example, in the first two plots the given points are quite uniformly spaced and the uniform distribution of parameters in plot (a) seems to connect the points with a 'nice' curve. In plot (b) the value of t_1 has been lowered, leaving more 'time' for travelling from $\mathbf{c}_1$ to $\mathbf{c}_2$ than from $\mathbf{c}_0$ to $\mathbf{c}_1$ with the effect that the curve bulges out between $\mathbf{c}_1$ and $\mathbf{c}_2$. This makes the journey between these points longer, and someone travelling along the curve can therefore spend the extra time allocated to this part of the 'journey'.

1.3.2 Interpolation by convex combinations?

A basic premise for our curve construction techniques is that we want to make use of convex combinations. It is clear that this is true in the linear case where we only have two points and the interpolant is the straight line that connects the points. By merely looking at the curves in Figure 1.5, it is clear that this is not the case in the quadratic case. Let us consider this in

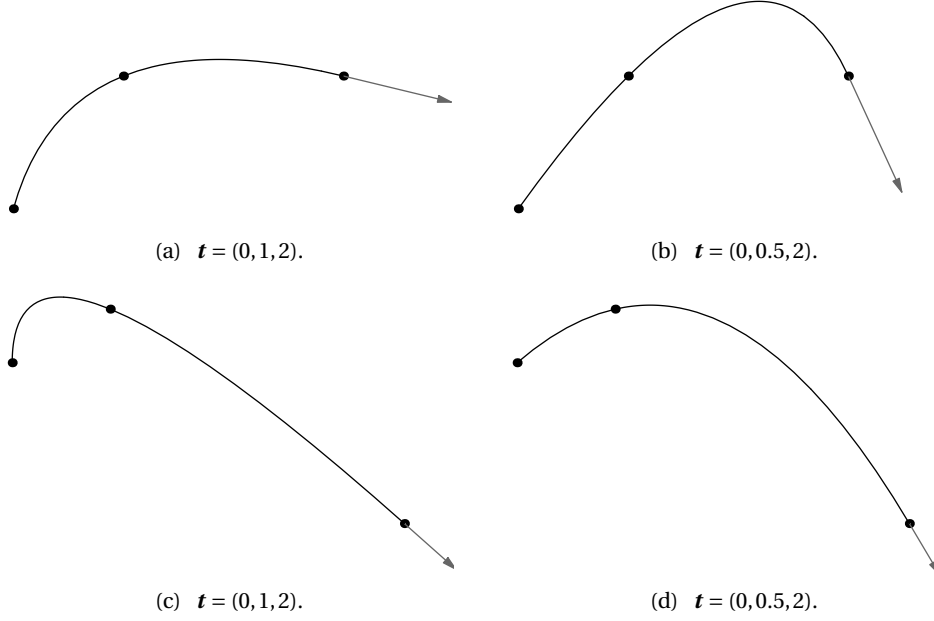


Figure 1.5. Some examples of quadratic interpolation.

more detail. Given the points $(\mathbf{c}_i)_{i=0}^2$ and the parameters $(t_i)_{i=0}^2$, we first form the two straight lines

$$\mathbf{q}_{0,1}(t) = \frac{t_1 - t}{t_1 - t_0} \mathbf{c}_0 + \frac{t - t_0}{t_1 - t_0} \mathbf{c}_1, \quad (1.4)$$

$$\mathbf{q}_{1,1}(t) = \frac{t_2 - t}{t_2 - t_1} \mathbf{c}_1 + \frac{t - t_1}{t_2 - t_1} \mathbf{c}_2, \quad (1.5)$$

and from these the quadratic segment

$$\mathbf{q}_{0,2}(t) = \frac{t_2 - t}{t_2 - t_0} \mathbf{q}_{0,1}(t) + \frac{t - t_0}{t_2 - t_0} \mathbf{q}_{1,1}(t). \quad (1.6)$$

The combination in (1.4) is convex as long as t is in $[t_0, t_1]$, the combination in (1.5) is convex when t lies within $[t_1, t_2]$, and the combination in (1.6) is convex when t is restricted to $[t_0, t_2]$. But in computing $\mathbf{q}_{0,2}(t)$ we also have to compute $\mathbf{q}_{0,1}(t)$ and $\mathbf{q}_{1,1}(t)$, and one of these latter combinations will not be convex when t is in $[t_0, t_2]$ (except when $t = t_1$). The problem lies in the fact that the two line segments are defined over different intervals, namely $[t_0, t_1]$ and $[t_1, t_2]$, that only has t_1 in common, so t cannot be in both intervals simultaneously. The situation is illustrated in Figure 1.6.

In the next section we shall see how we can construct polynomial curves from points in the plane by only forming convex combinations. The resulting curve will then lie within the convex hull of the given points, but will not interpolate the points.

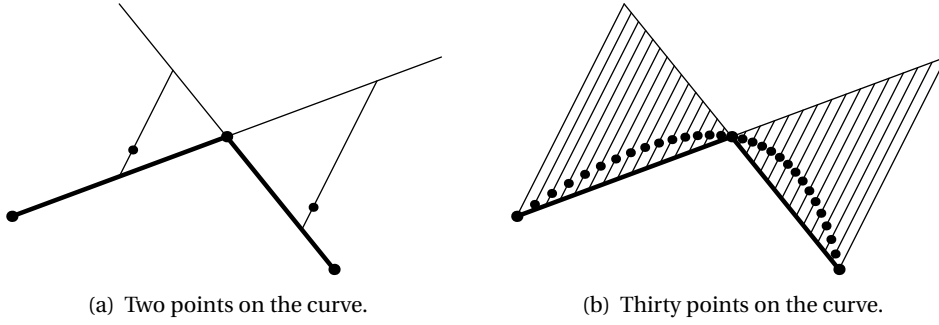


Figure 1.6. The geometry of quadratic interpolation.

1.4 Bézier curves

The curve construction method that we consider in this section is an alternative to polynomial interpolation and produces what we call *Bézier curves*, named after the French engineer Pierre Bézier (1910–1999) who worked for the car manufacturer Renault. Bézier curves are also polynomial curves, but they avoid the problem of wiggles and bulges because all computations are true convex combinations. It also turns out that segments of Bézier curves can easily be joined smoothly together to form more complex shapes. This avoids the problem of using curves of high polynomial degree when many points are approximated. Bézier curves are a special case of the spline curves that we will construct in Section 1.5.

1.4.1 Quadratic Bézier curves

We have three points in the plane $\mathbf{c}_0$, $\mathbf{c}_1$ and $\mathbf{c}_2$, and based on these points we want to construct a smooth curve, by forming convex combinations of the given points. With polynomial interpolation this did not work because the two line segments (1.4) and (1.5) are defined over different intervals. The natural solution is to start by defining the two line segments over the same interval, say $[0, 1]$ for simplicity,

$$\mathbf{p}_{1,1}(t) = \mathbf{p}(t | \mathbf{c}_0, \mathbf{c}_1) = (1-t)\mathbf{c}_0 + t\mathbf{c}_1, \quad (1.7)$$

$$\mathbf{p}_{2,1}(t) = \mathbf{p}(t | \mathbf{c}_1, \mathbf{c}_2) = (1-t)\mathbf{c}_1 + t\mathbf{c}_2. \quad (1.8)$$

(The curves we construct in this section and the next are related and will be denoted by $\mathbf{p}$ to distinguish them from the interpolating curves of Section 1.3.) Now we have no problem forming a true convex combination,

$$\mathbf{p}_{2,2}(t) = \mathbf{p}(t | \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2) = (1-t)\mathbf{p}_{1,1}(t) + t\mathbf{p}_{2,1}(t). \quad (1.9)$$

The construction is illustrated in Figure 1.7 (a). In Figure 1.7 (b) we have repeated the construction for 15 uniformly spaced values of t .

If we insert the explicit expressions for the two lines in (1.7) and (1.8) in (1.9) we obtain

$$\mathbf{p}_{2,2}(t) = (1-t)^2\mathbf{c}_0 + 2t(1-t)\mathbf{c}_1 + t^2\mathbf{c}_2 = b_{0,2}(t)\mathbf{c}_0 + b_{1,2}(t)\mathbf{c}_1 + b_{2,2}(t)\mathbf{c}_2. \quad (1.10)$$

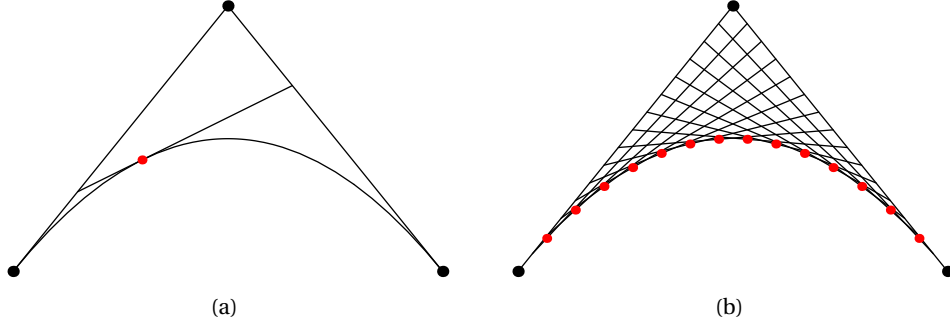


Figure 1.7. A Bézier curve based on three points.

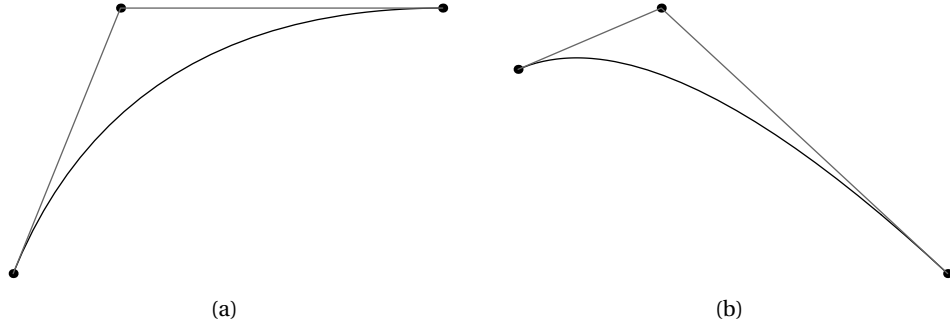


Figure 1.8. Two examples of quadratic Bézier curves.

This is called a quadratic *Bézier curve*; the points $(\mathbf{c}_i)_{i=0}^2$ are called the *control points* of the curve and the piecewise linear curve connecting the control points is called the *control polygon* of the curve. The polynomials multiplying the control points are the quadratic *Bernstein polynomials*. Two examples of quadratic Bézier curves with their control points and control polygons are shown in Figure 1.8 (the two sets of interpolation points in Figure 1.5 have been used as control points).

Some striking geometric features are clearly visible in Figures 1.7 and 1.8. We note that the curve interpolates $\mathbf{c}_0$ at $t = 0$ and $\mathbf{c}_2$ at $t = 1$. This can be verified algebraically by observing that $b_{0,2}(0) = 1$ and $b_{1,2}(0) = b_{2,2}(0) = 0$, and similarly $b_{2,2}(1) = 1$ while $b_{0,2}(1) = b_{1,2}(1) = 0$. The line from $\mathbf{c}_0$ to $\mathbf{c}_1$ coincides with the direction of the tangent to the curve at $t = 0$ while the line from $\mathbf{c}_1$ to $\mathbf{c}_2$ coincides with the direction of the tangent at $t = 1$. This observation can be confirmed by differentiating equation (1.10),

$$\mathbf{p}'_{2,2}(0) = 2(\mathbf{c}_1 - \mathbf{c}_0), \quad \mathbf{p}'_{2,2}(1) = 2(\mathbf{c}_2 - \mathbf{c}_1).$$

Note that the three polynomials in (1.10) add up to 1,

$$(1-t)^2 + 2t(1-t) + t^2 = (1-t+t)^2 = 1,$$

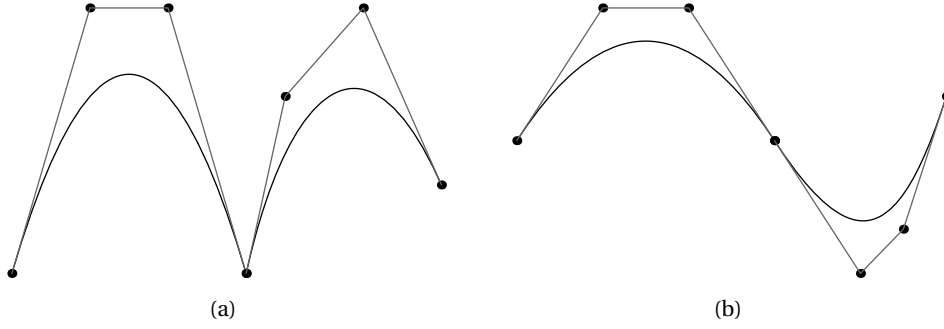


Figure 1.9. Different forms of continuity between two segments of a cubic Bézier curve.

and since t varies in the interval $[0, 1]$, we also have $0 \leq b_{i,2}(t) \leq 1$ for $i = 0, 1, 2$. This confirms that $\mathbf{p}_{2,2}(t)$ is a convex combination of the three points $(\mathbf{c}_i)_{i=0}^2$. As we saw in Section 1.1, the geometric interpretation of this is that the curve lies entirely within the triangle formed by the three given points.

The construction can easily be extended to an arbitrary number of points $d + 1$; then the curve will be of degree d , see exercise 3. The general construction inherits the interpolation and tangent properties at the beginning and end of the curve segment.

1.4.2 Composite Bézier curves

By using Bézier curves of sufficiently high degree we can represent a variety of shapes. However, Bézier curves of high degree suffer from the same shortcomings as interpolating polynomial curves:

1. As the degree increases, the complexity and therefore the processing time increases.
2. Because of the increased complexity, curves of high degree are more sensitive to round-off errors.
3. The relation between the given data points and the curve itself becomes less intuitive as the degree increases.

Because of the shortcomings of high degree polynomials it is advantageous to form complex shapes by joining together several Bézier curves, most commonly of degree two or three. Such composite Bézier curves are also referred to as Bézier curves.

A Bézier curve of degree d consisting of n segments is given by n sets of control points $(\mathbf{c}_0^i, \dots, \mathbf{c}_d^i)_{i=1}^n$. It is then convenient to let the complete curve be defined over the interval $[0, n]$, with segment i defined on the interval $[i - 1, i]$. By adjusting the control points appropriately it is possible to 'glue' together the segments with varying degrees of continuity. The minimal form of continuity is to let $\mathbf{c}_d^{i-1} = \mathbf{c}_0^i$ which ensures that segments $i - 1$ and i join together continuously as in Figure 1.9 (a). We obtain a smoother join by also letting the tangents be continuous at the join. The tangent property described above (in the quadratic case) means that the join between segments $i - 1$ and i will be continuous if

$$\mathbf{c}_d^{i-1} - \mathbf{c}_{d-1}^{i-1} = \mathbf{c}_1^i - \mathbf{c}_0^i.$$

An example for degree three is shown in Figure 1.9 (b).

Figure 1.10 shows one example of a complex Bézier curve. It is the letter S in the Postscript font Times Roman, shown with its control polygon and control points. This is essentially a cubic Bézier curve, interspersed with a few straight line segments. Each cubic curve segment can be identified by the two control points on the curve giving the ends of the segment and the two intermediate control points that lie off the curve.

Quadratic Bézier curves form the basis for the TrueType font technology, while cubic Bézier curves lie at the heart of PostScript and the Portable Document Format (PDF). Bézier curves also form a common primitive in many illustration programs.

1.5 A geometric construction of spline curves

The disadvantage of Bézier curves is that the smoothness between neighbouring polynomial pieces can only be controlled by choosing the control points appropriately. It turns out that by adjusting the construction of Bézier curves slightly, we can produce pieces of polynomial curves that automatically tie together smoothly. These piecewise polynomial curves are called *spline curves*.

1.5.1 Linear spline curves

The construction of spline curves is also based on repeated averaging, but we need a slight generalization of the Bézier curves, reminiscent of the construction of the interpolating polynomials in Section 1.3. In Section 1.3 we introduced the general representation (1.3) for a straight line connecting two points. In this section we use the same general representation, but with a different labelling of the points and parameters. If we have two points $\mathbf{c}_1$ and $\mathbf{c}_2$ we now represent the straight line between them by

$$\mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2; t_2, t_3) = \frac{t_3 - t}{t_3 - t_2} \mathbf{c}_1 + \frac{t - t_2}{t_3 - t_2} \mathbf{c}_2, \quad t \in [t_2, t_3], \quad (1.11)$$

provided $t_2 < t_3$. By setting $t_2 = 0$ and $t_3 = 1$ we get back to the linear Bézier curve.

The construction of a piecewise linear curve based on some given points $(\mathbf{c}_i)_{i=1}^n$ is quite obvious; we just connect each pair of neighbouring points by a straight line. More specifically, we choose n numbers $(t_i)_{i=2}^{n+1}$ with $t_i < t_{i+1}$ for $i = 2, 3, \dots, n$, and define the curve $\mathbf{f}$ by

$$\mathbf{f}(t) = \begin{cases} \mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2; t_2, t_3), & t \in [t_2, t_3), \\ \mathbf{p}(t \mid \mathbf{c}_2, \mathbf{c}_3; t_3, t_4), & t \in [t_3, t_4), \\ \vdots & \vdots \\ \mathbf{p}(t \mid \mathbf{c}_{n-1}, \mathbf{c}_n; t_n, t_{n+1}), & t \in [t_n, t_{n+1}]. \end{cases} \quad (1.12)$$

The points $(\mathbf{c}_i)_{i=1}^n$ are called the *control points* of the curve, while the parameters $\mathbf{t} = (t_i)_{i=2}^{n+1}$, which give the value of t at the control points, are referred to as the *knots*, or the *knot vector*, of the curve. If we introduce the piecewise constant functions $B_{i,0}(t)$ defined by

$$B_{i,0}(t) = \begin{cases} 1, & t_i \leq t < t_{i+1}, \\ 0, & \text{otherwise,} \end{cases} \quad (1.13)$$

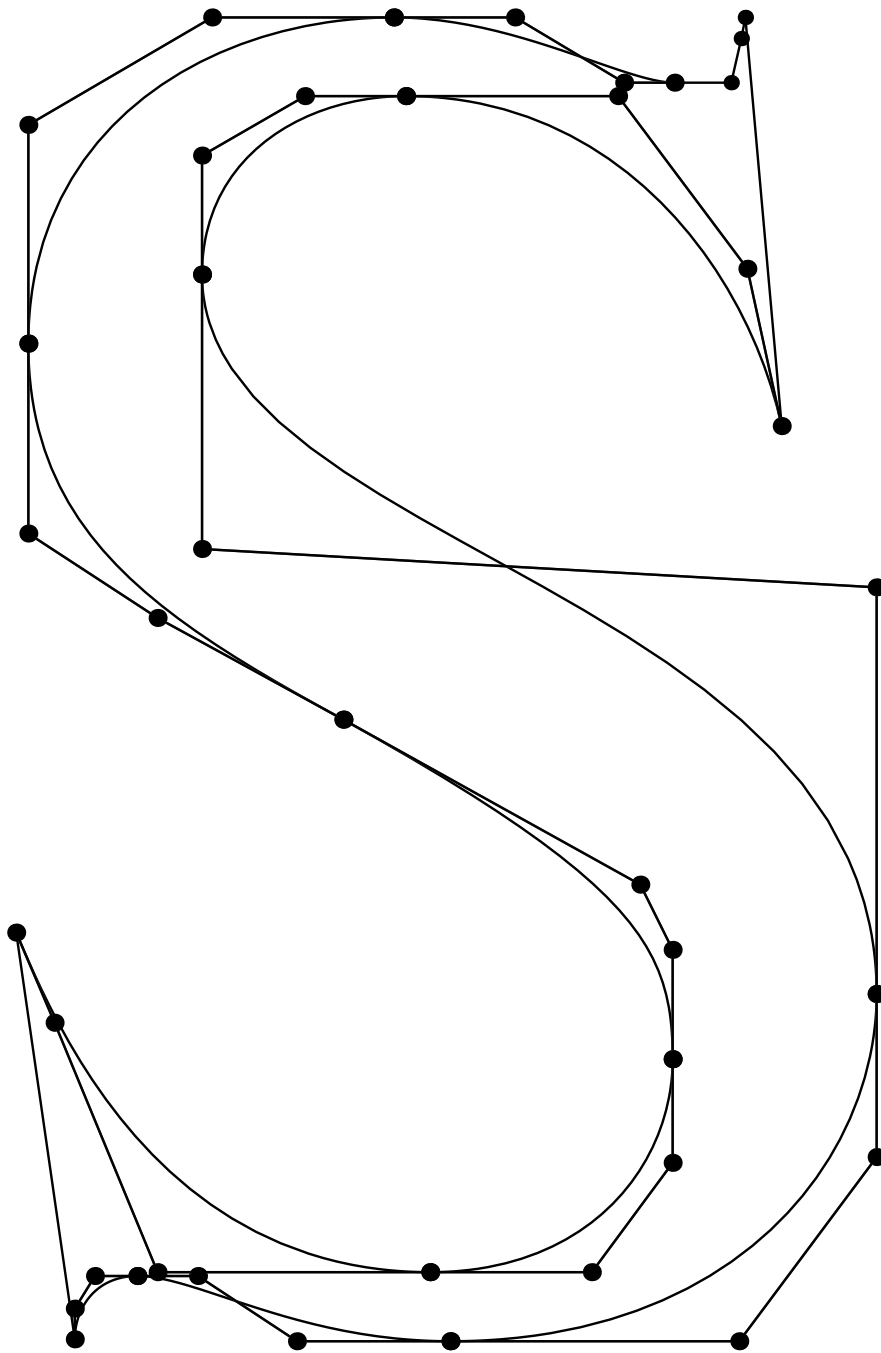


Figure 1.10. The letter S in the Postscript font Times Roman.

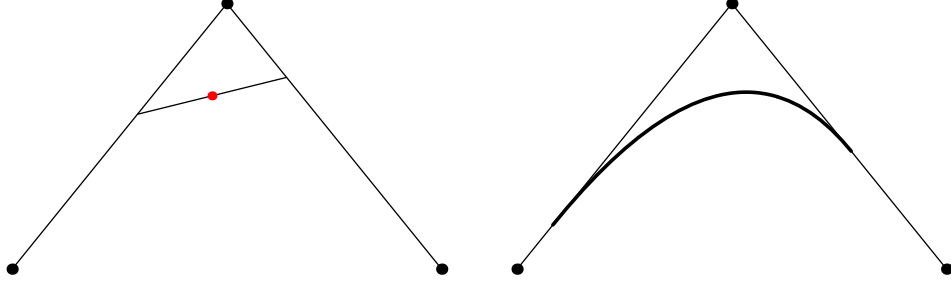


Figure 1.11. Construction of a segment of a quadratic spline curve.

and set $\mathbf{p}_{i,1}(t) = \mathbf{p}(t \mid \mathbf{c}_{i-1}, \mathbf{c}_i; t_i, t_{i+1})$, we can write $\mathbf{f}(t)$ more succinctly as

$$\mathbf{f}(t) = \sum_{i=2}^n \mathbf{p}_{i,1}(t) B_{i,0}(t). \quad (1.14)$$

This construction can be generalized to produce smooth, piecewise polynomial curves of higher degrees.

1.5.2 Quadratic spline curves

In the definition of the quadratic Bézier curve, a point on $\mathbf{p}_{2,2}(t)$ is determined by taking three averages, all with weights $1 - t$ and t since both the two line segments (1.7) and (1.8), and the quadratic curve itself (1.9), are defined with respect to the interval $[0, 1]$. The construction of spline functions is a hybrid between the interpolating polynomials of Section 1.3 and the Bézier curves of Section 1.4 in that we retain the convex combinations, but use more general weighted averages of the type in (1.11).

We construct a spline curve based on the three control points $\mathbf{c}_1$, $\mathbf{c}_2$, and $\mathbf{c}_3$ by introducing four knots $(t_i)_{i=2}^5$, with the assumption that $t_2 \leq t_3 < t_4 \leq t_5$. We represent the line connecting $\mathbf{c}_1$ and $\mathbf{c}_2$ by $\mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2; t_2, t_4)$ for $t \in [t_2, t_4]$, and the line connecting $\mathbf{c}_2$ and $\mathbf{c}_3$ by $\mathbf{p}(t \mid \mathbf{c}_2, \mathbf{c}_3; t_3, t_5)$ for $t \in [t_3, t_5]$. The reason for picking every other knot in the representation of the line segments is that then the interval $[t_3, t_4]$ is within the domain of both segments. This ensures that the two line segments can be combined in a convex combination to form a quadratic curve,

$$\mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3; t_2, t_3, t_4, t_5) = \frac{t_4 - t}{t_4 - t_3} \mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2; t_2, t_4) + \frac{t - t_3}{t_4 - t_3} \mathbf{p}(t \mid \mathbf{c}_2, \mathbf{c}_3; t_3, t_5) \quad (1.15)$$

with t varying in $[t_3, t_4]$. Of course we are free to vary t throughout the real line $\mathbb{R}$ since $\mathbf{p}$ is a polynomial in t , but then the three combinations involved are no longer all convex. The construction is illustrated in Figure 1.11. Note that if $t_2 = t_3 = 0$ and $t_4 = t_5 = 1$ we are back in the Bézier setting.

Just like for Bézier curves we refer to the given points as *control points* while the piecewise linear curve obtained by connecting neighbouring control points is the *control polygon*.

The added flexibility provided by the knots t_2 , t_3 , t_4 and t_5 turns out to be exactly what we need to produce smooth, piecewise quadratic curves, and by including sufficiently many control points and knots we can construct curves of almost any shape.

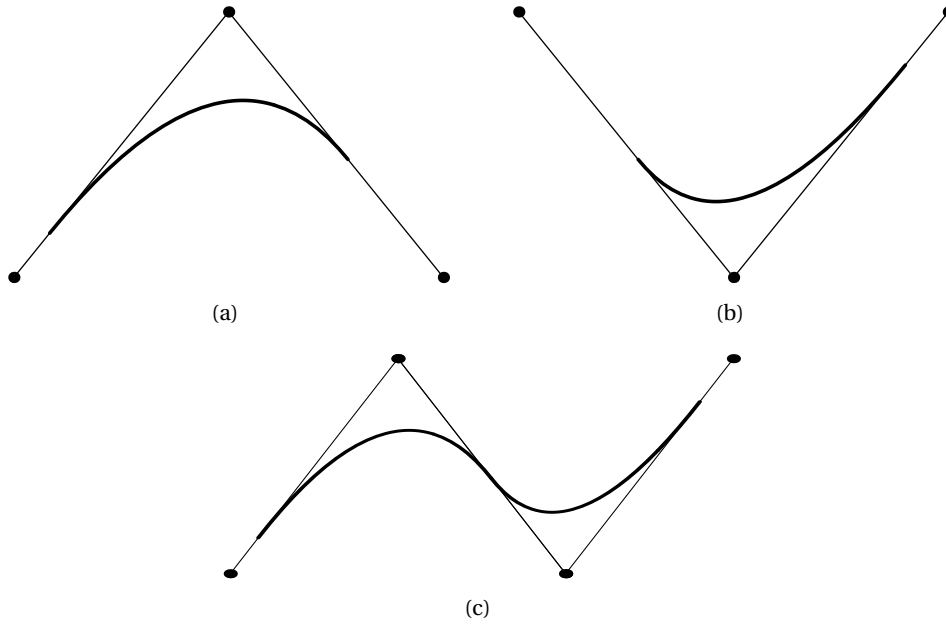


Figure 1.12. A quadratic spline curve (c) and its two polynomial segments (a) and (b).

Suppose we have n control points $(\mathbf{c}_i)_{i=1}^n$ and a sequence of knots $(t_i)_{i=2}^{n+2}$ that are assumed to be increasing except that we allow $t_2 = t_3$ and $t_{n+1} = t_{n+2}$. We define the quadratic spline curve $\mathbf{f}(t)$ by

$$\mathbf{f}(t) = \begin{cases} \mathbf{p}(t \mid \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3; t_2, t_3, t_4, t_5), & t_3 \leq t \leq t_4, \\ \mathbf{p}(t \mid \mathbf{c}_2, \mathbf{c}_3, \mathbf{c}_4; t_3, t_4, t_5, t_6), & t_4 \leq t \leq t_5, \\ \vdots & \vdots \\ \mathbf{p}(t \mid \mathbf{c}_{n-2}, \mathbf{c}_{n-1}, \mathbf{c}_n; t_{n-1}, t_n, t_{n+1}, t_{n+2}), & t_n \leq t \leq t_{n+1}. \end{cases} \quad (1.16)$$

An example with $n = 4$ is shown in Figure 1.12. Part (a) of the figure shows a quadratic curve defined on $[t_3, t_4]$ and part (b) a curve defined on the adjacent interval $[t_4, t_5]$. In part (c) the two curves in (a) and (b) have been superimposed in the same plot, and, quite strikingly, it appears that the curves meet smoothly at t_4 . The precise smoothness properties of splines will be proved in Section 3.2.4 of Chapter 3; see also exercise 7.

By making use of the piecewise constant functions $\{B_{i,0}\}_{i=3}^n$ defined in (1.13) and the abbreviation $\mathbf{p}_{i,2}(t) = \mathbf{p}(t \mid \mathbf{c}_{i-2}, \mathbf{c}_{i-1}, \mathbf{c}_i; t_{i-1}, t_i, t_{i+1}, t_{i+2})$, we can write $\mathbf{f}(t)$ as

$$\mathbf{f}(t) = \sum_{i=3}^n \mathbf{p}_{i,2}(t) B_{i,0}(t). \quad (1.17)$$

Two examples of quadratic spline curves are shown in Figure 1.13. We observe that the curves behave like Bézier curves at the two ends.

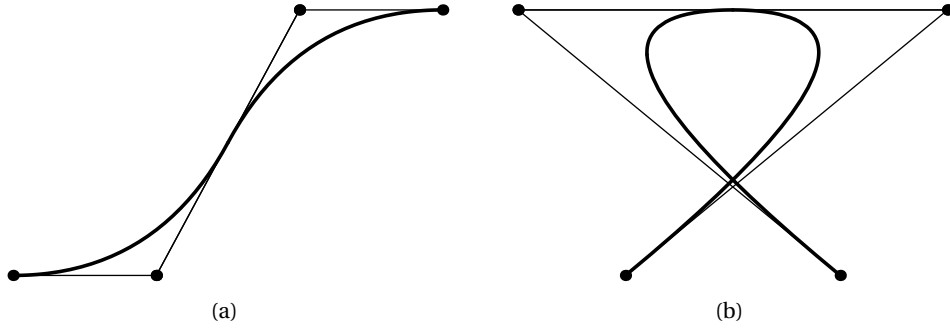


Figure 1.13. Two quadratic spline curves, both with knots $\mathbf{t} = (0, 0, 0, 1, 2, 2, 2)$.

1.5.3 Smoothness of spline curves

The geometric construction of one segment of a spline curve, however elegant and numerically stable it may be, would hardly be of much practical interest was it not for the fact that it is possible to smoothly join together neighbouring segments. We will study this in detail in Chapter 3, but the following theorem summarises the quadratic case.

Theorem 1.2. *If t_{i+1} occurs once among the knots $(t_j)_{j=i-2}^{i+3}$, the spline function*

$$f(t) = \mathbf{p}_{i,2,1}(t)B_{i,0}(t) + \mathbf{p}_{i+1,2,1}(t)B_{i+1,0}(t)$$

has continuous first derivative at the join t_{i+1} . If t_{i+1} occurs twice among the knots $(t_j)_{j=i-2}^{i+4}$, then

$$f(t) = \mathbf{p}_{i,2,1}(t)B_{i,0}(t) + \mathbf{p}_{i+2,2,1}(t)B_{i+2,0}(t)$$

is in general just continuous at t_{i+1} .

This ability to control the smoothness of a spline by varying the multiplicity of the knots is important in practical applications. For example it is often necessary to represent curves with a sharp corner (discontinuous derivative).

Two examples of spline curves with reduced smoothness are shown in Figure 1.14. Figure (a) shows a quadratic spline with a double knot and a discontinuous derivative at the encircled point, while Figure (b) shows a cubic spline with a double knot and a discontinuous second derivative at the encircled point.

1.5.4 Representing spline curves in terms of basis functions

In Section 1.4 we saw that a Bézier curve can be written as a linear combination of Bernstein polynomials with the control points as coefficients, see equation (1.10). In this section we want to sketch a similar representation for spline curves.

Let us consider the case of linear splines. Suppose we have n control points $(\mathbf{c}_i)_{i=1}^n$ and the n knots $\mathbf{t} = (t_i)_{i=2}^{n+1}$ which we assume are distinct for simplicity. We have seen that a typical

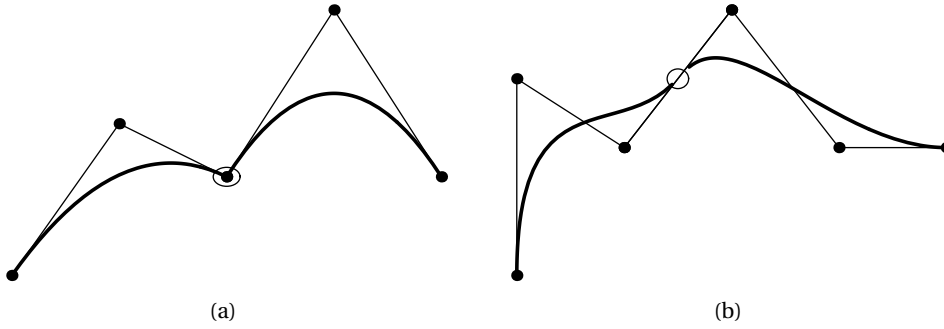


Figure 1.14. A quadratic spline with a double knot at the circled point (a) and a cubic spline with a double knot at the circled point (b).

linear spline can be written

$$\mathbf{f}(t) = \sum_{i=2}^n \mathbf{p}_{i,1}(t) B_{i,0}(t), \quad t \in [t_2, t_{n+1}], \quad (1.18)$$

where $\{B_{i,0}\}_{i=2}^n$ are given by (1.13). By making use of (1.11), adapted to segment i , we obtain

$$\begin{aligned} \mathbf{f}(t) &= \sum_{i=2}^n \left(\frac{t-t_i}{t_{i+1}-t_i} \mathbf{p}_i B_{i,0}(t) + \frac{t_{i+1}-t}{t_{i+1}-t_i} \mathbf{p}_{i-1} B_{i,0}(t) \right) \\ &= \sum_{i=2}^{n-1} \left(\frac{t-t_i}{t_{i+1}-t_i} B_{i,0}(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} B_{i+1,0}(t) \right) \mathbf{p}_i + \\ &\quad \frac{t_3-t}{t_3-t_2} B_{2,0}(t) \mathbf{p}_1 + \frac{t-t_n}{t_{n+1}-t_n} B_{n,0}(t) \mathbf{p}_n. \end{aligned} \quad (1.19)$$

By introducing the functions

$$B_{i,1}(t) = \frac{t-t_i}{t_{i+1}-t_i} B_{i,0}(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} B_{i+1,0}(t) \quad (1.20)$$

for $i = 2, \dots, n$, we can then write $\mathbf{f}$ as

$$\mathbf{f}(t) = \sum_{i=1}^n \mathbf{p}_i B_{i,1}(t).$$

These functions are called basic splines, or more briefly, *B-splines*.

The same argument can be repeated in the quadratic case and leads to quadratic B-splines, defined by the recurrence

$$B_{i,2}(t) = \frac{t-t_i}{t_{i+2}-t_i} B_{i,1}(t) + \frac{t_{i+3}-t}{t_{i+3}-t_{i+1}} B_{i+1,1}(t), \quad (1.21)$$

starting with $B_{i,0}$ as defined in (1.13). The next chapter starts with this recurrence relation, generalised to degree d , and from this explores the most basic properties B-splines.

1.6 Conclusion

Our starting point in this chapter was the need for efficient and numerically stable methods for determining smooth curves from a set of points. We considered three possibilities, namely polynomial interpolation, Bézier curves and spline curves. In their simplest forms, all three methods produce polynomial curves that can be expressed as

$$\mathbf{g}(t) = \sum_{i=0}^d \mathbf{a}_i F_i(t),$$

where d is the polynomial degree, $(\mathbf{a}_i)_{i=0}^d$ are the coefficients and $\{F_i\}_{i=0}^d$ are the basis polynomials.

The differences between the three methods lie in the choice of basis polynomials, or equivalently, how the given points relate to the final curve. In the case of interpolation the coefficients are points on the curve with the Lagrange polynomials as basis polynomials. For Bézier and spline curves the coefficients are control points with the property that the curve itself lies inside the convex hull of the control points, while the basis polynomials are the Bernstein polynomials and (one segment of) B-splines respectively. For our purposes Bézier and spline curves are preferable since they can be constructed by forming repeated convex combinations which should ensure that the resulting curves are relatively insensitive to round-off errors.

The use of convex combinations also means that the constructions have simple geometric interpretations. This has the advantage that a Bézier curve or spline curve can conveniently be manipulated interactively by manipulating the curve's control points. As we saw in Section 1.4.2 this also makes it quite simple to link several Bézier curves smoothly together. The advantage of spline curves over Bézier curves is that smoothness between neighbouring polynomial pieces is built into the basis functions (B-splines) instead of being controlled by constraining control points according to specific rules.

In the coming chapters we are going to study various aspects of splines, primarily by uncovering properties of B-splines. This means that our point of view will shift somewhat, from spline curves to spline functions (each control point is a real number), since B-splines are functions. However, virtually all the properties we obtain for spline functions also make sense for spline curves, and even tensor product spline surfaces, see Chapters 6 and 7.

1.7 Exercises

1.1 Convex set

Recall that a subset $\mathbb{A}$ of $\mathbb{R}^n$ is said to be *convex* if whenever we pick two points in $\mathbb{A}$, the line connecting the two points is also in $\mathbb{A}$. In this exercise we are going to prove that the convex hull of a finite set of points is the smallest convex set that contains the points. This is obviously true if we only have one or two points. To gain some insight we will first show that it is also true in the case of three points before we proceed to the general case. We will use the notation $\mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_n)$ to denote the convex hull of the points $\mathbf{c}_1, \dots, \mathbf{c}_n$.

- a) Suppose we have three points $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$. We know that the convex hull of $\mathbf{c}_1$ and $\mathbf{c}_2$ is the straight line segment that connects the points. Let $\tilde{\mathbf{c}}$ be a point on this line, i.e.,

$$\tilde{\mathbf{c}} = (1 - \lambda)\mathbf{c}_1 + \lambda\mathbf{c}_2 \quad (1.22)$$

for some λ with $0 \leq \lambda \leq 1$. Show that any convex combination of $\tilde{\mathbf{c}}$ and $\mathbf{c}_3$ is a convex combination of $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$. Explain why this proves that $\mathbb{CH}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ contains the triangle with the three points at its vertexes. The situation is depicted graphically in Figure 1.2.

- b) It could be that $\mathbb{CH}(\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ is larger than the triangle formed by the three points since the convex combination that we considered above was rather special. We will now show that this is not the case.

Show that any convex combination of $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ gives rise to a convex combination on the form (1.22). Hint: Show that if $\mathbf{c}$ is a convex combination of the three points, then we can write

$$\begin{aligned} \mathbf{c} &= \lambda_1 \mathbf{c}_1 + \lambda_2 \mathbf{c}_2 + \lambda_3 \mathbf{c}_3 \\ &= (1 - \lambda_3) \tilde{\mathbf{c}} + \lambda_3 \mathbf{c}_3, \end{aligned}$$

where $\tilde{\mathbf{c}}$ is some convex combination of $\mathbf{c}_1$ and $\mathbf{c}_2$. Why does this prove that the convex hull of three points coincides with the triangle formed by the points? Explain why this shows that if $\mathbb{B}$ is a convex set that contains $\mathbf{c}_1$, $\mathbf{c}_2$ and $\mathbf{c}_3$ then $\mathbb{B}$ must also contain the convex hull of the three points which allows us to conclude that the convex hull of three points is the smallest convex set that contains the points.

- c) The general proof that the convex hull of n points is the smallest convex set that contains the points is by induction on n . We know that this is true for $n = 2$ and $n = 3$ so we assume that $n \geq 4$. Let $\mathbb{B}$ be a convex set that contains $\mathbf{c}_1, \dots, \mathbf{c}_n$. Use the induction hypothesis and show that $\mathbb{B}$ contains any point on a straight line that connects $\mathbf{c}_n$ and an arbitrary point in $\mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_{n-1})$.
- d) From what we have found in (c) it is not absolutely clear that any convex set $\mathbb{B}$ that contains $\mathbf{c}_1, \dots, \mathbf{c}_n$ also contains *all* convex combinations of the points. To settle this show that any point $\mathbf{c}$ in $\mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_n)$ can be written $\mathbf{c} = \lambda \tilde{\mathbf{c}} + (1 - \lambda)\mathbf{c}_n$ for some λ in $[0, 1]$ and some point $\tilde{\mathbf{c}}$ in $\mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_{n-1})$. Hint: Use a trick similar to that in (b).

Explain why this lets us conclude that $\mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_n)$ is the smallest convex set that contains $\mathbf{c}_1, \dots, \mathbf{c}_n$.

1.2 Interpolating curves of general degree

In this exercise we are going to extend the construction of interpolating curves to higher degrees.

- a) Suppose we are given four points $(\mathbf{c}_i)_{i=0}^3$ and four strictly increasing parameters

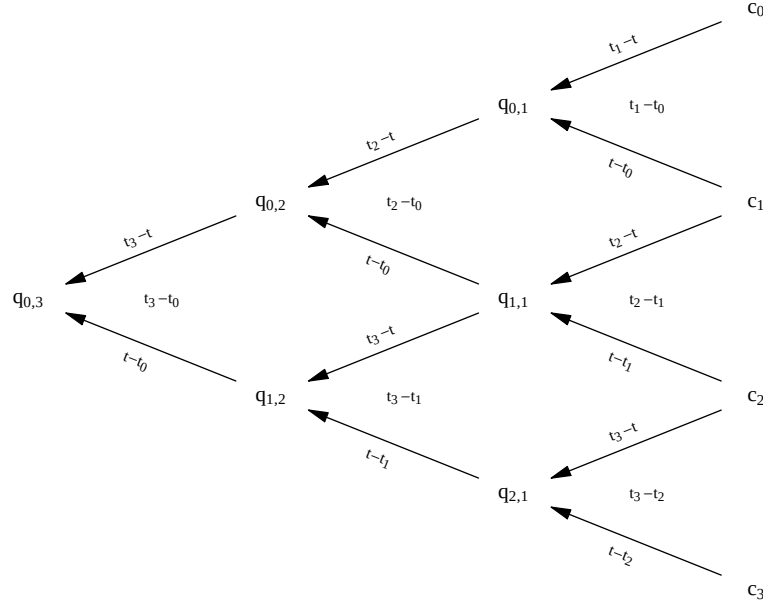


Figure 1.15. Computing a point on a cubic interpolating curve.

$t = (t_i)_{i=0}^3$. Form the two quadratic interpolants

$$q_{0,2}(t) = q(t \mid c_0, c_1, c_2; t_0, t_1, t_2),$$

$$q_{1,2}(t) = q(t \mid c_1, c_2, c_3; t_1, t_2, t_3),$$

and combine these to obtain the cubic curve $q_{0,3}(t)$,

$$q_{0,3}(t) = \frac{t_3 - t}{t_3 - t_0} q_{0,2}(t) + \frac{t - t_0}{t_3 - t_0} q_{1,2}(t).$$

Verify that $q_{0,3}$ satisfies the interpolation conditions $q_{0,3}(t_i) = c_i$ for $i = 0, 1, 2, 3$.

- b) Suppose more generally that we are given $d + 1$ points $(c_i)_{i=0}^d$ and parameters $(t_i)_{i=0}^d$. Construct the curve $q_{0,d}$ of degree d by the affine combination

$$q_{0,d}(t) = \frac{t_d - t}{t_d - t_0} q_{0,d-1}(t) + \frac{t - t_0}{t_d - t_0} q_{1,d-1}(t) \quad (1.23)$$

where $q_{0,d-1}$ is the curve of degree $d - 1$ that interpolates $(c_i)_{i=0}^{d-1}$ at $(t_i)_{i=0}^{d-1}$ and similarly for $q_{1,d-1}$.

Verify by induction that $q_{0,d}$ satisfies the interpolation conditions $q_{0,d}(t_i) = c_i$ for $i = 0, \dots, d$.

- c) Write a program that implements interpolation by repeated use of (1.23) (Figure 1.15 may be helpful). Test your program on some examples like the ones in Figure 1.16, and try to explain the somewhat strange behaviour in examples (b) and (d).

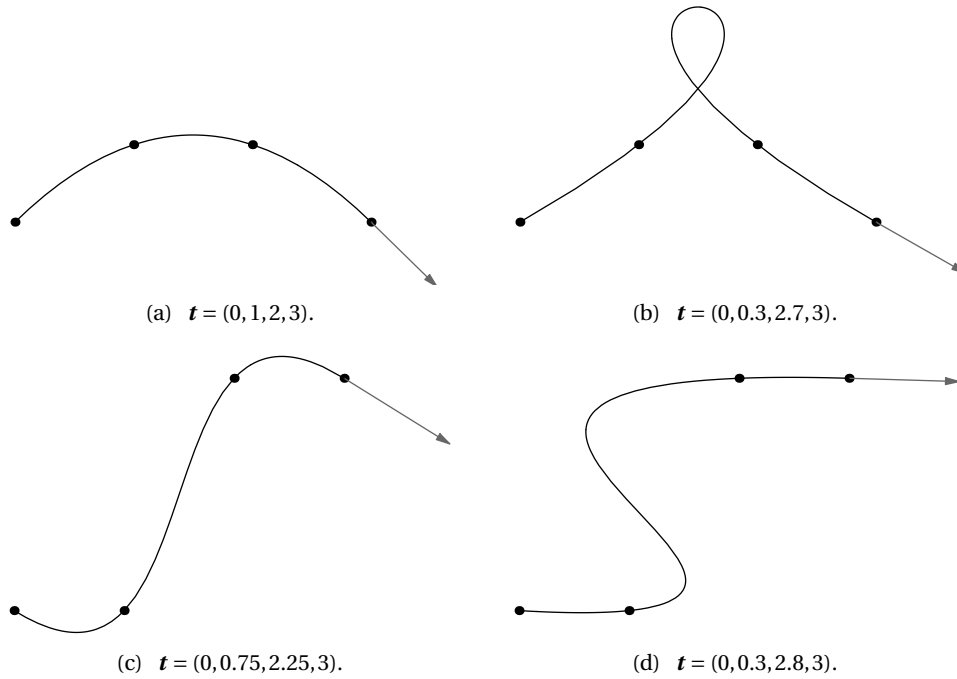


Figure 1.16. Some examples of cubic interpolation.

d) Show that $\mathbf{q}_{0,d}$ can be written as

$$\mathbf{q}_{0,d}(t) = \mathbf{c}_0 \ell_{0,d}(t) + \mathbf{c}_1 \ell_{1,d}(t) + \cdots + \mathbf{c}_d \ell_{d,d}(t), \quad (1.24)$$

where the functions $\{\ell_{i,d}\}_{i=0}^d$ are the *Lagrange polynomials* of degree d given by

$$\ell_{i,d}(t) = \prod_{\substack{0 \leq j \leq d \\ j \neq i}} \frac{(t - t_j)}{t_i - t_j}. \quad (1.25)$$

Verify that

$$\ell_{i,d}(t_k) = \begin{cases} 1, & \text{if } k = i, \\ 0, & \text{otherwise.} \end{cases}$$

1.3 Bézier curves of general degree

Bézier curves can be extended to higher degrees, just like the interpolating curves in exercise 2.

a) Start with four points $(\mathbf{c}_i)_{i=0}^3$ and form the curve $\mathbf{p}_{3,3}(t) = \mathbf{p}(t \mid \mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ by taking a convex combination of two quadratic curves,

$$\mathbf{p}_{3,3}(t) = (1 - t)\mathbf{p}_{2,2}(t) + t\mathbf{p}_{3,2}(t).$$

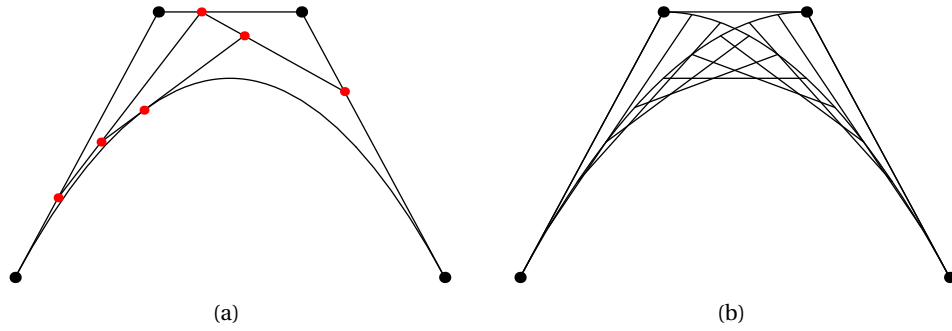


Figure 1.17. Constructing a Bézier curve from four points.

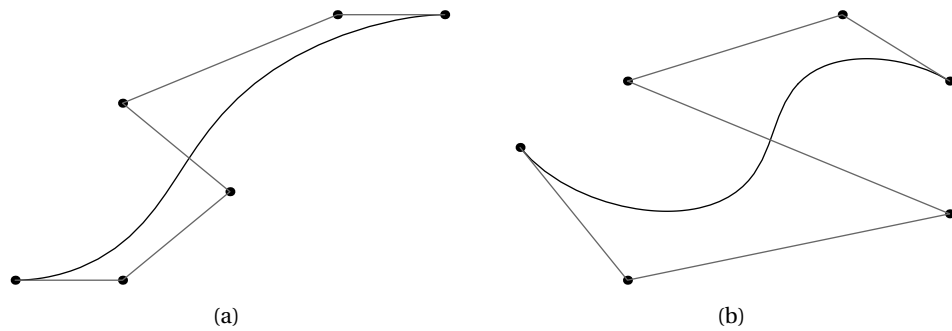


Figure 1.18. Two Bézier curves of degree five.

The two curves on the right are defined as in (1.9).

Show that $\mathbf{p}_{3,3}(t)$ can be expressed as

$$\mathbf{p}_{3,3}(t) = (1-t)^3 \mathbf{c}_0 + 3t(1-t)^2 \mathbf{c}_1 + 3t^2(1-t) \mathbf{c}_2 + t^3 \mathbf{c}_3,$$

and conclude that $\mathbf{p}_{3,3}(t)$ is a convex combination of the points $(\mathbf{c}_i)_{i=0}^3$.

This is a *cubic Bézier curve*, and the four polynomials on the right are the *cubic Bernstein polynomials*. The construction is illustrated in Figure 1.17. Figure (a) shows the construction for a given value of t , and in Figure (b) the cubic and the two quadratic curves are shown together with the lines connecting corresponding points on the two quadratics (every point on the cubic lies on such a line).

- b) Generalise the construction and the result in (a) by starting with $d+1$ points $(\mathbf{c}_i)_{i=0}^d$.

Two examples of Bézier curves of degree five are shown in Figure 1.18.

1.4 Visualise the computation of a point on a cubic Bézier curve by arranging the computations in a table similar to the one in Figure 1.15.

1.5 It is sometimes useful to represent Bézier curves on an interval $[a, b]$ rather than $[0, 1]$.

Show that if $\mathbf{p}_{d,d}(t)$ is a Bézier curve defined on $[0, 1]$ then

$$\tilde{\mathbf{p}}_{d,d}(t) = \mathbf{p}_{d,d}((t-a)/(b-a))$$

is a Bézier curve defined on the interval $[a, b]$, and derive formulas for the corresponding basis polynomials, the Bernstein polynomials for the interval $[a, b]$.

- 1.6** Repeat the derivations in equations (1.19) for the quadratic spline curve (1.17) and show that it can be expressed in terms of basis functions that satisfy the recurrence relation (1.21).
- 1.7** Show that a quadratic spline is continuous and has a continuous derivative at a single knot.

Basic properties of splines and B-splines

In Chapter 1 we introduced splines through a geometric construction of curves based on repeated averaging, and it turned out that a natural representation of spline curves was as linear combinations of B-splines. In this chapter we study the most basic properties of B-splines in detail. The starting point is the definition of B-splines via the recurrence relation that generalises (1.21). In Section 2.1 we explore some of the most obvious properties of B-splines, and then in Section 2.3 we transfer some of these properties to linear combinations of B-splines. In Section 2.4 we give a matrix representation of splines and B-splines from which it is easy to deduce computational algorithms in Section 2.5. The matrix representation is also the basis for our development of much of the theory in later chapters.

2.1 The recurrence relation for B-splines

We saw in Section 1.5 that a quadratic spline curve can be represented in terms of certain basis functions which we called B-splines. In this section we will define B-splines of any degree and deduce some of their most basic properties. Since we will mainly be working with functions in the remainder of the book, we use x as the independent variable.

Definition 2.1. Let d be a nonnegative integer and let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$, the knot vector or knot sequence, be a nondecreasing sequence of real numbers of length at least $d+2$. If $1 \leq j \leq n$, the j th B-spline of degree d with knots $\mathbf{t}$ is defined by

$$B_{j,d,\mathbf{t}}(x) = \frac{x - t_j}{t_{j+d} - t_j} B_{j,d-1,\mathbf{t}}(x) + \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} B_{j+1,d-1,\mathbf{t}}(x), \quad (2.1)$$

for all real numbers x , starting with

$$B_{j,0,\mathbf{t}}(x) = \begin{cases} 1, & \text{if } t_j \leq x < t_{j+1}; \\ 0, & \text{otherwise.} \end{cases} \quad (2.2)$$

Here, the convention is assumed that $0/0 = 0$.

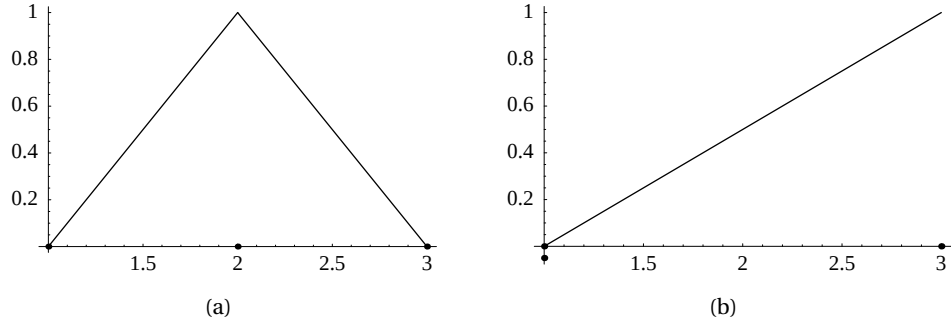


Figure 2.1. A linear B-spline with simple knots (a) and double knots (b).

When there is no chance of ambiguity, some of the subscripts will be dropped and the B-spline written as either $B_{j,d}$, $B_{j,t}$, or simply B_j . We say that a knot has *multiplicity* m if it appears m times in the knot sequence. Knots of multiplicity one, two and three are also called *simple*, *double* and *triple* knots.

Example 2.2 (B-splines of degree 1). One application of the recurrence relation gives

$$B_{j,1}(x) = \frac{x - t_j}{t_{j+1} - t_j} B_{j,0}(x) + \frac{t_{j+2} - x}{t_{j+2} - t_{j+1}} B_{j+1,0}(x) = \begin{cases} (x - t_j)/(t_{j+1} - t_j), & \text{if } t_j \leq x < t_{j+1}; \\ (t_{j+2} - x)/(t_{j+2} - t_{j+1}), & \text{if } t_{j+1} \leq x < t_{j+2}; \\ 0, & \text{otherwise.} \end{cases}$$

A plot of this *hat function* is shown in Figure 2.1 (a) in a typical case where $t_j < t_{j+1} < t_{j+2}$. The figure shows clearly that $B_{j,1}$ consists of linear polynomial pieces, with breaks at the knots. In Figure 2.1 (b), the two knots t_{j+1} and t_{j+2} are identical; then the second linear piece is identically zero since $B_{j+1,0} = 0$, and $B_{j,1}$ is discontinuous. This provides an illustration of the smoothness properties of B-splines: a linear B-spline is discontinuous at a double knot, but continuous at simple knots.

2.2 Some simple consequences of the recurrence relation

When are B-splines zero and when are they positive? What are the knots that influence a B-spline? How smooth are B-splines? How can they be computed efficiently? These are some basic questions concerning B-splines that we will attempt to answer in this chapter, and Lemma 2.3 below sums up a number of the answers.

2.2.1 Basic properties

Many properties of B-splines can be deduced directly from the definition. One of the most basic properties is that

$$B_{j,d}(x) = 0 \quad \text{for all } x \text{ when } t_j = t_{j+d+1}.$$

This is true by definition for $d = 0$. If it is true for B-splines of degree $d-1$, the zero convention means that if $t_j = t_{j+d+1}$ then both $B_{j,d-1}(x)/(t_{j+d} - t_j)$ and $B_{j+1,d-1}(x)/(t_{j+1+d} - t_{j+1})$ on the

right in (2.1) are zero, and hence $B_{j,d}(x)$ is zero. The recurrence relation can therefore be expressed more explicitly as

$$B_{j,d}(x) = \begin{cases} 0, & \text{if } t_j = t_{j+1+d}; \\ s_1(x), & \text{if } t_j < t_{j+d} \text{ and } t_{j+1} = t_{j+1+d}; \\ s_2(x), & \text{if } t_j = t_{j+d} \text{ and } t_{j+1} < t_{j+1+d}; \\ s_1(x) + s_2(x), & \text{otherwise;} \end{cases} \quad (2.3)$$

where

$$s_1(x) = \frac{x - t_j}{t_{j+d} - t_j} B_{j,d-1}(x) \quad \text{and} \quad s_2(x) = \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} B_{j+1,d-1}(x)$$

for all x .

The B-spline $B_{j,d}$ depends only on the knots $(t_k)_{k=j}^{j+d+1}$. For B-splines of degree 0 this is clear from equation (2.2), and Example 2.2 shows that it is also true for B-splines of degree 1. To show that it is true in general, we use induction and assume that $B_{j,d-1}$ only depends on $(t_k)_{k=j}^{j+d}$ and $B_{j+1,d-1}$ only depends on $(t_k)_{k=j+1}^{j+d+1}$. By examining the recurrence relation (2.1) we see that then $B_{j,d}$ can only depend on the knots $(t_k)_{k=j}^{j+d+1}$, as we claimed.

The notation $B_{j,d}(x) = B[t_j, \dots, t_{j+d+1}](x)$ will sometimes be used to emphasise the dependence of a B-spline on the individual knots. For example, if $d \geq 2$ and if we set $(a, b, \dots, c, d) = (t_j, t_{j+1}, \dots, t_{j+d}, t_{j+d+1})$, then (2.1) can be written

$$B[a, b, \dots, c, d](x) = \frac{x-a}{c-a} B[a, b, \dots, c](x) + \frac{d-x}{d-b} B[b, \dots, c, d](x). \quad (2.4)$$

A more complex property of B-splines is that they are *translation invariant*. Mathematically this is expressed by the formula

$$B[t_j + y, \dots, t_{j+d+1} + y](x + y) = B[t_j, \dots, t_{j+d+1}](x) \quad x, y \in \mathbb{R}. \quad (2.5)$$

We argue by induction, and start by checking the case $d = 0$. We have

$$B[t_j + y, t_{j+1} + y](x + y) = \begin{cases} 1, & \text{if } t_j + y \leq x + y < t_{j+1} + y; \\ 0, & \text{otherwise} \end{cases} = \begin{cases} 1, & \text{if } t_j \leq x < t_{j+1}; \\ 0, & \text{otherwise,} \end{cases}$$

so equation (2.5) holds for $d = 0$. Suppose that the translation invariance holds for B-splines of degree $d - 1$. In the recurrence (2.1) for the left-hand-side of (2.5) the first coefficient $(x - t_j)/(t_{j+d} - t_j)$ can be written

$$\frac{(x + y) - (t_j + y)}{(t_{j+d} + y) - (t_j + y)} = \frac{x - t_j}{t_{j+d} - t_j},$$

i.e., the same as before translation. This also holds for the other coefficient $(t_{j+d+1} - x)/(t_{j+d+1} - t_{j+1})$ in (2.1). Since the two B-splines of degree $d - 1$ are translation invariant by the induction hypothesis, we conclude that (2.5) holds for all polynomial degrees.

We have now seen a number of examples of B-splines and some characteristic features are evident. The following lemma sums up the most basic properties.

Lemma 2.3. *Let d be a nonnegative polynomial degree and let $\mathbf{t} = (t_j)$ be a knot sequence. The B-splines on $\mathbf{t}$ have the following properties:*

1. Local knots. *The j th B-spline $B_{j,d}$ depends only on the knots $t_j, t_{j+1}, \dots, t_{j+d+1}$.*
2. Local support.
 - (a) *If x is outside the interval $[t_j, t_{j+d+1})$ then $B_{j,d}(x) = 0$. In particular, if $t_j = t_{j+d+1}$ then $B_{j,d}$ is identically zero.*
 - (b) *If x lies in the interval $[t_\mu, t_{\mu+1})$ then $B_{j,d}(x) = 0$ if $j < \mu - d$ or $j > \mu$.*
3. Positivity. *If $x \in (t_j, t_{j+d+1})$ then $B_{j,d}(x) > 0$. The closed interval $[t_j, t_{j+d+1}]$ is called the support of $B_{j,d}$.*
4. Piecewise polynomial. *The B-spline $B_{j,d}$ can be written*

$$B_{j,d}(x) = \sum_{k=j}^{j+d} B_{j,d}^k(x) B_{k,0}(x) \quad (2.6)$$

where each $B_{j,d}^k(x)$ is a polynomial of degree d .

5. Special values. *If $z = t_{j+1} = \dots = t_{j+d} < t_{j+d+1}$ then $B_{j,d}(z) = 1$ and $B_{i,d}(z) = 0$ for $i \neq j$.*
6. Smoothness. *If the number z occurs m times among $t_j, \dots, t_{j+d+1}$ then the derivatives of $B_{j,d}$ of order $0, 1, \dots, d - m$ are all continuous at z .*
7. Translation invariance. *The value of a B-spline is not changed when the argument of the B-spline and all its knots are shifted by the same number,*

$$B[t_j + y, \dots, t_{j+d+1} + y](x + y) = B[t_j, \dots, t_{j+d+1}](x) \quad \text{for all } x \text{ and } y \text{ in } \mathbb{R}. \quad (2.7)$$

Proof. Properties 1–3 follow directly, by induction, from the recurrence relation, see exercise 5. In Section 1.5 in Chapter 1 we saw that the construction of splines produced piecewise polynomials, so this explains property 4. Property 5 is proved in exercise 8 and property 6 will be proved in Chapter 3. Property 7 was proved above. ■

2.2.2 More examples of B-splines

A number of different quadratic B-splines are shown in Figure 2.2. Note that the B-spline $B[0, 1, 2, 3](x)$ consists of three nonzero polynomial pieces. In general the number of nonzero pieces depends on the multiplicity of the knots. As an example, the functions $B[0, 0, 0, 1](x)$ and $B[0, 1, 1, 1](x)$ consist of only one nonzero piece. Figure 2.2 also illustrates the smoothness properties of B-splines: At a single knot a quadratic B-spline is continuous and has a

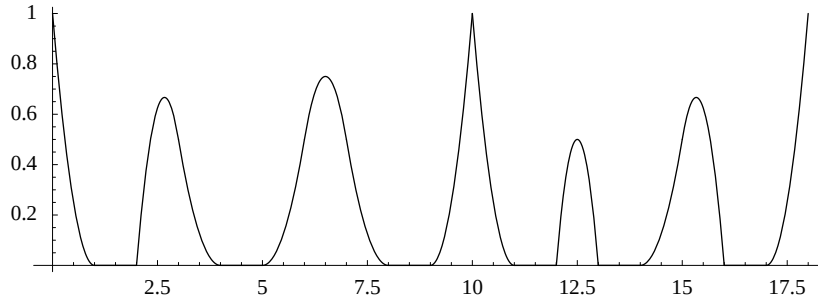


Figure 2.2. From left to right we see the quadratic B-splines $B[0,0,0,1](x)$, $B[2,2,3,4](x)$, $B[5,6,7,8](x)$, $B[9,10,10,11](x)$, $B[12,12,13,13](x)$, $B[14,15,16,16](x)$, and $B[17,18,18,18](x)$.

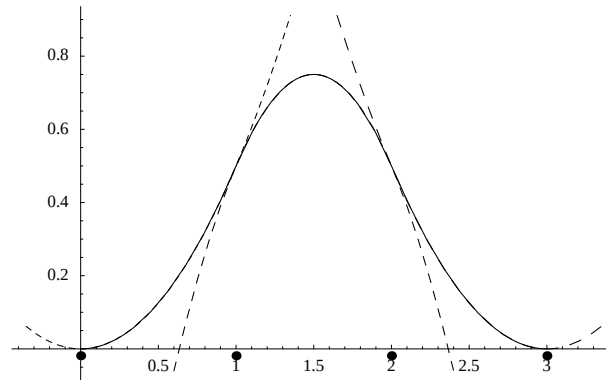


Figure 2.3. The different polynomial pieces of a quadratic B-spline.

continuous derivative, at a double knot it is continuous, while at a triple knot it is discontinuous. These claims will be proved in the next chapter.

Figure 2.3 shows the quadratic B-spline $B[0,1,2,3](x)$ together with its constituent polynomial pieces. Note how the three parabolas join together smoothly to make the B-spline have a continuous first derivative at every point.

Figure 2.4 shows some cubic B-splines. The middle B-spline, $B[9,10,11,12,13]$, has simple knots and its second derivative is therefore continuous for all real numbers x , including the knots. We will show that in general a cubic B-spline has $3 - m$ continuous derivatives at a knot of multiplicity m for $m = 1, 2, 3$. A cubic B-spline with a knot of multiplicity 4 is discontinuous at the knot.

Example 2.4 (Uniform B-splines). The B-splines on a uniform knot vector are of special interest. Let the knots be the set $\mathbb{Z}$ of all integers. We index this knot sequence by letting $t_j = j$ for all integers j . We denote the uniform B-spline of degree $d \geq 0$ by

$$M_d(x) = B_{0,p}(x) = B[0,1,\dots,d+1](x), \quad x \in \mathbb{R}. \quad (2.8)$$

The functions M_d are also called *cardinal B-splines*.

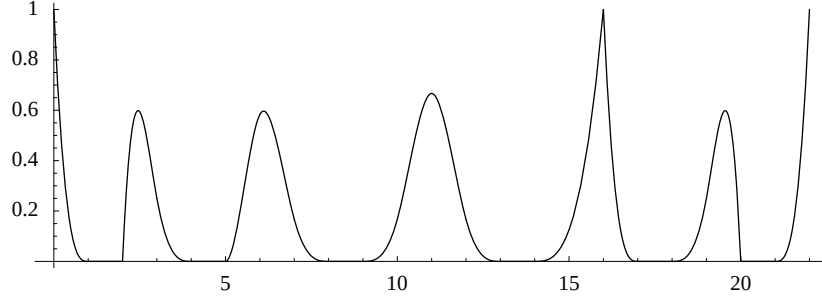


Figure 2.4. From left to right we see the cubic B-splines $B[0, 0, 0, 0, 1](x)$, $B[2, 2, 2, 3, 4](x)$, $B[5, 5, 6, 7, 8](x)$, $B[9, 10, 11, 12, 13](x)$, $B[14, 16, 16, 16, 17](x)$, $B[18, 19, 20, 20, 20](x)$, and $B[21, 22, 22, 22, 22](x)$.

On the knots $\mathbb{Z}$ all B-splines can be written as translates of the function M_d . Using (2.5) we have

$$B_{j,d}(x) = B[j, j+1, \dots, j+d+1](x) = B[0, 1, \dots, d+1](x-j) = M_d(x-j) \quad \text{for all } j.$$

In particular, $B_{1,d-1}(x) = B[1, \dots, d+1](x) = M_{d-1}(x-1)$ and the recurrence relation implies that for $d \geq 1$

$$M_d(x) = \frac{x}{d} M_{d-1}(x) + \frac{d+1-x}{d} M_{d-1}(x-1). \quad (2.9)$$

The first few uniform B-splines are computed in Exercise 3. As we shall see in Chapter 3, the B-spline M_d has $d-1$ continuous derivatives at the knots. The quadratic cardinal B-spline M_2 is shown in Figure 2.2, translated to the interval $[5, 8]$, while M_3 is shown in Figure 2.4, translated to $[9, 13]$.

The Bernstein polynomials that appeared in the representation of Bézier curves in Section 1.4 are special cases of B-splines.

Theorem 2.5 (Bernstein polynomials). *On the interval $[0, 1]$ the j th Bernstein polynomial is given by*

$$b_{j,d}(x) = \binom{d}{j} x^j (1-x)^{d-j} = B[\overbrace{0, \dots, 0}^{d+1-j}, \overbrace{1, \dots, 1}^{j+1}](x), \quad \text{for } j = 0, \dots, d, \quad x \in [0, 1]. \quad (2.10)$$

Proof. We use induction on the degree. Clearly (2.10) holds for $d = 0$. Suppose it holds for $d-1$. By the recurrence relation (2.4) and Exercise 7 we find

$$\begin{aligned} B[\overbrace{0, \dots, 0}^{d+1-j}, \overbrace{1, \dots, 1}^{j+1}](x) &= x B[\overbrace{0, \dots, 0}^{d+1-j}, \overbrace{1, \dots, 1}^j](x) + (1-x) B[\overbrace{0, \dots, 0}^{d-j}, \overbrace{1, \dots, 1}^{j+1}](x) \\ &= x b_{j-1,d-1}(x) + (1-x) b_{j,d-1}(x) = b_{j,d}(x). \end{aligned} \quad (2.11)$$

This is also valid for $j = 0$ and $j = d$ if we define $b_{j,d-1}(x) = 0$ for $j < 0$ and $j \geq d$. ■

2.3 Linear combinations of B-splines

B-splines play a central role in the representation of spline curves. The purpose of this section is to define precisely what we mean by spline functions and spline curves and related concepts like the control polygon.

2.3.1 Spline functions

The B-spline $B_{j,d}$ depends on the knots $t_j, \dots, t_{j+1+d}$. This means that if the knot vector is given by $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ for some positive integer n , we can form n B-splines $\{B_{j,d}\}_{j=1}^n$ of degree d associated with this knot vector. A linear combination of B-splines, or a spline function, is a combination of B-splines on the form

$$f = \sum_{j=1}^n c_j B_{j,d}, \quad (2.12)$$

where $\mathbf{c} = (c_j)_{j=1}^n$ are n real numbers. We formalise this in a definition.

Definition 2.6 (Spline functions). *Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a nondecreasing sequence of real numbers, i.e., a knot vector for a total of n B-splines. The linear space of all linear combinations of these B-splines is the spline space $\mathbb{S}_{d,\mathbf{t}}$ defined by*

$$\mathbb{S}_{d,\mathbf{t}} = \text{span}\{B_{1,p}, \dots, B_{n,p}\} = \left\{ \sum_{j=1}^n c_j B_{j,d} \mid c_j \in \mathbb{R} \text{ for } 1 \leq j \leq n \right\}.$$

An element $f = \sum_{j=1}^n c_j B_{j,d}$ of $\mathbb{S}_{d,\mathbf{t}}$ is called a spline function, or just a spline, of degree d with knots $\mathbf{t}$, and $(c_j)_{j=1}^n$ are called the B-spline coefficients of f .

If $f, g \in \mathbb{S}_{d,\mathbf{t}}$ and $\alpha \in \mathbb{R}$ then $f + g \in \mathbb{S}_{d,\mathbf{t}}$ and $\alpha f \in \mathbb{S}_{d,\mathbf{t}}$. It follows that $\mathbb{S}_{d,\mathbf{t}}$ is a linear space of functions.

It will often be the case that the exact length of the knot vector is of little interest. Then we may write a spline as $\sum_j c_j B_{j,d}$ without specifying the upper and lower bounds on j .

Example 2.7 (A linear and quadratic spline). In Figure 2.5 (a) we show two splines. In (a) on the knot vector $\{0, 0, 1, 2, 4, 7, 8, 9, 9\}$ we see the linear spline

$$f = B_{1,1} + 2B_{3,1} - B_{4,1} + B_{5,1} + B_{6,1} + 2B_{7,1}.$$

In (b) we plotted the function $\sin(\pi x/2)$ together with the spline approximation

$$s(x) = B_{2,2}(x) + B_{3,2}(x) - B_{4,2}(x) - \sqrt{2}B_{5,2}(x)$$

on the knot vector $\mathbf{t} = (t_j)_{j=1}^8 = (0, 0, 0, 1, 2, 3, 3, 3)$. See Exercises 12 and 13 for further discussion of these examples

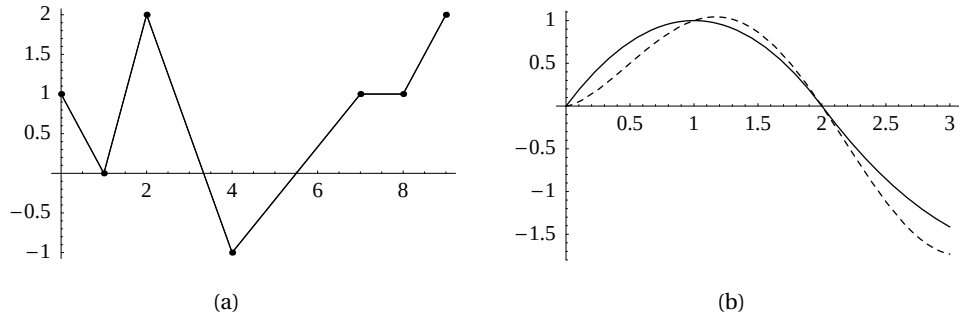


Figure 2.5. A linear spline interpolating data (a), and a quadratic spline (solid) that approximates $\sin(\pi x/2)$ (dashed).

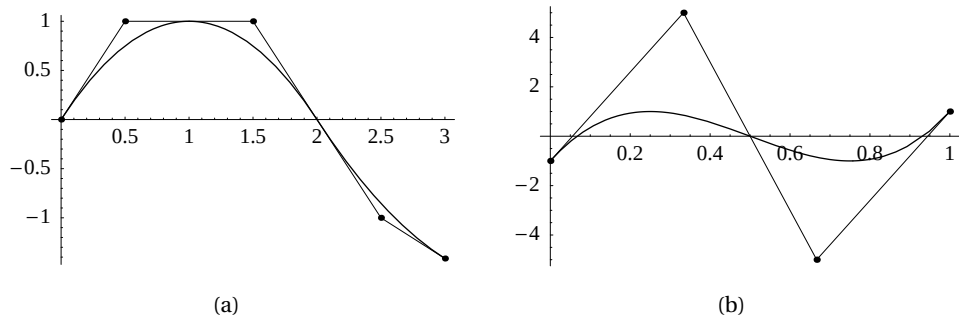


Figure 2.6. The quadratic spline from 2.7 with its control polygon (a) and the cubic Chebyshev polynomial with its control polygon (b) developed in Exercise 15.

Note that the knot vectors in the above example both have knots of multiplicity $d + 1$ at both ends. If in addition no knot occurs with multiplicity higher than $d + 1$ (as in the examples), the knot vector is said to be $d + 1$ -regular.

When we introduced spline curves in Chapter 1, we saw that a curve mimicked the shape of its control polygon in an intuitive way. The control polygon of a spline function is not quite as simple as for curves since the B-spline coefficients of a spline function is a number. What is needed is an abscissa to associate with each coefficient.

Definition 2.8 (Control polygon for spline functions). Let $f = \sum_{j=1}^n c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,t}$. The control points of f are the points with coordinates (t_j^*, c_j) for $j = 1, \dots, n$, where

$$t_j^* = \frac{t_{j+1} + \dots + t_{j+d}}{d}$$

are the knot averages of t . The control polygon of f is the piecewise linear function obtained by connecting neighbouring control points by straight lines.

Some spline functions are shown with their control polygons in Figures 2.6–2.7. It is quite

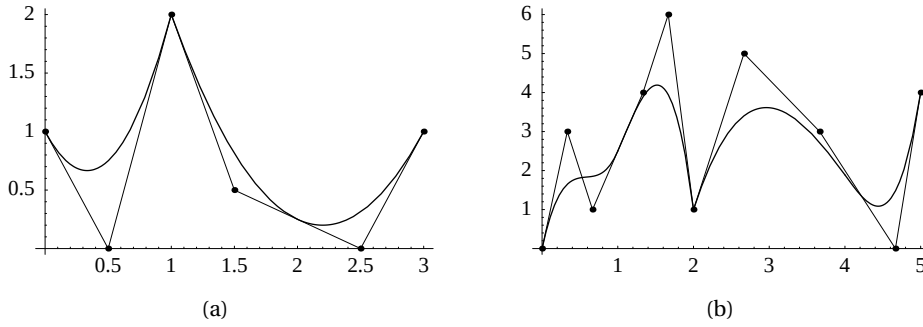


Figure 2.7. Two splines with corresponding control polygons. The spline in (a) is quadratic with knots $\mathbf{t} = (0, 0, 0, 1, 1, 2, 3, 3, 3)$ and B-spline coefficients $\mathbf{c} = (1, 0, 2, 1/2, 0, 1)$, while the spline in (b) is cubic with knots $\mathbf{t} = (0, 0, 0, 0, 1, 1, 2, 2, 2, 4, 5, 5, 5, 5)$ and B-spline coefficients $0, 3, 1, 4, 6, 1, 5, 3, 0, 4$.

striking how the spline is a smoothed out version of the control polygon. In particular we notice that at a knot with multiplicity at least d , the spline and its control polygon agree. This happens at the beginning and end of all the splines since we have used $d + 1$ -regular knot vectors, and also at some points in the interior for the splines in Figure 2.7. We also note that the control polygon is tangent to the spline function at a knot of multiplicity d or $d + 1$. This close relationship between a spline and its control polygon is a geometric instance of one of the many nice properties possessed by splines represented in terms of B-splines.

From our knowledge of B-splines we immediately obtain some basic properties of splines.

Lemma 2.9. Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for splines of degree d with $n \geq d + 1$, and let $f = \sum_{j=1}^n c_j B_{j,d}$ be a spline in $\mathcal{S}_{d,\mathbf{t}}$. Then f has the following properties:

1. If x is in the interval $[t_\mu, t_{\mu+1})$ for some μ in the range $d + 1 \leq \mu \leq n$ then

$$f(x) = \sum_{j=\mu-d}^{\mu} c_j B_{j,d}(x).$$

2. If $z = t_{j+1} = \dots = t_{j+d} < t_{j+d+1}$ for some j in the range $1 \leq j \leq n$ then $f(z) = c_j$.
3. If z occurs m times in $\mathbf{t}$ then f has continuous derivatives of order $0, \dots, d - m$ at z .

Proof. This follows directly from Lemma 2.3. ■

2.3.2 Spline curves

For later reference we give a precise definition of spline curves, although we have already made extensive use of them in Chapter 1.

In many situations spline functions will be the right tool to represent a set of data or some desired shape. But as we saw in Section 1.2 functions have some inherent restrictions in that

for a given x , a function can only take one scalar value. We saw that one way to overcome this restriction was by representing the x - and y -components by two different functions,

$$\mathbf{f}(u) = (f_1(u), f_2(u)).$$

Vector functions in higher dimensions are obtained by adding more components. We will be particularly interested in the special case where all the components are spline functions on a common knot vector.

Definition 2.10 (Spline curves). Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a nondecreasing sequence of real numbers, and let $s \geq 2$ be an integer. The space of all spline curves in $\mathbb{R}^s$ of degree d and with knots $\mathbf{t}$ is defined as

$$\mathbb{S}_{d,\mathbf{t}}^s = \left\{ \sum_{j=1}^n \mathbf{c}_j B_{j,d} \mid \mathbf{c}_j \in \mathbb{R}^s \text{ for } 1 \leq j \leq n \right\}.$$

More precisely, an element $f = \sum_{j=1}^n \mathbf{c}_j B_{j,d}$ of $\mathbb{S}_{d,\mathbf{t}}^s$ is called a spline vector function or a parametric spline curve of degree d with knots $\mathbf{t}$, and $(\mathbf{c}_j)_{j=1}^n$ are called the B-spline coefficients or control points of f .

We have already defined what we mean by the control polygon of a spline curve, but for easy reference we repeat the definition here.

Definition 2.11 (Control polygon for spline curves). Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for splines of degree d , and let $\mathbf{f} = \sum_{j=1}^n \mathbf{c}_j B_{j,d}$ be a spline curve in $\mathbb{S}_{d,\mathbf{t}}^s$ for $s \geq 2$. The control polygon of f is the piecewise linear function obtained by connecting neighbouring control points $(\mathbf{c}_j)_{j=1}^n$ by straight lines.

Some examples of spline curves with their control polygons can be found in Section 1.5.

Spline curves may be thought of as spline functions with B-spline coefficients that are vectors. This means that virtually all the algorithms that we develop for spline functions can be generalised to spline curves by simply applying the functional version of the algorithm to each component of the curve in turn.

2.4 A matrix representation of B-splines

Mathematical objects defined by recurrence relations can become very complex even if the recurrence relation is simple. This is certainly the case for B-splines. The structure of the recurrence relation (2.1) is relatively simple, but if we try to determine the symbolic expressions of the individual pieces of a B-spline in terms of the knots and the variable x , for degree five or six, the algebraic complexity of the expressions is perhaps the most striking feature.

It turns out that these rather complex formulas can be represented in terms of products of

simple matrices, and this is the theme of this section. This representation will be used in Section 3.1 to show how polynomials can be represented in terms of B-splines and to prove that B-splines are linearly independent. In Section 2.5 we will make use of the matrix notation to develop algorithms for computing function values and derivatives of splines. The matrix representation will also be useful in the theory of knot insertion in Chapter 4.

We start by introducing the matrix representation for linear and quadratic splines in two examples.

Example 2.12 (Vector representation of linear B-splines). Consider the case of linear B-splines with knots $\mathbf{t}$, and focus on one nonempty knot interval $[t_\mu, t_{\mu+1})$. We have already seen in previous sections that in this case the B-splines are quite simple. From the support properties of B-splines we know that the only linear B-splines that are nonzero on this interval are $B_{\mu-1,1}$ and $B_{\mu,1}$ and their restriction to the interval can be given in vector form as

$$(B_{\mu-1,1} \quad B_{\mu,1}) = \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_\mu} & \frac{x-t_\mu}{t_{\mu+1}-t_\mu} \end{pmatrix}. \quad (2.13)$$

Example 2.13 (Matrix representation of quadratic B-splines). The matrices appear when we come to quadratic splines. We consider the same nonempty knot interval $[t_\mu, t_{\mu+1})$; the only nonzero quadratic B-splines on this interval are $\{B_{j,2}\}_{j=\mu-2}^\mu$. By checking with Definition 2.1 we see that for x in $[t_\mu, t_{\mu+1})$, the row vector of these B-splines may be written as the product of two simple matrices,

$$\begin{aligned} (B_{\mu-2,2} \quad B_{\mu-1,2} \quad B_{\mu,2}) &= (B_{\mu-1,1} \quad B_{\mu,1}) \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+1}-t_{\mu-1}} & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_\mu} & \frac{x-t_\mu}{t_{\mu+2}-t_\mu} \end{pmatrix} \\ &= \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_\mu} & \frac{x-t_\mu}{t_{\mu+1}-t_\mu} \end{pmatrix} \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+1}-t_{\mu-1}} & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_\mu} & \frac{x-t_\mu}{t_{\mu+2}-t_\mu} \end{pmatrix}. \end{aligned} \quad (2.14)$$

If the matrices in (2.14) are multiplied together the result would of course agree with that in Exercise 1. However, the power of the matrix representation lies in the factorisation itself, as we will see in the next section. To obtain the value of the B-splines we can multiply the matrices together, but this should be done numerically, after values have been assigned to the variables. In practise this is only done implicitly, see the algorithms in Section 2.5.

