Spline Methods
Draft

Tom Lyche and Knut Mørken

January 5, 2005

Contents

CHAPTER 1

Splines and B-splines an Introduction

In this first chapter, we consider the following fundamental problem: Given a set of points in the plane, determine a smooth curve that approximates the points. The algorithm for determining the curve from the points should be well suited for implementation on a computer. That is, it should be efficient and it should not be overly sensitive to round-off errors in the computations. We only consider methods that

involve a relatively small number of elementary arithmetic operations; this ensures that the methods are efficient. The sensitivity of the methods to round-off errors is controlled by insisting that all

the operations involved should amount to forming weighted averages of the given points. This has the added advantage that the constructions are geometrical in nature and easy to visualise.

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 1.6 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* .

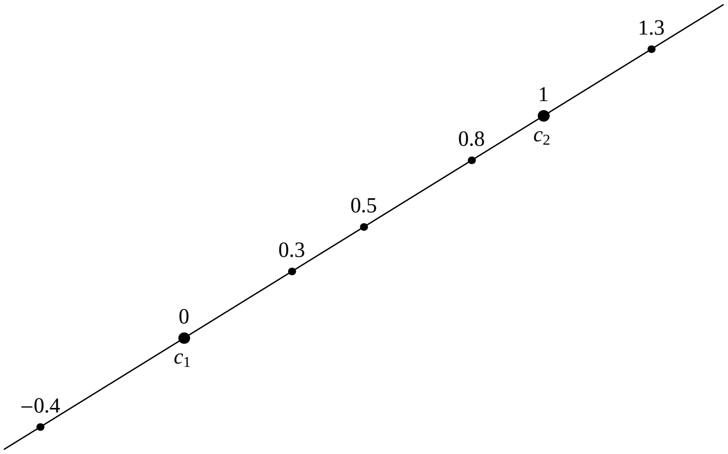


Figure 1.1. Some points on the line $(1-\lambda)\boldsymbol{c}_1+\lambda\boldsymbol{c}_2$ and the corresponding values of λ.

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

$$c=(1-\lambda)c_1+\lambda c_2. \tag{1.1}$$

Here c_1 and c_2 are two given numbers and λ a given weight 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 $\boldsymbol{c}_1=(x_1,y_1)$ and $\boldsymbol{c}_2=(x_2,y_2)$ then a convex combination of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ is an expression on the form

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

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,

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

where $(x,y)=\boldsymbol{c}$.

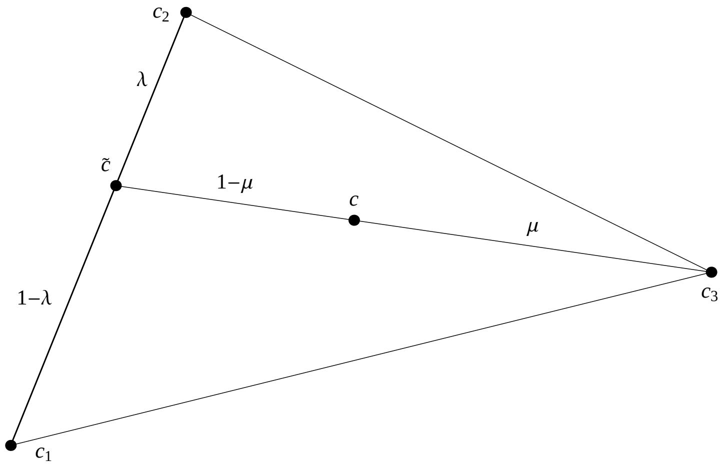


Figure 1.2. Determining the convex hull of three points.

Sometimes combinations on the form (1.1) or (1.2) with $\lambda < 0$ or $\lambda > 1$ are required. A combination of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ as in (1.2) with no restriction on λ other than $\lambda \in \mathbb{R}$ is called an *affine combination* of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. As λ takes on all real numbers, the point $\boldsymbol{c}$ in (1.2) will trace out the whole straight line that passes through $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. If we restrict λ to lie in the interval $[0, 1]$, we only get the part of the line that lies between $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. This is the *convex hull* , or the set of all weighted averages, of the two points. Figure 1.1 shows two points $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ and the line they define, together with some points on the line and their corresponding values of λ.