The matrix notation generalises to B-splines of arbitrary degree in the obvious way.

Theorem 2.14. Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for B-splines of degree d , and let μ be an integer such that $t_\mu < t_{\mu+1}$ and $d+1 \leq \mu \leq n$. For each positive integer k with $k \leq d$ define

the matrix $\mathbf{R}_k^\mu(x) = \mathbf{R}_k(x)$ by

$$\mathbf{R}_k(x) = \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu+1-k}} & \frac{x-t_{\mu+1-k}}{t_{\mu+1}-t_{\mu+1-k}} & 0 & \cdots & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_{\mu+2-k}} & \frac{x-t_{\mu+2-k}}{t_{\mu+2}-t_{\mu+2-k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{t_{\mu+k}-x}{t_{\mu+k}-t_\mu} & \frac{x-t_\mu}{t_{\mu+k}-t_\mu} \end{pmatrix}. \quad (2.15)$$

Then, for x in the interval $[t_\mu, t_{\mu+1})$, the $d+1$ B-splines $\{B_{j,d}\}_{j=\mu-d}^\mu$ of degree d that are nonzero on this interval can be written

$$\mathbf{B}_d^T = (B_{\mu-d,d} \ B_{\mu-d+1,d} \ \cdots \ B_{\mu,d}) = \mathbf{R}_1(x)\mathbf{R}_2(x)\cdots\mathbf{R}_d(x). \quad (2.16)$$

If $f = \sum_j c_j B_{j,d}$ is a spline in $\mathbb{S}_{d,t}$, and x is restricted to the interval $[t_\mu, t_{\mu+1})$, then $f(x)$ is given by

$$f(x) = \mathbf{R}_1(x)\mathbf{R}_2(x)\cdots\mathbf{R}_d(x)\mathbf{c}_0, \quad (2.17)$$

where the vector $\mathbf{c}_0$ is given by $\mathbf{c}_0 = (c_{\mu-d}, c_{\mu-d+1}, \dots, c_\mu)^T$. The matrix $\mathbf{R}_k$ is called a B-spline matrix.

For $d=0$ the usual convention of interpreting an empty product as 1 is assumed in equations (2.16) and (2.17).

Theorem 2.14 shows how one polynomial piece of splines and B-splines are built up, by multiplying and adding together (via matrix multiplications) certain linear polynomials. This representation is only an alternative way to write the recurrence relation (2.1), but the advantage is that all the recursive steps are captured in one equation. This will be convenient for developing the theory of splines in Section 3.1.2. The factorisation (2.17) will also be helpful for designing algorithms for computing $f(x)$. This is the theme of Section 2.5.

It should be emphasised that equation (2.16) is a representation of $d+1$ polynomials, namely the $d+1$ polynomials that make up the $d+1$ B-splines on the interval $[t_\mu, t_{\mu+1})$. This equation can therefore be written

$$\begin{pmatrix} B_{\mu-d,d}^\mu(x) & B_{\mu-d+1,d}^\mu(x) & \cdots & B_{\mu,d}^\mu(x) \end{pmatrix} = \mathbf{R}_1^\mu(x)\mathbf{R}_2^\mu(x)\cdots\mathbf{R}_d^\mu(x),$$

see Lemma 2.3.

Likewise, equation (2.17) gives a representation of the polynomial f^μ that agrees with the spline f on the interval $[t_\mu, t_{\mu+1})$,

$$f^\mu(x) = \mathbf{R}_1(x)\mathbf{R}_2(x)\cdots\mathbf{R}_d(x)\mathbf{c}_0.$$

Once μ has been fixed we may let x take values outside the interval $[t_\mu, t_{\mu+1})$ in both these equations. In this way the B-spline pieces and the polynomial f^μ can be evaluated at any real number x . Figure 2.3 was produced in this way.

Example 2.15 (Matrix representation of a quadratic spline). In Example ?? we considered the spline

$$s(x) = B_{2,2}(x) + B_{3,2}(x) - B_{4,2}(x) - \sqrt{2}B_{5,2}(x)$$

on the knot vector

$$\mathbf{t} = (t_j)_{j=1}^8 = (0, 0, 0, 1, 2, 3, 3, 3).$$

Let us use the matrix representation to determine this spline explicitly on each of the subintervals $[0, 1]$, $[1, 2]$, and $[2, 3]$. If $x \in [0, 1]$ then $t_3 \leq x < t_4$ so $s(x)$ is determined by (2.17) with $\mu = 3$ and $d = 2$. To determine the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ we use the knots

$$(t_{\mu-1}, t_{\mu}, t_{\mu+1}, t_{\mu+2}) = (0, 0, 1, 2)$$

and the coefficients

$$(c_{\mu-2}, c_{\mu-1}, c_{\mu}) = (0, 1, 1).$$

Then equation (2.17) becomes

$$s(x) = \begin{pmatrix} 1-x & x \end{pmatrix} \begin{pmatrix} 1-x & x & 0 \\ 0 & (2-x)/2 & x/2 \end{pmatrix} \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = x(2-x)$$

If $x \in [1, 2]$ then $t_4 \leq x < t_5$ so $s(x)$ is determined by (2.17) with $\mu = 4$ and $d = 2$. To determine the matrices $\mathbf{R}_1$ and $\mathbf{R}_2$ in this case we use the knots

$$(t_{\mu-1}, t_{\mu}, t_{\mu+1}, t_{\mu+2}) = (0, 1, 2, 3)$$

and the coefficients

$$(c_{\mu-2}, c_{\mu-1}, c_{\mu}) = (1, 1, -1).$$

From this we find

$$s(x) = \frac{1}{2} \begin{pmatrix} 2-x & x-1 \end{pmatrix} \begin{pmatrix} 2-x & x & 0 \\ 0 & 3-x & x-1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix} = 2x - x^2.$$

For $x \in [2, 3]$ we use $\mu = 5$, and on this interval $s(x)$ is given by

$$s(x) = \begin{pmatrix} 3-x & x-2 \end{pmatrix} \begin{pmatrix} (3-x)/2 & (x-1)/2 & 0 \\ 0 & 3-x & x-2 \end{pmatrix} \begin{pmatrix} 1 \\ -1 \\ -\sqrt{2} \end{pmatrix} = (2-x)(6 - 2\sqrt{2} - (2 - \sqrt{2})x).$$

2.5 Algorithms for evaluating a spline

We originally introduced spline curves as the result of the geometric construction given in Chapter 1. In this section we will relate this algorithm to the matrix representation of B-splines and develop an alternative algorithm for computing splines.

2.5.1 High level description

Recall from Theorem 2.14 that a spline f of degree d with knots $\mathbf{t}$ and B-spline coefficients $\mathbf{c}$ can be expressed as

$$f(x) = \mathbf{R}_1(x) \cdots \mathbf{R}_d(x) \mathbf{c}_0 \quad (2.18)$$

for any x in the interval $[t_\mu, t_{\mu+1})$. Here $\mathbf{c}_0 = (c_{\mu-d}, \dots, c_\mu)$ denotes the B-spline coefficients that are active on this interval. To compute $f(x)$ from this representation we have two options: We can accumulate the matrix products from left to right or from right to left.

If we start from the right, the computations are

$$\mathbf{c}_{d-k+1} = \mathbf{R}_k \mathbf{c}_{d-k}, \quad \text{for } k = d, d-1, \dots, 1. \quad (2.19)$$

Upon completion of this we have $f(x) = \mathbf{c}_d$ (note that $\mathbf{c}_d$ is a vector of dimension 1, i.e., a scalar). We see that this algorithm amounts to post-multiplying each matrix $\mathbf{R}_k$ by a vector which in component form becomes

$$(\mathbf{R}_k(x) \mathbf{c}_{d-k})_j = \frac{t_{j+k} - x}{t_{j+k} - t_j} c_{j-1, d-k} + \frac{x - t_j}{t_{j+k} - t_j} c_{j, d-k} \quad (2.20)$$

for $j = \mu - k + 1, \dots, \mu$. This we immediately recognise as Algorithm ??.

The alternative algorithm accumulates the matrix products in (2.18) from left to right. This is equivalent to building up the nonzero B-splines at x degree by degree until we have all the nonzero B-splines of degree d , then multiplying with the corresponding B-spline coefficients and summing. Computing the B-splines is accomplished by starting with $\mathbf{B}_0(x)^T = 1$ and then performing the multiplications

$$\mathbf{B}_k(x)^T = \mathbf{B}_{k-1}(x)^T \mathbf{R}_k(x), \quad k = 1, \dots, d.$$

The vector $\mathbf{B}_p(x)$ will contain the value of the nonzero B-splines of degree d at x ,

$$\mathbf{B}_p(x) = (B_{\mu-d, d}(x), \dots, B_{\mu, d}(x))^T.$$

We can then multiply with the B-spline coefficients and add up.

Algorithm 2.16 (L). Let the polynomial degree d , the $2d$ knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$, the B-spline coefficients $\mathbf{c}_0 = (c_j)_{j=\mu-d}^\mu$ of a spline f , and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of the products

$$\mathbf{c}_{d-k+1} = \mathbf{R}_k(x) \mathbf{c}_{d-k}, \quad k = d, d-1, \dots, 1,$$

the function value $f(x)$ is given by

$$f(x) = \mathbf{c}_d.$$

Algorithm 2.17 (R). Let the polynomial degree d , the knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$ and a number x in $[t_\mu, t_{\mu+1})$ be given and set $\mathbf{B}_0 = 1$. After evaluation of the products

$$\mathbf{B}_k(x)^T = \mathbf{B}_{k-1}(x)^T \mathbf{R}_k(x), \quad k = 1, \dots, d,$$

the vector $\mathbf{B}_d(x)$ will contain the value of the $d+1$ B-splines at x ,

$$\mathbf{B}_d(x) = (B_{\mu-d, d}(x), \dots, B_{\mu, d}(x))^T.$$

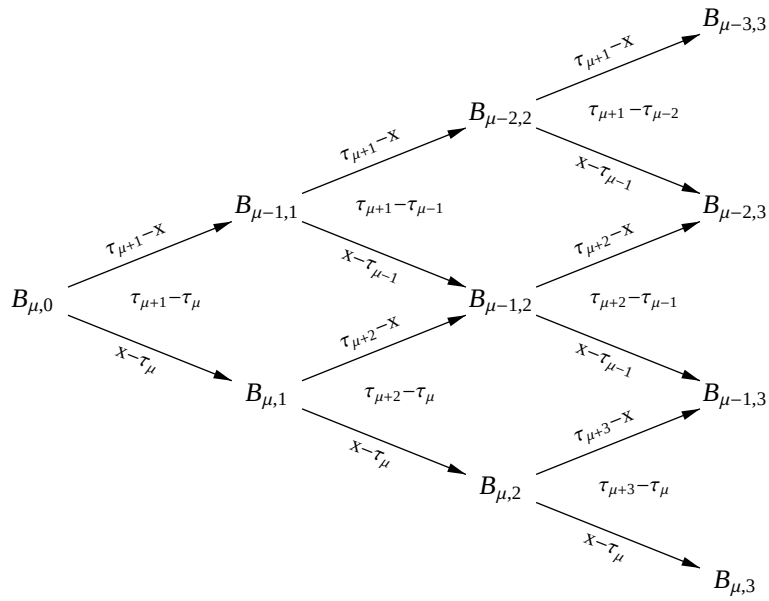


Figure 2.8. A triangular algorithm for computation of all the nonzero cubic B-splines at x .

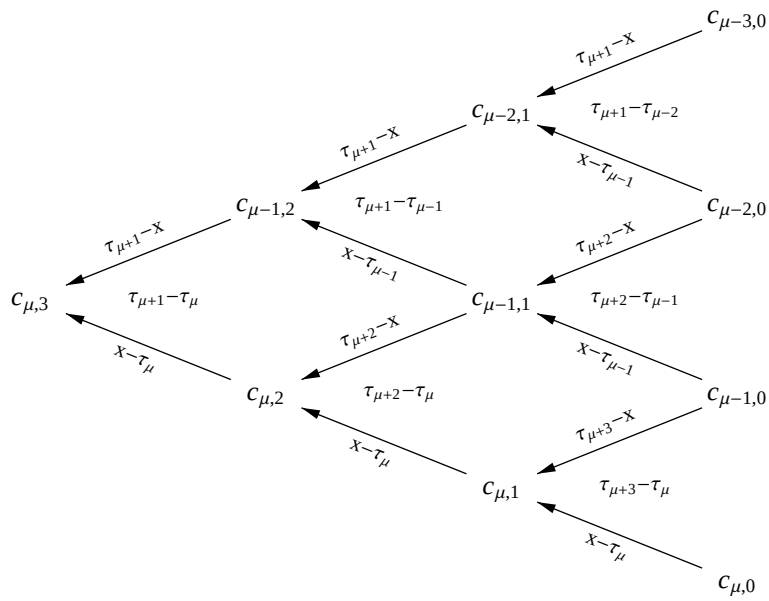


Figure 2.9. A triangular algorithm for computing the value of a cubic spline with B-spline coefficients c at $x \in [t_{\mu}, t_{\mu+1})$.

These algorithms have a simple triangular structure, just like Algorithm ??, see figures 2.8–2.9. Figure 2.8 shows the computation of all the nonzero B-splines at a point x , while Figure 2.9 shows how the value of a cubic spline can be computed.

In Algorithms 2.16 and 2.17 it is assumed that there are $2d$ knots to the left and right of x . This may not always be the case, especially near the ends of the knot vector, unless it is $d + 1$ -regular. Exercise 26 discusses evaluation in such a case.

2.6 Exercises

2.1 Show that

- a) $B[0, 0, 0, 1](x) = (1 - x)B[0, 0, 1](x) = (1 - x)^2 B[0, 1](x).$
- b) $B[0, 0, 1, 2](x) = x(2 - \frac{3}{2}x)B[0, 1](x) + \frac{1}{2}(2 - x)^2 B[1, 2](x).$
- c) $B[0, 1, 2, 3](x) = \frac{x^2}{2}B[0, 1](x) + \left(\frac{3}{4} - (x - \frac{3}{2})^2\right)B[1, 2](x) + \frac{(3-x)^2}{2}B[2, 3](x).$
- d) $B[0, 1, 1, 2](x) = x^2 B[0, 1](x) + (2 - x)^2 B[1, 2](x).$
- e) $B[0, 0, 1, 1](x) = 2x(1 - x)B[0, 1](x).$
- f) $B[0, 1, 2, 2](x) = \frac{1}{2}x^2 B[0, 1](x) + (2 - x)(\frac{3}{2}x - 1)B[1, 2](x).$
- g) $B[0, 1, 1, 1](x) = x^2 B[0, 1](x).$
- h) $B[0, 3, 4, 6](x) = \frac{1}{12}x^2 B[0, 3](x) + \frac{1}{12}(-7x^2 + 48x - 72)B[3, 4](x) + \frac{1}{6}(6 - x)^2 B[4, 6](x).$

2.2 Find the individual polynomial pieces of the following cubic B-splines and discuss smoothness properties at knots

- a) $B[0, 0, 0, 0, 1](x)$ and $B[0, 1, 1, 1, 1](x)$
- b) $B[0, 1, 1, 1, 2](x)$

2.3 Show the following explicit expressions for the first few uniform B-splines

$$\begin{aligned}
 M_1(x) &= xM_0(x) + (2 - x)M_0(x - 1) \\
 M_2(x) &= \frac{x^2}{2}M_0(x) + \left(\frac{3}{4} - (x - \frac{3}{2})^2\right)M_0(x - 1) + \frac{(3 - x)^2}{2}M_0(x - 2) \\
 M_3(x) &= \frac{x^3}{6}M_0(x) + \left(\frac{2}{3} - \frac{1}{2}x(x - 2)^2\right)M_0(x - 1) \\
 &\quad + \left(\frac{2}{3} - \frac{1}{2}(4 - x)(x - 2)^2\right)M_0(x - 2) + \frac{(4 - x)^3}{6}M_0(x - 3)
 \end{aligned} \tag{2.21}$$

2.4 Show the following explicit expression for a generic quadratic B-spline

$$\begin{aligned}
 B_{j,2}(x) &= \frac{x-t_j}{t_{j+2}-t_j} \left[\frac{x-t_j}{t_{j+1}-t_j} B_{j,0}(x) + \frac{t_{j+2}-x}{t_{j+2}-t_{j+1}} B_{j+1,0}(x) \right] \\
 &\quad + \frac{t_{j+3}-x}{t_{j+3}-t_{j+1}} \left[\frac{x-t_{j+1}}{t_{j+2}-t_{j+1}} B_{j+1,0}(x) + \frac{t_{j+3}-x}{t_{j+3}-t_{j+2}} B_{j+2,0}(x) \right] \\
 &= \frac{(x-t_j)^2}{(t_{j+2}-t_j)(t_{j+1}-t_j)} B_{j,0}(x) + \frac{(t_{j+3}-x)^2}{(t_{j+3}-t_{j+1})(t_{j+3}-t_{j+2})} B_{j+2,0}(x) \\
 &\quad + \left(\frac{(x-t_j)(t_{j+2}-x)}{(t_{j+2}-t_j)(t_{j+2}-t_{j+1})} + \frac{(t_{j+3}-x)(x-t_{j+1})}{(t_{j+3}-t_{j+1})(t_{j+2}-t_{j+1})} \right) B_{j+1,0}(x).
 \end{aligned} \tag{2.22}$$

The complexity of this expression gives us a good reason to work with B-splines through other means than explicit formulas.

2.5 Show that the B-spline $B_{j,d}$ satisfies properties 1–3 of Lemma 2.3.

2.6 Show that $B_{j,d}$ is a piecewise polynomial by establishing equation 2.6. Use induction on the degree d .

2.7 In this exercise we are going to establish some properties of the Bernstein polynomials of degree d . Recall that

$$b_{j,d}(x) := \binom{d}{j} x^j (1-x)^{d-j}, \quad j = 0, \dots, d$$

and that $b_{j,d}(x) := 0$ for $j < 0$ and $j > d$. Prove the following for $j = 0, 1, \dots, d$

a) The recurrence relation

$$b_{j,d}(x) = x b_{j-1,d-1}(x) + (1-x) b_{j,d-1}(x).$$

b) The differentiation formula

$$D b_{j,d}(x) = d(b_{j-1,d-1}(x) - b_{j,d-1}(x)).$$

c) The Bernstein basis function $b_{j,d}(x)$ has a maximum at $x = j/d$, and this is the only maximum.

d)

$$\int_0^1 b_{j,d}(x) dx = 1/(d+1).$$

2.8 a) When a B-spline is evaluated at one of its knots it can be simplified according to the formula

$$B(t_i | t_j, \dots, t_{j+1+d}) = B(t_i | t_j, \dots, t_{i-1}, t_{i+1}, \dots, t_{j+1+d}) \tag{2.23}$$

which is valid for $j = j, j+1, \dots, j+1+d$. Prove this by induction on the degree d .

- b) Use the formula in (2.23) to compute the following values of a quadratic B-spline at the interior knots:

$$B_{j,2}(t_{j+1}) = \frac{t_{j+1} - t_j}{t_{j+2} - t_j}, \quad B_{j,2}(t_{j+2}) = \frac{t_{j+3} - t_{j+2}}{t_{j+3} - t_{j+1}}. \quad (2.24)$$

- c) Prove property (5) of Lemma 2.3.

2.9 Prove the following formula using (2.4) and (??)

$$B[a, \overbrace{b, \dots, b}^p, c](x) = \frac{(x-a)^p}{(b-a)^p} B[a, b](x) + \frac{(c-x)^p}{(c-b)^p} B[b, c](x).$$

Show that this function is continuous at all real numbers.

2.10 Prove the following formulas by induction on d .

$$\begin{aligned} B[\overbrace{a, \dots, a}^d, b, c](x) &= \frac{x-a}{b-a} \sum_{j=0}^{d-1} \frac{(c-x)^j (b-x)^{d-1-j}}{(c-a)^j (b-a)^{d-1-j}} B[a, b](x) \\ &\quad + \frac{(c-x)^d}{(c-a)^{d-1} (c-b)} B[b, c](x), \\ B[a, b, \overbrace{c, \dots, c}^d](x) &= \frac{(x-a)^d}{(c-a)^{d-1} (b-a)} B[a, b](x) \\ &\quad + \frac{c-x}{c-b} \sum_{j=0}^{d-1} \frac{(x-a)^j (x-b)^{d-1-j}}{(c-a)^j (c-b)^{d-1-j}} B[b, c](x). \end{aligned}$$

2.11 When the knots are simple we can give explicit formulas for the B-splines.

- a) Show by induction that if $t_j < \dots < t_{j+1+d}$ then

$$B_{j,d}(x) = (t_{j+1+d} - t_j) \sum_{i=j}^{j+1+d} \frac{(x-t_i)_+^d}{\prod_{\substack{k=j \\ k \neq i}}^{j+1+d} (t_k - t_i)}$$

where

$$(x-t_i)_+^d = \begin{cases} (x-t_i)^d, & \text{if } x \geq t_i; \\ 0, & \text{otherwise.} \end{cases}$$

- b) Show that $B_{j,d}$ can also be written

$$B_{j,d}(x) = (t_{j+1+d} - t_j) \sum_{i=j}^{j+1+d} \frac{(t_i - x)_+^d}{\prod_{\substack{k=j \\ k \neq i}}^{j+1+d} (t_i - t_k)}$$

but now the $(\cdot)_+$ -function must be defined by

$$(t_i - x)_+^d = \begin{cases} (t_i - x)^d, & \text{if } t_i > x; \\ 0, & \text{otherwise.} \end{cases}$$

- 2.12** Let $(x_i, y_i)_{i=1}^m$ be a set of data points with $x_i < x_{i+1}$ for $j = 1, 2, \dots, m-1$. On the knot vector

$$\mathbf{t} = (t_j)_{j=1}^{m+2} = (x_1, x_1, x_2, x_3, \dots, x_{m-1}, x_m, x_m)$$

we consider the linear ($d = 1$) spline function $s(x) = \sum_{j=1}^m y_j B_{j,1}(x)$, for $x \in [x_1, x_m]$.

- Show that $s(x_i) = y_i$ for $i = 1, \dots, m$.
- Show the explicit expression $s(x) = \frac{x_{i+1}-x}{x_{i+1}-x_i} y_i + \frac{x-x_i}{x_{i+1}-x_i} y_{i+1}$, whenever $x \in [x_i, x_{i+1}]$ for $i = 1, \dots, m-1$.

Conclude that s is the piecewise linear interpolant to the data $(x_i, y_i)_{i=1}^m$.

- 2.13** Let $f : [a, b] \rightarrow \mathbb{R}$ be a given function defined on some interval $[a, b]$, and let n be an integer greater than 2. On $[a, b]$ we assume that we have a knot vector $\mathbf{t} = (t_j)_{j=1}^{n+3}$, where

$$a = t_1 = t_2 = t_3 < t_4 < \dots < t_n < t_{n+1} = t_{n+2} = t_{n+3} = b.$$

We can then define the quadratic spline function

$$s(x) = Qf(x) = \sum_{j=1}^n f(t_j^*) B_{j,2}(x),$$

where

$$t_j^* = (t_{j+1} + t_{j+2})/2, \quad j = 1, \dots, n.$$

The function Qf is called the *Variation Diminishing Spline Approximation* to f of degree 2.

- Show that

$$a = t_1^* < t_2^* < \dots < t_n^* = b.$$

- As a particular instance of this approximation we consider the function $f(x) = \sqrt{2} \sin(\frac{\pi}{2}x)$ on the interval $[0, 3]$ and the knot vector $\mathbf{t} = (t_j)_{j=1}^8 = (0, 0, 0, 1, 2, 3, 3, 3)$.
- Show that $(t_j^*)_{j=1}^5 = (0, 1/2, 3/2, 5/2, 3)$ and $s(x) = B_{2,2}(x) + B_{3,2}(x) - B_{4,2}(x) - \sqrt{2}B_{5,2}(x)$. A plot of this function together with $f(x)$ is shown in Figure 2.5 (b).

- 2.14** Write down the matrix $\mathbf{R}_3(x)$ for $\mu = 4$ in the case of uniform splines ($t_j = j$ for all j). Do the same for the Bernstein basis ($\mathbf{t} = (0, 0, 0, 0, 1, 1, 1, 1](x)$).

- 2.15** On the knot vector

$$\mathbf{t} = (t_j)_{j=1}^8 = (0, 0, 0, 0, 1, 1, 1, 1)$$

we consider the cubic spline function

$$s(x) = -B_{1,3}(x) + 5B_{2,3}(x) - 5B_{3,3}(x) + B_{4,3}(x).$$

In terms of the cubic Bernstein basis we have

$$s(x) = -b_{0,3}(x) + 5b_{1,3}(x) - 5b_{2,3}(x) + b_{3,3}(x), \quad 0 \leq x \leq 1.$$

This polynomial is shown in Figure 2.6 (b). Show that It is the cubic *Chebyshev polynomial* $T_3(2x - 1)$ with respect to the interval $[0, 1]$.

2.16 Matrix representation of cubic B-splines In the cubic case the only nonzero B-splines on $[t_\mu, t_{\mu+1})$ are $\{B_{j,3}\}_{j=\mu-3}^\mu$. Check that for x in this interval these B-splines may be written

$$\begin{aligned}
 (B_{\mu-3,3} \quad B_{\mu-2,3} \quad B_{\mu-1,3} \quad B_{\mu,3}) &= (B_{\mu-2,2} \quad B_{\mu-1,2} \quad B_{\mu,2}) \\
 &\quad \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-2}} & \frac{x-t_{\mu-2}}{t_{\mu+1}-t_{\mu-2}} & 0 & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+2}-t_{\mu-1}} & 0 \\ 0 & 0 & \frac{t_{\mu+3}-x}{t_{\mu+3}-t_\mu} & \frac{x-t_\mu}{t_{\mu+3}-t_\mu} \end{pmatrix} \\
 &= \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_\mu} & \frac{x-t_\mu}{t_{\mu+1}-t_\mu} \end{pmatrix} \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+1}-t_{\mu-1}} & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_\mu} & \frac{x-t_\mu}{t_{\mu+2}-t_\mu} \end{pmatrix} \\
 &\quad \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-2}} & \frac{x-t_{\mu-2}}{t_{\mu+1}-t_{\mu-2}} & 0 & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+2}-t_{\mu-1}} & 0 \\ 0 & 0 & \frac{t_{\mu+3}-x}{t_{\mu+3}-t_\mu} & \frac{x-t_\mu}{t_{\mu+3}-t_\mu} \end{pmatrix}.
 \end{aligned}$$

2.17 Given a knot vector $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ and a real number x with $x \in [t_1, t_{n+d+1})$, write a procedure for determining the index μ such that $t_\mu \leq x < t_{\mu+1}$. A call to this routine is always needed before Algorithms 2.16 and 2.17 are run. By letting μ be an input parameter as well as an output parameter you can minimise the searching for example during plotting when the routine is called with many values of x in the same knot interval.

2.18 Implement Algorithm 2.17 in your favourite programming language.

2.19 Implement Algorithm 2.16 in your favourite programming language.

2.20 Count the number of operations (additions, multiplications, divisions) involved in Algorithm 2.16.

2.21 Count the number of operations (additions, multiplications, divisions) involved in Algorithm 2.17.

2.22 Write a program that plots the cubic B-spline $B[0, 1, 3, 5, 6]$ and its polynomial pieces. Present the results as in Figure 2.3.

2.23 a) What is computed by Algorithm 2.16 if x does not belong to the interval $[t_\mu, t_{\mu+1})$?

b) Repeat (a) for Algorithm 2.17.

2.24 Algorithms 2.16 and 2.17 are high level algorithms. Although they may be implemented directly by forming the matrices $\{\mathbf{R}_k\}_{k=1}^d$, it can be better to polish the algorithms a bit more. In this exercise we will discuss Algorithm 2.17 in more detail. For more details on Algorithm 2.16, we refer to Algorithm ?? in Chapter 1 and exercise 25 below.

We are going to give two more detailed versions of Algorithm 2.17. In the first one, we make use of vector operations. This version would be suitable for a language like Matlab or Mathematica where for-loops are relatively slow, but the built-in vector operations are fast.

We assume that the elementary arithmetic operations may be applied to vectors of the same size. For example, the vector operation $\mathbf{a}/\mathbf{b}$ would produce a vector of the same length as $\mathbf{a}$ and $\mathbf{b}$, with entry i equal to a_i/b_i . We can also combine a scalar and a vector as in $x + \mathbf{a}$; then the first operand is converted to a vector of the same length as $\mathbf{a}$ by duplicating x the correct number of times.

We will need two more vector operations which we denote $\mathbf{a}_{+l}$ and $\mathbf{a}_{+f}$. The first denotes the vector obtained by appending a zero to the end of $\mathbf{a}$, while $\mathbf{a}_{+f}$ denotes the result of prepending a zero element at the beginning of $\mathbf{a}$. In Matlab syntax this would be written as $\mathbf{a}_{+l} = [\mathbf{a}, 0]$ and $\mathbf{a}_{+f} = [0, \mathbf{a}]$.

a) Algorithm 2.17 corresponds to pre-multiplying each matrix $\mathbf{R}_k$ by a row vector. In component form show that this can be written

$$(\mathbf{B}_{k-1}(x))^T \mathbf{R}_k(x)_j = \frac{x - t_j}{t_{j+k} - t_j} B_{j,k-1}(x) + \frac{t_{j+1+k} - x}{t_{j+1+k} - t_{j+1}} B_{j+1,k-1}(x) \quad (2.25)$$

for $j = \mu - k, \dots, \mu$, where $B_{\mu-k,k-1}(x) = B_{\mu+1,k-1}(x) = 0$ when $x \in [t_\mu, t_{\mu+1})$.

b) Verify that Algorithm 2.17 can be written in the following more explicit form. A vector version of Algorithm 2.16 can be found in exercise 25.

Algorithm 2.18 (R—vector version). *Let the polynomial degree d , the knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$ and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of*

1. $\mathbf{b} = \mathbf{1}$;
2. For $k = 1, 2, \dots, d$
 1. $\mathbf{t1} = (t_{\mu-k+1}, \dots, t_\mu)$;
 2. $\mathbf{t2} = (t_{\mu+1}, \dots, t_{\mu+k})$;
 3. $\omega = (x - \mathbf{t1}) / (\mathbf{t2} - \mathbf{t1})$;
 4. $\mathbf{b} = ((1 - \omega) * \mathbf{b})_{+l} + (\omega * \mathbf{b})_{+f}$;

the vector $\mathbf{b}$ will contain the value of the $d + 1$ B-splines at x ,

$$\mathbf{b} = (B_{\mu-d,d}(x), \dots, B_{\mu,d}(x))^T.$$

c) When programming in a traditional procedural programming language, the vector operations will usually have to be replaced by for-loops. This can be accomplished as follows.

Algorithm 2.19 (R—scalar version). *Let the polynomial degree d , the knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$ and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of*

1. $b_{d+1} = 1; b_j = 0, j = 1, \dots, d;$
2. *For* $r = 1, 2, \dots, d$
 1. $k = \mu - r + 1;$
 2. $\omega_2 = (t_{k+r} - x) / (t_{k+r} - t_k);$
 3. $b_{d-r} = \omega_2 b_{d-r+1};$
4. *For* $j = d - r + 1, d - r + 2, \dots, d - 1$
 1. $k = k + 1;$
 2. $\omega_1 = \omega_2;$
 3. $\omega_2 = (t_{k+r} - x) / (t_{k+r} - t_k);$
 4. $b_i = (1 - \omega_1) b_i + \omega_2 b_{i+1};$
5. $b_d = (1 - \omega_2) b_d$

the vector $\mathbf{b}$ will contain the value of the $d + 1$ B-splines at x ,

$$\mathbf{b} = (B_{\mu-d,d}(x), \dots, B_{\mu,d}(x))^T.$$

2.25 In Exercise 24 we gave a vector version of Algorithm 2.17 for computing the nonzero B-splines at a point x . Below is a similar vector version of Algorithm 2.16 for computing the value of a spline at x . Verify that the algorithm is correct and compare it with Algorithm 2.18.

Algorithm 2.20. *Let $f = \sum_i c_j B_{i,d,t}$ be a spline in $\mathbb{S}_{d,t}$, and let x be a real number in the interval $[t_\mu, t_{\mu+1})$. Then $f(x)$ can be computed as follows:*

1. $\mathbf{c} = (c_{\mu-d}, \dots, c_\mu);$
2. *For* $k = d, d - 1, \dots, 1$
 1. $\mathbf{t1} = (t_{\mu-k+1}, \dots, t_\mu);$
 2. $\mathbf{t2} = (t_{\mu+1}, \dots, t_{\mu+k});$
 3. $\omega = (x - \mathbf{t1}) / (\mathbf{t2} - \mathbf{t1});$
 4. $\mathbf{c} = (1 - \omega) * \mathbf{c}_{-l} + \omega * \mathbf{c}_{-f};$

After these statements $\mathbf{c}$ will be a vector of length 1 that contains the number $f(x)$. Here the notation $\mathbf{c}_{-l}$ and $\mathbf{c}_{-f}$ denote the vectors obtained by dropping the last, respectively the first, entry from $\mathbf{c}$.

2.26 Suppose that $d = 3$ and that the knot vector is given by

$$\hat{\mathbf{t}} = (t_j)_{j=1}^5 = (0, 1, 2, 3, 4).$$

With this knot vector we can only associate one cubic B-spline, namely $B_{1,3}$. Therefore, if we are to compute $B_{1,3}(x)$ for some x in $(0, 4)$, none of the algorithms of this section apply. Define the augmented knot vector $\mathbf{t}$ by

$$\mathbf{t} = (-1, -1, -1, -1, 0, 1, 2, 3, 4, 5, 5, 5, 5).$$

Explain how this knot vector can be exploited to compute the B-spline $B_{1,3}(x)$ by Algorithms 2.16 or 2.17.

Further properties of splines

In Chapter 2 we established some of the most elementary properties of B-splines. In this chapter, our focus is on the question “What kind of functions can be represented as linear combinations of B-splines?” The answer is that our spline space contains a large class of piecewise polynomials, and this ensures that splines are reasonably flexible, much more so than polynomials. To prove this, we start by showing that polynomials of degree d can be represented in terms of splines of degree d in Section 3.1. This is proved by making use of some simple properties of the B-spline matrices. As a bonus, we also prove that B-splines are linearly independent and therefore provide a basis for spline spaces, a result that is crucial for practical computations. In Section 3.2, we investigate the smoothness of splines and B-splines in detail, and this allows us to conclude in Section 3.3 that spline spaces contain a large class of piecewise polynomials.

3.1 Linear independence of B-splines and representation of polynomials

Our aim in this section is to show that any polynomial can be represented as a linear combination of B-splines, and also that B-splines are linearly independent. To do this, we first need some simple properties of the B-spline matrices defined in Theorem 2.14.

3.1.1 Some properties of the B-spline matrices

To study the B-spline matrices, we associate a certain polynomial with each B-spline.

Definition 3.1. To each B-spline $B_{j,d,t}$ we associate a polynomial $\rho_{j,d,t}$ of degree d . It is given by $\rho_{j,0,t}(y) = \rho_{j,0}(y) := 1$ if $d = 0$ and for $d \geq 1$

$$\rho_{j,d,t}(y) = \rho_{j,d}(y) := (y - t_{j+1})(y - t_{j+2}) \cdots (y - t_{j+d}). \quad (3.1)$$

This polynomial is called the dual polynomial of the B-spline $B_{j,d,t}$.

On the interval $[t_\mu, t_{\mu+1})$, we have $d + 1$ nonzero B-splines that we collect in the vector $\mathbf{B}_d(x) := (B_{\mu-d,d}(x), \dots, B_{\mu,d}(x))^T$. We collect the corresponding dual polynomials in the vector

$$\boldsymbol{\rho}_d(y) := (\rho_{\mu-d,d}(y), \dots, \rho_{\mu,d}(y))^T. \quad (3.2)$$

The following lemma shows the effect of applying the matrix $\mathbf{R}_d$ to $\boldsymbol{\rho}_d$.

Lemma 3.2. *Let μ be an integer such that $t_\mu < t_{\mu+1}$ and let $\boldsymbol{\rho}_d(y)$ be the dual polynomials defined by (3.2). For $d \geq 1$ the relation*

$$\mathbf{R}_d(x)\boldsymbol{\rho}_d(y) = (y - x)\boldsymbol{\rho}_{d-1}(y). \quad (3.3)$$

holds for all $x, y \in \mathbb{R}$.

Proof. Writing out (3.3) in component form, we see that what we need to prove is

$$\frac{(x - t_j)\rho_{j,d}(y) + (t_{j+d} - x)\rho_{j-1,d}(y)}{t_{j+d} - t_j} = (y - x)\rho_{j,d-1}(y), \quad (3.4)$$

for $j = \mu - d + 1, \dots, \mu$. Since $\rho_{j,d}(y) = (y - t_{j+d})\rho_{j,d-1}(y)$ and $\rho_{j-1,d}(y) = (y - t_j)\rho_{j,d-1}(y)$, the numerator on the left-hand side of (3.4) can be written

$$((x - t_j)(y - t_{j+d}) + (t_{j+d} - x)(y - t_j))\rho_{j,d-1}(y).$$

A simple calculation reveals that

$$(x - t_j)(y - t_{j+d}) + (t_{j+d} - x)(y - t_j) = (y - x)(t_{j+d} - t_j). \quad (3.5)$$

Inserting this on the left in (3.4) and simplifying, we obtain the right-hand side. ■

The crucial relation (3.5) is an example of linear interpolation. For if we define the linear function g by $g(x) = y - x$ for a fixed number y , then linear interpolation at t_j and t_{j+d} gives the relation

$$\frac{t_{j+d} - x}{t_{j+d} - t_j} g(t_j) + \frac{x - t_j}{t_{j+d} - t_j} g(t_{j+d}) = g(x),$$

see Section 1.3 in Chapter 1. If we multiply both sides of this equation by $t_{j+d} - t_j$, we obtain equation (3.5).

In equation 3.3, the $d + 1$ -vector $\boldsymbol{\rho}_d$ is transformed to a vector with d components. We can reduce the number of components further by applying more $\mathbf{R}$'s. By making use of all the matrices $\mathbf{R}_1, \dots, \mathbf{R}_d$ we end up with a scalar.

Corollary 3.3. *Let μ be an integer such that $t_\mu < t_{\mu+1}$ and let $\boldsymbol{\rho}_d(y)$ be the dual polynomials defined by (3.2). Then the relation*

$$\mathbf{R}_1(x_1)\mathbf{R}_2(x_2)\cdots\mathbf{R}_d(x_d)\boldsymbol{\rho}_d(y) = (y - x_1)(y - x_2)\cdots(y - x_d). \quad (3.6)$$

holds for all real numbers $x_1, x_2, \dots, x_d$ and y .

We need one more property of the B-spline matrices. This property will not be established completely until we have proved that the dual polynomials are linearly independent.

Lemma 3.4. *For $d \geq 2$ and for any x and z in $\mathbb{R}$, the matrices $\mathbf{R}_{d-1}$ and $\mathbf{R}_d$ satisfy the relation*

$$\mathbf{R}_{d-1}(z)\mathbf{R}_d(x) = \mathbf{R}_{d-1}(x)\mathbf{R}_d(z). \quad (3.7)$$

Proof. Applying (3.3) twice, we obtain

$$\mathbf{R}_{d-1}(x)\mathbf{R}_d(z)\boldsymbol{\rho}_d(y) = (y-x)(y-z)\boldsymbol{\rho}_{d-2}(y).$$

By symmetry we also have

$$\mathbf{R}_{d-1}(z)\mathbf{R}_d(x)\boldsymbol{\rho}_d(y) = (y-x)(y-z)\boldsymbol{\rho}_{d-2}(y),$$

Equivalently,

$$\mathbf{B}\boldsymbol{\rho}_d(y) = \mathbf{0} \quad (3.8)$$

for all y , where the $(d-1) \times (d+1)$ matrix $\mathbf{B}$ is defined by

$$\mathbf{B} = \mathbf{R}_{d-1}(x)\mathbf{R}_d(z) - \mathbf{R}_{d-1}(z)\mathbf{R}_d(x).$$

To complete the proof, we must show that $\mathbf{B} = \mathbf{0}$. Let $\mathbf{a}$ be any vector in $\mathbb{R}^{d-1}$. Then we know from (3.8) that $\mathbf{a}^T \mathbf{B}\boldsymbol{\rho}_d(y) = 0$ for all y . Since the $d+1$ polynomials in $\boldsymbol{\rho}_d$ are linearly independent, see Lemma 3.8, this means that $\mathbf{a}^T \mathbf{B} = \mathbf{0}$. But $\mathbf{a}$ was arbitrary, so $\mathbf{B}$ maps all vectors to $\mathbf{0}$, in other words $\mathbf{B} = \mathbf{0}$. ■

3.1.2 Marsden's identity and representation of polynomials

The relation (3.6) is a key to finding the B-spline representation of polynomials. If we set $x_1 = \cdots = x_d = x$ and remember that $R_1(x) \cdots R_d(x) = \mathbf{B}_d(x)$, the relation becomes

$$(y-x)^d = \mathbf{B}_d(x)^T \boldsymbol{\rho}_d(y) = \sum_{j=\mu-d}^{\mu} B_{j,d}(x) \rho_{j,d}(y), \quad (3.9)$$

provided x is in the interval $[t_\mu, t_{\mu+1})$. The interpretation of this is that if for fixed y , we use the sequence of numbers $(\rho_{j,d}(y))_{j=\mu-d}^{\mu}$ as B-spline coefficients, the resulting spline is the polynomial $(y-x)^d$, as long as we restrict our attention to the interval $[t_\mu, t_{\mu+1})$. But since the coefficients $(\rho_{j,d}(y))_{j=\mu-d}^{\mu}$ are independent of μ and therefore of the knot interval, the polynomial formula (3.9) can be generalised to a statement about how the polynomial $(y-x)^d$ is represented in terms of B-splines.

Theorem 3.5 (Marsden's identity). *Let the knot vector $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be given. Then the relation*

$$(y-x)^d = \sum_{j=1}^n \rho_{j,d}(y) B_{j,d}(x) \quad (3.10)$$

holds for all real numbers y , and all real numbers x in the interval $[t_{d+1}, t_{n+1})$.

The restriction on x cannot be avoided since we do not have a complete set of B-splines outside the interval $[t_{d+1}, t_{n+1})$. The relation (3.9) is therefore not valid if x is outside this interval.

The power of Theorem 3.5 lies in the fact that the coefficients ρ_d depend on y . Making use of this result, we can show explicitly how the powers $1, x, \dots, x^d$ can be written in terms of B-splines.

Corollary 3.6. *On the interval $[t_{d+1}, t_{n+1})$, the power basis $\{x^i\}_{i=0}^d$ can be expressed in terms of B-splines through the relations*

$$1 = \sum_{j=1}^n B_{j,d}(x), \quad \text{for } d \geq 0, \quad (3.11)$$

$$x = \sum_{j=1}^n t_{j,d}^* B_{j,d}(x), \quad \text{for } d \geq 1, \quad (3.12)$$

$$x^2 = \sum_{j=1}^n t_{j,d}^{**} B_{j,d}(x), \quad \text{for } d \geq 2, \quad (3.13)$$

where

$$t_{j,d}^* = (t_{j+1} + \dots + t_{j+d})/d \quad (3.14)$$

$$t_{j,d}^{**} = \sum_{i=j+1}^{j+d-1} \sum_{k=i+1}^{j+d} t_i t_k / \binom{d}{2}. \quad (3.15)$$

In general, for $r = 0, 1, \dots, d$, the relation

$$x^r = \sum_{j=1}^n \sigma_{j,d}^r B_{j,d}(x) \quad (3.16)$$

holds for any x in the interval $[t_{d+1}, t_{n+1})$. Here $\sigma_{j,d}^r$ are the symmetric polynomials given by

$$\sigma_{j,d}^r = \left(\sum t_{j_1} t_{j_2} \dots t_{j_r} \right) / \binom{d}{r}, \quad \text{for } r = 0, 1, \dots, d, \quad (3.17)$$

where the sum is over all integers $j_1, \dots, j_r$ with $j+1 \leq j_1 < \dots < j_r \leq j+d$, a total of $\binom{d}{r}$ terms.

Proof. If we differentiate both sides of equation (3.10) a total of $d-r$ times with respect to y , set $y = 0$, and rearrange constants, we end up with

$$x^r = (-1)^r \frac{r!}{d!} \mathbf{B}_d(x)^T D^{d-r} \boldsymbol{\rho}_d(0) = (-1)^r \frac{r!}{d!} \sum_j B_{j,d}(x) D^{d-r} \rho_{j,d}(0). \quad (3.18)$$

Multiplying together the factors of $\rho_{j,d}$, we find

$$\rho_{j,d}(y) = y^d - dt_{j,d}^* y^{d-1} + \binom{d}{2} t_{j,d}^{**} y^{d-2} + \text{lower order terms.} \quad (3.19)$$

From this it follows that

$$D^d \rho_{j,d}(0) = p!, \quad D^{d-1} \rho_{j,d}(0) = -(d-1)! p t_{j,d}^*, \quad D^{d-2} \rho_{j,d}(0) = (d-2)! \binom{d}{2} t_{j,d}^{**}. \quad (3.20)$$

Setting $r = 0, 1$ and 2 in (3.18) and inserting the appropriate formula in (3.20), leads to equations (3.11), (3.12), and (3.13). In general, we have the formula

$$\rho_{j,d}(y) = \sum_{r=0}^d (-1)^r \binom{d}{r} \sigma_{j,d}^r y^{d-r}.$$

Using the same reasoning as above, we therefore find that

$$(-1)^r \frac{r!}{d!} D^{d-r} \rho_{j,d}(0) = \frac{r!(d-r)!}{d!} \binom{d}{r} \sigma_{j,d}^r = \sigma_{j,d}^r,$$

so (3.16) follows from (3.18). ■

The coefficients $\sigma_{j,d}^r$ are scaled versions of the *elementary symmetric polynomials* of degree d . They play an important role in the study of polynomial rings.

Example 3.7. In the cubic case, the relations (3.11)–(3.13) are

$$1 = \sum_{j=1}^n B_{j,3}(x), \quad (3.21)$$

$$x = \sum_{j=1}^n \frac{t_{j+1} + t_{j+2} + t_{j+3}}{3} B_{j,3}(x), \quad (3.22)$$

$$x^2 = \sum_{j=1}^n \frac{t_{j+1} t_{j+2} + t_{j+1} t_{j+3} + t_{j+2} t_{j+3}}{3} B_{j,3}(x), \quad (3.23)$$

$$x^3 = \sum_{j=1}^n t_{j+1} t_{j+2} t_{j+3} B_{j,3}(x), \quad (3.24)$$

which are valid for all x in $[t_{d+1}, t_{n+1})$.

3.1.3 Linear independence of B-splines

Recall from Section A.3.1 in the Appendix that a set of functions $\{\phi_j\}_{j=1}^n$ are linearly independent on an interval I if $\sum_{j=1}^n c_j \phi_j(x) = 0$ for all $x \in I$ implies that $c_j = 0$ for all j . In other words, the only way to represent the 0-function on I is by letting all the coefficients be zero. A consequence of this is that any function that can be represented by $\{\phi_j\}_{j=1}^n$ has a unique representation.

To prove that B-splines are linearly independent, we start by showing that the B-splines that are nonzero on a single knot interval are linearly independent.

Lemma 3.8. *The B-splines $\{B_{j,d}\}_{j=\mu-d}^{\mu}$ and the dual polynomials $\{\rho_{j,d}\}_{j=\mu-d}^{\mu}$ are both linearly independent on the interval $[t_{\mu}, t_{\mu+1})$.*

Proof. From Corollary 3.6, we know that the power basis $1, x, \dots, x^d$, and therefore any polynomial of degree d , can be represented by linear combinations of B-splines. Recall that the linear space of polynomials of degree d has dimension $d + 1$. On the interval $[t_{\mu}, t_{\mu+1})$, the only nonzero B-splines are $\{B_{j,d}\}_{j=\mu-d}^{\mu}$. Since all polynomials of degree d can be written in terms of these $d + 1$ B-splines, they must be linearly independent. These B-splines therefore form a basis for polynomials of degree d on $[t_{\mu}, t_{\mu+1})$. The symmetry of x and y in (3.9) leads to the same conclusion for the dual polynomials. ■

From this local result, we are going to obtain a global linear independence result for B-splines. But first we need to be more precise about the type of knot vectors we consider.

Definition 3.9. *A knot vector $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ is said to be $d + 1$ -extended if*

1. $n \geq d + 1$,
2. $t_{d+1} < t_{d+2}$ and $t_n < t_{n+1}$,
3. $t_j < t_{j+d+1}$ for $j = 1, 2, \dots, n$.

A $d + 1$ -extended knot vector for which $t_1 = t_{d+1}$ and $t_{n+1} = t_{n+d+1}$ is said to be $d + 1$ -regular or $d + 1$ -open.

The norm is to use $d + 1$ -regular knot vectors, but linear independence can be proved in the more general situation of a $d + 1$ -extended knot vector.

Theorem 3.10. *Suppose that $\mathbf{t}$ is a $d + 1$ -extended knot vector. Then the B-splines in $\mathbb{S}_{d,\mathbf{t}}$ are linearly independent on the interval $[t_{d+1}, t_{n+1})$.*

Proof. Suppose that the spline $f = \sum_{j=1}^n c_j B_{j,d}$ is identically zero on $[t_{d+1}, t_{n+1})$; we must prove that $c_j = 0$ for $j = 1, \dots, n$. Let j be an arbitrary integer in the range $[1, n]$. Since no knot occurs more than $d + 1$ times, there is a nonempty interval $[t_{\mu}, t_{\mu+1})$ contained in $[t_j, t_{j+d+1}]$, the support of $B_{j,d}$. But all the nonzero B-splines on $[t_{\mu}, t_{\mu+1})$ are linearly independent, so $f(x) = 0$ on this interval implies that $c_k = 0$ for $k = \mu - d, \dots, \mu$. Since $B_{j,d}$ is one of the nonzero B-splines, we have in particular that $c_j = 0$. ■

The condition that no knot must occur with multiplicity higher than $d + 1$ is essential, for otherwise one of the B-splines will be identically zero and then they will certainly be linearly dependent. The other conditions are not essential for the linear independence, see Exercise 3.

3.2 Differentiation and smoothness of B-splines

Our study of differentiation and smoothness is based on the matrix representation of B-splines. But first of all we need to be fairly precise about what we mean by smoothness and jumps in the derivatives of a function.

3.2.1 Piecewise smooth functions

A C^r -function is a function whose derivatives up to order r are continuous at all points of its domain of definition. A piecewise smooth function is a function that is smooth except at some isolated points. Although this concept is quite simple, we need to be precise about the definition as we are going to perform computations with the jumps in a derivative of a spline.

Definition 3.11. A function f defined on some interval $[a, b]$ is piecewise continuous on $[a, b]$ provided f is continuous on $[a, b]$ except at a finite number of points $\{x_i\}_{i=1}^n$ where the one-sided limits

$$f(z+) = \lim_{\substack{x \rightarrow z \\ x > z}} f(x), \quad f(z-) = \lim_{\substack{x \rightarrow z \\ x < z}} f(x). \quad (3.25)$$

exist for $z = x_i$, and $i = 1, 2, \dots, n$. The number

$$J_z f = f(z+) - f(z-) \quad (3.26)$$

is called the jump of f at z .

Of course there are many functions with infinite jumps, but since the one-sided limits of a piecewise continuous function exist, these jumps are finite. In particular this holds for a piecewise polynomial.

We will also need to consider functions with piecewise continuous derivatives.

Definition 3.12. If the function f has piecewise continuous r th derivative $f^{(r)}$ on $[a, b]$ for some integer $r \geq 0$, it is said to be piecewise C^r . If $J_z(f^{(k)}) = 0$ for $k = 0, \dots, r$ at some $z \in (a, b)$, then f is said to be C^r at z . Differentiation of functions that are piecewise C^r is defined by

$$D^r f(x) = \begin{cases} D_+^r f(x), & x \in [a, b), \\ D_-^r f(x), & x = b, \end{cases}$$

where the right derivative D_+^r and the left derivative D_-^r are defined by

$$\begin{aligned} D_+^r f(x) &= f^{(r)}(x+), & x \in [a, b), \\ D_-^r f(x) &= f^{(r)}(x-), & x \in (a, b]. \end{aligned}$$

If f is piecewise C^r on $[a, b]$ then $D^r f(x)$ exists for every $x \in [a, b]$. At a point where the r th derivative of f is continuous, this definition of differentiation agrees with the standard one since the two one-sided derivatives $D_+^r f$ and $D_-^r f$ are equal at such a point.

3.2.2 Derivatives of B-splines

From Definition 3.12 and equation (2.6), we see that the r th derivative of a B-spline $B_{j,d}$ is given by

$$D^r B_{j,d} = \sum_{k=j}^{j+d} D^r B_{j,d}^k B_{k,0}, \quad r \geq 0, \quad (3.27)$$

where $D^r B_{j,d}^k$ is the ordinary r th derivative of the polynomial representing $B_{j,d}$ on the interval $[t_k, t_{k+1})$. This explicit formula is of little interest in practice because it is difficult to compute. What we want is something similar to the recurrence relation (2.1).

Our approach to derivatives of B-splines will instead follow a customary strategy: We start by considering what happens on one knot interval. We will then see that the formulas we obtain are independent of the specific knot interval so they can be generalized to splines.

Recall from Theorem 2.14 that on a knot interval $[t_\mu, t_{\mu+1})$, the row vector of the nonzero B-splines $\mathbf{B}_d$ is given by

$$\mathbf{B}_d(x)^T = \mathbf{R}_1(x) \cdots \mathbf{R}_d(x). \quad (3.28)$$

It turns out that we can differentiate this product of matrices as if the factors were numbers. Indeed, let $\mathbf{A}$ be a matrix where each entry is a function of x . The derivative $D\mathbf{A}$ of $\mathbf{A}$ is defined as the matrix obtained by differentiating each entry of $\mathbf{A}$ with respect to x . We have the following familiar rule for differentiating a product of two matrices.

Lemma 3.13. *Let $\mathbf{A}$ and $\mathbf{B}$ be two matrices with entries that are functions of x and with dimensions such that the matrix product $\mathbf{AB}$ makes sense. Then*

$$D(\mathbf{AB}) = (D\mathbf{A})\mathbf{B} + \mathbf{A}(D\mathbf{B}).$$

Proof. Let $(\mathbf{AB})_{ij}$ be an arbitrary entry of the matrix $\mathbf{AB}$. Then

$$\begin{aligned} D(\mathbf{AB})_{ij} &= D\left(\sum_k a_{ik} b_{kj}\right) = \sum_k D(a_{ik} b_{kj}) \\ &= \sum_k (Da_{ik} b_{kj} + a_{ik} Db_{kj}) \\ &= \sum_k (Da_{ik}) b_{kj} + \sum_k a_{ik} (Db_{kj}) \\ &= ((D\mathbf{A})\mathbf{B})_{ij} + (\mathbf{A}(D\mathbf{B}))_{ij} \end{aligned}$$

which proves the lemma. ■

Applying this rule to the product (3.28), we get

$$D\mathbf{B}_d(x)^T = \sum_{k=1}^d \mathbf{R}_1(x) \cdots \mathbf{R}_{k-1}(x) D\mathbf{R}_k(x) \mathbf{R}_{k+1}(x) \cdots \mathbf{R}_d(x), \quad (3.29)$$

where $D\mathbf{R}_k$ denotes the matrix obtained by differentiating each entry in $\mathbf{R}_k(x)$ with respect to x ,

$$D\mathbf{R}_k = \begin{pmatrix} \frac{-1}{t_{\mu+1}-t_{\mu+1-k}} & \frac{1}{t_{\mu+1}-t_{\mu+1-k}} & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & \frac{-1}{t_{\mu+k}-t_{\mu}} & \frac{1}{t_{\mu+k}-t_{\mu}} \end{pmatrix}. \quad (3.30)$$

The dimensions of the matrix $D\mathbf{R}_k$ are the same as those of $\mathbf{R}_k$, so both are transformations from $\mathbb{R}^{k+1}$ to $\mathbb{R}^k$.

The following lemma will help us simplify equation 3.29.

Lemma 3.14. *For $k \geq 2$ and any real number x , the matrices $\mathbf{R}_k$ and $\mathbf{R}_{k+1}$ satisfy the relation*

$$D\mathbf{R}_k \mathbf{R}_{k+1}(x) = \mathbf{R}_k(x) D\mathbf{R}_{k+1}. \quad (3.31)$$

Proof. Equation 3.31 follows by differentiating both sides of 3.7 with respect to z and letting $d = k + 1$. ■

By making use of equation 3.31, we can move the differentiation operator d in (3.29) from $\mathbf{R}_k$ to $\mathbf{R}_d$ in term k of the sum. The end result is

$$D\mathbf{B}_d(x)^T = d \mathbf{R}_1(x) \cdots \mathbf{R}_{d-1}(x) D\mathbf{R}_d = d \mathbf{B}_{d-1}(x)^T D\mathbf{R}_d. \quad (3.32)$$

Let us now see how higher derivatives of B-splines can be determined. To find the second derivative, we differentiate (3.32). Since $D(D\mathbf{R}_d) = 0$, we obtain

$$D^2 \mathbf{B}_d(x)^T = d D\mathbf{B}_{d-1}(x)^T D\mathbf{R}_d.$$

If we apply (3.32) to $D\mathbf{B}_{d-1}$, we find

$$D^2 \mathbf{B}_d(x)^T = d(d-1) \mathbf{B}_{d-2}(x)^T D\mathbf{R}_{d-1} D\mathbf{R}_d.$$

In general, for the r th derivative, we find

$$D^r \mathbf{B}_d(x)^T = \frac{d!}{(d-r)!} \mathbf{B}_{d-r}(x)^T D\mathbf{R}_{d-r+1} \cdots D\mathbf{R}_d.$$

Since in addition $\mathbf{B}_{d-r}(x)^T = \mathbf{R}_1(x) \cdots \mathbf{R}_{d-r}(x)$, the following theorem has been proved.

Theorem 3.15. Let x be a number in $[t_\mu, t_{\mu+1})$. Then the r th derivative of the vector of B-splines $\mathbf{B}_d(x) = (B_{\mu-d,d}(x), \dots, B_{\mu,d}(x))^T$ is given by

$$D^r \mathbf{B}_d(x)^T = \frac{d!}{(d-r)!} \mathbf{B}_{d-r}(x)^T D\mathbf{R}_{d-r+1} \cdots D\mathbf{R}_d. \quad (3.33)$$

Suppose that $f(x) = \sum_{j=1}^n c_j B_{j,d}(x)$. With $\mathbf{c}_0 = (c_{\mu-d}, \dots, c_\mu)^T$ the r 'th derivative of f at x is given by

$$D^r f(x) = \frac{d!}{(d-r)!} \mathbf{R}_1(x) \cdots \mathbf{R}_{d-r}(x) D\mathbf{R}_{d-r+1} \cdots D\mathbf{R}_d \mathbf{c}_0, \quad (3.34)$$

for any integer r such that $0 \leq r \leq d$.

Note that the symmetry property (3.31) gives us a curious freedom in how to represent the r th derivative: It does not matter which of the d matrices $\mathbf{R}_k$ we differentiate as long as we differentiate r of them. In Theorem 3.15 it is the r matrices of largest dimension that have been differentiated.

Theorem 3.15 is the basis for algorithms for differentiating splines and B-splines, see Section 3.2.3 below. But let us first record the following recurrence relation for the derivative of a B-spline.

Theorem 3.16. The derivative of the j th B-spline of degree d on $\mathbf{t}$ is given by

$$DB_{j,d}(x) = d \left(\frac{B_{j,d-1}(x)}{t_{j+d} - t_j} - \frac{B_{j+1,d-1}(x)}{t_{j+1+d} - t_{j+1}} \right) \quad (3.35)$$

for $d \geq 1$ and for any real number x . The derivative of $B_{j,d}$ can also be expressed as

$$DB_{j,d}(x) = \frac{d}{d-1} \left(\frac{x - t_j}{t_{j+d} - t_j} DB_{j,d-1}(x) + \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} DB_{j+1,d-1}(x) \right) \quad (3.36)$$

for $d \geq 2$ and any x in $\mathbb{R}$. Here the $'0/0 = 0'$ convention is used.

Proof. Equation (3.35) clearly holds if $x \notin [t_j, t_{j+1+d})$, as then both sides of the equation are identically zero. Suppose therefore that $x \in [t_\mu, t_{\mu+1})$ for some $j \leq \mu \leq j+d$. Equation (3.33) with $r = 1$ states that

$$(DB_{\mu-d,d}(x), \dots, DB_{\mu,d}(x)) = d(B_{\mu-d+1,d-1}(x), \dots, B_{\mu,d-1}(x)) D\mathbf{R}_d.$$

Carrying out the matrix multiplication on the right and comparing the j th component on both sides, we obtain (3.35), with x restricted to the interval $[t_\mu, t_{\mu+1})$. But since (3.35) is independent of μ , it actually holds for all $x \in [t_j, t_{j+d+1})$.

Equation (3.36) is proved in a similar way, we just use Lemma 3.14 and differentiate the matrix $\mathbf{R}_1$ instead of $\mathbf{R}_d$, see Exercise 6. ■

3.2.3 Computing derivatives of splines and B-splines

From Theorem 3.15, we know that the r th derivative of a spline f is given by

$$D^r f(x) = \frac{d!}{(d-r)!} \mathbf{R}_1(x) \cdots \mathbf{R}_{d-r}(x) D \mathbf{R}_{d-r+1} \cdots D \mathbf{R}_d \mathbf{c}_0. \quad (3.37)$$

Just as for evaluation (see Section 2.5), there are two algorithms for computing this derivative; either from left to right or from right to left.

As before, we assume that x lies in the interval $[t_\mu, t_{\mu+1})$ and that the vector $\mathbf{c}_0 = (c_{\mu-d}, \dots, c_\mu)^T$ contains the B-spline coefficients that multiply the B-splines that are nonzero on $[t_\mu, t_{\mu+1})$. We then have the DL (Derivative Left) Algorithm which computes $D^r f(x)$ by accumulating matrix products from right to left in (3.37). The DR (Derivative Right) Algorithm computes the r th derivative of all the nonzero B-splines at x by accumulating matrix products from left to right, then multiplying with the coefficients and summing.

Algorithm 3.17 (DL). Let the polynomial degree d , the $2d$ knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$, the B-spline coefficients $\mathbf{c}_0^{(0)} = \mathbf{c}_0 = (c_{\mu-d}, \dots, c_\mu)^T$ of a spline f , and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of the products

$$\begin{aligned} \mathbf{c}_{d-k+1}^{(d-k+1)} &= D \mathbf{R}_k \mathbf{c}_{d-k}^{(d-k)}, & k &= d, \dots, d-r+1, \\ \mathbf{c}_{d-k+1}^{(r)} &= \mathbf{R}_k(x) \mathbf{c}_{d-k}^{(r)}, & k &= d-r, \dots, 1, \end{aligned}$$

the r th derivative of f at x is given by

$$D^r f(x) = d! \mathbf{c}_d^{(r)} / (d-r)!.$$

Algorithm 3.18 (DR). Let the polynomial degree d , the knots $t_{\mu-d+1} \leq t_\mu < t_{\mu+1} \leq t_{\mu+d}$ and a number x in $[t_\mu, t_{\mu+1})$ be given and set $\mathbf{B}_0 = 1$. After evaluation of the products

$$\begin{aligned} \mathbf{B}_k(x)^T &= \mathbf{B}_{k-1}(x)^T \mathbf{R}_k(x), & k &= 1, \dots, d-r, \\ D^{k-d+r} \mathbf{B}_k(x)^T &= D^{k-d+r-1} \mathbf{B}_{k-1}(x)^T D \mathbf{R}_k, & k &= d-r+1, \dots, d, \end{aligned}$$

the vector $\frac{d!}{(d-r)!} D^r \mathbf{B}_d(x)$ will contain the value of the r th derivative of the nonzero B-splines at x ,

$$\frac{d!}{(d-r)!} D^r \mathbf{B}_d(x) = (D^r B_{\mu-d,d}(x), \dots, D^r B_{\mu,d}(x))^T.$$

Figure 3.1 shows how the second derivative of a cubic spline can be computed, while Figure 3.2 shows the computation of the first derivative of all the nonzero B-splines at a point. In Algorithm 3.17, we have to compute the two matrix-vector products $D \mathbf{R}_k \mathbf{c}_{d-k}$ and $\mathbf{R}_k(x) \mathbf{c}_{d-k}$. The component form of the latter product is given in (2.20), while the component

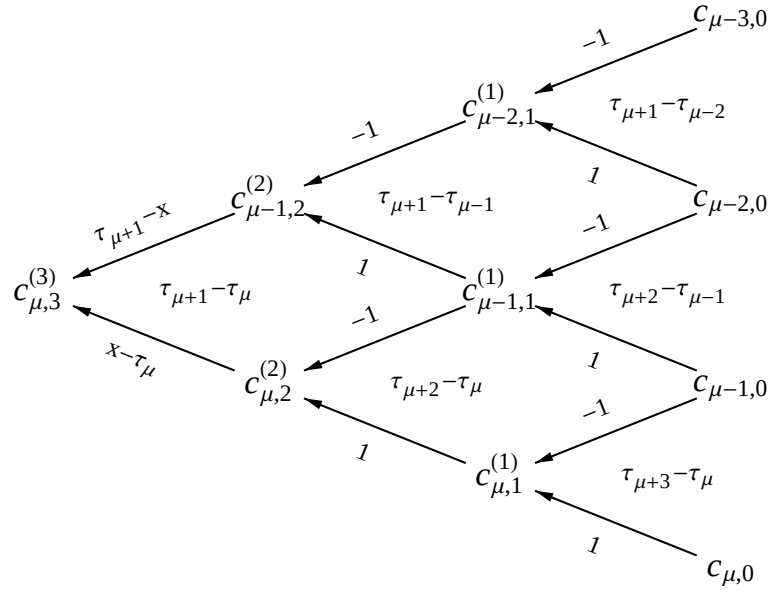


Figure 3.1. A triangular algorithm for computation of the second derivative of a cubic spline at x .

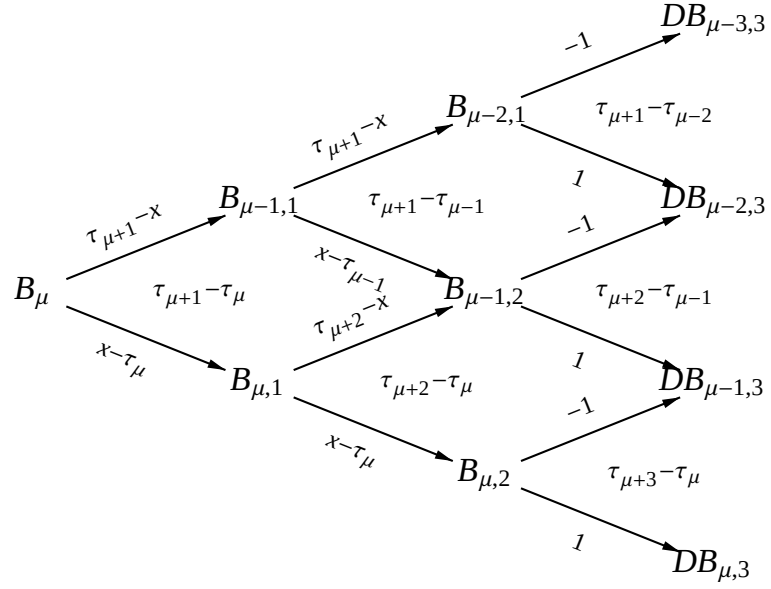


Figure 3.2. A triangular algorithm for computation of the derivative of the nonzero cubic B-splines at x .

form of the former is obtained by differentiating the linear factors in (2.20) with respect to x . The result is

$$(D\mathbf{R}_k \mathbf{c}_{d-k})_j = \frac{c_{d-k,j} - c_{d-k,j-1}}{t_{j+k} - t_j} \quad (3.38)$$

for $j = \mu - k + 1, \dots, \mu$.

The alternative algorithm accumulates the matrix products in (2.18) from left to right. The component form of the product $\mathbf{B}_{k-1}(x)^T \mathbf{R}_k$ is given in (2.25), while the component form of the product $\mathbf{B}_{k-1}(x)^T D\mathbf{R}_k$ is

$$(\mathbf{B}_{k-1}(x)^T D\mathbf{R}_k)_j = \frac{B_{j,k-1}(x)}{t_{j+k} - t_j} - \frac{B_{j+1,k-1}(x)}{t_{j+1+k} - t_{j+1}} \quad (3.39)$$

for $j = \mu - k, \dots, \mu$.

3.2.4 Smoothness of B-splines

A characteristic feature of splines is their smoothness properties as stated in Theorem 1.2 in Chapter 1. In this section we will prove the smoothness properties of splines. We start by stating the precise smoothness of a B-spline.

Theorem 3.19. Suppose that the number z occurs m times among the knots $t_j, t_{j+1}, \dots, t_{j+d+1}$, defining the B-spline $B_{j,d}$. If $1 \leq m \leq d+1$, then $D^r B_{j,d}$ is continuous at z for $r = 0, 1, \dots, d-m$, but $D^{d-m+1} B_{j,d}$ is discontinuous at z .

This theorem will be proved via a sequence of steps. We first note from the explicit formula (??) for the Bernstein basis that Theorem 3.19 holds for $m = d+1$. At such a knot the B-spline is discontinuous with a jump of size 1. In particular Theorem 3.19 holds for $d = 0$ and $d = 1$.

The first step in the proof of Theorem 3.19 is to show that a B-spline is continuous at a knot of multiplicity at most d .

Lemma 3.20. Suppose that no knot among $t_j, t_{j+1}, \dots, t_{j+d+1}$ occurs more than d times. Then the B-spline $B_{j,d}$ is continuous everywhere.

Proof. The proof is by induction on the degree d . If no knots occur more than $d-1$ times, then by the induction hypothesis the two B-splines $B_{j,d-1}$ and $B_{j+1,d-1}$ are both continuous which means that $B_{j,d}$ is also continuous. Suppose next that x is equal to a knot which occurs exactly d times among $t_j, t_{j+1}, \dots, t_{j+d+1}$. There are three cases. Suppose first that $x = t_j = \dots = t_{j+d-1} < t_{j+d} \leq t_{j+d+1}$. Then $B_{j,d-1}$ is discontinuous, but the term $(x - t_j)B_{j,d-1}(x)$ in (2.1) is continuous at x . Moreover, from the induction hypothesis $B_{j+1,d-1}$ is continuous, and it follows that $B_{j,d}$ is continuous. The proof in the case $x = t_{j+1+d}$ is similar. Finally, if $t_j < x < t_{j+1+d}$, then $x = t_{j+1} = \dots = t_{j+d}$ and the result follows from the explicit formula in Exercise 9. This completes the proof. ■

Proof. [The continuity part of Theorem 3.19]

For $r = 0$ the result follows from Lemma 3.20. Differentiating the differentiation formula (3.35) a total of $r - 1$ times leads to

$$D^r B_{j,d}(x) = d \left(\frac{D^{r-1} B_{j,d-1}(x)}{t_{j+d} - t_j} - \frac{D^{r-1} B_{j+1,d-1}(x)}{t_{j+1+d} - t_{j+1}} \right).$$

By induction on d it follows that $D^r B_{j,d}$ is continuous for r in the range $1 \leq r \leq d - m$. ■

To complete the proof of the continuity property, we determine the jump in the first discontinuous derivative of a B-spline.

Lemma 3.21. Suppose that the number z occurs exactly m times among the knots $t_j, \dots, t_{j+1+d}$. Then the $d - m + 1$ th derivative of $B_{j,d}$ has a jump at z given by

$$J_z(D^{d-m+1} B_{j,d}) = \frac{d!}{(m-1)!} (t_{j+1+d} - t_j)^l \prod_{\substack{k=j \\ t_k \neq z}}^{j+1+d} (t_k - z) \neq 0. \quad (3.40)$$

Proof. As usual, the proof is by induction of the degree d . We use the jump recurrence relation

$$J_x(D^r B_{j,d}) = d \left(\frac{J_x(D^{r-1} B_{j,d-1})}{t_{j+d} - t_j} - \frac{J_x(D^{r-1} B_{j+1,d-1})}{t_{j+1+d} - t_{j+1}} \right), \quad \text{for } x \in \mathbb{R} \text{ and } r \geq 1, \quad (3.41)$$

see Exercise 7 for a proof. We first note that (3.40) holds in the case where $m = d + 2$, so we may assume that $m \leq d + 1$. It is easy to check that equation (3.40) holds when $d = 0$ and $m = 1$. Suppose that (3.40) holds for B-splines of degree $d - 1$. For a B-spline of degree d , we apply (3.41) with $r = d - m + 1$. There are three cases to consider. Suppose first that $z = t_j$. Since z occurs $m - 1$ times among the knots of $B_{j+1,d-1}$, it follows from the continuity property that $J_z(D^{d-m} B_{j+1,d-1}) = 0$. In view of the induction hypothesis, equation (3.41) therefore takes the form

$$J_z(D^{d-m+1} B_{j,d}) = d \frac{J_z(D^{d-m} B_{j,d-1})}{t_{j+d} - t_j} = \frac{d!}{(m-1)!} \prod_{\substack{k=j \\ t_k \neq t_j}}^{j+d} (t_k - t_j).$$

Multiplying the numerator and denominator by $t_{j+1+d} - t_j$ proves (3.40) in this case. A similar argument is valid when $z = t_{j+1+d}$.

The remaining situation is $t_j < z < t_{j+1+d}$. In this case both $B_{j,d-1}$ and $B_{j+1,d-1}$ have a

knot of multiplicity m at z . Applying (3.41) and the induction hypothesis, we then obtain

$$\begin{aligned} J_z(D^{d-m+1}B_{j,d}) &= \frac{d!}{(m-1)!} \left(\prod_{\substack{k=j \\ t_k \neq z}}^{j+d} (t_k - z)^{-1} - \prod_{\substack{k=j+1 \\ t_k \neq z}}^{j+1+d} (t_k - z)^{-1} \right) \\ &= \frac{d!}{(m-1)!} \prod_{\substack{k=j+1 \\ t_k \neq z}}^{j+d} (t_k - z)^{-1} \left(\frac{1}{t_j - z} - \frac{1}{t_{j+1+d} - z} \right) \\ &= \frac{d!}{(m-1)!} (t_{j+1+d} - t_j)^{-1} \prod_{\substack{k=j \\ t_k \neq z}}^{j+1+d} (t_k - z) \end{aligned}$$

which completes the proof. ■

3.2.5 The integral of a B-spline

Using the continuity property and differentiation formula we can show a formula for the integral of a B-spline.

Theorem 3.22. *The integral of the j th B-spline of degree $d \geq 0$ on t is given by*

$$\int_{t_j}^{t_{j+d+1}} B_{j,d,t}(x) dx = \frac{t_{j+d+1} - t_j}{d+1}. \quad (3.42)$$

Proof. If $t_{j+d+1} = t_j$ there is nothing to prove. Assume first $t_{j+d} > t_j$. Define a knot vector $\mathbf{s} = (s_i)_{i=1}^{2d+3}$ by

$$s_1 = \cdots = s_{d+1} = a, \quad s_{d+2+k} = t_{j+k}, \quad k = 0, 1, \dots, d+1,$$

where a is any number strictly less than t_j . Integrating the differentiation formula (3.35) for B-splines of degree $d+1$ on $\mathbf{s}$ gives

$$\int_{s_i}^{s_{i+d+2}} DB_{i,d+1,\mathbf{s}}(x) dx = J_i - J_{i+1}, \quad i = 1, 2, \dots, d+1, \quad (3.43)$$

where

$$J_r = \frac{d+1}{s_{r+d+1} - s_r} \int_{s_r}^{s_{r+d+1}} B_{r,d,\mathbf{s}}(x) dx, \quad r = i, i+1,$$

and we used the local support property of $B_{r,d,\mathbf{s}}$. Since no knot in $\mathbf{s}$ occurs more than $d+1$ times it follows from Lemma 3.20 that each $B_{i,d+1,\mathbf{s}}$ is continuous. Therefore $B_{i,d+1,\mathbf{s}}$ vanishes at the endpoints s_i and s_{i+d+2} of its support. Moreover, since $t_{j+d} > t_j$, each $B_{i,d+1,\mathbf{s}}$ has a continuous derivative in the interior of its support. Therefore the integral on the left of (3.43) is zero and $J_{i+1} = J_i$ for $i = 1, \dots, d+1$. In particular, $J_{d+2} = J_1$ and this implies (3.42). Indeed, since $s_{d+2} = t_j$, $s_{2d+3} = t_{j+d+1}$ and $s_1 = \cdots = s_{d+1}$ we have

$$J_{d+2} = \frac{d+1}{t_{j+d+1} - t_j} \int_{t_j}^{t_{j+d+1}} B_{j,d,t}(x) dx = J_1 = \frac{d+1}{s_{d+2} - s_1} \int_{s_1}^{s_{d+2}} \frac{(s_{d+2} - x)^d}{(s_{d+2} - s_1)^d} dx = 1.$$

Finally, if $t_j = t_{j+d} < t_{j+d+1}$ then (3.42) follows by direct calculation, (Cf. the calculation for J_1). ■

3.3 B-splines as a basis for piecewise polynomials

Our ultimate purpose is to use B-splines as building blocks for constructing and representing functions and data, but what exactly are the functions in a spline space $\mathbb{S}_{d,t}$? We know that they are piecewise polynomials, with different polynomial pieces meeting at the knots. We also know that the exact continuity between two pieces is controlled by the multiplicity of the knot at the join. If the knot z occurs with multiplicity m , we know from Theorem 3.19 that there is at least one B-spline with its first $d - m$ derivatives continuous, but with the derivative of order $d - m + 1$ discontinuous. When we take linear combinations of the B-splines and form $\mathbb{S}_{d,t}$, the spline functions will in general inherit this smoothness at z , although there will be some functions that will be even smoother, like for example the function with all coefficients zero, the zero function. In this section we will start by defining piecewise polynomial spaces in terms of the smoothness at the joins and show that $\mathbb{S}_{d,t}$ can be characterised in this way. We start by defining piecewise polynomial spaces.

Definition 3.23. Let d be a nonnegative integer, let $[a, b]$ be a real interval, let the sequence $\Delta = (\xi_i)_{i=1}^N$ be a partition of $[a, b]$,

$$a = \xi_1 < \xi_2 < \cdots < \xi_{N-1} < \xi_N = b,$$

and let $\mathbf{r} = (r_i)_{i=2}^{N-1}$ be a sequence of integers. By $\mathbb{S}_d^{\mathbf{r}}(\Delta)$ we denote the linear space of piecewise polynomials of degree d on $[a, b]$ with r_i continuous derivatives at ξ_i . In other words $f \in \mathbb{S}_d^{\mathbf{r}}(\Delta)$ if and only if the restriction of f to (ξ_{i-1}, ξ_i) is a polynomial of degree d for $i = 2, \dots, N$, and $D^k f$ is continuous at ξ_i for $k = 0, \dots, r_i$ and $i = 2, \dots, N - 1$.

It is quite simple to check that linear combinations of functions in $\mathbb{S}_d^{\mathbf{r}}(\Delta)$ are again in $\mathbb{S}_d^{\mathbf{r}}(\Delta)$; it is therefore a linear space.

To derive a representation for functions $f \in \mathbb{S}_d^{\mathbf{r}}(\Delta)$ we define the functions $\theta_{i,k} : \mathbb{R} \rightarrow \mathbb{R}$ by

$$\theta_{i,k}(x) = (x - \xi_i)_+^k, \quad i = 1, \dots, N, \quad k = 0, 1, \dots, \quad (3.44)$$

where

$$a_+^k = \begin{cases} a^k, & \text{if } a > 0; \\ 0, & \text{otherwise;} \end{cases}$$

except that we use the convention $0^0 = 1$. The functions $\theta_{i,k}$ are called *truncated powers*. We have

$$\text{Jump}_{\xi_i}(D^r \theta_{i,k}) = \begin{cases} 0, & \text{if } r < k; \\ k!, & \text{if } r = k; \end{cases} \quad i = 1, \dots, N, \quad k = 0, 1, \dots \quad (3.45)$$

Lemma 3.24. Any $f \in \mathbb{S}_d^r(\Delta)$ can be written in the form

$$f = \sum_{i=1}^{N-1} \sum_{k=r_i+1}^d c_{i,k} \theta_{i,k}, \quad c_{i,k} \in \mathbb{R}, \quad r_1 = -1. \quad (3.46)$$

Proof. By (3.45) it follows that $\theta_{i,k} \in \mathbb{S}_d^r(\Delta)$ for $r_i < k \leq d$ and $i = 1, \dots, N-1$. Since $\mathbb{S}_d^r(\Delta)$ is a linear space any f in the form (3.46) must belong to $\mathbb{S}_d^r(\Delta)$. Suppose next $f \in \mathbb{S}_d^r(\Delta)$ and let f_i be the polynomial representing f on (ξ_i, ξ_{i+1}) . Now $f_1(x) = \sum_{k=0}^d c_{1,k} (x - \xi_1)^k = \sum_{k=r_1+1}^d c_{1,k} \theta_{1,k}(x)$ for $x \in [\xi_1, \xi_N)$. Suppose for some $j \geq 2$ that $f_{j-1}(x) = \sum_{i=1}^{j-1} \sum_{k=r_i+1}^d c_{i,k} \theta_{i,k}(x)$ for $x \in [\xi_1, \xi_N)$. Since $f \in C^{r_j}(\xi_j)$ we have for some $c_{j,k} \in \mathbb{R}$

$$f_j(x) - f_{j-1}(x) = \sum_{k=r_j+1}^d c_{j,k} (x - \xi_j)^k.$$

It follows that

$$f_j(x) = \sum_{i=1}^j \sum_{k=r_i+1}^d c_{i,k} \theta_{i,k}(x), \quad x \in [\xi_1, \xi_N), \quad j = 1, \dots, N-1.$$

Now if $x \in [\xi_1, \xi_N)$ then $x \in [\xi_j, \xi_{j+1})$ for some j , and since $\theta_{i,k}(x) = 0$ for $i > j$ we have

$$f(x) = f_j(x) = \sum_{i=1}^{N-1} \sum_{k=r_i+1}^d c_{i,k} \theta_{i,k}(x).$$

This completes the proof. ■

The following theorem shows that any $f \in \mathbb{S}_d^r(\Delta)$ can in fact be represented in terms of B-splines on an appropriate knot vector.

Theorem 3.25 (Curry-Schoenberg). Let $\mathbb{S}_d^r(\Delta)$ be a given space of piecewise polynomials and let the $d+1$ -extended knot vector $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be defined by

$$\mathbf{t} = (t_1, \dots, t_{d+1}, \overbrace{\xi_2, \dots, \xi_2}^{d-r_2}, \dots, \overbrace{\xi_i, \dots, \xi_i}^{d-r_i}, \dots, \overbrace{\xi_{N-1}, \dots, \xi_{N-1}}^{d-r_{N-1}}, t_{n+1}, \dots, t_{n+d+1}),$$

where

$$n = \sum_{i=1}^{N-1} d - r_i, \quad r_1 = -1, \quad (3.47)$$

and for any end knots satisfying $t_1 \leq \dots \leq t_{d+1} \leq a$ and $b \leq t_{n+1} \leq \dots \leq t_{n+d+1}$. Then

$$\mathbb{S}_d^r(\Delta) = \mathbb{S}_{d,\mathbf{t}}|_{[a,b]},$$

where $\mathbb{S}_{d,\mathbf{t}}|_{[a,b]}$ is the space obtained by restricting the functions in $\mathbb{S}_{d,\mathbf{t}}$ to the interval $[a, b]$.

Proof. Let $\mathbb{S} = \mathbb{S}_{d,\mathbf{t}}|_{[a,b]}$. We note that by the construction of the knot vector, the B-splines in $\mathbb{S}$ satisfy the smoothness conditions of $\mathbb{S}_d^r(\Delta)$ so $\mathbb{S} \subseteq \mathbb{S}_d^r(\Delta)$. Moreover, since $\mathbf{t}$ is $d+1$ -extended, the B-splines in $\mathbb{S}$ are linearly independent so $\dim \mathbb{S} = n$. On the other hand, $\mathbb{S}_d^r(\Delta)$ is spanned by n functions $\theta_{i,k}$ so $\dim \mathbb{S} \geq \dim \mathbb{S}_d^r(\Delta)$. But a subspace that has the same dimension as the full space must agree with the full space so $\mathbb{S} = \mathbb{S}_d^r(\Delta)$. Moreover the $\theta_{i,k}$ must be linearly independent. ■

Corollary 3.26 (Nestedness). *Let n be a positive integer, $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ a given knot vector and let $\mathbf{t} = (t_i)_{i=1}^{n+d+2}$ be the knot vector obtained by inserting a knot anywhere in $\boldsymbol{\tau}$. Then*

$$\mathbb{S}_{d,\boldsymbol{\tau}} \subset \mathbb{S}_{d,\mathbf{t}}. \quad (3.48)$$

Proof. We extend $\boldsymbol{\tau}$ and $\mathbf{t}$ by adding $d+1$ knots $x_1 = \dots = x_{d+1} < \min(\tau_1, t_1)$, and similarly $d+1$ knots $y_1 = \dots = y_{d+1} > \max(\tau_{n+d+1}, t_{n+d+2})$. For simplicity we denote these extended knot vectors also by $\boldsymbol{\tau}$ and $\mathbf{t}$ and we have the corresponding piecewise polynomial spaces $\mathbb{S}_d^r(\Delta_{\boldsymbol{\tau}})$ and $\mathbb{S}_d^r(\Delta_{\mathbf{t}})$. Since $\mathbb{S}_d^r(\Delta_{\mathbf{t}})$ is obtained from $\mathbb{S}_d^r(\Delta_{\boldsymbol{\tau}})$ by either adding a segment or reducing the continuity at a breakpoint we have $\mathbb{S}_d^r(\Delta_{\boldsymbol{\tau}}) \subset \mathbb{S}_d^r(\Delta_{\mathbf{t}})$. The result then follows from Theorem 3.25. ■

3.4 Exercises

3.1 a) Show that the quadratic B-spline

$$B[0,0,1,2](x) = (2x - \frac{3}{2}x^2)B[0,1](x) + \frac{1}{2}(2-x)^2B[1,2](x)$$

is continuous on $\mathbb{R}$.

b) Show that the first derivative

$$DB[0,0,1,2](x) = (2-3x)B[0,1](x) - (2-x)B[1,2](x)$$

is piecewise continuous on $\mathbb{R}$ with a discontinuity at $x = 0$.

c) Show that the second derivative

$$D^2B[0,0,1,2](x) = -3B[0,1](x) + B[1,2](x)$$

is piecewise continuous on $\mathbb{R}$ with discontinuities at 0, 1, and 2.

d) What about the third derivative?

3.2 Suppose that $d = 3$ and that $\hat{\mathbf{t}} = (0,0,1,3,4,5)$ so we can associate two cubic B-splines $\hat{B}_{1,3}$ and $\hat{B}_{2,3}$ with $\hat{\mathbf{t}}$. We want to prove that these two B-splines are linearly independent on $[1,3]$.

- a) Let $\mathbf{t}$ denote the augmented knot vector $\mathbf{t} = (0, 0, 0, 1, 3, 4, 5, 5)$. Show that we can associate 4 B-splines $\{B_{i,3}\}_{i=1}^4$ with $\mathbf{t}$ and that these are linearly independent on $[1, 3]$.
- b) Show that the two B-splines $\hat{B}_{1,3}$ and $\hat{B}_{2,3}$ are linearly independent.
- 3.3** Let $\mathbf{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector with $n \geq 1$ and such that no knot occurs more than $d+1$ times. Show that the B-splines $\{B_{j,d}\}_{j=1}^n$ are linearly independent on the interval $[t_1, t_{n+d+1})$.

- 3.4** Let $\mathbf{A}$ be matrix where each entry is a function of x and let α be a scalar function of x . Prove the formula

$$D(\alpha \mathbf{A}) = (D\alpha) \mathbf{A} + \alpha (D\mathbf{A}).$$

- 3.5** a) Count the number of operations (additions/subtractions, multiplications, divisions) involved in computing the matrix $\mathbf{R}_k(x)$ defined in (2.15). Do the same for the matrix $D\mathbf{R}_k$ defined in (3.30).
- b) Recall that in the formula (3.34) for the r th derivative of f , we have the freedom to differentiate any r of the d matrices $\{\mathbf{R}_k(x)\}_{k=1}^d$. Based on the count in (a), show that the choice made in (3.34) is the most efficient.

- 3.6** In this exercise we are going to prove the differentiation formula (3.36).

- a) Show that

$$(DB_{\mu-d,d}(x), \dots, dB_{\mu,d}(x)) = dD\mathbf{R}_1\mathbf{R}_2(x) \cdots \mathbf{R}_d(x) \quad (3.49)$$

for any x in $[t_\mu, t_{\mu+1})$.

- b) Show that (3.49) leads to (3.36) and that the latter equation is valid for any x . Why do we need the restriction $d \geq 2$?

- 3.7** a) Show that the jump in $B_{j,d}$ at x satisfies the recurrence relation

$$J_x(B_{j,d}) = \frac{x - t_j}{t_{j+d} - t_j} J_x(B_{j,d-1}) + \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} J_x(B_{j+1,d-1}), \quad (3.50)$$

with

$$J_x(B_{j,0}) = \begin{cases} 1, & \text{if } x = t_j, \\ -1, & \text{if } x = t_{j+1}, \\ 0, & \text{otherwise.} \end{cases} \quad (3.51)$$

- b) Show that for $r \geq 1$, the jump in the r th derivative at any $x \in \mathbb{R}$ is given by

$$J_x(D^r B_{j,d}) = d \left(\frac{J_x(D^{r-1} B_{j,d-1})}{t_{j+d} - t_j} - \frac{J_x(D^{r-1} B_{j+1,d-1})}{t_{j+1+d} - t_{j+1}} \right), \quad \text{for } x \in \mathbb{R} \text{ and } r \geq 1. \quad (3.52)$$

The convention that $'0/0 = 0'$ is used in (3.50) and (3.41).

3.8 Suppose in Definition 3.23 that $d = 3$ and $r_i = 1$, $i = 2, \dots, N - 1$. Show using (3.46) that any $f \in \mathbb{S}_d^r(\Delta)$ can be represented in the form

$$f(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + \sum_{i=2}^{N-1} c_{i,3}(x - \xi_i)_+^3, \quad x \in [\xi_1, \xi_N].$$

Knot insertion

In Chapter 1 we were led to B-splines, defined via the recurrence relation, as a convenient way to represent spline functions. In Chapters 2 and 3 we then established some of the basic properties of splines, with the recurrence relation as the major tool. We have seen that splines can be evaluated efficiently and stably, we have studied the smoothness of splines, we have shown that B-splines are linearly independent and that they form a basis for certain spaces of piecewise polynomials.

This chapter supplements the recurrence relation for B-splines with another very versatile tool, namely the idea of *knot insertion* or *knot refinement*. We have already seen that the control polygon of a spline provides a rough sketch of the spline itself. It turns out that the control polygon approaches the spline it represents as the distance between the knots of a spline is reduced, a fact that will be proved in Chapter 9. This indicates that it is of interest to see how the B-spline coefficients of a fixed spline depend on the knots.

Knot insertion amounts to what the name suggests, namely insertion of knots into an existing knot vector. The result is a new spline space with more B-splines and therefore more flexibility than the original spline space. This can be useful in many situations, for example in interactive design of spline curves. It turns out that the new spline space contains the original spline space as a subspace, so any spline in the original space can also be represented in terms of the B-splines in the refined space. As mentioned above, an important property of this new representation is that the control polygon will have moved closer to the spline itself. This provides us with a new and very powerful tool both for algorithmic manipulation and theoretical investigations of spline functions.

We start, in Section 4.1, by showing some simple examples of knot insertion. In Section 4.2 we then develop algorithms for expressing the B-spline coefficients relative to a refined knot vector in terms of the B-spline coefficients relative to the original knot vector. It turns out that the B-spline coefficients of a spline are completely characterised by three simple properties, and this is the topic of Section 4.3. This characterisation is often useful for developing the theory of splines, and in Section 4.4 this characterisation is used to obtain formulas for inserting one new knot into a spline function. Finally, in Section 4.5, we make use of knot insertion to prove that the number of sign changes in a spline is bounded by the

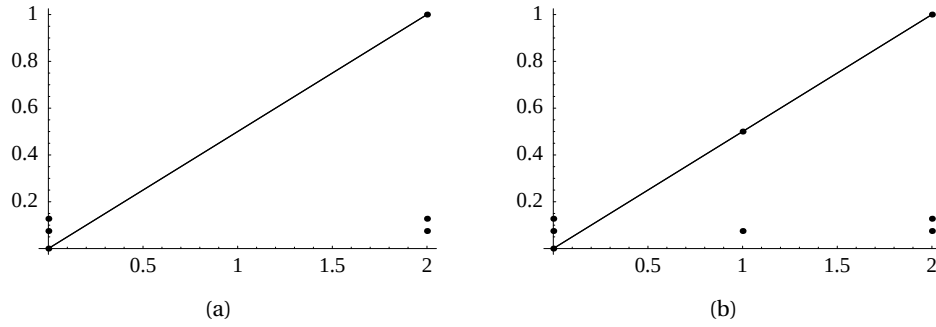


Figure 4.1. A line segment represented as a linear spline with no interior knots (a), and with one interior knot (b).

number of sign changes in its control polygon; another instance of the close relationship between a spline and its control polygon.

4.1 The control polygon relative to different knot vectors

In this introductory section we will consider some examples of knot insertion with the purpose of gaining an intuitive understanding of this important concept.

Figure 4.1 shows spline representations of a line segment. We all know that a straight line is uniquely determined by two points and in (a) the line segment is represented by its two end points. Although one may wonder what the point is, we can of course also represent the line segment by cutting it into smaller pieces and represent each of these pieces. This is what is shown in Figure 4.1 (b) where the line segment is represented by a linear spline with an interior knot at 1 which in effect means that we are using a redundant representation of three points to represent a line segment.

The redundancy in the representation is obvious and seems useless in the linear case. But let us increase the degree and consider a quadratic example. Figure 4.2 shows part of the parabola $y = (4x - x^2)/6$ represented as a spline without interior knots in (a) and with one interior knot in (b). In general, the representation in (b) requires a spline function and its first derivative to be continuous at $x = 1$, whereas a jump is allowed in the second derivative. The parabola in the figure is certainly continuous and has continuous first derivative at $x = 1$, but the jump in the second derivative happens to be 0. The knot at $x = 1$ is therefore redundant, but it has the nice effect of bringing the control polygon closer to the spline. We shall see later that there may be many other good reasons for inserting knots into a spline function.

An example with a cubic spline is shown in Figure 4.3. The situation is the same as before: The refined knot vector allows jumps in the second derivative at $x = 1$ and the third derivative at $x = 2$, but the jumps may be 0. For the specific spline in (a) these jumps are indeed 0, but one advantage of representing it in the refined spline space is that the control polygon comes closer to the spline.

The examples have hopefully shown that insertion of knots can be useful; at the very least it seems like it may be a useful tool for plotting splines. In the next sections we are going to develop algorithms for computing the B-spline coefficients on a refined knot vector and

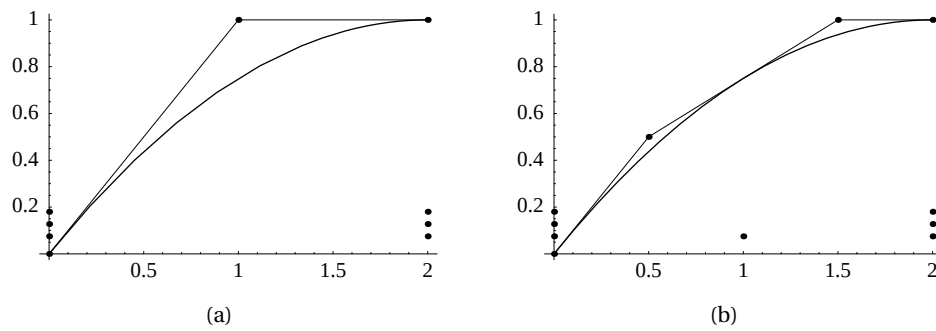


Figure 4.2. A piece of a parabola represented as a quadratic spline with no interior knots (a), and with one interior knot (b).

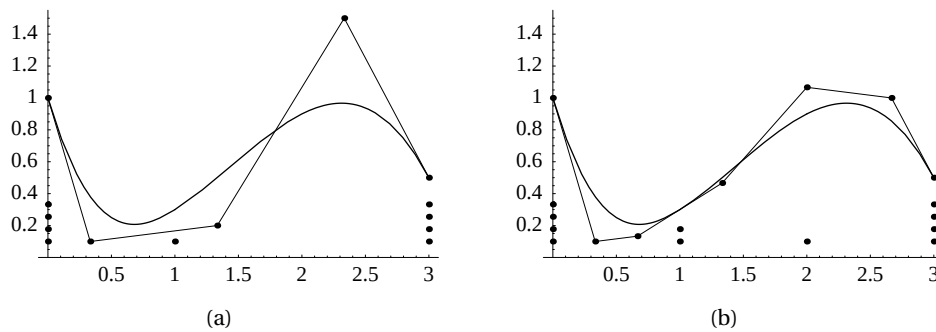


Figure 4.3. A cubic spline with one interior knot (a). In (b) the same spline is represented with two extra knots (the knot at $x=1$ is now double).

deduct various properties of the B-splines coefficients as functions of the knots. A proof of the fact that the control polygon converges to the spline it represents as the knot spacing goes to zero has to wait until Chapter 9.

4.2 Knot insertion

In this section we are going to develop two algorithms for computing the B-spline coefficients of a given spline relative to a refined knot vector. The two algorithms for knot insertion are closely related to Algorithms 2.16 and 2.17; in fact these two algorithms are special cases of the algorithms we develop here.

4.2.1 Basic idea

Knot insertion is exactly what the name suggests: extension of a given knot vector by adding new knots. Let us first define precisely what we mean by knot insertion, or knot refinement as it is also called.

Definition 4.1. A knot vector $\mathbf{t}$ is said to be a refinement of a knot vector $\boldsymbol{\tau}$ if any real number occurs at least as many times in $\mathbf{t}$ as in $\boldsymbol{\tau}$.

Note that if $\mathbf{t}$ is a refinement of $\boldsymbol{\tau}$ then $\boldsymbol{\tau}$ is a subsequence of $\mathbf{t}$, and this we will write $\boldsymbol{\tau} \subseteq \mathbf{t}$ even though knot vectors are sequences and not sets. The term knot insertion is used because in most situations the knot vector $\boldsymbol{\tau}$ is given and $\mathbf{t}$ is obtained by ‘inserting’ knots into $\boldsymbol{\tau}$. A simple example of a knot vector and a refinement is given by

$$\boldsymbol{\tau} = (0, 0, 0, 3, 4, 5, 5, 6, 6, 6) \quad \text{and} \quad \mathbf{t} = (0, 0, 0, 2, 2, 3, 3, 4, 5, 5, 6, 6, 6).$$

Here two knots have been inserted at 2, one at 3 and one at 5.

With some polynomial degree d given, we can associate the spline spaces $\mathbb{S}_{d,\boldsymbol{\tau}}$ and $\mathbb{S}_{d,\mathbf{t}}$ with the two knot vectors $\boldsymbol{\tau}$ and $\mathbf{t}$. When $\boldsymbol{\tau}$ is a subsequence of $\mathbf{t}$, the two spline spaces are also related.

Lemma 4.2. Let d be a positive integer and let $\boldsymbol{\tau}$ be a knot vector with at least $d + 2$ knots. If $\mathbf{t}$ is a knot vector which contains $\boldsymbol{\tau}$ as a subsequence then $\mathbb{S}_{d,\boldsymbol{\tau}} \subseteq \mathbb{S}_{d,\mathbf{t}}$.

Proof. Suppose first that both $\boldsymbol{\tau}$ and $\mathbf{t}$ are $d + 1$ -regular knot vectors with common knots at the ends. By the Curry-Schoenberg theorem (Theorem 3.25) we know that $\mathbb{S}_{d,\mathbf{t}}$ contains all splines with smoothness prescribed by the knot vector $\mathbf{t}$. Since all knots occur at least as many times in $\mathbf{t}$ as in $\boldsymbol{\tau}$, we see that at any knot, a spline f in $\mathbb{S}_{d,\boldsymbol{\tau}}$ is at least as smooth as required for a spline in $\mathbb{S}_{d,\mathbf{t}}$. But then $f \in \mathbb{S}_{d,\boldsymbol{\tau}}$ and $\mathbb{S}_{d,\boldsymbol{\tau}} \subseteq \mathbb{S}_{d,\mathbf{t}}$.

A proof in the general case where $\boldsymbol{\tau}$ and $\mathbf{t}$ are not $d + 1$ -regular with common knots at the ends, is outlined in exercise 5. ■

Suppose that $f = \sum_{j=1}^n c_j B_{j,d,\boldsymbol{\tau}}$ is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$ with B-spline coefficients $\mathbf{c} = (c_j)$. If $\boldsymbol{\tau}$ is a subsequence of $\mathbf{t}$, we know from Lemma 4.2 that $\mathbb{S}_{d,\boldsymbol{\tau}}$ is a subspace of $\mathbb{S}_{d,\mathbf{t}}$ so f must also lie in $\mathbb{S}_{d,\mathbf{t}}$. Hence there exist real numbers $\mathbf{b} = (b_i)$ with the property that $f = \sum_{i=1}^m b_i B_{i,d,\mathbf{t}}$, i.e., the vector $\mathbf{b}$ contains the B-spline coefficients of f in $\mathbb{S}_{d,\mathbf{t}}$. Knot insertion is therefore nothing but a change of basis from the B-spline basis in $\mathbb{S}_{d,\boldsymbol{\tau}}$ to the B-spline basis in $\mathbb{S}_{d,\mathbf{t}}$.

Since $\mathbb{S}_{d,\tau} \subseteq \mathbb{S}_{d,t}$, all the B-splines in $\mathbb{S}_{d,\tau}$ are also in $\mathbb{S}_{d,t}$. We can therefore write

$$B_{j,d,\tau} = \sum_{i=1}^m \alpha_{j,d}(i) B_{i,d,t}, \quad j = 1, 2, \dots, n, \quad (4.1)$$

for certain numbers $\alpha_{j,d}(i)$. In the matrix form we have used earlier this can be written

$$\mathbf{B}_{\tau}^T = \mathbf{B}_t^T \mathbf{A}, \quad (4.2)$$

where $\mathbf{B}_{\tau}^T = (B_{1,d,\tau}, \dots, B_{n,d,\tau})$ and $\mathbf{B}_t^T = (B_{1,d,t}, \dots, B_{m,d,t})$ are row vectors, and the $m \times n$ -matrix $\mathbf{A} = (\alpha_{j,d}(i))$ is the basis transformation matrix. Using this notation and remembering equation (4.2), we can write f in the form

$$f = \mathbf{B}_t^T \mathbf{b} = \mathbf{B}_{\tau}^T \mathbf{c} = \mathbf{B}_t^T \mathbf{A} \mathbf{c}.$$

The linear independence of the B-splines in $\mathbb{S}_{d,t}$ therefore means that $\mathbf{b}$ and $\mathbf{c}$ must be related by

$$\mathbf{b} = \mathbf{A} \mathbf{c}, \quad \text{or} \quad b_i = \sum_{j=1}^n a_{i,j} c_j \quad \text{for } i = 1, 2, \dots, m. \quad (4.3)$$

The basis transformation $\mathbf{A}$ is called *the knot insertion matrix of degree d from τ to t* and we will use the notation $\alpha_{j,d}(i) = \alpha_{j,d,\tau,t}(i)$ for its entries. The discrete function $\alpha_{j,d}$ has many properties similar to those of $B_{j,d}$, and it is therefore called a *discrete B-spline on t with knots τ* .

To illustrate these ideas, let us consider a couple of simple examples of knot insertion for splines.

Example 4.3. Let us determine the transformation matrix $\mathbf{A}$ for splines with $d = 0$, when the coarse knot vector is given by $\tau = (0, 1, 2)$, and the refined knot vector is $t = (0, 1/2, 1, 3/2, 2) = (t_i)_{i=1}^5$. In this case

$$\mathbb{S}_{d,\tau} = \text{span}\{B_{1,0,\tau}, B_{2,0,\tau}\} \quad \text{and} \quad \mathbb{S}_{d,t} = \text{span}\{B_{1,0,t}, B_{2,0,t}, B_{3,0,t}, B_{4,0,t}\}.$$

We clearly have

$$B_{1,0,\tau} = B_{1,0,t} + B_{2,0,t}, \quad B_{2,0,\tau} = B_{3,0,t} + B_{4,0,t}.$$

This means that the knot insertion matrix in this case is given by

$$\mathbf{A} = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{pmatrix}.$$

Example 4.4. Let us also consider an example with linear splines. Let $d = 1$, and let τ and t be as in the preceding example. In this case $\dim \mathbb{S}_{d,\tau} = 1$ and we find that

$$B(x | 0, 1, 2) = \frac{1}{2} B(x | 0, 1/2, 1) + B(x | 1/2, 1, 3/2) + \frac{1}{2} B(x | 1, 3/2, 2).$$

The situation is shown in Figure 4.4. The linear B-spline on τ is a weighted sum of the three B-splines (dashed) on t . The knot insertion matrix $\mathbf{A}$ is therefore the 3×1 -matrix, or row vector, given by

$$\mathbf{A} = \begin{pmatrix} 1/2 \\ 1 \\ 1/2 \end{pmatrix}.$$

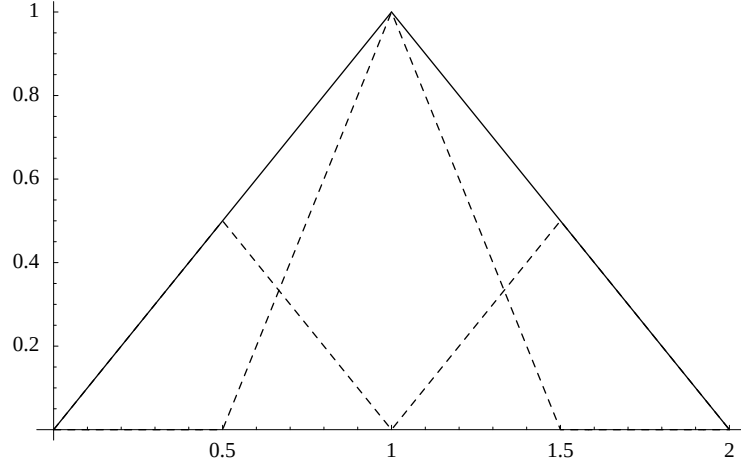


Figure 4.4. Refining a linear B-spline.

4.2.2 Conversion between B-spline polynomials

We would obviously like to compute the B-spline coefficients on a refined knot vector by computer and therefore need a detailed and efficient algorithm. For this we need to study the matrix A in (4.2) and (4.3) in some more detail. We are going to use the strategy of considering what happens on individual knot intervals, which has proved successful in earlier chapters.

It will be helpful to specialise the linear algebra that led us to the two relations (4.2) and (4.3) to the space π_d of polynomials of degree d . Suppose we have two bases $\mathbf{p}^T = (p_0, \dots, p_d)$ and $\mathbf{q}^T = (q_0, \dots, q_d)$ of π_d . We then know that there exists a nonsingular matrix M of dimension $d+1$ such that

$$\mathbf{p}^T = \mathbf{q}^T M. \quad (4.4)$$

Let f be a polynomial of degree d with coefficients $\mathbf{b}$ relative to $\mathbf{p}$ and $\mathbf{c}$ relative to $\mathbf{q}$. Since $f = \mathbf{p}^T \mathbf{b} = \mathbf{q}^T M \mathbf{b} = \mathbf{q}^T \mathbf{c}$ it follows that

$$\mathbf{c} = M \mathbf{b}. \quad (4.5)$$

Conversely, it is not difficult to see that if the representations of any polynomial in two bases are related as in (4.5), then the bases must be related as in (4.4).

We are specially interested in polynomial bases obtained from B-splines. If $\mathbf{u} = (u_i)_{i=1}^{2d+2}$ is a knot vector with $u_{d+1} < u_{d+2}$, the theory in Chapter 3 shows that the corresponding B-splines form a basis for π_d on the interval $[u_{d+1}, u_{d+2}]$. On this interval the B-splines reduce to polynomials and therefore correspond to a polynomial basis $\mathbf{B}_u$. And as all polynomials, these basis polynomials are defined on the whole real line (they can be computed for any x by always using $\mu = d+1$ in the spline evaluation algorithms in Chapter 2).

Suppose now that we have another knot vector $\mathbf{v} = (v_i)_{i=1}^{2d+2}$ with no relation to $\mathbf{u}$. This will give rise to a similar polynomial basis $\mathbf{B}_v$, and these two bases must be related by some matrix $M_{u,v}$,

$$\mathbf{B}_v^T = \mathbf{B}_u^T M_{u,v}.$$

We want to find a formula for $\mathbf{M}_{\mathbf{u},\mathbf{v}}$ and to do this we consider the representation of the polynomial $f(x) = (y - x)^d$ where y is any real number. We know from Marsden's identity (Theorem 3.5) that the coefficients of f relative to the basis $\mathbf{B}_{\mathbf{u}}$ are the dual polynomials $\boldsymbol{\rho}_{\mathbf{u}} = \{\rho_{i,\mathbf{u}}\}_{i=1}^{d+1}$ where $\rho_{i,\mathbf{u}}(y)$ is given by

$$\rho_{i,\mathbf{u}}(y) = (y - u_{i+1}) \cdots (y - u_{i+d}).$$

The B-spline coefficients of f relative to $\mathbf{B}_{\mathbf{v}}$ are given similarly by $\boldsymbol{\rho}_{\mathbf{v}}$, and the general discussion above shows that the two sets of coefficients must be related by the matrix $\mathbf{M}_{\mathbf{u},\mathbf{v}}$, as in (4.5),

$$\boldsymbol{\rho}_{\mathbf{v}}(y) = \mathbf{M}_{\mathbf{u},\mathbf{v}} \boldsymbol{\rho}_{\mathbf{u}}(y).$$

The i th component of this equation is

$$\rho_{i,\mathbf{v}}(y) = (\mathbf{M}_{\mathbf{u},\mathbf{v}})_i \boldsymbol{\rho}_{\mathbf{u}}(y).$$

On the other hand we also know from Corollary 3.3 that

$$\rho_{i,\mathbf{v}}(y) = (y - v_{i+1}) \cdots (y - v_{i+d}) = \mathbf{R}_1(v_{i+1}) \mathbf{R}_2(v_{i+2}) \cdots \mathbf{R}_d(v_{i+d}) \boldsymbol{\rho}_{\mathbf{u}}(y),$$

where the matrices $\mathbf{R}_1, \dots, \mathbf{R}_d$ are the bidiagonal B-spline matrices given in Theorem 2.14,

$$\mathbf{R}_k(x) = \mathbf{R}_{k,\mathbf{u}}^{d+1}(x) = \begin{pmatrix} \frac{u_{d+2} - x}{u_{d+2} - u_{d+2-k}} & \frac{x - u_{d+2-k}}{u_{d+2} - u_{d+2-k}} & & \\ & & \ddots & \\ & & & \frac{u_{d+1+k} - x}{u_{d+1+k} - u_{d+1}} & \frac{x - u_{d+1}}{u_{d+1+k} - u_{d+1}} \end{pmatrix}.$$

Since the dual polynomials $\{\rho_{i,\mathbf{u}}\}_{i=1}^{d+1}$ are linearly independent we therefore have

$$(\mathbf{M}_{\mathbf{u},\mathbf{v}})_i = \mathbf{R}_1(v_{i+1}) \mathbf{R}_2(v_{i+2}) \cdots \mathbf{R}_d(v_{i+d}).$$

Let us sum up our findings so far.

Proposition 4.5. *Let $\mathbf{u} = (u_i)_{i=1}^{2d+2}$ and $\mathbf{v} = (v_i)_{i=1}^{2d+2}$ be two knot vectors with $u_{d+1} < u_{d+2}$ and $v_{d+1} < v_{d+2}$, and let $\mathbf{B}_{\mathbf{u}}$ and $\mathbf{B}_{\mathbf{v}}$ be the corresponding B-spline polynomials on the intervals $[u_{d+1}, u_{d+2}]$ and $[v_{d+1}, v_{d+2}]$ respectively. Then the two polynomial bases are related by*

$$\mathbf{B}_{\mathbf{v}}^T = \mathbf{B}_{\mathbf{u}}^T \mathbf{M}_{\mathbf{u},\mathbf{v}} \quad (4.6)$$

where $\mathbf{M}_{\mathbf{u},\mathbf{v}}$ is a square matrix of dimension $d + 1$ with rows given by

$$(\mathbf{M}_{\mathbf{u},\mathbf{v}})_i = \mathbf{R}_1(v_{i+1}) \mathbf{R}_2(v_{i+2}) \cdots \mathbf{R}_d(v_{i+d}) \quad (4.7)$$

for $i = 1, \dots, d + 1$. Here $\mathbf{R}_k(x) = \mathbf{R}_{k,\mathbf{u}}^{d+1}(x)$ for $k = 1, \dots, d$ are the B-spline matrices of the interval $[u_{d+1}, u_{d+2}]$ defined in Theorem 2.14.

Although the expression (4.7) is slightly more complicated than what we encountered when developing algorithms for computing the value of splines and B-splines, those algorithms can easily be adapted to computing the matrix $M_{u,v}$ or converting from the representation in terms of B_u to a representation in terms of B_v , see Algorithms 4.10 and 4.11 below. Note also that because of the symmetry in the construction, it is easy to find the inverse of the matrix $M_{u,v}$,

$$M_{u,v}^{-1} = M_{v,u},$$

i.e., just reverse the roles of u and v .

4.2.3 Formulas and algorithms for knot insertion

We have seen how we can find formulas for conversion between two polynomial pieces of two completely unrelated B-spline bases. Let us now apply this to the special situation of knot insertion.

Suppose as before that we have two knot vectors τ and t with $\tau \subseteq t$ and a spline function $f = \sum_j c_j B_{j,d,\tau} = \sum_i b_i B_{i,d,t}$ which lies in $\mathbb{S}_{d,\tau}$ and therefore also in $\mathbb{S}_{d,t}$. Recall from (4.1) and (4.2) that the two spaces are related by the basis transformation matrix A whose (i, j) -entry we denote $\alpha_{j,d}(i)$. In other words we have

$$b_i = \sum_{j=1}^n \alpha_{j,d}(i) c_j \quad (4.8)$$

for $i = 1, \dots, m$, and

$$B_{j,d,\tau} = \sum_{i=1}^m \alpha_{j,d}(i) B_{i,d,t} \quad (4.9)$$

for $j = 1, \dots, n$. An important observation here is that a B-spline will usually consist of several polynomial pieces and according to (4.9), all the pieces of a B-spline in $\mathbb{S}_{d,\tau}$ must be expressible as the same linear combination of the corresponding pieces of the B-splines in $\mathbb{S}_{d,t}$. An example should help to clarify this.

Example 4.6. Suppose that $d = 3$ and that the knot vector $\tau = (0, 0, 0, 0, 1, 4, 4, 4, 4)$ has been refined to $t = (0, 0, 0, 0, 1, 2, 3, 4, 4, 4, 4)$. In $\mathbb{S}_{3,\tau}$ we then have the five B-splines $\{B_{j,\tau}\}_{j=1}^5$ and in $\mathbb{S}_{3,t}$ we have seven B-splines $\{B_{i,t}\}_{i=1}^7$ (we have dropped the degree from the notation as it will remain fixed in this example). Relation (4.9) therefore becomes

$$B_{j,\tau} = \sum_{i=1}^7 \alpha_{j,d}(i) B_{i,t} \quad (4.10)$$

for $j = 1, \dots, 5$. What does this really mean? It does of course mean that the B-splines in $\mathbb{S}_{3,\tau}$ are linear combinations of the B-splines in $\mathbb{S}_{3,t}$. But a consequence of this is that each polynomial piece of $B_{j,\tau}$ can be written as a linear combination of the corresponding pieces of the B-splines in $\mathbb{S}_{3,t}$.

Let us be more specific. The interval of interest is $[0, 4]$ and a B-spline $B_{j,\tau}$ in $\mathbb{S}_{3,\tau}$ consists of two polynomial pieces within this interval, one piece on $[\tau_4, \tau_5] = [0, 1]$ which we denote $B_{j,\tau}^4$ and one piece on $[\tau_5, \tau_6] = [1, 4]$ which we denote $B_{j,\tau}^5$. Similarly, a B-spline $B_{i,t}$ in $\mathbb{S}_{3,t}$ consists of four polynomial pieces which we denote $B_{i,t}^4$, $B_{i,t}^5$, $B_{i,t}^6$ and $B_{i,t}^7$. With this notation, we can elaborate more on the meaning of relation (4.10).

If we restrict x to the interval $[0, 1]$ we can write (4.10) as

$$B_{j,\tau}^4 = \sum_{i=1}^4 \alpha_{j,d}(i) B_{i,t}^4$$

for $j = 1, \dots, 5$, since the other B-splines in $\mathbb{S}_{3,t}$ vanish on this interval. If we ignore $B_{5,\tau}$, this is just a relation between two polynomial bases on B-spline form for the interval $[\tau_4, \tau_5]$, so we can use Proposition 4.5 to determine the coefficients $(\alpha_j(i))_{i,j=1}^4$. We find that

$$\begin{pmatrix} \alpha_1(1) & \alpha_2(1) & \alpha_3(1) & \alpha_4(1) \\ \alpha_1(2) & \alpha_2(2) & \alpha_3(2) & \alpha_4(2) \\ \alpha_1(3) & \alpha_2(3) & \alpha_3(3) & \alpha_4(3) \\ \alpha_1(4) & \alpha_2(4) & \alpha_3(4) & \alpha_4(4) \end{pmatrix} = \begin{pmatrix} R_1^4(t_2)R_2^4(t_3)R_3^4(t_4) \\ R_1^4(t_3)R_2^4(t_4)R_3^4(t_5) \\ R_1^4(t_4)R_2^4(t_5)R_3^4(t_6) \\ R_1^4(t_5)R_2^4(t_6)R_3^4(t_7) \end{pmatrix}$$

where $R_k^4 = R_{k,\tau}^4(x)$ for $k = 1, 2, 3$ are B-spline matrices for the interval $[\tau_4, \tau_5]$. We can also determine $(\alpha_5(i))_{i=1}^4$ since $B_{5,\tau}^4$ is identically zero. In fact the linear independence of the polynomials $\{B_{i,\tau}^4\}_{i=1}^4$ on $[0, 1]$ means that $\alpha_5(i) = 0$ for $i = 1, 2, 3, 4$.

If we move to the right, the next subinterval of τ is $[\tau_5, \tau_6] = [1, 4]$ while the next subinterval of t is $[t_5, t_6] = [1, 2]$. On the smallest common subinterval $[1, 2]$ relation (4.10) reduces to

$$B_{j,\tau}^5 = \sum_{i=2}^5 \alpha_j(i) B_{i,t}^5$$

for $j = 1, \dots, 5$. Similarly to the previous subinterval we can conclude that $(\alpha_1(i))_{i=2}^5$ is zero since $B_{1,\tau}^5$ is identically zero on this interval. The remaining $\alpha_j(i)$ s involved in the sum can be determined from Proposition 4.5,

$$\begin{pmatrix} \alpha_2(2) & \alpha_3(2) & \alpha_4(2) & \alpha_5(2) \\ \alpha_2(3) & \alpha_3(3) & \alpha_4(3) & \alpha_5(3) \\ \alpha_2(4) & \alpha_3(4) & \alpha_4(4) & \alpha_5(4) \\ \alpha_2(5) & \alpha_3(5) & \alpha_4(5) & \alpha_5(5) \end{pmatrix} = \begin{pmatrix} R_1^5(t_3)R_2^5(t_4)R_3^5(t_5) \\ R_1^5(t_4)R_2^5(t_5)R_3^5(t_6) \\ R_1^5(t_5)R_2^5(t_6)R_3^5(t_7) \\ R_1^5(t_6)R_2^5(t_7)R_3^5(t_8) \end{pmatrix}.$$

If we move further to the right we come to the interval $[t_6, t_7] = [2, 3]$ which is a subinterval of $[\tau_5, \tau_6] = [1, 4]$. Relation (4.10) now becomes

$$B_{j,\tau}^5 = \sum_{i=3}^6 \alpha_j(i) B_{i,t}^6$$

for $j = 1, \dots, 5$. Again we can conclude that $\alpha_1(i) = 0$ for $i = 3, \dots, 6$ while

$$\begin{pmatrix} \alpha_2(3) & \alpha_3(3) & \alpha_4(3) & \alpha_5(3) \\ \alpha_2(4) & \alpha_3(4) & \alpha_4(4) & \alpha_5(4) \\ \alpha_2(5) & \alpha_3(5) & \alpha_4(5) & \alpha_5(5) \\ \alpha_2(6) & \alpha_3(6) & \alpha_4(6) & \alpha_5(6) \end{pmatrix} = \begin{pmatrix} R_1^5(t_4)R_2^5(t_5)R_3^5(t_6) \\ R_1^5(t_5)R_2^5(t_6)R_3^5(t_7) \\ R_1^5(t_6)R_2^5(t_7)R_3^5(t_8) \\ R_1^5(t_7)R_2^5(t_8)R_3^5(t_9) \end{pmatrix}.$$

We can move one more interval to the right, to $[t_7, t_8] = [3, 4]$, which is also a subinterval of $[\tau_5, \tau_6] = [1, 4]$. On this interval we can conclude that $\alpha_1(i) = 0$ for $i = 4, \dots, 7$ and determine the part of A given by $(\alpha_j(i))_{i=4, j=2}^{7,5}$.

Note that many of the entries in the matrix A are determined several times in this example simply because a B-spline consists of several polynomial pieces. This is not really a problem as we will get the same value (up to round-off) each time.

Example 4.6 makes an important point clear: Since (4.9) is a relation between piecewise polynomials, the number $\alpha_j(i)$ must be the coefficient multiplying $B_{i,t}$ in the representation of $B_{j,\tau}$, irrespective of which polynomial piece we consider. Therefore, by considering relation (4.9) as a relation between polynomials on different intervals we get several possibilities for determining most entries in the matrix A . This leaves us with the question of which polynomial pieces we should use to determine a certain entry in A . Theorem 4.7 uses a standard choice, but it is worth remembering that other choices are possible.

For simplicity we will make the assumption that $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ and $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ are both $d+1$ -regular knot vectors with $d+1$ common knots at the two ends. Exercise 6 shows that this causes no loss of generality. The technique in Example 4.6 works in general and can be used to obtain an explicit formula for the knot insertion matrix $\mathbf{A}$.

Theorem 4.7. *Let the polynomial degree d be given, and let $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ and $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ be two $d+1$ -regular knot vectors with common knots at the ends and $\boldsymbol{\tau} \subseteq \boldsymbol{t}$. In row i of the knot insertion matrix $\mathbf{A}$ the entries are given by $\alpha_{j,d}(i) = 0$ for $j < \mu - d$ and $j > \mu$, where μ is determined by $\tau_\mu \leq t_i < \tau_{\mu+1}$ and*

$$\boldsymbol{\alpha}_d(i)^T = (\alpha_{\mu-d,d}(i), \dots, \alpha_{\mu,d}(i)) = \begin{cases} 1, & \text{if } d = 0, \\ \mathbf{R}_{1,\boldsymbol{\tau}}^\mu(t_{i+1}) \cdots \mathbf{R}_{d,\boldsymbol{\tau}}^\mu(t_{i+d}), & \text{if } d > 0, \end{cases} \quad (4.11)$$

and the matrix $\mathbf{R}_{k,\boldsymbol{\tau}}^\mu$ is defined in Theorem 2.14. If $f = \sum_j c_j B_{j,d,\boldsymbol{\tau}}$ is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$, with B-spline coefficients $\mathbf{b}$ in $\mathbb{S}_{d,\boldsymbol{t}}$, then b_i is given by

$$b_i = \sum_{j=\mu-d}^{\mu} \alpha_{j,d}(i) c_j = \mathbf{R}_{1,\boldsymbol{\tau}}^\mu(t_{i+1}) \cdots \mathbf{R}_{d,\boldsymbol{\tau}}^\mu(t_{i+d}) \mathbf{c}_d, \quad (4.12)$$

where $\mathbf{c}_d = (c_{\mu-d}, \dots, c_\mu)$.

Proof. We note that (4.12) follows from the general discussion earlier in this chapter so we focus on the proof of (4.11). For degree $d = 0$ this is easy so we concentrate on the general case. We fix the integer i and are going to show how row no. i of $\mathbf{A}$ can be determined. Row i consists of the numbers $(\alpha_j(i))_{j=1}^n$ where $\alpha_j(i)$ gives the coefficient of $B_{j,\boldsymbol{t}}$ in the linear combination of the B-splines in $\mathbb{S}_{d,\boldsymbol{t}}$ that make up $B_{j,\boldsymbol{\tau}}$, see (4.9). We will deduce (4.11) by considering different polynomial pieces of the B-splines that are involved. Let μ be as stated in the theorem, and let v be the largest integer such that $t_v = t_i$. We then have the two bases of B-spline polynomials,

$$\begin{aligned} \mathbf{B}_{\boldsymbol{\tau}}^\mu &= (B_{\mu-d,\boldsymbol{\tau}}^\mu, \dots, B_{\mu,d}^\mu)^T, \\ \mathbf{B}_{\boldsymbol{t}}^v &= (B_{v-d,\boldsymbol{t}}^v, \dots, B_{v,t}^v)^T. \end{aligned}$$

The first basis consists of the polynomial pieces of the nonzero B-splines in $\mathbb{S}_{d,\boldsymbol{\tau}}$ on the interval $[\tau_\mu, \tau_{\mu+1}]$ and the other consists of the polynomial pieces of the nonzero B-splines in $\mathbb{S}_{d,\boldsymbol{t}}$ on the interval $[t_v, t_{v+1}]$. Note that the definition of v means that $B_{i,t}^v$ is one of the B-spline polynomials in $\mathbf{B}_{\boldsymbol{t}}^v$. From Proposition 4.5 we know that these two bases are related by a $(d+1) \times (d+1)$ -matrix $\mathbf{M}_{\boldsymbol{\tau},\boldsymbol{t}}$. Each row of this matrix is associated with one of the B-spline polynomials in the basis $\mathbf{B}_{\boldsymbol{t}}^v$ and the row associated with $B_{i,t}^v$ is given by

$$\mathbf{R}_{1,\boldsymbol{\tau}}^\mu(t_{i+1}) \cdots \mathbf{R}_{d,\boldsymbol{\tau}}^\mu(t_{i+d}).$$

On other hand, we also know that the matrix $\mathbf{M}_{\boldsymbol{\tau},\boldsymbol{t}}$ is a submatrix of the knot insertion matrix $\mathbf{A}$,

$$\mathbf{M}_{\boldsymbol{\tau},\boldsymbol{t}} = (\alpha_j(\ell))_{j=\mu-d, \ell=v-d}^{\mu,v}$$

since the two bases $\mathbf{B}_\tau^\mu$ and $\mathbf{B}_t^\nu$ are part of the two B-spline bases for $\mathbb{S}_{d,\tau}$ and $\mathbb{S}_{d,t}$. In particular we have

$$(\alpha_{\mu-d}(i), \dots, \alpha_\mu(i)) = \mathbf{R}_{1,\tau}^\mu(t_{i+1}) \cdots \mathbf{R}_{d,\tau}^\mu(t_{i+d}).$$

What remains is to prove that the other entries in row i of $\mathbf{A}$ are zero. Suppose that $j < \mu - d$. By the support properties of B-splines we must then have $B_{j,\tau}(x) = B_{j,\tau}^\mu(x) = 0$ for $x \in [t_\nu, t_{\nu+1}]$. When x varies in this interval we have

$$0 = B_{j,\tau}^\mu(x) = \sum_{\ell=\nu-d}^{\nu} \alpha_j(\ell) B_{\ell,t}^\nu(x).$$

From the linear independence of the B-spline polynomials $\{B_{\ell,t}^\nu\}_{\ell=\nu-d}^{\nu}$ we can then conclude that $\alpha_j(\ell) = 0$ for $\ell = \nu - d, \dots, \nu$. In particular we have $\alpha_j(i) = 0$. The case $j > \mu$ is similar. ■

Theorem 4.7 shows that the knot insertion matrix is banded: In any row, there are first some zeros, then some nonzero entries, and then more zeros. As we have already noted there are several possibilities when it comes to computing the nonzero entries since a B-spline consists of different polynomial pieces which are all transformed in the same way. In Theorem 4.7 we compute the nonzero entries in row i by considering the knot interval in $\mathbf{t}$ which has t_i as its left end and the knot interval in τ whose left end is closest to t_i . In general, there are many other possibilities. With i given, we could for example choose μ by requiring that $\tau_{\mu+d} \leq t_{i+d+1} < \tau_{\mu+d+1}$.

It should be noted that, in general, not all the $d+1$ entries of row i of $\mathbf{A}$ given by (4.11) will be nonzero. It is in fact quite easy to see that $\alpha_j(i)$ will only be nonzero if the whole support of $B_{j,t}$ is a subset of the support of $B_{j,\tau}$. More specifically, it can be shown that if there are r new knots among $t_{i+1}, \dots, t_{i+d}$ then there will be $r+1$ nonzero entries in row i of $\mathbf{A}$.

Note that if no new knots are inserted ($\tau = \mathbf{t}$) then the two sets of B-spline coefficients $\mathbf{c}$ and $\mathbf{b}$ are obviously the same. Equation (4.12) then shows that

$$\mathbf{c}_i = \mathbf{R}_{1,\tau}^\mu(\tau_{i+1}) \cdots \mathbf{R}_{d,\tau}^\mu(\tau_{i+d}) \mathbf{c}_d. \quad (4.13)$$

This simple observation will be useful later.

A couple of examples will illustrate the use of Theorem 4.7.

Example 4.8. We consider quadratic splines ($d = 2$) on the knot vector $\tau = (-1, -1, -1, 0, 1, 1, 1)$, and insert two new knots, at $-1/2$ and $1/2$ so $\mathbf{t} = (-1, -1, -1, -1/2, 0, 1/2, 1, 1, 1)$. We note that $\tau_3 \leq t_i < \tau_4$ for $1 \leq i \leq 4$ so the first three entries of the first four rows of the 6×4 knot insertion matrix $\mathbf{A}$ are given by

$$\alpha_2(i) = \mathbf{R}_{1,\tau}^3(t_{i+1}) \mathbf{R}_{2,\tau}^3(t_{i+2})$$

for $i = 1, \dots, 4$. Since

$$\mathbf{R}_{1,\tau}^3(x) = \begin{pmatrix} -x & 1+x \end{pmatrix}, \quad \mathbf{R}_{2,\tau}^3(x) = \begin{pmatrix} -x & 1+x & 0 \\ 0 & (1-x)/2 & (1+x)/2 \end{pmatrix},$$

we have from (4.11)

$$\alpha_2(i) = \frac{1}{2} (2t_{i+1}t_{i+2}, \quad 1 - t_{i+1} - t_{i+2} - 3t_{i+1}t_{i+2}, \quad (1+t_{i+1})(1+t_{i+2})).$$

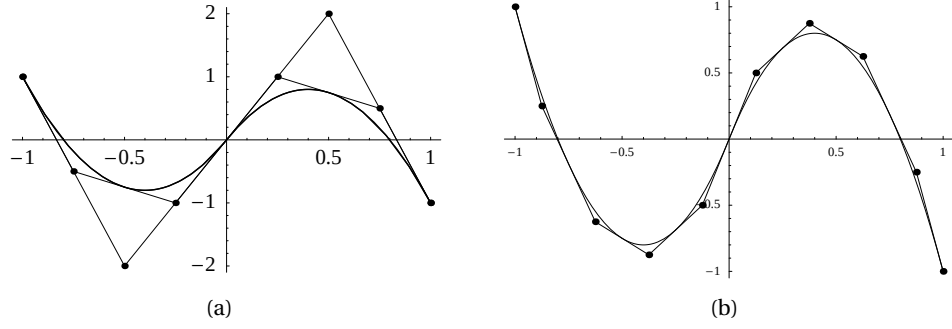


Figure 4.5. A quadratic spline together with its control polygon relative to a coarse and a finer knot vector (a), and the same spline as in (a) with its control polygon relative to an even more refined knot vector (b).

Inserting the correct values for t_{i+1} and t_{i+2} and adding one zero at the end of each row, we find that the first four rows of A are given by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 1/4 & 3/4 & 0 \end{pmatrix}.$$

To determine the remaining two rows of A we have to move to the interval $[\tau_4, \tau_5) = [0, 1)$. Here we have

$$R_{1,\tau}^4(x) = (1-x \quad x) \quad R_{2,\tau}^4(x) = \begin{pmatrix} (1-x)/2 & (1+x)/2 & 0 \\ 0 & 1-x & x \end{pmatrix},$$

so

$$a_2(i) = R_{1,\tau}^4(t_{i+1})R_{2,\tau}^4(t_{i+2}) = \frac{1}{2}((1-t_{i+1})(1-t_{i+2}), \quad 1+t_{i+1}+t_{i+2}-3t_{i+1}t_{i+2}, \quad 2t_{i+1}t_{i+2}).$$

Evaluating this for $i = 5, 6$ and inserting one zero as the first entry, we obtain the last two rows as

$$\begin{pmatrix} 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

To see visually the effect of knot insertion, let $f = B_{1,2,\tau} - 2B_{2,2,\tau} + 2B_{3,2,\tau} - B_{4,2,\tau}$ be a spline in $\mathbb{S}_{d,\tau}$ with B-spline coefficients $\mathbf{c} = (1, -2, 2, -1)^T$. Its coefficients $\mathbf{b} = (b_i)_{i=1}^6$ are then given by

$$\mathbf{b} = A\mathbf{c} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1/2 \\ -1 \\ 1 \\ 1/2 \\ -1 \end{pmatrix}.$$

Figure 4.5 (a) shows a plot of f together with its control polygons relative to τ and $\mathbf{t}$. We note that the control polygon relative to $\mathbf{t}$ is much closer to f and that both control polygons give a rough estimate of f .

The knot insertion process can be continued. If we insert one new knot halfway between each old knot in $\mathbf{t}$, we obtain the new knot vector

$$\mathbf{t}^1 = (-1, -1, -1, -3/4, -1/2, -1/4, 0, 1/4, 1/2, 3/4, 1, 1, 1).$$

A plot of f and its control polygon relative to this knot vector is shown in Figure 4.5 (b).

Example 4.9. Let us again consider quadratic splines on a uniform knot vector with multiple knots at the ends,

$$\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+3} = (3, 3, 3, 4, 5, 6, \dots, n, n+1, n+1, n+1),$$

and form $\mathbf{t}$ by inserting one knot half way between each pair of old knots,

$$\mathbf{t} = (t_i)_{i=1}^{2n+1} = (3, 3, 3, 7/2, 4, 9/2, 5, \dots, n, (2n+1)/2, n+1, n+1, n+1).$$

Since $\dim \mathbb{S}_{d,\boldsymbol{\tau}} = n$ and $\dim \mathbb{S}_{d,\mathbf{t}} = 2n-2$, the knot insertion matrix $\mathbf{A}$ is now a $(2n-2) \times n$ matrix. As in Example 4.8 we find that the first three columns of the first four rows of $\mathbf{A}$ are

$$\begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1/2 & 0 \\ 0 & 3/4 & 1/4 \\ 0 & 1/4 & 3/4 \end{pmatrix}.$$

To determine rows $2\mu-3$ and $2\mu-2$ with $4 \leq \mu \leq n-1$, we need the matrices $\mathbf{R}_{1,\boldsymbol{\tau}}^\mu$ and $\mathbf{R}_{2,\boldsymbol{\tau}}^\mu$ which are given by

$$\mathbf{R}_{1,\boldsymbol{\tau}}^\mu(x) = (\mu+1-x \quad x-\mu), \quad \mathbf{R}_{2,\boldsymbol{\tau}}^\mu(x) = \begin{pmatrix} (\mu+1-x)/2 & (x+1-\mu)/2 & 0 \\ 0 & (\mu+2-x)/2 & (x-\mu)/2 \end{pmatrix}.$$

Observe that $\tau_i = i$ for $i = 3, \dots, n+1$ and $t_i = (i+3)/2$ for $i = 3, \dots, 2n-1$. Entries $\mu-2$, $\mu-1$ and μ of row $2\mu-3$ are therefore given by

$$\mathbf{R}_{1,\boldsymbol{\tau}}^\mu(t_{2\mu-2})\mathbf{R}_{2,\boldsymbol{\tau}}^\mu(t_{2\mu-1}) = \mathbf{R}_{1,\boldsymbol{\tau}}^\mu(\mu+1/2)\mathbf{R}_{2,\boldsymbol{\tau}}^\mu(\mu+1) = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \end{pmatrix} = \begin{pmatrix} 0 & 3/4 & 1/4 \end{pmatrix}.$$

Similarly, entries $\mu-3$, $\mu-2$ and μ of row $2\mu-2$ are given by

$$\mathbf{R}_{1,\boldsymbol{\tau}}^\mu(t_{2\mu-1})\mathbf{R}_{2,\boldsymbol{\tau}}^\mu(t_{2\mu}) = \mathbf{R}_{1,\boldsymbol{\tau}}^\mu(\mu+1)\mathbf{R}_{2,\boldsymbol{\tau}}^\mu(\mu+3/2) = \begin{pmatrix} 0 & 1 \end{pmatrix} \begin{pmatrix} -1/4 & 5/4 & 0 \\ 0 & 1/4 & 3/4 \end{pmatrix} = \begin{pmatrix} 0 & 1/4 & 3/4 \end{pmatrix}.$$

Finally, we find as in Example 4.8 that the last three entries of the last two rows are

$$\begin{pmatrix} 0 & 1/2 & 1/2 \\ 0 & 0 & 1 \end{pmatrix}.$$

The complete knot insertion matrix is therefore

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & \dots & 0 & 0 & 0 \\ 0 & 3/4 & 1/4 & 0 & \dots & 0 & 0 & 0 \\ 0 & 1/4 & 3/4 & 0 & \dots & 0 & 0 & 0 \\ 0 & 0 & 3/4 & 1/4 & \dots & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 3/4 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & 3/4 & 1/4 & 0 \\ 0 & 0 & 0 & 0 & \dots & 1/4 & 3/4 & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix}.$$

The formula for $\alpha_d(i)$ shows very clearly the close relationship between B-splines and discrete B-splines, and it will come as no surprise that $\alpha_{j,d}(i)$ satisfies a recurrence relation

similar to that of B-splines, see Definition 2.1. The recurrence for $\alpha_{j,d}(i)$ is obtained by setting $x = t_{i+d}$ in the recurrence (2.1) for $B_{j,d}(x)$,

$$\alpha_{j,d}(i) = \frac{t_{i+d} - \tau_j}{\tau_{j+d} - \tau_j} \alpha_{j,d-1}(i) + \frac{\tau_{j+1+d} - t_{i+d}}{\tau_{j+1+d} - \tau_{j+1}} \alpha_{j+1,d-1}(i), \quad (4.14)$$

starting with $\alpha_{j,0}(i) = B_{j,0}(t_i)$.

The two evaluation algorithms for splines, Algorithms 3.17 and 3.18, can be adapted to knot insertion quite easily. For historical reasons these algorithms are usually referred to as *the Oslo algorithms*.

Algorithm 4.10 (Oslo-Algorithm 1). *Let the polynomial degree d , and the two $d+1$ -regular knot vectors $\tau = (\tau_j)_{j=1}^{n+d+1}$ and $t = (t_i)_{i=1}^{m+d+1}$ with common knots at the ends be given. To compute the $m \times n$ knot insertion matrix $A = (\alpha_{j,d}(i))_{i,j=1}^{m,n}$ from τ to t perform the following steps:*

1. For $i = 1, \dots, m$.

1.1 Determine μ such that $\tau_\mu \leq t_i < \tau_{\mu+1}$.

1.2 Compute entries $\mu-d, \dots, \mu$ of row i by evaluating

$$\alpha_d(i)^T = (\alpha_{\mu-d,d}(i), \dots, \alpha_{\mu,d}(i))^T = \begin{cases} 1, & \text{if } d = 0. \\ R_1(t_{i+1}) \cdots R_d(t_{i+d}), & \text{if } d > 0. \end{cases}$$

All other entries in row i are zero.

An algorithm for converting a spline from a B-spline representation in $\mathbb{S}_{d,\tau}$ to $\mathbb{S}_{d,t}$ is as follows.

Algorithm 4.11 (Oslo-Algorithm 2). *Let the polynomial degree d , and the two $d+1$ -regular knot vectors $\tau = (\tau_j)_{j=1}^{n+d+1}$ and $t = (t_i)_{i=1}^{m+d+1}$ with common knots at the ends be given together with the spline f in $\mathbb{S}_{d,\tau}$ with B-spline coefficients $c = (c_j)_{j=1}^n$. To compute the B-spline coefficients $b = (b_i)_{i=1}^m$ of f in $\mathbb{S}_{d,t}$ perform the following steps:*

1. For $i = 1, \dots, m$.

1.1 Determine μ such that $\tau_\mu \leq t_i < \tau_{\mu+1}$.

1.2 Set $c_d = (c_j)_{j=\mu-d}^\mu$ and compute b_i by evaluating

$$b_i = \begin{cases} c_\mu, & \text{if } d = 0. \\ R_1(t_{i+1}) \cdots R_d(t_{i+d}) c_d, & \text{if } d > 0. \end{cases}$$

4.3 B-spline coefficients as functions of the knots

Knot insertion allows us to represent the same spline function on different knot vectors. In fact, any spline function can be given any real numbers as knots, as long as we also include the original knots. It therefore makes sense to consider the B-spline coefficients as functions

of the knots, and we shall see that this point of view allows us to characterise the B-spline coefficients completely by three simple properties.

Initially, we assume that the spline $f = \sum_{j=1}^n c_j B_{j,d,\tau}$ is a polynomial represented on a $d+1$ -extended knot vector τ . On the knot interval $[\tau_\mu, \tau_{\mu+1})$ we know that f can be written as

$$f(x) = \mathbf{R}_1(x) \cdots \mathbf{R}_d(x) \mathbf{c}_d, \quad (4.15)$$

where $\mathbf{c}_d = (c_{\mu-d}, \dots, c_\mu)^T$, see Section 2.4. Since f is assumed to be a polynomial this representation is valid for all real numbers x , although when x is outside $[\tau_\mu, \tau_{\mu+1})$ it is no longer a true B-spline representation.

Consider the function

$$F(x_1, \dots, x_d) = \mathbf{R}_1(x_1) \cdots \mathbf{R}_d(x_d) \mathbf{c}_d. \quad (4.16)$$

We recognise the right-hand side of this expression from equation (4.12) in Theorem 4.7: If we have a knot vector that includes the knots $(x_0, x_1, \dots, x_d, x_{d+1})$, then $F(x_1, \dots, x_d)$ gives the B-spline coefficient that multiplies the B-spline $B(x | x_0, \dots, x_{d+1})$ in the representation of the polynomial f on the knot vector $\mathbf{x}$. When f is a polynomial, it turns out that the function F is completely independent of the knot vector τ that underlie the definition of the $\mathbf{R}$ -matrices in (4.16). The function F is referred to as the *blossom* of f , and the whole theory of splines can be built from properties of this function.

4.3.1 The blossom

In this subsection we develop some of the properties of the blossom. We will do this in an abstract fashion, by starting with a formal definition of the blossom. In the next subsection we will then show that the function F in (4.16) satisfies this definition.

Definition 4.12. *A function on the form $f(x) = ax$, where a is a real number, is called a linear function. A function on the form $f(x) = ax + b$ with a and b real constants is called an affine function. A function of d variables $f(x_1, \dots, x_d)$ is said to be affine if it is affine viewed as a function of each x_i for $i = 1, \dots, d$, with the other variables fixed. A symmetric affine function is an affine function that is not altered when the order of the variables is changed.*

It is common to say that a polynomial $p(x) = a + bx$ of degree one is a linear polynomial, even when a is nonzero. According to Definition 4.12 such a polynomial is an affine polynomial, and this (algebraic) terminology will be used in the present section. Outside this section however, we will use the term linear polynomial.

For a linear function of one variable we have

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y), \quad x, y \in \mathbb{R} \quad (4.17)$$

for all real numbers α and β , while for an affine function f with $b \neq 0$ equation (4.17) only holds if $\alpha + \beta = 1$. This is in fact a complete characterisation of affine functions: If (4.17) holds with $\alpha + \beta = 1$, then f is affine, see exercise 9.

A general affine function of 2 variables is given by

$$\begin{aligned} f(x_1, x_2) &= ax_2 + b = (a_2x_1 + b_2)x_2 + a_1x_1 + b_1 \\ &= c_0 + c_1x_1 + c_2x_2 + c_{1,2}x_1x_2. \end{aligned} \quad (4.18)$$

Similarly, an affine function of three variables is a function on the form

$$f(x_1, x_2, x_3) = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_3 + c_{1,2} x_1 x_2 + c_{1,3} x_1 x_3 + c_{2,3} x_2 x_3 + c_{1,2,3} x_1 x_2 x_3.$$

In general, an affine function can be written as a linear combination of 2^d terms. This follows by induction as in (4.18) where we passed from one argument to two.

A symmetric and affine function satisfies the equation

$$f(x_1, x_2, \dots, x_d) = f(x_{\pi_1}, x_{\pi_2}, \dots, x_{\pi_d}),$$

for any permutation $(\pi_1, \pi_2, \dots, \pi_d)$ of the numbers $1, 2, \dots, d$. We leave it as an exercise to show that symmetric, affine functions of two and three variables can be written in the form

$$\begin{aligned} f(x_1, x_2) &= a_0 + a_1(x_1 + x_2) + a_2 x_1 x_2, \\ f(x_1, x_2, x_3) &= a_0 + a_1(x_1 + x_2 + x_3) + a_2(x_1 x_2 + x_1 x_3 + x_2 x_3) + a_3 x_1 x_2 x_3. \end{aligned}$$

We are now ready to give the definition of the blossom of a polynomial.

Definition 4.13. Let p be a polynomial of degree at most d . The blossom $\mathcal{B}[p](x_1, \dots, x_d)$ of p is a function of d variables with the properties:

1. *Symmetry.* The blossom is a symmetric function of its arguments,

$$\mathcal{B}[p](x_1, \dots, x_d) = \mathcal{B}[p](x_{\pi_1}, \dots, x_{\pi_d})$$

for any permutation $\pi_1, \dots, \pi_d$ of the integers $1, \dots, d$.

2. *Affine.* The blossom is affine in each of its variables,

$$\mathcal{B}[p](\dots, \alpha x + \beta y, \dots) = \alpha \mathcal{B}[p](\dots, x, \dots) + \beta \mathcal{B}[p](\dots, y, \dots)$$

whenever $\alpha + \beta = 1$.

3. *Diagonal property.* The blossom agrees with p on the diagonal,

$$\mathcal{B}[p](x, \dots, x) = p(x)$$

for all real numbers x .

The blossom of a polynomial exists and is unique.

Theorem 4.14. Each polynomial p of degree d has a unique blossom $\mathcal{B}[p](x_1, \dots, x_d)$. The blossom acts linearly on p , i.e., if p_1 and p_2 are two polynomials and c_1 and c_2 are two real constants then

$$\mathcal{B}[c_1 p_1 + c_2 p_2](x_1, \dots, x_d) = c_1 \mathcal{B}[p_1](x_1, \dots, x_d) + c_2 \mathcal{B}[p_2](x_1, \dots, x_d). \quad (4.19)$$

Proof. The proof of uniqueness follows along the lines sketched at the beginning of this section for small d . Start with a general affine function F of d variables

$$F(x_1, \dots, x_d) = c_0 + \sum_{j=1}^d \sum_{1 \leq i_1 < \dots < i_j \leq d} c_{i_1, \dots, i_j} x_{i_1} \cdots x_{i_j}.$$

Symmetry forces all the coefficients multiplying terms of the same degree to be identical. To see this we note first that

$$F(1, 0, \dots, 0) = c_0 + c_1 = F(0, \dots, 1, \dots, 0) = c_0 + c_i$$

for all i with $1 \leq i \leq d$. Hence we have $c_1 = \dots = c_d$. To prove that the terms of degree j all have the same coefficients we use induction and set j of the variables to 1 and the rest to 0. By the induction hypothesis we know that all the terms of degree less than j are symmetric; denote the contribution from these terms by p_{j-1} . Symmetry then gives

$$p_{j-1} + c_{1,2,\dots,j} = p_{j-1} + c_{1,2,\dots,j-1,j+1} = \dots = p_{j-1} + c_{d-j+1,\dots,d}.$$

From this we conclude that all the coefficients multiplying terms of degree j must be equal. We can therefore write F as

$$F(x_1, \dots, x_d) = a_0 + \sum_{j=1}^d a_j \sum_{1 \leq i_1 < \dots < i_j \leq d} x_{i_1} \cdots x_{i_j}, \quad (4.20)$$

for suitable constants $(a_j)_{j=0}^d$. From the diagonal property $F(x, \dots, x) = f(x)$ the coefficients $(a_j)_{j=0}^d$ are all uniquely determined (since $1, x, \dots, x^d$ is basis for π_d).

The linearity of the blossom with regards to p follows from its uniqueness: The right-hand side of (4.19) is affine in each of the x_i , it is symmetric, and it reduces to $c_1 p_1(x) + c_2 p_2(x)$ on the diagonal $x_1 = \dots = x_d = x$. ■

Recall that the elementary symmetric polynomials

$$s_j(x_1, \dots, x_d) = \left(\sum_{1 \leq i_1 < \dots < i_j \leq d} x_{i_1} x_{i_2} \cdots x_{i_j} \right) / \binom{d}{j}$$

that appear in (4.20) (apart from the binomial coefficient) agree with the B-spline coefficients of the polynomial powers,

$$\sigma_{k,d}^j = s_j(\tau_{k+1}, \dots, \tau_{k+d}),$$

see Corollary 3.6. In fact, the elementary symmetric polynomials are the blossoms of the powers,

$$\mathcal{B}[x^j](x_1, \dots, x_d) = s_j(x_1, \dots, x_d) \quad \text{for } j = 0, \dots, d.$$

They can also be defined by the relation

$$(x - x_1) \cdots (x - x_d) = \sum_{k=0}^d (-1)^{d-k} \binom{d}{k} s_{d-k}(x_1, \dots, x_d) x^k.$$

Note that the blossom depends on the degree of the polynomial in a nontrivial way. If we consider the polynomial $p(x) = x$ to be of degree one, then $\mathcal{B}[p](x_1) = x_1$. But we can also think of p as a polynomial of degree three (the cubic and quadratic terms are zero); then we obviously have $\mathcal{B}[p](x_1, x_2, x_3) = (x_1 + x_2 + x_3)/3$.

4.3.2 B-spline coefficients as blossoms

Earlier in this chapter we have come across a function that is both affine and symmetric. Suppose we have a knot vector τ for B-splines of degree d . On the interval $[\tau_\mu, \tau_{\mu+1})$ the only nonzero B-splines are $\mathbf{B}_d = (B_{\mu-d,d}, \dots, B_{\mu,d})^T$ which can be expressed in terms of matrices as

$$\mathbf{B}_d(x)^T = \mathbf{R}_1(x) \cdots \mathbf{R}_d(x).$$

If we consider the polynomial piece $f = \mathbf{B}_d^T \mathbf{c}_d$ with coefficients $\mathbf{c}_d = (c_{\mu-d}, \dots, c_\mu)^T$ we can define a function F of d variables by

$$F(x_1, \dots, x_d) = \mathbf{R}_1(x_1) \cdots \mathbf{R}_d(x_d) \mathbf{c}_d. \quad (4.21)$$

From equation (4.12) we recognise $F(x_1, \dots, x_d)$ as the coefficient multiplying a B-spline with knots $x_0, x_1, \dots, x_{d+1}$ in the representation of the polynomial f .

Equation (3.7) in Lemma 3.4 shows that F is a symmetric function. It is also affine in each of its variables. To verify this, we note that because of the symmetry it is sufficient to check that it is affine with respect to the first variable. Recall from Theorem 2.14 that $\mathbf{R}_1 = \mathbf{R}_{1,\tau}$ is given by

$$\mathbf{R}_1(x) = \begin{pmatrix} \frac{\tau_{\mu+1} - x}{\tau_{\mu+1} - \tau_\mu}, & \frac{x - \tau_\mu}{\tau_{\mu+1} - \tau_\mu} \end{pmatrix}$$

which is obviously an affine function of x .

The function F is also related to the polynomial f in that $F(x, \dots, x) = f(x)$. We have proved the following lemma.

Lemma 4.15. *Let $f = \sum_{j=\mu-d}^{\mu} c_j B_{j,d}$ be a polynomial represented in terms of the B-splines in $\mathbb{S}_{d,\tau}$ on the interval $[\tau_\mu, \tau_{\mu+1})$, with coefficients $\mathbf{c}_d = (c_{\mu-d}, \dots, c_\mu)^T$. Then the function*

$$F(x_1, \dots, x_d) = \mathbf{R}_1(x_1) \cdots \mathbf{R}_d(x_d) \mathbf{c}_d$$

is symmetric and affine, and agrees with f on the diagonal,

$$F(x, \dots, x) = f(x).$$

Lemma 4.15 and Theorem 4.14 show that the blossom of f is given by

$$\mathcal{B}[f](x_1, \dots, x_d) = \mathbf{R}_1(x_1) \cdots \mathbf{R}_d(x_d) \mathbf{c}_d.$$

Blossoming can be used to give explicit formulas for the B-spline coefficients of a spline.

Theorem 4.16. *Let $f = \sum_{j=1}^n c_j B_{j,d,\tau}$ be a spline on a $d+1$ -regular knot vector $\tau = (\tau_j)_{j=1}^{n+d+1}$. Its B-spline coefficients are then given by*

$$c_j = \mathcal{B}[f_k](\tau_{j+1}, \dots, \tau_{j+d}), \quad \text{for } k = j, j+1, \dots, j+d, \quad (4.22)$$

provided $\tau_k < \tau_{k+1}$. Here $f_k = f|_{(\tau_k, \tau_{k+1})}$ is the restriction of f to the interval (τ_k, τ_{k+1}) .

Proof. Let us first restrict x to the interval $[\tau_\mu, \tau_{\mu+1})$ and only consider one polynomial piece f_μ of f . From Lemma 4.15 we know that $\mathcal{B}[f_\mu](x_1, \dots, x_d) = \mathbf{R}_1(x_1) \cdots \mathbf{R}_d(x_d) \mathbf{c}_d$, where $\mathbf{c}_d = (c_j)_{j=\mu-d}^\mu$ are the B-spline coefficients of f active on the interval $[\tau_\mu, \tau_{\mu+1})$. From (4.13) we then obtain

$$c_j = \mathcal{B}[f_\mu](\tau_{j+1}, \dots, \tau_{j+d}) \quad (4.23)$$

which is (4.22) in this special situation.

To prove (4.22) in general, fix j and choose the integer k in the range $j \leq k \leq j+d$. We then have

$$f_k(x) = \sum_{i=k-d}^k c_i B_{i,d}(x), \quad (4.24)$$

By the choice of k we see that the sum in (4.24) includes the term $c_j B_{j,d}$. Equation (4.22) therefore follows by applying (4.23) to f_k . ■

The affine property allows us to perform one important operation with the blossom; we can change the arguments.

Lemma 4.17. *The blossom of p satisfies the relation*

$$\mathcal{B}[p](\dots, x, \dots) = \frac{b-x}{b-a} \mathcal{B}[p](\dots, a, \dots) + \frac{x-a}{b-a} \mathcal{B}[p](\dots, b, \dots) \quad (4.25)$$

for all real numbers a, b and x with $a \neq b$.

Proof. Observe that x can be written as an affine combination of a and b ,

$$x = \frac{b-x}{b-a} a + \frac{x-a}{b-a} b.$$

Equation (4.25) then follows from the affine property of the blossom. ■

The next result will be useful later.

Lemma 4.18. *Let $\mathcal{B}_x[p(x, y)]$ denote the blossom of p with respect to the variable x . Then*

$$\mathcal{B}_x[(y-x)^k](x_1, \dots, x_d) = \frac{k!}{d!} D^{d-k}((y-x_1) \cdots (y-x_d)), \quad (4.26)$$

for $k = 0, 1, \dots, d$, and

$$\mathcal{B}_x[(y_1-x) \cdots (y_\ell-x)](x_1, \dots, x_d) = \frac{(d-\ell)!}{d!} \sum_{1 \leq i_1, \dots, i_\ell \leq d} (y_1-x_{i_1}) \cdots (y_\ell-x_{i_\ell}), \quad (4.27)$$

where the sum is over all distinct choices $i_1, \dots, i_\ell$ of ℓ integers from the d integers $1, \dots, d$.

Proof. For $k = d$ equation (4.26) follows since the right-hand side is symmetric and affine in each of the variables x_i and it agrees with $(y-x)^d$ on the diagonal $x_1 = \cdots = x_d = x$. The general result is then obtained by differentiating both sides k times.

Equation (4.27) follows since the right-hand side is affine, symmetric and reduces to $(y_1-x) \cdots (y_\ell-x)$ when $x = x_1 = \cdots = x_d$, i.e., it must be the blossom of $(y-x)^d$. ■

4.4 Inserting one knot at a time

With blossoming we have a simple but powerful tool for determining the B-spline coefficients of splines. Here we will apply blossoming to develop an alternative knot insertion strategy. Instead of inserting all new knots simultaneously we can insert them sequentially. We insert one knot at a time and update the B-spline coefficients between each insertion. This leads to simple, explicit formulas.

Lemma 4.19 (Böhm's method). *Let $\tau = (\tau_j)_{j=1}^{n+d+1}$ be a given knot vector and let $t = (t_i)_{i=1}^{n+d+2}$ be the knot vector obtained by inserting a knot z in τ in the interval $[\tau_\mu, \tau_{\mu+1})$. If*

$$f = \sum_{j=1}^n c_j B_{j,d,\tau} = \sum_{i=1}^{n+1} b_i B_{i,d,t},$$

then $(b_i)_{i=1}^{n+1}$ can be expressed in terms of $(c_j)_{j=1}^n$ through the formulas

$$b_i = \begin{cases} c_i, & \text{if } 1 \leq i \leq \mu - d; \\ \frac{z - \tau_i}{\tau_{i+d} - \tau_i} c_i + \frac{\tau_{i+d} - z}{\tau_{i+d} - \tau_i} c_{i-1}, & \text{if } \mu - d + 1 \leq i \leq \mu; \\ c_{i-1}, & \text{if } \mu + 1 \leq i \leq n + 1. \end{cases} \quad (4.28)$$

Proof. Observe that for $j \leq \mu$ we have $\tau_j = t_j$. For $i \leq \mu - d$ and with k an integer such that $i \leq k \leq i + d$ it therefore follows from (4.22) that

$$b_i = \mathcal{B}[f_k](t_{i+1}, \dots, t_{i+d}) = \mathcal{B}[f_k](\tau_{i+1}, \dots, \tau_{i+d}) = c_i.$$

Similarly, we have $t_i = \tau_{i-1}$ for $i \geq \mu + 1$ so

$$b_i = \mathcal{B}[f_k](t_{i+1}, \dots, t_{i+d}) = \mathcal{B}[f_k](\tau_i, \dots, \tau_{i+d-1}) = c_{i-1}$$

for such values of i .

When i satisfies $\mu - d + 1 \leq i \leq \mu$ we note that z will appear in the sequence $(t_{i+1}, \dots, t_{i+d})$. From (4.22) we therefore obtain

$$b_i = \mathcal{B}[f_\mu](t_{i+1}, \dots, z, \dots, t_{i+d}) = \mathcal{B}[f_\mu](\tau_{i+1}, \dots, z, \dots, \tau_{i+d-1})$$

since we now may choose $k = \mu$. Applying Lemma 4.17 with $x = z$, $a = \tau_i$ and $b = \tau_{i+d}$ yields

$$b_i = \frac{\tau_{i+d} - z}{\tau_{i+d} - \tau_i} \mathcal{B}[f_\mu](\tau_{i+1}, \dots, \tau_i, \dots, \tau_{i+d}) + \frac{z - \tau_i}{\tau_{i+d} - \tau_i} \mathcal{B}[f_\mu](\tau_i, \dots, \tau_{i+d}, \dots, \tau_{i+d-1}).$$

Exploiting the symmetry of the blossom and again applying (4.22) leads to the middle formula in (4.28). ■

It is sometimes required to insert the same knot several times; this can of course be accomplished by applying the formulas in (4.28) several times. Since blossoms have the property $\mathcal{B}[f](z, \dots, z) = f(z)$, we see that inserting a knot d times in a spline of degree d gives as

a by-product the function value of f at z . This can be conveniently illustrated by listing old and new coefficients in a triangular scheme. Consider the following triangle ($d = 3$),

$$\begin{array}{cccccccc} \cdots & c_{\mu-4}^0 & c_{\mu-3}^0 & c_{\mu-2}^0 & c_{\mu-1}^0 & c_{\mu}^0 & c_{\mu+1}^0 & \cdots \\ & & & c_{\mu-2}^1 & c_{\mu-1}^1 & c_{\mu}^1 & & \\ & & & & c_{\mu-1}^2 & c_{\mu}^2 & & \\ & & & & & c_{\mu}^3 & & \end{array}$$

In the first row we have the coefficients of f on the original knot vector $\boldsymbol{\tau}$. After inserting z in $(\tau_{\mu}, \tau_{\mu+1})$ once, the coefficients relative to the knot vector $\boldsymbol{\tau}^1 = \boldsymbol{\tau} \cup \{z\}$ are

$$(\dots, c_{\mu-4}^0, c_{\mu-3}^0, c_{\mu-2}^1, c_{\mu-1}^1, c_{\mu}^1, c_{\mu}^0, c_{\mu+1}^0, \dots),$$

i.e., we move down one row in the triangle. Suppose that z is inserted once more. The new B-spline coefficients on $\boldsymbol{\tau}^2 = \boldsymbol{\tau}^1 \cup \{z\}$ are now found by moving down to the second row, across this row, and up the right hand side,

$$(\dots, c_{\mu-4}^0, c_{\mu-3}^0, c_{\mu-2}^1, c_{\mu-1}^2, c_{\mu}^2, c_{\mu}^1, c_{\mu}^0, c_{\mu+1}^0, \dots).$$

Similarly, if z is inserted 3 times, we move around the whole triangle. We can also insert z a full $d = 4$ times. We then simply repeat c_{μ}^3 two times in the last row.

Lemma 4.19 shows that Oslo Algorithm 2 (Algorithm 4.11) is not always efficient. To compute a new coefficient in the case where only one new knot is inserted requires at most one convex combination according to Lemma 4.19 while Algorithm 4.11 requires the computation of a full triangle (two nested loops). More efficient versions of the Oslo algorithms can be developed, but this will not be considered here.

The simplicity of the formulas (4.28) indicates that the knot insertion matrix $\mathbf{A}$ must have a simple structure when only one knot is inserted. Setting $\mathbf{c} = (c_i)_{i=1}^n$ and $\mathbf{b} = (b_i)_{i=1}^{n+1}$ and remembering that $\mathbf{b} = \mathbf{A}\mathbf{c}$, we see that $\mathbf{A}$ is given by the $(n+1) \times n$ matrix

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & & 1 & 0 & & & \\ & & & 1 - \lambda_{\mu-d+1} & \lambda_{\mu-d+1} & & & \\ & & & & \ddots & \ddots & & \\ & & & & & 1 - \lambda_{\mu} & \lambda_{\mu} & \\ & & & & & 0 & 1 & \\ & & & & & & \ddots & \ddots \\ & & & & & & & 0 & 1 \end{pmatrix}, \quad (4.29)$$

where $\lambda_i = (z - \tau_i) / (\tau_{i+d} - \tau_i)$ for $\mu - d + 1 \leq i \leq \mu$. All the entries off the two diagonals are zero and such matrices are said to be bi-diagonal. Since z lies in the interval $[\tau_{\mu}, \tau_{\mu+1})$ all the entries in $\mathbf{A}$ are nonnegative. This property generalises to arbitrary knot insertion matrices.

Lemma 4.20. Let $\tau = (\tau_j)_{j=1}^{n+d+1}$ and $t = (t_i)_{i=1}^{m+d+1}$ be two knot vectors for splines of degree d with $\tau \subseteq t$. All the entries of the knot insertion matrix A from $\mathbb{S}_{d,\tau}$ to $\mathbb{S}_{d,t}$ are nonnegative and A can be factored as

$$A = A_{m-n} A_{m-n-1} \cdots A_1, \quad (4.30)$$

where A_i is a bi-diagonal $(n+i) \times (n+i-1)$ -matrix with nonnegative entries.