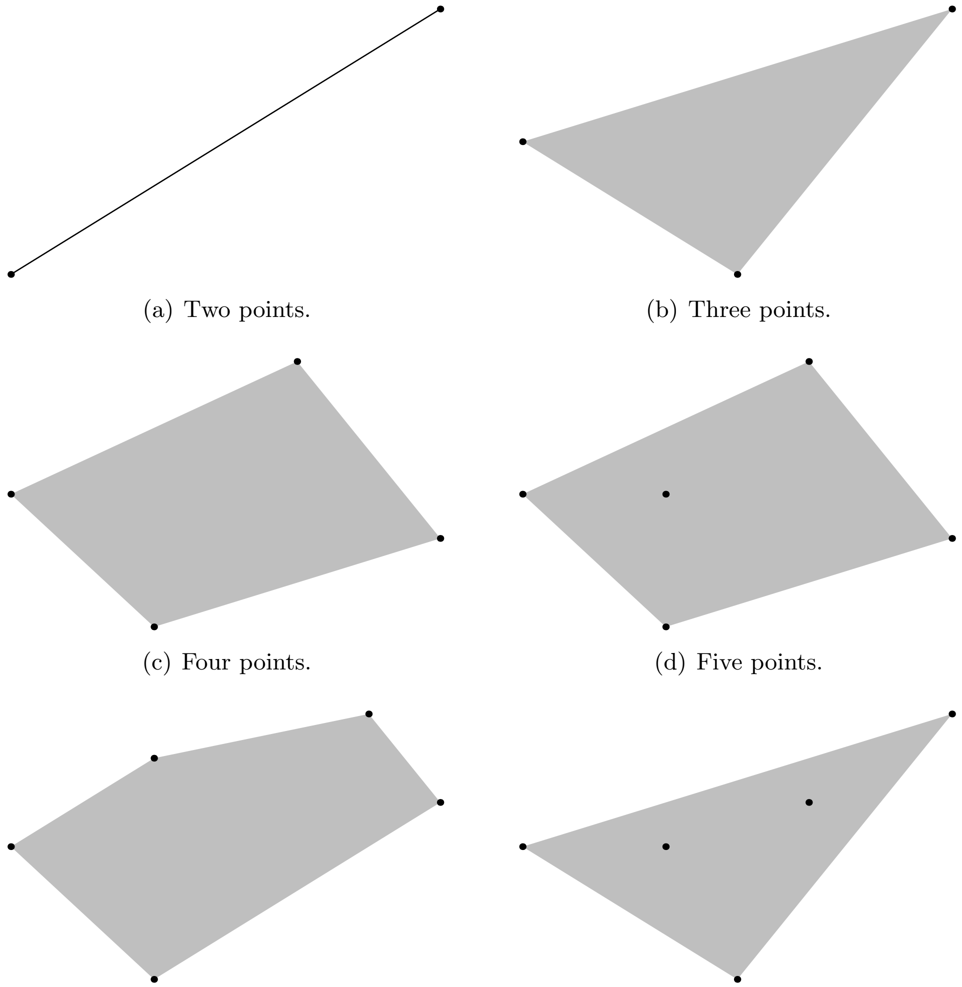
We can form convex and affine combinations in any space dimension, we just let $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ be points in the appropriate space. If we are working in $\mathbb{R}^n$ for instance, then $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ have n components. In our examples we will mostly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

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

$$\boldsymbol{c} = \lambda_1 \boldsymbol{c}_1 + \lambda_2 \boldsymbol{c}_2 + \cdots + \lambda_n \boldsymbol{c}_n$$

where the n numbers λ_i sum to one, $\sum_{i=1}^n \lambda_i = 1$, and also satisfy $0 \le \lambda_i \le 1$ for $i = 1$, 2, ..., n. As for two points, the convex hull of the points $(\boldsymbol{c}_i)_{i=1}^n$ 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 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.



(a) Two points.

(b) Three points.