Proof. Let us denote the $m-n$ knots that are in t but not in τ by $(z_i)_{i=1}^{m-n}$. Set $t^0 = \tau$ and $t^i = t^{i-1} \cup (z_i)$ for $i = 1, \dots, m-n$. Denote by A_i the knot insertion matrix from t^{i-1} to t^i . By applying Böhm's method $m-n$ times we obtain (4.30). Since all the entries in each of the matrices A_i are nonnegative the same must be true of A . ■

4.5 Bounding the number of sign changes in a spline

In this section we will make use of Böhm's method for knot insertion to prove that the number of sign changes in a spline function is bounded by the number of sign changes in its B-spline coefficient vector. This provides a generalisation of an interesting property of polynomials known as Descartes' rule of signs. Bearing the name of Descartes, this result is of course classical, but it is rarely mentioned in elementary mathematics textbooks. Before stating Descartes' rule of signs let us record what we mean by sign changes in a definition.

Definition 4.21. Let $c = (c_i)_{i=1}^n$ be a vector of real numbers. The number of sign changes in c (zeros are ignored) is denoted $S^-(c)$. The number of sign changes in a function f in an interval (a, b) is denoted $S_{(a,b)}^-(f) = S^-(f)$, provided this number is finite. It is given by the largest possible integer r such that an increasing sequence of $r+1$ real numbers $x_1 < \cdots < x_{r+1}$ in (a, b) can be found with the property that $S^-(f(x_1), \dots, f(x_{r+1})) = r$.

Example 4.22. Let us consider some simple examples of counting sign changes. It is easily checked that

$$\begin{aligned} S^-(1, -2) &= 1, & S^-(1, 0, -1, 3) &= 2, \\ S^-(1, 0, 2) &= 0, & S^-(2, 0, 0, 0, -1) &= 1, \\ S^-(1, -1, 2) &= 2, & S^-(2, 0, 0, 0, 1) &= 0. \end{aligned}$$

As stated in the definition, we simply count sign changes by counting the number of jumps from positive to negative values and from negative to positive, ignoring all components that are zero.

Descartes' rule of signs bounds the number of zeros in a polynomial by the number of sign changes in its coefficients. Recall that z is a zero of f of multiplicity $r \geq 1$ if $f(z) = Df(z) = \cdots = D^{r-1}f(z) = 0$ but $D^r f(z) \neq 0$.

Theorem 4.23 (Descartes' rule of signs). Let $p = \sum_{i=0}^d c_i x^i$ be a polynomial of degree d with coefficients $c = (c_0, \dots, c_d)^T$, and let $Z(p)$ denote the total number of zeros of p in the interval $(0, \infty)$, counted with multiplicities. Then

$$Z(p) \leq S^-(c),$$

i.e., the number of positive zeros of p is bounded by the number of sign changes in its coefficients.

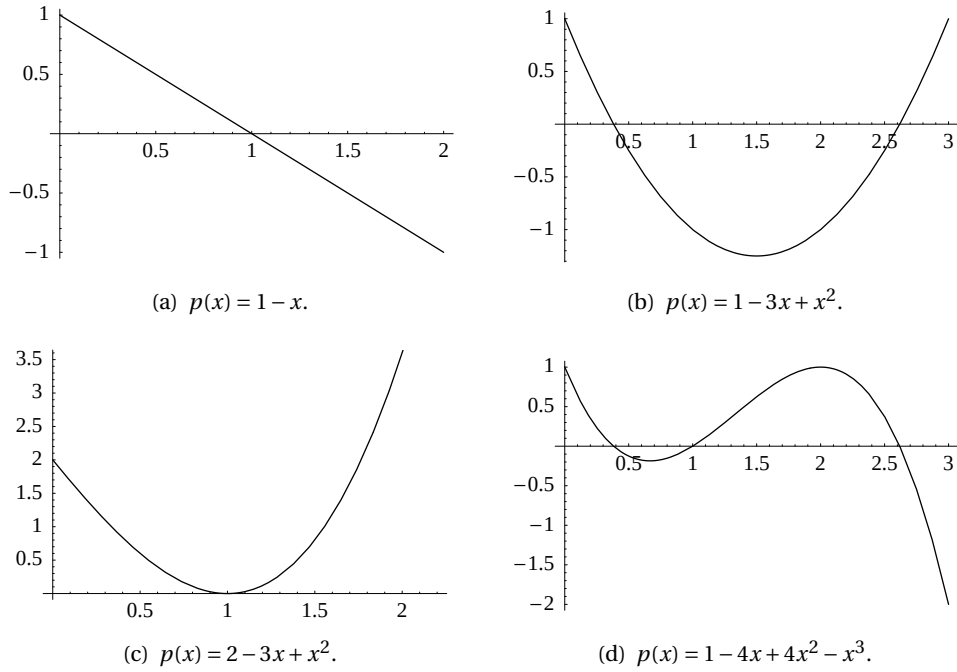


Figure 4.6. Illustrations of Descartes' rule of signs: the number of zeros in $(0, \infty)$ is no greater than the number of strong sign changes in the coefficients.

Figures 4.6 (a)–(d) show some polynomials and their zeros in $(0, \infty)$.

Our aim is to generalise this result to spline functions, written in terms of B-splines. This is not so simple because it is difficult to count zeros for splines. In contrast to polynomials, a spline may for instance be zero on an interval without being identically zero. In this section we will therefore only consider zeros that are also sign changes. In the next section we will then generalise and allow multiple zeros.

To bound the number of sign changes of a spline we will investigate how knot insertion influences the number of sign changes in the B-spline coefficients. Let $\mathbb{S}_{d,\tau}$ and $\mathbb{S}_{d,t}$ be two spline spaces of degree d , with $\mathbb{S}_{d,\tau} \subseteq \mathbb{S}_{d,t}$. Recall from Section 4.4 that to get from the knot vector τ to the refined knot vector t , we can insert one knot at a time. If there are ℓ more knots in τ than in t , this leads to a factorisation of the knot insertion matrix A as

$$A = A_\ell A_{\ell-1} \cdots A_1, \quad (4.31)$$

where A_k is a $(n+k) \times (n+k-1)$ matrix for $k = 1, \dots, \ell$, if $\dim \mathbb{S}_{d,\tau} = n$. Each of the matrices A_k corresponds to insertion of only one knot, and all the nonzero entries of the bi-diagonal matrix A_k are found in positions (i, i) and $(i+1, i)$ for $i = 1, \dots, n+k-1$, and these entries are all nonnegative (in general many of them will be zero).

We start by showing that the number of sign changes in the B-spline coefficients is reduced when the knot vector is refined.

Lemma 4.24. Let $\mathbb{S}_{d,\tau}$ and $\mathbb{S}_{d,t}$ be two spline spaces such that t is a refinement of τ . Let $f = \sum_{j=1}^n c_j B_{j,d,\tau} = \sum_{i=1}^m b_i B_{i,d,t}$ be a spline in $\mathbb{S}_{d,\tau}$ with B-spline coefficients c in $\mathbb{S}_{d,\tau}$ and b in $\mathbb{S}_{d,t}$. Then b has no more sign changes than c , i.e.,

$$S^-(Ac) = S^-(b) \leq S^-(c), \quad (4.32)$$

where A is the knot insertion matrix from τ to t .

Proof. Since we can insert the knots one at a time, it clearly suffices to show that (4.32) holds in the case where there is only one more knot in t than in τ . In this case we know from Lemma 4.19 that A is bidiagonal so

$$b_i = \alpha_{i-1}(i)c_{i-1} + \alpha_i(i)c_i, \quad \text{for } i = 1, \dots, n+1,$$

where $(\alpha_j(i))_{i,j=1}^{n+1,n}$ are the entries of A (for convenience of notation we have introduced two extra entries that are zero, $\alpha_0(1) = \alpha_{n+1}(n+1) = 0$). Since $\alpha_{i-1}(i)$ and $\alpha_i(i)$ both are nonnegative, the sign of b_i must be the same as either c_{i-1} or c_i (or be zero). Since the number of sign changes in a vector is not altered by inserting zeros or a number with the same sign as one of its neighbours we have

$$S^-(c) = S^-(b_1, c_1, b_2, c_2, \dots, b_{n-1}, c_{n-1}, b_n, c_n, b_{n+1}) \geq S^-(b).$$

The last inequality follows since the number of sign changes in a vector is always reduced when entries are removed. ■

From Lemma 4.24 we can quite easily bound the number of sign changes in a spline in terms of the number of sign changes in its B-spline coefficients.

Theorem 4.25. Let $f = \sum_{j=1}^n c_j B_{j,d,\tau}$ be a spline in $\mathbb{S}_{d,\tau}$. Then

$$S^-(f) \leq S^-(c) \leq n-1. \quad (4.33)$$

Proof. Suppose that $S^-(f) = \ell$, and let $(x_i)_{i=1}^{\ell+1}$ be $\ell+1$ points chosen so that $S^-(f) = S^-(f(x_1), \dots, f(x_{\ell+1}))$. We form a new knot vector t that includes τ as a subsequence, but in addition each of the x_i occurs exactly $d+1$ times in t . From our study of knot insertion we know that f may be written $f = \sum_j b_j B_{j,d,t}$ for suitable coefficients (b_j) , and from Lemma 2.3 we know that each of the function values $f(x_i)$ will appear as a B-spline coefficient in b . We therefore have

$$S^-(f) \leq S^-(b) \leq S^-(c),$$

the last inequality following from Lemma 4.24. The last inequality in (4.33) follows since an n -vector can only have $n-1$ sign changes. ■

The validity of Theorem 4.25 can be checked with the two plots in Figure 4.7 as well as all other figures which include both a spline function and its control polygon.

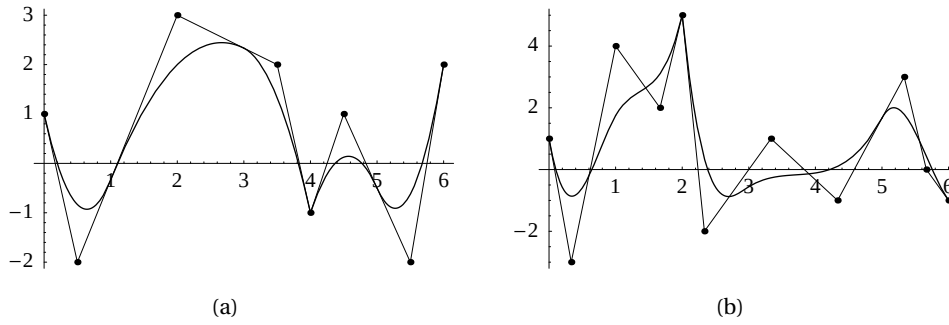


Figure 4.7. A quadratic spline (a) and a cubic spline (b) with their control polygons.

4.6 Exercises

- 4.1** In this exercise we are going to study a change of polynomial basis from the Bernstein basis to the Monomial basis. Recall that the Bernstein basis of degree d is defined by

$$B_j^d(x) = \binom{d}{j} x^j (1-x)^{d-j}, \quad \text{for } j = 0, 1, \dots, d. \quad (4.34)$$

A polynomial p of degree d is said to be written in Monomial form if $p(x) = \sum_{j=0}^d b_j x^j$ and in Bernstein form if $p(x) = \sum_{j=0}^d c_j B_j^d(x)$. In this exercise the binomial formula

$$(a+b)^d = \sum_{k=0}^d \binom{d}{k} a^k b^{d-k} \quad (4.35)$$

will be useful.

- a) By applying (4.35), show that

$$B_j^d(x) = \sum_{i=j}^d (-1)^{i-j} \binom{d}{j} \binom{d-j}{i-j} x^i, \quad \text{for } j = 0, 1, \dots, d.$$

Also show that $\binom{d}{j} \binom{d-j}{i-j} = \binom{d}{i} \binom{i}{j}$ for $i = j, \dots, d$ and $j = 0, \dots, d$.

- b) The two basis vectors $\mathbf{B}_d = (B_0^d(x), \dots, B_d^d(x))^T$ and $\mathbf{P}_d = (1, x, \dots, x^d)^T$ are related by $\mathbf{B}_d^T = \mathbf{P}_d^T \mathbf{A}_d$ where $\mathbf{A}_d$ is a $(d+1) \times (d+1)$ -matrix $\mathbf{A}_d$. Show that the entries of $\mathbf{A}_d = (a_{i,j})_{i,j=0}^d$ are given by

$$a_{i,j} = \begin{cases} 0, & \text{if } i < j, \\ (-1)^{i-j} \binom{d}{i} \binom{i}{j}, & \text{otherwise.} \end{cases}$$

- c) Show that the entries of $\mathbf{A}_d$ satisfy the recurrence relation

$$a_{i,j} = \beta_i (a_{i-1,j-1} - a_{i-1,j}), \quad \text{where } \beta_i = (d-i+1)/i.$$

Give a detailed algorithm for computing $\mathbf{A}_d$ based on this formula.

- d) Explain how we can find the coefficients of a polynomial relative to the Monomial basis if A_d is known and the coefficients relative to the Bernstein basis are known.

4.2 In this exercise we are going to study the opposite conversion of that in Exercise 1, namely from the Monomial basis to the Bernstein basis.

- a) With the aid of (4.35), show that for all x and t in $\mathbb{R}$ we have

$$(tx + (1-x))^d = \sum_{k=0}^d B_k^d(x) t^k. \quad (4.36)$$

The function $G(t) = (tx + (1-x))^d$ is called a generating function for the Bernstein polynomials.

- b) Show that $\sum_{k=0}^d B_k^d(x) = 1$ for all x by choosing a suitable value for t in (4.36).
 c) Find two different expressions for $G^{(j)}(1)/j!$ and show that this leads to the formulas

$$\binom{d}{j} x^j = \sum_{i=j}^d \binom{i}{j} B_k^d(x), \quad \text{for } j = 0, \dots, d. \quad (4.37)$$

- d) Show that the entries of the matrix $B_d = (b_{i,j})_{i,j=0}^d$ such that $P_d^T = B_d^T B_d$ are given by

$$b_{i,j} = \begin{cases} 0, & \text{if } i < j, \\ \binom{i}{j} / \binom{d}{j}, & \text{otherwise.} \end{cases}$$

4.3 Let P denote the cubic Bernstein basis on the interval $[0, 1]$ and let Q denote the cubic Bernstein basis on the interval $[2, 3]$. Determine the matrix A_3 such that $P(x)^T = Q(x)^T A_3$ for all real numbers x .

4.4 Let A denote the knot insertion matrix for the linear ($d = 1$) B-splines on $\tau = (\tau_j)_{j=1}^{n+2}$ to the linear B-splines in $t = (t_i)_{i=1}^{m+2}$. We assume that τ and t are 2-extended with $\tau_1 = t_1$ and $\tau_{n+2} = t_{m+2}$ and $\tau \subseteq t$.

- a) Determine A when $\tau = (0, 0, 1/2, 1, 1)$ and $t = (0, 0, 1/4, 1/2, 3/4, 1, 1)$.
 b) Device a detailed algorithm that computes A for general τ and t and requires $O(m)$ operations.
 c) Show that the matrix $A^T A$ is tridiagonal.

4.5 Prove Lemma 4.2 in the general case where τ and t are not $d+1$ -regular. Hint: Augment both τ and t by inserting $d+1$ identical knots at the beginning and end.

4.6 Prove Theorem 4.7 in the general case where the knot vectors are not $d+1$ -regular with common knots at the ends. Hint: Use the standard trick of augmenting τ and t with $d+1$ identical knots at both ends to obtain new knot vectors $\hat{\tau}$ and $\hat{t}$. The knot insertion matrix from τ to t can then be identified as a sub-matrix of the knot insertion matrix from $\hat{\tau}$ to $\hat{t}$.

4.7 Show that if τ and t are $d + 1$ -regular knot vectors with $\tau \subseteq t$ whose knots agree at the ends then $\sum_j \alpha_{j,d}(i) = 1$.

4.8 Implement Algorithm 4.11 and test it on two examples. Verify graphically that the control polygon converges to the spline as more and more knots are inserted.

4.9 Let f be a function that satisfies the identity

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \quad (4.38)$$

for all real numbers x and y and all real numbers α and β such that $\alpha + \beta = 1$. Show that then f must be an affine function. Hint: Use the alternative form of equation (4.38) found in Lemma 4.17.

4.10 Find the cubic blossom $\mathcal{B}[p](x_1, x_2, x_3)$ when p is given by:

- a) $p(x) = x^3$.
- b) $p(x) = 1$.
- c) $p(x) = 2x + x^2 - 4x^3$.
- d) $p(x) = 0$.
- e) $p(x) = (x - a)^2$ where a is some real number.

Spline Approximation

This chapter introduces a number of methods for obtaining spline approximations to given functions, or more precisely, to data obtained by sampling a function. In Section 5.1, we focus on local methods where the approximation at a point x only depends on data values near x . Connecting neighbouring data points with straight lines is one such method where the value of the approximation at a point only depends on the two nearest data points.

In order to get smoother approximations, we must use splines of higher degree. With cubic polynomials we can prescribe, or *interpolate*, position and first derivatives at two points. Therefore, given a set of points with associated function values and first derivatives, we can determine a sequence of cubic polynomials that interpolate the data, joined together with continuous first derivatives. This is the cubic Hermite interpolant of Section 5.1.2.

In Section 5.2 we study global cubic approximation methods where we have to solve a system of equations involving all the data points in order to obtain the approximation. Like the local methods in Section 5.1, these methods *interpolate* the data, which now only are positions. The gain in turning to global methods is that the approximation may have more continuous derivatives and still be as accurate as the local methods.

The cubic spline interpolant with so called natural end conditions solves an interesting extremal problem. Among all functions with a continuous second derivative that interpolate a set of data, the natural cubic spline interpolant is the one whose integral of the square of the second derivative is the smallest. This is the foundation for various interpretations of splines, and is all discussed in Section 5.2.

Two approximation methods for splines of arbitrary degree are described in Section 5.3. The first method is spline interpolation with B-splines defined on some rather arbitrary knot vector. The disadvantage of using interpolation methods is that the approximations have a tendency to oscillate. If we reduce the dimension of the approximating spline space, and instead minimize the error at the data points this problem can be greatly reduced. Such *least squares methods* are studied in Section 5.3.2.

We end the chapter by a discussing a very simple approximation method, the *Variation Diminishing Spline Approximation*. This approximation scheme has the desirable ability to transfer the sign of some of the derivatives of a function to the approximation. This is impor-

tant since many important characteristics of the shape of a function is closely related to the sign of the derivatives.

5.1 Local Approximation Methods

When we construct an approximation to data, it is usually an advantage if the approximation at a point x only depends on the data near x . If this is the case, changing the data in some small area will only affect the approximation in the same area. The variation diminishing approximation method and in particular piecewise linear interpolation has this property, it is a *local* method. In this section we consider another local approximation method.

5.1.1 Piecewise linear interpolation

The simplest way to obtain a continuous approximation to a set of ordered data points is to connect neighbouring data points with straight lines. This approximation is naturally enough called the *piecewise linear interpolant* to the data. It is clearly a linear spline and can therefore be written as a linear combination of B-splines on a suitable knot vector. The knots must be at the data points, and since the interpolant is continuous, each interior knot only needs to occur once in the knot vector. The construction is given in the following proposition.

Proposition 5.1. *Let $(x_i, y_i)_{i=1}^m$ be a set of data points with $x_i < x_{i+1}$ for $i = 1, \dots, m-1$, and construct the 2-regular knot vector $\mathbf{t}$ as*

$$\mathbf{t} = (t_i)_{i=1}^{m+2} = (x_1, x_1, x_2, x_3, \dots, x_{m-1}, x_m, x_m).$$

Then the linear spline g given by

$$g(x) = \sum_{i=1}^m y_i B_{i,1}(x)$$

satisfies the interpolation conditions

$$g(x_i) = y_i, \quad \text{for } i = 1, \dots, m-1, \quad \text{and} \quad \lim_{x \rightarrow x_m^-} g(x) = y_m. \quad (5.1)$$

The last condition states that the limit of g from the left at x_m is y_m . If the data are taken from a function f so that $y_i = f(x_i)$ for $i = 1, \dots, m$, the interpolant g is often denoted by $I_1 f$.

Proof. From Example 2.2 in Chapter 2, we see that the B-spline $B_{i,1}$ for $1 \leq i \leq m$ is given by

$$B_{i,1}(x) = \begin{cases} (x - x_{i-1})/(x_i - x_{i-1}), & \text{if } x_{i-1} \leq x < x_i, \\ (x_{i+1} - x)/(x_{i+1} - x_i), & \text{if } x_i \leq x < x_{i+1}, \\ 0, & \text{otherwise,} \end{cases}$$

where we have set $x_0 = x_1$ and $x_{m+1} = x_m$. This means that $B_{i,1}(x_i) = 1$ for $i < m$ and $\lim_{x \rightarrow x_m^-} B_{m,1}(x) = 1$, while $B_{i,1}(x_j) = 0$ for all $j \neq i$, so the interpolation conditions (5.1) are satisfied. ■

The piecewise linear interpolant preserves the shape of the data extremely well. The obvious disadvantage of this approximation is its lack of smoothness.

Intuitively, it seems reasonable that if f is continuous, it should be possible to approximate it to within any accuracy by piecewise linear interpolants, if we let the distance between the data points become small enough. This is indeed the case. Note that the symbol $C^j[a, b]$ denotes the set of all functions defined on $[a, b]$ with values in $\mathbb{R}$ whose first j derivatives are continuous.

Proposition 5.2. *Suppose that $a = x_1 < x_2 < \cdots < x_m = b$ are given points, and set $\Delta x = \max_{1 \leq i \leq m-1} \{x_{i+1} - x_i\}$.*

1. *If $f \in C[a, b]$, then for every $\epsilon > 0$ there is a $\delta > 0$ such that if $\Delta x < \delta$, then $|f(x) - I_1 f(x)| < \epsilon$ for all $x \in [a, b]$.*
2. *If $f \in C^2[a, b]$ then for all $x \in [a, b]$,*

$$|f(x) - (I_1 f)(x)| \leq \frac{1}{8} (\Delta x)^2 \max_{a \leq z \leq b} |f''(z)|, \quad (5.2)$$

$$|f'(x) - (I_1 f)'(x)| \leq \frac{1}{2} \Delta x \max_{a \leq z \leq b} |f''(z)|. \quad (5.3)$$

Part (i) of Proposition 5.2 states that piecewise linear interpolation to a continuous function converges to the function when the distance between the data points goes to zero. More specifically, given a tolerance ϵ , we can make the error less than the tolerance by choosing Δx sufficiently small.

Part (ii) of Proposition 5.2 gives an upper bound for the error in case the function f is smooth, which in this case means that f and its first two derivatives are continuous. The inequality in (5.2) is often stated as “piecewise linear approximation has approximation order two”, meaning that Δx is raised to the power of two in (5.2).

The bounds in Proposition 5.2 depend both on Δx and the size of the second derivative of f . Therefore, if the error is not small, it must be because one of these quantities are large. If in some way we can find an upper bound M for f'' , i.e.,

$$|f''(x)| \leq M, \quad \text{for } x \in [a, b], \quad (5.4)$$

we can determine a value of Δx such that the error, measured as in (5.2), is smaller than some given tolerance ϵ . We must clearly require $(\Delta x)^2 M/8 < \epsilon$. This inequality holds provided $\Delta x < \sqrt{8\epsilon/M}$. We conclude that for any $\epsilon > 0$, we have the implication

$$\Delta x < \sqrt{\frac{8\epsilon}{M}} \implies |f(x) - I_1 f(x)| < \epsilon, \quad \text{for } x \in [x_1, x_m]. \quad (5.5)$$

This estimate tells us how densely we must sample f in order to have error smaller than ϵ everywhere.

We will on occasions want to compute the piecewise linear interpolant to a given higher degree spline f . A spline does not necessarily have continuous derivatives, but at least we know where the discontinuities are. The following proposition is therefore meaningful.

Proposition 5.3. Suppose that $f \in \mathbb{S}_{d,\mathbf{t}}$ for some d and $\mathbf{t}$ with interior knots of multiplicity at most d (so f is continuous). If the break points $(x_i)_{i=1}^m$ are chosen so as to include all the knots in $\mathbf{t}$ where f' is discontinuous, the bounds in (5.2) and (5.3) continue to hold.

5.1.2 Cubic Hermite interpolation

The piecewise linear interpolant has the nice property of being a local construction: The interpolant on an interval $[x_i, x_{i+1}]$ is completely defined by the value of f at x_i and x_{i+1} . The other advantage of f is that it does not oscillate between data points and therefore preserves the shape of f if $\Delta \mathbf{x}$ is small enough. In this section we construct an interpolant which, unlike the piecewise linear interpolant, has continuous first derivative, and which, like the piecewise linear interpolant, only depends on data values locally. The price of the smoothness is that this interpolant requires information about derivatives, and shape preservation in the strong sense of the piecewise linear interpolant cannot be guaranteed. The interpolant we seek is the solution of the following problem.

Problem 5.4 (Piecewise Cubic Hermite Interpolation). Let the discrete data $(x_i, f(x_i), f'(x_i))_{i=1}^m$ with $a = x_1 < x_2 < \dots < x_m = b$ be given. Find a function $g = H_3 f$ that satisfies the following conditions:

1. On each subinterval (x_i, x_{i+1}) the function g is a cubic polynomial.
2. The given function f is interpolated by g in the sense that

$$g(x_i) = f(x_i), \quad \text{and} \quad g'(x_i) = f'(x_i), \quad \text{for } i = 1, \dots, m. \quad (5.6)$$

A spline g that solves Problem 5.4 must be continuous and have continuous first derivative since two neighbouring pieces meet with the same value $f(x_i)$ and first derivative $f'(x_i)$ at a join x_i . Since Hf should be a piecewise cubic polynomial, it is natural to try and define a knot vector so that Hf can be represented as a linear combination of B-splines on this knot vector. To get the correct smoothness, we need at least a double knot at each data point. Since $d = 3$ and we have $2m$ interpolation conditions, the length of the knot vector should be $2m + 4$, and we might as well choose to use a 4-regular knot vector. We achieve this by making each interior data point a knot of multiplicity two and place four knots at the two ends. This leads to the knot vector

$$\boldsymbol{\tau} = (\tau_i)_{i=1}^{2m+4} = (x_1, x_1, x_1, x_1, x_2, x_2, \dots, x_{m-1}, x_{m-1}, x_m, x_m, x_m, x_m), \quad (5.7)$$

which we call *the Cubic Hermite knot vector* on $\mathbf{x} = (x_1, \dots, x_m)$. This allows us to construct the solution to Problem 5.4.

Proposition 5.5. Problem 5.4 has a unique solution Hf in the spline space $\mathbb{S}_{3,\boldsymbol{\tau}}$, where $\boldsymbol{\tau}$ is given in equation (5.7). More specifically, the solution is given by

$$Hf = \sum_{i=1}^{2m} c_i B_{i,3}, \quad (5.8)$$

where

$$\left. \begin{aligned} c_{2i-1} &= f(x_i) - \frac{1}{3}\Delta x_{i-1}f'(x_i), \\ c_{2i} &= f(x_i) + \frac{1}{3}\Delta x_i f'(x_i), \end{aligned} \right\} \quad \text{for } i = 1, \dots, m, \quad (5.9)$$

where $\Delta x_j = x_{j+1} - x_j$, and the points x_0 and x_{m+1} are defined by $x_0 = x_1$ and $x_{m+1} = x_m$.

Proof. We leave the proof that the spline defined by (5.9) satisfies the interpolation conditions in Problem 5.4 to the reader.

By construction, the solution is clearly a cubic polynomial. That there is only one solution follows if we can show that the only solution that solves the problem with $f(x_i) = f'(x_i) = 0$ for all i is the function that is zero everywhere. For if the general problem has two solutions, the difference between these must solve the problem with all the data equal to zero. If this difference is zero, the two solutions must be equal.

To show that the solution to the problem where all the data are zero is the zero function, it is clearly enough to show that the solution is zero in one subinterval. On each subinterval the function Hf is a cubic polynomial with value and derivative zero at both ends, and it therefore has four zeros (counting multiplicity) in the subinterval. But the only cubic polynomial with four zeros is the polynomial that is identically zero. From this we conclude that Hf must be zero in each subinterval and therefore identically zero. ■

Let us see how this method of approximation behaves in a particular situation.

Example 5.6. We try to approximate the function $f(x) = x^4$ on the interval $[0, 1]$ with only one polynomial piece so that $m = 2$ and $[a, b] = [x_1, x_m] = [0, 1]$. Then the cubic Hermite knots are just the Bernstein knots. From (5.9) we find $(c_1, c_2, c_3, c_4) = (0, 0, -1/3, 1)$, and

$$(Hf)(x) = -\frac{1}{3}3x^2(1-x) + x^3 = 2x^3 - x^2.$$

The two functions f and Hf are shown in Figure 5.1.

Example 5.7. Let us again approximate $f(x) = x^4$ on $[0, 1]$, but this time we use two polynomial pieces so that $m = 3$ and $\mathbf{x} = (0, 1/2, 1)$. In this case the cubic Hermite knots are $\mathbf{\tau} = (0, 0, 0, 0, 1/2, 1/2, 1, 1, 1, 1)$, and we find the coefficients $\mathbf{c} = (0, 0, -1/48, 7/48, 1/3, 1)$. The two functions f and Hf are shown in Figure 5.1 (a). With the extra knots at $1/2$ (cf. Example 5.6), we get a much more accurate approximation to x^4 . In fact, we see from the error plots in Figures 5.1 (b) and 5.1 (b) that the maximum error has been reduced from 0.06 to about 0.004, a factor of about 15.

Note that in Example 5.6 the approximation becomes negative even though f is nonnegative in all of $[0, 1]$. This shows that in contrast to the piecewise linear interpolant, the cubic Hermite interpolant Hf does not preserve the sign of f . However, it is simple to give conditions that guarantee Hf to be nonnegative.

Proposition 5.8. Suppose that the function f to be approximated by cubic Hermite interpolation satisfies the conditions

$$\left. \begin{aligned} f(x_i) - \frac{1}{3}\Delta x_{i-1}f'(x_i) &\geq 0, \\ f(x_i) + \frac{1}{3}\Delta x_i f'(x_i) &\geq 0, \end{aligned} \right\} \quad \text{for } i = 1, \dots, m.$$

Then the cubic Hermite interpolant Hf is nonnegative on $[a, b]$.

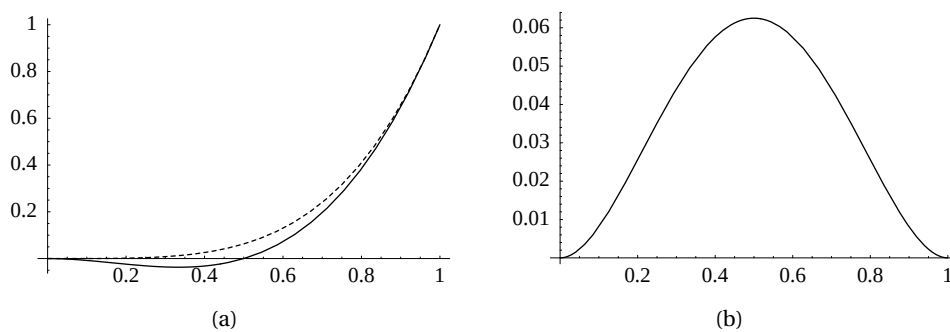


Figure 5.1. Figure (a) shows the cubic Hermite interpolant (solid) to $f(x) = x^4$ (dashed), see Example 5.6, while the error in this approximation is shown in (b).

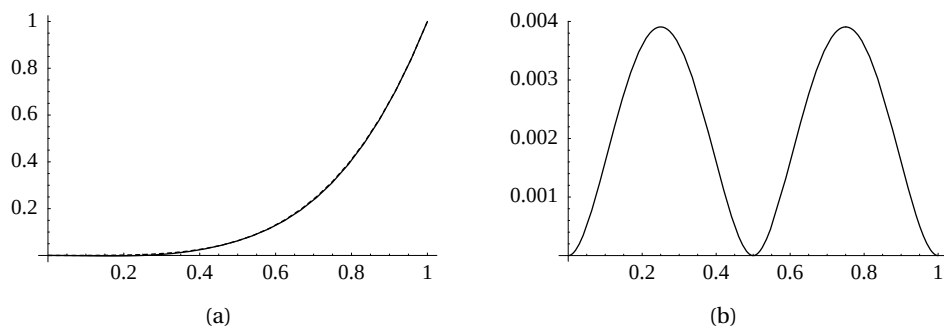


Figure 5.2. Figure (a) shows the cubic Hermite interpolant (solid) to $f(x) = x^4$ (dashed) with two polynomial pieces, see Example 5.7, while the error in the approximation is shown in (b).

Proof. In this case, the spline approximation Hf given by Proposition 5.5 has nonnegative B-spline coefficients, so that $(Hf)(x)$ for each x is a sum of nonnegative quantities and therefore nonnegative. ■

As for the piecewise linear interpolant, it is possible to relate the error to the spacing in $\mathbf{x}$ and the size of some derivative of f .

Proposition 5.9. *Suppose that f has continuous derivatives up to order four on the interval $[x_1, x_m]$. Then*

$$|f(x) - (Hf)(x)| \leq \frac{1}{384}(\Delta \mathbf{x})^4 \max_{a \leq z \leq b} |f^{(iv)}(z)|, \quad \text{for } x \in [a, b]. \quad (5.10)$$

This estimate also holds whenever f is in some spline space $\mathbb{S}_{d,\tau}$ provided f has a continuous derivative at all the x_i .

Proof. See a text on numerical analysis. ■

The error estimate in (5.10) says that if we halve the distance between the interpolation points, then we can expect the error to decrease by a factor of $2^4 = 16$. This is usually referred to as “fourth order convergence”. This behaviour is confirmed by Examples 5.6 and 5.7 where the error was reduced by a factor of about 15 when $\Delta \mathbf{x}$ was halved.

From Proposition 5.9, we can determine a spacing between data points that guarantees that the error is smaller than some given tolerance. Suppose that

$$|f^{(iv)}(x)| \leq M, \quad \text{for } x \in [a, b].$$

For any $\epsilon > 0$ we then have

$$\Delta \mathbf{x} \leq \left(\frac{384\epsilon}{M} \right)^{1/4} \implies |f(x) - (Hf)(x)| \leq \epsilon, \quad \text{for } x \in [a, b].$$

When $\epsilon \rightarrow 0$, the number $\epsilon^{1/4}$ goes to zero more slowly than the term $\epsilon^{1/2}$ in the corresponding estimate for piecewise linear interpolation. This means that when ϵ becomes small, we can usually use a larger $\Delta \mathbf{x}$ in cubic Hermite interpolation than in piecewise linear interpolation, or equivalently, we generally need fewer data points in cubic Hermite interpolation than in piecewise linear interpolation to obtain the same accuracy.

5.1.3 Estimating the derivatives

Sometimes we have function values available, but no derivatives, and we still want a smooth interpolant. In such cases we can still use cubic Hermite interpolation if we can somehow estimate the derivatives. This can be done in many ways, but one common choice is to use the slope of the parabola interpolating the data at three consecutive data-points. To find this slope we observe that the parabola p_i such that $p_i(x_j) = f(x_j)$, for $j = i-1, i$ and $i+1$, is given by

$$p_i(x) = f(x_{i-1}) + (x - x_{i-1})\delta_{i-1} + (x - x_{i-1})(x - x_i) \frac{\delta_i - \delta_{i-1}}{\Delta x_{i-1} + \Delta x_i},$$

where

$$\delta_j = (f(x_{j+1}) - f(x_j)) / \Delta x_j.$$

We then find that

$$p'_i(x_i) = \delta_{i-1} + \Delta x_{i-1} \frac{\delta_i - \delta_{i-1}}{\Delta x_{i-1} + \Delta x_i}.$$

After simplification, we obtain

$$p'_i(x_i) = \frac{\Delta x_{i-1} \delta_i + \Delta x_i \delta_{i-1}}{\Delta x_{i-1} + \Delta x_i}, \quad \text{for } i = 2, \dots, m-1, \quad (5.11)$$

and this we use as an estimate for $f'(x_i)$. Using cubic Hermite interpolation with the choice (5.11) for derivatives is known as *cubic Bessel interpolation*. It is equivalent to a process known as *parabolic blending*. The end derivatives $f'(x_1)$ and $f'(x_m)$ must be estimated separately. One possibility is to use the value in (5.11) with $x_0 = x_3$ and $x_{m+1} = x_{m-2}$.

5.2 Cubic Spline Interpolation

Cubic Hermite interpolation works well in many cases, but it is inconvenient that the derivatives have to be specified. In Section 5.1.3 we saw one way in which the derivatives can be estimated from the function values. There are many other ways to estimate the derivatives at the data points; one possibility is to demand that the interpolant should have a continuous second derivative at each interpolation point. As we shall see in this section, this leads to a system of linear equations for the unknown derivatives so the locality of the construction is lost, but we gain one more continuous derivative which is important in some applications. A surprising property of this interpolant is that it has the smallest second derivative of all C^2 -functions that satisfy the interpolation conditions. The cubic spline interpolant therefore has a number of geometric and physical interpretations that we discuss briefly in Section 5.2.1.

Our starting point is m points $a = x_1 < x_2 < \dots < x_m = b$ with corresponding values $y_i = f(x_i)$. We are looking for a piecewise cubic polynomial that interpolates the given values and belongs to $C^2[a, b]$. In this construction, it turns out that we need two extra conditions to specify the interpolant uniquely. One of the following boundary conditions is often used.

$$\begin{array}{ll} \text{(i)} & g'(a) = f'(a) \text{ and } g'(b) = f'(b); \quad \text{H(ermite)} \\ \text{(ii)} & g''(a) = g''(b) = 0; \quad \text{N(atural)} \\ \text{(iii)} & g''' \text{ is continuous at } x_2 \text{ and } x_{m-1}. \quad \text{F(ree)} \\ \text{(iv)} & D^j g(a) = D^j g(b) \text{ for } j = 1, 2. \quad \text{P(eriodic)} \end{array} \quad (5.12)$$

The periodic boundary conditions are suitable for closed parametric curves where $f(x_1) = f(x_m)$.

In order to formulate the interpolation problems more precisely, we will define the appropriate spline spaces. Since we want the splines to have continuous derivatives up to order two, we know that all interior knots must be simple. For the boundary conditions H, N, and E, we therefore define the 4-regular knot vectors

$$\begin{aligned} \tau_H = \tau_N = (\tau_i)_{i=1}^{m+6} &= (x_1, x_1, x_1, x_1, x_2, x_3, \dots, x_{m-1}, x_m, x_m, x_m, x_m), \\ \tau_F &= (\tau_i)_{i=1}^{m+4} = (x_1, x_1, x_1, x_1, x_3, x_4, \dots, x_{m-2}, x_m, x_m, x_m, x_m). \end{aligned} \quad (5.13)$$

This leads to three cubic spline spaces $\mathbb{S}_{3,\tau_H}$, $\mathbb{S}_{3,\tau_N}$ and $\mathbb{S}_{3,\tau_F}$, all of which will have two continuous derivatives at each interior knot. Note that x_2 and x_{m-1} are missing in τ_F . This means that any $h \in \mathbb{S}_{3,\tau_F}$ will automatically satisfy the free boundary conditions.

We consider the following interpolation problems.

Problem 5.10. Let the data $(x_i, f(x_i))_{i=1}^m$ with $a = x_1 < x_2 < \dots < x_m = b$ be given, together with $f'(x_1)$ and $f'(x_m)$ if they are needed. For Z denoting one of H, N , or F , we seek a spline $g = g_Z = I_Z f$ in the spline space $\mathbb{S}_{3,\tau_Z}$, such that $g(x_i) = f(x_i)$ for $i = 1, 2, \dots, m$, and such that boundary condition Z holds.

We consider first Problem 5.10 in the case of Hermite boundary conditions. Our aim is to show that the problem has a unique solution, and this requires that we study it in some detail.

It turns out that any solution of Problem 5.10 H has a remarkable property. It is the interpolant which, in some sense, has the smallest second derivative. To formulate this, we need to work with integrals of the splines. An interpretation of these integrals is that they are generalizations of the dot product or *inner product* for vectors. Recall that if $\mathbf{u}$ and $\mathbf{v}$ are two vectors in $\mathbb{R}^n$, then their inner product is defined by

$$\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u} \cdot \mathbf{v} = \sum_{i=1}^n u_i v_i,$$

and the length or *norm* of $\mathbf{u}$ can be defined in terms of the inner product as

$$\|\mathbf{u}\| = \langle \mathbf{u}, \mathbf{u} \rangle^{1/2} = \left(\sum_{i=1}^n u_i^2 \right)^{1/2}.$$

The corresponding inner product and norm for functions are

$$\langle u, v \rangle = \int_a^b u(x) v(x) dx = \int_a^b uv$$

and

$$\|u\| = \left(\int_a^b u(t)^2 dt \right)^{1/2} = \left(\int_a^b u^2 \right)^{1/2}.$$

It therefore makes sense to say that two functions u and v are orthogonal if $\langle u, v \rangle = \int uv = 0$.

The first result that we prove says that the error $f - I_H f$ is orthogonal to a family of linear splines.

Lemma 5.11. Denote the error in cubic spline interpolation with Hermite end conditions by $e = f - I_H f$, and let τ be the 2-regular knot vector

$$\tau = (\tau_i)_{i=1}^{m+2} = (x_1, x_1, x_2, x_3, \dots, x_{m-1}, x_m, x_m).$$

Then the second derivative of e is orthogonal to the spline space $\mathbb{S}_{1,\tau}$. In other words

$$\int_a^b e''(x) h(x) dx = 0, \quad \text{for all } h \in \mathbb{S}_{1,\tau}.$$

Proof. Dividing $[a, b]$ into the subintervals $[x_i, x_{i+1}]$ for $i = 1, \dots, m-1$, and using integration by parts, we find

$$\int_a^b e'' h = \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} e'' h = \sum_{i=1}^{m-1} \left(e' h \Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} e' h' \right).$$

Since $e'(a) = e'(b) = 0$, the first term is zero,

$$\sum_{i=1}^{m-1} e' h \Big|_{x_i}^{x_{i+1}} = e'(b) h(b) - e'(a) h(a) = 0. \quad (5.14)$$

For the second term, we observe that since h is a linear spline, its derivative is equal to some constant h_i in the subinterval (x_i, x_{i+1}) , and therefore can be moved outside the integral. Because of the interpolation conditions we have $e(x_{i+1}) = e(x_i) = 0$, so that

$$\sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} e' h' = \sum_{i=1}^{m-1} h_i \int_{x_i}^{x_{i+1}} e'(x) dx = 0.$$

This completes the proof. ■

We can now show that the cubic spline interpolant solves a minimization problem. In any minimization problem, we must specify the space over which we minimize. The space in this case is $\mathbb{E}_H(f)$, which is defined in terms of the related space $\mathbb{E}(f)$

$$\begin{aligned} \mathbb{E}(f) &= \{g \in C^2[a, b] \mid g(x_i) = f(x_i) \text{ for } i = 1, \dots, m\}, \\ \mathbb{E}_H(f) &= \{g \in \mathbb{E}(f) \mid g'(a) = f'(a) \text{ and } g'(b) = f'(b)\}. \end{aligned} \quad (5.15)$$

The space $\mathbb{E}(f)$ is the set of all functions with continuous derivatives up to the second order that interpolate f at the data points. If we restrict the derivatives at the ends to coincide with the derivatives of f we obtain $\mathbb{E}_H(f)$.

The following theorem shows that the second derivative of a cubic interpolating spline has the smallest second derivative of all functions in $\mathbb{E}_H(f)$.

Theorem 5.12. Suppose that $g = I_H f$ is the solution of Problem 5.10 H. Then

$$\int_a^b (g''(x))^2 dx \leq \int_a^b (h''(x))^2 dx \quad \text{for all } h \text{ in } \mathbb{E}_H(f), \quad (5.16)$$

with equality if and only if $h = g$.

Proof. Select some $h \in \mathbb{E}_H(f)$ and set $e = h - g$. Then we have

$$\int_a^b h''^2 = \int_a^b (e'' + g'')^2 = \int_a^b e''^2 + 2 \int_a^b e'' g'' + \int_a^b g''^2. \quad (5.17)$$

Since $g \in \mathbb{S}_{3, \tau_H}$ we have $g'' \in \mathbb{S}_{1, \tau}$, where τ is the knot vector given in Lemma 5.11. Since $g = I_H h = I_H f$, we have $e = h - I_H h$ so we can apply Lemma 5.11 and obtain $\int_a^b e'' g'' = 0$. We conclude that $\int_a^b h''^2 \geq \int_a^b g''^2$.

To show that we can only have equality in (5.16) when $h = g$, suppose that $\int_a^b h''^2 = \int_a^b g''^2$. Using (5.17), we observe that we must have $\int_a^b e''^2 = 0$. But since e'' is continuous, this means that we must have $e'' = 0$. Since we also have $e(a) = e'(a) = 0$, we conclude that $e = 0$. This can be shown by using Taylor's formula

$$e(x) = e(a) + (x-a)e'(a) + \int_a^x e''(t)(x-t) dt.$$

Since $e = 0$, we end up with $g = h$. ■

Lemma 5.11 and Theorem 5.12 allow us to show that the Hermite problem has a unique solution.

Theorem 5.13. *Problem 5.10 H has a unique solution.*

Proof. We seek a function

$$g = I_H f = \sum_{i=1}^{m+2} c_i B_{i,3}$$

in $\mathbb{S}_{3,\tau_H}$ such that

$$\begin{aligned} \sum_{j=1}^{m+2} c_j B_{j,3}(x_i) &= f(x_i), & \text{for } i = 1, \dots, m, \\ \sum_{j=1}^{m+2} c_j B'_{j,3}(x_i) &= f'(x_i), & \text{for } i = 1 \text{ and } m. \end{aligned} \tag{5.18}$$

This is a linear system of $m+2$ equations in the $m+2$ unknown B-spline coefficients. From linear algebra we know that such a system has a unique solution if and only if the corresponding system with zero right-hand side only has the zero solution. This means that existence and uniqueness of the solution will follow if we can show that Problem 5.10 H with zero data only has the zero solution. Suppose that $g \in \mathbb{S}_{3,\tau_H}$ solves Problem 5.10 H with zero data. Clearly $g = 0$ is a solution. According to Theorem 5.12, any other solution must also minimize the integral of the second derivative. By the uniqueness assertion in Theorem 5.12, we conclude that $g = 0$ is the only solution. ■

We have similar results for the “natural” case.

Lemma 5.14. *If $e = f - I_N f$ and τ the knot vector*

$$\tau = (\tau_i)_{i=1}^m = (x_1, x_2, x_3, \dots, x_{m-1}, x_m),$$

the second derivative of e is orthogonal to $\mathbb{S}_{1,\tau}$,

$$\int_a^b e''(x) h(x) dx = 0, \quad \text{for all } h \text{ in } \mathbb{S}_{1,\tau}.$$

Proof. The proof is similar to Lemma 5.11. The relation in (5.14) holds since every $h \in \mathbb{S}_{1,\tau}$ now satisfies $h(a) = h(b) = 0$. ■

Lemma 5.14 allows us to prove that the cubic spline interpolation problem with natural boundary conditions has a unique solution.

Theorem 5.15. *Problem 5.10 N has a unique solution $g = I_N f$. The solution is the unique function in $C^2[a, b]$ with the smallest possible second derivative in the sense that*

$$\int_a^b (g''(x))^2 dx \leq \int_a^b (h''(x))^2 dx, \quad \text{for all } h \in E(f),$$

with equality if and only if $h = g$.

Proof. The proof of Theorem 5.12 carries over to this case. We only need to observe that the natural boundary conditions imply that $g'' \in \mathbb{S}_{1,\tau}$. ■

From this it should be clear that the cubic spline interpolants with Hermite and natural end conditions are extraordinary functions. If we consider all continuous functions with two continuous derivatives that interpolate f at the x_i , the cubic spline interpolant with natural end conditions is the one with the smallest second derivative in the sense that the integral of the square of the second derivative is minimized. This explains why the N boundary conditions in (5.12) are called natural. If we restrict the interpolant to have the same derivative as f at the ends, the solution is still a cubic spline.

For the free end interpolant we will show existence and uniqueness in the next section. No minimization property is known for this spline.

5.2.1 Interpretations of cubic spline interpolation

Today engineers use computers to fit curves through their data points; this is one of the main applications of splines. But splines have been used for this purpose long before computers were available, except that at that time the word spline had a different meaning. In industries like for example ship building, a thin flexible ruler was used to draw curves. The ruler could be clamped down at fixed data points and would then take on a nice smooth shape that interpolated the data and minimized the bending energy in accordance with the physical laws. This allowed the user to interpolate the data in a visually pleasing way. This flexible ruler was known as a *draftmans spline*.

The physical laws governing the classical spline used by ship designers tell us that the ruler will take on a shape that minimizes the total bending energy. The linearised bending energy is given by $\int g''^2$, where $g(x)$ is the position of the centreline of the ruler. Outside the first and last fixing points the ruler is unconstrained and will take the shape of a straight line. From this we see that the natural cubic spline models such a linearised ruler. The word spline was therefore a natural choice for the cubic interpolants we have considered here when they were first studied systematically in 1940's.

The cubic spline interpolant also has a related, geometric interpretation. From differential geometry we know that the curvature of a function $g(x)$ is given by

$$\kappa(x) = \frac{g''(x)}{(1 + (g'(x))^2)^{3/2}}.$$

The curvature $\kappa(x)$ measures how much the function curves at x and is important in the study of parametric curves. If we assume that $1 + g'^2 \approx 1$ on $[a, b]$, then $\kappa(x) \approx g''(x)$. The cubic spline interpolants $I_H f$ and $I_N f$ can therefore be interpreted as the interpolants with the smallest linearised curvature.

5.2.2 Numerical solution and examples

If we were just presented with the problem of finding the C^2 function that interpolate a given function at some points and have the smallest second derivative, without the knowledge that we obtained in Section 5.2, we would have to work very hard to write a reliable computer program that could solve the problem. With Theorem 5.15, the most difficult part of the work has been done, so that in order to compute the solution to say Problem 5.10 H, we only have to solve the linear system of equations (5.18). Let us take a closer look at this system. We order the equations so that the boundary conditions correspond to the first and last equation, respectively. Because of the local support property of the B-splines, only a few unknowns appear in each equation, in other words we have a banded linear system. Indeed, since $\tau_{i+3} = x_i$, we see that only $\{B_{j,3}\}_{j=i}^{i+3}$ can be nonzero at x_i . But we note also that x_i is located at the first knot of $B_{i+3,3}$, which means that $B_{i+3,3}(x_i) = 0$. Since we also have $B'_{j,3}(x_1) = 0$ for $j \geq 3$ and $B'_{j,3}(x_m) = 0$ for $j \leq m$, we conclude that the system can be written in the tridiagonal form

$$\mathbf{A}\mathbf{c} = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_2 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m+1} & \alpha_{m+1} & \gamma_{m+1} \\ & & & \beta_{m+2} & \alpha_{m+2} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{m+1} \\ c_{m+2} \end{pmatrix} = \begin{pmatrix} f'(x_1) \\ f(x_1) \\ \vdots \\ f(x_m) \\ f'(x_m) \end{pmatrix} = \mathbf{f}, \quad (5.19)$$

where the elements of $\mathbf{A}$ are given by

$$\begin{aligned} \alpha_1 &= B'_{1,3}(x_1), & \alpha_{m+2} &= B'_{m+2,3}(x_m), \\ \gamma_1 &= B'_{2,3}(x_1), & \beta_{m+2} &= B'_{m+1,3}(x_m), \\ \beta_{i+1} &= B_{i,3}(x_i), & \alpha_{i+1} &= B_{i+1,3}(x_i), & \gamma_{i+1} &= B_{i+2,3}(x_i). \end{aligned} \quad (5.20)$$

The elements of $\mathbf{A}$ can be computed by one of the triangular algorithms for B-bases.

For $H_3 f$ we had explicit formulas for the B-spline coefficients that only involved a few function values and derivatives, in other words the approximation was local. In cubic spline interpolation the situation is quite different. All the equations in (5.19) are coupled and we have to solve a linear system of equations. Each coefficient will therefore in general depend on all the given function values which means that the value of the interpolant at a point also depends on all the given function values. This means that cubic spline interpolation is not a local process.

Numerically it is quite simple to solve (5.19). It follows from the proof of Theorem 5.13 that the matrix $\mathbf{A}$ is nonsingular, since otherwise the solution could not be unique. Since it has a tridiagonal form it is recommended to use Gaussian elimination. It can be shown that the elimination can be carried out without changing the order of the equations (pivoting), and a detailed error analysis shows that this process is numerically stable.

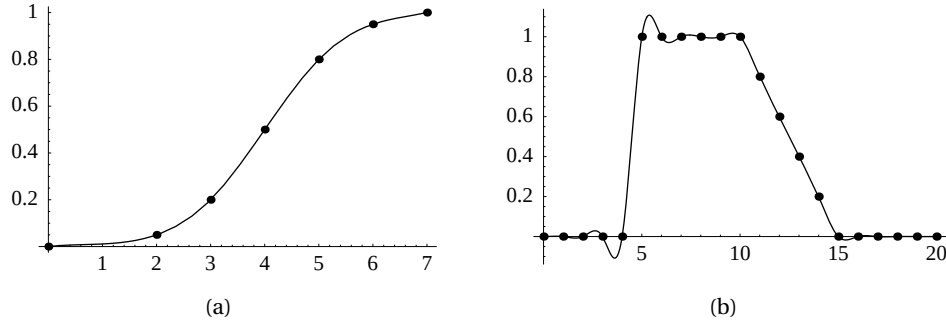


Figure 5.3. Cubic spline interpolation to smoothly varying data (a) and data with sharp corners (b).

In most cases, the underlying function f is only known through the data $y_i = f(x_i)$, for $i = 1, \dots, m$. We can still use Hermite end conditions if we estimate the end slopes $f'(x_1)$ and $f'(x_m)$. A simple estimate is $f'(a) = d_1$ and $f'(b) = d_2$, where

$$d_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad \text{and} \quad d_2 = \frac{f(x_m) - f(x_{m-1})}{x_m - x_{m-1}}. \quad (5.21)$$

More elaborate estimates like those in Section 5.1.3 are of course also possible.

Another possibility is to turn to natural and free boundary conditions which also lead to linear systems similar to the one in equation (5.19), except that the first and last equations which correspond to the boundary conditions must be changed appropriately. For natural end conditions we know from Theorem 5.15 that there is a unique solution. Existence and uniqueness of the solution with free end conditions is established in Corollary 5.19.

The free end condition is particularly attractive in a B-spline formulation, since by not giving any knot at x_2 and x_{m-1} these conditions take care of themselves. The free end conditions work well in many cases, but extra wiggles can sometimes occur near the ends of the range. The Hermite conditions give us better control in this respect.

Example 5.16. In Figure 5.3 (a) and 5.3 (b) we show two examples of cubic spline interpolation. In both cases we used the Hermite boundary conditions with the estimate in (5.21) for the slopes. The data to be interpolated is shown as bullets. Note that in Figure 5.3 (a) the interpolant behaves very nicely and predictably between the data points.

In comparison, the interpolant in Figure 5.3 (b) has some unexpected wiggles. This is a characteristic feature of spline interpolation when the data have sudden changes or sharp corners. For such data, least squares approximation by splines usually gives better results, see Section 5.3.2.

5.3 General Spline Approximation

So far, we have mainly considered spline approximation methods tailored to specific degrees. In practise, cubic splines are undoubtedly the most common, but there is an obvious advantage to have methods available for splines of all degrees. In this section we first consider spline interpolation for splines of arbitrary degree. The optimal properties of the cubic spline interpolant can be generalized to spline interpolants of any odd degree, but here we only focus on the practical construction of the interpolant. Least squares approximation,

which we study in Section 5.3.2, is a completely different approximation procedure that often give better results than interpolation, especially when the data changes abruptly like in Figure 1.16 (b).

5.3.1 Spline interpolation

Given points $(x_i, y_i)_{i=1}^m$, we again consider the problem of finding a spline g such that

$$g(x_i) = y_i, \quad i = 1, \dots, m.$$

In the previous section we used cubic splines where the knots of the spline were located at the data points. This works well if the data points are fairly evenly spaced, but can otherwise give undesirable effects. In such cases the knots should not be chosen at the data points. However, how to choose good knots in general is difficult.

In some cases we might also be interested in doing interpolation with splines of degree higher than three. We could for example be interested in a smooth representation of the second derivative of f . However, if we want f''' to be continuous, say, then the degree d must be higher than three. We therefore consider the following interpolation problem.

Problem 5.17. Let there be given data $(x_i, y_i)_{i=1}^m$ and a spline space $\mathbb{S}_{d,\tau}$ whose knot vector $\tau = (\tau_i)_{i=1}^{m+d+1}$ satisfies $\tau_{i+d+1} > \tau_i$, for $i = 1, \dots, m$. Find a spline g in $\mathbb{S}_{d,\tau}$ such that

$$g(x_i) = \sum_{j=1}^m c_j B_{j,d}(x_i) = y_i, \quad \text{for } i = 1, \dots, m. \quad (5.22)$$

The equations in (5.22) form a system of m equations in m unknowns. In matrix form these equations can be written

$$\mathbf{A}\mathbf{c} = \begin{pmatrix} B_{1,d}(x_1) & \dots & B_{m,d}(x_1) \\ \vdots & \ddots & \vdots \\ B_{1,d}(x_m) & \dots & B_{m,d}(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \mathbf{y}. \quad (5.23)$$

Theorem 5.18 gives necessary and sufficient conditions for this system to have a unique solution, in other words for $\mathbf{A}$ to be nonsingular.

Theorem 5.18. The matrix $\mathbf{A}$ in (5.23) is nonsingular if and only if the diagonal elements $a_{i,i} = B_{i,d}(x_i)$ are positive for $i = 1, \dots, m$.

Proof. See Theorem 10.6 in Chapter 10. ■

The condition that the diagonal elements of $\mathbf{A}$ should be nonzero can be written

$$\tau_i < x_i < \tau_{i+d+1}, \quad i = 1, 2, \dots, m, \quad (5.24)$$

provided we allow $x_i = \tau_i$ if $\tau_i = \dots = \tau_{i+d}$. Conditions (5.24) are known as the *Schoenberg-Whitney nesting conditions*.

As an application of Theorem 5.18, let us verify that the coefficient matrix for cubic spline interpolation with free end conditions is nonsingular.

Corollary 5.19. *Cubic spline interpolation with free end conditions (Problem 5.10 F) has a unique solution.*

Proof. The coefficients of the interpolant are found by solving a linear system of equations of the form (5.22). Recall that the knot vector $\tau = \tau_F$ is given by

$$\tau = (\tau_i)_{i=1}^{m+4} = (x_1, x_1, x_1, x_1, x_3, x_4, \dots, x_{m-2}, x_m, x_m, x_m, x_m).$$

From this we note that $B_1(x_1)$ and $B_2(x_2)$ are both positive. Since $\tau_{i+2} = x_i$ for $i = 3, \dots, m-2$, we also have $\tau_i < x_{i-1} < \tau_{i+4}$ for $3 \leq i \leq m-2$. The last two conditions follow similarly, so the coefficient matrix is nonsingular. ■

For implementation of general spline interpolation, it is important to make use of the fact that at most $d+1$ B-splines are nonzero for a given x , just like we did for cubic spline interpolation. This means that in any row of the matrix A in (5.22), at most $d+1$ entries are nonzero, and those entries are consecutive. This gives A a band structure that can be exploited in Gaussian elimination. It can also be shown that nothing is gained by rearranging the equations or unknowns in Gaussian elimination, so the equations can be solved without pivoting.

5.3.2 Least squares approximation

In this chapter we have described a number of spline approximation techniques based on interpolation. If it is an absolute requirement that the spline should pass exactly through the data points, there is no alternative to interpolation. But such perfect interpolation is only possible if all computations can be performed without any round-off error. In practise, all computations are done with floating

point numbers, and round-off errors are inevitable. A small error is

therefore always present and must be tolerable whenever computers are used for approximation. The question is what is a tolerable error? Often the data are results of measurements with a certain known resolution. To interpolate such data is not recommended since it means that the error is also approximated. If it is known that the underlying function is smooth, it is usually better to use a method that will only approximate the data, but approximate in such a way that the error at the data points is minimized. Least squares approximation is a general and simple approximation method for accomplishing this. The problem can be formulated as follows.

Problem 5.20. *Given data $(x_i, y_i)_{i=1}^m$ with $x_1 < \dots < x_m$, positive real numbers w_i for $i = 1, \dots, m$, and an n -dimensional spline space $\mathbb{S}_{d,\tau}$, find a spline g in $\mathbb{S}_{d,\tau}$ which solves the minimization problem*

$$\min_{h \in \mathbb{S}_{d,\tau}} \sum_{i=1}^m w_i (y_i - h(x_i))^2. \quad (5.25)$$

The expression (5.25) that is minimized is a sum of the squares of the errors at each data point, weighted by the numbers w_i which are called *weights*. This explains the name *least squares approximation*, or more precisely *weighted least squares approximation*. If w_i is large in comparison to the other weights, the error $y_i - h(x_i)$ will count more in the minimization.

As the the weight grows, the error at this data point will go to zero. On the other hand, if the weight is small in comparison to the other weights, the error at that data point gives little contribution to the total least squares deviation. If the weight is zero, the approximation is completely independent of the data point. Note that the actual value of the weights is irrelevant, it is the relative size that matters. The weights therefore provides us with the opportunity to attach a measure of confidence to each data point. If we know that y_i is a very accurate data value we can give it a large weight, while if y_i is very inaccurate we can give it a small weight. Note that it is the relative size of the weights that matters, a natural ‘neutral’ value is therefore $w_i = 1$.

From our experience with interpolation, we see that if we choose the spline space $\mathbb{S}_{d,\tau}$ so that the number of B-splines equals the number of data points and such that $B_i(x_i) > 0$ for all i , then the least squares approximation will agree with the interpolant and give zero error, at least in the absence of round-off errors. Since the

whole point of introducing the least squares approximation is to avoid interpolation of the data, we must make sure that n is smaller than m and that the knot vector is appropriate. This all means that the spline space $\mathbb{S}_{d,\tau}$ must be chosen appropriately, but this is not easy. Of course we would like the spline space to be such that a “good” approximation g can be found. Good, will have different interpretations for different applications. A statistician would like g to have certain statistical properties. A designer would like an aesthetically pleasing curve, and maybe some other shape and tolerance requirements to be satisfied. In practise, one often starts with a small spline space, and then adds knots in problematic areas until hopefully a satisfactory approximation is obtained.

Different points of view are possible in order to analyse Problem 5.20 mathematically. Our approach is based on linear algebra. Our task is to find the vector $\mathbf{c} = (c_1, \dots, c_n)$ of B-spline coefficients of the spline g solving Problem 5.20. The following matrix-vector formulation is convenient.

Lemma 5.21. *Problem 5.20 is equivalent to the linear least squares problem*

$$\min_{\mathbf{c} \in \mathbb{R}^n} \|\mathbf{A}\mathbf{c} - \mathbf{b}\|^2,$$

where $\mathbf{A} \in \mathbb{R}^{m,n}$ and $\mathbf{b} \in \mathbb{R}^m$ have components

$$a_{i,j} = \sqrt{w_i} B_j(x_i) \quad \text{and} \quad b_i = \sqrt{w_i} y_i, \quad (5.26)$$

and for any $\mathbf{u} = (u_1, \dots, u_m)$,

$$\|\mathbf{u}\| = \sqrt{u_1^2 + \dots + u_m^2},$$

is the usual Euclidean length of a vector in $\mathbb{R}^m$.

Proof. Suppose $\mathbf{c} = (c_1, \dots, c_n)$ are the B-spline coefficients of some $h \in \mathbb{S}_{d,\mathbf{r}}$. Then

$$\begin{aligned}\|\mathbf{Ac} - \mathbf{b}\|_2^2 &= \sum_{i=1}^m \left(\sum_{j=1}^n a_{i,j} c_j - b_i \right)^2 \\ &= \sum_{i=1}^m \left(\sum_{j=1}^n \sqrt{w_i} B_j(x_i) c_j - \sqrt{w_i} y_i \right)^2 \\ &= \sum_{i=1}^m w_i \left(h(x_i) - y_i \right)^2.\end{aligned}$$

This shows that the two minimization problems are equivalent. ■

In the next lemma, we collect some facts about the general linear least squares problem. Recall that a symmetric matrix $\mathbf{N}$ is positive semidefinite if $\mathbf{c}^T \mathbf{N} \mathbf{c} \geq 0$ for all $\mathbf{c} \in \mathbb{R}^n$, and positive definite if in addition $\mathbf{c}^T \mathbf{N} \mathbf{c} > 0$ for all nonzero $\mathbf{c} \in \mathbb{R}^n$.

Lemma 5.22. Suppose m and n are positive integers with $m \geq n$, and let the matrix $\mathbf{A}$ in $\mathbb{R}^{m,n}$ and the vector $\mathbf{b}$ in $\mathbb{R}^m$ be given. The linear least squares problem

$$\min_{\mathbf{c} \in \mathbb{R}^n} \|\mathbf{Ac} - \mathbf{b}\|^2 \quad (5.27)$$

always has a solution $\mathbf{c}^*$ which can be found by solving the linear set of equations

$$\mathbf{A}^T \mathbf{Ac}^* = \mathbf{A}^T \mathbf{b}. \quad (5.28)$$

The coefficient matrix $\mathbf{N} = \mathbf{A}^T \mathbf{A}$ is symmetric and positive semidefinite. It is positive definite, and therefore nonsingular, and the solution of (5.27) is unique if and only if $\mathbf{A}$ has linearly independent columns.

Proof. Let $\text{span}(\mathbf{A})$ denote the n -dimensional linear subspace of $\mathbb{R}^m$ spanned by the columns of $\mathbf{A}$,

$$\text{span}(\mathbf{A}) = \{\mathbf{Ac} \mid \mathbf{c} \in \mathbb{R}^n\}.$$

From basic linear algebra we know that a vector $\mathbf{b} \in \mathbb{R}^m$ can be written uniquely as a sum $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where $\mathbf{b}_1$ is a linear combination of the columns of $\mathbf{A}$ so that $\mathbf{b}_1 \in \text{span}(\mathbf{A})$, and $\mathbf{b}_2$ is orthogonal to $\text{span}(\mathbf{A})$, i.e., we have $\mathbf{b}_2^T \mathbf{d} = 0$ for all $\mathbf{d}$ in $\text{span}(\mathbf{A})$. Using this decomposition of $\mathbf{b}$, and the Pythagorean theorem, we have for any $\mathbf{c} \in \mathbb{R}^n$,

$$\|\mathbf{Ac} - \mathbf{b}\|^2 = \|\mathbf{Ac} - \mathbf{b}_1 - \mathbf{b}_2\|^2 = \|\mathbf{Ac} - \mathbf{b}_1\|^2 + \|\mathbf{b}_2\|^2.$$

It follows that $\|\mathbf{Ac} - \mathbf{b}\|_2^2 \geq \|\mathbf{b}_2\|_2^2$ for any $\mathbf{c} \in \mathbb{R}^n$, with equality if $\mathbf{Ac} = \mathbf{b}_1$. A $\mathbf{c} = \mathbf{c}^*$ such that $\mathbf{Ac}^* = \mathbf{b}_1$ clearly exists since $\mathbf{b}_1$ is in $\text{span}(\mathbf{A})$, and $\mathbf{c}^*$ is unique if and only if $\mathbf{A}$ has linearly independent columns. To derive the linear system for $\mathbf{c}^*$, we note that any $\mathbf{c}$ that is minimising satisfies $\mathbf{Ac} - \mathbf{b} = -\mathbf{b}_2$. Since we also know that $\mathbf{b}_2$ is orthogonal to $\text{span}(\mathbf{A})$, we must have

$$\mathbf{d}^T (\mathbf{Ac} - \mathbf{b}) = \mathbf{c}_1^T \mathbf{A}^T (\mathbf{Ac} - \mathbf{b}) = 0$$

for all $\mathbf{d} = \mathbf{Ac}_1$ in $\text{span}(\mathbf{A})$, i.e., for all $\mathbf{c}_1$ in $\mathbb{R}^n$. But this is only possible if $\mathbf{A}^T (\mathbf{Ac} - \mathbf{b}) = 0$. This proves (5.28).

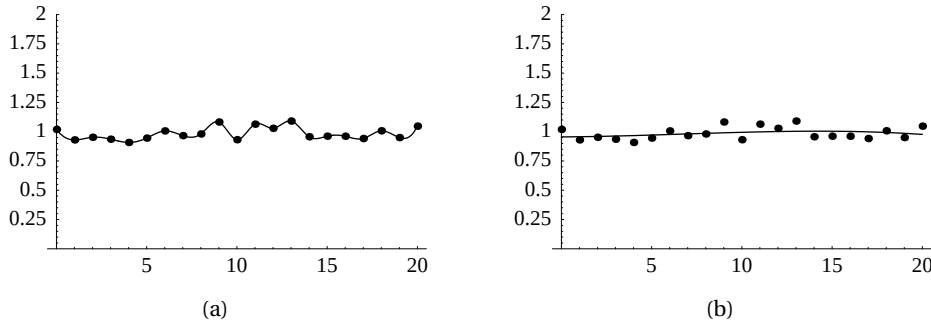


Figure 5.4. Figure (a) shows the cubic spline interpolation to the noisy data of Example 5.24, while least squares approximation to the same data is shown in (b).

The $n \times n$ -matrix $N = A^T A$ is clearly symmetric and

$$\mathbf{c}^T N \mathbf{c} = \|A\mathbf{c}\|_2^2 \geq 0, \quad (5.29)$$

for all $\mathbf{c} \in \mathbb{R}^n$, so that N is positive semi-definite. From (5.29) we see that we can find a nonzero $\mathbf{c}$ such that $\mathbf{c}^T N \mathbf{c} = 0$ if and only if $A\mathbf{c} = 0$, i.e., if and only if A has linearly dependent columns. We conclude that N is positive definite if and only if A has linearly independent columns. ■

Applying these results to Problem 5.20 we obtain.

Theorem 5.23. *Problem 5.20 always has a solution. The solution is unique if and only if we can find a sub-sequence $(x_{i_\ell})_{\ell=1}^n$ of the data abscissas such that*

$$B_\ell(x_{i_\ell}) \neq 0 \quad \text{for } \ell = 1, \dots, n.$$

Proof. By Lemma 5.21 and Lemma 5.22 we conclude that Problem 5.20 always has a solution, and the solution is unique if and only if the matrix A in Lemma 5.21 has linearly independent columns. Now A has linearly independent columns if and only if we can find a subset of n rows of A such that the square submatrix consisting of these rows and all columns of A is nonsingular. But such a matrix is of the form treated in Theorem 5.18. Therefore, the submatrix is nonsingular if and only if the diagonal elements are nonzero. But the diagonal elements are given by $B_\ell(x_{i_\ell})$. ■

Theorem 5.23 provides a nice condition for checking that we have a unique least squares spline approximation to a given data set; we just have to check that each B-spline has its ‘own’ x_{i_ℓ} in its support. To find the B-spline coefficients of the approximation, we must solve the linear system of equations (5.28). These equations are called the *normal equations* of the least squares system and can be solved by Cholesky factorization of a banded matrix followed by back substitution. The least squares problem can also be solved by computing a QR-factorization of the matrix A ; for both methods we refer to a standard text on numerical linear algebra for details.

Example 5.24. Least squares approximation is especially appropriate when the data is known to be noisy. Consider the data represented as bullets in Figure 5.4 (a). These data were obtained by adding random perturbations in the interval $[-0.1, 0.1]$ to the function $f(x) = 1$. In Figure 5.4 (a) we show the cubic spline interpolant (with free end conditions) to the data, while Figure 5.4 (b) shows the cubic least squares approximation to the same data, using no interior knots. We see that the least squares approximation smooths out the data nicely. We also see that the cubic spline interpolant gives a nice approximation to the given data, but it also reproduces the noise that was added artificially.

Once we have made the choice of approximating the data in Example 5.24 using cubic splines with no interior knots, we have no chance of representing the noise in the data. The flexibility of cubic polynomials is nowhere near rich enough to represent all the oscillations that we see in Figure 5.4 (a), and this gives us the desired smoothing effect in Figure 5.4 (b). The advantage of the method of least squares is that it gives a reasonably simple method for computing a reasonably good approximation to quite arbitrary data on quite arbitrary knot vectors. But it is largely the knot vector that decides how much the approximation is allowed to oscillate, and good methods for choosing the knot vector is therefore of fundamental importance. Once the knot vector is given there are in fact many approximation methods that will provide good approximations.

5.4 The Variation Diminishing Spline Approximation

In this section we describe a simple, but very useful method for obtaining spline approximations to a function f defined on an interval $[a, b]$. This method is a generalization of piecewise linear interpolation and has a nice shape preserving behaviour. For example, if the function f is positive, then the spline approximation will also be positive.

Definition 5.25. Let f be a given continuous function on the interval $[a, b]$, let d be a given positive integer, and let $\tau = (\tau_1, \dots, \tau_{n+d+1})$ be a $d+1$ -regular knot vector with boundary knots $\tau_{d+1} = a$ and $\tau_{n+1} = b$. The spline given by

$$(Vf)(x) = \sum_{j=1}^n f(\tau_j^*) B_{j,d}(x) \quad (5.30)$$

where $\tau_j^* = (\tau_{j+1} + \dots + \tau_{j+d})/d$ are the knot averages, is called the Variation Diminishing Spline Approximation of degree d to f on the knot vector τ .

The approximation method that assigns to f the spline approximation Vf is about the simplest method of approximation that one can imagine. Unlike some of the other methods discussed in this chapter there is no need to solve a linear system. To obtain Vf , we simply evaluate f at certain points and use these function values as B-spline coefficients directly.

Note that if all interior knots occur less than $d+1$ times in τ , then

$$a = \tau_1^* < \tau_2^* < \dots < \tau_{n-1}^* < \tau_n^* = b. \quad (5.31)$$

This is because τ_1 and τ_{n+d+1} do not occur in the definition of τ_1^* and τ_n^* so that all selections of d consecutive knots must be different.

Example 5.26. Suppose that $d = 3$ and that the interior knots of τ are uniform in the interval $[0, 1]$, say

$$\tau = (0, 0, 0, 0, 1/m, 2/m, \dots, 1 - 1/m, 1, 1, 1, 1). \quad (5.32)$$

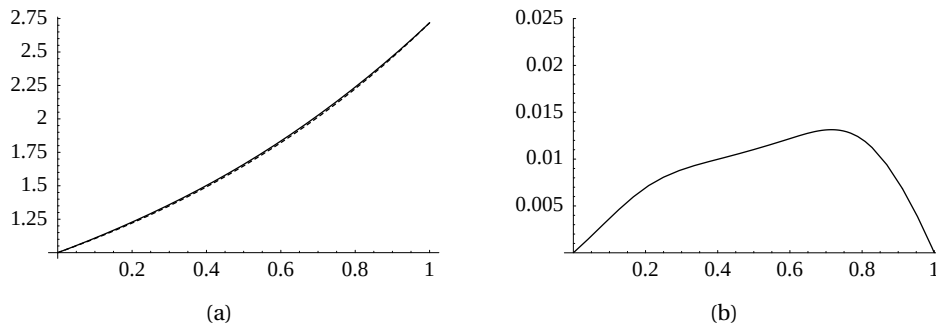


Figure 5.5. The exponential function together with the cubic variation diminishing approximation of Example 5.26 in the special case $m = 5$ is shown in (a). The error in the approximation is shown in (b).

For $m \geq 2$ we then have

$$\tau^* = (0, 1/(3m), 1/m, 2/m, \dots, 1 - 1/m, 1 - 1/(3m), 1). \quad (5.33)$$

Figure 5.5 (a) shows the cubic variation diminishing approximation to the exponential function on the knot vector in (5.32) with $m = 5$, and the error is shown in Figure 5.5 (b). The error is so small that it is difficult to distinguish between the two functions in Figure 5.5 (a).

The variation diminishing spline can also be used to approximate functions with singularities, that is, functions with discontinuities in a derivative of first or higher orders.

Example 5.27. Suppose we want to approximate the function

$$f(x) = 1 - e^{-50|x|}, \quad x \in [-1, 1], \quad (5.34)$$

by a cubic spline Vf . In order to construct a suitable knot vector, we take a closer look at the function, see Figure 5.6 (a). The graph of f has a cusp at the origin, so f' is discontinuous and changes sign there. Our spline approximation should therefore also have some kind of singularity at the origin. Recall from Theorem 3.19 that a B-spline can have a discontinuous first derivative at a knot provided the knot has multiplicity at least d . Since we are using cubic splines, we therefore place a triple knot at the origin. The rest of the interior knots are placed uniformly in $[-1, 1]$. A suitable knot vector is therefore

$$\tau = (-1, -1, -1, -1, -1 + 1/m, \dots, -1/m, 0, 0, 0, 1/m, \dots, 1 - 1/m, 1, 1, 1). \quad (5.35)$$

The integer m is a parameter which is used to control the number of knots and thereby the accuracy of the approximation. The spline Vf is shown in Figure 5.6 (a) for $m = 4$ together with the function f itself. The error is shown in Figure 5.6 (b), and we note that the error is zero at $x = 0$, but quite large just outside the origin.

Figures 5.6 (c) and 5.6 (d) show the first and second derivatives of the two functions, respectively. Note that the sign of f and its derivatives seem to be preserved by the variation diminishing spline approximation.

The variation diminishing spline approximation is a very simple procedure for obtaining spline approximations. In Example 5.27 we observed that the approximation has the same sign as f everywhere, and more than this, even the sign of the first two derivatives is preserved in passing from f to the approximation Vf . This is important since the sign of the derivative gives important information about the shape of the graph of the function. A non-negative derivative for example, means that the function is nondecreasing, while a nonnegative second derivative roughly means that the function is convex, in other words it curves in

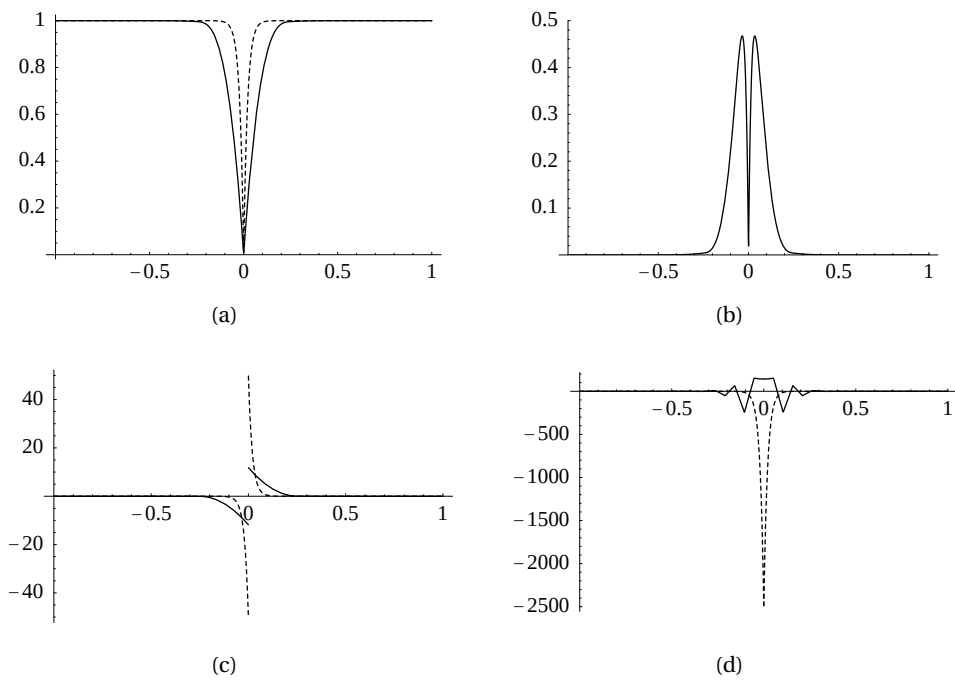


Figure 5.6. Figure (a) shows the function $f(x) = 1 - e^{-50|x|}$ (dashed) and its cubic variation diminishing spline approximation (solid) on the knot vector described in Example 5.27, and the error in the approximation is shown in Figure (b). The first derivative of the two functions is shown in (c), and the second derivatives in (d).

the same direction everywhere. Approximation methods that preserve the sign of the derivative are therefore important in practical modelling of curves. We will now study such *shape preservation* in more detail.

5.4.1 Preservation of bounds on a function

Sometimes it is important that the maximum and minimum values of a function are preserved under approximation. Splines have some very useful properties in this respect.

Lemma 5.28. *Let g be a spline in some spline space $\mathbb{S}_{d,\tau}$ of dimension n . Then g is bounded by its smallest and largest B-spline coefficients,*

$$\min_i \{c_i\} \leq \sum_i c_i B_i(x) \leq \max_i \{c_i\}, \quad \text{for all } x \in [\tau_{d+1}, \tau_{n+1}). \quad (5.36)$$

Proof. Let $c_{\max}$ be the largest coefficient. Then we have

$$\sum_i c_i B_i(x) \leq \sum_i c_{\max} B_i(x) = c_{\max} \sum_i B_i(x) = c_{\max},$$

since $\sum_i B_i(x) = 1$. This proves the second inequality in (5.36). The proof of the first inequality is similar. ■

Note that this lemma only says something interesting if $n \geq d+1$. Any plot of a spline with its control polygon will confirm the inequalities (5.36), see for example Figure 5.7.

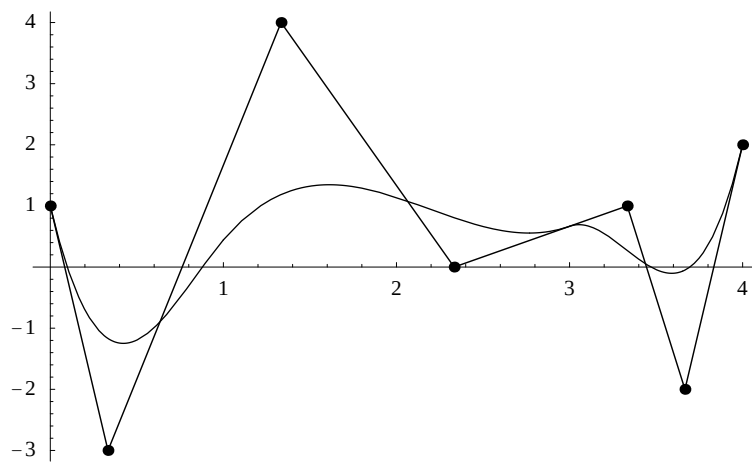


Figure 5.7. A cubic spline with its control polygon. Note how the extrema of the control polygon bound the extrema of the spline.

With Lemma 5.28 we can show that bounds on a function are preserved by its variation diminishing approximation.

Proposition 5.29. *Let f be a function that satisfies*

$$f_{\min} \leq f(x) \leq f_{\max} \quad \text{for all } x \in \mathbb{R}.$$

Then the variation diminishing spline approximation to f from some spline space $\mathbb{S}_{d,\tau}$ has the same bounds,

$$f_{\min} \leq (Vf)(x) \leq f_{\max} \quad \text{for all } x \in \mathbb{R}. \quad (5.37)$$

Proof. Recall that the B-spline coefficients c_i of Vf are given by

$$c_i = f(\tau_i^*) \quad \text{where } \tau_i^* = (\tau_{i+1} + \cdots + \tau_{i+d})/d.$$

We therefore have that all the B-spline coefficients of Vf are bounded below by $f_{\min}$ and above by $f_{\max}$. The inequalities in (5.37) therefore follow as in Lemma 5.28. ■

5.4.2 Preservation of monotonicity

Many geometric properties of smooth functions can be characterized in terms of the derivative of the function. In particular, the sign of the derivative tells us whether the function is increasing or decreasing. The variation diminishing approximation also preserves information about the derivatives in a very convenient way. Let us first define exactly what we mean by increasing and decreasing functions.

Definition 5.30. A function f defined on an interval $[a, b]$ is increasing if the inequality $f(x_0) \leq f(x_1)$ holds for all x_0 and x_1 in $[a, b]$ with $x_0 < x_1$. It is decreasing if $f(x_0) \geq f(x_1)$ for all x_0 and x_1 in $[a, b]$ with $x_0 < x_1$. A function that is increasing or decreasing is said to be monotone.

The two properties of being increasing and decreasing are clearly completely symmetric and we will only prove results about increasing functions.

If f is differentiable, monotonicity can be characterized in terms of the derivative.

Proposition 5.31. A differentiable function is increasing if and only if its derivative is non-negative.

Proof. Suppose that f is increasing. Then $(f(x+h) - f(x))/h \geq 0$ for all x and positive h such that both x and $x+h$ are in $[a, b]$. Taking the limit as h tends to zero, we must have $f'(x) \geq 0$ for an increasing differentiable function. At $x = b$ a similar argument with negative h may be used.

If the derivative of f is nonnegative, let x_0 and x_1 be two distinct points in $[a, b]$ with $x_0 < x_1$. The mean value theorem then states that

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(\theta)$$

for some $\theta \in (x_0, x_1)$. Since $f'(\theta) \geq 0$, we conclude that $f(x_1) \geq f(x_0)$. ■

Monotonicity of a spline can be characterized in terms of some simple conditions on its B-spline coefficients.

Proposition 5.32. Let τ be a $d+1$ -extended knot vector for splines on the interval $[a, b] = [\tau_{d+1}, \tau_{n+1}]$, and let $g = \sum_{i=1}^n c_i B_i$ be a spline in $\mathbb{S}_{d,\tau}$. If the coefficients are increasing, that is $c_i \leq c_{i+1}$ for $i = 1, \dots, n-1$, then g is increasing. If the coefficients are decreasing then g is decreasing.

Proof. The proposition is certainly true for $d = 0$, so we can assume that $d \geq 1$. Suppose first that there are no interior knots in τ of multiplicity $d + 1$. If we differentiate g we find $g'(x) = \sum_{i=1}^n \Delta c_i B_{i,d-1}(x)$ for $x \in [a, b]$, where

$$\Delta c_i = d \frac{c_i - c_{i-1}}{\tau_{i+d} - \tau_i}.$$

Since all the coefficients of g' are nonnegative we must have $g'(x) \geq 0$ (really the one sided derivative from the right) for $x \in [a, b]$. Since we have assumed that there are no knots of multiplicity $d + 1$ in (a, b) , we know that g is continuous and hence that it must be increasing.

If there is an interior knot at $\tau_i = \tau_{i+d}$ of multiplicity $d + 1$, we conclude from the above that g is increasing on both sides of τ_i . But we also know that $g(\tau_i) = c_i$ while the limit of g from the left is c_{i-1} . The jump is therefore $c_i - c_{i-1}$ which is nonnegative so g increases across the jump. ■

An example of an increasing spline with its control polygon is shown in Figure 5.8 (a). Its derivative is shown in Figure 5.8 (b) and is, as expected, positive.

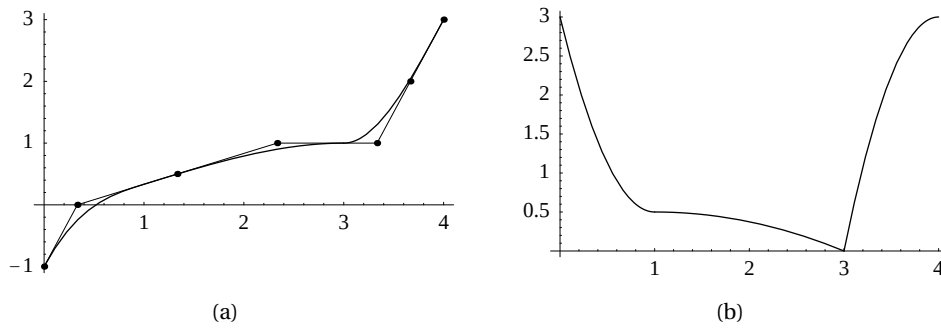


Figure 5.8. An increasing cubic spline (a) and its derivative (b).

From Proposition 5.32 it is now easy to deduce that Vf preserves monotonicity in f .

Proposition 5.33. *Let f be function defined on the interval $[a, b]$ and let τ be a $d + 1$ -extended knot vector with $\tau_{d+1} = a$ and $\tau_{n+1} = b$. If f is increasing (decreasing) on $[a, b]$, then the variation diminishing approximation Vf is also increasing (decreasing) on $[a, b]$.*

Proof. The variation diminishing approximation Vf has as its i 'th coefficient $c_i = f(t_i^*)$, and since f is increasing these coefficients are also increasing. Proposition 5.32 then shows that Vf is increasing. ■

That Vf preserves monotonicity means that the oscillations we saw could occur in spline interpolation are much less pronounced in the variation diminishing spline approximation. In fact, we shall also see that Vf preserves the sign of the second derivative of f which reduces further the possibility of oscillations.

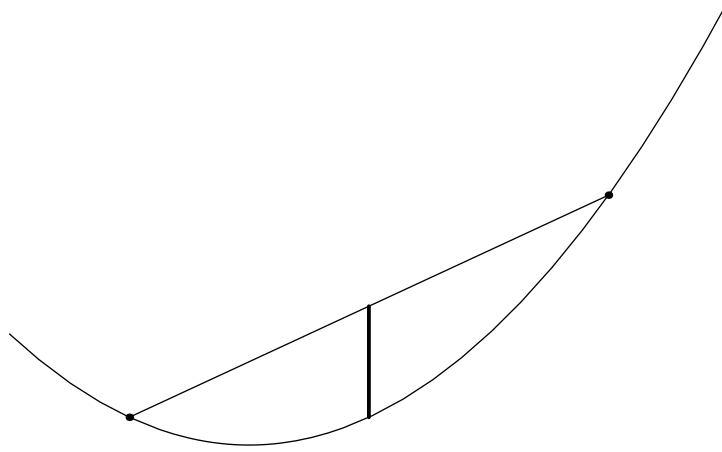


Figure 5.9. A convex function and the cord connecting two points on the graph.

5.4.3 Preservation of convexity

From elementary calculus, we know that the sign of the second derivative of a function tells us in whether the function curves upward or downward, or whether the function is *convex* or *concave*. These concepts can be defined for functions that have no a priori smoothness.

Definition 5.34. A function f is said to be convex on an interval (a, b) if

$$f((1 - \lambda)x_0 + \lambda x_2) \leq (1 - \lambda)f(x_0) + \lambda f(x_2) \quad (5.38)$$

for all x_0 and x_2 in $[a, b]$ and for all λ in $[0, 1]$. If $-f$ is convex then f is said to be concave.

From Definition 5.34 we see that f is convex if the line between two points on the graph of f is always above the graph, see Figure 5.9. It therefore ‘curves upward’ just like smooth functions with nonnegative second derivative.

Convexity can be characterized in many different ways, some of which are listed in the following lemma.

Lemma 5.35. Let f be a function defined on the open interval (a, b) .

1. The function f is convex if and only if

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} \leq \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad (5.39)$$

for all x_0, x_1 and x_2 in (a, b) with $x_0 < x_1 < x_2$.

2. If f is differentiable on (a, b) , it is convex if and only if $f'(y_0) \leq f'(y_1)$ when $a < y_0 < y_1 < b$, that is, the derivative of f is increasing.
3. If f is two times differentiable it is convex if and only if $f''(x) \geq 0$ for all x in (a, b) .

Proof. Let $\lambda = (x_1 - x_0)/(x_2 - x_0)$ so that $x_1 = (1 - \lambda)x_0 + \lambda x_2$. Then (5.38) is equivalent to the inequality

$$(1 - \lambda)(f(x_1) - f(x_0)) \leq \lambda(f(x_2) - f(x_1)).$$

Inserting the expression for λ gives (5.39), so (i) is equivalent to Definition 5.34.

To prove (ii), suppose that f is convex and let y_0 and y_1 be two points in (a, b) with $y_0 < y_1$. From (5.39) we deduce that

$$\frac{f(y_0) - f(x_0)}{y_0 - x_0} \leq \frac{f(y_1) - f(x_1)}{y_1 - x_1},$$

for any x_0 and x_1 with $x_0 < y_0 < x_1 < y_1$. Letting x_0 and x_1 tend to y_0 and y_1 respectively, we see that $f'(y_0) \leq f'(y_1)$.

For the converse, suppose that f' is increasing, and let $x_0 < x_1 < x_2$ be three points in (a, b) . By the mean value theorem we have

$$\frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(\theta_0) \quad \text{and} \quad \frac{f(x_2) - f(x_1)}{x_2 - x_1} = f'(\theta_1),$$

where $x_0 < \theta_0 < x_1 < \theta_1 < x_2$. Since f' is increasing, conclude that (5.39) holds and therefore that f is convex.

For part (iii) we use part (ii) and Proposition 5.31. From (ii) we know that f is convex if and only if f' is increasing, and by Proposition 5.31 we know that f' is increasing if and only if f'' is nonnegative. ■

It may be a bit confusing that the restrictions on $x_0 < x_1 < x_2$ in Lemma 5.35 are stronger than the restrictions on x_0, x_2 and λ in Definition 5.34. But this is only superficial since in the special cases $x_0 = x_2$, and $\lambda = 0$ and $\lambda = 1$, the inequality (5.38) is automatically satisfied.

It is difficult to imagine a discontinuous convex function. This is not so strange since all convex functions are in fact continuous.

Proposition 5.36. *A function that is convex on an open interval is continuous on that interval.*

Proof. Let f be a convex function on (a, b) , and let x and y be two points in some subinterval (c, d) of (a, b) . Using part (i) of Lemma 5.35 repeatedly, we find that

$$\frac{f(c) - f(a)}{c - a} \leq \frac{f(y) - f(x)}{y - x} \leq \frac{f(b) - f(d)}{b - d}. \quad (5.40)$$

Set $M = \max\{(f(c) - f(a))/(c - a), (f(b) - f(d))/(b - d)\}$. Then (5.40) is equivalent to

$$|f(y) - f(x)| \leq M|y - x|.$$

But this means that f is continuous at each point in (c, d) . For if z is in (c, d) we can choose $x = z$ and $y > z$ and obtain that f is continuous from the right at z . Similarly, we can also choose $y = z$ and $x < z$ to find that f is continuous from the left as well. Since (c, d) was arbitrary in (a, b) , we have showed that f is continuous in all of (a, b) . ■

The assumption in Proposition 5.36 that f is defined on an open interval is essential. On the interval $(0, 1]$ for example, the function f that is identically zero except that $f(1) = 1$, is convex, but clearly discontinuous at $x = 1$. For splines however, this is immaterial if we assume a spline to be continuous from the right at the left end of the interval of interest and continuous from the left at the right end of the interval of interest. In addition, since a polynomial never is infinite, we see that our results in this section remain true for splines defined on some closed interval $[a, b]$.

We can now give a simple condition that ensures that a spline function is convex.

Proposition 5.37. *Let τ be a $d+1$ -extended knot vector for some $d \geq 1$, and let $g = \sum_{i=1}^n c_i B_{i,d}$ be a spline in $\mathbb{S}_{d,\tau}$. Define Δc_i by*

$$\Delta c_i = \begin{cases} (c_i - c_{i-1})/(\tau_{i+d} - \tau_i), & \text{if } \tau_i < \tau_{i+d}, \\ \Delta c_{i-1}, & \text{if } \tau_i = \tau_{i+d}; \end{cases}$$

for $i = 2, \dots, n$. Then g is convex on $[\tau_{d+1}, \tau_{n+1}]$ if it is continuous and

$$\Delta c_{i-1} \leq \Delta c_i \quad \text{for } i = 2, \dots, n. \quad (5.41)$$

Proof. Note that $(\Delta c_i)_{i=2}^n$ are the B-spline coefficients of g' on the interval $[\tau_{d+1}, \tau_{n+1}]$, bar the constant d . Since (5.41) ensures that these are increasing, we conclude from Proposition 5.32 that g' is increasing. If g is also differentiable everywhere in $[\tau_{d+1}, \tau_{n+1}]$, part (ii) of Lemma 5.35 shows that g is convex.

In the rest of the proof, the short hand notation

$$\delta(u, v) = \frac{g(v) - g(u)}{v - u}$$

will be convenient. Suppose now that there is only one point z where g is not differentiable, and let $x_0 < x_1 < x_2$ be three points in $[\tau_{d+1}, \tau_{n+1}]$. We must show that

$$\delta(x_0, x_1) \leq \delta(x_1, x_2). \quad (5.42)$$

The case where all three x 's are on one side of z are covered by the first part of the proof. Suppose therefore that $x_0 < z \leq x_1 < x_2$. Since $\delta(u, v) = g'(\theta)$ with $u < \theta < v$ when g is differentiable on $[a, b]$, and since g' is increasing, we certainly have $\delta(x_0, z) \leq \delta(z, x_2)$, so that (5.42) holds in the special case where $x_1 = z$. When $x_1 > z$ we use the simple identity

$$\delta(x_0, x_1) = \delta(x_0, z) \frac{z - x_0}{x_1 - x_0} + \delta(z, x_1) \frac{x_1 - z}{x_1 - x_0},$$

which shows that $\delta(x_0, x_1)$ is a weighted average of $\delta(x_0, z)$ and $\delta(z, x_1)$. But then we have

$$\delta(x_0, x_1) \leq \delta(z, x_1) \leq \delta(x_1, x_2),$$

the first inequality being valid since $\delta(x_0, z) \leq \delta(z, x_1)$ and the second one because g is convex to the right of z . This shows that g is convex.

The case where $x_0 < x_1 < z < x_2$ and the case of several discontinuities can be treated similarly. ■

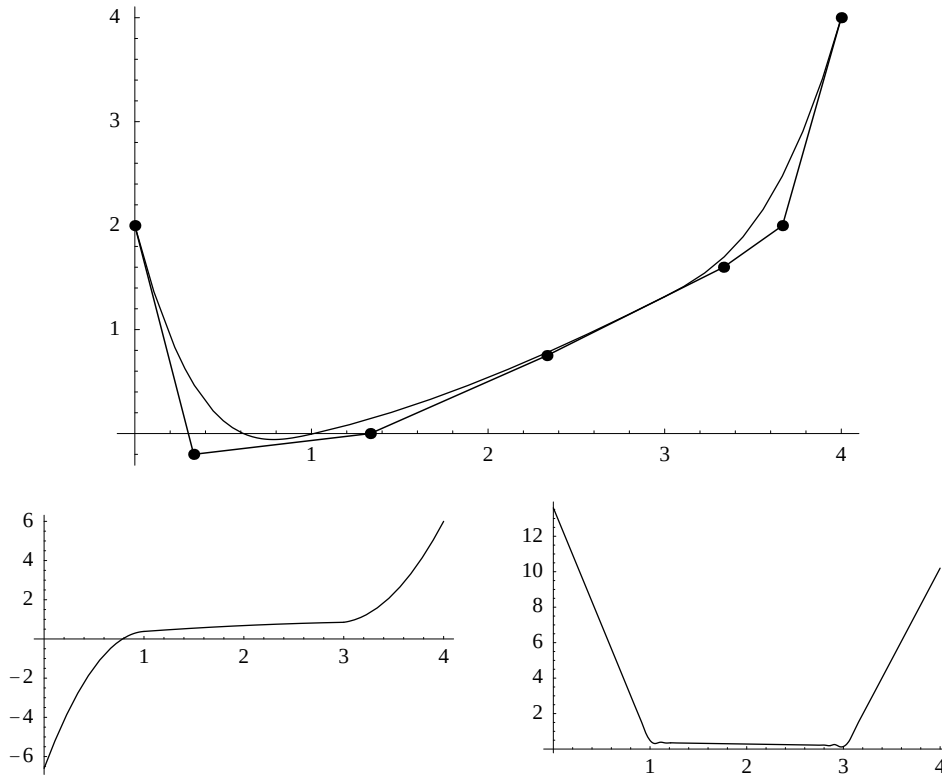


Figure 5.10. A convex spline with its control polygon (a), its first derivative (b) and its second derivative (c).

An example of a convex spline is shown in Figure 5.10, together with its first and second derivatives in.

With Proposition 5.37 at hand, it is simple to show that the variation diminishing spline approximation preserves convexity.

Proposition 5.38. *Let f be a function defined on the interval $[a, b]$, let $d \geq 1$ be an integer, and let τ be a $d + 1$ -extended knot vector with $\tau_{d+1} = a$ and $\tau_{n+1} = b$. If f is convex on $[a, b]$ then Vf is also convex on $[a, b]$.*

Proof. Recall that the coefficients of Vf are $(f(\tau_i^*))_{i=1}^n$ so that the differences in Proposition 5.37 are

$$\Delta c_i = \frac{f(\tau_i^*) - f(\tau_{i-1}^*)}{\tau_{i+d} - \tau_i} = \frac{f(\tau_i^*) - f(\tau_{i-1}^*)}{(\tau_i^* - \tau_{i-1}^*)d},$$

if $\tau_i < \tau_{i+d}$. Since f is convex, these differences must be increasing. Proposition 5.37 then shows that Vf is convex. ■

At this point, we can undoubtedly say that the variation diminishing spline approximation is a truly remarkable method of approximation. In spite of its simplicity, it preserves the shape of f both with regards to convexity, monotonicity and bounds on the function values.

This makes it very attractive as an approximation method in for example design where the shape of a curve is more important than how accurately it approximates given data.

It should be noted that the shape preserving properties of the variation diminishing approximation is due to the properties of B-splines. When we determine Vf we give its control polygon directly by sampling f at the knot averages, and the results that we have obtained about the shape preserving properties of Vf are all consequences of relationships between a spline and its control polygon: *A spline is bounded by the extrema of its control polygon, a spline is monotone if the control polygon is monotone, a spline is convex if the control polygon is convex.* In short: *A spline is a smoothed out version of its control polygon.* We will see many more realisations of this general principle in later chapters

Parametric Spline Curves

When we introduced splines in Chapter 1 we focused on spline curves, or more precisely, vector valued spline functions. In Chapters 2, 3 and 4 we then established the basic theory of spline functions and B-splines, and in Chapter 5 we studied a number of methods for constructing spline functions that approximate given data. In this chapter we return to spline curves and show how the approximation methods in Chapter 5 can be adapted to this more general situation.

We start by giving a formal definition of parametric curves in Section 6.1, and introduce parametric spline curves in Section 6.2.1. In the rest of Section 6.2 we then generalise the approximation methods in Chapter 5 to curves. It turns out that the generalisation is virtually trivial, except for one difficult point.

6.1 Definition of Parametric Curves

In Section 1.2 we gave an intuitive introduction to parametric curves and discussed the significance of different parameterisations. In this section we will give a more formal definition of parametric curves, but the reader is encouraged to first go back and reread Section 1.2 in Chapter 1.

6.1.1 Regular parametric representations

A parametric curve will be defined in terms of *parametric representations*.

Definition 6.1. A vector function or mapping $\mathbf{f} : [a, b] \rightarrow \mathbb{R}^s$ of the interval $[a, b]$ into $\mathbb{R}^s$ for $s \geq 2$ is called a *parametric representation of class C^m* for $m \geq 1$ if each of the s components of $\mathbf{f}$ has continuous derivatives up to order m . If, in addition, the first derivative of $\mathbf{f}$ does not vanish in $[a, b]$,

$$D\mathbf{f}(u) = \mathbf{f}'(u) \neq 0, \quad \text{for } u \in [a, b],$$

then $\mathbf{f}$ is called a *regular parametric representation of class C^m* .

A parametric representation will often be referred to informally as a curve, although the term parametric curve will be given a more precise meaning later. In this chapter we will always assume the parametric representations to be sufficiently smooth for all operations to

make sense. Note that a function $y = h(x)$ always can be considered as a curve through the parametric representation $\mathbf{f}(u) = (u, h(u))$.

If we imagine travelling along the curve and let u denote the elapsed time of our journey, then the length of $\mathbf{f}'(u)$ which we denote by $\|\mathbf{f}'(u)\|$, gives the speed with which we travel at time u , while the direction of $\mathbf{f}'(u)$ gives the direction in which we travel, in other words the tangent to the curve at time u . With these interpretations a regular curve is one where we never stop as we travel along the curve.

The straight line segment

$$\mathbf{f}(u) = (1 - u)\mathbf{p}_0 + u\mathbf{p}_1, \quad \text{for } u \in [0, 1],$$

where $\mathbf{p}_0$ and $\mathbf{p}_1$ are points in the plane, is a simple example of a parametric representation. Since $\mathbf{f}'(u) = \mathbf{p}_1 - \mathbf{p}_0$ for all u , we have in fact that $\mathbf{f}$ is a regular parametric representation, provided that $\mathbf{p}_0 \neq \mathbf{p}_1$. The tangent vector is, as expected, parallel to the curve, and the speed along the curve is constant.

As another example, let us consider the unit circle. It is easy to check that the mapping given by

$$\mathbf{f}(u) = (x(u), y(u)) = (\cos u, \sin u)$$

satisfies the equation $x(u)^2 + y(u)^2 = 1$, so that if u varies from 0 to 2π , the whole unit circle will be traced out. We also have $\|\mathbf{f}'(u)\| = 1$ for all u , so that $\mathbf{f}$ is a regular parametric representation.

One may wonder what the significance of the regularity condition $\mathbf{f}'(u) \neq 0$ is. Let us consider the parametric representation given by

$$\mathbf{f}(u) = \begin{cases} (0, u^2), & \text{for } u < 0; \\ (u^2, 0), & \text{for } u \geq 0; \end{cases}$$

in other words, for $u < 0$ the image of $\mathbf{f}$ is the positive y -axis and for $u > 0$, the image is the positive x -axis. A plot of $\mathbf{f}$ for $u \in [-1, 1]$ is shown in Figure 6.1 (a). The geometric figure traced out by $\mathbf{f}$ clearly has a right angle corner at the origin, but $\mathbf{f}'$ which is given by

$$\mathbf{f}'(u) = \begin{cases} (0, 2u), & \text{for } u < 0; \\ (2u, 0), & \text{for } u > 0; \end{cases}$$

is still continuous for all u . The source of the problem is the fact that $\mathbf{f}'(0) = 0$. For this means that as we travel along the curve, the speed becomes zero at $u = 0$ and cancels out the discontinuity in the tangent direction, so that we can manage to turn the corner. On the other hand, if we consider the unit tangent vector $\boldsymbol{\theta}(u)$ defined by

$$\boldsymbol{\theta}(u) = \mathbf{f}'(u) / \|\mathbf{f}'(u)\|,$$

we see that

$$\boldsymbol{\theta}(u) = \begin{cases} (0, -1), & \text{for } u < 0; \\ (1, 0), & \text{for } u > 0. \end{cases}$$

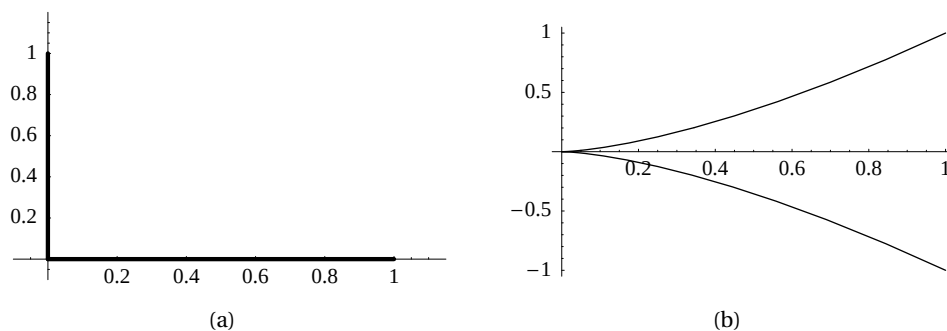


Figure 6.1. A parametric representation with continuous first derivative but discontinuous unit tangent (a), and the parametric representation $\mathbf{f}(u) = (u^2, u^3)$ (b).

As expected, the unit tangent vector is discontinuous at $u = 0$.

A less obvious example where the same problem occurs is shown in Figure 6.1 (b). The parametric representation is $\mathbf{f}(u) = (u^2, u^3)$ which clearly has a continuous tangent, but again we have $\mathbf{f}'(0) = (0, 0)$ which cancels the discontinuity in the unit tangent vector at $u = 0$. To avoid the problems that may occur when the tangent becomes zero, it is common, as in Definition 6.1, to assume that the parametric representation is regular.

6.1.2 Changes of parameter and parametric curves

If we visualise a parametric representation through its graph as we have done here, it is important to know whether the same graph may be obtained from different parametric representations. It is easy to see that the answer to this question is yes. As an example, consider again the unit circle $\mathbf{f}(u) = (\cos u, \sin u)$. If we substitute $u = 2\pi v$, we obtain the parametric representation

$$\hat{\mathbf{r}}(v) = (\cos 2\pi v, \sin 2\pi v).$$

As v varies in the interval $[0, 1]$, the original parameter u will vary in the interval $[0, 2\pi]$ so that $\hat{\mathbf{r}}(v)$ will trace out the same set of points in $\mathbb{R}^2$ and therefore yield the same graph as $\mathbf{f}(u)$. The mapping $u = 2\pi v$ is called a *change of parameter*.

Definition 6.2. A real function $u(v)$ defined on an interval I is called an *allowable change of parameter* of class C^m if it has m continuous derivatives, and the derivative $u'(v)$ is nonzero for all v in I . If $u'(v)$ is positive for all v then it is called an *orientation preserving change of parameter*.

From the chain rule we observe that if $\mathbf{g}(v) = \mathbf{f}(u(v))$ then

$$\mathbf{g}'(v) = u'(v)\mathbf{f}'(u(v)).$$

This means that even if $\mathbf{f}$ is a regular parametric representation, we can still have $\mathbf{g}'(v) = 0$ for some v if $u'(v) = 0$. This is avoided by requiring $u'(v) \neq 0$ as in Definition 6.2.

If $u'(v) > 0$ for all v , the points on the graph of the curve are traced in the same order both by $\mathbf{f}$ and $\mathbf{g}$, the two representations have the same orientation. If $u'(v) < 0$ for all v , then $\mathbf{f}$

and $\mathbf{g}$ have opposite orientation, the points on the graph are traced in opposite orders. The change of parameter $u(v) = 2\pi v$ of the circle above was orientation preserving.

Note that since $u'(v) \neq 0$, the function $u(v)$ is one-to-one so that the inverse $v(u)$ exists and is an allowable change of parameter as well.

The redundancy in the representation of geometric objects can be resolved in a standard way. We simply say that two parametric representations are equivalent if they are related by a change of parameter. If this is the case we will often say that one representation is a reparametrisation of the other.

Definition 6.3. *A regular parametric curve is the equivalence class of parameterisations of a given regular parametric representation. A particular parametric representation of a curve is called a parametrisation of the curve.*

We will use this definition very informally. Most of the time we will just have a parametric representation $\mathbf{f}$ which we will refer to as a parametrisation of a curve or simply a curve.

As an interpretation of the different parameterisations of a curve it is constructive to extend the analogy to travelling along a road. As mentioned above, we can think of the parameter u as measuring the elapsed time as we travel along the curve, and the length of the tangent vector as the speed with which we travel. The road with its hills and bends is fixed, but there are still many ways to travel along it. We can both travel at different velocities and in different directions. This corresponds to different parameterisations.

A natural question is whether there is a preferred way of travelling along the road. A mathematician would probably say that the best way to travel is to maintain a constant speed, and we shall see later that this does indeed simplify the analysis of a curve. On the other hand, a physicist (and a good automobile driver) would probably say that it is best to go slowly around sharp corners and faster along straighter parts of the curve. For the purpose of constructing spline curves it turns out that this latter point of view usually gives the best results.

6.1.3 Arc length parametrisation

Let us end this brief introduction to parametric curves by a discussion of parameterisations with constant speed. Suppose that we have a parametrisation such that the tangent vector has constant unit length along the curve. Then the difference in parameter value at the beginning and end of the curve equals the length of the curve, which is reason enough to study such parameterisations. This justifies the next definition.

Definition 6.4. *A regular parametric curve $\mathbf{g}(\sigma)$ in $\mathbb{R}^s$ is said to be parametrised by arc length if $\|\mathbf{g}'(\sigma)\| = 1$ for all σ .*

Let $\mathbf{f}(u)$ be a given regular curve with $u \in [a, b]$, and let $\mathbf{g}(\sigma) = \mathbf{f}(u(\sigma))$ be a reparametrisation such that $\|\mathbf{g}'(\sigma)\| = 1$ for all σ . Since $\mathbf{g}'(\sigma) = u'(\sigma)\mathbf{f}'(u(\sigma))$, we see that we must have $|u'(\sigma)| = 1/\|\mathbf{f}'(u(\sigma))\|$ or $|\sigma'(u)| = \|\mathbf{f}'(u)\|$ (this follows since $u(\sigma)$ is invertible with inverse $\sigma(u)$ and $u'(\sigma)\sigma'(u) = 1$). The natural way to achieve this is to define $\sigma(u)$ by

$$\sigma(u) = \int_a^u \|\mathbf{f}'(v)\| dv. \quad (6.1)$$

We sum this up in a proposition.

Proposition 6.5. *Let $\mathbf{f}(u)$ be a given regular parametric curve. The change of parameter given by (6.1) reparametrises the curve by arc length, so that if $\mathbf{g}(\sigma) = \mathbf{f}(u(\sigma))$ then $\|\mathbf{g}'(\sigma)\| = 1$.*

Note that $\sigma(u)$ as given by (6.1) gives the length of the curve from the starting point $\mathbf{f}(a)$ to the point $\mathbf{f}(u)$. This can be seen by sampling $\mathbf{f}$ at a set of points, computing the length of the piecewise linear interpolant to these points, and then letting the density of the points go to infinity.

Proposition 6.6. *The length of a curve $\mathbf{f}$ defined on an interval $[a, b]$ is given by*

$$L(\mathbf{f}) = \int_a^b \|\mathbf{f}'(u)\| du$$

It should be noted that parametrisation by arc length is not unique. The orientation can be reversed and the parameterisation may be translated by a constant. Note also that if we have a parametrisation that is constant but not arc length, then arc length parametrisation can be obtained by a simple scaling.

Parametrisation by arc length is not of much practical importance in approximation since the integral in (6.1) very seldom can be expressed in terms of elementary functions, and the computation of the integral is usually too expensive. One important exception is the circle. As we saw at the beginning of the chapter, the parametrisation $\mathbf{r}(u) = (\cos u, \sin u)$ is by arc length.

6.2 Approximation by Parametric Spline Curves

Having defined parametric curves formally, we are now ready to define parametric spline curves. This is very simple, we just let the coefficients that multiply the B-splines be points in $\mathbb{R}^s$ instead of real numbers. We then briefly consider how the spline approximation methods that we introduced for spline functions can be generalised to curves.

6.2.1 Definition of parametric spline curves

A spline curve $\mathbf{f}$ must, as all curves, be defined on an interval I and take its values in $\mathbb{R}^s$. There is a simple and obvious way to achieve this.

Definition 6.7. *A parametric spline curve in $\mathbb{R}^s$ is a spline function where each B-spline coefficient is a point in $\mathbb{R}^s$. More specifically, let $\boldsymbol{\tau} = (\tau_i)_{i=1}^{n+d+1}$ be a knot vector for splines of degree d . Then a parametric spline curve of degree d with knot vector $\boldsymbol{\tau}$ and coefficients $\mathbf{c} = (\mathbf{c}_i)_{i=1}^n$ is given by*

$$\mathbf{g}(u) = \sum_{i=1}^n \mathbf{c}_i B_{i,d,\boldsymbol{\tau}}(u),$$

where each $\mathbf{c}_i = (c_i^1, c_i^2, \dots, c_i^s)$ is a vector in $\mathbb{R}^s$. The set of all spline curves in $\mathbb{R}^s$ of degree d with knot vector $\boldsymbol{\tau}$ is denoted by $\mathbb{S}_{d,\boldsymbol{\tau}}^s$.

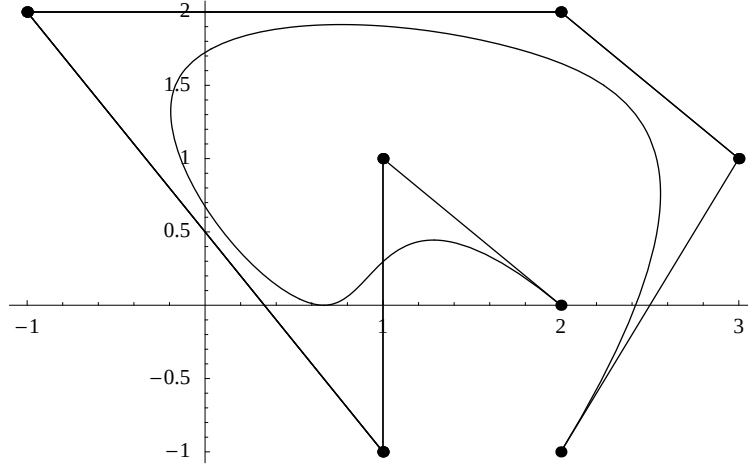


Figure 6.2. A cubic parametric spline curve with its control polygon.

In Definition 6.7, a spline curve is defined as a spline function where the coefficients are points in $\mathbb{R}^s$. From this it follows that

$$\begin{aligned}
 \mathbf{g}(u) &= \sum_i \mathbf{c}_i B_i(u) = \sum_i (c_i^1, \dots, c_i^s) B_i(u) \\
 &= \left(\sum_i c_i^1 B_i(u), \dots, \sum_i c_i^s B_i(u) \right) \\
 &= (g^1(u), \dots, g^s(u)),
 \end{aligned} \tag{6.2}$$

so that $\mathbf{g}$ is a vector of spline functions. This suggests a more general definition of spline curves where the degree and the knot vector in the s components need not be the same, but this is not common and seems to be of little practical interest.

Since a spline curve is nothing but a vector of spline functions as in (6.2), it is simple to compute $\mathbf{f}(u)$: Just apply a routine like Algorithm 2.16 to each of the component spline functions $g^1, \dots, g^s$. If the algorithm has been implemented in a language that supports vector arithmetic, then evaluation is even simpler. Just apply Algorithm 2.16 directly to $\mathbf{g}$, with vector coefficients. The result will be the vector $\mathbf{g}(u) = (g^1(u), \dots, g^s(u))$.

Example 6.8. As an example of a spline curve, suppose that we are given n points $\mathbf{p} = (\mathbf{p}_i)_{i=1}^n$ in the plane with $\mathbf{p}_i = (x_i, y_i)$, and define the knot vector $\boldsymbol{\tau}$ by

$$\boldsymbol{\tau} = (1, 1, 2, 3, 4, \dots, n-2, n-1, n, n).$$

Then the linear spline curve

$$\mathbf{g}(u) = \sum_{i=1}^n \mathbf{p}_i B_{i,1,\boldsymbol{\tau}}(u) = \left(\sum_{i=1}^n x_i B_{i,1,\boldsymbol{\tau}}(u), \sum_{i=1}^n y_i B_{i,1,\boldsymbol{\tau}}(u) \right)$$

is a representation of the piecewise linear interpolant to the points $\mathbf{p}$.

An example of a cubic spline curve with its control polygon is shown in Figure 6.2, and this example gives a good illustration of the fact that a spline curve is contained in the convex

hull of its control points. This, we remember, is clear from the geometric construction of spline curves in Chapter 1.

Proposition 6.9. *A spline curve $\mathbf{g} = \sum_{i=1}^n \mathbf{c}_i B_{i,d,\boldsymbol{\tau}}$ defined on a $d+1$ -extended knot vector $\boldsymbol{\tau}$ is a subset of the convex hull of its coefficients,*

$$\mathbf{g}(u) \in \mathbb{CH}(\mathbf{c}_1, \dots, \mathbf{c}_n), \quad \text{for any } u \in [\tau_{d+1}, \tau_{n+1}].$$

If u is restricted to the interval $[\tau_\mu, \tau_{\mu+1}]$ then

$$\mathbf{g}(u) \in \mathbb{CH}(\mathbf{c}_{\mu-d}, \dots, \mathbf{c}_\mu).$$

To create a spline curve, we only have to be able to create spline functions, since a spline curve is just a vector with spline functions in each component. All the methods described in previous chapters for approximation with spline functions can therefore also be utilised for construction of spline curves. To differentiate between curve approximation and function approximation, we will often refer to the methods of Chapter 5 as *functional approximation methods*.

6.2.2 The parametric variation diminishing spline approximation

In Section 5.4, we introduced the variation diminishing spline approximation to a function. This generalises nicely to curves.

Definition 6.10. *Let $\mathbf{f}$ be a parametric curve defined on the interval $[a, b]$, and let $\boldsymbol{\tau}$ be a $d+1$ -extended knot vector with $\tau_{d+1} = a$ and $\tau_{n+1} = b$. The parametric variation diminishing spline approximation $V\mathbf{f}$ is defined by*

$$(V\mathbf{f})(u) = \sum_{i=1}^n \mathbf{f}(\tau_i^*) B_{i,d,\boldsymbol{\tau}}(u),$$

where $\tau_i^* = (\tau_{i+1} + \dots + \tau_{i+d})/d$.

Note that the definition of $V\mathbf{f}$ means that

$$V\mathbf{f} = (Vf^1, \dots, Vf^s).$$

If we have implemented a routine for determining the variation diminishing approximation to a scalar function ($s = 1$), we can therefore determine $V\mathbf{f}$ by calling the scalar routine s times, just as was the case with evaluation of the curve at a point. Alternatively, if the implementation uses vector arithmetic, we can call the function once but with vector data.

A variation diminishing approximation to a segment of the unit circle is shown in Figure 6.3.

It is much more difficult to employ the variation diminishing spline approximation when only discrete data are given, since somehow we must determine a knot vector. This is true for functional data, and for parametric data we have the same problem. In addition, we must also determine a parametrisation of the points. This is common for all parametric approximation schemes when they are applied to discrete data and is most easily discussed for cubic spline interpolation where it is easy to determine a knot vector.

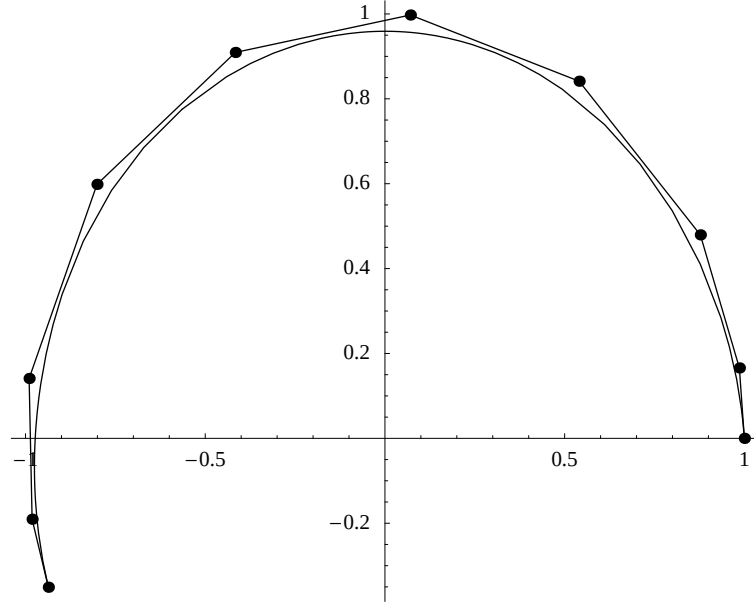


Figure 6.3. A cubic variation diminishing approximation to part of a circle.

6.2.3 Parametric spline interpolation

In Section 5.2, we considered interpolation of a given function or given discrete data by cubic splines, and we found that the cubic C^2 spline interpolant in a sense was the best of all C^2 interpolants. How can this be generalised to curves?

Proposition 6.11. Let $(u_i, \mathbf{f}(u_i))_{i=1}^m$ be given data sampled from the curve $\mathbf{f}$ in $\mathbb{R}^s$, and form the knot vector

$$\boldsymbol{\tau} = (u_1, u_1, u_1, u_1, u_2, \dots, u_{m-1}, u_m, u_m, u_m, u_m).$$

Then there is a unique spline curve $\mathbf{g} = I_N \mathbf{f}$ in $\mathbb{S}_{3,\boldsymbol{\tau}}^s$ that satisfies

$$\mathbf{g}(u_i) = \mathbf{f}(u_i), \quad \text{for } i = 1, \dots, m, \quad (6.3)$$

with the natural end conditions $\mathbf{g}''(u_1) = \mathbf{g}''(u_m) = \mathbf{0}$, and this spline curve $\mathbf{g}$ uniquely minimises

$$\left\| \int_{u_1}^{u_m} \mathbf{h}''(u) du \right\|$$

when $\mathbf{h}$ varies over the class of C^2 parametric representations that satisfy the interpolation conditions (6.3).

Proof. All the statements follow by considering the s functional interpolation problems separately. ■

Note that Proposition 6.11 can also be expressed in the succinct form

$$I_N \mathbf{f} = (I_N f^1, \dots, I_N f^s).$$

This means that the interpolant can be computed by solving s functional interpolation problems. If we go back to Section 5.2.2, we see that the interpolant is determined by solving a system of linear equations. If we consider the s systems necessary to determine $I_N \mathbf{f}$, we see that it is only the right hand side that differs; the coefficient matrix $\mathbf{A}$ remains the same. This can be exploited to speed up the computations since the LU -factorisation of the coefficient matrix can be computed once and for all and the s solutions computed by back substitution; for more information consult a text on numerical linear algebra. As for evaluation and the variation diminishing approximation, this makes it very simple to implement cubic spline interpolation in a language that supports vector arithmetic: Simply call the routine for functional interpolation with vector data.

We have focused here on cubic spline interpolation with natural end conditions, but Hermite and free end conditions can be treated completely analogously.

Let us turn now to cubic parametric spline interpolation in the case where the data are just given as discrete data.

Problem 6.12. Let $(\mathbf{p}_i)_{i=1}^m$ be a set of points in $\mathbb{R}^s$. Find a cubic spline $\mathbf{g}$ in some spline space $\mathbb{S}_{3,\boldsymbol{\tau}}^s$ such that

$$\mathbf{g}(u_i) = \mathbf{p}_i, \quad \text{for } i = 1, \dots, m,$$

for some parameter values $(u_i)_{i=1}^m$ with $u_1 < u_2 < \dots < u_m$.

Problem 6.12 is a realistic problem. A typical situation is that somehow a set of points on a curve has been determined, for instance through measurements; the user then wants the computer to draw a ‘nice’ curve through the points. In such a situation the knot vector is of course not known in advance, but for functional approximation it could easily be determined from the abscissae. In the present parametric setting this is a fundamentally more difficult problem as long as we have no parameter values associated with the data points. An example may be illuminating.

Example 6.13. Suppose that m points in the plane $\mathbf{p} = (\mathbf{p}_i)_{i=1}^m$ with $\mathbf{p}_i = (x_i, y_i)$ are given. We seek a cubic spline curve that interpolates the points $\mathbf{p}$. We can proceed as follows. Associate with each data point $\mathbf{p}_i$ the parameter value i . If we are also given the derivatives (tangents) at the ends as (x'_1, y'_1) and (x'_m, y'_m) , we can apply cubic spline interpolation with Hermite end conditions to the two sets of data $(i, x_i)_{i=1}^m$ and $(i, y_i)_{i=1}^m$. The knot vector will then for both of the two components be

$$\boldsymbol{\tau} = (1, 1, 1, 1, 2, 3, 4, \dots, m-2, m-1, m, m, m, m).$$

We can then perform the two steps

- (i) Find the spline function $p^1 \in \mathbb{S}_{3,\boldsymbol{\tau}}$ with coefficients $\mathbf{c}^1 = (c_i^1)_{i=1}^{m+2}$ that interpolates the points $(i, x_i)_{i=1}^m$ and satisfies $Dp^1(1) = x'_1$ and $Dp^1(m) = x'_m$.
- (ii) Find the spline function $p^2 \in \mathbb{S}_{3,\boldsymbol{\tau}}$ with coefficients $\mathbf{c}^2 = (c_i^2)_{i=1}^{m+2}$ that interpolates the points $(i, y_i)_{i=1}^m$ and satisfies $Dp^2(1) = y'_1$ and $Dp^2(m) = y'_m$.

Together this yields a cubic spline curve

$$\mathbf{g}(u) = \sum_{i=1}^{m+2} \mathbf{c}_i B_{i,3,\boldsymbol{\tau}}(u)$$

that satisfies $\mathbf{g}(i) = \mathbf{p}_i$ for $i = 1, 2, \dots, m$.

The only part of the construction of the cubic interpolant in Example 6.13 that is different from the corresponding construction for spline functions is the assignment of the parameter value i to the point $\mathbf{f}_i = (x_i, y_i)$ for $i = 1, 2, \dots, n$, and therefore also the construction of the knot vector. When working with spline functions, the abscissas of the data points became the knots; for curves we have to choose the knots specifically by giving the parameter values at the data points. Somewhat arbitrarily we gave point number i parameter value i in Example 6.13, this is often termed *uniform parametrisation*.

Going back to Problem 6.12 and the analogy with driving, we have certain places that we want to visit (the points $\mathbf{p}_i$) and the order in which they should be visited, but we do not know when we should visit them (the parameter values u_i). Should one for example try to drive with a constant speed between the points, or should one try to make the time spent between points constant? With the first strategy one might get into problems around a sharp corner where a good driver would usually slow down, and the same can happen with the second strategy if two points are far apart (you must drive fast to keep the time), with a sharp corner just afterwards.

In more mathematical terms, the problem is to guess how the points are meant to be parametrised—which parametric representation are they taken from? This is a difficult problem that so far has not been solved in a satisfactory way. There are methods available though, and in the next section we suggest three of the simplest.

6.2.4 Assigning parameter values to discrete data

Let us recall the setting. We are given m points $(\mathbf{p}_i)_{i=1}^m$ in $\mathbb{R}^s$ and need to associate a parameter value u_i with each point that will later be used to construct a knot vector for spline approximation. Here we give three simple alternatives.

1. **Uniform parametrisation** which amounts to $u_i = i$ for $i = 1, 2, \dots, m$. This has the shortcomings discussed above.
2. **Cord length parametrisation** which is given by

$$u_1 = 0 \quad \text{and} \quad u_i = u_{i-1} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\| \quad \text{for } i = 2, 3, \dots, m.$$

If the final approximation should happen to be the piecewise linear interpolant to the data, this method will correspond to parametrisation by arc length. This often causes problems near sharp corners in the data where it is usually wise to move slowly.

3. **Centripetal parametrisation** is given by

$$u_1 = 0 \quad \text{and} \quad u_i = u_{i-1} + \|\mathbf{p}_i - \mathbf{p}_{i-1}\|^{1/2} \quad \text{for } i = 2, 3, \dots, m.$$

For this method, the difference $u_i - u_{i-1}$ will be smaller than when cord length parametrisation is used. But like the other two methods it does not take into consideration sharp corners in the data, and may therefore fail badly on difficult data.

There are many other methods described in the literature for determining good parameter values at the data points, but there is no known ‘best’ method. In fact, the problem of finding good parameterisations is an active research area.

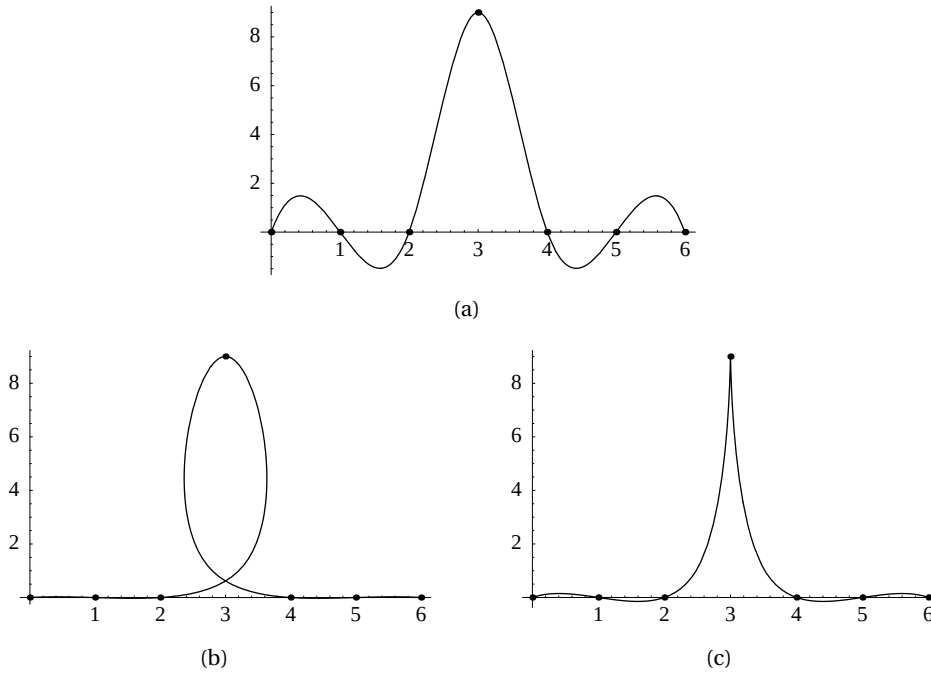


Figure 6.4. Parametric, cubic spline interpolation with uniform parametrisation (a), cord length parametrisation (b), and centripetal parametrisation (c).

Figures 6.4 (a)–(c) show examples of how the three methods of parametrisation described above perform on a difficult data set.

6.2.5 General parametric spline approximation

In Chapter 5, we also defined other methods for spline approximation like cubic Hermite interpolation, general spline interpolation and least squares approximation by splines. All these and many other methods for functional spline approximation can be generalised very simply to parametric curves. If the data is given in the form of a parametric curve, the desired functional method can just be applied to each component function of the given curve. If the data is given as a set of discrete points $(\mathbf{p}_i)_{i=1}^m$, a parametrisation of the points must be determined using for example one of the methods in Section 6.2.4. Once this has been done, a functional method can be applied to each of the s data sets $(u_i, p_i^j)_{i,j=1,1}^{m,d}$. If we denote the functional approximation scheme by A and denote the data by $\mathbf{f}$, so that $\mathbf{f}_i = (u_i, \mathbf{p}_i)$ for $i = 1, \dots, m$, the parametric spline approximation satisfies

$$A\mathbf{f} = (Af^1, \dots, Af^s), \quad (6.4)$$

where f^j denotes the data set $(u_i, p_i^j)_{i=1}^m$ which we think of as $(u_i, f^j(u_i))_{i=1}^m$. As we have seen several times now, the advantage of the relation (6.4) is that the parametric approximation can be determined by applying the corresponding functional approximation scheme to the s components, or, if we use a language that supports vector arithmetic, we simply call the

routine for functional approximation with vector data. In Chapter 7, we shall see that the functional methods can be applied repeatedly in a similar way to compute tensor product spline approximations to surfaces.

Tensor Product Spline Surfaces

Earlier we introduced parametric spline curves by simply using vectors of spline functions, defined over a common knot vector. In this chapter we introduce spline surfaces, but again the construction of *tensor product surfaces* is deeply dependent on univariate spline functions. We first construct spline functions of two variables of the form $z = f(x, y)$, so called *explicit* spline surfaces, whose graph can be visualized as a surface in three dimensional space. We then pass to *parametric* surfaces in the same way that we passed from spline functions to spline curves.

The advantage of introducing tensor product surfaces is that all the approximation methods that we introduced in Chapter 5 generalize very easily as we shall see below. The methods also generalize nicely to parametric tensor product surfaces, but here we get the added complication of determining a suitable parametrisation in the case where we are only given discrete data.

7.1 Explicit tensor product spline surfaces

The reader is undoubtedly familiar with polynomial surfaces of degree one and two. A linear surface

$$z = ax + by + c$$

represents a plane in 3-space. An example of a quadratic surface is the circular paraboloid

$$z = x^2 + y^2$$

shown in Figure 7.1 (a). The spline surfaces we will consider are made by gluing together polynomial “patches” like these.

7.1.1 Definition of the tensor product spline

For $x \in [0, 1]$ the line segment

$$b_0(1 - x) + b_1 x$$

connects the two values b_0 and b_1 . Suppose $b_0(y)$ and $b_1(y)$ are two functions defined for y in some interval $[c, d]$. Then for each $y \in [c, d]$ the function $b_0(y)(1 - x) + b_1(y)x$ is a line segment

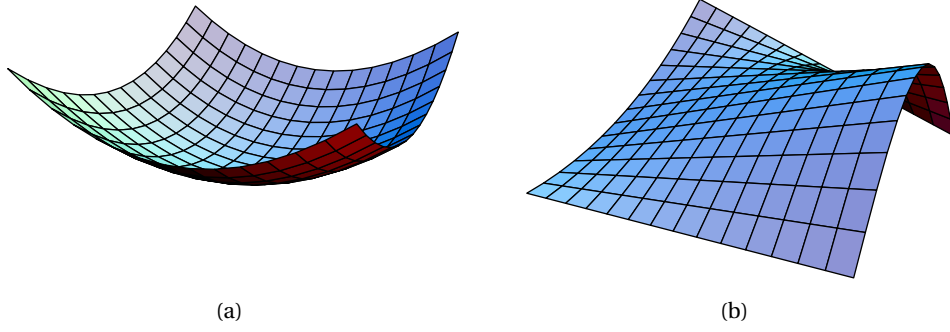


Figure 7.1. A piece of the circular paraboloid $z = x^2 + y^2$ is shown in (a), while the surface $(1-x)y^2 + x \sin(\pi y)$ is shown in (b).

connecting $b_0(y)$ and $b_1(y)$. When y varies we get a family of straight lines representing a surface

$$z = b_0(y)(1-x) + b_1(y)x.$$

Such a “ruled” surface is shown in Figure 7.1 (b). Here we have chosen $b_0(y) = y^2$ and $b_1(y) = \sin(\pi y)$ for $y \in [0, 1]$.

An interesting case is obtained if we take b_0 and b_1 to be linear polynomials. Specifically, if

$$b_0(y) = c_{0,0}(1-y) + c_{0,1}y, \quad \text{and} \quad b_1(y) = c_{1,0}(1-y) + c_{1,1}y,$$

we obtain

$$f(x, y) = c_{0,0}(1-x)(1-y) + c_{0,1}(1-x)y + c_{1,0}x(1-y) + c_{1,1}xy,$$

for suitable coefficients $c_{i,j}$. In fact these coefficients are the values of f at the corners of the unit square. This surface is ruled in both directions. For each fixed value of one variable we have a linear function in the other variable. We call f a *bilinear polynomial*. Note that f reduces to a quadratic polynomial along the diagonal line $x = y$.

We can use similar ideas to construct spline surfaces from families of spline functions. Suppose that for some integer d and knot vector σ we have the spline space

$$\mathbb{S}_1 = \mathbb{S}_{d,\sigma} = \text{span}\{\phi_1, \dots, \phi_{n_1}\}.$$

To simplify the notation we have denoted the B-splines by $\{\phi_i\}_{i=1}^{n_1}$. Consider a spline in $\mathbb{S}_1$ with coefficients that are functions of y ,

$$f(x, y) = \sum_{i=1}^{n_1} c_i(y) \phi_i(x). \quad (7.1)$$

For each value of y we now have a spline in $\mathbb{S}_1$, and when y varies we get a family of spline functions that each depends on x . Any choice of functions c_i results in a surface, but a particularly useful construction is obtained if we choose the c_i to be splines as well. Suppose we have another spline space of degree ℓ and with knots τ ,

$$\mathbb{S}_2 = \mathbb{S}_{\ell,\tau} = \text{span}\{\psi_1, \dots, \psi_{n_2}\}$$

where $\{\psi_j\}_{j=1}^{n_2}$ denotes the B-spline basis in $\mathbb{S}_2$. If each coefficient function $c_i(y)$ is a spline in $\mathbb{S}_2$, then

$$c_i(y) = \sum_{j=1}^{n_2} c_{i,j} \psi_j(y) \quad (7.2)$$

for suitable numbers $(c_{i,j})_{i,j=1}^{n_1, n_2}$. Combining (7.1) and (7.2) we obtain

$$f(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{i,j} \phi_i(x) \psi_j(y). \quad (7.3)$$

Definition 7.1. The tensor product of the two spaces $\mathbb{S}_1$ and $\mathbb{S}_2$ is defined to be the family of all functions of the form

$$f(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} c_{i,j} \phi_i(x) \psi_j(y),$$

where the coefficients $(c_{i,j})_{i,j=1}^{n_1, n_2}$ can be any real numbers. This linear space of functions is denoted $\mathbb{S}_1 \otimes \mathbb{S}_2$.

The space $\mathbb{S}_1 \otimes \mathbb{S}_2$ is spanned by the functions $\{\phi_i(x) \psi_j(y)\}_{i,j=1}^{n_1, n_2}$ and therefore has dimension $n_1 n_2$. Some examples of these basis functions are shown in Figure 7.2. In Figure 7.2 (a) we have $\phi = \psi = B(\cdot | 0, 1, 2)$. The resulting function is a bilinear polynomial in each of the four squares $[i, i+1) \times [j, j+1)$ for $i, j = 0, 1$. It has the shape of a curved pyramid with value one at the top. In Figure 7.2 (b) we show the result of taking $\phi = \psi = B(\cdot | 0, 1, 2, 3)$. This function is a biquadratic polynomial in each of the 9 squares $[i, i+1) \times [j, j+1)$ for $i, j = 0, 1, 2$. In Figure 7.2 (c) we have changed ϕ to $B(\cdot | 0, 0, 0, 1)$.

Tensor product surfaces are piecewise polynomials on rectangular domains. A typical example is shown in Figure 7.3. Each vertical line corresponds to a knot for the $\mathbb{S}_1$ space, and similarly, each horizontal line stems from a knot in the $\mathbb{S}_2$ space. The surface will usually have a discontinuity across the knot lines, and the magnitude of the discontinuity is inherited directly from the univariate spline spaces. For example, across a vertical knot line, partial derivatives with respect to x have the continuity properties of the univariate spline functions in $\mathbb{S}_1$. This follows since the derivatives, say the first derivative, will involve sums of terms of the form

$$\frac{\partial}{\partial x} (c_{i,j} \phi_i(x) \psi_j(y)) = c_{i,j} \phi'_i(x) \psi_j(y).$$

A tensor product surface can be written conveniently in matrix-vector form. If $f(x, y)$ is given by (7.3) then

$$f(x, y) = \boldsymbol{\phi}(x)^T \mathbf{C} \boldsymbol{\psi}(y), \quad (7.4)$$

where

$$\boldsymbol{\phi} = (\phi_1, \dots, \phi_{n_1})^T, \quad \boldsymbol{\psi} = (\psi_1, \dots, \psi_{n_2})^T,$$

and $\mathbf{C} = (c_{i,j})_{i,j=1}^{n_1, n_2}$ is the matrix of coefficients. This can be verified quite easily by expanding the matrix products in (7.4).

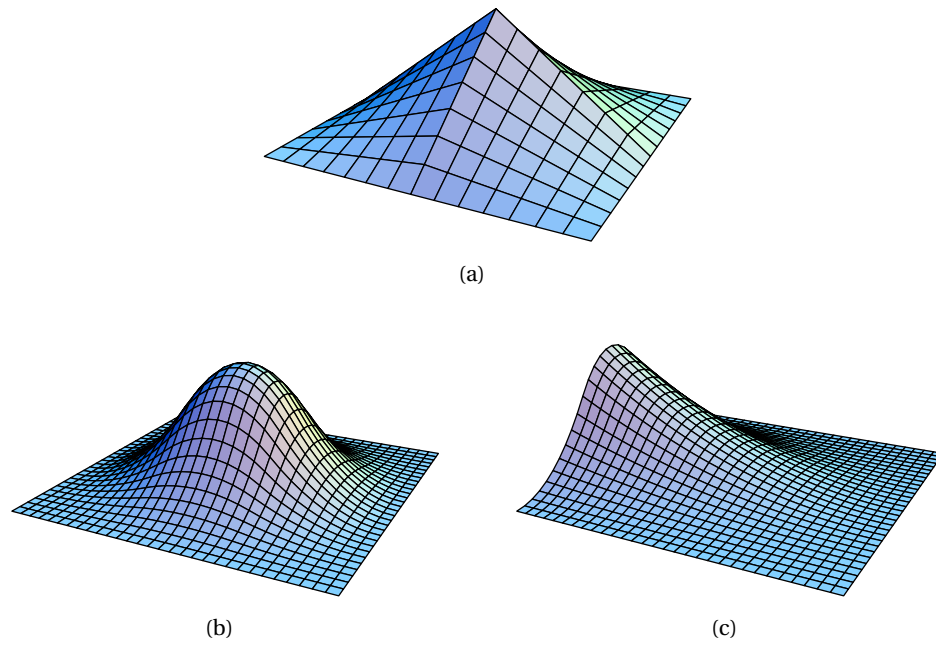


Figure 7.2. A bilinear B-spline (a), a biquadratic B-spline (b) and biquadratic B-spline with a triple knot in one direction (c).

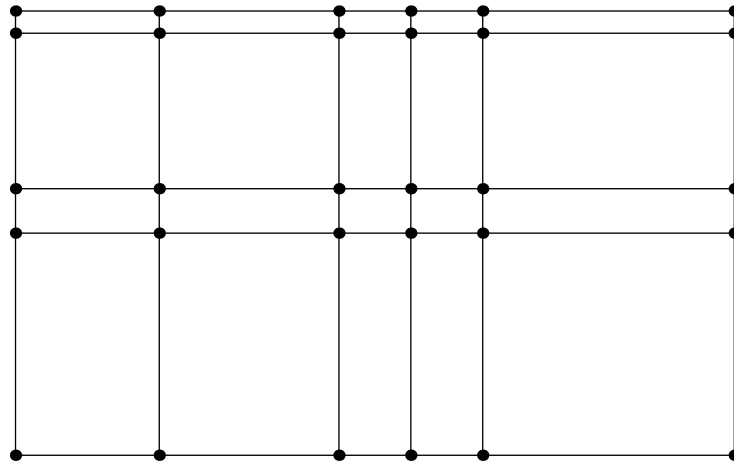


Figure 7.3. The knot lines for a tensor product spline surface.

7.1.2 Evaluation of tensor product spline surfaces

There are many ways to construct surfaces from two spaces of univariate functions, but the tensor product has one important advantage: many standard operations that we wish to perform with the surfaces are very simple generalizations of corresponding univariate operations. We will see several examples of this, but start by showing how to compute a point on a tensor product spline surface.

To compute a point on a tensor product spline surface, we can make use of the algorithms we have for computing points on spline functions. Suppose we want to compute $f(x, y) = \boldsymbol{\phi}(x)^T \mathbf{C} \boldsymbol{\psi}(y)^T$, and suppose for simplicity that the polynomial degree in the two directions are equal, so that $d = \ell$. If the integers μ and ν are such that $\sigma_\nu \leq x < \sigma_{\nu+1}$ and $\tau_\mu \leq y < \tau_{\mu+1}$, then we know that only $(\phi_i(x))_{i=\nu-d}^\nu$ and $(\psi_j(y))_{j=\mu-d}^\mu$ can be nonzero at (x, y) . To compute

$$f(x, y) = \boldsymbol{\phi}(x)^T \mathbf{C} \boldsymbol{\psi}(y) \quad (7.5)$$

we therefore first make use of Algorithm 2.17 to compute the $d + 1$ nonzero B-splines at x and the $d + 1$ nonzero B-splines at y . We can then pick out that part of the coefficient matrix $\mathbf{C}$ which corresponds to these B-splines and multiply together the right-hand side of (7.5).

A pleasant feature of this algorithm is that its operation count is of the same order of magnitude as evaluation of univariate spline functions. If we assume, for simplicity, that $\ell = d$, we know that roughly $3(d + 1)^2/2$ multiplications are required to compute the nonzero B-splines at x , and the same number of multiplications to compute the nonzero B-splines at y . To finish the computation of $f(x, y)$, we have to evaluate a product like that in (7.5), with $\mathbf{C}$ a $(d + 1) \times (d + 1)$ -matrix and the two vectors of dimension $d + 1$. This requires roughly $(d + 1)^2$ multiplications, giving a total of $4(d + 1)^2$ multiplications. The number of multiplications required to compute a point on a spline surface is therefore of the same order as the number of multiplications required to compute a point on a univariate spline function. The reason we can compute a point on a surface this quickly is the rather special structure of tensor products.

7.2 Approximation methods for tensor product splines

One of the main advantages of the tensor product definition of surfaces is that the approximation methods that we developed for functions and curves can be utilised directly for approximation of surfaces. In this section we consider some of the approximation methods in Chapter 5 and show how they can be generalized to surfaces.

7.2.1 The variation diminishing spline approximation

Consider first the variation diminishing approximation. Suppose f is a function defined on a rectangle

$$\Omega = \{(x, y) \mid a_1 \leq x \leq b_1 \text{ \& } a_2 \leq y \leq b_2\} = [a_1, b_1] \times [a_2, b_2].$$

Let $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^{n_1+d+1}$ be a $d + 1$ -regular knot vector with boundary knots $\sigma_d = a_1$ and $\sigma_{n_1} = b_1$, and let $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n_2+\ell+1}$ be an $\ell + 1$ -regular knot vector with boundary knots $\tau_\ell = a_2$ and $\tau_{n_2} =$

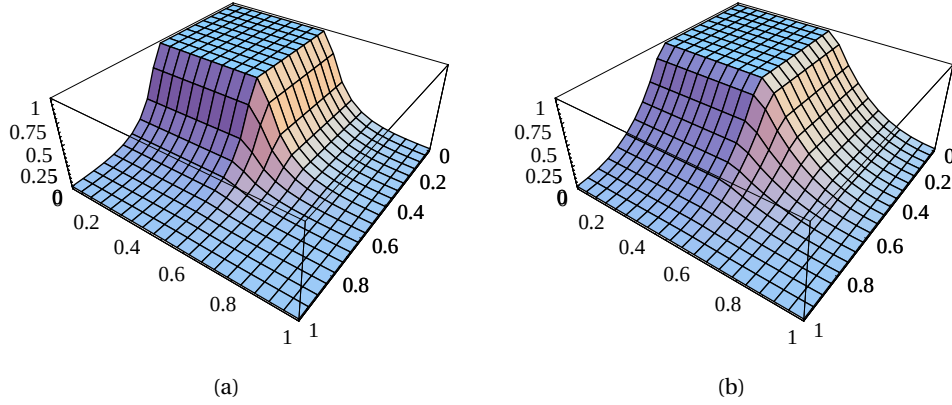


Figure 7.4. The function $f(x, y)$ given in Example 7.2 is shown in (a) and its variation diminishing spline approximation is shown in (b).

b_2 . As above we let $\phi_i = B_{i,d,\sigma}$ and $\psi_j = B_{j,\ell,\tau}$ be the B-splines on σ and τ respectively. The spline

$$Vf(x, y) = \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f(\sigma_i^*, \tau_j^*) \phi_i(x) \psi_j(y) \quad (7.6)$$

where

$$\begin{aligned} \sigma_i^* &= \sigma_{i,d}^* = (\sigma_{i+1} + \dots + \sigma_{i+d})/d \\ \tau_j^* &= \tau_{j,\ell}^* = (\tau_{j+1} + \dots + \tau_{j+\ell})/\ell, \end{aligned} \quad (7.7)$$

is called the *variation diminishing spline approximation on (σ, τ) of degree (d, ℓ)* . If no interior knots in σ has multiplicity $d+1$ then

$$a_1 = \sigma_1^* < \sigma_2^* < \dots < \sigma_{n_1}^* = b_1,$$

and similarly, if no interior knots in τ has multiplicity $\ell+1$ then

$$a_2 = \tau_1^* < \tau_2^* < \dots < \tau_{n_2}^* = b_2.$$

This means that the nodes $(\sigma_i^*, \tau_j^*)_{i,j=1}^{n_1, n_2}$ divides the domain Ω into a rectangular grid.

Example 7.2. Suppose we want to approximate the function

$$f(x, y) = g(x)g(y), \quad (7.8)$$

where

$$g(x) = \begin{cases} 1, & 0 \leq x \leq 1/2, \\ e^{-10(x-1/2)}, & 1/2 < x \leq 1, \end{cases}$$

on the unit square

$$\Omega = \{(x, y) \mid 0 \leq x \leq 1 \text{ \& } 0 \leq y \leq 1\} = [0, 1]^2.$$

A graph of this function is shown in Figure 7.4 (a), and we observe that f has a flat spot on the square $[0, 1/2]^2$ and falls off exponentially on all sides. In order to approximate this function by a bicubic variation diminishing

spline we observe that the surface is continuous, but that it has discontinuities partial derivatives across the lines $x = 1/2$ and $y = 1/2$. We obtain a tensor product spline space with similar continuity properties across these lines by making the value $1/2$ a knot of multiplicity 3 in σ and τ . For an integer q with $q \geq 2$ we define the knot vectors by

$$\begin{aligned}\sigma = \tau = & (0, 0, 0, 0, 1/(2q), \dots, 1/2 - 1/(2q), 1/2, 1/2, 1/2, \\ & 1/2 + 1/(2q), \dots, 1 - 1/(2q), 1, 1, 1, 1).\end{aligned}$$

The corresponding variation diminishing spline approximation is shown in Figure 7.4 (b) for $q = 2$.

The tensor product variation diminishing approximation Vf has shape preserving properties analogous to those discussed in Section 5.4 for curves. In Figures 7.4 (a) and (b) we observe that the constant part of f in the region $[0, 1/2] \times [0, 1/2]$ is reproduced by Vf , and Vf appears to have the same shape as f . These and similar properties can be verified formally, just like for functions.

7.2.2 Tensor Product Spline Interpolation

We consider interpolation at a set of gridded data

$$(x_i, y_j, f_{i,j})_{i=1, j=1}^{m_1, m_2}, \quad (7.9)$$

where

$$a_1 = x_1 < x_2 < \dots < x_{m_1} = b_1, \quad a_2 = y_1 < y_2 < \dots < y_{m_2} = b_2.$$

For each i, j we can think of $f_{i,j}$ as the value of an unknown function $f = f(x, y)$ at the point (x_i, y_j) . Note that these data are given on a grid of the same type as that of the knot lines in Figure 7.3.

We will describe a method to find a function $g = g(x, y)$ in a tensor product space $\mathbb{S}_1 \otimes \mathbb{S}_2$ such that

$$g(x_i, y_j) = f_{i,j}, \quad i = 1, \dots, m_1, \quad j = 1, \dots, m_2. \quad (7.10)$$

We think of $\mathbb{S}_1$ and $\mathbb{S}_2$ as two univariate spline spaces

$$\mathbb{S}_1 = \text{span}\{\phi_1, \dots, \phi_{m_1}\}, \quad \mathbb{S}_2 = \text{span}\{\psi_1, \dots, \psi_{m_2}\}, \quad (7.11)$$

where the ϕ 's and ψ 's are bases of B-splines for the two spaces. Here we have assumed that the dimension of $\mathbb{S}_1 \otimes \mathbb{S}_2$ agrees with the number of given data points since we want to approximate using interpolation. With g on the form

$$g(x, y) = \sum_{p=1}^{m_1} \sum_{q=1}^{m_2} c_{p,q} \psi_q(y) \phi_p(x) \quad (7.12)$$

the interpolation conditions (7.10) lead to a set of equations

$$\sum_{p=1}^{m_1} \sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) \phi_p(x_i) = f_{i,j}, \quad \text{for all } i \text{ and } j.$$

This double sum can be split into two sets of simple sums

$$\sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) = f_{i,j}, \quad (7.13)$$

$$\sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) = d_{p,j}. \quad (7.14)$$

In order to study existence and uniqueness of solutions, it is convenient to have a matrix formulation of the equations for the $c_{p,q}$. We define the matrices

$$\begin{aligned} \Phi &= (\phi_{i,p}) \in \mathbb{R}^{m_1, m_1}, \quad \phi_{i,p} = \phi_p(x_i), \\ \Psi &= (\psi_{j,q}) \in \mathbb{R}^{m_2, m_2}, \quad \psi_{j,q} = \psi_q(y_j), \\ \mathbf{D} &= (d_{p,j}) \in \mathbb{R}^{m_1, m_2}, \quad \mathbf{F} = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}, \quad \mathbf{C} = (c_{p,q}) \in \mathbb{R}^{m_1, m_2}. \end{aligned} \quad (7.15)$$

We then see that (7.13) and (7.14) may be written as

$$\begin{aligned} \sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) &= \sum_{p=1}^{m_1} \phi_{i,p} d_{p,j} = (\Phi \mathbf{D})_{i,j} = (\mathbf{F})_{i,j}, \\ \sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) &= \sum_{q=1}^{m_2} \psi_{j,q} c_{p,q} = (\Psi \mathbf{C}^T)_{j,p} = (\mathbf{D}^T)_{j,p}. \end{aligned}$$

It follows that (7.13) and (7.14) can be expressed on matrix form as

$$\Phi \mathbf{D} = \mathbf{F} \quad \text{and} \quad \mathbf{C} \Psi^T = \mathbf{D}. \quad (7.16)$$

From these equations we obtain the following proposition.

Proposition 7.3. *Suppose the matrices Φ and Ψ are nonsingular. Then there is a unique $g \in S_1 \otimes S_2$ such that (7.10) holds. This g is given by (7.12) where the coefficient matrix $\mathbf{C} = (c_{p,q})$ satisfies the matrix equation*

$$\Phi \mathbf{C} \Psi^T = \mathbf{F}.$$

Proof. The above derivation shows that there is a unique $g \in S_1 \otimes S_2$ such that (7.10) holds if and only if the matrix equations in (7.16) have unique solutions $\mathbf{D}$ and $\mathbf{C}$. But this is the case if and only if the matrices Φ and Ψ are nonsingular. The final matrix equation is just the two equations in (7.16) combined. ■

There is a geometric interpretation of the interpolation process. Let us define a family of x -curves by

$$X_j(x) = \sum_{p=1}^{m_1} d_{p,j} \phi_p(x), \quad j = 1, 2, \dots, m_2.$$

Here the $d_{p,j}$ are taken from (7.13). Then for each j we have

$$X_j(x_i) = f_{i,j}, \quad i = 1, 2, \dots, m_1.$$

We see that X_j is a curve which interpolates the data $\mathbf{f}_j = (f_{1,j}, \dots, f_{m_1,j})$ at the y -level y_j . Moreover, by using (7.10) we see that for all x

$$X_j(x) = g(x, y_j), \quad j = 1, 2, \dots, m_2.$$

This means that we can interpret (7.13) and (7.14) as follows:

- (i) Interpolate in the x -direction by determining the curves X_j interpolating the data $\mathbf{f}_j$.
- (ii) Make a surface by filling in the space between these curves.

This process is obviously symmetric in x and y . Instead of (7.13) and (7.14) we can use the systems

$$\sum_{q=1}^{m_2} e_{i,q} \psi_q(y_j) = f_{i,j}, \quad (7.17)$$

$$\sum_{p=1}^{m_1} c_{p,q} \phi_p(x_i) = e_{i,q}. \quad (7.18)$$

In other words we first make a family of y -curves $Y_i(y) = \sum_{q=1}^{m_2} e_{i,q} \psi_q(y)$ interpolating the row data vectors $F_i = (f_{i,1}, \dots, f_{i,m_2})$. We then blend these curves to obtain the same surface $g(x, y)$.

The process we have just described is a special instance of a more general process which we is called *lofting*. By lofting we mean any process to construct a surface from a family of parallel curves. The word lofting originated in ship design. To draw a ship hull, the designer would first make parallel cross-sections of the hull. These curves were drawn in full size using mechanical splines. Then the cross-sections were combined into a surface by using longitudinal curves. Convenient space for this activity was available at the loft of the shipyard.

We have seen that tensor product interpolation is a combination of univariate interpolation processes. We want to take a second look at this scheme. The underlying univariate interpolation process can be considered as a map converting the data $\mathbf{x}, \mathbf{f}$ into a spline interpolating this data. We can write such a map as

$$g = I[\mathbf{x}, \mathbf{f}] = \sum_{p=1}^{m_1} c_p \phi_p.$$

The coefficients $\mathbf{c} = (c_p)$ are determined from the interpolation requirements $g(x_i) = f_i$ for $i = 1, 2, \dots, m_1$. We also have a related map $\tilde{I}$ which maps the data into the coefficients

$$\mathbf{c} = \tilde{I}[\mathbf{x}, \mathbf{f}].$$

Given m_2 data sets $(x_i, f_{i,j})_{i=1}^{m_1}$ for $j = 1, 2, \dots, m_2$, we combine the function values into a matrix

$$\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_n) = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}$$

and define

$$\mathbf{C} = \tilde{I}[\mathbf{x}, \mathbf{F}] = (\tilde{I}[\mathbf{x}, \mathbf{f}_1], \dots, \tilde{I}[\mathbf{x}, \mathbf{f}_n]). \quad (7.19)$$

With this notation the equations in (7.16) correspond to

$$\mathbf{D} = \tilde{I}_1[\mathbf{x}, \mathbf{F}], \quad \mathbf{C}^T = \tilde{I}_2[\mathbf{y}, \mathbf{D}^T],$$

where $\tilde{I}_1$ and $\tilde{I}_2$ are the univariate interpolation operators in the x and y directions, respectively. Combining these two equations we have

$$\mathbf{C} = (\tilde{I}_1 \otimes \tilde{I}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}] = \tilde{I}_2[\mathbf{y}, \tilde{I}_1[\mathbf{x}, \mathbf{F}]^T]^T. \quad (7.20)$$

We call $\tilde{I}_1 \otimes \tilde{I}_2$, defined in this way, for the tensor product of $\tilde{I}_1$ and $\tilde{I}_2$. We also define $(I_1 \otimes I_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}]$ as the spline in $\mathbb{S}_1 \otimes \mathbb{S}_2$ with coefficients $(\tilde{I}_1 \otimes \tilde{I}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}]$.

These operators can be applied in any order. We can apply I_1 on each of the data vectors $\mathbf{f}_j$ to create the X_j curves, and then use I_2 for the lofting. Or we could start by using I_2 to create y -curves $Y_i(y)$ and then loft in the x -direction using I_1 . From this it is clear that

$$(\tilde{I}_1 \otimes \tilde{I}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}] = (\tilde{I}_2 \otimes \tilde{I}_1)[\mathbf{y}, \mathbf{x}, \mathbf{F}^T].$$

Tensor product interpolation is quite easy to program on a computer. In order to implement the $\tilde{I}[\mathbf{x}, \mathbf{F}]$ operation we need to solve linear systems of the form given in (7.16). These systems have one coefficient matrix, but several right hand sides.

Two univariate programs can be combined easily and efficiently as in (7.20) provided we have a linear equation solver that can handle several right-hand sides simultaneously. Corresponding to the operator $I[\mathbf{x}, \mathbf{f}]$ we would have a program

$$I^P[\mathbf{x}, \mathbf{f}, d, \boldsymbol{\tau}, \mathbf{c}],$$

which to given data $\mathbf{x}$ and $\mathbf{f}$ will return a spline space represented by the degree d and the knot vector $\boldsymbol{\tau}$, and the coefficients $\mathbf{c}$ of an interpolating spline curve in the spline space. Suppose we have two such programs I_1^P and I_2^P corresponding to interpolation in spline spaces $\mathbb{S}_1 = \mathbb{S}_{q,\sigma}$ and $\mathbb{S}_2 = \mathbb{S}_{\ell,\tau}$. Assuming that these programs can handle data of the form $\mathbf{x}, \mathbf{F}$, a program to carry out the process in (7.20) would be

1. $I_1^P[\mathbf{x}, \mathbf{F}, d, \sigma, \mathbf{D}];$
2. $I_2^P[\mathbf{y}, \mathbf{D}^T, \ell, \tau, \mathbf{G}];$
3. $\mathbf{C} = \mathbf{G}^T;$

7.2.3 Least Squares for Gridded Data

The least squares technique is a useful and important technique for fitting of curves and surfaces to data. In principle, it can be used for approximation of functions of any number of variables. Computationally there are several problems however, the main one being that usually a large linear system has to be solved. The situation is better when the data is gridded, say of the form (7.9). We study this important special case in this section and consider the following problem:

Problem 7.4. *Given data*

$$(x_i, y_j, f_{i,j})_{i=1, j=1}^{m_1, m_2},$$

positive weights $(w_i)_{i=1}^{m_1}$ and $(v_j)_{j=1}^{m_2}$, and univariate spline spaces $\mathbb{S}_1$ and $\mathbb{S}_2$, find a spline surface g in $\mathbb{S}_1 \otimes \mathbb{S}_2$ which solves the minimisation problem

$$\min_{g \in \mathbb{S}_1 \otimes \mathbb{S}_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} w_i v_j [g(x_i, y_j) - f_{i,j}]^2.$$

We assume that the vectors of data abscissas $\mathbf{x} = (x_i)_{i=1}^{m_1}$ and $\mathbf{y} = (y_j)_{j=1}^{m_2}$ have distinct components, but that they do not need to be ordered. Note that we only have $m_1 + m_2$ independent weights. Since we have $m_1 \times m_2$ data points it would have been more natural to have $m_1 \times m_2$ weights, one for each data point. The reason for associating weights with grid lines instead of points is computational. As we will see, this assures that the problem splits into a sequence of univariate problems.

We assume that the spline spaces $\mathbb{S}_1$ and $\mathbb{S}_2$ are given in terms of B-splines

$$\mathbb{S}_1 = \text{span}\{\phi_1, \dots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \text{span}\{\psi_1, \dots, \psi_{n_2}\},$$

and seek the function g in the form

$$g(x, y) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \psi_q(y) \phi_p(x).$$

Our goal in this section is to show that Problem 7.4 is related to the univariate least squares problem just as the interpolation problem in the last section was related to univariate interpolation. We start by giving a matrix formulation analogous to Lemma 5.21 for the univariate case.

Lemma 7.5. *Problem 7.4 is equivalent to the following matrix problem*

$$\min_{\mathbf{C} \in \mathbb{R}^{n_1, n_2}} \|\mathbf{ACB}^T - \mathbf{G}\|^2, \quad (7.21)$$

where

$$\begin{aligned} \mathbf{A} &= (a_{i,p}) \in \mathbb{R}^{m_1, n_1}, \quad a_{i,p} = \sqrt{w_i} \phi_p(x_i), \\ \mathbf{B} &= (b_{j,q}) \in \mathbb{R}^{m_2, n_2}, \quad b_{j,q} = \sqrt{v_j} \psi_q(y_j), \\ \mathbf{G} &= (\sqrt{w_i} \sqrt{v_j} f_{i,j}) \in \mathbb{R}^{m_1, m_2}, \quad \mathbf{C} = (c_{p,q}) \in \mathbb{R}^{n_1, n_2}. \end{aligned} \quad (7.22)$$

Here, the norm $\|\cdot\|$ is the Frobenius norm,

$$\|\mathbf{E}\| = \left(\sum_{i=1}^m \sum_{j=1}^n |e_{i,j}|^2 \right)^{1/2} \quad (7.23)$$

for any rectangular $m \times n$ matrix $\mathbf{E} = (e_{i,j})$.