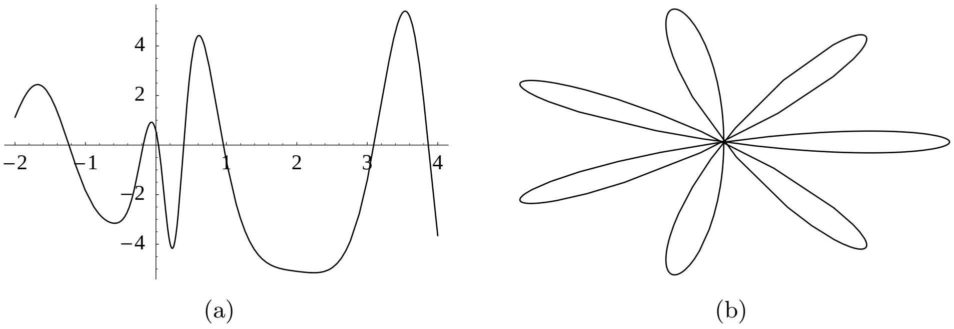
(c) Four points.

(d) Five points.

(e) Five points.

(f) Five points.

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



(a)

(b)

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

1.2 Some fundamental concepts

Our basic challenge in this chapter is to construct a curve from some given points in the plane. The underlying numerical algorithms should be simple and efficient and preferably based on forming repeated convex combinations as in (1.1). To illustrate some fundamental concepts let us consider the case where we are given two points $\boldsymbol{c}_0 = (x_0, y_0)$ and $\boldsymbol{c}_1 = (x_1, y_1)$ (we always denote points and vectors by bold type). The most natural curve to construct from these points is the straight line segment which connects the two points. In Section 1.1.2 we saw that this line segment coincides with the convex hull of the two points and that a point on the line could be represented by a convex combination, see (1.2). More generally we can express this line segment as

$$\boldsymbol{q}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1) = \frac{t_1 - t}{t_1 - t_0}\boldsymbol{c}_0 + \frac{t - t_0}{t_1 - t_0}\boldsymbol{c}_1 \qquad \text{for } t \in [t_0, t_1]. \tag{1.3}$$

Here t_0 and t_1 are two arbitrary real numbers with $t_0 < t_1$. Note that 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 $[t_0, t_1]$. The expression in (1.3) is therefore a convex combination of $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$. In fact, if we set $\lambda = (t - t_0)/(t_1 - t_0)$ then (1.3) becomes (1.2).

A representation of a line as in (1.3), where we have a function 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 a linear function

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

but here we run into problems if $x_0 = x_1$, i.e., if the line is vertical. Vertical lines can only be expressed as $x = c$ (with each constant c characterising a line) if we insist on using functions. In general, a parametric representation can cross itself or return to its starting point, but this is impossible for a function, which always maps 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 our examples the points lie in the plane, but we emphasise again that the constructions will work for curves in any space dimension; just replace the planar points with points with the appropriate number of components. For example, a line in space is obtained by letting $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ in (1.3) be points in space with three components. In particular, we can construct a function by letting the points be real numbers. In later chapters we will work mainly with functions since the core of the 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 are arbitrary except that we assumed $t_0 < t_1$. Regardless of how we choose the parameters, the resulting curve is always the same. If we consider the variable t to denote time, the parametric representation $\boldsymbol{q}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1)$ gives a way to travel from $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$. The parameter t_0 gives the time at which we start at $\boldsymbol{c}_0$ and t_1 the time at which we arrive at $\boldsymbol{c}_1$. With this interpretation, different choices of t_0 and t_1 correspond to different ways of travelling along the line. The speed of travel along the curve is given by the tangent vector or derivative

$$\boldsymbol{q}'(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1) = \frac{\boldsymbol{c}_1 - \boldsymbol{c}_0}{t_1 - t_0},$$