Proof. Suppose $\mathbf{C} = (c_{p,q})$ are the B-spline coefficients of some $g \in \mathbb{S}_1 \otimes \mathbb{S}_2$. Then

$$\begin{aligned} \|\mathbf{ACB}^T - \mathbf{G}\|^2 &= \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \left(\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} a_{i,p} c_{p,q} b_{j,q} - g_{i,j} \right)^2 \\ &= \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} \left(\sum_{p=1}^{n_1} \sum_{q=1}^{n_2} \sqrt{w_i} \phi_p(x_i) c_{p,q} \sqrt{v_j} \psi_q(y_j) - \sqrt{w_i} \sqrt{v_j} f_{i,j} \right)^2 \\ &= \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} w_i v_j [g(x_i, y_j) - f_{i,j}]^2. \end{aligned}$$

This shows that the two minimisation problems are equivalent. ■

We next state some basic facts about the matrix problem (7.21).

Proposition 7.6. *The problem (7.21) always has a solution $\mathbf{C} = \mathbf{C}^*$, and the solution is unique if and only if both matrices $\mathbf{A}$ and $\mathbf{B}$ have linearly independent columns. The solution $\mathbf{C}^*$ can be found by solving the matrix equation*

$$\mathbf{A}^T \mathbf{AC}^* \mathbf{B}^T \mathbf{B} = \mathbf{A}^T \mathbf{GB}. \quad (7.24)$$

Proof. By arranging the entries of $\mathbf{C}$ in a one dimensional vector it can be seen that the minimisation problem (7.21) is a linear least squares problem. The existence of a solution then follows from Lemma 5.22. For the rest of the proof we introduce some additional notation. For matrices $\mathbf{H} = (h_{i,j})$ and $\mathbf{K} = (k_{i,j})$ in $\mathbb{R}^{m,n}$ we define the scalar product

$$(\mathbf{H}, \mathbf{K}) = \sum_{i=1}^m \sum_{j=1}^n h_{i,j} k_{i,j}.$$

This is a scalar product of the matrices $\mathbf{H}$ and $\mathbf{K}$ regarded as vectors. We have $(\mathbf{H}, \mathbf{H}) = \|\mathbf{H}\|^2$, the Frobenius norm of $\mathbf{H}$, squared. We also observe that for any $m \times n$ matrices $\mathbf{H}$ and $\mathbf{K}$, we have

$$\|\mathbf{H} + \mathbf{K}\|^2 = \|\mathbf{H}\|^2 + 2(\mathbf{H}, \mathbf{K}) + \|\mathbf{K}\|^2.$$

Moreover,

$$(\mathbf{E}, \mathbf{HK}) = (\mathbf{H}^T \mathbf{E}, \mathbf{K}) = (\mathbf{EK}^T, \mathbf{H}), \quad (7.25)$$

for any matrices $\mathbf{E}, \mathbf{H}, \mathbf{K}$ such that the matrix operations make sense. For any $\mathbf{C} \in \mathbb{R}^{n_1, n_2}$ we let

$$q(\mathbf{C}) = \|\mathbf{ACB}^T - \mathbf{G}\|^2.$$

This is the function we want to minimize. Suppose $\mathbf{C}^*$ is the solution of (7.24). We want to show that $q(\mathbf{C}^* + \epsilon \mathbf{D}) \geq q(\mathbf{C}^*)$ for any real ϵ and any $\mathbf{D} \in \mathbb{R}^{n_1 \times n_2}$. This will follow from the relation

$$q(\mathbf{C}^* + \epsilon \mathbf{D}) = q(\mathbf{C}^*) + 2\epsilon(\mathbf{A}^T \mathbf{AC}^* \mathbf{B}^T \mathbf{B} - \mathbf{A}^T \mathbf{GB}, \mathbf{D}) + \epsilon^2 \|\mathbf{ADB}^T\|^2. \quad (7.26)$$

For if $\mathbf{C}^*$ satisfies (7.24) then the complicated middle term vanishes and

$$q(\mathbf{C}^* + \epsilon \mathbf{D}) = q(\mathbf{C}^*) + \epsilon^2 \|\mathbf{ADB}^T\|^2 \geq q(\mathbf{C}^*).$$

To establish (7.26) we have to expand $q(\mathbf{C}^* + \epsilon \mathbf{D})$,

$$\begin{aligned} q(\mathbf{C}^* + \epsilon \mathbf{D}) &= \|(\mathbf{A}\mathbf{C}^*\mathbf{B}^T - \mathbf{G}) + \epsilon \mathbf{A}\mathbf{D}\mathbf{B}^T\|^2 \\ &= q(\mathbf{C}^*) + 2\epsilon(\mathbf{A}\mathbf{C}^*\mathbf{B}^T - \mathbf{G}, \mathbf{A}\mathbf{D}\mathbf{B}^T) + \epsilon^2 \|\mathbf{A}\mathbf{D}\mathbf{B}^T\|^2. \end{aligned}$$

Using (7.25) on the middle term, we can move $\mathbf{A}$ and $\mathbf{B}^T$ to the left-hand side of the inner product form, and we obtain (7.26). The uniqueness is left as a problem.

Conversely, suppose that $\mathbf{C}$ does not satisfy (7.24). We need to show that $\mathbf{C}$ does not minimize q . Now, for at least one matrix component i, j we have

$$z = (\mathbf{A}^T \mathbf{A} \mathbf{C} \mathbf{B}^T \mathbf{B} - \mathbf{A}^T \mathbf{G} \mathbf{B})_{i,j} \neq 0.$$

We choose $\mathbf{D}$ as the matrix where the i, j element is equal to 1 and all other entries are 0. Then (7.26) takes the form

$$q(\mathbf{C} + \epsilon \mathbf{D}) = q(\mathbf{C}) + 2\epsilon z + \epsilon^2 \|\mathbf{A}\mathbf{D}\mathbf{B}^T\|^2,$$

and this implies that $q(\mathbf{C} + \epsilon \mathbf{D}) < q(\mathbf{C})$ for $\epsilon z < 0$ and $|\epsilon|$ sufficiently small. But then $\mathbf{C}$ cannot minimize q . ■

In order to find the solution of Problem 7.4, we have to solve the matrix equation (7.24). We can do this in two steps:

1. Find $\mathbf{D}$ from the system $\mathbf{A}^T \mathbf{A} \mathbf{D} = \mathbf{A}^T \mathbf{G}$.
2. Find $\mathbf{C}$ from the system $\mathbf{B}^T \mathbf{B} \mathbf{C}^T = \mathbf{B}^T \mathbf{D}^T$.

The matrix $\mathbf{C}$ is then the solution of (7.24). The first step is equivalent to

$$\mathbf{A}^T \mathbf{A} \mathbf{d}_j = \mathbf{A}^T \mathbf{g}_j, \quad j = 1, 2, \dots, m_2,$$

where $\mathbf{D} = (\mathbf{d}_1, \dots, \mathbf{d}_{m_2})$ and $\mathbf{G} = (\mathbf{g}_1, \dots, \mathbf{g}_{m_2})$. This means that we need to solve m_2 linear least squares problems

$$\min \|\mathbf{A} \mathbf{d}_j - \mathbf{g}_j\|_2^2, \quad j = 1, 2, \dots, m_2.$$

We then obtain a family of x -curves

$$X_j(x) = \sum_{p=1}^{n_1} d_{p,j} \phi_p(x).$$

In the second step we solve n_1 linear least squares problems of the form

$$\min \|\mathbf{B} \mathbf{h}_i - \mathbf{e}_i\|_2^2, \quad i = 1, 2, \dots, n_1,$$

where the $\mathbf{e}_i$ are the rows of $\mathbf{D}$, and the $\mathbf{h}_i$ are the rows of $\mathbf{C}$

$$\mathbf{D} = \begin{pmatrix} \mathbf{e}_1^T \\ \vdots \\ \mathbf{e}_{n_1}^T \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} \mathbf{h}_1^T \\ \vdots \\ \mathbf{h}_{n_1}^T \end{pmatrix}.$$

Alternatively we can do the computation by first performing a least squares approximation in the y -direction by constructing a family of y -curves, and then use least squares in the x -direction for the lofting. The result will be the same as before. To minimize the number of arithmetic operations one should start with the direction corresponding to the largest of the integers m_1 and m_2 .

Corresponding to Problem 7.4 we have the univariate least squares problem defined in Problem 5.20. Associated with this problem we have an operator $L[\mathbf{x}, \mathbf{w}, \mathbf{f}]$ which to given univariate data $\mathbf{x} = (x_i)_{i=1}^m$ and $\mathbf{f} = (f_i)_{i=1}^m$, and positive weights $\mathbf{w} = (w_i)_{i=1}^m$, assigns a spline

$$g = L[\mathbf{x}, \mathbf{w}, \mathbf{f}] = \sum_{p=1}^n c_p \phi_p,$$

in a spline space $\mathbb{S} = \text{span}\{\phi_1, \dots, \phi_n\}$. We also have the operator $\tilde{L}[\mathbf{x}, \mathbf{w}, \mathbf{f}]$ which maps the data into the B-spline coefficients and is defined analogously to (7.19). With $\tilde{L}_1$ and $\tilde{L}_2$ being least squares operators in the x and y direction, respectively, the B-spline coefficients of the solution of Problem 7.4 can now be written

$$\mathbf{C} = (\tilde{L}_1 \otimes \tilde{L}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}, \mathbf{w}, \mathbf{v}] = \tilde{L}_2[\mathbf{y}, \mathbf{v}, \tilde{L}_1[\mathbf{x}, \mathbf{w}, \mathbf{F}]^T]^T, \quad (7.27)$$

in analogy with the interpolation process (7.20).

7.3 General tensor product methods

In the previous sections we saw how univariate approximation schemes could be combined into a surface scheme for gridded data. Examples of this process is given by (7.20) and (7.27). This technique can in principle be applied quite freely. We could for example combine least squares in the x direction with cubic spline interpolation in the y direction. If $\tilde{Q}_1[\mathbf{x}, \mathbf{f}]$ and $\tilde{Q}_2[\mathbf{y}, \mathbf{g}]$ define univariate approximation methods then we define their tensor product as

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}] = \tilde{Q}_2[\mathbf{y}, \tilde{Q}_1[\mathbf{x}, \mathbf{F}]^T]^T. \quad (7.28)$$

In this section we want to show that

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[x, y, \mathbf{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[y, x, \mathbf{F}^T]$$

for a large class of operators Q_1 and Q_2 . Thus, for such operators we are free to use Q_2 in the y -direction first and then Q_1 in the x -direction, or vice-versa.

We need to specify more abstractly the class of approximation schemes we consider. Suppose $Q[\mathbf{x}, \mathbf{f}]$ is a univariate approximation operator mapping the data into a spline in a univariate spline space

$$\mathbb{S} = \text{span}\{\phi_1, \dots, \phi_n\}.$$

Thus

$$Q[\mathbf{x}, \mathbf{f}] = \sum_{p=1}^n a_p(\mathbf{f}) \phi_p(x). \quad (7.29)$$

The coefficients $a_p(\mathbf{f})$ of the spline are functions of both $\mathbf{x}$ and $\mathbf{f}$, but here we are mostly interested in the dependence of $\mathbf{f}$. We also let $(a_p(\mathbf{f})) = \tilde{Q}[\mathbf{x}, \mathbf{f}]$ be the coefficients of $Q[\mathbf{x}, \mathbf{f}]$. We are interested in the following class of operators Q .

Definition 7.7. The operator $Q: \mathbb{R}^m \rightarrow \mathbb{S}$ given by (7.29) is linear if

$$a_p(\mathbf{f}) = \sum_{i=1}^m a_{p,i} f_i, \quad (7.30)$$

for suitable numbers $a_{p,i}$ independent of $\mathbf{f}$.

If Q is linear then

$$Q[\mathbf{x}, \alpha \mathbf{g} + \beta \mathbf{h}] = \alpha Q[\mathbf{x}, \mathbf{g}] + \beta Q[\mathbf{x}, \mathbf{h}]$$

for all $\alpha, \beta \in \mathbb{R}$ and all $\mathbf{g}, \mathbf{h} \in \mathbb{R}^m$.

Example 7.8. All methods in Chapter 5 are linear approximation schemes.

1. For the Schoenberg Variation Diminishing Spline Approximation we have $\mathbf{f} = (f_1, \dots, f_m) = (f(\tau_1^*), \dots, f(\tau_m^*))$. Thus Vf is of the form (7.29) with $a_p(\mathbf{f}) = f_p$, and $a_{p,i} = \delta_{p,i}$.
2. All the interpolation schemes in Chapter 5, like cubic Hermite, and cubic spline with various boundary conditions are linear. This follows since the coefficients $\mathbf{c} = (c_p)$ are found by solving a linear system $\Phi \mathbf{c} = \mathbf{f}$. Thus $\mathbf{c} = \Phi^{-1} \mathbf{f}$, and c_p is of the form (7.30) with $a_{p,i}$ being the (p, i) -element of Φ^{-1} . For cubic Hermite interpolation we also have the explicit formulas in Proposition 5.5.
3. The least squares approximation method is also a linear approximation scheme. Recall that Q in this case is constructed from the solution of the minimisation problem

$$\min_{\mathbf{c}} \sum_{i=1}^m w_i \left[\sum_{p=1}^n c_p \phi_p(x_i) - f_i \right]^2.$$

The vector $\mathbf{c}$ is determined as the solution of a linear system

$$\mathbf{A}^T \mathbf{A} \mathbf{c} = \mathbf{A}^T \mathbf{f}.$$

Thus $a_{p,i}$ is the (p, i) -element of the matrix $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$.

Consider now the surface situation. Suppose we are given a set of gridded data and two univariate approximation operators Q_1 and Q_2 , and associated with these operators we have the coefficient operators $\tilde{Q}_1$ and $\tilde{Q}_2$ assigning the coefficient vectors to the data.

Proposition 7.9. Suppose Q_1 and Q_2 are linear operators of the form given by (7.29). Then for all data

$$(\mathbf{x}, \mathbf{y}, \mathbf{F}) = (x_i, y_j, f_{i,j})_{i=1, j=1}^{m_1, m_2}, \quad (7.31)$$

we have

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\mathbf{x}, \mathbf{y}, \mathbf{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[\mathbf{y}, \mathbf{x}, \mathbf{F}^T].$$

Proof. To see this we go through the constructions in detail. Suppose that

$$Q_1[\mathbf{x}, \mathbf{f}] = \sum_{p=1}^{n_1} a_p(\mathbf{f}) \phi_p, \quad a_p(\mathbf{f}) = \sum_{i=1}^{m_1} a_{p,i} f_i,$$

$$Q_2[\mathbf{y}, \mathbf{g}] = \sum_{q=1}^{n_2} b_q(\mathbf{g}) \psi_q, \quad b_q(\mathbf{g}) = \sum_{j=1}^{m_2} b_{q,j} g_j.$$

The matrix $\mathbf{F} = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}$ can be partitioned either by rows or by columns.

$$\mathbf{F} = (\mathbf{f}_1, \dots, \mathbf{f}_{m_2}) = \begin{pmatrix} \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{m_1} \end{pmatrix}.$$

If we use Q_1 first then we obtain a family of x -curves from the columns $\mathbf{f}_j$ of the data $\mathbf{F}$

$$Q_1[\mathbf{x}, \mathbf{f}_j] = \sum_{p=1}^{n_1} a_p(\mathbf{f}_j) \phi_p(x), \quad j = 1, 2, \dots, m_2.$$

From these curves we get the final surface

$$g(x, y) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} c_{p,q} \psi_q(y) \phi_p(x),$$

where

$$c_{p,q} = b_q(a_p(\mathbf{f}_1), \dots, a_p(\mathbf{f}_{m_2})).$$

Using the linearity we obtain

$$c_{p,q} = \sum_{j=1}^{m_2} b_{q,j} a_p(\mathbf{f}_j) = \sum_{j=1}^{m_2} \sum_{i=1}^{m_1} b_{q,j} a_{p,i} f_{i,j}. \quad (7.32)$$

Suppose now we use Q_2 first and then Q_1 . We then obtain a surface

$$h(x, y) = \sum_{q=1}^{n_2} \sum_{p=1}^{n_1} d_{p,q} \psi_q(y) \phi_p(x),$$

where

$$d_{p,q} = a_p(b_q(\mathbf{g}_1), \dots, b_q(\mathbf{g}_{m_1})).$$

Thus,

$$d_{p,q} = \sum_{i=1}^{m_1} a_{p,i} b_q(\mathbf{g}_i) = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} a_{p,i} b_{q,j} f_{i,j}.$$

Comparing this with (7.32) we see that $d_{p,q} = c_{p,q}$ for all integers p and q , and hence $g = h$. We conclude that we end up with the same surface in both cases. ■

7.4 Trivariate Tensor Product Methods

The tensor product construction can be extended to higher dimensions. For trivariate approximation we can combine three univariate approximation schemes into a method to approximate trivariate data

$$(x_i, y_j, z_k, f_{i,j,k})_{i=1, j=1, k=1}^{m_1, m_2, m_3}. \quad (7.33)$$

Here the f 's are function values of an unknown trivariate function

$$f = f(x, y, z).$$

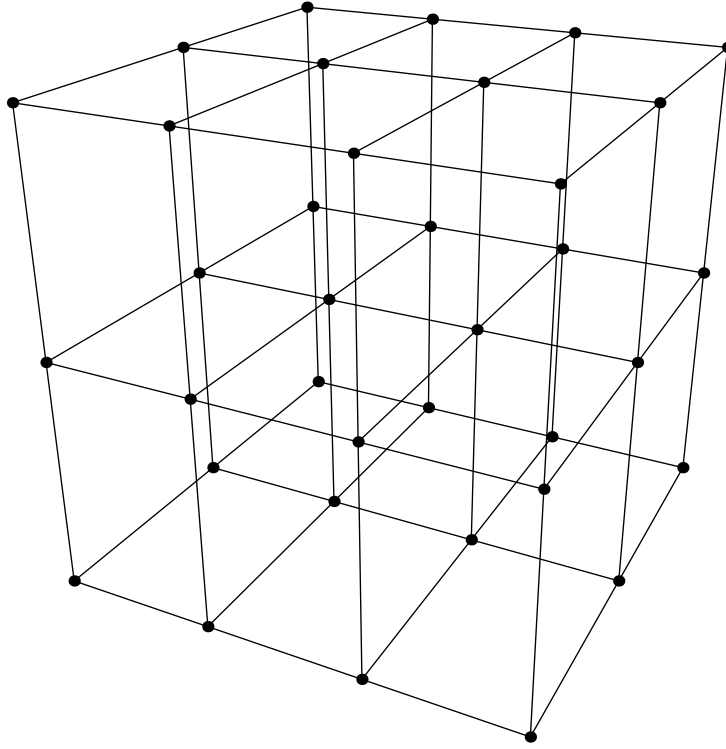


Figure 7.5. A cubical gridded region in space.

The data is given on a cubical region determined from the grid points (x_i, y_j, z_k) in space. We write

$$\mathbf{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1, m_2, m_3}$$

to indicate that the data can be thought of as sitting in a cube of dimensions m_1, m_2, m_3 . Such a cubical grid is shown in Figure 7.5.

The approximation we seek have the form

$$g(x, y, z) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} \sum_{r=1}^{n_3} c_{p,q,r} \omega_r(z) \psi_q(y) \phi_p(x). \quad (7.34)$$

Here

$$\mathbb{S}_1 = \text{span}\{\phi_1, \dots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \text{span}\{\psi_1, \dots, \psi_{n_2}\}, \quad \mathbb{S}_3 = \text{span}\{\omega_1, \dots, \omega_{n_3}\},$$

are three univariate spline spaces spanned by some B-splines. We can construct g by forming

a a sequence of simpler sums as follows

$$\begin{aligned} g(x, y, z) &= \sum_{p=1}^{n_1} d_p(y, z) \phi_p(x), \\ d_p(y, z) &= \sum_{q=1}^{n_2} e_{p,q}(z) \psi_q(y), \\ e_{p,q}(z) &= \sum_{r=1}^{n_3} c_{p,q,r} \omega_r(z). \end{aligned} \tag{7.35}$$

In order to interpolate the data given by (7.33) we obtain the following set of equations

$$\begin{aligned} \sum_{p=1}^{n_1} d_p(y_j, z_k) \phi_p(x_i) &= f_{i,j,k}, \quad i = 1, 2, \dots, m_1, \\ \sum_{q=1}^{n_2} e_{p,q}(z_k) \psi_q(y_j) &= d_p(y_j, z_k), \quad j = 1, 2, \dots, m_2, \\ \sum_{r=1}^{n_3} c_{p,q,r} \omega_r(z_k) &= e_{p,q}(z_k), \quad k = 1, 2, \dots, m_3, \end{aligned} \tag{7.36}$$

These are square systems if $n_i = m_i$, and have to be solved in the least squares sense if $m_i > n_i$ for one or more i .

Consider now writing these systems in matrix form. The equations involve arrays with 3 subscripts. For a positive integer s we define a rank s tensor to be a s -dimensional table of the form

$$\mathbf{A} = (a_{i_1, i_2, \dots, i_s})_{i_1=1, i_2=1, \dots, i_s=1}^{m_1, m_2, \dots, m_s}.$$

We write

$$\mathbf{A} \in \mathbb{R}^{m_1, m_2, \dots, m_s} = \mathbb{R}^{\mathbf{m}},$$

for membership in the class of all rank s tensors with real elements. These *tensors* are generalisations of ordinary vectors and matrices. A rank s tensor can be arranged in a s -dimensional cuboidal array. This is the usual rectangular array for $s = 2$ and a rectangular parallelepiped for $s = 3$.

The operations of addition and scalar multiplication for vectors and matrices extend easily to tensors. The product of two tensors, say $\mathbf{A} \in \mathbb{R}^{m_1, m_2, \dots, m_s}$ and $\mathbf{B} \in \mathbb{R}^{n_1, n_2, \dots, n_e}$ can be defined if the last dimension of $\mathbf{A}$ equals the first dimension of $\mathbf{B}$. Indeed, with $m = m_s = n_1$, we define the product $\mathbf{AB}$ as the tensor

$$\mathbf{C} = \mathbf{AB} \in \mathbb{R}^{m_1, m_2, \dots, m_{s-1}, n_2, \dots, n_e}$$

with elements

$$c_{i_1, \dots, i_{s-1}, j_2, \dots, j_e} = \sum_{i=1}^m a_{i_1, \dots, i_{s-1}, i} b_{i, j_2, \dots, j_e}.$$

For $s = e = 2$ this is the usual product of two matrices, while for $s = e = 1$ we have the inner product of vectors. In general this ‘inner product’ of tensors is a tensor of rank $s + e - 2$. We

just contract the last index of $\mathbf{A}$ and the first index of $\mathbf{B}$. Another product is known as the outer product.

Let us now write the equations in (7.36) in tensor form. The first equation can be written

$$\Phi \mathbf{D} = \mathbf{F}. \quad (7.37)$$

Here

$$\begin{aligned} \Phi &= (\phi_{i,p}) = (\phi_p(x_i)) \in \mathbb{R}^{m_1, n_1}, \\ \mathbf{D} &= (d_{p,j,k}) = d_p(y_j, z_k) \in \mathbb{R}^{n_1, m_2, m_3}, \quad \mathbf{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1, m_2, m_3}. \end{aligned}$$

The system (7.37) is similar to the systems we had earlier for bivariate approximation. We have the same kind of coefficient matrix, but many more right-hand sides.

For the next equation in (7.36) we define

$$\begin{aligned} \Psi &= (\psi_{j,q}) = (\psi_q(y_j)) \in \mathbb{R}^{m_2, n_2}, \\ \mathbf{E} &= (e_{q,k,p}) = (e_{p,q}(z_k)) \in \mathbb{R}^{n_2, m_3, n_1}, \quad \mathbf{D}' = (d_{j,k,p}) \in \mathbb{R}^{m_2, m_3, n_1}. \end{aligned}$$

The next equation can then be written

$$\Psi \mathbf{E} = \mathbf{D}'. \quad (7.38)$$

The construction of $\mathbf{D}'$ from $\mathbf{D}$ involves a cyclic rotation of the dimensions from (n_1, m_2, m_3) to (m_2, m_3, n_1) . The same operation is applied to $\mathbf{E}$ for the last equation in (7.36). We obtain

$$\Omega \mathbf{G} = \mathbf{E}', \quad (7.39)$$

where

$$\begin{aligned} \Omega &= (\omega_{k,r}) = (\omega_r(z_k)) \in \mathbb{R}^{m_3, n_3}, \\ \mathbf{E}' &= (e_{k,p,q}) = (e_{p,q}(z_k)) \in \mathbb{R}^{m_3, n_1, n_2}, \quad \mathbf{G} = (g_{r,p,q}) \in \mathbb{R}^{n_3, n_1, n_2}. \end{aligned}$$

The coefficients $\mathbf{C}'$ are obtained by a final cyclic rotation of the dimensions

$$\mathbf{C} = \mathbf{G}'. \quad (7.40)$$

The systems (7.37), (7.38), and (7.39) corresponds to three univariate operators of the form $Q[\mathbf{x}, \mathbf{f}]$. We denote these Q_1, Q_2 , and Q_3 . We assume that Q_i can be applied to a tensor. The tensor product of these three operators can now be defined as follows

$$(Q_1 \otimes Q_2 \otimes Q_3)[\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{F}] = Q_3[\mathbf{z}, Q_2[\mathbf{y}, Q_1[\mathbf{x}, \mathbf{F}]]']'. \quad (7.41)$$

The actual implementation of this scheme on a computer will depend on how arrays are sorted in the actual programming language used. Some languages arrange by columns, while others arrange by rows.

7.5 Parametric Surfaces

Parametric curves and explicit surfaces have a natural generalisation to parametric surfaces. Let us consider the plane P through three points in space which we call $\mathbf{p}_0$, $\mathbf{p}_1$ and $\mathbf{p}_2$. We define the function $\mathbf{f} : \mathbb{R}^2 \mapsto P$ by

$$\mathbf{f}(u, v) = \mathbf{p}_0 + (\mathbf{p}_1 - \mathbf{p}_0)u + (\mathbf{p}_2 - \mathbf{p}_0)v. \quad (7.42)$$

We see that $\mathbf{f}(0, 0) = \mathbf{p}_0$, while $\mathbf{f}(1, 0) = \mathbf{p}_1$ and $\mathbf{f}(0, 1) = \mathbf{p}_2$, so that $\mathbf{f}$ interpolates the three points. Since $\mathbf{f}$ is also a linear function, we conclude that it is indeed a representation for the plane P .

We start by generalising and formalising this.

Definition 7.10. A parametric representation of class C^m of a set $S \subseteq \mathbb{R}^3$ is a mapping $\mathbf{f}$ of an open set $\Omega \subseteq \mathbb{R}^2$ onto S such that

- (i) $\mathbf{f}$ has continuous derivatives up to order m .

Suppose that $\mathbf{f}(u, v) = (f^1(u, v), f^2(u, v), f^3(u, v))$ and let $D_1\mathbf{f}$ and $D_2\mathbf{f}$ denote differentiation with respect to the first and second variables of $\mathbf{f}$, respectively. The parametric representation $\mathbf{f}$ is said to be regular if in addition

- (ii) the Jacobian matrix of $\mathbf{f}$ given by

$$J(\mathbf{f}) = \begin{pmatrix} D_1 f^1(u, v) & D_2 f^1(u, v) \\ D_1 f^2(u, v) & D_2 f^2(u, v) \\ D_1 f^3(u, v) & D_2 f^3(u, v) \end{pmatrix}$$

has full rank for all (u, v) in Ω .

That $J(\mathbf{f})$ has full rank means that its two columns must be linearly independent for all $(u, v) \in \Omega$, or equivalently, that for all (u, v) there must be at least one nonsingular 2×2 submatrix of $J(\mathbf{f})$.

A function of two variables $z = h(x, y)$ can always be considered as a parametric surface through the representation $\mathbf{f}(u, v) = (u, v, h(u, v))$. In the following we will always assume that $\mathbf{f}$ is sufficiently smooth for all operations on $\mathbf{f}$ to make sense.

It turns out that there are many surfaces that cannot be described as the image of a regular parametric representation. One example is a sphere. It can be shown that it is impossible to find one regular parametric representation that can cover the whole sphere. Instead one uses several parametric representations to cover different parts of the sphere and call the collection of such representations a parametric surface. For our purposes this is unnecessary, since we are only interested in analysing a single parametric representation given as a spline. We will therefore often adopt the sloppy convention of referring to a parametric representation as a surface.

Let us check that the surface given by (7.42) is regular. The Jacobian matrix is easily computed as

$$J(\mathbf{f}) = (\mathbf{p}_1 - \mathbf{p}_0, \mathbf{p}_2 - \mathbf{p}_0),$$

(the two vectors $\mathbf{p}_1 - \mathbf{p}_0$ and $\mathbf{p}_2 - \mathbf{p}_0$ give the columns of $J(\mathbf{f})$). We see that $J(\mathbf{f})$ has full rank unless $\mathbf{p}_1 - \mathbf{p}_0 = \lambda(\mathbf{p}_2 - \mathbf{p}_0)$ for some real number λ , i.e., unless all three points lie on a straight line.

A curve on the surface S of the form $\mathbf{f}(u, v_0)$ for fixed v_0 is called a u -curve, while a curve of the form $\mathbf{f}(u_0, v)$ is called a v -curve. A collective term for such curves is *iso-parametric curves*.

Iso-parametric curves are often useful for plotting. By drawing a set of u - and v -curves, one gets a simple but good impression of the surface.

The first derivatives $D_1\mathbf{f}(u, v)$ and $D_2\mathbf{f}(u, v)$ are derivatives of, and therefore tangent to, a u - and v -curve respectively. For a regular surface the two first derivatives are linearly independent and therefore the cross product $D_1\mathbf{f}(u, v) \times D_2\mathbf{f}(u, v)$ is nonzero and normal to the two tangent vectors.

Definition 7.11. The unit normal of the regular parametric representation $\mathbf{f}$ is the vector

$$\mathbf{N}(u, v) = \frac{D_1\mathbf{f}(u, v) \times D_2\mathbf{f}(u, v)}{\|D_1\mathbf{f}(u, v) \times D_2\mathbf{f}(u, v)\|}.$$

The normal vector will play an important role when we start analysing the curvature of surfaces.

Let $(u(\sigma), v(\sigma))$ be a regular curve in the domain Ω of a parametric representation $\mathbf{f}$. This curve is mapped to a curve $\mathbf{g}(\sigma)$ on the surface,

$$\mathbf{g}(\sigma) = \mathbf{f}(u(\sigma), v(\sigma)).$$

The tangent of $\mathbf{g}$ is given by

$$\mathbf{g}'(\sigma) = u'(\sigma)D_1\mathbf{f}(u(\sigma), v(\sigma)) + v'(\sigma)D_2\mathbf{f}(u(\sigma), v(\sigma)),$$

in other words, a linear combination of the two tangent vectors $D_1\mathbf{f}(u(\sigma), v(\sigma))$ and $D_2\mathbf{f}(u(\sigma), v(\sigma))$. Note that $\mathbf{g}$ is regular since $\mathbf{g}'(\sigma) = 0$ implies $u'(\sigma) = v'(\sigma) = 0$.

All regular curves on S through the point $\mathbf{f}(u, v)$ has a tangent vector on the form $\delta_1 D_1\mathbf{f} + \delta_2 D_2\mathbf{f}$, where $\boldsymbol{\delta} = (\delta^1, \delta^2)$ is a vector in $\mathbb{R}^2$. The space of all such tangent vectors is the tangent plane of S at $\mathbf{f}(u, v)$.

Definition 7.12. Let S be a surface with a regular parametric representation $\mathbf{f}$. The tangent space or tangent plane $T\mathbf{f}(u, v)$ of S at $\mathbf{f}(u, v)$ is the plane in $\mathbb{R}^3$ spanned by the two vectors $D_1\mathbf{f}(u, v)$ and $D_2\mathbf{f}(u, v)$, i.e., all vectors on the form $\delta_1 D_1\mathbf{f}(u, v) + \delta_2 D_2\mathbf{f}(u, v)$.

Note that the normal of the tangent plane $T\mathbf{f}(u, v)$ is the normal vector $\mathbf{N}(u, v)$.

7.5.1 Parametric Tensor Product Spline Surfaces

Recalling how we generalized from spline functions to parametric spline curves, we see that the definition of parametric tensor product spline surfaces is the obvious generalization of tensor product spline functions.

Definition 7.13. A parametric tensor product spline surface is given by a parametric representation on the form

$$\mathbf{f}(u, v) = \sum_{i=1}^m \sum_{j=1}^n \mathbf{c}_{i,j} B_{i,d,\sigma}(u) B_{j,\ell,\tau}(v),$$

where the coefficients $(\mathbf{c}_{i,j})_{i,j=1}^{m,n}$ are points in space,

$$\mathbf{c}_{i,j} = (c_{i,j}^1, c_{i,j}^2, c_{i,j}^3),$$

and $\boldsymbol{\sigma} = (\sigma_i)_{i=1}^{m+d+1}$ and $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+\ell+1}$ are knot vectors for splines of degrees d and ℓ .

As for curves, algorithms for tensor product spline surfaces can easily be adapted to give methods for approximation with parametric spline surfaces. Again, as for curves, the only complication is the question of parametrization, but we will not consider this in more detail here.

Quasi-interpolation methods

In Chapter 5 we considered a number of methods for computing spline approximations. The starting point for the approximation methods is a data set that is usually discrete and in the form of function values given at a set of abscissas. The methods in Chapter 5 roughly fall into two categories: global methods and local methods. A global method is one where any B-spline coefficient depends on all initial data points, whereas a local method is one where a B-spline coefficient only depends on data points taken from the neighbourhood of the support of the corresponding B-spline. Typical global methods are cubic spline interpolation and least squares approximation, while cubic Hermite interpolation and the Schoenberg variation diminishing spline approximation are popular local methods.

In this chapter we are going to describe a general recipe for developing local spline approximation methods. This will enable us to produce an infinite number of approximation schemes that can be tailored to any special needs that we may have or that our given data set dictates. In principle, the methods are local, but by allowing the area of influence for a given B-spline coefficient to grow, our general recipe may even encompass the global methods in Chapter 5.

The recipe we describe produces approximation methods known under the collective term *quasi-interpolation methods*. Their advantage is their flexibility and their simplicity. There is considerable freedom in the recipe to produce tailor-made approximation schemes for initial data sets with special structure. Quasi-interpolants also allow us to establish important properties of B-splines. In the next chapter we will employ them to study how well a given function can be approximated by splines, and to show that B-splines form a stable basis for splines.

8.1 A general recipe

A spline approximation method consists of two main steps: First the degree and knot vector are determined, and then the B-spline coefficients of the approximation are computed from given data according to some formula. For some methods like spline interpolation and least squares approximation, this formula corresponds to the solution of a linear system of

equations. In other cases, like cubic Hermite interpolation and Schoenberg's Variation Diminishing spline approximation, the formula for the coefficients is given directly in terms of given values of the function to be interpolated.

8.1.1 The basic idea

The basic idea behind the construction of quasi-interpolants is very simple. We focus on how to compute the B-spline coefficients of the approximation and assume that the degree and knot vector are known. The procedure depends on two versions of the local support property of B-splines that we know well from earlier chapters: (i) The B-spline B_j is nonzero only within the interval $[\tau_j, \tau_{j+d+1}]$, and (ii) on the interval $[\tau_\mu, \tau_{\mu+1})$ there are only $d+1$ B-splines in $\mathbb{S}_{d,\tau}$ that are nonzero so a spline g in $\mathbb{S}_{d,\tau}$ can be written as $g(x) = \sum_{i=\mu-d}^{\mu} b_i B_i(x)$ when x is restricted to this interval.

Suppose we are to compute an approximation $g = \sum_i c_i B_i$ in $\mathbb{S}_{d,\tau}$ to a given function f . To compute c_j we can select one knot interval $I = [\tau_\mu, \tau_{\mu+1}]$ which is a subinterval of $[\tau_j, \tau_{j+d+1}]$. We denote the restriction of f to this interval by f^I and determine an approximation $g^I = \sum_{i=\mu-d}^{\mu} b_i B_i$ to f^I . One of the coefficients of g^I will be b_j and we fix c_j by setting $c_j = b_j$. The whole procedure is then repeated until all the c_i have been determined.

It is important to note the flexibility of this procedure. In choosing the interval I we will in general have the $d+1$ choices $\mu = j, j+1, \dots, j+d$ (fewer if there are multiple knots). As we shall see below we do not necessarily have to restrict I to be one knot interval; all that is required is that $I \cap [\tau_\mu, \tau_{\mu+d+1}]$ is nonempty. When approximating f^I by g^I we have a vast number of possibilities. We may use interpolation or least squares approximation, or any other approximation method. Suppose we settle for interpolation, then we have complete freedom in choosing the interpolation points within the interval I . In fact, there is so much freedom that we can have no hope of exploring all the possibilities.

It turns out that some of this freedom is only apparent — to produce useful quasi-interpolants we have to enforce certain conditions. With the general setup described above, a useful restriction is that if f^I should happen to be a polynomial of degree d then g^I should reproduce f^I , i.e., in this case we should have $g^I = f^I$. This has the important consequence that if f is a spline in $\mathbb{S}_{d,\tau}$ then the approximation g will reproduce f exactly (apart from rounding errors in the numerical computations). To see why this is the case, suppose that $f = \sum_i \hat{b}_i B_i$ is a spline in $\mathbb{S}_{d,\tau}$. Then f^I will be a polynomial that can be written as $f^I = \sum_{i=\mu-d}^{\mu} \hat{b}_i B_i$. Since we have assumed that polynomials will be reproduced we know that $g^I = f^I$ so $\sum_{i=\mu-d}^{\mu} b_i B_i = \sum_{i=\mu-d}^{\mu} \hat{b}_i B_i$, and by the linear independence of the B-splines involved we conclude that $b_i = \hat{b}_i$ for $i = \mu-d, \dots, \mu$. But then we see that $c_j = b_j = \hat{b}_j$ so g will agree with f . An approximation scheme with the property that $Pf = f$ for all f in a space $\mathbb{S}$ is said to *reproduce* the space.

8.1.2 A more detailed description

Hopefully, the basic idea behind the construction of quasi-interpolants became clear above. In this section we describe the construction in some more detail with the generalisations mentioned before. We first write down the general procedure for determining quasi-interpolants and then comment on the different steps afterwards.

Algorithm 8.1 (Construction of quasi-interpolants). *Let the spline space $\mathbb{S}_{d,\tau}$ of dimension n and the real function f defined on the interval $[\tau_{d+1}, \tau_{n+1}]$ be given, and suppose that τ is a $d+1$ -regular knot vector. To approximate f from the space $\mathbb{S}_{d,\tau}$ perform the following steps for $j = 1, 2, \dots, n$:*

1. *Choose a subinterval $I = [\tau_\mu, \tau_\nu]$ of $[\tau_{d+1}, \tau_{n+1}]$ with the property that $I \cap (\tau_j, \tau_{j+d+1})$ is nonempty, and let f^I denote the restriction of f to this interval.*
2. *Choose a local approximation method P^I and determine an approximation g^I to f^I ,*

$$g^I = P^I f^I = \sum_{i=\mu-d}^{\nu-1} b_i B_i, \quad (8.1)$$

on the interval I .

3. *Set coefficient j of the global approximation Pf to b_j , i.e.,*

$$c_j = b_j.$$

The spline $Pf = \sum_{j=1}^n c_j B_j$ will then be an approximation to f .

The coefficient c_j obviously depends on f and this dependence on f is often indicated by using the notation $\lambda_j f$ for c_j . This will be our normal notation in the rest of the chapter.

An important point to note is that the restriction $\mathbb{S}_{d,\tau,I}$ of the spline space $\mathbb{S}_{d,\tau}$ to the interval I can be written as a linear combination of the B-splines $\{B_i\}_{i=\mu-d}^{\nu-1}$. These are exactly the B-splines whose support intersect the interior of the interval I , and by construction, one of them must clearly be B_j . This ensures that the coefficient b_j that is needed in step 3 is computed in step 2.

Algorithm 8.1 generalizes the simplified procedure in Section 8.1.1 in that I is no longer required to be a single knot interval in $[\tau_j, \tau_{j+d+1}]$. This gives us considerably more flexibility in the choice of local approximation methods. Note in particular that the classical global methods are included as special cases since we may choose $I = [\tau_{d+1}, \tau_{n+1}]$.

As we mentioned in Section 8.1.1, we do not get good approximation methods for free. If Pf is going to be a decent approximation to f we must make sure that the local methods used in step 2 reproduce polynomials or splines.

Lemma 8.2. *Suppose that all the local methods used in step 2 of Algorithm 8.1 reproduce all polynomials of some degree $d_1 \leq d$. Then the global approximation method P will also reproduce polynomials of degree d_1 . If all the local methods reproduce all the splines in $\mathbb{S}_{d,\tau,I}$ then P will reproduce the whole spline space $\mathbb{S}_{d,\tau}$.*

Proof. The proof of both claims follow just as in the special case in Section 8.1.1, but let us even so go through the proof of the second claim. We want to prove that if all the local methods P^I reproduce the local spline spaces $\mathbb{S}_{d,\tau,I}$ and f is a spline in $\mathbb{S}_{d,\tau}$, then $Pf = f$. If f is in $\mathbb{S}_{d,\tau}$ we clearly have $f = \sum_{i=1}^n \hat{b}_i B_i$ for appropriate coefficients $(\hat{b}_i)_{i=1}^n$, and the restriction

of f to I can be represented as $f^I = \sum_{i=\mu-d}^{v-1} \hat{b}_i B_i$. Since P^I reproduces $\mathbb{S}_{d,\tau,I}$ we will have $P^I f^I = f^I$ or

$$\sum_{i=\mu-d}^{v-1} b_i B_i = \sum_{i=\mu-d}^{v-1} \hat{b}_i B_i.$$

The linear independence of the B-splines involved over the interval I then allows us to conclude that $b_i = \hat{b}_i$ for all indices i involved in this sum. Since j is one of the indices we therefore have $c_j = b_j = \hat{b}_j$. When this holds for all values of j we obviously have $Pf = f$. ■

The reader should note that if I is a single knot interval, the local spline space $\mathbb{S}_{d,\tau,I}$ reduces to the space of polynomials of degree d . Therefore, when I is a single knot interval, local reproduction of polynomials of degree d leads to global reproduction of the whole spline space.

Why does reproduction of splines or polynomials ensure that P will be a good approximation method? We will study this in some detail in Chapter 9, but as is often the case the basic idea is simple: The functions we want to approximate are usually nice and smooth, like the exponential functions or the trigonometric functions. An important property of polynomials is that they approximate such smooth functions well, although if the interval becomes wide we may need to use polynomials of high degree. A quantitative manifestation of this phenomenon is that if we perform a Taylor expansion of a smooth function, then the error term will be small, at least if the degree is high enough. If our approximation method reproduces polynomials it will pick up the essential behaviour of the Taylor polynomial, while the approximation error will pick up the essence of the error in the Taylor expansion. The approximation method will therefore perform well whenever the error in the Taylor expansion is small. If we reproduce spline functions we can essentially reproduce Taylor expansions on each knot interval as long as the function we approximate has at least the same smoothness as the splines in the spline space we are using. So instead of increasing the polynomial degree because we are approximating over a wide interval, we can keep the spacing in the knot vector small and thereby keep the polynomial degree of the spline low. Another way to view this is that by using splines we can split our function into suitable pieces that each can be approximated well by polynomials of relatively low degree, even though this is not possible for the complete function. By constructing quasi-interpolants as outlined above we obtain approximation methods that actually utilise this approximation power of polynomials on each subinterval. In this way we can produce good approximations even to functions that are only piecewise smooth.

8.2 Some quasi-interpolants

It is high time to try out our new tool for constructing approximation methods. Let us see how some simple methods can be obtained from Algorithm 8.1.

8.2.1 Piecewise linear interpolation

Perhaps the simplest, local approximation method is piecewise linear interpolation. We assume that our n -dimensional spline space $\mathbb{S}_{1,\tau}$ is given and that τ is a 2-regular knot vector. For simplicity we also assume that all the interior knots are simple. The function f is given

on the interval $[\tau_2, \tau_{n+1}]$. To determine c_j we choose the local interval to be $I = [\tau_j, \tau_{j+1}]$. In this case, we have no interior knots in I so $\mathbb{S}_{1,\tau,I}$ is the two dimensional space of linear polynomials. A basis for this space is given by the two linear B-splines B_{j-1} and B_j , restricted to the interval I . A natural candidate for our local approximation method is interpolation at τ_j and τ_{j+1} . On the interval I , the B-spline B_{j-1} is a straight line with value 1 at τ_j and value 0 at τ_{j+1} , while B_j is a straight line with value 0 at τ_j and value 1 at τ_{j+1} . The local interpolant can therefore be written

$$P_1^I f(x) = f(\tau_j)B_{j-1}(x) + f(\tau_{j+1})B_j(x).$$

From Algorithm 8.1 we know that the coefficient multiplying B_j is the one that should multiply B_j also in our global approximation, in other words $c_j = \lambda_j f = f(\tau_{j+1})$. The global approximation is therefore

$$P_1 f(x) = \sum_{j=1}^n f(\tau_{j+1})B_j(x).$$

Since a straight line is completely characterized by its value at two points, the local approximation will always give zero error and therefore reproduce all linear polynomials. Then we know from Lemma 8.2 that P_1 will reproduce all splines $\mathbb{S}_{1,\tau}$.

This may seem like unnecessary formalism in this simple case where the conclusions are almost obvious, but it illustrates how the construction works in a very transparent situation.

8.2.2 A 3-point quadratic quasi-interpolant

In our repertoire of approximation methods, we only have one local, quadratic method, Schoenberg's variation diminishing spline. With the quasi-interpolant construction it is easy to construct alternative, local methods. Our starting point is a quadratic spline space $\mathbb{S}_{2,\tau}$ based on a 3-regular knot vector with distinct interior knots, and a function f to be approximated by a scheme which we denote P_2 . The support of the B-spline B_j is $[\tau_j, \tau_{j+3}]$, and we choose our local interval as $I = [\tau_{j+1}, \tau_{j+2}]$. Since I is one knot interval, we need a local approximation method that reproduces quadratic polynomials. One such method is interpolation at three distinct points. We therefore choose three distinct points $x_{j,0}$, $x_{j,1}$ and $x_{j,2}$ in I . Some degree of symmetry is always a good guide so we choose

$$x_{j,0} = \tau_{j+1}, \quad x_{j,1} = \frac{\tau_{j+1} + \tau_{j+2}}{2}, \quad x_{j,2} = \tau_{j+2}.$$

To determine $P_2^I f$ we have to solve the linear system of three equations in the three unknowns b_{j-1} , b_j and b_{j+1} given by

$$P_2^I f(x_{j,k}) = \sum_{i=j-1}^{j+1} b_i B_i(x_{j,k}) = f(x_{j,k}), \quad \text{for } k = 0, 1, 2.$$

With the aid of a tool like Mathematica we can solve these equations symbolically. The result is that

$$b_j = \frac{1}{2}(-f(\tau_{j+1}) + 4f(\tau_{j+3/2}) - f(\tau_{j+2})),$$

where $\tau_{j+3/2} = (\tau_{j+1} + \tau_{j+2})/2$. The expressions for b_{j-1} and b_{j+1} are much more complicated and involve the knots τ_j and τ_{j+3} as well. The simplicity of the expression for b_j stems from the fact that $x_{j,1}$ was chosen as the midpoint between τ_{j+1} and τ_{j+2} .

The expression for b_j is valid whenever $\tau_{j+1} < \tau_{j+2}$ which is not the case for $j = 1$ and $j = n$ since $\tau_1 = \tau_2 = \tau_3$ and $\tau_{n+1} = \tau_{n+2} = \tau_{n+3}$. But from Lemma 2.9 we know that any spline g in $\mathbb{S}_{3,\tau}$ will interpolate its first and last B-spline coefficient at these points so we simply set $c_1 = f(\tau_1)$ and $c_n = f(\tau_{n+1})$.

Having constructed the local interpolants, we have all the ingredients necessary to construct the quasi-interpolant $P_2 f = \sum_{j=1}^n \lambda_j f B_j$, namely

$$\lambda_j f = \begin{cases} f(\tau_1), & \text{when } j = 1; \\ \frac{1}{2}(-f(x_{j,0}) + 4f(x_{j,1}) - f(x_{j,2})), & \text{when } 1 < j < n; \\ f(\tau_{n+1}), & \text{when } j = n. \end{cases}$$

Since the local approximation reproduced the local spline space (the space of quadratic polynomials in this case), the complete quasi-interpolant will reproduce the whole spline space $\mathbb{S}_{2,\tau}$.

8.2.3 A 5-point cubic quasi-interpolant

The most commonly used splines are cubic, so let us construct a cubic quasi-interpolant. We assume that the knot vector is 4-regular and that the interior knots are all distinct. As usual we focus on the coefficient $c_j = \lambda_j f$. It turns out that the choice $I = [\tau_{j+1}, \tau_{j+3}]$ is convenient. The local spline space $\mathbb{S}_{3,\tau,I}$ has dimension 5 and is spanned by the (restriction of the) B-splines $\{B_i\}_{i=j-2}^{j+2}$. We want the quasi-interpolant to reproduce the whole spline space and therefore need P^I to reproduce $\mathbb{S}_{3,\tau,I}$. We want to use interpolation as our local approximation method, and we know from Chapter 5 that spline interpolation reproduces the spline space as long as it has a unique solution. The solution is unique if the coefficient matrix of the resulting linear system is nonsingular, and from Theorem 5.18 we know that a B-spline coefficient matrix is nonsingular if and only if its diagonal is positive. Since the dimension of $\mathbb{S}_{3,\tau,I}$ is 5 we need 5 interpolation points. We use the three knots τ_{j+1} , τ_{j+2} and τ_{j+3} and one point from each of the knot intervals in I ,

$$x_{j,0} = \tau_{j+1}, \quad x_{j,1} \in (\tau_{j+1}, \tau_{j+2}), \quad x_{j,2} = \tau_{j+2}, \quad x_{j,3} \in (\tau_{j+2}, \tau_{j+3}), \quad x_{j,4} = \tau_{j+3}.$$

Our local interpolation problem is

$$\sum_{i=j-2}^{j+2} b_i B_i(x_{j,k}) = f(x_{j,k}), \quad \text{for } k = 0, 1, \dots, 4.$$

In matrix-vector form this becomes

$$\begin{pmatrix} B_{j-2}(x_{j,0}) & B_{j-1}(x_{j,0}) & B_j(x_{j,0}) & 0 & 0 \\ B_{j-2}(x_{j,1}) & B_{j-1}(x_{j,1}) & B_j(x_{j,1}) & B_{j+1}(x_{j,1}) & 0 \\ B_{j-2}(x_{j,2}) & B_{j-1}(x_{j,2}) & B_j(x_{j,2}) & B_{j+1}(x_{j,2}) & B_{j+2}(x_{j,2}) \\ 0 & B_{j-1}(x_{j,3}) & B_j(x_{j,3}) & B_{j+1}(x_{j,3}) & B_{j+2}(x_{j,3}) \\ 0 & 0 & B_j(x_{j,4}) & B_{j+1}(x_{j,4}) & B_{j+2}(x_{j,4}) \end{pmatrix} \begin{pmatrix} b_{j-2} \\ b_{j-1} \\ b_j \\ b_{j+1} \\ b_{j+2} \end{pmatrix} = \begin{pmatrix} f(x_{j,0}) \\ f(x_{j,1}) \\ f(x_{j,2}) \\ f(x_{j,3}) \\ f(x_{j,4}) \end{pmatrix}$$

when we insert the matrix entries that are zero. Because of the way we have chosen the interpolation points we see that all the entries on the diagonal of the coefficient matrix will be positive so the matrix is nonsingular. The local problem therefore has a unique solution and will reproduce $\mathbb{S}_{3,\tau,I}$. The expression for $\lambda_j f$ is in general rather complicated, but in the special case where the widths of the two knot intervals are equal and $x_{j,2}$ and $x_{j,4}$ are chosen as the midpoints of the two intervals we end up with

$$\lambda_j f = \frac{1}{6} (f(\tau_{j+1}) - 8f(\tau_{j+3/2}) + 20f(\tau_{j+2}) - 8f(\tau_{j+5/2}) + f(\tau_{j+3}))$$

where $\tau_{j+3/2} = (\tau_{j+1} + \tau_{j+2})/2$ and $\tau_{j+5/2} = (\tau_{j+2} + \tau_{j+3})/2$. Unfortunately, this formula is not valid when $j = 1, 2, n-1$ or n since then one or both of the knot intervals in I collapse to one point. However, our procedure is sufficiently general to derive alternative formulas for computing the first two coefficients. The first value of j for which the general procedure works is $j = 3$. In this case $I = [\tau_4, \tau_6]$ and our interpolation problem involves the B-splines $\{B_i\}_{i=1}^5$. This means that when we solve the local interpolation problem we obtain B-spline coefficients multiplying all of these B-splines, including B_1 and B_2 . There is nothing stopping us from using the same interval I for computation of several coefficients, so in addition to obtaining $\lambda_3 f$ from this local interpolant, we also use it as our source for the first two coefficients. In the special case when the interior knots are uniformly distributed and $x_{3,1} = \tau_{9/2}$ and $x_{3,3} = \tau_{11/2}$, we find

$$\begin{aligned} \lambda_1 f &= f(\tau_4), \\ \lambda_2 f &= \frac{1}{18} (-5f(\tau_4) + 40f(\tau_{9/2}) - 36f(\tau_5) + 18f(\tau_{11/2}) - f(\tau_6)). \end{aligned}$$

In general, the second coefficient will be much more complicated, but the first one will not change.

This same procedure can obviously be used to determine values for the last two coefficients, and under the same conditions of uniformly distributed knots and interpolation points we find

$$\begin{aligned} \lambda_{n-1} f &= \frac{1}{18} (-f(\tau_{n-1}) + 18f(\tau_{n-1/2}) - 36f(\tau_n) + 40f(\tau_{n+1/2}) - 5f(\tau_{n+1})), \\ \lambda_n f &= f(\tau_{n+1}). \end{aligned}$$

8.2.4 Some remarks on the constructions

In all our constructions, we have derived specific formulas for the B-spline coefficients of the quasi-interpolants in terms of the function f to be approximated, which makes it natural to use the notation $c_j = \lambda_j f$. To do this, we had to solve the local linear system of equations symbolically. When the systems are small this can be done quite easily with a computer algebra system like Maple or Mathematica, but the solutions quickly become complicated and useless unless the knots and interpolation points are nicely structured, preferably with uniform spacing. The advantage of solving the equations symbolically is of course that we obtain explicit formulas for the coefficients once and for all and can avoid solving equations when we approximate a particular function.

For general knots, the local systems of equations usually have to be solved numerically, but quasi-interpolants can nevertheless prove very useful. One situation is real-time processing of data. Suppose we are in a situation where data are measured and need to be fitted with a spline in real time. With a global approximation method we would have to recompute the whole spline each time we receive new data. This would be acceptable at the beginning, but as the data set grows, we would not be able to compute the new approximation quickly enough. We could split the approximation into smaller pieces at regular intervals, but quasi-interpolants seem to be a perfect tool for this kind of application. In a real-time application the data will often be measured at fixed time intervals, and as we have seen it is then easy to construct quasi-interpolants with explicit formulas for the coefficients. Even if this is not practicable because the explicit expressions are not available or become too complicated, we just have to solve a simple, linear set of equations to determine each new coefficient. The important fact is that the size of the system is constant so that we can handle almost arbitrarily large data sets, the only limitation being available storage space.

Another important feature of quasi-interpolants is their flexibility. In our constructions we have assumed that the function we approximate can be evaluated at any point that we need. This may sometimes be the case, but often the function is only partially known by a few discrete, measured values at specific abscissas. The procedure for constructing quasi-interpolants has so much inherent freedom that it can be adapted in a number of ways to virtually any specific situation, whether the whole data set is available a priori or the approximation has to be produced in real-time as the data is generated.

8.3 Quasi-interpolants are linear operators

Now that we have seen some examples of quasi-interpolants, let us examine them from a more general point of view. The basic ingredient of quasi-interpolants is the computation of each B-spline coefficient, and we have used the notation $c_j = \lambda_j f = \lambda_j(f)$ to indicate that each coefficient depends on f . It is useful to think of λ_j as a 'function' that takes an ordinary function as input and gives a real number as output; such 'functions' are usually called functionals. If we go back and look at our examples, we notice that in each case the dependency of our coefficient functionals on f is quite simple: The function values occur explicitly in the coefficient expressions and are not multiplied or operated on in any way other than being added together and multiplied by real numbers. This is familiar from linear algebra.

Definition 8.3. *In the construction of quasi-interpolants, each B-spline coefficient is computed by evaluating a linear functional. A linear functional λ is a mapping from a suitable space of functions $\mathbb{S}$ into the real numbers $\mathbb{R}$ with the property that if f and g are two arbitrary functions in $\mathbb{S}$ and α and β are two real numbers then*

$$\lambda(\alpha f + \beta g) = \alpha \lambda f + \beta \lambda g.$$

Linearity is a necessary property of a functional that is being used to compute B-spline coefficients in the construction of quasi-interpolants. If one of the coefficient functionals is nonlinear, then the resulting approximation method is not a quasi-interpolant. Linearity of the coefficient functionals leads to linearity of the approximation scheme.

Lemma 8.4. Any quasi-interpolant P is a linear operator, i.e., for any two admissible functions f and g and any real numbers α and β ,

$$P(\alpha f + \beta g) = \alpha P f + \beta P g.$$

Proof. Suppose that the linear coefficient functionals are $(\lambda_j)_{j=1}^n$. Then we have

$$P(\alpha f + \beta g) = \sum_{i=1}^n \lambda_j(\alpha f + \beta g) B_i = \alpha \sum_{i=1}^n \lambda_j f B_i + \beta \sum_{i=1}^n \lambda_j g B_i = \alpha P f + \beta P g$$

which demonstrates the linearity of P . ■

This lemma is simple, but very important since there are so many powerful mathematical tools available to analyse linear operators. In Chapter 9 we are going to see how well a given function can be approximated by splines. We will do this by applying basic tools in the analysis of linear operators to some specific quasi-interpolants.

8.4 Different kinds of linear functionals and their uses

In our examples of quasi-interpolants in Section 8.2 the coefficient functionals were all linear combinations of function values, but there are other functionals that can be useful. In this section we will consider some of these and how they turn up in approximation problems.

8.4.1 Point functionals

Let us start by recording the form of the functionals that we have already encountered. The coefficient functionals in Section 8.2 were all in the form

$$\lambda f = \sum_{i=0}^{\ell} w_i f(x_i) \tag{8.2}$$

for suitable numbers $(w_i)_{i=0}^{\ell}$ and $(x_i)_{i=0}^{\ell}$. Functionals of this kind can be used if a procedure is available to compute values of the function f or if measured values of f at specific points are known. Most of our quasi-interpolants will be of this kind.

Point functionals of this type occur naturally in at least two situations. The first is when the local approximation method is interpolation, as in our examples above. The second is when the local approximation method is discrete least squares approximation. As a simple example, suppose our spline space is $\mathbb{S}_{2,\tau}$ and that in determining c_j we consider the single knot interval $I = [\tau_{j+1}, \tau_{j+2}]$. Suppose also that we have 10 function values at the points $(x_{j,k})_{k=0}^9$ in this interval. Since the dimension of $\mathbb{S}_{2,\tau,I}$ is 3, we cannot interpolate all 10 points. The solution is to perform a local least squares approximation and determine the local approximation by minimising the sum of the squares of the errors,

$$\min_{g \in \mathbb{S}_{2,\tau,I}} \sum_{k=0}^9 (g(x_{j,k}) - f(x_{j,k}))^2.$$

The result is that c_j will be a linear combination of the 10 function values,

$$c_j = \lambda_j f = \sum_{k=0}^9 w_{j,k} f(x_{j,k}).$$

8.4.2 Derivative functionals

In addition to function values, we can also compute derivatives of a function at a point. Since differentiation is a linear operator it is easy to check that a functional like $\lambda f = f''(x_i)$ is linear. The most general form of a derivative functional based at a point that we will consider is

$$\lambda f = \sum_{k=0}^r w_k f^{(k)}(x)$$

where x is a suitable point in the domain of f . We will construct a quasi-interpolant based on this kind of coefficient functionals in Section 8.6.1. By combining derivative functionals based at different points we obtain

$$\lambda f = \sum_{i=0}^{\ell} \sum_{k=0}^{r_i} w_{i,k} f^{(k)}(x_i)$$

where each r_i is a nonnegative integer. A typical functional of this kind is the divided difference of a function when some of the arguments are repeated. Such functionals are fundamental in interpolation with polynomials. Recall that if the same argument occurs $r+1$ times in a divided difference, this signifies that all derivatives of order $0, 1, \dots, r$ are to be interpolated at the point. Note that the point functionals above are derivative functionals with $r_i = 0$ for all i .

8.4.3 Integral functionals

The final kind of linear functionals that we will consider are based on integration. A typical functional of this kind is

$$\lambda f = \int_a^b f(x) \phi(x) dx \quad (8.3)$$

where ϕ is some fixed function. Because of basic properties of integration, it is easy to check that this is a linear functional. Just as with point functionals, we can combine several functionals like the one in (8.3) together,

$$\lambda f = w_0 \int_a^b f(x) \phi_0(x) dx + w_1 \int_a^b f(x) \phi_1(x) dx + \dots + w_\ell \int_a^b f(x) \phi_\ell(x) dx,$$

where $(w_i)_{i=0}^\ell$ are real numbers and $\{\phi_i\}_{i=0}^\ell$ are suitable functions. Note that the right-hand side of this equation can be written in the form (8.3) if we define ϕ by

$$\phi(x) = w_0 \phi_0(x) + w_1 \phi_1(x) + \dots + w_\ell \phi_\ell(x).$$

Point functionals can be considered a special case of integral functionals. For if ϕ_ϵ is a function that is positive on the interval $I_\epsilon = (x_i - \epsilon, x_i + \epsilon)$ and $\int_{I_\epsilon} \phi_\epsilon = 1$, then we know from the mean value theorem that

$$\int_{I_\epsilon} f(x) \phi_\epsilon(x) dx = f(\xi)$$

for some ξ in I_ϵ , as long as f is a nicely behaved (for example continuous) function. If we let ϵ tend to 0 we clearly have

$$\lim_{\epsilon \rightarrow 0} \int_{I_\epsilon} f(x) \phi_\epsilon(x) dx = f(x_i), \quad (8.4)$$

so by letting ϕ in (8.3) be a nonnegative function with small support around x and unit integral we can come as close to point interpolation as we wish.

If we include the condition that $\int_a^b \phi dx = 1$, then the natural interpretation of (8.3) is that λf gives a weighted average of the function f , with $\phi(x)$ giving the weight of the function value $f(x)$. A special case of this is when ϕ is the constant $\phi(x) = 1/(b-a)$; then λf is the traditional average of f . From this point of view the limit (8.4) is quite obvious: if we take the average of f over ever smaller intervals around x_i , the limit must be $f(x_i)$.

The functional $\int_a^b f(x) dx$ is often referred to as the *first moment* of f . As the name suggests there are more moments. The $i+1$ st moment of f is given by

$$\int_a^b f(x) x^i dx.$$

Moments of a function occur in many applications of mathematics like physics and the theory of probability.

8.4.4 Preservation of moments and interpolation of linear functionals

Interpolation of function values is a popular approximation method, and we have used it repeatedly in this book. However, is it a good way to approximate a given function f ? Is it not a bit haphazard to pick out a few, rather arbitrary, points on the graph of f and insist that our approximation should reproduce these points exactly and then ignore all other information about f ? As an example of what can happen, suppose that we are given a set of function values $(x_i, f(x_i))_{i=1}^m$ and that we use piecewise linear interpolation to approximate the underlying function. If f has been sampled densely and we interpolate all the values, we would expect the approximation to be good, but consider what happens if we interpolate only two of the values. In this case we cannot expect the resulting straight line to be a good approximation. If we are only allowed to reproduce two pieces of information about f we would generally do much better by reproducing its first two moments, i.e., the two integrals $\int f(x) dx$ and $\int f(x)x dx$, since this would ensure that the approximation would reproduce some of the *average* behaviour of f .

Reproduction of moments is quite easy to accomplish. If our approximation is g , we just have to ensure that the conditions

$$\int_a^b g(x) x^i dx = \int_a^b f(x) x^i dx, \quad i = 0, 1, \dots, n-1$$

are enforced if we want to reproduce n moments. In fact, this can be viewed as a generalisation of interpolation if we view interpolation to be preservation of the values of a set of linear functionals $(\rho_i)_{i=1}^n$,

$$\rho_i g = \rho_i f, \quad \text{for } i = 1, 2, \dots, n. \quad (8.5)$$

When $\rho_i f = \int_a^b f(x) x^{i-1} dx$ for $i = 1, \dots, n$ we preserve moments, while if $\rho_i f = f(x_i)$ for $i = 1, \dots, n$ we preserve function values. Suppose for example that g is required to lie in the linear space spanned by the basis $\{\psi_j\}_{j=1}^n$. Then we can determine coefficients $(c_j)_{j=1}^n$ so that $g(x) = \sum_{j=1}^n c_j \psi_j(x)$ satisfies the interpolation conditions (8.5) by inserting this expression for g into (8.5). By exploiting the linearity of the functionals, we end up with the n linear equations

$$c_1 \rho_i(\psi_1) + c_2 \rho_i(\psi_2) + \dots + c_n \rho_i(\psi_n) = \rho_i(f), \quad i = 1, \dots, n$$

in the n unknown coefficients $(c_i)_{i=1}^n$. In matrix-vector form this becomes

$$\begin{pmatrix} \rho_1(\psi_1) & \rho_1(\psi_2) & \dots & \rho_1(\psi_n) \\ \rho_2(\psi_1) & \rho_2(\psi_2) & \dots & \rho_2(\psi_n) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_n(\psi_1) & \rho_n(\psi_2) & \dots & \rho_n(\psi_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \rho_1(f) \\ \rho_2(f) \\ \vdots \\ \rho_n(f) \end{pmatrix}. \quad (8.6)$$

A fundamental property of interpolation by point functionals is that the only polynomial of degree d that interpolates the value 0 at $d+1$ points is the zero polynomial. This corresponds to the fact that when $\rho_i f = f(x_i)$ and $\psi_i(x) = x^i$ for $i = 0, \dots, d$, the matrix in (8.6) is nonsingular. Similarly, it turns out that the only polynomial of degree d whose $d+1$ first moments vanish is the zero polynomial, which corresponds to the fact that the matrix in (8.6) is nonsingular when $\rho_i f = \int_a^b f(x) x^i dx$ and $\psi_i(x) = x^i$ for $i = 0, \dots, d$.

If the equations (8.6) can be solved, each coefficient will be a linear combination of the entries on the right-hand side,

$$c_j = \lambda_j f = w_{j,1} \rho_1(f) + w_{j,2} \rho_2(f) + \dots + w_{j,n} \rho_n(f).$$

We recognise this as (8.2) when the ρ_i correspond to point functionals, whereas we have

$$\begin{aligned} c_j &= \lambda_j f = w_{j,1} \int_a^b f(x) dx + w_{j,2} \int_a^b f(x) x dx + \dots + w_{j,n} \int_a^b f(x) x^{n-1} dx \\ &= \int_a^b f(x) (w_{j,1} + w_{j,2} x + \dots + w_{j,n} x^{n-1}) dx \end{aligned}$$

when the ρ_i correspond to preservation of moments.

8.4.5 Least squares approximation

In the discussion of point functionals, we mentioned that least squares approximation leads to coefficients that are linear combinations of point functionals when the error is measured by summing up the squares of the errors at a given set of data points. This is naturally termed *discrete* least squares approximation. In *continuous* least squares approximation we minimise the integral of the square of the error. If the function to be approximated is f and the approximation g is required to lie in a linear space $\mathbb{S}$, we solve the minimisation problem

$$\min_{g \in \mathbb{S}} \int_a^b (f(x) - g(x))^2 dx.$$

If $\mathbb{S}$ is spanned by $(\psi_i)_{i=1}^n$, we can write g as $g = \sum_{i=1}^n c_i \psi$ and the minimisation problem becomes

$$\min_{(c_1, \dots, c_n) \in \mathbb{R}^n} \int_a^b \left(f(x) - \sum_{i=1}^n c_i \psi_i(x) \right)^2 dx.$$

To determine the minimum we differentiate with respect to each coefficient and set the derivatives to zero which leads to the so-called *normal equations*

$$\sum_{i=1}^n c_i \int_a^b \psi_i(x) \psi_j(x) dx = \int_a^b \psi_j(x) f(x) dx, \quad \text{for } j = 1, \dots, n.$$

If we use the notation above and introduce the linear functionals $\rho_i f = \int_a^b \psi_i(x) f(x)$ represented by the basis functions, we recognise this linear system as an instance of (8.6). In other words, least squares approximation is nothing but interpolation of the linear functionals represented by the basis functions. In particular, preservation of moments corresponds to least squares approximation by polynomials.

8.4.6 Computation of integral functionals

In our discussions involving integral functionals we have tacitly assumed that the values of integrals like $\int_a^b f(x) \psi(x) dx$ are readily available. This is certainly true if both f and ψ are polynomials, and it turns out that it is also true if both f and ψ are splines. However, if f is some general function, then the integral cannot usually be determined exactly, even when ψ_i is a polynomial. In such situations we have to resort to numerical integration methods. Numerical integration amounts to computing an approximation to an integral by evaluating the function to be integrated at certain points, multiplying the function values by suitable weights, and then adding up to obtain the approximate value of the integral,

$$\int_a^b f(x) dx \approx w_0 f(x_0) + w_1 f(x_1) + \dots + w_\ell f(x_\ell).$$

In other words, when it comes to practical implementation of integral functionals we have to resort to point functionals. In spite of this, integral functionals and continuous least squares approximation are such important concepts that it is well worth while to have an exact mathematical description. And it is important to remember that we do have exact formulas for the integrals of polynomials and splines.

8.5 Alternative ways to construct coefficient functionals

In Section 8.2 we constructed three quasi-interpolants by following the general procedure in Section 8.1. In this section we will deduce two alternative ways to construct quasi-interpolants.

8.5.1 Computation via evaluation of linear functionals

Let us use the 3-point, quadratic quasi-interpolant in subsection 8.2.2 as an example. In this case we used $I = [\tau_{j+1}, \tau_{j+2}]$ as the local interval for determining $c_j = \lambda_j f$. This meant that the local spline space $\mathbb{S}_{2,\tau,I}$ become the space of quadratic polynomials on I which has

dimension three. This space is spanned by the three B-splines $\{B_i\}_{i=j-1}^{j+1}$ and interpolation at the three points

$$\tau_{j+1}, \quad \tau_{j+3/2} = \frac{\tau_{j+1} + \tau_{j+2}}{2}, \quad \tau_{j+2}$$

allowed us to determine a local interpolant $g^I = \sum_{i=j-1}^{j+1} b_i B_i$ whose middle coefficient b_j we used as $\lambda_j f$.

An alternative way to do this is as follows. Since g^I is constructed by interpolation at the three points τ_{j+1} , $\tau_{j+3/2}$ and τ_{j+2} , we know that $\lambda_j f$ can be written in the form

$$\lambda_j f = w_1 f(\tau_{j+1}) + w_2 f(\tau_{j+3/2}) + w_3 f(\tau_{j+2}). \quad (8.7)$$

We want to reproduce the local spline space which in this case is just the space of quadratic polynomials. This means that (8.7) should be valid for all quadratic polynomials. Reproduction of quadratic polynomials can be accomplished by demanding that (8.7) should be exact when f is replaced by the three elements of a basis for $\mathbb{S}_{2,\tau,I}$. The natural basis to use in our situation is the B-spline basis $\{B_i\}_{i=j-1}^{j+1}$. Inserting this, we obtain the system

$$\begin{aligned} \lambda_j B_{j-1} &= w_1 B_{j-1}(\tau_{j+1}) + w_2 B_{j-1}(\tau_{j+3/2}) + w_3 B_{j-1}(\tau_{j+2}), \\ \lambda_j B_j &= w_1 B_j(\tau_{j+1}) + w_2 B_j(\tau_{j+3/2}) + w_3 B_j(\tau_{j+2}), \\ \lambda_j B_{j+1} &= w_1 B_{j+1}(\tau_{j+1}) + w_2 B_{j+1}(\tau_{j+3/2}) + w_3 B_{j+1}(\tau_{j+2}). \end{aligned}$$

in the three unknowns w_1 , w_2 and w_3 . The left-hand sides of these equations are easy to determine. Since $\lambda_j f$ denotes the j th B-spline coefficient, it is clear that $\lambda_j B_i = \delta_{i,j}$, i.e., it is 1 when $i = j$ and 0 otherwise.

To determine the right-hand sides we have to compute the values of the B-splines. For this it is useful to note that the w_j 's in equation (8.7) cannot involve any of the knots other than t_{j+1} and t_{j+2} since a general polynomial knows nothing about these knots. This means that we can choose the other knots so as to make life simple for ourselves. The easiest option is to choose the first three knots equal to t_{j+1} and the last three equal to t_{j+2} . But then we are in the Bézier setting, and we know that the B-splines in this case will have the same values if we choose $\tau_{j+1} = 0$ and $\tau_{j+2} = 1$. The knots are then $(0, 0, 0, 1, 1, 1)$ which means that $\tau_{j+3/2} = 1/2$. If we denote the B-splines on these knots by $\{\tilde{B}_i\}_{i=1}^3$, we can replace B_i in (8.5.1) by $\tilde{B}_{i-j+2}$ for $i = 1, 2, 3$. We can now simplify (8.5.1) to

$$\begin{aligned} 0 &= w_1 \tilde{B}_1(0) + w_2 \tilde{B}_1(1/2) + w_3 \tilde{B}_1(1), \\ 1 &= w_1 \tilde{B}_2(0) + w_2 \tilde{B}_2(1/2) + w_3 \tilde{B}_2(1), \\ 0 &= w_1 \tilde{B}_3(0) + w_2 \tilde{B}_3(1/2) + w_3 \tilde{B}_3(1). \end{aligned}$$

If we insert the values of the B-splines we end up with the system

$$\begin{aligned} w_1 + w_2/4 &= 0, \\ w_2/2 &= 1, \\ w_2/4 + w_3 &= 0, \end{aligned}$$

which has the solution $w_1 = -1/2$, $w_2 = 2$ and $w_3 = -1/2$. In conclusion we have

$$\lambda_j f = \frac{-f(t_{j+1}) + 4f(t_{j+3/2}) - f(t_{j+2})}{2},$$

as we found in Section 8.2.2.

This approach to determining the linear functional works quite generally and is often the easiest way to compute the weights (w_i).

8.5.2 Computation via explicit representation of the local approximation

There is a third way to determine the expression for $\lambda_j f$. For this we write down an explicit expression for the approximation g^I . Using the 3-point quadratic quasi-interpolant as our example again, we introduce the abbreviations $a = \tau_{j+1}$, $b = \tau_{j+3/2}$ and $c = \tau_{j+2}$. We can write the local interpolant g^I as

$$g^I(x) = \frac{(x-b)(x-c)}{(a-b)(a-c)}f(a) + \frac{(x-a)(x-c)}{(b-a)(b-c)}f(b) + \frac{(x-a)(x-b)}{(c-a)(c-b)}f(c),$$

as it is easily verified that g^I then satisfies the three interpolation conditions $g^I(a) = f(a)$, $g^I(b) = f(b)$ and $g^I(c) = f(c)$. What remains is to write this in terms of the B-spline basis $\{B_i\}_{i=j-1}^{j+1}$ and pick out coefficient number j . Recall that we have the notation $\gamma_j(f)$ for the j th B-spline coefficient of a spline f . Coefficient number j on the left-hand side is $\lambda_j f$. On the right, we find the B-spline coefficients of each of the three polynomials and add up. The numerator of the first polynomial is $(x-b)(x-c) = x^2 - (b+c)x + bc$. To find the j th B-spline of this polynomial, we make use of Corollary 3.6 which tells that, when $d = 2$, we have $\gamma_j(x^2) = \tau_{j+1}\tau_{j+2} = ac$ and $\gamma_j(x) = (\tau_{j+1} + \tau_{j+2})/2 = (a+c)/2 = b$, respectively. The j th B-spline coefficient of the first polynomial is therefore

$$\gamma_j\left(\frac{ac - (b+c)b + bc}{(a-b)(a-c)}\right) = \frac{ac - b^2}{(a-b)(a-c)} \quad (8.8)$$

which simplifies to $-1/2$ since $b = (a+c)/2$. Similarly, we find that the j th B-spline coefficient of the second and third polynomials are 2 and $-1/2$, respectively. The complete j th B-spline coefficient of the right-hand side of (8.8) is therefore $-f(a)/2 + 2f(b) - f(c)/2$. In total, we have therefore obtained

$$\lambda_j f = \gamma_j(g^I) = -\frac{f(\tau_{j+1})}{2} + 2f(\tau_{j+3/2}) - \frac{f(\tau_{j+2})}{2},$$

as required.

This general procedure also works generally, and we will see another example of it in Section 8.6.1.

8.6 Two quasi-interpolants based on point functionals

In this section we consider two particular quasi-interpolants that can be constructed for any polynomial degree. They may be useful for practical approximation problems, but we are going to use them to prove special properties of spline functions in Chapters 9 and 10. Both quasi-interpolants are based on point functionals: In the first case all the points are identical which leads to derivative functionals, in the second case all the points are distinct.

8.6.1 A quasi-interpolant based on the Taylor polynomial

A very simple local, polynomial approximation is the Taylor polynomial. This leads to a quasi-interpolant based on derivative functionals. Even though we use splines of degree d , our local approximation can be of lower degree; in Theorem 8.5 this degree is given by r .

Theorem 8.5 (de Boor-Fix). *Let r be an integer with $0 \leq r \leq d$ and let x_j be a number in $[\tau_j, \tau_{j+d+1}]$ for $j = 1, \dots, n$. Consider the quasi-interpolant*

$$Q_{d,r}f = \sum_{j=1}^n \lambda_j(f) B_{j,d}, \quad \text{where} \quad \lambda_j(f) = \frac{1}{d!} \sum_{k=0}^r (-1)^k D^{d-k} \rho_{j,d}(x_j) D^k f(x_j), \quad (8.9)$$

and $\rho_{j,d}(y) = (y - \tau_{j+1}) \cdots (y - \tau_{j+d})$. Then $Q_{d,r}$ reproduces all polynomials of degree r and $Q_{d,d}$ reproduces all splines in $\mathbb{S}_{d,\tau}$.

Proof. To construct $Q_{d,r}$ we let I be the knot interval that contains x_j and let the local approximation $g^I = P_r^I f$ be the Taylor polynomial of degree r at the point x_j ,

$$g^I(x) = P_r^I f(x) = \sum_{k=0}^r \frac{(x - x_j)^k}{k!} D^k f(x_j).$$

To construct the linear functional $\lambda_j f$, we have to find the B-spline coefficients of this polynomial. We use the same approach as in Section 8.5.2. For this Marsden's identity,

$$(y - x)^d = \sum_{j=1}^n \rho_{j,d}(y) B_{j,d}(x),$$

will be useful. Setting $y = x_j$, we see that the j th B-spline coefficient of $(x_j - x)^d$ is $\rho_{j,d}(x_j)$. Differentiating Marsden's identity $d - k$ times with respect to y , setting $y = x_j$ and rearranging, we obtain the j th B-spline coefficient of $(x - x_j)^k / k!$ as

$$\gamma_j((x - x_j)^k / k!) = (-1)^k D^{d-k} \rho_{j,d}(x_j) / d! \quad \text{for } k = 0, \dots, r.$$

Summing up, we find that

$$\lambda_j(f) = \frac{1}{d!} \sum_{k=0}^r (-1)^k D^{d-k} \rho_{j,d}(x_j) D^k f(x_j).$$

Since the Taylor polynomial of degree r reproduces polynomials of degree r , we know that the quasi-interpolant will do the same. If $r = d$, we reproduce polynomials of degree d which agrees with the local spline space $\mathbb{S}_{d,\tau,I}$ since I is a single knot interval. The quasi-interpolant therefore reproduces the whole spline space $\mathbb{S}_{d,\tau}$ in this case. ■

Example 8.6. We find

$$D^d \rho_{j,d}(y) / d! = 1, \quad D^{d-1} \rho_{j,d}(y) / d! = y - \tau_j^*, \quad \text{where} \quad \tau_j^* = \frac{\tau_{j+1} + \cdots + \tau_{j+d}}{d}. \quad (8.10)$$

For $r = 1$ and $x_j = \tau_j^*$ we therefore obtain

$$Q_{d,r}f = \sum_{j=1}^n f(\tau_j^*) B_{j,d}$$

which is the Variation Diminishing spline approximation. For $d = r = 2$ we obtain

$$Q_{2,2}f = \sum_{j=1}^n [f(x_j) - (x_j - \tau_{j+3/2})Df(x_j) + \frac{1}{2}(x_j - \tau_{j+1})(x_j - \tau_{j+2})D^2f(x_j)] B_{j,2}. \quad (8.11)$$

while for $d = r = 3$ and $x_j = \tau_{j+2}$ we obtain

$$Q_{3,3}f = \sum_{j=1}^n [f(\tau_{j+2}) + \frac{1}{3}(\tau_{j+3} - 2\tau_{j+2} + \tau_{j+1})Df(\tau_{j+2}) - \frac{1}{6}(\tau_{j+3} - \tau_{j+2})(\tau_{j+2} - \tau_{j+1})D^2f(\tau_{j+2})] B_{j,3}. \quad (8.12)$$

We leave the detailed derivation as a problem for the reader.

Since $Q_{d,d}f = f$ for all $f \in \mathbb{S}_{d,\tau}$ it follows that the coefficients of a spline $f = \sum_{j=1}^n c_j B_{j,d}$ can be written in the form

$$c_j = \frac{1}{d!} \sum_{k=0}^d (-1)^k D^{d-k} \rho_{j,d}(x_j) D^k f(x_j), \quad \text{for } j = 1, \dots, n, \quad (8.13)$$

where x_j is any number in $[\tau_j, \tau_{j+d+1}]$.

8.6.2 Quasi-interpolants based on evaluation

Another natural class of linear functionals is the one where each λ_j used to define Q is constructed by evaluating the data at $r + 1$ distinct points

$$\tau_j \leq x_{j,0} < x_{j,1} < \dots < x_{j,r} \leq \tau_{j+d+1} \quad (8.14)$$

located in the support $[\tau_j, \tau_{j+d+1}]$ of the B-spline $B_{j,d}$ for $j = 1, \dots, n$. We consider the quasi-interpolant

$$P_{d,r}f = \sum_{j=1}^n \lambda_{j,r}(f) B_{j,d}, \quad (8.15)$$

where

$$\lambda_{j,r}(f) = \sum_{k=0}^r w_{j,k} f(x_{j,k}). \quad (8.16)$$

From the preceding theory we know how to choose the constants $w_{j,k}$ so that $P_{d,r}f = f$ for all $f \in \pi_r$.