while the scalar speed or velocity is given by the length of the tangent vector

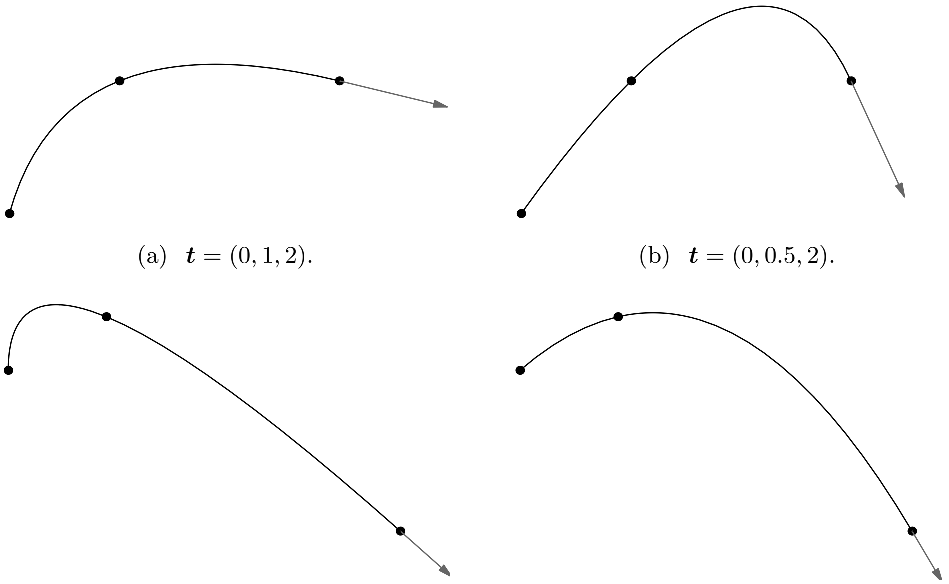
$$\bigl|\boldsymbol{q}'(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1)\bigr| = \frac{|\boldsymbol{c}_1 - \boldsymbol{c}_0|}{t_1 - t_0} = \frac{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}{t_1 - t_0}.$$

If $t_1 - t_0$ is small (compared to $|\boldsymbol{c}_1 - \boldsymbol{c}_0|$), then we have to travel quickly to reach $\boldsymbol{c}_1$ at time t_1 whereas if $t_1 - t_0$ is large then we have to move slowly to arrive at $\boldsymbol{c}_1$ exactly at time t_1. Note that regardless of our choice of t_0 and t_1, the speed along the curve is independent of t and therefore constant. This reflects the fact that all the representations of the line given by (1.3) are linear in t.

This discussion shows how we must differentiate between the geometric curve in question (a straight line in our case) and the parametric representation of the curve. Loosely speaking, a curve is defined as the collection of all the different parametric representations of the curve. In practise a curve is usually given by a particular parametric representation, and we will be sloppy and often refer to a parametric representation as a curve. The distinction between a curve and a particular parametric representation is not only 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, and the simplest example of this is the straight line between the points. In this section we show how to construct curves that interpolate any number of points.



(a) $\boldsymbol{t} = (0, 1, 2)$.

(b) $\boldsymbol{t} = (0, 0.5, 2)$.

(c) $\boldsymbol{t} = (0, 1, 2)$.

(d) $\boldsymbol{t} = (0, 0.5, 2)$.

Figure 1.5. Some examples of quadratic interpolation.

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 $\boldsymbol{c}_0$, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ we also need three parameters $(t_i)_{i=0}^2$. We first construct the two straight lines $\boldsymbol{q}_{0,1}(t) = \boldsymbol{q}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1)$ and $\boldsymbol{q}_{1,1}(t) = \boldsymbol{q}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2; t_1, t_2)$. If we now form the weighted average

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

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

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

Four examples are shown in Figure 1.5, with the interpolation points $(\boldsymbol{c}_i)_{i=0}^2$ given as black dots and the values of the three parameters $\boldsymbol{t} = (t_i)_{i=0}^2$ 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 $\boldsymbol{q}_{0,2}(t)$ as the position at time t of someone travelling along the curve. 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 $\boldsymbol{c}_1$ to $\boldsymbol{c}_2$ than from $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$ with the effect that the curve bulges out between $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. 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'. The curves in Figure 1.5 (c) and (d) can be explained similarly. The interpolation points are the same in both cases, but now they are not uniformly distributed. In plot (a) the parameters are uniform which means that we must travel much faster between $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ (which are far apart) than between $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$ (which are close together). The result is a curve that is almost a straight line between the last two points and bulges out between the first two points. In plot (d) the parameters have been chosen so as to reflect better the geometric spacing between the points, and this gives a more uniformly rounded curve.

1.3.2 General polynomial interpolation

To construct a cubic curve that interpolates four points we follow the same strategy that was used to construct the quadratic interpolant. If the given points are $(\boldsymbol{c}_i)_{i=0}^3$ we first choose four parameters $\boldsymbol{t} = (t_i)_{i=0}^3$. We then form the two quadratic interpolants

$$\begin{aligned}\boldsymbol{q}_{0,2}(t) &= \boldsymbol{q}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2; t_0, t_1, t_2),\\ \boldsymbol{q}_{1,2}(t) &= \boldsymbol{q}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3; t_1, t_2, t_3),\end{aligned}$$

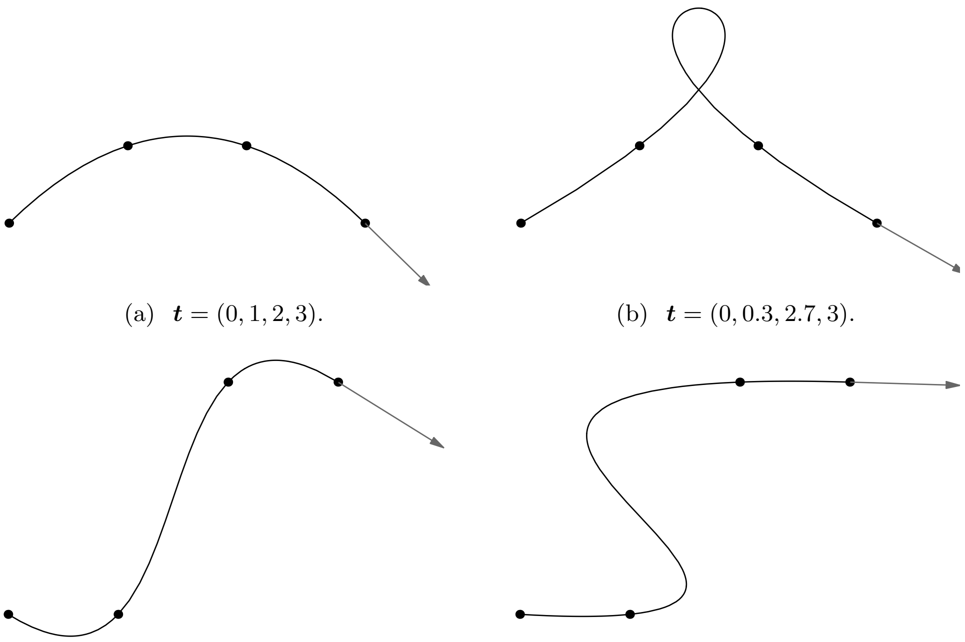
and combine these to obtain the cubic interpolant $\boldsymbol{q}_{0,3}(t)$,

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

At t_0 this interpolant agrees with $\boldsymbol{q}_{0,2}(t_0) = \boldsymbol{c}_0$ and at t_3 it agrees with $\boldsymbol{q}_{1,2}(t_3) = \boldsymbol{c}_3$. At an interior point t_i it is a convex combination of $\boldsymbol{q}_{0,1}(t_i)$ and $\boldsymbol{q}_{1,1}(t_i)$ which both interpolate $\boldsymbol{c}_i$ at t_i. Hence we also have $\boldsymbol{q}_{0,3}(t_i) = \boldsymbol{c}_i$ for $i = 1$ and $i = 2$ so $\boldsymbol{q}_{0,3}$ interpolates the four points $(\boldsymbol{c}_i)_{i=0}^3$ as it should.

Some examples of cubic interpolants are shown in Figure 1.6, and the same interpolation points are used in (a) and (b), and (c) and (d) respectively. The qualitative comments that we made about the quadratic interpolants also apply here. The pleasing shape of the curve in Figure 1.6 (a) is quite natural since both the interpolation points and parameters are quite uniformly spaced. However, by adjusting the parameters, quite strange behaviour can occur, even with these 'nice' interpolation points. In (b) there is so much time to 'waste' between $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ that the curve makes a complete loop. In (c) and (d) we see two different approaches to jumping from one level in the data to another. In (c) there is too much time to be spent between $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, and between $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$, the result being bulges between these points. In Figure 1.6 (d) there is too much time between $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ leading to the two big wiggles and almost straight lines between $\boldsymbol{c}_0$ and $\boldsymbol{c}_1$, and $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ respectively.