Theorem 8.7. Let $\mathbb{S}_{d,\tau}$ be a spline space with a $d + 1$ -regular knot vector $\tau = (\tau_i)_{i=1}^{n+d+1}$. Let $(x_{j,k})_{k=0}^r$ be $r + 1$ distinct points in $[\tau_j, \tau_{j+d+1}]$ for $j = 1, \dots, n$, and let $w_{j,k}$ be the j th B-spline coefficient of the polynomial

$$p_{j,k}(x) = \prod_{\substack{s=0 \\ s \neq k}}^r \frac{x - x_{j,s}}{x_{j,k} - x_{j,s}}.$$

Then $P_{d,r}f = f$ for all $f \in \pi_r$, and if $r = d$ and all the numbers $(x_{j,k})_{k=0}^r$ lie in one subinterval

$$\tau_j \leq \tau_{\ell_j} \leq x_{j,0} < x_{j,1} < \dots < x_{j,r} \leq \tau_{\ell_j+1} \leq \tau_{j+d+1} \quad (8.17)$$

then $P_{d,d}f = f$ for all $f \in \mathbb{S}_{d,\tau}$.

Proof. It is not hard to see that

$$p_{j,k}(x_{j,i}) = \delta_{k,i}, \quad k, i = 0, \dots, r$$

so that the polynomial

$$P_{d,r}^I f(x) = \sum_{k=0}^r p_{j,k}(x) f(x_{j,k})$$

satisfies the interpolation conditions $P_{d,r}^I f(x_{j,r}) = f(x_{j,r})$ for all j and r . The result therefore follows from the general recipe. ■

In order to give examples of quasi-interpolants based on evaluation we need to know the B-spline coefficients of the polynomials $p_{j,k}$. We will return to this in more detail in Chapter 9, see (9.15) in the case $r = d$. A similar formula can be given for $r < d$.

Example 8.8. For $r = 1$ we have

$$p_{j,0}(x) = \frac{x_{j,1} - x}{x_{j,1} - x_{j,0}}, \quad p_{j,1}(x) = \frac{x - x_{j,0}}{x_{j,1} - x_{j,0}}$$

and (8.15) takes the form

$$P_{d,1}f = \sum_{j=1}^n \left[\frac{x_{j,1} - \tau_j^*}{x_{j,1} - x_{j,0}} f(x_{j,0}) + \frac{\tau_j^* - x_{j,0}}{x_{j,1} - x_{j,0}} f(x_{j,1}) \right] B_{j,d}. \quad (8.18)$$

This quasi-interpolant reproduces straight lines for any choice of $\tau_j \leq x_{j,0} < x_{j,1} \leq \tau_{j+d+1}$. If we choose $x_{j,0} = \tau_j^*$ the method simplifies to

$$\tilde{P}_{d,1}f = \sum_{j=1}^n f(\tau_j^*) B_{j,d}. \quad (8.19)$$

This is again the *Variation diminishing method of Schoenberg*.

8.7 Exercises

8.1 In this exercise we assume that the points $(x_{i,k})$ and the spline space $\mathbb{S}_{d,\tau}$ are as in Theorem 8.7.

a) Show that for $r = d = 2$

$$\begin{aligned} P_{2,2}f = \sum_{j=1}^n & \left[\frac{(\tau_{j+1} - x_{j,1})(\tau_{j+2} - x_{j,2}) + (\tau_{j+2} - x_{j,1})(\tau_{j+1} - x_{j,2})}{2(x_{j,0} - x_{j,1})(x_{j,0} - x_{j,2})} f(x_{j,0}) \right. \\ & + \frac{(\tau_{j+1} - x_{j,0})(\tau_{j+2} - x_{j,2}) + (\tau_{j+2} - x_{j,0})(\tau_{j+1} - x_{j,2})}{2(x_{j,1} - x_{j,0})(x_{j,1} - x_{j,2})} f(x_{j,1}) \\ & \left. + \frac{(\tau_{j+1} - x_{j,0})(\tau_{j+2} - x_{j,1}) + (\tau_{j+2} - x_{j,0})(\tau_{j+1} - x_{j,1})}{2(x_{j,2} - x_{j,0})(x_{j,2} - x_{j,1})} f(x_{j,2}) \right] B_{j,2} \end{aligned} \quad (8.20)$$

b) Show that (8.20) reduces to the operator (9.6) for a suitable choice of $(x_{j,k})_{k=0}^2$.

8.2 Derive the following operators $Q_{d,l}$ and show that they are exact for π_r for the indicated r . Again we the points $(x_{j,k})$ and the spline space $\mathbb{S}_{d,\tau}$ are in Theorem 8.7. Which of the operators reproduce the whole spline space?

a) $Q_{d,0}f = \sum_{j=1}^n f(x_j)B_{j,d}, \quad (r=0).$

b) $Q_{d,1}f = \sum_{j=1}^n [f(x_j) + (\tau_j - x_j)Df(x_j)]B_{j,d}, \quad (r=1).$

c) $\tilde{Q}_{d,1}f = \sum_{j=1}^n f(\tau_j^*)B_{j,d}, \quad (r=1).$

d)

$$Q_{2,2}f = \sum_{j=1}^n [f(x_j) - (x_j - \tau_{j+3/2})Df(x_j) + \frac{1}{2}(x_j - \tau_{j+1})(x_j - \tau_{j+2})D^2f(x_j)]B_{j,2}, \quad (r=2).$$

e) $\tilde{Q}_{2,2}f = \sum_{j=1}^n [f(\tau_{j+3/2}) - \frac{1}{2}(\tau_{j+2} - \tau_{j+1})^2D^2f(\tau_{j+3/2})]B_{j,2}, \quad (r=2).$

f)

$$Q_{3,3}f = \sum_{j=1}^n [f(\tau_{j+2}) + \frac{1}{3}(\tau_{j+3} - 2\tau_{j+2} + \tau_{j+1})Df(\tau_{j+2}) - \frac{1}{6}(\tau_{j+3} - \tau_{j+2})(\tau_{j+2} - \tau_{j+1})D^2f(\tau_{j+2})]B_{j,3}, \quad (r=3).$$

Approximation theory and stability

Polynomials of degree d have $d + 1$ degrees of freedom, namely the $d + 1$ coefficients relative to some polynomial basis. It turns out that each of these degrees of freedom can be utilised to gain approximation power so that the possible rate of approximation by polynomials of degree d is h^{d+1} , see Section 9.1. The meaning of this is that when a smooth function is approximated by a polynomial of degree d on an interval of length h , the error is bounded by Ch^{d+1} , where C is a constant that is independent of h . The exponent $d + 1$ therefore controls how fast the error tends to zero with h .

When several polynomials are linked smoothly together to form a spline, each polynomial piece has $d + 1$ coefficients, but some of these are tied up in satisfying the smoothness conditions. It therefore comes as a nice surprise that the approximation power of splines of degree d is the same as for polynomials, namely h^{d+1} , where h is now the largest distance between two adjacent knots. In passing from polynomials to splines we have therefore gained flexibility without sacrificing approximation power. We prove this in Section 9.2, by making use of some of the simple quasi-interpolants that we constructed in Chapter 8; it turns out that these produce spline approximations with the required accuracy.

The quasi-interpolants also allow us to establish two important properties of B-splines. The first is that B-splines form a stable basis for splines, see Section 9.3. This means that small perturbations of the B-spline coefficients can only lead to small perturbations in the spline, which is of fundamental importance for numerical computations. An important consequence of the stability of the B-spline basis is that the control polygon of a spline converges to the spline as the knot spacing tends to zero; this is proved in Section 9.4.

9.1 The distance to polynomials

We start by determining how well a given real valued function f defined on an interval $[a, b]$ can be approximated by a polynomial of degree d . We measure the error in the approximation with the uniform norm which for a bounded function g defined on an interval $[a, b]$ is defined by

$$\|g\|_{\infty, [a, b]} = \sup_{a \leq x \leq b} |g(x)|.$$

Whenever we have an approximation p to f we can use the norm and measure the error by $\|f - p\|_{\infty, [a, b]}$. There are many possible approximations to f by polynomials of degree d , and the approximation that makes the error as small as possible is of course of special interest. This approximation is referred to as the *best approximation* and the corresponding error is referred to as the *distance* from f to the space π_d of polynomials of degree $\leq d$. This is defined formally as

$$\text{dist}_{\infty, [a, b]}(f, \pi_d) = \inf_{p \in \pi_d} \|f - p\|_{\infty, [a, b]}.$$

In order to bound this approximation error, we have to place some restrictions on the functions that we approximate, and we will only consider functions with piecewise continuous derivatives. Such functions lie in a space that we denote $\mathbb{C}_{\Delta}^k[a, b]$ for some integer $k \geq 0$. A function f lies in this space if it has $k - 1$ continuous derivatives on the interval $[a, b]$, and the k th derivative $D^k f$ is continuous everywhere except for a finite number of points in the interior (a, b) , given by $\Delta = (z_j)$. At the points of discontinuity Δ the limits from the left and right, given by $D^k f(z_j+)$ and $D^k f(z_j-)$, should exist so all the jumps are finite. If there are no continuous derivatives we write $\mathbb{C}_{\Delta}[a, b] = \mathbb{C}_{\Delta}^0[a, b]$. Note that we will often refer to these spaces without stating explicitly what the singularities Δ are.

It is quite simple to give an upper bound for the distance of f to polynomials of degree d by choosing a particular approximation, namely Taylor expansion.

Theorem 9.1. *Given a polynomial degree d and a function f in $\mathbb{C}_{\Delta}^{d+1}[a, b]$, then*

$$\text{dist}_{\infty, [a, b]}(f, \pi_d) \leq C_d h^{d+1} \|D^{d+1} f\|_{\infty, [a, b]},$$

where $h = b - a$ and the constant C_d only depends on d ,

$$C_d = \frac{1}{2^{d+1}(d+1)!}.$$

Proof. Consider the truncated Taylor series of f at the midpoint $m = (a + b)/2$ of $[a, b]$,

$$T_d f(x) = \sum_{k=0}^d \frac{(x-m)^k}{k!} D^k f(m), \quad \text{for } x \in [a, b].$$

Since $T_d f$ is a polynomial of degree d we clearly have

$$\text{dist}_{\infty, [a, b]}(f, \pi_d) \leq \|f - T_d f\|_{\infty, [a, b]}. \quad (9.1)$$

The error is given by the integral form of the remainder in the Taylor expansion,

$$f(x) - T_d f(x) = \frac{1}{d!} \int_m^x (x-y)^d D^{d+1} f(y) dy,$$

which is valid for any $x \in [a, b]$. If we restrict x to the interval $[m, b]$ we obtain

$$|f(x) - T_d f(x)| \leq \|D^{d+1} f\|_{\infty, [a, b]} \frac{1}{d!} \int_m^x (x-y)^d dy.$$

The integral is given by

$$\frac{1}{d!} \int_m^x (x-y)^d dy = \frac{1}{(d+1)!} (x-m)^{d+1} \leq \frac{1}{(d+1)!} \left(\frac{h}{2}\right)^{d+1},$$

so for $x \geq m$ we have

$$|f(x) - T_d f(x)| \leq \frac{1}{2^{d+1}(d+1)!} h^{d+1} \|D^{d+1} f\|_{\infty, [a, b]}.$$

By symmetry this estimate must also hold for $x \leq m$. Combining the estimate with (9.1) completes the proof. ■

It is in fact possible to compute the best possible constant C_d . It turns out that for each $f \in \mathbb{C}^{d+1}[a, b]$ there is a point $\xi \in [a, b]$ such that

$$\text{dist}_{\infty, [a, b]}(f, \pi_d) = \frac{2}{4^{d+1}(d+1)!} h^{d+1} |D^{d+1} f(\xi)|$$

Applying this formula to the function $f(x) = x^{d+1}$ we see that the exponent $d+1$ in h^{d+1} is best possible.

9.2 The distance to splines

Just as we defined the distance from a function f to the space of polynomials of degree d we can define the distance from f to a spline space. Our aim is to show that on one knot interval, the distance from f to a spline space of degree d is essentially the same as the distance from f to the space of polynomials of degree d on a slightly larger interval, see Theorem 9.2 and Corollary 9.12. Our strategy is to consider the cases $d = 0, 1$ and 2 separately and then generalise to degree d . The main ingredient in the proof is to construct a simple but good approximation method that we can use in the same way that Taylor expansion was used in the polynomial case above. Some of the quasi-interpolants that we constructed in Chapter 8 will do this job very nicely.

We consider a spline space $\mathbb{S}_{d, \tau}$ where d is a nonnegative integer and $\tau = (\tau_i)_{i=1}^{n+d+1}$ is a $d+1$ regular knot vector and set

$$a = \tau_1, \quad b = \tau_{n+d+1}, \quad h_j = \tau_{j+1} - \tau_j, \quad h = \max_{1 \leq j \leq n} h_j.$$

Given a function f we consider the distance from f to $\mathbb{S}_{d, \tau}$ defined by

$$\text{dist}_{\infty, [a, b]}(f, \mathbb{S}_{d, \tau}) = \inf_{g \in \mathbb{S}_{d, \tau}} \|f - g\|_{\infty, [a, b]}.$$

We want to show the following.

Theorem 9.2. *Let the polynomial degree d and the function f in $\mathbb{C}_{\Delta}^{d+1}[a, b]$ be given. The distance between f and the spline space $\mathbb{S}_{d, \tau}$ is bounded by*

$$\text{dist}_{\infty, [a, b]}(f, \mathbb{S}_{d, \tau}) \leq D_d h^{d+1} \|D^{d+1} f\|_{\infty, [a, b]}, \quad (9.2)$$

where the constant D_d depends on d , but not on f or τ .

We will prove this theorem by constructing a spline $P_d f$ such that

$$|f(x) - P_d f(x)| \leq D_d h^{d+1} \|D^{d+1} f\|_{\infty, [a, b]}, \quad x \in [a, b] \quad (9.3)$$

for a constant D_d that depends only on d . The approximation $P_d f$ will be a quasi-interpolant on the form

$$P_d f = \sum_{i=1}^n \lambda_i(f) B_{i,d}$$

where λ_i is a rule for computing the i th B-spline coefficient. We will restrict ourselves to rules λ_i like

$$\lambda_i(f) = \sum_{k=0}^d w_{i,k} f(x_{i,k})$$

where the points $(x_{i,k})_{k=0}^d$ all lie in one knot interval and $(w_{i,k})_{k=0}^d$ are suitable coefficients.

9.2.1 The constant and linear cases

We first prove Theorem 9.2 in the simplest cases $d = 0$ and $d = 1$. For $d = 0$ the knots form a partition $a = \tau_1 < \dots < \tau_{n+1} = b$ of $[a, b]$ and the B-spline $B_{i,0}$ is the characteristic function of the interval $[\tau_i, \tau_{i+1})$ for $i = 1, \dots, n-1$, while $B_{n,0}$ is the characteristic function of the closed interval $[\tau_n, \tau_{n+1}]$. We consider the step function

$$g(x) = P_0 f(x) = \sum_{i=1}^n f(\tau_{i+1/2}) B_{i,0}(x), \quad (9.4)$$

where $\tau_{i+1/2} = (\tau_i + \tau_{i+1})/2$. Fix $x \in [a, b]$ and let μ be an integer such that $\tau_\mu \leq x < \tau_{\mu+1}$. We then have

$$f(x) - P_0 f(x) = f(x) - f(\tau_{\mu+1/2}) = \int_{\tau_{\mu+1/2}}^x Df(y) dy$$

so

$$|f(x) - P_0 f(x)| \leq |x - \tau_{\mu+1/2}| \|Df\|_{\infty, [\tau_\mu, \tau_{\mu+1}]} \leq \frac{h}{2} \|Df\|_{\infty, [a, b]}.$$

In this way we obtain (9.2) with $D_0 = 1/2$.

In the linear case $d = 1$ we define $P_1 f$ to be the piecewise linear interpolant to f on τ defined by

$$g = P_1 f = \sum_{i=1}^n f(\tau_{i+1}) B_{i,1}. \quad (9.5)$$

Proposition 5.2 gives an estimate of the error in linear interpolation and by applying this result on each interval we obtain

$$\|f - P_1 f\|_{\infty, [a, b]} \leq \frac{h^2}{8} \|D^2 f\|_{\infty, [a, b]}$$

which is (9.2) with $D_1 = 1/8$.

9.2.2 The quadratic case

The quadratic case $d = 2$ is more involved. We shall approximate f by the quasi-interpolant P_2f that we constructed in Section 8.2.2 and then estimate the error. The relevant properties of P_2 are summarised in the following lemma.

Lemma 9.3. Suppose $\tau = (\tau_i)_{i=1}^{n+3}$ is a knot vector with $\tau_{i+3} > \tau_i$ for $i = 1, \dots, n$ and set $\tau_{i+3/2} = (\tau_{i+1} + \tau_{i+2})/2$. The operator

$$P_2f = \sum_{i=1}^n \lambda_i(f) B_{i,2,\tau} \quad \text{with} \quad \lambda_i(f) = -\frac{1}{2}f(\tau_{i+1}) + 2f(\tau_{i+3/2}) - \frac{1}{2}f(\tau_{i+2}) \quad (9.6)$$

is linear and satisfies $P_2f = f$ for all $f \in \mathbb{S}_{2,\tau}$.

Note that since the knot vector is 3-regular we have $\lambda_1(f) = f(\tau_2)$ and $\lambda_n(f) = f(\tau_{n+1})$. We also note that since P_2 reproduces all splines in $\mathbb{S}_{d,\tau}$ it certainly reproduces all quadratic polynomial. This fact that will be useful in the proof of Lemma 9.6.

Our aim is to show that (9.3) holds for $d = 2$ and we are going to do this by establishing a sequence of lemmas. The first lemma shows that $\lambda_i(f)$ can become at most 3 times as large as f , irrespective of what the knot vector is.

Lemma 9.4. Let $P_2(f)$ be as in (9.6). Then

$$|\lambda_i(f)| \leq 3\|f\|_{\infty, [\tau_{i+1}, \tau_{i+2}]}, \quad \text{for } i = 1, \dots, n. \quad (9.7)$$

Proof. Fix an integer i . Then

$$|\lambda_i(f)| = \left| -\frac{1}{2}f(\tau_{i+1}) + 2f(\tau_{i+3/2}) - \frac{1}{2}f(\tau_{i+2}) \right| \leq \left(\frac{1}{2} + 2 + \frac{1}{2} \right) \|f\|_{\infty, [\tau_{i+1}, \tau_{i+2}]}$$

from which the result follows. ■

Since the B-spline coefficients of P_2f are bounded it is easy to see that the spline P_2f is also bounded by the same constant.

Lemma 9.5. Select some interval $[\tau_\mu, \tau_{\mu+1}]$ of $[\tau_3, \tau_{n+1})$. On this interval the spline P_2f is bounded by

$$\|P_2f\|_{\infty, [\tau_\mu, \tau_{\mu+1}]} \leq 3\|f\|_{\infty, [\tau_{\mu-1}, \tau_{\mu+2}]}. \quad (9.8)$$

Proof. Fix $x \in [\tau_\mu, \tau_{\mu+1}]$. Since the B-splines are nonnegative and form a partition of unity we have

$$\begin{aligned} |P_2f(x)| &= \left| \sum_{i=\mu-2}^{\mu} \lambda_i(f) B_{i,2,\tau}(x) \right| \leq \max_{\mu-2 \leq i \leq \mu} |\lambda_i(f)| \\ &\leq 3 \max_{\mu-2 \leq i \leq \mu} \|f\|_{\infty, [\tau_{i+1}, \tau_{i+2}]} = 3\|f\|_{\infty, [\tau_{\mu-1}, \tau_{\mu+2}]}, \end{aligned}$$

where we used Lemma 9.4. This completes the proof. ■

The following lemma shows that on one knot interval the spline P_2f approximates f almost as well as the best quadratic polynomial over a slightly larger interval. The proof depends on a standard trick that we will also use in the general case.

Lemma 9.6. Let $[\tau_\mu, \tau_{\mu+1})$ be a subinterval of $[\tau_3, \tau_{n+1})$. On this interval the error $f - P_2 f$ is bounded by

$$\|f - P_2 f\|_{\infty, [\tau_\mu, \tau_{\mu+1}]} \leq 4 \operatorname{dist}_{\infty, [\tau_{\mu-1}, \tau_{\mu+2}]}(f, \pi_2). \quad (9.9)$$

Proof. Let $p \in \pi_2$ be any quadratic polynomial. Since $P_2 p = p$ and P_2 is a linear operator, application of (9.8) to $f - p$ yields

$$\begin{aligned} |f(x) - (P_2 f)(x)| &= |f(x) - p(x) - ((P_2 f)(x) - p(x))| \\ &\leq |f(x) - p(x)| + |P_2(f - p)(x)| \\ &\leq (1 + 3)\|f - p\|_{\infty, [\tau_{\mu-1}, \tau_{\mu+2}]}. \end{aligned} \quad (9.10)$$

Since p is arbitrary we obtain (9.9). ■

Proof of Theorem 9.2 for $d = 2$. Theorem 9.1 with $d = 2$ states that

$$\operatorname{dist}_{\infty, [a, b]}(f, \pi_2) \leq C_2 h^3 \|D^3 f\|_{\infty, [a, b]},$$

where $h = b - a$ and $C_2 = 1/(2^3 3!)$. Specialising this estimate to the interval $[a, b] = [\tau_{\mu-1}, \tau_{\mu+2}]$ and combining with (9.9) we obtain (9.3) and hence (9.2) with $D_2 = 1/12$. ■

9.2.3 The general case

The general case is analogous to the quadratic case, but the details are more involved. The crucial part is to find a sufficiently good local approximation operator. The operator P_2 is a quasi interpolant that is based on local interpolation with quadratic polynomials at the three points $x_{i,k} = \tau_{i+1} + k(\tau_{i+2} - \tau_{i+1})/2$ for $k = 0, 1, 2$. Those points are located symmetrically in the middle subinterval of the support of the B-spline $B_{i,2}$.

We will follow the same strategy for general degree. The resulting quasi-interpolant will be a special case of the one given in Theorem 8.7. The challenge is to choose the local interpolation points in such a way that the B-spline coefficients of the approximation can be bounded independently of the knots, as in Lemma 9.4. The key is to let all the $d + 1$ points be uniformly distributed in the *largest* subinterval $[a_i, b_i] = [\tau_\mu, \tau_{\mu+1}]$ of $[\tau_{i+1}, \tau_{i+d}]$,

$$x_{i,k} = a_i + \frac{k}{d}(b_i - a_i), \quad \text{for } k = 0, 1, \dots, d. \quad (9.11)$$

Given $f \in \mathbb{C}_\Delta[a, b]$ we define $P_d f \in \mathbb{S}_{d,\tau}$ by

$$P_d f(x) = \sum_{i=1}^n \lambda_i(f) B_{i,d}(x), \quad \text{where } \lambda_i(f) = \sum_{k=0}^d w_{i,k} f(x_{i,k}). \quad (9.12)$$

In this situation Theorem 8.7 specialises to the following.

Lemma 9.7. Suppose that the functionals λ_i in (9.12) are given by $\lambda_i(f) = f(\tau_{i+1})$ if $\tau_{i+d} = \tau_{i+1}$, while if $\tau_{i+d} > \tau_{i+1}$ the coefficients of $\lambda_i(f)$ are given by

$$w_{i,k} = \gamma_i(p_{i,k}), \quad \text{for } k = 0, 1, \dots, d, \quad (9.13)$$

where $\gamma_i(p_{i,k})$ is the i th B-spline coefficient of the polynomial

$$p_{i,k}(x) = \prod_{\substack{j=0 \\ j \neq k}}^d \frac{x - x_{i,j}}{x_{i,k} - x_{i,j}}. \quad (9.14)$$

Then the operator P_d in (9.12) satisfies $P_d f = f$ for all $f \in \mathbb{S}_{d,\tau}$.

We really only need reproduction of polynomials of degree d , but since all the interpolation points lie in one knot interval we automatically get reproduction of all of $\mathbb{S}_{d,\tau}$.

The first challenge is to find a formula for the B-spline coefficients of $p_{i,k}$. Blossoming makes this easy.

Lemma 9.8. Suppose the spline space $\mathbb{S}_{d,\tau}$ is given together with the numbers $v_1, \dots, v_d$. The i th B-spline coefficient of the polynomial $p(x) = (x - v_1) \dots (x - v_d)$ can be written

$$\gamma_i(p) = \frac{1}{d!} \sum_{(j_1, \dots, j_d) \in \Pi_d} (\tau_{i+j_1} - v_1) \dots (\tau_{i+j_d} - v_d), \quad (9.15)$$

where Π_d is the set of all permutations of the integers $\{1, 2, \dots, d\}$.

Proof. By Theorem 4.16 we have

$$\gamma_i(p) = \mathcal{B}[p](\tau_{i+1}, \dots, \tau_{i+d}),$$

where $\mathcal{B}[p]$ is the blossom of p . It therefore suffices to verify that the expression (9.15) satisfies the three properties of the blossom. This is simple and is left to the reader. ■

Let us consider the special case $d = 2$ as an example. The set of all permutations of $\{1, 2\}$ are $\Pi_2 = \{(1, 2), (2, 1)\}$ and therefore

$$\gamma_i((x - v_1)(x - v_2)) = \frac{1}{2} \left((\tau_{i+1} - v_1)(\tau_{i+2} - v_2) + (\tau_{i+2} - v_1)(\tau_{i+1} - v_2) \right).$$

The next and most difficult step is to obtain a bound for $\lambda_i(f)$.

Theorem 9.9. Let $P_d(f) = \sum_{i=1}^n \lambda_i(f) B_{i,d}$ be the operator in Lemma 9.7. Then

$$|\lambda_i(f)| \leq K_d \|f\|_{\infty, [\tau_{i+1}, \tau_{i+d}]}, \quad i = 1, \dots, n, \quad (9.16)$$

where

$$K_d = \frac{2^d}{d!} (d(d-1))^d \quad (9.17)$$

depends only on d .

Proof. Fix an integer i . We may as well assume that $\tau_{i+1} < \tau_{i+d}$ since otherwise the result is obvious. From Lemma 9.8 we have

$$w_{i,k} = \sum_{(j_1, \dots, j_d) \in \Pi_d} \prod_{r=1}^d \left(\frac{\tau_{i+j_r} - v_r}{x_{i,k} - v_r} \right) / d!, \quad (9.18)$$

where $(v_r)_{r=1}^d = (x_{i,0}, \dots, x_{i,k-1}, x_{i,k+1}, \dots, x_{i,d})$, and Π_d denotes the set of all permutations of the integers $\{1, 2, \dots, d\}$. Since the numbers τ_{i+j_r} and v_r belongs to the interval $[\tau_{i+1}, \tau_{i+d}]$ for all r we have the inequality

$$\prod_{r=1}^d (\tau_{i+j_r} - v_r) \leq (\tau_{i+d} - \tau_{i+1})^d. \quad (9.19)$$

We also note that $x_{i,k} - v_r = (k - q)(b_i - a_i)/d$ for some q in the range $1 \leq q \leq d$ but with $q \neq k$. Taking the product over all r we therefore obtain

$$\begin{aligned} \prod_{r=1}^d |x_{i,k} - v_r| &= \prod_{\substack{q=0 \\ q \neq k}}^d \frac{|k - q|}{d} (b_i - a_i) \\ &= k!(d - k)! \left(\frac{b_i - a_i}{d} \right)^d \geq k!(d - k)! \left(\frac{\tau_{i+d} - \tau_{i+1}}{d(d - 1)} \right)^d \end{aligned} \quad (9.20)$$

for all values of k and r since $[a_i, b_i]$ is the largest subinterval of $[\tau_{i+1}, \tau_{i+d}]$. The sum in (9.18) contains $d!$ terms which means that

$$\sum_{k=0}^d |w_{i,k}| \leq \frac{[d(d - 1)]^d}{d!} \sum_{k=0}^d \binom{d}{k} = \frac{2^d}{d!} [d(d - 1)]^d = K_d$$

and therefore

$$|\lambda_i(f)| \leq \|f\|_{\infty, [\tau_{i+1}, \tau_{i+d}]} \sum_{k=0}^d |w_{i,k}| \leq K_d \|f\|_{\infty, [\tau_{i+1}, \tau_{i+d}]} \quad (9.21)$$

which is the required inequality. ■

Theorem 9.9 is the central ingredient in the proof of Theorem 9.2, but it has many other consequences as well, some of which we will consider later in this chapter. In fact Theorem 9.9 gives one of the key properties of B-splines. If $f = \sum_{i=1}^n c_i B_{i,d,\tau}$ is a spline in $\mathbb{S}_{d,\tau}$ we know that $\lambda_i(f) = c_i$. The inequality (9.16) therefore states that a B-spline coefficient is at most K_d times larger than the spline it represents, where the constant K_d is *independent* of the knots. A similar conclusion holds for $d \leq 2$, see Lemma 9.4 and the definition of P_0 and P_1 in (9.4) and (9.5). For later reference we record this in a corollary.

Corollary 9.10. *For any spline $f = \sum_{i=1}^n c_i B_{i,d}$ in $\mathbb{S}_{d,\tau}$ the size of the B-spline coefficients is bounded by*

$$|c_i| \leq K_d \|f\|_{\infty, [\tau_{i+1}, \tau_{i+d}]},$$

where the the constant K_d depends only on d .

From the bound on $\lambda_i(f)$ we easily obtain a similar bound for the norm of $P_d f$.

Theorem 9.11. *Let f be a function in the space $C_\Delta[a, b]$. On any subinterval $(\tau_\mu, \tau_{\mu+1})$ of (τ_{d+1}, τ_{n+1}) the approximation $P_d f$ is bounded by*

$$\|P_d f\|_{\infty, [\tau_\mu, \tau_{\mu+1}]} \leq K_d \|f\|_{\infty, [\tau_{\mu-d+1}, \tau_{\mu+d}]}, \quad (9.22)$$

where K_d is the constant in Theorem 9.9.

Proof. Fix an x in some interval $[\tau_\mu, \tau_{\mu+1})$. Since the B-splines are nonnegative and form a partition of unity we have by Theorem 9.9

$$\begin{aligned} |P_d f(x)| &= \left| \sum_{i=\mu-d}^{\mu} \lambda_i(f) B_{i,d,\tau}(x) \right| \leq \max_{\mu-d \leq i \leq \mu} |\lambda_i(f)| \\ &\leq K_d \max_{\mu-d \leq i \leq \mu} \|f\|_{\infty, [\tau_{i+1}, \tau_{i+d}]} = K_d \|f\|_{\infty, [\tau_{\mu-d+1}, \tau_{\mu+d}]} \end{aligned}$$

This completes the proof. ■

The following corollary shows that $P_d f$ locally approximates f essentially as well as the best polynomial approximation of f of degree d .

Corollary 9.12. *On any subinterval $[\tau_\mu, \tau_{\mu+1})$ the error $f - P_d f$ is bounded by*

$$\|f - P_d f\|_{\infty, [\tau_\mu, \tau_{\mu+1}]} \leq (1 + K_d) \text{dist}_{\infty, [\tau_{\mu-d+1}, \tau_{\mu+d}]}(f, \pi_d), \quad (9.23)$$

where K_d is the constant in Theorem 9.9

Proof. We argue exactly as in the quadratic case. Let $p \in \pi_d$ be any polynomial in π_d . Since $P_d p = p$ and P_d is a linear operator we have

$$\begin{aligned} |f(x) - (P_d f)(x)| &= |f(x) - p(x) - ((P_d f)(x) - p(x))| \\ &\leq |f(x) - p(x)| + |P_d(f - p)(x)| \\ &\leq (1 + K_d) \|f - p\|_{\infty, [\tau_{\mu-d+1}, \tau_{\mu+d}]}. \end{aligned}$$

Since p is arbitrary we obtain (9.23). ■

Proof of Theorem 9.2 for general d . By Theorem 9.1 we have for any interval $[a, b]$

$$\text{dist}_{\infty, [a, b]}(f, \pi_d) \leq C_d h^{d+1} \|D^{d+1} f\|_{\infty, [a, b]},$$

where $h = b - a$ and C_d only depends on d . Combining this estimate on $[a, b] = [\tau_{\mu-d+1}, \tau_{\mu+d}]$ with (9.23) we obtain (9.3) and hence (9.2) with $D_d = (K_d + 1)C_d$. ■

We have accomplished our task of estimating the distance from a function in $\mathbb{C}_{\Delta}^{d+1}[a, b]$ to an arbitrary spline space $\mathbb{S}_{d,\tau}$. However, there are several unanswered questions. Perhaps the most obvious is whether the constant K_d is the best possible. A moment's thought will make you realise that it certainly is not. One reason is that we made use of some rather coarse estimates in the proof of Theorem 9.9. Another reason is that we may obtain better estimates by using a different approximation operator.

In fact, it is quite easy to find a better operator which is also a quasi-interpolant based on local interpolation. Instead of choosing the local interpolation points uniformly in the largest subinterval of $[\tau_{i+1}, \tau_{i+d}]$, we simply choose the points uniformly in $[\tau_{i+1}, \tau_{i+d}]$,

$$x_{i,k} = \tau_{i+1} + \frac{k}{d}(\tau_{i+d} - \tau_{i+1}), \quad \text{for } k = 0, 1, \dots, d.$$

It is easy to check that the bound (9.19) on the numerator still holds while the last estimate in the bound on the denominator (9.20) is now unnecessary so we have

$$\prod_{r=1}^d |x_{i,k} - v_r| = \prod_{\substack{q=0 \\ q \neq k}}^d \frac{|k-q|}{d} (\tau_{i+d} - \tau_{i+1}) = \frac{k!(d-k)!}{d^d} (\tau_{i+d} - \tau_{i+1})^d.$$

This gives a new constant

$$\tilde{K}_d = \frac{2^d d^d}{d!}.$$

Note that the new approximation operator will not reproduce the whole spline space for $d > 2$. This improved constant can therefore not be used in Corollary 9.10.

The constant can be improved further by choosing the interpolation points to be the extrema of the Chebyshev polynomial, adjusted to the interval $[\tau_{i+1}, \tau_{i+d}]$.

9.3 Stability of the B-spline basis

In order to compute with polynomials or splines we need to choose a basis to represent the functions. If a basis is to be suitable for computer manipulations it should be reasonably insensitive to round-off errors. In particular, functions with ‘small’ function values should have ‘small’ coefficients and vice versa. A basis with this property is said to be *well conditioned* or *stable* and the stability is measured by the *condition number* of the basis. In this section we will study the condition number of the B-spline basis.

9.3.1 A general definition of stability

The stability of a basis can be defined quite generally. Instead of considering polynomials we can consider a general linear vector space where we can measure the size of the elements through a norm; this is called a *normed linear space*.

Definition 9.13. Let $\mathbb{V}$ be a normed linear space. A basis (ϕ_j) for $\mathbb{V}$ is said to be *stable* with respect to a vector norm $\|\cdot\|$ if there are small positive constants C_1 and C_2 such that

$$C_1^{-1} \|(c_j)\| \leq \left\| \sum_j c_j \phi_j \right\| \leq C_2 \|(c_j)\|, \quad (9.24)$$

for all sets of coefficients $\mathbf{c} = (c_j)$. Let C_1^* and C_2^* denote the smallest possible values of C_1 and C_2 such that (9.24) holds. The condition number of the basis is then defined to be $\kappa = \kappa((\phi_i)_i) = C_1^* C_2^*$.

At the risk of confusion we have used the same symbol both for the norm in $\mathbb{V}$ and the vector norm of the coefficients. In our case $\mathbb{V}$ will be some spline space $\mathbb{S}_{d,t}$ and the basis (ϕ_j) will be the B-spline basis. The norms we will consider are the p -norms which are defined by

$$\|f\|_p = \|f\|_{p,[a,b]} = \left(\int_a^b |f(x)|^p dx \right)^{1/p} \quad \text{and} \quad \|\mathbf{c}\|_p = \left(\sum_j |c_j|^p \right)^{1/p}$$

where p is a real number in the range $1 \leq p < \infty$. Here f is a function on the interval $[a, b]$ and $\mathbf{c} = (c_j)$ is a real vector. For $p = \infty$ the norms are defined by

$$\|f\|_\infty = \|f\|_{\infty, [a, b]} = \max_{a \leq x \leq b} |f(x)| \quad \text{and} \quad \|\mathbf{c}\|_\infty = \|(c_j)\|_\infty = \max_j |c_j|,$$

In practice, the most important norms are the 1-, 2- and ∞ -norms.

In Definition 9.13 we require the constants C_1 and C_2 to be ‘small’, but how small is ‘small’? There is no unique answer to this question, but it is typically required that C_1 and C_2 should be independent of the dimension n of $\mathbb{V}$, or at least grow very slowly with n . Note that we always have $\kappa \geq 1$, and $\kappa = 1$ if and only if we have equality in both inequalities in (9.24).

A stable basis is desirable for many reasons, and the constant $\kappa = C_1 C_2$ crops up in many different contexts. The condition number κ does in fact act as a sort of derivative of the basis and gives a measure of how much an error in the coefficients is magnified in a function value.

Proposition 9.14. *Suppose (ϕ_j) is a stable basis for $\mathbb{V}$. If $f = \sum_j c_j \phi_j$ and $g = \sum_j b_j \phi_j$ are two elements in $\mathbb{V}$ with $f \neq 0$, then*

$$\frac{\|f - g\|}{\|f\|} \leq \kappa \frac{\|\mathbf{c} - \mathbf{b}\|}{\|\mathbf{c}\|}, \quad (9.25)$$

where κ is the condition number of the basis as in Definition 9.13.

Proof. From (9.24), we have the two inequalities $\|f - g\| \leq C_2 \|(c_j - b_j)\|$ and $1/\|f\| \leq C_1/\|(c_j)\|$. Multiplying these together gives the result. ■

If we think of g as an approximation to f then (9.25) says that the relative error in $f - g$ is bounded by at most κ times the relative error in the coefficients. If κ is small a small relative error in the coefficients gives a small relative error in the function values. This is important in floating point calculations on a computer. A function is usually represented by its coefficients relative to some basis. Normally, the coefficients are real numbers that must be represented inexactly as floating point numbers in the computer. This round-off error means that the computed spline, here g , will differ from the exact f . Proposition 9.14 shows that this is not so serious if the perturbed coefficients of g are close to those of f and the basis is stable.

Proposition 9.14 also provides some information as to what are acceptable values of C_1^* and C_2^* . If for example $\kappa = C_1^* C_2^* = 100$ we risk losing 2 decimal places in evaluation of a function; exactly how much accuracy one can afford to lose will of course vary.

One may wonder whether there are any unstable polynomial bases. It turns out that the power basis $1, x, x^2, \dots$, on the interval $[0, 1]$ is unstable even for quite low degrees. Already for degree 10, one risks losing as much as 4 or 5 decimal digits in the process of computing the value of a polynomial on the interval $[0, 1]$ relative to this basis, and other operations such as numerical root finding is even more sensitive.

9.3.2 Stability of the B-spline basis, $p = \infty$

Since splines and B-splines are defined via the knot vector, it is quite conceivable that the condition number of the B-spline basis could become arbitrarily large for certain knot configurations, for example in the limit when two knots merge into one. One of the key features of splines is that this cannot happen.

Theorem 9.15. *There is a constant K_d which depends only on the polynomial degree d , such that for all spline spaces $\mathbb{S}_{d,t}$ and all splines $f = \sum_{i=1}^n c_i B_{i,d} \in \mathbb{S}_{d,t}$ with B-spline coefficients $\mathbf{c} = (c_i)_{i=1}^n$, the two inequalities*

$$K_d^{-1} \|\mathbf{c}\|_\infty \leq \|f\|_{\infty, [t_1, t_{n+d}]} \leq \|\mathbf{c}\|_\infty \quad (9.26)$$

hold.

Proof. We have already proved variants of the second inequality several times; it follows since B-splines are nonnegative and sum to (at most) 1.

The first inequality is a consequence of Corollary 9.10. The value of the constant K_d is $K_0 = K_1 = 1$, $K_2 = 3$ while it is given by (9.17) for $d > 2$. ■

The condition number of the B-spline basis on the knot vector $\boldsymbol{\tau}$ with respect to the ∞ -norm is usually denoted $\kappa_{d,\infty,\boldsymbol{\tau}}$. By taking the supremum over all knot vectors we obtain the knot independent condition number $\kappa_{d,\infty}$,

$$\kappa_{d,\infty} = \sup_{\boldsymbol{\tau}} \kappa_{d,\infty,\boldsymbol{\tau}}.$$

Theorem 9.15 shows that $\kappa_{d,\infty}$ is bounded above by K_d .

Although K_d is independent of the knots, it grows quite quickly with d and seems to indicate that the B-spline basis may well be unstable for all but small values of d . However, by using different techniques it is possible to find better estimates for the condition number, and it is indeed known that the B-spline basis is very stable, at least for moderate values of d . It is simple to determine the condition number for $d \leq 2$; we have $\kappa_{0,\infty} = \kappa_{1,\infty} = 1$ and $\kappa_{2,\infty} = 3$. For $d \geq 3$ it has recently been shown that $\kappa_{d,\infty} = O(2^d)$. The first few values are known to be approximately $\kappa_{3,\infty} \approx 5.5680$ and $\kappa_{4,\infty} \approx 12.088$.

9.3.3 Stability of the B-spline basis, $p < \infty$

In this section we are going to generalise Theorem 9.15 to any p -norm. This is useful in some contexts, especially the case $p = 2$ which is closely related to least squares approximation. The proof uses standard tools from analysis, but may seem technical for the reader who is not familiar with the techniques.

Throughout this section p is a fixed real number in the interval $[1, \infty)$ and q is a related number defined by the identity $1/p + 1/q = 1$. A classical inequality for functions that will be useful is the Hölder inequality

$$\int_a^b |f(x)g(x)| dx \leq \|f\|_p \|g\|_q.$$

We will also need the Hölder inequality for vectors which is given by

$$\sum_{i=1}^n |b_i c_i| \leq \| (b_i)_{i=1}^n \|_p \| (c_i)_{i=1}^n \|_q.$$

In addition to the Hölder inequalities we need a fundamental inequality for polynomials. This states that for any polynomial $g \in \pi_d$ and any interval $[a, b]$ we have

$$|g(x)| \leq \frac{C}{b-a} \int_a^b |g(z)| dz, \quad \text{for any } x \in [a, b], \quad (9.27)$$

where the constant C only depends on the degree d . This is a consequence of the fact that all norms on a finite dimensional vector space are equivalent.

In order to generalise the stability result (9.26) to arbitrary p -norms we need to introduce a different scaling of the B-splines. We define the p -norm B-splines to be identically zero if $\tau_{i+d+1} = \tau_i$ and

$$B_{i,d,t}^p = \left(\frac{d+1}{\tau_{i+d+1} - \tau_i} \right)^{1/p} B_{i,d,t}, \quad (9.28)$$

otherwise. We can then state the p -norm stability result for B-splines.

Theorem 9.16. *There is a constant K that depends only on the polynomial degree d , such that for all $1 \leq p \leq \infty$, all spline spaces $\mathbb{S}_{d,t}$ and all splines $f = \sum_{i=1}^n c_i B_{i,d}^p \in \mathbb{S}_{d,t}$ with p -norm B-spline coefficients $\mathbf{c} = (c_i)_{i=1}^n$ the inequalities*

$$K^{-1} \|\mathbf{c}\|_p \leq \|f\|_{p, [\tau_1, \tau_{m+d}]} \leq \|\mathbf{c}\|_p \quad (9.29)$$

hold.

Proof. We first prove the upper inequality. Let $\gamma_i = (d+1)/(\tau_{i+d+1} - \tau_i)$ denote the p th power of the scaling factor in (9.28) for $i = 1, \dots, n$ and set $[a, b] = [\tau_1, \tau_{n+d+1}]$. Remembering the definition of $B_{i,d,t}^p$ and the identity $1/p + 1/q = 1$ and applying the Hölder inequality for sums we obtain

$$\sum_i |c_i B_{i,d}^p| = \sum_i |c_i \gamma_i^{1/p} B_{i,d}^{1/p} B_{i,d}^{1/q}| \leq \left(\sum_i |c_i|^p \gamma_i B_{i,d} \right)^{1/p} \left(\sum_i B_{i,d} \right)^{1/q}.$$

Raising both sides of this inequality to the p th power and recalling that B-splines sum to (at most) 1 we obtain the inequality

$$\left| \sum_i c_i B_{i,d}^p(x) \right|^p \leq \sum_i |c_i|^p \gamma_i B_{i,d}(x) \quad \text{for any } x \in \mathbb{R}. \quad (9.30)$$

It can be shown that the integral of a B-spline is given by

$$\int_{\tau_i}^{\tau_{i+d+1}} B_{i,d}(x) dx = \frac{\tau_{i+d+1} - \tau_i}{d+1} = \frac{1}{\gamma_i}.$$

Making use of this and (9.30) we find

$$\|f\|_{p, [a, b]}^p = \int_a^b \left| \sum_i c_i B_{i,d}^p(x) \right|^p dx \leq \sum_i |c_i|^p \gamma_i \int_a^b B_{i,d}(x) dx = \sum_i |c_i|^p.$$

Taking p th roots on both sides proves the upper inequality.

Consider now the lower inequality. The spline f is given as a linear combination of p -norm B-splines, but can very simply be written as a linear combination of the usual B-splines,

$$f = \sum_i c_i B_{i,d}^p = \sum_i c_i \gamma_i^{1/p} B_{i,d}.$$

From the first inequality in (9.26) we then obtain for each i

$$\left(\frac{d+1}{\tau_{i+d+1} - \tau_i} \right)^{1/p} |c_i| \leq K_d \max_{\tau_{i+1} \leq x \leq \tau_{i+d}} |f(x)|,$$

where the constant K_d only depends on d . Extending the maximum to a larger subinterval and applying the inequality (9.27) we find

$$\begin{aligned} |c_i| &\leq K_d (d+1)^{-1/p} (\tau_{i+d+1} - \tau_i)^{1/p} \max_{\tau_i \leq x \leq \tau_{i+d+1}} |f(x)| \\ &\leq CK_d (d+1)^{-1/p} (\tau_{i+d+1} - \tau_i)^{-1+1/p} \int_{\tau_i}^{\tau_{i+d+1}} |f(y)| dy. \end{aligned}$$

Next, we apply the Hölder inequality for integrals to the product $\int_{\tau_i}^{\tau_{i+d+1}} |f(y)| dy$ and obtain

$$|c_i| \leq CK_d (d+1)^{-1/p} \left(\int_{\tau_i}^{\tau_{i+d+1}} |f(y)|^p dy \right)^{1/p}.$$

Raising both sides to the p th power and summing over i we obtain

$$\sum_i |c_i|^p \leq C^p K_d^p (d+1)^{-1} \sum_i \int_{\tau_i}^{\tau_{i+d+1}} |f(y)|^p dy \leq C^p K_d^p \|f\|_{p,[a,b]}^p.$$

Taking p th roots we obtain the lower inequality in (9.29) with $K = CK_d$. ■

9.4 Convergence of the control polygon for spline functions

Recall that for a spline function $f(x) = \sum_i c_i B_{i,d,\tau}$ the control polygon is the piecewise linear interpolant to the points (τ_i^*, c_i) , where $\tau_i^* = (\tau_{i+1} + \dots + \tau_{i+d})/d$ is the i th knot average. In this section we are going to prove that the control polygon converges to the spline it represents when the knot spacing approaches zero. The main work is done in Lemma 9.17 which shows that a corner of the control polygon is close to the spline since c_i is close to $f(\tau_i^*)$, at least when the spacing in the knot vector is small. The proof of the lemma makes use of the fact that the size of a B-spline coefficient c_i can be bounded in terms of the size of the spline on the interval $[\tau_{i+1}, \tau_{i+d+1}]$, which we proved in Theorem 9.9 and Lemma 9.4 (and Section 9.2.1),

$$|c_i| \leq K_d \|f\|_{[\tau_{i+1}, \tau_{i+d+1}]}. \quad (9.31)$$

The norm used here and throughout this section is the ∞ -norm.

Lemma 9.17. *Let f be a spline in $\mathbb{S}_{d,\tau}$ with coefficients (c_i) . Then*

$$|c_i - f(\tau_i^*)| \leq K (\tau_{i+d} - \tau_{i+1})^2 \|D^2 f\|_{[\tau_{i+1}, \tau_{i+d+1}]}, \quad (9.32)$$

where $\tau_i^* = (\tau_{i+1} + \dots + \tau_{i+d})/d$, the operator D^2 denotes (one-sided) differentiation (from the right), and the constant K only depends on d .

Proof. Let i be fixed. If $\tau_{i+1} = \tau_{i+d}$ then we know from property 5 in Lemma 2.3 that $B_{i,d}(\tau_i^*) = 1$ so $c_i = f(\tau_i^*)$ and there is nothing to prove. Assume for the rest of the proof that the interval $J = (\tau_{i+1}, \tau_{i+d})$ is nonempty. Since J contains at most $d - 2$ knots, it follows from the continuity property of B-splines that f has at least two continuous derivatives in J . Let x_0 be a number in the interval J and consider the spline

$$g(x) = f(x) - f(x_0) - (x - x_0)Df(x_0)$$

which is the error in a first order Taylor expansion of f at x_0 . This spline lies in $\mathbb{S}_{d,\tau}$ and can therefore be written as $g = \sum_i b_i B_{i,d,\tau}$ for suitable coefficients (b_i) . More specifically we have

$$b_i = c_i - f(x_0) - (\tau_i^* - x_0)Df(x_0).$$

Choosing $x_0 = \tau_i^*$ we have $b_i = c_i - f(\tau_i^*)$ and according to the inequality (9.31) and the error term in first order Taylor expansion we find

$$|c_i - f(\tau_i^*)| = |b_i| \leq K_d \|g\|_J \leq \frac{K_d(\tau_{i+d} - \tau_{i+1})^2}{2} \|D^2 f\|_J.$$

The inequality (9.32) therefore holds with $K = K_d/2$ and the proof is complete. ■

Lemma 9.17 shows that the corners of the control polygon converge to the spline as the knot spacing goes to zero. This partly explains why the control polygon approaches the spline when we insert knots. What remains is to show that the control polygon as a whole also converges to the spline.

Theorem 9.18. Let $f = \sum_{i=1}^n c_i B_{i,d}$ be a spline in $\mathbb{S}_{d,\tau}$, and let $\Gamma_{d,\tau}(f)$ be its control polygon. Then

$$\|\Gamma_{d,\tau}(f) - f\|_{[\tau_1^*, \tau_n^*]} \leq K h^2 \|D^2 f\|_{[\tau_1, \tau_{n+d+1}]}, \quad (9.33)$$

where $h = \max_i \{\tau_{i+1} - \tau_i\}$ and the constant K only depends on d .

Proof. As usual, we assume that τ is $d + 1$ -regular (if not we extend it with $d + 1$ -tuple knots at either ends and add zero coefficients). Suppose that x is in $[\tau_1^*, \tau_m^*]$ and let j be such that $\tau_j^* \leq x < \tau_{j+1}^*$. Observe that since the interval $J^* = (\tau_j^*, \tau_{j+1}^*)$ is nonempty we have $\tau_{j+1} < \tau_{j+d+1}$ and J^* contains at most $d - 1$ knots. From the continuity property of B-splines we conclude that f has a continuous derivative and the second derivative of f is at least piecewise continuous in J^* . Let

$$g(x) = \frac{(\tau_{j+1}^* - x)f(\tau_j^*) + (x - \tau_j^*)f(\tau_{j+1}^*)}{\tau_{j+1}^* - \tau_j^*}$$

be the linear interpolant to f on this interval. We will show that both $\Gamma = \Gamma_{d,\tau}(f)$ and f are close to g on J^* and then deduce that Γ is close to f because of the triangle inequality

$$|\Gamma(x) - f(x)| \leq |\Gamma(x) - g(x)| + |g(x) - f(x)|. \quad (9.34)$$

Let us first consider the difference $\Gamma - g$. Note that

$$\Gamma(x) - g(x) = \frac{(\tau_{j+1}^* - x)(b_j - f(\tau_j^*)) + (x - \tau_j^*)(b_{j+1} - f(\tau_{j+1}^*))}{\tau_{j+1}^* - \tau_j^*}$$

for any x in J^* . We therefore have

$$|\Gamma(x) - g(x)| \leq \max\{|b_j - f(\tau_j^*)|, |b_{j+1} - f(\tau_{j+1}^*)|\},$$

for $x \in J^*$. From Lemma 9.17 we then conclude that

$$|\Gamma(x) - g(x)| \leq K_1 h^2 \|D^2 f\|_J, \quad x \in J^*, \quad (9.35)$$

where $J = [\tau_1, \tau_{m+d+1}]$ and K_1 is a constant that only depends on d .

The second difference $f(x) - g(x)$ in (9.34) is the error in linear interpolation to f at the endpoints of J^* . For this process we have the standard error estimate

$$|f(x) - g(x)| \leq \frac{1}{8} (\tau_{j+1}^* - \tau_j^*)^2 \|D^2 f\|_{J^*} \leq \frac{1}{8} h^2 \|D^2 f\|_J, \quad x \in J^*. \quad (9.36)$$

If we now combine (9.35) and (9.36) as indicated in (9.34), we obtain the Theorem with constant $K = K_1 + 1/8$. ■

Because of the factor h^2 in Theorem 9.18 we say (somewhat loosely) that the control polygon converges quadratically to the spline.

9.5 Exercises

9.1 In this exercise we will study the order of approximation by the Schoenberg Variation Diminishing Spline Approximation of degree $d \geq 2$. This approximation is given by

$$V_d f = \sum_{i=1}^n f(\tau_i^*) B_{i,d}, \quad \text{with} \quad \tau_i^* = \frac{\tau_{i+1} + \cdots + \tau_{i+d}}{d}.$$

Here $B_{i,d}$ is the i th B-spline of degree d on a $d+1$ -regular knot vector $\boldsymbol{\tau} = (\tau_i)_{i=1}^{n+d+1}$. We assume that $\tau_{i+d} > \tau_i$ for $i = 2, \dots, n$. Moreover we define the quantities

$$a = \tau_1, \quad b = \tau_{n+d+1}, \quad h = \max_{1 \leq i \leq n} \tau_{i+1} - \tau_i.$$

We want to show that $V_d f$ is an $O(h^2)$ approximation to a sufficiently smooth f .

We first consider the more general spline approximation

$$\tilde{V}_d f = \sum_{i=1}^n \lambda_i(f) B_{i,d}, \quad \text{with} \quad \lambda_i(f) = w_{i,0} f(x_{i,0}) + w_{i,1} f(x_{i,1}).$$

Here $x_{i,0}$ and $x_{i,1}$ are two distinct points in $[\tau_i, \tau_{i+d}]$ and $w_{i,0}, w_{i,1}$ are constants, $i = 1, \dots, n$.

Before attempting to solve this exercise the reader might find it helpful to review Section 9.2.2

- a) Suppose for $i = 1, \dots, n$ that $w_{i,0}$ and $w_{i,1}$ are such that

$$\begin{aligned} w_{i,0} + w_{i,1} &= 1 \\ x_{i,0} w_{i,0} + x_{i,1} w_{i,1} &= \tau_i^* \end{aligned}$$

Show that then $\tilde{V}_d p = p$ for all $p \in \pi_1$. (Hint: Consider the polynomials $p(x) = 1$ and $p(x) = x$.)

- b) Show that if we set $x_{i,0} = \tau_i^*$ for all i then $\tilde{V}_d f = V_d f$ for all f , regardless of how we choose the value of $x_{i,1}$.

In the rest of this exercise we set $\lambda_i(f) = f(\tau_i^*)$ for $i = 1, \dots, n$, i.e. we consider $V_d f$. We define the usual uniform norm on an interval $[c, d]$ by

$$\|f\|_{[c,d]} = \sup_{c \leq x \leq d} |f(x)|, \quad f \in C_\Delta[c, d].$$

- c) Show that for $d+1 \leq l \leq n$

$$\|V_d f\|_{[\tau_l, \tau_{l+1}]} \leq \|f\|_{[\tau_{l-d}^*, \tau_l^*]}, \quad f \in C_\Delta[a, b].$$

- d) Show that for $f \in C_\Delta[\tau_{l-d}^*, \tau_l^*]$ and $d+1 \leq l \leq n$

$$\|f - V_d f\|_{[\tau_l, \tau_{l+1}]} \leq 2 \operatorname{dist}_{[\tau_{l-d}^*, \tau_l^*]}(f, \pi_1).$$

- e) Explain why the following holds for $d+1 \leq l \leq n$

$$\operatorname{dist}_{[\tau_{l-d}^*, \tau_l^*]}(f, \pi_1) \leq \frac{(\tau_l^* - \tau_{l-d}^*)^2}{8} \|D^2 f\|_{[\tau_{l-d}^*, \tau_l^*]}.$$

- f) Show that the following $O(h^2)$ estimate holds

$$\|f - V_d f\|_{[a,b]} \leq \frac{d^2}{4} h^2 \|D^2 f\|_{[a,b]}.$$

(Hint: Verify that $\tau_l^* - \tau_{l-d}^* \leq hd$.)

9.2 In this exercise we want to perform a numerical simulation experiment to determine the order of approximation by the quadratic spline approximations

$$\begin{aligned} V_2 f &= \sum_{i=1}^n f(\tau_i^*) B_{i,2}, \quad \text{with} \quad \tau_i^* = \frac{\tau_{i+1} + \tau_{i+2}}{2}, \\ P_2 f &= \sum_{i=1}^n \left(-\frac{1}{2} f(\tau_{i+1}) + 2f(\tau_i^*) - \frac{1}{2} f(\tau_{i+2}) \right) B_{i,2}. \end{aligned}$$

We want to test the hypotheses $f - V_2 f = O(h^2)$ and $f - P_2 f = O(h^3)$ where $h = \max_i \tau_{i+1} - \tau_i$. We test these on the function $f(x) = \sin x$ on $[0, \pi]$ for various values of h . Consider

for $m \geq 0$ and $n_m = 2 + 2^m$ the 3-regular knot vector $\boldsymbol{\tau}^m = (\tau_i^m)_{i=1}^{n_m+3}$ on the interval $[0, \pi]$ with uniform spacing $h_m = \pi 2^{-m}$. We define

$$V_2^m f = \sum_{i=1}^n f(\tau_{i+3/2}^m) B_{i,2}^m, \quad \text{with} \quad \tau_i^m = \frac{\tau_{i+1}^m + \tau_{i+2}^m}{2},$$

$$P_2^m f = \sum_{i=1}^n \left(-\frac{1}{2} f(\tau_{i+1}^m) + 2f(\tau_{i+3/2}^m) - \frac{1}{2} f(\tau_{i+2}^m) \right) B_{i,2}^m,$$

and $B_{i,2}^m$ is the i th quadratic B-spline on $\boldsymbol{\tau}^m$. As approximations to the norms $\|f - V_2^m f\|_{[0,\pi]}$ and $\|f - P_2^m f\|_{[0,\pi]}$ we use

$$E_V^m = \max_{0 \leq j \leq 100} |f(j\pi/100) - V_2^m f(j\pi/100)|,$$

$$E_P^m = \max_{0 \leq j \leq 100} |f(j\pi/100) - P_2^m f(j\pi/100)|.$$

Write a computer program to compute numerically the values of E_V^m and E_P^m for $m = 0, 1, 2, 3, 4, 5$, and the ratios E_V^m/E_V^{m-1} and E_P^m/E_P^{m-1} for $1 \leq m \leq 5$. What can you deduce about the approximation order of the two methods?

Make plots of $V_2^m f$, $P_2^m f$, $f - V_2^m f$, and $f - P_2^m f$ for some values of m .

- 9.3** Suppose we have $m \geq 3$ data points $(x_i, f(x_i))_{i=1}^m$ sampled from a function f , where the abscissas $\mathbf{x} = (x_i)_{i=1}^m$ satisfy $x_1 < \dots < x_m$. In this exercise we want to derive a local quasi-interpolation scheme which only uses the data values at the x_i 's and which has $O(h^3)$ order of accuracy if the y -values are sampled from a smooth function f . The method requires m to be odd.

From $\mathbf{x}$ we form a 3-regular knot vector by using every second data point as a knot

$$\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+3} = (x_1, x_1, x_1, x_3, x_5, \dots, x_{m-2}, x_m, x_m, x_m), \quad (9.37)$$

where $n = (m+3)/2$. In the quadratic spline space $\mathbb{S}_{2,\boldsymbol{\tau}}$ we can then construct the spline

$$Q_2 f = \sum_{j=1}^n \lambda_j(f) B_{j,2}, \quad (9.38)$$

where the B-spline coefficients $\lambda_j(f)_{j=1}^n$ are defined by the rule

$$\lambda_j(f) = \frac{1}{2} \left(-\theta_j^{-1} f(x_{2j-3}) + \theta_j^{-1} (1 + \theta_j)^2 f(x_{2j-2}) - \theta_j f(x_{2j-1}) \right), \quad (9.39)$$

for $j = 1, \dots, n$. Here $\theta_1 = \theta_n = 1$ and

$$\theta_j = \frac{x_{2j-2} - x_{2j-3}}{x_{2j-1} - x_{2j-2}}$$

for $j = 2, \dots, n-1$.

- a) Show that Q_2 simplifies to P_2 given by (9.6) when the data abscissas are uniformly spaced.
- b) Show that $Q_2 p = p$ for all $p \in \pi_2$ and that because of the multiple abscissas at the ends we have $\lambda_1(f) = f(x_1)$, $\lambda_n(f) = f(x_m)$, so only the original data are used to define $Q_2 f$. (Hint: Use the formula in Exercise 1.
- c) Show that for $j = 1, \dots, n$ and $f \in \mathbb{C}_\Delta[x_1, x_m]$

$$|\lambda_j(f)| \leq (2\theta + 1) \|f\|_{\infty, [\tau_{j+1}, \tau_{j+2}]},$$

where

$$\theta = \max_{1 \leq j \leq n} \{\theta_j^{-1}, \theta_j\}.$$

- d) Show that for $l = 3, \dots, n$, $f \in \mathbb{C}_\Delta[x_1, x_m]$, and $x \in [\tau_l, \tau_{l+1}]$

$$|Q_2(f)(x)| \leq (2\theta + 1) \|f\|_{\infty, [\tau_{l-1}, \tau_{l+2}]}.$$

- e) Show that for $l = 3, \dots, n$ and $f \in \mathbb{C}_\Delta[x_1, x_m]$

$$\|f - Q_2 f\|_{\infty, [\tau_l, \tau_{l+1}]} \leq (2\theta + 2) \text{dist}_{[\tau_{l-1}, \tau_{l+2}]}(f, \pi_2).$$

- f) Show that for $f \in \mathbb{C}_\Delta^3[x_1, x_m]$ we have the $O(h^3)$ estimate

$$\|f - Q_2 f\|_{\infty, [x_1, x_m]} \leq K(\theta) |\Delta x|^3 \|D^3 f\|_{\infty, [x_1, x_m]},$$

where

$$|\Delta x| = \max_j |x_{j+1} - x_j|$$

and the constant $K(\theta)$ only depends on θ .

Chapter 10

Shape Preserving Properties of B-splines

In earlier chapters we have seen a number of examples of the close relationship between a spline function and its B-spline coefficients. This is especially evident in the properties of the Schoenberg operator, but the same phenomenon is apparent in the diagonal property of the blossom, the stability of the B-spline basis, the convergence of the control polygon to the spline it represents and so on. In the present chapter we are going to add to this list by relating the number of zeros of a spline to the number of sign changes in the sequence of its B-spline coefficients. From this property we shall obtain an accurate characterisation of when interpolation by splines is uniquely solvable. In the final section we show that the knot insertion matrix and the B-spline collocation matrix are totally positive, i.e., all their square submatrices have nonnegative determinants.

10.1 Bounding the number of zeros of a spline

In Section 4.5 of Chapter 4 we showed that the number of sign changes in a spline is bounded by the number of sign changes in its B-spline coefficients, a generalisation of Descartes' rule of signs for polynomials, Theorem 4.23. Theorem 4.25 is not a completely satisfactory generalisation of Theorem 4.23 since it does not allow multiple zeros. In this section we will prove a similar result that does allow multiple zeros, but we cannot allow the most general spline functions. we have to restrict ourselves to *connected splines*.

Definition 10.1. A spline $f = \sum_{j=1}^n c_j B_{j,d}$ in $\mathbb{S}_{d,t}$ is said to be *connected* if for each x in (t_1, t_{n+d+1}) there is some j such that $t_j < x < t_{j+d+1}$ and $c_j \neq 0$. A point x where this condition fails is called a *splitting point* for f .

To develop some intuition about connected splines, let us see when a spline is not connected. A splitting point of f can be of two kinds:

- (i) The splitting point x is not a knot. If $t_\mu < x < t_{\mu+1}$, then $t_j < x < t_{j+d+1}$ for $j = \mu - d, \dots, \mu$ (assuming the knot vector is long enough) so we must have $c_{\mu-d} = \dots = c_\mu = 0$. In other words f must be identically zero on $(t_\mu, t_{\mu+1})$. In this case f splits into two spline

functions f_1 and f_2 with knot vectors $\mathbf{t}^1 = (t_j)_{j=1}^\mu$ and $\mathbf{t}^2 = (t_j)_{j=\mu+1}^{n+d+1}$. We clearly have

$$f_1 = \sum_{j=1}^{\mu-d-1} c_j B_{j,d}, \quad f_2 = \sum_{j=\mu+1}^n c_j B_{j,d}.$$

(ii) The splitting point x is a knot of multiplicity m , say

$$t_\mu < x = t_{\mu+1} = \cdots = t_{\mu+m} < t_{\mu+m+1}.$$

In this case we have $t_j < x < t_{j+1+d}$ for $j = \mu + m - d, \dots, \mu$. We must therefore have $c_{\mu+m-d} = \cdots = c_\mu = 0$. (Note that if $m = d + 1$, then no coefficients need to be zero). This means that all the B-splines that “cross” x do not contribute to $f(x)$. It therefore splits into two parts f_1 and f_2 , but now the two pieces are not separated by an interval, but only by the single point x . The knot vector of f_1 is $\mathbf{t}^1 = (t_j)_{j=1}^{\mu+m}$ while the knot vector of f_2 is $\mathbf{t}^2 = (t_j)_{j=\mu+1}^{n+d+1}$. The two spline functions are given by

$$f_1 = \sum_{j=1}^{\mu+m-d-1} c_j B_{j,d}, \quad f_2 = \sum_{j=\mu+1}^n c_j B_{j,d}.$$

Before getting on with our zero counts we need the following lemma.

Lemma 10.2. Suppose that z is a knot that occurs m times in $\mathbf{t}$,

$$t_i < z = t_{i+1} = \cdots = t_{i+m} < t_{i+m+1}$$

for some i . Let $f = \sum_j c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,\mathbf{t}}$. Then

$$c_j = \frac{1}{d!} \sum_{k=0}^{d-m} (-1)^k D^{d-k} \rho_{j,d}(z) D^k f(z) \quad (10.1)$$

for all j such that $t_j < z < t_{j+d+1}$, where $\rho_{j,d}(y) = (y - t_{j+1}) \cdots (y - t_{j+d})$.

Proof. Recall from Theorem 8.5 that the B-spline coefficients of f can be written as

$$c_j = \lambda_j f = \frac{1}{d!} \sum_{k=0}^d (-1)^k D^{d-k} \rho_{j,d}(y) D^k f(y),$$

where y is a number such that $B_{j,d}(y) > 0$. In particular, we may choose $y = z$ for $j = i + m - d, \dots, i$ so

$$c_j = \lambda_j f = \frac{1}{d!} \sum_{k=0}^d (-1)^k D^{d-k} \rho_{j,d}(z) D^k f(z), \quad (10.2)$$

for these values of j . But in this case $\rho_{j,d}(y)$ contains the factor $(y - t_{i+1}) \cdots (y - t_{i+m}) = (y - z)^m$ so $D^{d-k} \rho_{j,d}(z) = 0$ for $k > d - m$ and $j = i + m - d, \dots, i$, i.e., for all values of j such that $t_j < z < t_{j+d+1}$. The formula (10.1) therefore follows from (10.2). ■

In the situation of Lemma 10.2, we know from Lemma 2.3 that $D^k f$ is continuous at z for $k = 0, \dots, d - m$, but $D^{d+1-m} f$ may be discontinuous. Equation (10.1) therefore shows that the B-spline coefficients of f can be computed solely from continuous derivatives of f at a point.

Lemma 10.3. *Let f be a spline that is connected. For each x in (t_1, t_{n+d+1}) there is then a nonnegative integer r such that $D^r f$ is continuous at x and $D^r f(x) \neq 0$.*

Proof. The claim is clearly true if x is not a knot, for otherwise f would be identically zero on an interval and therefore not connected. Suppose next that x is a knot of multiplicity m . Then the first discontinuous derivative at x is $D^{d-m+1} f$, so if the claim is not true, we must have $D^k f(x) = 0$ for $k = 0, \dots, d - m$. But then we see from Lemma 10.2 that $c_l = \lambda_l f = 0$ for all l such that $t_l < x < t_{l+d+1}$. But this is impossible since f is connected. ■

The lemma shows that we can count zeros of connected splines precisely as for smooth functions. If f is a connected spline then a zero must be of the form $f(z) = Df(z) = \dots = D^{r-1} f(z) = 0$ with $D^r f(z) \neq 0$ for some integer r . Moreover $D^r f$ is continuous at z . The total number of zeros of f on (a, b) , counting multiplicities, is denoted $Z(f) = Z_{(a,b)}(f)$. Recall from Definition 4.21 that $S^-(\mathbf{c})$ denotes the number of sign changes in the vector $\mathbf{c}$ (zeros are completely ignored).

Example 10.4. Below are some examples of zero counts of functions. For comparison we have also included counts of sign changes. All zero counts are over the whole real line.

$$\begin{array}{llll} Z(x) = 1, & S^-(x) = 1, & Z(x(1-x)^2) = 3, & S^-(x(1-x)^2) = 1, \\ Z(x^2) = 2, & S^-(x^2) = 0, & Z(x^3(1-x)^2) = 5, & S^-(x^3(1-x)^2) = 1, \\ Z(x^7) = 7, & S^-(x^7) = 1, & Z(-1-x^2+\cos x) = 2, & S^-(-1-x^2+\cos x) = 0. \end{array}$$

We are now ready to prove a generalization of Theorem 4.23 that allows zeros to be counted with multiplicities.

Theorem 10.5. *Let $f = \sum_{j=1}^n c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,t}$ that is connected. Then*

$$Z_{(t_1, t_{n+d+1})}(f) \leq S^-(\mathbf{c}) \leq n - 1.$$

Proof. Let $z_1 < z_2 < \dots < z_\ell$ be the zeros of f in the interval (t_1, t_{n+d+1}) , each of multiplicity r_i ; Lemma 10.2 shows that z_i occurs at most $d - r_i$ times in $\mathbf{t}$. For if z_i occurred $m > d - r_i$ times in $\mathbf{t}$ then $d - m < r_i$, and hence all the derivatives of f involved in (10.1) would be zero for all j such that $t_j < z < t_{j+d+1}$. But this means that z is a splitting point for f which is impossible since f is connected.

Now we form a new knot vector $\hat{\mathbf{t}}$ where z_i occurs exactly $d - r_i$ times and the numbers $z_i - h$ and $z_i + h$ occur $d + 1$ times. Here h is a number that is small enough to ensure that there are no other zeros of f or knots from $\mathbf{t}$ other than z_i in $[z_i - h, z_i + h]$ for $1 \leq i \leq \ell$. Let $\hat{\mathbf{c}}$ be the B-spline coefficients of f relative to $\hat{\mathbf{t}}$. By Lemma 4.24 we then have $S^-(\hat{\mathbf{c}}) \leq S^-(\mathbf{c})$ so it suffices to prove that $Z_{(t_1, t_{n+d+1})}(f) \leq S^-(\hat{\mathbf{c}})$. But since

$$Z_{(t_1, t_{n+d+1})}(f) = \sum_{i=1}^{\ell} Z_{(z_i-h, z_i+h)}(f),$$

it suffices to establish the theorem in the following situation: The knot vector is given by

$$\mathbf{t} = (\overbrace{z-h, \dots, z-h}^{d+1}, \overbrace{z, \dots, z}^{d-r}, \overbrace{z+h, \dots, z+h}^{d+1})$$

and z is a zero of f of multiplicity r . The key to proving the theorem in this more specialised situation is to show that

$$c_j = \frac{(d-r)!}{d!} (-1)^{d+1-j} h^r D^r f(z), \quad j = d+1-r, \dots, d+1, \quad (10.3)$$

as this means that the $r+1$ coefficients $(c_j)_{j=d+1-r}^{d+1}$ alternate in sign and $S^-(\mathbf{c}) \geq r = Z_{(z-h, z+h)}(f)$. Fix j in the range $d+1-r \leq j \leq d+1$. By equation (10.1) we have

$$c_j = \frac{1}{d!} \sum_{k=0}^r (-1)^k D^{d-k} \rho_{j,d}(z) D^k f(z) = \frac{(-1)^r}{d!} D^{d-r} \rho_{j,d}(z) D^r f(z),$$

since $D^j f(z) = 0$ for $j = 0, \dots, r-1$. With our special choice of knot vector we have

$$\rho_{j,d}(y) = (y-z+h)^{d+1-j} (y-z)^{d-r} (y-z-h)^{r-d-1+j}.$$

Taking $d-r$ derivatives we therefore obtain

$$D^{d-r} \rho_{j,d}(z) = (d-r)! h^{d+1-j} (-h)^{r-d-1+j} = (d-r)! (-1)^{r-d-1+j} h^r$$

and (10.3) follows. ■

Figures 10.1 (a)–(d) show some examples of splines with multiple zeros of the sort discussed in the proof of Theorem 10.5. All the knot vectors are $d+1$ -regular on the interval $[0, 2]$, with additional knots at $x = 1$. In Figure 10.1 (a) there is one knot at $x = 1$ and the spline is the polynomial $(x-1)^2$ which has a double zero at $x = 1$. The control polygon models the spline in the normal way and has two sign changes. In Figure 10.1 (b) the knot vector is the same, but the spline is now the polynomial $(x-1)^3$. In this case the multiplicity of the zero is so high that the spline has a splitting point at $x = 1$. The construction in the proof of Theorem 10.5 prescribes a knot vector with no knots at $x = 1$ in this case. Figure 10.1 (c) shows the polynomial $(x-1)^3$ as a degree 5 spline on a 6-regular knot vector with a double knot at $x = 1$. As promised by the theorem and its proof the coefficients change sign exactly three times. The spline in Figure 10.1 (d) is more extreme. It is the polynomial $(x-1)^8$ represented as a spline of degree 9 with one knot at $x = 1$. The control polygon has the required 8 changes of sign.

10.2 Uniqueness of spline interpolation

Having established Theorem 10.5, we return to the problem of showing that the B-spline collocation matrix that occurs in spline interpolation, is nonsingular. We first consider Lagrange interpolation, and then turn to Hermite interpolation where we also allow interpolation derivatives.

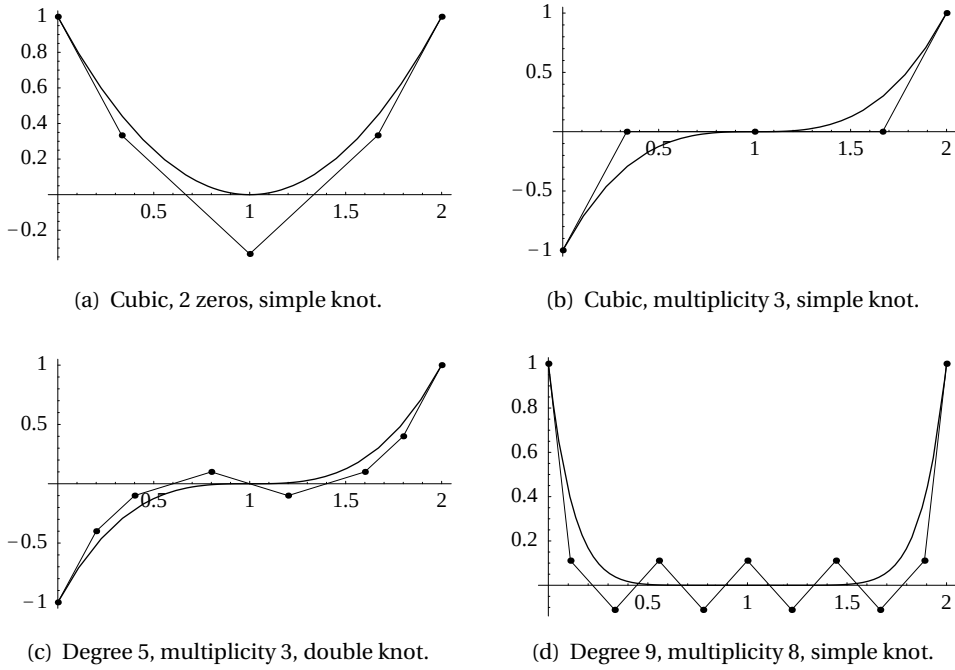


Figure 10.1. Splines of varying degree with a varying number of zeros and knots at $x = 1$.

10.2.1 Lagrange Interpolation

In Chapter 8 we studied spline interpolation. With a spline space $\mathbb{S}_{d,t}$ of dimension n and data $(y_i)_{i=1}^n$ given at n distinct points $x_1 < x_2 < \dots < x_n$, the aim is to determine a spline $g = \sum_{i=1}^n c_i B_{i,d}$ in $\mathbb{S}_{d,t}$ such that

$$g(x_i) = y_i, \quad \text{for } i = 1, \dots, n. \quad (10.4)$$

This leads to the linear system of equations

$$A\mathbf{c} = \mathbf{y},$$

where

$$A = \begin{pmatrix} B_{1,d}(x_1) & B_{2,d}(x_1) & \dots & B_{n,d}(x_1) \\ B_{1,d}(x_2) & B_{2,d}(x_2) & \dots & B_{n,d}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_{1,d}(x_n) & B_{2,d}(x_n) & \dots & B_{n,d}(x_n) \end{pmatrix}, \quad \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

The matrix A is often referred to as the *B-spline collocation matrix*. Since $B_{i,d}(x)$ is nonzero only if $t_i < x < t_{i+d+1}$ (we may allow $t_i = x$ if $t_i = t_{i+d} < t_{i+d+1}$), the matrix A will in general be sparse. The following theorem tells us exactly when A is nonsingular.

Theorem 10.6. Let $\mathbb{S}_{d,t}$ be a given spline space, and let $x_1 < x_2 < \dots < x_n$ be n distinct numbers. The collocation matrix A with entries $(B_{j,d}(x_i))_{i,j=1}^n$ is nonsingular if and only if its

diagonal is positive, i.e.,

$$B_{i,d}(x_i) > 0 \quad \text{for } i = 1, \dots, n. \quad (10.5)$$

Proof. We start by showing that $\mathbf{A}$ is singular if a diagonal entry is zero. Suppose that $x_q \leq t_q$ (strict inequality if $t_q = t_{q+d} < t_{q+d+1}$) for some q so that $B_{q,d}(x_q) = 0$. By the support properties of B-splines we must have $a_{i,j} = 0$ for $i = 1, \dots, q$ and $j = q, \dots, n$. But this means that only the $n - q$ last entries of each of the last $n - q + 1$ columns of $\mathbf{A}$ can be nonzero; these columns must therefore be linearly dependent and $\mathbf{A}$ must be singular. A similar argument shows that $\mathbf{A}$ is also singular if $x_q \geq t_{q+d+1}$.

To show the converse, suppose that (10.5) holds but $\mathbf{A}$ is singular. Then there is a nonzero vector $\mathbf{c}$ such that $\mathbf{A}\mathbf{c} = 0$. Let $f = \sum_{i=1}^n c_i B_{i,d}$ denote the spline with B-spline coefficients $\mathbf{c}$. We clearly have $f(x_q) = 0$ for $q = 1, \dots, n$. Let G denote the set

$$G = \cup_i \{(t_i, t_{i+d+1}) \mid c_i \neq 0\}.$$

Since each x in G must be in (t_i, t_{i+d+1}) for some i with $c_i \neq 0$, we note that G contains no splitting points of f . Note that if $x_i = t_i = t_{i+d} < t_{i+d+1}$ occurs at a knot of multiplicity $d + 1$, then $0 = f(x_i) = c_i$. To complete the proof, suppose first that G is an open interval. Since x_i is in G if $c_i \neq 0$, the number of zeros of f in G is greater than or equal to the number ℓ of nonzero coefficients in $\mathbf{c}$. Since we also have $S^-(\mathbf{c}) < \ell \leq Z_G(f)$, we have a contradiction to Theorem 10.5. In general G consists of several subintervals which means that f is not connected, but can be written as a sum of connected components, each defined on one of the subintervals. The above argument then leads to a contradiction on each subinterval, and hence we conclude that $\mathbf{A}$ is nonsingular. ■

Theorem 10.6 makes it simple to ensure that the collocation matrix is nonsingular. We just place the knots and interpolation points in such a way that $t_i < x_i < t_{i+d+1}$ for $i = 1, \dots, n$ (note again that if $t_i = t_{i+d} < t_{i+d+1}$, then $x_i = t_i$ is allowed).

10.2.2 Hermite Interpolation

In earlier chapters, particularly in Chapter 8, we made use of polynomial interpolation with Hermite data—data based on derivatives as well as function values. This is also of interest for splines, and as for polynomials this is conveniently indicated by allowing the interpolation point to coalesce. If for example $x_1 = x_2 = x_3 = x$, we take x_1 to signify interpolation of function value at x , the second occurrence of x signifies interpolation of first derivative, and the third tells us to interpolate second derivative at x . If we introduce the notation

$$\lambda_x(i) = \max_j \{j \mid x_{i-j} = x_i\}$$

and assume that the interpolation points are given in nondecreasing order as $x_1 \leq x_2 \leq \dots \leq x_n$, then the interpolation conditions are

$$D^{\lambda_x(i)} g(x_i) = D^{\lambda_x(i)} f(x_i) \quad (10.6)$$

where f is a given function and g is the spline to be determined. Since we are dealing with splines of degree d we cannot interpolate derivatives of higher order than d ; we therefore

assume that $x_i < x_{i+d+1}$ for $i = 1, \dots, n-d-1$. At a point of discontinuity (10.6) is to be interpreted according to our usual convention of taking limits from the right. The (i, j) -entry of the collocation matrix A is now given by

$$a_{i,j} = D^{\lambda_x(i)} B_{j,d}(x_i),$$

and as before the interpolation problem is generally solvable if and only if the collocation matrix is nonsingular. Also as before, it turns out that the collocation matrix is nonsingular if and only if $t_i \leq x_i < t_{i+d+1}$, where equality is allowed in the first inequality only if $D^{\lambda_x(i)} B_{j,d}(x_i) \neq 0$. This result will follow as a special case of our next theorem where we consider an even more general situation.