The general strategy for constructing interpolating curves should now be clear. Given d+1 points $(\boldsymbol{c}_i)_{i=0}^d$ and parameters $(t_i)_{i=0}^d$, the curve $\boldsymbol{q}_{0,d}$ of degree d that satisfies $\boldsymbol{q}_{0,d}(t_j) =$



(a) $\boldsymbol{t} = (0, 1, 2, 3)$.

(b) $\boldsymbol{t} = (0, 0.3, 2.7, 3)$.

(c) $\boldsymbol{t} = (0, 0.75, 2.25, 3)$.

(d) $\boldsymbol{t} = (0, 0.3, 2.8, 3)$.

Figure 1.6. Some examples of cubic interpolation.

$\boldsymbol{c}_j$ for $j = 0, \ldots, d$ is constructed by forming a convex combination between the two curves of degree $d-1$ that interpolate $(\boldsymbol{c}_i)_{i=0}^{d-1}$ and $(\boldsymbol{c}_i)_{i=1}^{d}$,

$$\boldsymbol{q}_{0,d}(t) = \frac{t_d - t}{t_d - t_0}\boldsymbol{q}_{0,d-1}(t) + \frac{t - t_0}{t_d - t_0}\boldsymbol{q}_{1,d-1}(t). \tag{1.4}$$

If we expand out this equation we find that $\boldsymbol{q}_{0,d}(t)$ can be written

$$\boldsymbol{q}_{0,d}(t) = \boldsymbol{c}_0\ell_{0,d}(t) + \boldsymbol{c}_1\ell_{1,d}(t) + \cdots + \boldsymbol{c}_d\ell_{d,d}(t), \tag{1.5}$$

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

$$\ell_{i,d}(t) = \prod_{\substack{0 \leq j \leq d \\ j \neq i}} \frac{(t - t_j)}{t_i - t_j}. \tag{1.6}$$

It is easy to check that these polynomials satisfy the condition

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

which is necessary since $\boldsymbol{q}_{0,d}(t_k) = \boldsymbol{c}_k$.

The complete computations involved in computing $\boldsymbol{q}_{0,d}(t)$ is summarised in the following algorithm.

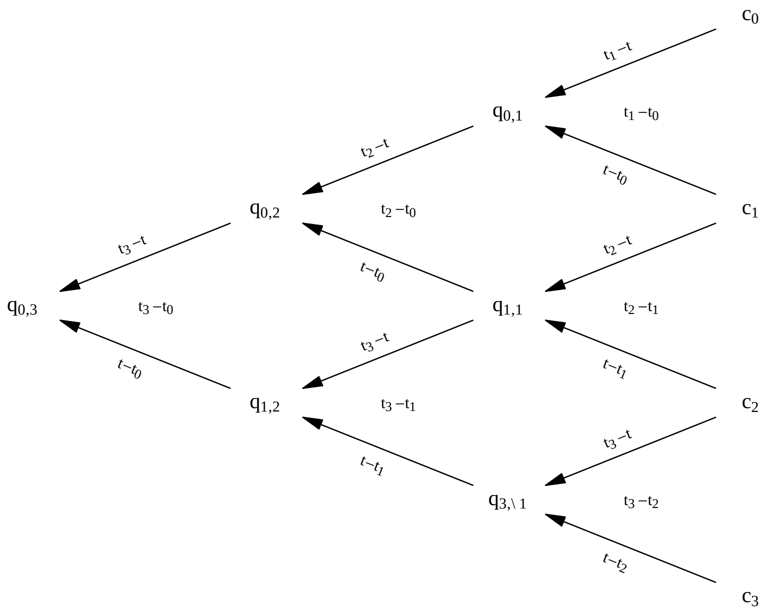


Figure 1.7. Computing a point on a cubic interpolating curve.

Algorithm 1.1 (Neville-Aitken method)**.** *Let d be a positive integer and let the $d+1$ points $(\boldsymbol{c}_i)_{i=0}^d$ be given together with $d+1$ strictly increasing parameter values $\boldsymbol{t} = (t_i)_{i=0}^d$. There is a polynomial curve $\boldsymbol{q}_{0,d}$ of degree d that satisfies the conditions*

$$\boldsymbol{q}_{0,d}(t_i) = \boldsymbol{c}_i \qquad \text{for } i = 0,\, 1,\, \ldots,\, d,$$

and for any real number t the following algorithm computes the point $\boldsymbol{q}_{0,d}(t)$. First set $\boldsymbol{q}_{i,0}(t) = \boldsymbol{c}_i$ for $i = 0,\, 1,\, \ldots,\, d$ and then compute