At times it is of interest to know exactly when a submatrix of the collocation matrix is nonsingular. The submatrices we consider are obtained by removing the same number of rows and columns from A . Any columns may be removed, or equivalently, we consider a subset $\{B_{j_1,d}, \dots, B_{j_\ell,d}\}$ of the B-splines. When removing rows we have to be a bit more careful. The convention is that if a row with derivatives of order r at z is included, then we also include all the lower order derivatives at z . This is most easily formulated by letting the sequence of interpolation points only contain ℓ points as in the following theorem.

Theorem 10.7. *Let $\mathbb{S}_{d,t}$ be a spline space and let $\{B_{j_1,d}, \dots, B_{j_\ell,d}\}$ be a subsequence of its B-splines. Let $x_1 \leq \dots \leq x_\ell$ be a sequence of interpolation points with $x_i \leq x_{i+d+1}$ for $i = 1, \dots, \ell-d-1$. Then the $\ell \times \ell$ matrix $A(\mathbf{j})$ with entries given by*

$$a_{i,q} = D^{\lambda_x(i)} B_{j_q,d}(x_i)$$

for $i = 1, \dots, \ell$ and $q = 1, \dots, \ell$ is nonsingular if and only if

$$t_{j_i} \leq x_i < t_{j_i+d+1}, \quad \text{for } i = 1, \dots, \ell, \quad (10.7)$$

where equality is allowed in the first inequality if $D^{\lambda_x(i)} B_{j_i,d}(x_i) \neq 0$.

Proof. The proof follows along the same lines as the proof of Theorem 10.6. The most challenging part is the proof that condition (10.7) is necessary so we focus on this. Suppose that (10.7) holds, but $A(\mathbf{j})$ is singular. Then we can find a nonzero vector $\mathbf{c}$ such that $A(\mathbf{j})\mathbf{c} = \mathbf{0}$. Let $f = \sum_{i=1}^{\ell} c_i B_{j_i,d}$ denote the spline with $\mathbf{c}$ as its B-spline coefficients, and let G denote the set

$$G = \cup_{i=1}^{\ell} \{(t_{j_i}, t_{j_i+d+1}) \mid c_i \neq 0\}.$$

To carry through the argument of Theorem 10.6 we need to verify that in the exceptional case where $x_i = t_{j_i}$ then $c_i = 0$.

Set $r = \lambda_x(i)$ and suppose that the knot t_{j_i} occurs m times in $\mathbf{t}$ and that $t_{j_i} = x_i$ so $D^r B_{j_i,d}(x_i) \neq 0$. In other words

$$t_\mu < x_i = t_{\mu+1} = \dots = t_{\mu+m} < t_{\mu+m+1}$$

for some integer μ , and in addition $j_i = \mu + k$ for some integer k with $1 \leq k \leq m$. Note that f satisfies

$$f(x_i) = Df(x_i) = \dots = D^r f(x_i) = 0.$$

(Remember that if a derivative is discontinuous at x_i we take limits from the right.) Recall from Lemma 2.3 that all B-splines have continuous derivatives up to order $d - m$ at x_i . Since $D^r B_{j_i}$ clearly is discontinuous at x_i , it must be true that $r > d - m$. We therefore have $f(x_i) = Df(x_i) = \dots = D^{d-m}f(x_i) = 0$ and hence $c_{\mu+m-d} = \dots = c_\mu = 0$ by Lemma 10.2. The remaining interpolation conditions at x_i are $D^{d-m+1}f(x_i) = D^{d-m+2}f(x_i) = \dots = D^r f(x_i) = 0$. Let us consider each of these in turn. By the continuity properties of B-splines we have $D^{d-m+1}B_{\mu+1}(x_i) \neq 0$ and $D^{d-m+1}B_{\mu+v} = 0$ for $v > 1$. This means that

$$0 = D^{d-m+1}f(x_i) = c_{\mu+1}D^{d-m+1}B_{\mu+1}(x_i)$$

and $c_{\mu+1} = 0$. Similarly, we also have

$$0 = D^{d-m+2}f(x_i) = c_{\mu+2}D^{d-m+2}B_{\mu+2}(x_i),$$

and hence $c_{\mu+2} = 0$ since $D^{d-m+2}B_{\mu+2}(x_i) \neq 0$. Continuing this process we find

$$0 = D^r f(x_i) = c_{\mu+r+m-d}D^r B_{\mu+r+m-d}(x_i),$$

so $c_{\mu+r+m-d} = 0$ since $D^r B_{\mu+r+m-d}(x_i) \neq 0$. This argument also shows that j_i cannot be chosen independently of r ; we must have $j_i = \mu + r + m - d$.

For the rest of the proof it is sufficient to consider the case where G is an open interval, just as in the proof of Theorem 10.6. Having established that $c_i = 0$ if $x_i = t_{j_i}$, we know that if $c_i \neq 0$ then $x_i \in G$. The number of zeros of f in G (counting multiplicities) is therefore greater than or equal to the number of nonzero coefficients. But this is impossible according to Theorem 10.5. ■

10.3 Total positivity

In this section we are going to deduce another interesting property of the knot insertion matrix and the B-spline collocation matrix, namely that they are totally positive. We follow the same strategy as before and establish this first for the knot insertion matrix and then obtain the total positivity of the collocation matrix by recognising it as a submatrix of a knot insertion matrix.

Definition 10.8. A matrix A in $\mathbb{R}^{m,n}$ is said to be totally positive if all its square submatrices have nonnegative determinant. More formally, let $\mathbf{i} = (i_1, i_2, \dots, i_\ell)$ and $\mathbf{j} = (j_1, j_2, \dots, j_\ell)$ be two integer sequences such that

$$1 \leq i_1 < i_2 < \dots < i_\ell \leq m, \quad (10.8)$$

$$1 \leq j_1 < j_2 < \dots < j_\ell \leq n, \quad (10.9)$$

and let $A(\mathbf{i}, \mathbf{j})$ denote the submatrix of A with entries $(a_{i_p, j_q})_{p,q=1}^\ell$. Then A is totally positive if $\det A(\mathbf{i}, \mathbf{j}) \geq 0$ for all sequences $\mathbf{i}$ and $\mathbf{j}$ on the form (10.8) and (10.9), for all ℓ with $1 \leq \ell \leq \min\{m, n\}$.

We first show that knot insertion matrices are totally positive.

Theorem 10.9. Let τ and t be two knot vectors with $\tau \subseteq t$. Then the knot insertion matrix from $\mathbb{S}_{d,\tau}$ to $\mathbb{S}_{d,t}$ is totally positive.

Proof. Suppose that there are k more knots in t than in τ ; our proof is by induction on k . We first note that if $k = 0$, then $A = I$, the identity matrix, while if $k = 1$, then A is a bi-diagonal matrix with one more rows than columns. Let us denote the entries of A by $(\alpha_j(i))_{i,j=1}^{n+1,n}$ (if $k = 0$ the range of i is $1, \dots, n$). In either case all the entries are nonnegative and $\alpha_j(i) = 0$ for $j < i-1$ and $j > i$. Consider now the determinant of $A(i, j)$. If $j_\ell \geq i_\ell$ then $j_\ell > i_q$ for $q = 1, \dots, \ell-1$ so $\alpha_{j_\ell}(i_q) = 0$ for $q < \ell$. This means that only the last entry of the last column of $A(i, j)$ is nonzero. The other possibility is that $j_\ell \leq i_\ell - 1$ so that $j_q < i_\ell - 1$ for $q < \ell$. Then $\alpha_{j_q}(i_\ell) = 0$ for $q < \ell$ so only the last entry of the last row of $A(i, j)$ is nonzero. Expanding the determinant either by the last column or last row we therefore have $\det A(i, j) = \alpha_{j_\ell}(i_\ell) \det A(i', j')$ where $i' = (i_1, \dots, i_{\ell-1})$ and $j' = (j_1, \dots, j_{\ell-1})$. Continuing this process we find that

$$\det A(i, j) = \alpha_{j_1}(i_1) \alpha_{j_2}(i_2) \cdots \alpha_{j_\ell}(i_\ell)$$

which clearly is nonnegative.

For $k \geq 2$, we make use of the factorization

$$A = A_k \cdots A_1 = A_k B, \quad (10.10)$$

where each A_r corresponds to insertion of one knot and $B = A_{k-1} \cdots A_1$ is the knot insertion matrix for inserting $k-1$ of the knots. By the induction hypothesis we know that both A_k and B are totally positive; we must show that A is totally positive. Let (a_i) and (b_i) denote the rows of A and B , and let $(\alpha_j(i))_{i,j=1}^{m,m-1}$ denote the entries of A_k . From (10.10) we have

$$a_i = \alpha_{i-1}(i) b_{i-1} + \alpha_i(i) b_i \quad \text{for } i = 1, \dots, m,$$

where $\alpha_0(1) = \alpha_m(m) = 0$. Let $a_i(j)$ and $b_i(j)$ denote the vectors obtained by keeping only entries $(j_q)_{q=1}^\ell$ of a_i and b_i respectively. Row q of $A(i, j)$ of A is then given by

$$a_{i_q}(j) = \alpha_{i_q-1}(i_q) b_{i_q-1}(j) + \alpha_{i_q}(i_q) b_{i_q}(j).$$

Using the linearity of the determinant in row q we therefore have

$$\begin{aligned} \det \begin{pmatrix} a_{i_1}(j) \\ \vdots \\ a_{i_q}(j) \\ \vdots \\ a_{i_\ell}(j) \end{pmatrix} &= \det \begin{pmatrix} a_{i_1}(j) \\ \vdots \\ \alpha_{i_q-1}(i_q) b_{i_q-1}(j) + \alpha_{i_q}(i_q) b_{i_q}(j) \\ \vdots \\ a_{i_\ell}(j) \end{pmatrix} \\ &= \alpha_{i_q-1}(i_q) \det \begin{pmatrix} a_{i_1}(j) \\ \vdots \\ b_{i_q-1}(j) \\ \vdots \\ a_{i_\ell}(j) \end{pmatrix} + \alpha_{i_q}(i_q) \det \begin{pmatrix} a_{i_1}(j) \\ \vdots \\ b_{i_q}(j) \\ \vdots \\ a_{i_\ell}(j) \end{pmatrix}. \end{aligned}$$

By expanding the other rows similarly we find that $\det \mathbf{A}(\mathbf{i}, \mathbf{j})$ can be written as a sum of determinants of submatrices of $\mathbf{B}$, multiplied by products of $\alpha_j(i)$'s. By the induction hypothesis all these quantities are nonnegative, so the determinant of $\mathbf{A}(\mathbf{i}, \mathbf{j})$ must also be nonnegative. Hence $\mathbf{A}$ is totally positive. ■

Knowing that the knot insertion matrix is totally positive, we can prove a similar property of the B-spline collocation matrix, even in the case where multiple collocation points are allowed.

Theorem 10.10. *Let $\mathbb{S}_{d,\tau}$ be a spline space and let $\{B_{j_1,d}, \dots, B_{j_\ell,d}\}$ be a subsequence of its B-splines. Let $x_1 \leq \dots \leq x_\ell$ be a sequence of interpolation points with $x_i \leq x_{i+d+1}$ for $i = 1, \dots, \ell - d - 1$, and denote by $\mathbf{A}(\mathbf{j})$ the $\ell \times \ell$ matrix with entries given by*

$$a_{i,q} = D^{\lambda_x(i)} B_{j_q,d}(x_i)$$

for $i = 1, \dots, \ell$ and $q = 1, \dots, \ell$. Then

$$\det \mathbf{A}(\mathbf{j}) \geq 0.$$

Proof. We first prove the claim in the case $x_1 < x_2 < \dots < x_\ell$. By inserting knots of multiplicity $d+1$ at each of $(x_i)_{i=1}^\ell$ we obtain a knot vector $\mathbf{t}$ that contains $\boldsymbol{\tau}$ as a subsequence. If $t_{i-1} < t_i = t_{i+d} < t_{i+d+1}$ we know from Lemma 2.3 that $B_{j,d,\tau}(t_i) = \alpha_{j,d}(i)$. This means that the matrix $\mathbf{A}(\mathbf{j})$ appears as a submatrix of the knot insertion matrix from $\boldsymbol{\tau}$ to $\mathbf{t}$. It therefore follows from Theorem 10.9 that $\det \mathbf{A}(\mathbf{j}) \geq 0$ in this case.

To prove the theorem in the general case we consider a set of distinct collocation points $y_1 < \dots < y_\ell$ and let $\mathbf{A}(\mathbf{j}, \mathbf{y})$ denote the corresponding collocation matrix. Set $\lambda^i = \lambda_x(i)$ and let ρ_i denote the linear functional given by

$$\rho_i f = \lambda^i! [y_{i-\lambda^i}, \dots, y_i] f \quad (10.11)$$

for $i = 1, \dots, \ell$. Here $[\cdot, \dots, \cdot] f$ is the divided difference of f . By standard properties of divided differences we have

$$\rho_i B_{j,d} = \sum_{s=i-\lambda^i}^i \gamma_{i,s} B_{j,d}(y_s)$$

and $\gamma_{i,i} > 0$. Denoting by $\mathbf{D}$ the matrix with (i, j) -entry $\rho_i B_{j,d}$, we find by properties of determinants and (10.11) that

$$\det \mathbf{D} = \gamma_{1,1} \cdots \gamma_{\ell,\ell} \det \mathbf{A}(\mathbf{j}, \mathbf{y}).$$

If we now let $\mathbf{y}$ tend to $\mathbf{x}$ we know from properties of the divided difference functional that $\rho_i B_j$ tends to $D^{\lambda^i} B_j$ in the limit. Hence $\mathbf{D}$ tends to $\mathbf{A}(\mathbf{j})$ so $\det \mathbf{A}(\mathbf{j}) \geq 0$. ■

Computing Zeros of Splines

A commonly occurring mathematical operation in many applications is to find the zeros of a function, and a number of methods for accomplishing this have been devised. Most of these methods work for quite general types of functions, including splines. Two popular methods are Newton's method and the secant method which both leave us with a problem that is typical of root-finding methods, namely how to find a good starting value for the iterations. For this we can exploit some standard properties of splines like the fact that a spline whose control polygon is everywhere of one sign cannot have any zeros. Likewise, a good starting point for an iterative procedure is a point in the neighbourhood of a zero of the control polygon. More refined methods exploit the fact that the control polygon converges to the spline as the spacing of the knots goes to zero. One can then start by inserting knots to obtain a control polygon where the zeros are clearly isolated and then apply a suitable iterative method to determine the zeros accurately. Although hybrid methods of this type can be tuned to perform well, there are important unanswered problems. Where should the knots be inserted in the initial phase? How many knots should be inserted in the initial phase? How should the starting value for the iterations be chosen? Will the method always converge?

In this paper we propose a simple method that provides answers to all the above questions. The method is very simple: Iteratively insert zeros of the control polygon as new knots of the spline. It turns out that all accumulation points of this procedure will be zeros of the spline function and we prove below that the method is unconditionally convergent. In addition it is essentially as efficient as Newton's method, and asymptotically it behaves like this method.

11.1 Counting zeros of the control polygon

Our root-finding algorithm reduces the problem of computing a zero of a spline to iteratively calculating zeros of the control polygon. Before we turn to the details of the algorithm, we therefore need to recall some basic facts about the control polygon.

As before we denote the control polygon of the spline $f = \sum_{i=1}^n c_i B_{i,p,t}$ in $\mathbb{S}_{p,t}$ by $\Gamma_{p,t}(f)$,

or just $\Gamma(f)$, or even Γ , when no confusion is possible. In this section we will use the notation

$$\bar{t}_{i,p} = \bar{t}_i = \frac{t_{i+1} + \cdots + t_{i+p}}{p}$$

for the knot averages, so the i th vertex of the control polygon of f is $(\bar{t}_i, c_i)$.

Recall from section 4.4 that if we insert a new knot z in $\mathbf{t}$ and form the new knot vector $\mathbf{t}^1 = \mathbf{t} \cup \{z\}$ then $f = \sum_{i=1}^n c_i B_{i,p,\mathbf{t}} = \sum_{i=1}^{n+1} c_i^1 B_{i,p,\mathbf{t}^1}$. More specifically, if $z \in [t_\mu, t_{\mu+1})$ we have $c_i^1 = c_i$ for $i = 1, \dots, \mu - p$;

$$c_i^1 = (1 - \mu_i) c_{i-1} + \mu_i c_i \quad \text{with} \quad \mu_i = \frac{z - t_i}{t_{i+p} - t_i} \quad (11.1)$$

for $i = \mu - p + 1, \dots, \mu$; and $c_i^1 = c_{i-1}$ for $i = \mu + 1, \dots, n + 1$. It is not hard to verify that the same relation holds for the knot averages,

$$\bar{t}_i^1 = (1 - \mu_i) \bar{t}_{i-1} + \mu_i \bar{t}_i \quad (11.2)$$

for $i = \mu - p + 1, \dots, \mu$. This means that the corners of Γ^1 , the refined control polygon, lie on Γ . This property is useful when studying the effect of knot insertion on the number of zeros of Γ .

We count the number of zeros of the control polygon Γ as the number of strict sign changes in the coefficient sequence $(c_i)_{i=1}^n$. The position of a zero of the control polygon is obvious when the two ends of a line segment have opposite signs. However, the control polygon can also be identically zero on an interval in which case we associate the zero with the left end point of the zero interval. More formally, if $c_{k-1} c_{k+\ell} < 0$ and $c_{k+i} = 0$ for $i = 0, \dots, \ell - 1$, we say that k is the *index* of the zero z , which is given by

$$z = \min\{x \in [\bar{t}_{k-1}, \bar{t}_{k+\ell}] \mid \Gamma(x) = 0\}.$$

Note that in this case $c_{k-1} \neq 0$ and $c_{k-1} c_k \leq 0$.

We let $S_{i,j}^-(\Gamma)$ denote the number of zeros of Γ in the half-open interval $(\bar{t}_i, \bar{t}_j]$. It is clear that $S_{1,n}^-(\Gamma) = S^-(\Gamma)$ and that $S_{i,k}^-(\Gamma) = S_{i,j}^-(\Gamma) + S_{j,k}^-(\Gamma)$ for i, j, k such that $1 \leq i < j < k \leq n$. We note that if Γ^1 is the control polygon of f after inserting one knot, then for any $k = 2, \dots, n$

$$S^-(\Gamma^1) \leq S^-(\Gamma) - S_{k-1,k}^-(\Gamma) + S_{k-1,k+1}^-(\Gamma^1). \quad (11.3)$$

To prove this we first observe that the two inequalities

$$\begin{aligned} S_{1,k-1}^-(\Gamma^1) &\leq S_{1,k-1}^-(\Gamma), \\ S_{k+1,n+1}^-(\Gamma^1) &\leq S_{k,n}^-(\Gamma), \end{aligned}$$

are true since the corners of Γ^1 lie on Γ , see (11.1). The inequality (11.3) follows from the identity

$$S^-(\Gamma^1) = S_{1,k-1}^-(\Gamma^1) + S_{k-1,k+1}^-(\Gamma^1) + S_{k+1,n+1}^-(\Gamma^1),$$

and the two inequalities above.

11.2 Root finding algorithm

The basic idea of the root finding algorithm is to exploit the close relationship between the control polygon and the spline, and we do this by using the zeros of the control polygon as an initial guess for the zeros of the spline. In the next step we refine the control polygon by inserting these zeros as knots. We can then find the zeros of the new control polygon, insert these zeros as knots and so on. The method can be formulated in a particularly simple way if we focus on determining the left-most zero. There is no loss in this since once the left-most zero has been found, we can split the spline at this point by inserting a knot of multiplicity p into $\mathbf{t}$ and then proceed with the other zeros.

Note that the case where f has a zero at the first knot t_1 can easily be detected a priori; the spline is then disconnected at t_1 , see definition 10.1 for a definition of disconnectedness. In fact, disconnected splines are degenerate, and this degeneracy is easy to detect. We therefore assume that the spline under consideration is connected.

We give a more refined version of the method in Algorithm 11.3 which focuses on determining an arbitrary zero of f .

Algorithm 11.1. *Let f be a connected spline in $\mathbb{S}_{p,\mathbf{t}}$ and set $\mathbf{t}^0 = \mathbf{t}$. Repeat the following steps for $j = 0, 1, \dots$, until the sequence (x_j) is found to converge or no zero of the control polygon can be found.*

1. *Determine the first zero x_{j+1} of the control polygon Γ^j of f relative to the space $\mathbb{S}_{p,\mathbf{t}^j}$.*
2. *Form the knot vector $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$.*

We will show below that if f has zeros, this procedure will converge to the first zero of f , otherwise it will terminate after a finite number of steps. A typical example of how the algorithm behaves is shown in Figure 11.1.

In the following, we only discuss the first iteration through Algorithm 11.1 and therefore omit the superscripts. In case $p = 1$ the control polygon and the spline are identical and so the zero is found in the first iteration. We will therefore assume $p > 1$ in the rest of this chapter. The first zero of the control polygon is the zero of the linear segment connecting the two points $(\bar{t}_{k-1}, c_{k-1})$ and $(\bar{t}_k, c_k)$ where k is the smallest zero index, i.e. k is the smallest integer such that $c_{k-1}c_k \leq 0$ and $c_{k-1} \neq 0$, see page 216 for the definition of zero index. The zero is characterised by the equation

$$(1 - \lambda)c_{k-1} + \lambda c_k = 0$$

which has the solution

$$\lambda = \frac{-c_{k-1}}{c_k - c_{k-1}}.$$

The control polygon therefore has a zero at

$$\begin{aligned} x_1 &= (1 - \lambda)\bar{t}_{k-1} + \lambda\bar{t}_k = \bar{t}_{k-1} - \frac{c_{k-1}(t_{k+p} - t_k)}{p(c_k - c_{k-1})} \\ &= \bar{t}_k - \frac{c_k(t_{k+p} - t_k)}{p(c_k - c_{k-1})}. \end{aligned} \tag{11.4}$$

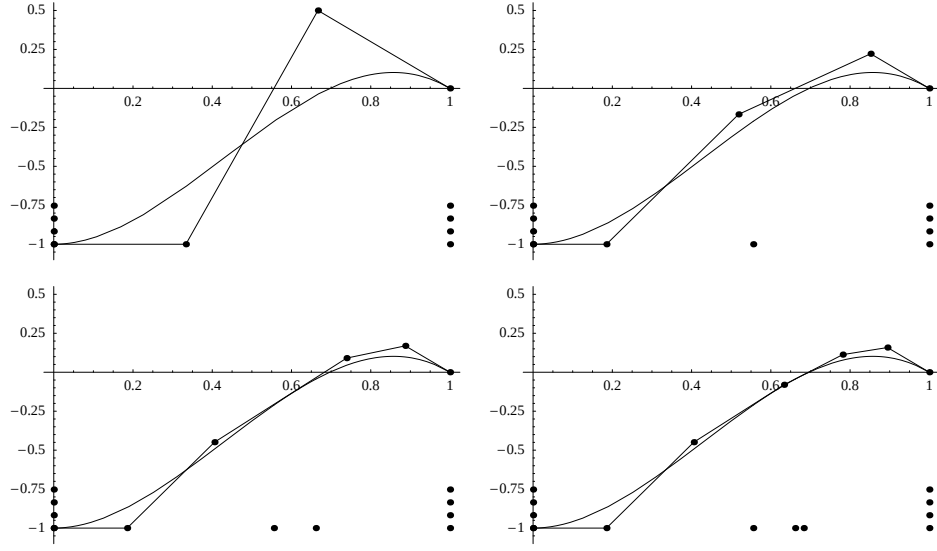


Figure 11.1. Our algorithm applied to a cubic spline with knot vector $\mathbf{t} = (0, 0, 0, 0, 1, 1, 1, 1)$ and B-spline coefficients $\mathbf{c} = (-1, -1, 1/2, 0)$

Using the notation $\Delta c_i = p(c_i - c_{i-1})/(t_{i+p} - t_i)$, we can write this in the simple form

$$x_1 = \bar{t}_k - \frac{c_k}{\Delta c_k} = \bar{t}_{k-1} - \frac{c_{k-1}}{\Delta c_k} \quad (11.5)$$

from which it is apparent that the method described by Algorithm 11.1 resembles a discrete version of Newton's method.

When x_1 is inserted in $\mathbf{t}$, we can express f on the resulting knot vector via a new coefficient vector $\mathbf{c}^1$ as in (11.1). The new control points lie on the old control polygon and hence this process is variation diminishing in the sense that the number of zeros of the control polygon is non-increasing. In fact, the knot insertion step in Algorithm 11.1 either results in a refined control polygon that has at least one zero in the interval $(\bar{t}_{k-1}^1, \bar{t}_{k+1}^1]$ or the number of zeros in the refined control polygon has been reduced by at least 2 compared with the original control polygon.

Lemma 11.2. *If k is the index of a zero of Γ and $S_{k-1, k+1}^-(\Gamma^1) = 0$ then $S^-(\Gamma^1) \leq S^-(\Gamma) - 2$.*

Proof. From (11.3) we see that the number of sign changes in Γ^1 is at least one less than in Γ , and since the number of sign changes has to decrease in even numbers the result follows. ■

This means that if Γ^1 has no zero in $(\bar{t}_{k-1}^1, \bar{t}_{k+1}^1]$, the zero in $(\bar{t}_{k-1}, \bar{t}_k]$ was a false warning; there is no corresponding zero in f . In fact, we have accomplished more than this since we have also removed a second false zero from the control polygon. If we still wish to find the first zero of f we can restart the algorithm from the leftmost zero of the refined control polygon. However, it is useful to be able to detect that zeros in the control polygon have disappeared

so we reformulate our algorithm with this ingredient. In addition, we need this slightly more elaborate algorithm to carry out a detailed convergence analysis.

Algorithm 11.3. Let f be a connected spline in $\mathbb{S}_{p,t}$, and set $\mathbf{t}^0 = \mathbf{t}$ and $\mathbf{c}^0 = \mathbf{c}$. Let $k_0 = k$ be a zero index for Γ . Repeat the following steps for $j = 0, 1, \dots$, or until the process is halted in step 3:

1. Compute $x_{j+1} = \bar{t}_{k_j}^j - c_{k_j}^j / \Delta c_{k_j}^j$.
2. Form the knot vector $\mathbf{t}^{j+1} = \mathbf{t}^j \cup \{x_{j+1}\}$ and compute the B-spline coefficients $\mathbf{c}^{j+1}$ of f in $\mathbb{S}_{p,\mathbf{t}^{j+1}}$.
3. Choose k_{j+1} to be the smallest of the two integers k_j and $k_j + 1$, but such that k_{j+1} is the index of a zero of Γ^{j+1} . Stop if no such number can be found or if f is disconnected at x_{j+1} .

Before turning to the analysis of convergence, we establish a few basic facts about the algorithm. We shall call an infinite sequence (x_j) generated by this algorithm a *zero sequence*. We also introduce the notation $\hat{\mathbf{t}}^j = (t_{k_j}^j, \dots, t_{k_j+p}^j)$ to denote what we naturally term the *active knots* at level j . In addition we denote by

$$a^j = \frac{1}{p-1} \sum_{i=1}^{p-1} t_{k_j+i}^j$$

the average of the interior, active knots.

Lemma 11.4. The zero x_{j+1} computed in Algorithm 11.3 satisfies the relations $x_{j+1} \in (\bar{t}_{k_j-1}^j, \bar{t}_{k_j}^j] \subseteq (t_{k_j}^j, t_{k_j+p}^j]$, and if $x_{j+1} = t_{k_j+p}^j$ then f is disconnected at x_{j+1} with $f(x_{j+1}) = 0$.

Proof. Since $c_{k-1}^j \neq 0$ we must have $x_{j+1} \in (\bar{t}_{k-1}^j, \bar{t}_k^j]$. Since we also have $(\bar{t}_{k-1}^j, \bar{t}_k^j] \subseteq (t_k^j, t_{k+p}^j]$, the first assertion follows.

For the second assertion, we observe that we always have $x_{j+1} \leq \bar{t}_k^j \leq t_{k+p}^j$. This means that if $x_{j+1} = t_{k+p}^j$ we must have $x_{j+1} = \bar{t}_k^j$ and $c_k^j = 0$. But then $x_{j+1} = t_{k+1}^j = \dots = t_{k+p}^j$ so $f(x_{j+1}) = c_k^j = 0$. ■

Our next result shows how the active knots at one level are derived from the active knots on the previous level.

Lemma 11.5. If $k_{j+1} = k_j$ then $\hat{\mathbf{t}}^{j+1} = \hat{\mathbf{t}}^j \cup \{x_{j+1}\} \setminus \{t_{k_j+p}^j\}$. Otherwise, if $k_{j+1} = k_j + 1$ then $\hat{\mathbf{t}}^{j+1} = \hat{\mathbf{t}}^j \cup \{x_{j+1}\} \setminus \{t_{k_j}^j\}$.

Proof. We know that $x_{j+1} \in (t_{k_j}^j, t_{k_j+p}^j]$. Therefore, if $k_{j+1} = k_j$ the latest zero x_{j+1} becomes a new active knot while $t_{k_j+p}^j$ is lost. The other case is similar. ■

11.3 Convergence

We now have the necessary tools to prove that a zero sequence (x_j) converges; afterwards we will then prove that the limit is a zero of f .

We first show convergence of the first and last active knots.

Lemma 11.6. *Let (x_j) be a zero sequence. The corresponding sequence of initial active knots $(t_{k_j}^j)_j$ is an increasing sequence that converges to some real number t_- from below, and the sequence of last active knots $(t_{k_j+p}^j)$ is a decreasing sequence that converges to some real number t_+ from above with $t_- \leq t_+$.*

Proof. From Lemma 11.4 we have $x_{j+1} \in (t_{k_j}^j, t_{k_j+p}^j]$, and due to Lemma 11.5 we have $t_{k_{j+1}}^{j+1} \geq t_{k_j}^j$ and $t_{k_{j+1}+p}^{j+1} \leq t_{k_j+p}^j$ for each j . Since $t_{k_j}^j \leq t_{k_j+p}^j$ the result follows. ■

This Lemma implies that $x_j \in (t_{k_\ell}^\ell, t_{k_\ell+p}^\ell]$ for all j and ℓ such that $j > \ell$. Also, the set of intervals $\{[t_{k_j}^j, t_{k_j+p}^j]\}_{j=0}^\infty$ in which we insert the new knots is nested and these intervals tend to a limit,

$$[t_{k_0}, t_{k_0+p}] \supseteq [t_{k_1}^1, t_{k_1+p}^1] \supseteq [t_{k_2}^2, t_{k_2+p}^2] \supseteq \cdots \supseteq [t_-, t_+].$$

Proposition 11.7. *A zero sequence converges to either t_- or t_+ .*

The proof of convergence goes via several lemmas; however in one situation the result is quite obvious: From Lemma 11.6 we deduce that if $t_- = t_+$, the active knots and hence the zero sequence must all converge to this number so there is nothing to prove. We therefore focus on the case $t_- < t_+$.

Lemma 11.8. *None of the knot vectors $(t^j)_{j=0}^\infty$ have knots in (t_-, t_+) .*

Proof. Suppose that there is at least one knot in (t_-, t_+) ; by the definition of t_- and t_+ this must be an active knot for all j . Then, for all j sufficiently large, the knot $t_{k_j}^j$ will be so close to t_- and $t_{k_j+p}^j$ so close to t_+ that the two averages $\bar{t}_{k_{j-1}}^j$ and $\bar{t}_{k_j}^j$ will both lie in (t_-, t_+) . Since $x_{j+1} \in (\bar{t}_{k_{j-1}}^j, \bar{t}_{k_j}^j]$, this means that $x_{j+1} \in (t_-, t_+)$. As a consequence, there are infinitely many knots in (t_-, t_+) . But this is impossible since for any given j only the knots $(t_{k_j+i}^j)_{i=1}^{p-1}$ can possibly lie in this interval. ■

Lemma 11.9. *Suppose $t_- < t_+$. Then there is an integer ℓ such that for all $j \geq \ell$ either $t_{k_j}^j, \dots, t_{k_j+p-1}^j \leq t_-$ and $t_{k_j+p}^j = t_+$ or $t_{k_j+1}^j, \dots, t_{k_j+p}^j \geq t_+$ and $t_{k_j}^j = t_-$.*

Proof. Let K denote the constant $K = (t_+ - t_-)/(p - 1) > 0$. From Lemma 11.6 we see that there is an ℓ such that $t_{k_j}^j > t_- - K$ and $t_{k_j+p}^j < t_+ + K$ for $j \geq \ell$. If the lemma was not true, it is easy to check that $\bar{t}_{k_{j-1}}^j$ and $\bar{t}_{k_j}^j$ would have to lie in (t_-, t_+) and hence x_{j+1} would lie in (t_-, t_+) which contradicts the previous lemma. ■

Lemma 11.10. *Suppose that $t_- < t_+$. Then the zero sequence (x_j) and the sequences of interior active knots $(t_{k_j+1}^j), \dots, (t_{k_j+p-1}^j)$ all converge and one of the following is true: Either all the sequences converge to t_- and $x_j \leq t_-$ for j larger than some ℓ , or all the sequences converge to t_+ and $x_j \geq t_+$ for all j larger than some ℓ .*

Proof. We consider the two situations described in Lemma 11.9 in turn. Suppose that $t_{k_j}^j, \dots, t_{k_j+p-1}^j \leq t_-$ for $j \geq \ell$. This means that $\bar{t}_{k_j}^j < t_+$ and since x_{j+1} cannot lie in (t_-, t_+) , we must have $x_{j+1} \leq t_-$ for $j \geq \ell$. Since no new knots can appear to the right of t_+ we must have $t_{k_j+p}^j = t_+$ for $j \geq \ell$. Moreover, since $t_{k_j}^j < x_{j+1} \leq t_-$, we conclude that (x_j) and all the sequences of interior active knots converge to t_- . The proof for the case $t_{k_j+1}^j, \dots, t_{k_j+p}^j \geq t_+$ is similar. ■

Lemma 11.10 completes the proof of Proposition 11.7. It remains to show that the limit of a zero sequence is a zero of f .

Lemma 11.11. *Any accumulation point of a zero sequence is a zero of f .*

Proof. Let z be an accumulation point for a zero sequence (x_j) , and let ϵ be any positive, real number. Then there must be positive integers ℓ and k such that $t_{k+1}^\ell, \dots, t_{k+p}^\ell$ and $x_{\ell+1}$ all lie in the interval $(z - \epsilon/2, z + \epsilon/2)$. Let $\bar{t} = \bar{t}_k^\ell$ and let $\Gamma = \Gamma_{t^\ell}(f)$ be the control polygon of f in $\mathbb{S}_{p, t^\ell}$. We know that the derivative $f' = \sum_i \Delta c_i B_{i, p-1, t^\ell}$ is a spline in $\mathbb{S}_{p-1, t^\ell}$, and from Theorem 9.15 it follows that $\|(\Delta c_i)\|_\infty \leq K_{p-1} \|f'\|_\infty$ for some constant K_{p-1} depending only on p . From this we note that for any real numbers x and y we have the inequalities

$$|\Gamma(x) - \Gamma(y)| \leq \int_x^y |\Gamma'(t)| dt \leq \|(\Delta c_i)\|_\infty |y - x| \leq K_{p-1} \|f'\|_\infty |y - x|.$$

In particular, since $\Gamma(x_{\ell+1}) = 0$ it follows that

$$|\Gamma(\bar{t})| = |\Gamma(\bar{t}) - \Gamma(x_{\ell+1})| \leq K_{p-1} \|f'\|_\infty \epsilon.$$

In addition it follows from Theorem 9.18 that

$$|f(\bar{t}) - \Gamma(\bar{t})| \leq C(t_{k+p}^\ell - t_{k+1}^\ell)^2 \leq C\epsilon^2,$$

where C is another constant depending on f and p , but not on t^ℓ . Combining these estimates we obtain

$$\begin{aligned} |f(z)| &\leq |f(z) - f(\bar{t})| + |f(\bar{t}) - \Gamma(\bar{t})| + |\Gamma(\bar{t})| \\ &\leq \|f'\|_\infty \epsilon + C\epsilon^2 + K_{p-1} \|f'\|_\infty \epsilon. \end{aligned}$$

Since this is valid for any positive value of ϵ we must have $f(z) = 0$. ■

Lemmas 11.6, 11.10 and 11.11 lead to our main result.

Theorem 11.12. *A zero sequence converges to a zero of f .*

Recall that the zero finding algorithm does not need a starting value and there are no conditions in Theorem 11.12. On the other hand all control polygons of a spline with a zero must have at least one zero. For such splines the algorithm is therefore unconditionally convergent (for splines without zeros the algorithm will detect that the spline is of one sign in a finite number of steps).

11.4 Rate of convergence

In the previous section we saw that algorithm 11.3 always converges to the first zero of f , or halts in a finite number of steps if there are no zeros. In this section we will state two results about how quickly the algorithm converges to a simple zero. The proofs are based on Taylor expansions of the blossom of f and are omitted here. The first result is the most general one.

Theorem 11.13. *Let (x_j) be a zero sequence converging to a zero z and suppose that $f'(z) \neq 0$. Then there is a constant C depending on f and p but not on t , such that for sufficiently large j we have*

$$|x_{j+1} - z| \leq C \max_{i=1, \dots, p-1} |t_{k_j+i}^j - z|^2.$$

Recall that the knots $(t_{k_j+i}^j)_{i=1}^{p-1}$ typically correspond to recently inserted knots, i.e., recent estimates of the zero. The result therefore shows that the algorithm has a kind of quadratic convergence to simple zeros. This can be made more explicit in case a zero sequence (x_j) converges monotonically to z .

Theorem 11.14. *Let (x_j) be a zero sequence converging to a zero z , suppose that $f'(z)$ and $f''(z)$ are both nonzero, and set $e_n = |x_n - z|$. Then there is a constant C depending on f and p , but not on t , such that for sufficiently large j we have*

$$e_{j+1} \leq C e_{j-p+2}^2.$$

This result implies that if we sample the error in Algorithm 11.3 after every $p-1$ knot insertions, the resulting sequence converges quadratically to zero. Insertion of $p-1$ knots is roughly equivalent to computation of a function value, so Theorem 11.14 roughly says that for each computation of a function value the error is squared. This is similar to Newton's method which also converges quadratically and requires computation of $f(x)$ and $f'(x)$ during each iteration.

11.5 Stability

In this section we briefly discuss the stability properties of Algorithm 11.3. It is well known that a situation where large rounding errors may occur is when a small value is computed from relatively large values. Computing zeros of functions fall in this category as we need to compute values of the function near the zero, while the function is usually described by reasonably large parameters. For example, spline functions are usually given by reasonably large values of the knots and B-spline coefficients, but near a zero these numbers combine such that the result is small. It is therefore particularly important to keep an eye on rounding errors when computing zeros.

Our method consists of two parts where rounding errors potentially may cause problems, namely the computation of x_{j+1} by the first step in Algorithm 11.3 and the computation of the new B-spline coefficients in step 2. Let us consider each of these steps in turn.

The new estimate for the zero is given by the formula

$$x_{j+1} = \bar{t}_{k_j}^j - \frac{c_{k_j}^j (t_{k_j+p}^j - t_{k_j}^j)}{(c_{k_j}^j - c_{k_{j-1}}^j)p},$$

which is in fact a convex combination of the two numbers $\bar{t}_{k_{j-1}}^j$ and $\bar{t}_{k_j}^j$, see (11.4). Recall that $c_{k_{j-1}}^j$ and $c_{k_j}^j$ have opposite signs while $t_{k_j}^j$ and $t_{k_j+p}^j$ are usually well separated so the second term on the right can usually be computed without much cancellation. The total estimate x_{j+1} is then inserted as a new knot, and new coefficients are computed via (11.1) as a series of convex combinations. Convex combinations are generally well suited to floating-point computations except when combining two numbers of opposite signs to obtain a number near zero. This can potentially happen when computing the new coefficient

$$c_k^{j+1} = (1 - \mu_k) c_{k-1}^j + \mu_k c_k^j,$$

since we know that c_{k-1}^j and c_k^j have opposite signs. However, it can be shown that in most cases, the magnitude of one of the two coefficients tends to zero with j whereas the other one remains bounded away from zero. This ensures that the most critical convex combination usually behaves nicely, so in most cases there should not be problems with numerical stability, which corresponds well with practical experience. However, as with Newton's method and many others, we must expect the numerical performance to deteriorate when $f'(z)$ becomes small.

11.6 Implementation and numerical examples

Algorithm is very simple to implement and does not require any elaborate spline software. To illustrate this fact we provide pseudo code for an algorithm to compute the smallest zero of a spline, returned in the variable x . The knots t and the coefficients c are stored in vectors (indexed from 1). For efficiency the algorithm overwrites the old coefficients with the new ones during knot insertion.

Pseudo code for Algorithm 11.1

```
// Connected spline of degree d
// with knots t and coefficients c given
if (c(1)==0) return t(1);
k=2;
for (it = 1; it<=max_iterations; it++) {

    // Compute the index of the smallest zero
    // of the control polygon
```

```

n = size(c);
while (k<=n AND c(k-1)*c(k)>0) k++;
if (k>n) return NO_ZERO;

// Find zero of control polygon and check convergence
x = knotAverage(t,p,k)
    - c(k) * (t(k+p)-t(k))/(c(k)-c(k-1))/p;
xlist.append(x);
if ( converged(t,p,xlist) ) return x;

// Refine spline by Boehms algorithm
mu = k;
while (x>t(mu+1)) mu++;
c.append(c(n)); //Length of c increased by one
for (i=n; i>=mu+1; i--) c(i) = c(i-1);
for (i=mu ; i>=mu-p+1; i--) {
    alpha = (x-t(i))/(t(i+p)-t(i));
    c(i) = (1-alpha)*c(i-1) + alpha*c(i);
}
t.insert(mu+1,x);
}
// Max_iterations too small for convergence

```

This code will return an approximation to the leftmost root of the spline unless the total number of allowed iterations `max_iterations` is too low (or the tolerance is too small, see below). Note that it is assumed that the spline is connected. In particular, this means that the first coefficient must be nonzero.

The function `converged` returns true when the last inserted knot x equals t_{k+p} (in which case the spline has become disconnected at x , see Lemma 11.4), or when the sequence of computed zeros of the control polygons are deemed to converge in a traditional sense. Our specific criterion for convergence is (after at least p knots have been inserted),

$$\frac{\max_{i,j} |x_i - x_j|}{\max(|t_k|, |t_{k+p}|)} < \epsilon,$$

where the maximum is taken over the p last inserted knots and $\epsilon > 0$ is a small user-defined constant. This expression measures the relative difference of the last p knots and $\epsilon = 10^{-15}$ is a good choice when the computations are performed in double precision arithmetic.

In principle, our method should always converge, so there should be no need for a bound on the number of iterations. However, this is always a good safety net, as long as the maximum number of iterations is chosen as a sufficiently big integer.

There is of course a similar algorithm for computing the largest zero. If one needs to compute all zeros of a spline, this can be done sequentially by first computing the smallest zero, split the spline at that point, compute the second smallest zero and so on. Alternatively,

the computations can be done in parallel by inserting all the zeros of the control polygon in each iteration. We leave the details to the interested reader.

Before considering some examples and comparing our method with other methods, we need to have a rough idea of the complexity of the method. To determine the correct segment of the control polygon requires a search the first time, thereafter choosing the right segment only involves one comparison. Computing the new estimate for the zero is also very quick as it only involves one statement. What takes time is computing the new B-spline coefficients after the new knot has been inserted. This usually requires p convex combinations. As we saw above, we often have quadratic convergence if we sample the error every $p - 1$ iterations, and the work involved in $p - 1$ knot insertions is $p(p - 1)$ convex combinations.

We have estimated the errors $e_j = |x_j - z|$ of our algorithm for the four examples shown in Figure 11.2. The first three have simple roots, while the last has a double root. In Table 11.6 we have compared the errors produced by our method with those produced by Newton's method and a Bezier based method suggested by Rockwood. We have compared every $p - 1$ th step in our method with every iteration of the other methods. Quadratic convergence is confirmed for all the methods for the first three examples, whereas all methods only have linear convergence for the double zero (as for Newton's method, we have observed higher than second order convergence in cases with $f'(z) \neq 0$ and $f''(z) = 0$.)

We used the smallest zero of the control polygon as starting value for Newton's method. Note that it is not hard to find examples where this starting point will make Newton's method diverge. When using Newton's method to compute zeros of spline functions we have to evaluate $f(x)$ and $f'(x)$ in every step. With careful coding, this requires essentially the same number of operations as inserting p knots at a single point when f is a spline, or roughly $p(p - 1)/2$ convex combinations. On the other hand, Rockwood's method is based on inserting p knots at the zeros of the control polygon, in effect splitting the curve into two pieces by Bézier subdivision. The complexity of this is the same as for Newton's method.

Example	Method	E_0	E_1	E_2	E_3	E_4	E_5
Cubic Bézier Simple root	Alg.11.3	1.41e-1	1.31e-2	1.46e-4	1.70e-8	2.30e-16	4.22e-32
	Rockwood	1.41e-1	2.05e-2	8.71e-4	1.80e-6	7.72e-12	1.42e-22
	Newton	1.41e-1	2.05e-2	8.71e-4	1.80e-6	7.72e-12	1.42e-22
Quintic Simple root	Alg.11.3	1.34e-1	3.06e-3	1.24e-6	1.53e-13	2.21e-27	4.54e-55
	Rockwood	1.34e-1	5.47e-3	2.72e-5	6.87e-10	4.39e-19	1.80e-37
	Newton	1.34e-1	5.06e-3	2.33e-5	5.04e-10	2.36e-19	5.21e-38
Degree 25 spline Simple root	Alg.11.3	3.79e-3	2.64e-5	7.72e-11	2.60e-22	1.48e-45	5.10e-71
	Rockwood	3.79e-3	3.50e-5	5.49e-9	1.35e-16	8.22e-32	8.22e-32
	Newton	3.79e-3	7.64e-5	2.62e-8	3.08e-15	4.25e-29	8.13e-57
Cubic polynomial Double root	Alg.11.3	5.19e-1	2.68e-1	1.37e-1	6.95e-2	3.52e-2	1.78e-2
	Rockwood	5.19e-1	3.34e-1	2.23e-1	1.49e-1	9.90e-2	6.60e-2
	Newton	5.19e-1	3.46e-1	2.31e-1	1.54e-1	1.03e-1	6.84e-2

Table 11.1. The absolute errors $|x_j - z|$ for our method (inserting $p - 1$ knots), Rockwoods method and Newtons method for the three examples. The computations have been performed in extended arithmetic in order to include more iterations.

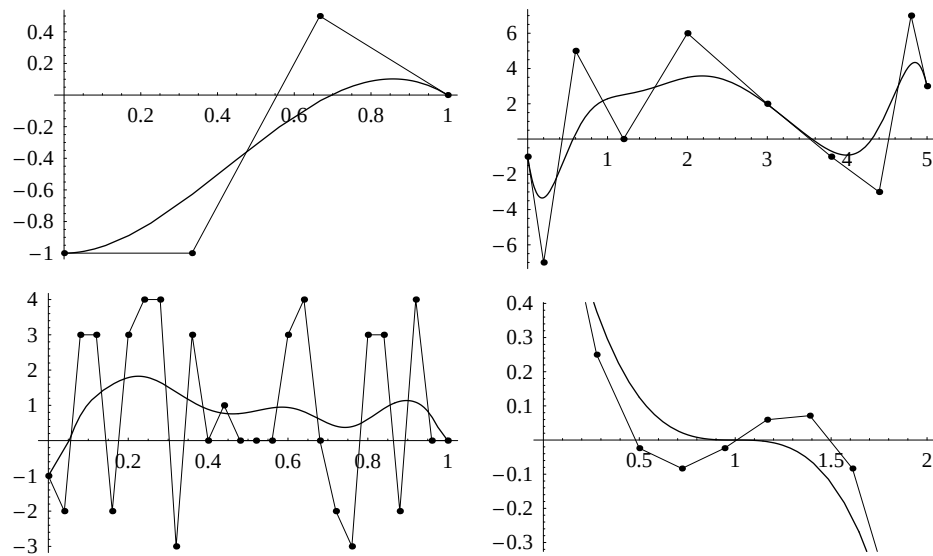


Figure 11.2. Our test examples in reading order: cubic Bézier function, quintic spline, degree 25 spline, cubic polynomial with a double root.

Appendix A

Some Linear Algebra

A.1 Matrices

The collection of m, n matrices

$$A = \begin{pmatrix} a_{1,1} & \cdots & a_{1,n} \\ \cdots & & \cdots \\ a_{m,1} & \cdots & a_{m,n} \end{pmatrix}$$

with real elements $a_{i,j}$ is denoted by $\mathbb{R}^{m,n}$. If $n = 1$ then A is called a column vector. Similarly, if $m = 1$ then A is a row vector. We let $\mathbb{R}^m$ denote the collection of all column or row vectors with m real components.

A.1.1 Nonsingular matrices, and inverses.

Definition A.1. A collection of vectors $\mathbf{a}_1, \dots, \mathbf{a}_n \in \mathbb{R}^m$ is linearly independent if $x_1 \mathbf{a}_1 + \cdots + x_n \mathbf{a}_n = \mathbf{0}$ for some real numbers $x_1, \dots, x_n$, implies that $x_1 = \cdots = x_n = 0$.

Suppose $\mathbf{a}_1, \dots, \mathbf{a}_n$ are the columns of a matrix $A \in \mathbb{R}^{m,n}$. For a vector $\mathbf{x} = (x_1, \dots, x_n)^T \in \mathbb{R}^n$ we have $A\mathbf{x} = \sum_{j=1}^n x_j \mathbf{a}_j$. It follows that the collection $\mathbf{a}_1, \dots, \mathbf{a}_n$ is linearly independent if and only if $A\mathbf{x} = \mathbf{0}$ implies $\mathbf{x} = \mathbf{0}$.

Definition A.2. A square matrix A such that $A\mathbf{x} = \mathbf{0}$ implies $\mathbf{x} = \mathbf{0}$ is said to be nonsingular.

Definition A.3. A square matrix $A \in \mathbb{R}^{n,n}$ is said to be invertible if for some $B \in \mathbb{R}^{n,n}$

$$BA = AB = I,$$

where $I \in \mathbb{R}^{n,n}$ is the identity matrix.

An invertible matrix A has a unique inverse $B = A^{-1}$. If A, B , and C are square matrices, and $A = BC$, then A is invertible if and only if both B and C are also invertible. Moreover, the inverse of A is the product of the inverses of B and C in reverse order, $A^{-1} = C^{-1}B^{-1}$.

A.1.2 Determinants.

The determinant of a square matrix A will be denoted $\det(A)$ or

$$\begin{vmatrix} a_{1,1} & \cdots & a_{1,n} \\ \vdots & & \vdots \\ a_{n,1} & \cdots & a_{n,n} \end{vmatrix}.$$

Recall that the determinant of a 2×2 matrix is

$$\begin{vmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \end{vmatrix} = a_{1,1}a_{2,2} - a_{1,2}a_{2,1}.$$

A.1.3 Criteria for nonsingularity and singularity.

We state without proof the following criteria for nonsingularity.

Theorem A.4. *The following is equivalent for a square matrix $A \in \mathbb{R}^{n,n}$.*

1. A is nonsingular.
2. A is invertible.
3. $A\mathbf{x} = \mathbf{b}$ has a unique solution $\mathbf{x} = A^{-1}\mathbf{b}$ for any $\mathbf{b} \in \mathbb{R}^n$.
4. A has linearly independent columns.
5. A^T is nonsingular.
6. A has linearly independent rows.
7. $\det(A) \neq 0$.

We also have a number of criteria for a matrix to be singular.

Theorem A.5. *The following is equivalent for a square matrix $A \in \mathbb{R}^{n,n}$.*

1. There is a nonzero $\mathbf{x} \in \mathbb{R}^n$ so that $A\mathbf{x} = \mathbf{0}$.
2. A has no inverse.
3. $A\mathbf{x} = \mathbf{b}$ has either no solution or an infinite number of solutions.
4. A has linearly dependent columns.
5. There is a nonzero $\mathbf{x}$ so that $\mathbf{x}^T A = \mathbf{0}$.
6. A has linearly dependent rows.
7. $\det(A) = 0$.

Corollary A.6. *A matrix with more columns than rows has linearly dependent columns.*

Proof. Suppose $A \in \mathbb{R}^{m,n}$ with $n > m$. By adding $n - m$ rows of zeros to A we obtain a square matrix $B \in \mathbb{R}^{n,n}$. This matrix has linearly dependent rows. By Theorem A.4 the matrix B has linearly dependent columns. But then the columns of A are also linearly dependent. ■

A.2 Vector Norms

Formally, a *vector norm* $\|\cdot\| = \|\mathbf{x}\|$, is a function $\|\cdot\| : \mathbb{R}^n \rightarrow [0, \infty)$ that satisfies for $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ the following properties

1. $\|\mathbf{x}\| = 0$ implies $\mathbf{x} = \mathbf{0}$.
 2. $\|\alpha\mathbf{x}\| = |\alpha|\|\mathbf{x}\|$.
 3. $\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\|$.
- (A.1)

Property 3 is known as the *Triangle Inequality*. For us the most useful class of norms are the p or ℓ^p norms. They are defined for $p \geq 1$ and $\mathbf{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ by

$$\begin{aligned}\|\mathbf{x}\|_p &= (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p}, \\ \|\mathbf{x}\|_\infty &= \max_i |x_i|.\end{aligned}$$
(A.2)

Since

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_p \leq n^{1/p} \|\mathbf{x}\|_\infty, \quad p \geq 1$$
(A.3)

and $\lim_{p \rightarrow \infty} n^{1/p} = 1$ for any $n \in \mathbb{N}$ we see that $\lim_{p \rightarrow \infty} \|\mathbf{x}\|_p = \|\mathbf{x}\|_\infty$.

The 1, 2, and ∞ norms are the most important. We have

$$\|\mathbf{x}\|_2^2 = x_1^2 + \dots + x_n^2 = \mathbf{x}^T \mathbf{x}.$$
(A.4)

Lemma A.7 (The Hölder inequality). *We have for $1 \leq p \leq \infty$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*

$$\sum_{i=1}^n |x_i y_i| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q, \quad \text{where } \frac{1}{p} + \frac{1}{q} = 1.$$
(A.5)

Proof. We base the proof on properties of the exponential function. Recall that the exponential function is convex, i.e. with $f(x) = e^x$ we have the inequality

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$
(A.6)

for every $\lambda \in [0, 1]$ and $x, y \in \mathbb{R}$.

If $\mathbf{x} = \mathbf{0}$ or $\mathbf{y} = \mathbf{0}$, there is nothing to prove. Suppose $\mathbf{x}, \mathbf{y} \neq \mathbf{0}$. Define $\mathbf{u} = \mathbf{x}/\|\mathbf{x}\|_p$ and $\mathbf{v} = \mathbf{y}/\|\mathbf{y}\|_q$. Then $\|\mathbf{u}\|_p = \|\mathbf{v}\|_q = 1$. If we can prove that $\sum_i |u_i v_i| \leq 1$, we are done because then $\sum_i |x_i y_i| = \|\mathbf{x}\|_p \|\mathbf{y}\|_q \sum_i |u_i v_i| \leq \|\mathbf{x}\|_p \|\mathbf{y}\|_q$. Since $|u_i v_i| = |u_i| |v_i|$, we can assume that $u_i \geq 0$ and $v_i \geq 0$. Moreover, we can assume that $u_i > 0$ and $v_i > 0$ because a zero term contributes no more to the left hand side than to the right hand side of (A.5). Let s_i, t_i be such that $u_i = e^{s_i/p}$, $v_i = e^{t_i/q}$. Taking $f(x) = e^x$, $\lambda = 1/p$, $1 - \lambda = 1/q$, $x = s_i$ and $y = t_i$ in (A.6) we find

$$e^{s_i/p + t_i/q} \leq \frac{1}{p} e^{s_i} + \frac{1}{q} e^{t_i}.$$

But then

$$\sum_i |u_i v_i| = \sum_i e^{s_i/p + t_i/q} \leq \frac{1}{p} \sum_i e^{s_i} + \frac{1}{q} \sum_i e^{t_i} = \frac{1}{p} \sum_i u_i^p + \frac{1}{q} \sum_i v_i^q = \frac{1}{p} + \frac{1}{q} = 1.$$

This completes the proof of (A.5). ■

When $p = 2$ then $q = 2$ and the Hölder inequality is associated with the names Buniakowski-Cauchy-Schwarz.

Lemma A.8 (The Minkowski inequality). *We have for $1 \leq p \leq \infty$ and $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$*

$$\|\mathbf{x} + \mathbf{y}\|_p \leq \|\mathbf{x}\|_p + \|\mathbf{y}\|_p. \quad (\text{A.7})$$

Proof. Let $\mathbf{u} = (u_1, \dots, u_n)$ with $u_i = |x_i + y_i|^{p-1}$. Since $q(p-1) = p$ and $p/q = p-1$, we find

$$\|\mathbf{u}\|_q = \left(\sum_i |x_i + y_i|^{q(p-1)} \right)^{1/q} = \left(\sum_i |x_i + y_i|^p \right)^{1/q} = \|\mathbf{x} + \mathbf{y}\|_p^{p/q} = \|\mathbf{x} + \mathbf{y}\|_p^{p-1}.$$

Using this and the Hölder inequality we obtain

$$\begin{aligned} \|\mathbf{x} + \mathbf{y}\|_p^p &= \sum_i |x_i + y_i|^p \leq \sum_i |u_i| |x_i| + \sum_i |u_i| |y_i| \leq (\|\mathbf{x}\|_p + \|\mathbf{y}\|_p) \|\mathbf{u}\|_q \\ &\leq (\|\mathbf{x}\|_p + \|\mathbf{y}\|_p) \|\mathbf{x} + \mathbf{y}\|_p^{p-1}. \end{aligned}$$

Dividing by $\|\mathbf{x} + \mathbf{y}\|_p^{p-1}$ proves Minkowski. ■

Using the Minkowski inequality it follows that the p norms satisfies the axioms for a vector norm.

In (A.3) we established the inequality

$$\|\mathbf{x}\|_\infty \leq \|\mathbf{x}\|_p \leq n^{1/p} \|\mathbf{x}\|_\infty, \quad p \geq 1.$$

More generally, we say that two vector norms $\|\cdot\|$ and $\|\cdot\|'$ are *equivalent* if there exists positive constants μ and M such that

$$\mu \|\mathbf{x}\| \leq \|\mathbf{x}\|' \leq M \|\mathbf{x}\| \quad (\text{A.8})$$

for all $\mathbf{x} \in \mathbb{R}^n$.

Theorem A.9. *All vector norms on $\mathbb{R}^n$ are equivalent.*

Proof. It is enough to show that a vector norm $\|\cdot\|$ is equivalent to the l_∞ norm, $\|\cdot\|_\infty$. Let $\mathbf{x} \in \mathbb{R}^n$ and let $\mathbf{e}_i, i = 1, \dots, n$ be the unit vectors in $\mathbb{R}^n$. Writing $\mathbf{x} = x_1 \mathbf{e}_1 + \dots + x_n \mathbf{e}_n$ we have

$$\|\mathbf{x}\| \leq \sum_i |x_i| \|\mathbf{e}_i\| \leq \|\mathbf{x}\|_\infty M, \quad M = \sum_i \|\mathbf{e}_i\|.$$

To find $\mu > 0$ such that $\|\mathbf{x}\| \geq \mu \|\mathbf{x}\|_\infty$ for all $\mathbf{x} \in \mathbb{R}^n$ is less elementary. Consider the function f given by $f(\mathbf{x}) = \|\mathbf{x}\|$ defined on the l_∞ “unit ball”

$$S = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_\infty = 1\}.$$

S is a closed and bounded set. From the inverse triangle inequality

$$|\|\mathbf{x}\| - \|\mathbf{y}\|| \leq \|\mathbf{x} - \mathbf{y}\|, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^n.$$

it follows that f is continuous on S . But then f attains its maximum and minimum on S , i.e. there is a point $\mathbf{x}^* \in S$ such that

$$\|\mathbf{x}^*\| = \min_{\mathbf{x} \in S} \|\mathbf{x}\|.$$

Moreover, since $\mathbf{x}^*$ is nonzero we have $\mu := \|\mathbf{x}^*\| > 0$. If $\mathbf{x} \in \mathbb{R}^n$ is nonzero then $\mathbf{x} = \mathbf{x}/\|\mathbf{x}\|_\infty \in S$. Thus

$$\mu \leq \|\mathbf{x}\| = \left\| \frac{\mathbf{x}}{\|\mathbf{x}\|_\infty} \right\| = \frac{1}{\|\mathbf{x}\|_\infty} \|\mathbf{x}\|,$$

and this establishes the lower inequality. ■

It can be shown that for the p norms we have for any q with $1 \leq q \leq p \leq \infty$

$$\|\mathbf{x}\|_p \leq \|\mathbf{x}\|_q \leq n^{1/q-1/p} \|\mathbf{x}\|_p, \quad \mathbf{x} \in \mathbb{R}^n. \quad (\text{A.9})$$

<

A.3 Vector spaces of functions

In $\mathbb{R}^m$ we have the operations $\mathbf{x} + \mathbf{y}$ and $a\mathbf{x}$ of vector addition and multiplication by a scalar $a \in \mathbb{R}$. Such operations can also be defined for functions. As an example, if $f(x) = x$, $g(x) = 1$, and a, b are real numbers then $af(x) + bg(x) = ax + b$. In general, if f and g are two functions defined on the same set I and $a \in \mathbb{R}$, then the sum $f + g$ and the product af are functions defined on I by

$$\begin{aligned} (f + g)(x) &= f(x) + g(x), \\ (af)(x) &= af(x). \end{aligned}$$

Two functions f and g defined on I are equal if $f(x) = g(x)$ for all $x \in I$. We say that f is the zero function, i.e. $f = 0$, if $f(x) = 0$ for all $x \in I$.

Definition A.10. Suppose S is a collection of real valued or vector valued functions, all defined on the same set I . The collection S is called a vector space if $af + bg \in S$ for all $f, g \in S$ and all $a, b \in \mathbb{R}$. A subset T of S is called a subspace of S if T itself is a vector space.

Example A.11. Vector spaces

- All polynomials π_d of degree at most d .
- All polynomials of all degrees.
- All trigonometric polynomials $a_0 + \sum_{k=1}^d (a_k \cos kx + b_k \sin kx)$ of degree at most d .
- The set $C(I)$ of all continuous real valued functions defined on I .
- The set $C^r(I)$ of all real valued functions defined on I with continuous j' th derivative for $j = 0, 1, \dots, r$.

Definition A.12. A vector space S is said to be finite dimensional if

$$S = \text{span}(\phi_1, \dots, \phi_n) = \left\{ \sum_{j=1}^n c_j \phi_j : c_j \in \mathbb{R} \right\},$$

for a finite number of functions $\phi_1, \dots, \phi_n$ in S . The functions $\phi_1, \dots, \phi_n$ are said to span or generate S .

Of the examples above the space $\pi_d = \text{span}(1, x, x^2, \dots, x^d)$ generated by the monomials $1, x, x^2, \dots, x^d$ is finite dimensional. Also the trigonometric polynomials are finite dimensional. The space of all polynomials of all degrees is not finite dimensional. To see this we observe that any finite set cannot generate the monomial x^{d+1} where d is the maximal degree of the elements in the spanning set. Finally we observe that $C(I)$ and $C^r(I)$ contain the space of polynomials of all degrees as a subspace. Hence they are not finite dimensional,

If $f \in S = \text{span}(\phi_1, \dots, \phi_n)$ then $f = \sum_{j=1}^n c_j \phi_j$ for some $\mathbf{c} = (c_1, \dots, c_n)$. With $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)^T$ we will often use the vector notation

$$f(x) = \boldsymbol{\phi}(x)^T \mathbf{c} \quad (\text{A.10})$$

for f .

A.3.1 Linear independence and bases

All vector spaces in this section will be finite dimensional.

Definition A.13. A set of functions $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)^T$ in a vector space S is said to be linearly independent on a subset J of I if $\boldsymbol{\phi}(x)^T \mathbf{c} = c_1 \phi_1(x) + \dots + c_n \phi_n(x) = 0$ for all $x \in J$ implies that $\mathbf{c} = \mathbf{0}$. If $J = I$ then we simply say that $\boldsymbol{\phi}$ is linearly independent.

If $\boldsymbol{\phi}$ is linearly independent then the representation in (A.10) is unique. For if $f = \boldsymbol{\phi}^T \mathbf{c} = \boldsymbol{\phi}^T \mathbf{b}$ for some $\mathbf{c}, \mathbf{b} \in \mathbb{R}^n$ then $f = \boldsymbol{\phi}^T (\mathbf{c} - \mathbf{b}) = 0$. Since $\boldsymbol{\phi}$ is linearly independent we have $\mathbf{c} - \mathbf{b} = \mathbf{0}$, or $\mathbf{c} = \mathbf{b}$.

Definition A.14. A set of functions $\boldsymbol{\phi}^T = (\phi_1, \dots, \phi_n)$ in a vector space S is a basis for S if the following two conditions hold

1. $\boldsymbol{\phi}$ is linearly independent.
2. $S = \text{span}(\boldsymbol{\phi})$.

Theorem A.15. The monomials $1, x, x^2, \dots, x^d$ are linearly independent on any set $J \subset \mathbb{R}$ containing at least $d+1$ distinct points. In particular these functions form a basis for π_d .

Proof. Let $x_0, \dots, x_d$ be $d+1$ distinct points in J , and let $p(x) = c_0 + c_1 x + \dots + c_d x^d = 0$ for all $x \in J$. Then $p(x_i) = 0$, for $i = 0, 1, \dots, d$. Since a nonzero polynomial of degree d can have at most d zeros we conclude that p must be the zero polynomial. But then $c_k = p^{(k)}(0)/k! = 0$ for $k = 0, 1, \dots, d$. It follows that the monomial is a basis for π_d since they span π_d by definition. ■

To prove some basic results about bases in a vector space of functions it is convenient to introduce a matrix transforming one basis into another.

Lemma A.16. Suppose S and T are finite dimensional vector spaces with $S \subset T$, and let $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)^T$ be a basis for S and $\boldsymbol{\psi} = (\psi_1, \dots, \psi_m)^T$ a basis for T . Then

$$\boldsymbol{\phi} = \mathbf{A}^T \boldsymbol{\psi}, \quad (\text{A.11})$$

for some matrix $\mathbf{A} \in \mathbb{R}^{m,n}$. If $f = \boldsymbol{\phi}^T \mathbf{c} \in S$ is given then $f = \boldsymbol{\psi}^T \mathbf{b}$ with

$$\mathbf{b} = \mathbf{A} \mathbf{c}. \quad (\text{A.12})$$

Moreover $\mathbf{A}$ has linearly independent columns.

Proof. Since $\phi_j \in T$ there are real numbers $a_{i,j}$ such that

$$\phi_j = \sum_{i=1}^m a_{i,j} \psi_i, \quad \text{for } j = 1, \dots, n,$$

This equation is simply the component version of (A.11). If $f \in S$ then $f \in T$ and $f = \boldsymbol{\psi}^T \mathbf{b}$ for some $\mathbf{b}$. By (A.11) we have $\boldsymbol{\phi}^T = \boldsymbol{\psi}^T \mathbf{A}$ and $f = \boldsymbol{\phi}^T \mathbf{c} = \boldsymbol{\psi}^T \mathbf{A} \mathbf{c}$ or $\boldsymbol{\psi}^T \mathbf{b} = \boldsymbol{\psi}^T \mathbf{A} \mathbf{c}$. Since $\boldsymbol{\psi}$ is linearly independent we get (A.12). Finally, to show that $\mathbf{A}$ has linearly independent columns suppose $\mathbf{A} \mathbf{c} = \mathbf{0}$. Define $f \in S$ by $f = \boldsymbol{\phi}^T \mathbf{c}$. By (A.11) we have $f = \boldsymbol{\psi}^T \mathbf{A} \mathbf{c} = \mathbf{0}$. But then $f = \boldsymbol{\phi}^T \mathbf{c} = \mathbf{0}$. Since $\boldsymbol{\phi}$ is linearly independent we conclude that $\mathbf{c} = \mathbf{0}$. ■

The matrix $\mathbf{A}$ in Lemma A.16 is called a *change of basis matrix*.

A basis for a vector space generated by n functions can have at most n elements.

Lemma A.17. If $\boldsymbol{\psi} = (\psi_1, \dots, \psi_k)^T$ is a linearly independent set in a vector space $S = \text{span}(\phi_1, \dots, \phi_n)$, then $k \leq n$.

Proof. With $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)^T$ we have

$$\boldsymbol{\psi} = \mathbf{A}^T \boldsymbol{\phi}, \quad \text{for some } \mathbf{A} \in \mathbb{R}^{n,k}.$$

If $k > n$ then $\mathbf{A}$ is a rectangular matrix with more columns than rows. From Corollary A.6 we know that the columns of such a matrix must be linearly dependent; i.e. there is some nonzero $\mathbf{c} \in \mathbb{R}^k$ such that $\mathbf{A} \mathbf{c} = \mathbf{0}$. But then $\boldsymbol{\psi}^T \mathbf{c} = \boldsymbol{\phi}^T \mathbf{A} \mathbf{c} = \mathbf{0}$, for some nonzero $\mathbf{c}$. This implies that $\boldsymbol{\psi}$ is linearly dependent, a contradiction. We conclude that $k \leq n$. ■

Lemma A.18. Every basis for a vector space must have the same number of elements.

Proof. Suppose $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)^T$ and $\boldsymbol{\psi} = (\psi_1, \dots, \psi_m)^T$ are two bases for the vector space. We need to show that $m = n$. Now

$$\boldsymbol{\phi} = \mathbf{A}^T \boldsymbol{\psi}, \quad \text{for some } \mathbf{A} \in \mathbb{R}^{m,n},$$

$$\boldsymbol{\psi} = \mathbf{B}^T \boldsymbol{\phi}, \quad \text{for some } \mathbf{B} \in \mathbb{R}^{n,m}.$$

By Lemma A.16 we know that both $\mathbf{A}$ and $\mathbf{B}$ have linearly independent columns. But then by Corollary A.6 we see that $m = n$. ■

Definition A.19. The number of elements in a basis in a vector space S is called the dimension of S , and is denoted by $\dim(S)$.

The following lemma shows that every set of linearly independent functions in a vector space S can be extended to a basis for S . In particular every finite dimensional vector space has a basis.

Lemma A.20. A set $\boldsymbol{\phi}^T = (\phi_1, \dots, \phi_k)$ of linearly independent elements in a finite dimensional vector space S , can be extended to a basis $\boldsymbol{\psi}^T = (\psi_1, \dots, \psi_m)$ for S .

Proof. Let $S_k = \text{span}(\psi_1, \dots, \psi_k)$ where $\psi_j = \phi_j$ for $j = 1, \dots, k$. If $S_k = S$ then we set $m = k$ and stop. Otherwise there must be an element $\psi_{k+1} \in S$ such that $\psi_1, \dots, \psi_{k+1}$ are linearly independent. We define a new vector space S_{k+1} by $S_{k+1} = \text{span}(\psi_1, \dots, \psi_{k+1})$. If $S_{k+1} = S$ then we set $m = k + 1$ and stop the process. Otherwise we continue to generate vector spaces $S_{k+2}, S_{k+3}, \dots$. Since S is finitely generated we must by Lemma A.17 eventually find some m such that $S_m = S$. ■

The following simple, but useful lemma, shows that a spanning set must be a basis if it contains the correct number of elements.

Lemma A.21. Suppose $S = \text{span}(\phi)$. If ϕ contains $\dim(S)$ elements then ϕ is a basis for S .

Proof. Let $n = \dim(S)$ and suppose $\phi = (\phi_1, \dots, \phi_n)$ is a linearly dependent set. Then there is one element, say ϕ_n which can be written as a linear combination of $\phi_1, \dots, \phi_{n-1}$. But then $S = \text{span}(\phi_1, \dots, \phi_{n-1})$ and $\dim(S) < n$ by Lemma A.17, a contradiction to the assumption that ϕ is linearly dependent. ■

A.4 Normed Vector Spaces

Suppose S is a vector space of functions. A *norm* $\| \cdot \| = \|f\|$, is a function $\| \cdot \| : S \rightarrow [0, \infty)$ that satisfies for $f, g \in S$, and $\alpha \in \mathbb{R}$ the following properties

1. $\|f\| = 0$ implies $f = 0$.
 2. $\|\alpha f\| = |\alpha| \|f\|$.
 3. $\|f + g\| \leq \|f\| + \|g\|$.
- (A.13)

Property 3 is known as the *Triangle Inequality*. The pair $(S, \| \cdot \|)$ is called a normed vector space (of functions).

In the rest of this section we assume that the functions in S are continuous, or at least piecewise continuous on some interval $[a, b]$.

Analogous to the p or ℓ^p norms for vectors in $\mathbb{R}^n$ we have the p or L^p norms for functions. They are defined for $1 \leq p \leq \infty$ and $f \in S$ by

$$\begin{aligned} \|f\|_p &= \|f\|_{L^p[a,b]} = \left(\int_a^b |f(x)|^p dx \right)^{1/p}, \quad p \geq 1, \\ \|f\|_\infty &= \|f\|_{L^\infty[a,b]} = \max_{a \leq x \leq b} |f(x)|. \end{aligned} \quad (\text{A.14})$$

The 1, 2, and ∞ norms are the most important.

We have for $1 \leq p \leq \infty$ and $f, g \in S$ the Hölder inequality

$$\int_a^b |f(x)g(x)| dx \leq \|f\|_p \|g\|_q, \quad \text{where } \frac{1}{p} + \frac{1}{q} = 1, \quad (\text{A.15})$$

and the Minkowski inequality

$$\|f + g\|_p \leq \|f\|_p + \|g\|_p. \quad (\text{A.16})$$

For $p = 2$ (A.15) is known as the Schwarz inequality, the Cauchy-Schwarz inequality, or the Buniakowski-Cauchy-Schwarz inequality.

Index

- affine combination, 4–6
- B-spline matrix, 38
 - differentiation, 57
- B-spline polynomials, 74
- B-splines, 17
 - $d + 1$ -extended knot vector, 54
 - $d + 1$ -regular knot vector, 54
 - Oslo1 algorithm, 82
 - algorithms for differentiation, 59
 - B-spline matrix and dual polynomials, 50, 75
 - basic properties, 30
 - basic spline, 19
 - basis for piecewise polynomials, 65
 - Bernstein polynomials, 32
 - cardinal, 31
 - change of basis, 72
 - coefficients, 33
 - commutation of B-spline matrices, 51, 57
 - complexity of algebraic form, 36
 - continuity, 61
 - definition, 27
 - definition of derivative, 56
 - dependence on knots, 29
 - derivative, 58
 - alternative recurrence relation, 58
 - recurrence relation, 58
 - derivative of any order, 58
 - detailed algorithms, 47
 - detailed version of R-algorithm, 47
 - determining μ , 46
 - differentiation, 57
 - algorithms, 59
 - DL algorithm, 59
 - DR algorithm, 59
 - explicit form of DL algorithm, 59
 - explicit form of DR algorithm, 61
 - freedom in choice of matrix, 58
 - left algorithm, 59
 - right algorithm, 59
 - discrete, 73, 82
 - division by zero convention, 27
 - dual polynomials, 49, 75
 - evaluation from the left, 40
 - explicit form, 38
 - explicit form of recurrence relation, 29
 - explicit notation, 29
 - extending the knot vector, 48
 - first derivative, 57, 58
 - formular for jump in a derivative, 62
 - integral, 63
 - knot averages, 34
 - knot insertion, 69, 72
 - spline subspaces, 72
 - knot insertion matrix
 - bandedness, 79
 - no knots inserted, 79
 - quadratic example, 80, 81
 - explicit formula, 78
 - freedom in determining, 77
 - knot refinement, 69
 - linear, 28
 - linear combination, 33
 - linear independence, 67

- $d + 1$ -extended knot vector, 54
 - local, 53
 - near ends of knot vector, 66
 - short knot vectors, 66
- linear space, 33
- local knots, 30
- local linear independence, 53
- local support, 30
- Marsden's identity, 52, 75
 - on one knot interval, 51
- Matlab algorithm, 47, 48
- matrix representation, 36, 56, 57
 - cubic case, 46
 - differentiation, 56
 - general degree, 38
 - linear case, 37
 - quadratic case, 37
- open knot vector, 54
- Oslo2 algorithm, 82
- piecewise polynomial, 30
- polynomial pieces, 31, 74, 76
- positivity within support, 30
- recurrence relation, 19, 27
 - for jumps, 67
 - for uniform knots, 32
- refined knot vector, 72
- refinement relation, 76
 - polynomial pieces, 76
- repeated application of B-spline matrices, 51
- representation of power basis, 52
- right algorithm, 40, 76
- second derivative, 57
- smoothness, 61
 - cubic case, 31
 - linear case, 28
 - quadratic case, 31, 55
- special values, 30, 43
- transformation matrix, 76
- transformation matrix
 - bandedness, 79
 - explicit formula, 78
 - freedom in determining, 77
 - no knots inserted, 79
 - quadratic example, 80, 81
 - translation invariance, 29
 - triangular algorithm, 59
 - complexity, 67
 - triangular algorithms, 42
 - trivially zero, 28
 - uniform knots, 31
 - vector algorithms, 47
 - vector version of R-algorithm, 47
- Bernstein basis, 12, 18, 24
 - partition of unity, 12
- Bernstein polynomial
 - general interval, 25
- Bernstein polynomials, 32
 - basic properties, 43
- Bézier curve, 11, 12
 - basis, 12, 24
 - Bernstein basis, 12, 18
 - composite, 13
 - control points, 12
 - control polygon, 12
 - cubic, 23
 - end conditions, 12
 - general interval, 25
 - on general interval, 24
 - partition of unity, 13
 - problems with high degree, 13
- Pierre Bézier, 11
- change of basis, 74
- Chebyshev polynomial, cubic, 45
- combination
 - affine, 4–6, 87
 - convex, 4, 5, 8–11, 13, 16, 20–22, 24, 89
- commutation of B-spline matrices, 51, 57
- complexity of triangular algorithms, 67
- composite Bézier curve
 - continuity, 14
 - smoothness, 14
- control points, 12, 14, 16
- control polygon, 12, 16, 69
- conversion between polynomial bases, 74
- convex hull, 4–7, 9, 10, 20, 21, 133

- of n points, 5
- convex set, 5, 20, 21
- C^r continuity, 55
- Curry-Schoenberg theorem, 65, 72
- curve
 - Bézier, 11, 12
 - interpolating polynomial, 9
 - spline, 11, 14
 - travelling along, 9
- detailed spline evaluation algorithms, 47
- differentiation of matrix product, 56, 67
- dimension of space of piecewise polynomials, 64
- discrete B-spline, 73
- dual polynomial of B-spline, 49
- elementary symmetric polynomials, 53
- floating point, 4
- function, 8
 - jump, 55
 - left derivative, 55
 - one-sided derivative, 55
 - piecewise continuous, 55
 - piecewise C^r , 55
 - piecewise smooth, 55
 - right derivative, 55
 - vector valued, 8
- hat function, 28
- interpolation
 - convex combinations, 9
 - cubic, 22
 - general degree, 23
 - Lagrange polynomials, 23
 - quadratic, 9
- jump in a function, 55
- knot averages, 34
- knot insertion, 69, 72
 - change of basis, 72
 - convergence of control polygon, 69, 70
 - example, 73
 - flexibility, 69
 - Oslo algorithms, 82
 - redundant knot, 70
 - redundant representation, 70
 - spline subspaces, 72
- knot insertion matrix, 73
 - bandedness, 79
 - no knots inserted, 79
 - quadratic example, 80, 81
 - explicit formula, 78
 - freedom in determining, 77
- knot refinement, 69
- knot sequence, 27
- knot vector, 14, 27
 - $d + 1$ -extended, 54
 - $d + 1$ -regular, 54
 - $d + 1$ -regular, 34
 - refinement, 72
 - subsequence, 72
- knots, 14
- left algorithm, 40
- linear independence, 53
- linear interpolation, 50
- Matlab algorithm, 47, 48
- multiple knots, 28
- numerical stability, 4, 18, 20, 109, 161, 181, 190–194, 201
- Oslo algorithms, 82
- parametric curve, 8
- parametric representation, 6
- parametric spline curve, 36
- piecewise polynomials, 64
 - truncated power basis, 64, 65
- polynomials
 - b-spline basis, 74
 - change of basis, 74
- proof by induction, 21, 29, 30, 43, 44, 61–63, 84, 85, 209, 210

- Renault, 11
- right algorithm, 40
- round-off error, 3, 4, 13, 20, 112, 113, 162, 190, 191
- S in Times Roman font, 15
- smoothness, 18
- spline curve, 11, 14, 36
 - control points, 14, 16, 36
 - control polygon, 16, 36
 - controlling smoothness, 18
 - definition, 36
 - end conditions, 17
 - finding B-splines, 19
 - interpolating at d -tuple knot, 35
 - knot vector, 14
 - knots, 14
 - linear, 14
 - Marsden's identity, 52
 - on one knot interval, 51
 - on one knot interval, 35
 - parametric, 36
 - quadratic, 16, 17
 - right algorithm, 40
 - smoothness, 17, 18
 - special case of Bézier curve, 16
 - undoing convex combinations, 19
 - unwrapping, 19
- spline function, 33
 - $d + 1$ -extended knot vector, 54
 - $d + 1$ -regular knot vector, 54
 - algorithms for evaluation, 39
 - B-spline matrix and dual polynomials, 50
 - basic properties, 35
 - change of basis, 72
 - commutation of B-spline matrices, 51
 - control points, 34
 - control polygon, 34, 69
 - convergence, 69, 70
 - convergence of control polygon, 69, 70
 - definition at right end of knot vector, 45
 - detailed algorithms, 47
 - determining μ , 46
 - differentiation, 59
 - DL algorithm, 59
 - DR algorithm, 59
 - explicit form of DL algorithm, 59
 - explicit form of DR algorithm, 61
 - left algorithm, 59
 - right algorithm, 59
 - evaluation from the right, 40
 - extending the knot vector, 48
 - generalizing algorithms to spline curves, 36
 - generating one polynomial piece on all of $\mathbb{R}$, 38
 - interpolating at d -tuple knot, 35
 - knot insertion, 69, 72
 - spline subspaces, 72
 - knot insertion matrix
 - bandedness, 79
 - no knots inserted, 79
 - quadratic example, 80
 - explicit formula, 78
 - freedom in determining, 77
 - knot refinement, 69
 - left algorithm, 40, 76
 - linear, 45
 - Matlab algorithm, 47, 48
 - matrix representation, 38
 - Oslo2 algorithm, 82
 - plotting, 72
 - refined knot vector, 72
 - repeated application of B-spline matrices, 51
 - representation of power basis, 52
 - smoothness at multiple knot, 35
 - triangular algorithm
 - complexity, 67
 - triangular algorithms, 42
 - vector algorithms, 47, 48
 - vector version of L-algorithm, 48
- spline space, 33
- spline vector function, 36
- tangent vector, 9
- triangular algorithms, 42
- truncated power basis, 65

truncated powers, 64

variation diminishing spline approximation,
45

weighted averages, 4