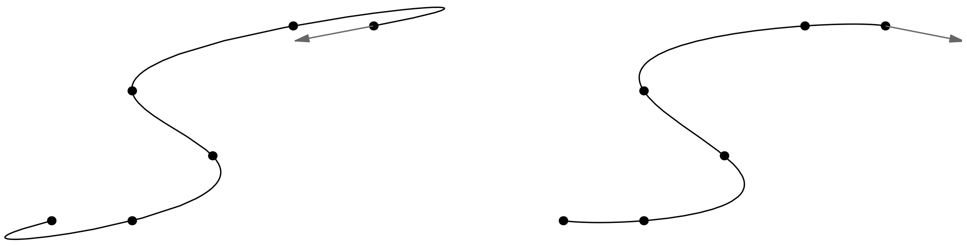
$$\boldsymbol{q}_{i,r}(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i}\boldsymbol{q}_{i,r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i}\boldsymbol{q}_{i+1,r-1}(t)$$

for $i = 0,\, 1,\, \ldots,\, d-r$ and $r = 1,\, 2,\, \ldots,\, d$.

The computations involved in determining a cubic interpolating curve are shown in the triangular table in Figure 1.7. The computations start from the right and proceed to the left and at any point a quantity $\boldsymbol{q}_{i,r}$ is computed by combining, in an affine combination, the two quantities at the beginning of the two arrows meeting at $\boldsymbol{q}_{i,r}$. The expression between the two arrows is the denominator of the weights in the affine combination while the two numerators are written along the respective arrows.

Two examples of curves of degree five are shown in Figure 1.8, both interpolating the same points. The wiggles in (a) indicate that $t_1 - t_0$ and $t_6 - t_5$ should be made smaller and the result in (b) confirms this.

It should be emphasised that choosing the correct parameter values is a complex problem. Our simple analogy with travelling along a road may seem to explain some of the behaviour we have observed, but to formalise these observations into a foolproof algorithm for choosing parameter values is a completely different matter. As we shall see later, selection of parameter values is also an issue when working with spline curves.



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

(b) $\boldsymbol{t} = (0, 0.5, 2, 3, 4.5, 5)$.

Figure 1.8. Two examples of interpolation with polynomial curves of degree five.

The challenge of determining good parameter values is not the only problem with polynomial interpolation. A more serious limitation is the fact that the polynomial degree is only one less than the number of interpolation points. In a practical situation we may be given several thousand points which would require a polynomial curve of an impossibly high degree. To compute a point on a curve of degree d requires a number of multiplications and additions that are at best proportional to d (using the *Newton form* of the interpolating polynomial); the algorithm we have presented here requires roughly d^2 additions and multiplications. If for example $d = 1000$, computer manipulations like plotting and interactive editing of the curve would be much too slow to be practical, even on today's fast computers. More importantly, it is well known that round-off errors in the computer

makes numerical manipulations of high degree polynomials increasingly (with the degree) inaccurate. We therefore need alternative ways to approximate a set of points by a smooth curve.

1.3.3 Interpolation by convex combinations?

In the interpolation algorithm for polynomials of degree d, Algorithm 1.1, the last step is to form a convex combination between two polynomials of degree $d-1$,

$$\boldsymbol{q}_{0,d}(t) = \frac{t_d - t}{t_d - t_0}\boldsymbol{q}_{0,d-1}(t) + \frac{t - t_0}{t_d - t_0}\boldsymbol{q}_{1,d-1}(t).$$

More precisely, the combination is convex as long as t lies in the interval $[t_0, t_d]$. But if the algorithm is based on forming convex combinations, any point on the final curve should be within the convex hull of the given interpolation points. By merely looking at the figures it is clear that this is not true, except in the case where we only have two points and the interpolant is the straight line that connects the points. To see what is going on, let us consider the quadratic case in detail. Given the points $(\boldsymbol{c}_i)_{i=0}^2$ and the parameters $(t_i)_{i=0}^2$, we first form the two straight lines

$$\boldsymbol{q}_{0,1}(t) = \frac{t_1 - t}{t_1 - t_0}\boldsymbol{c}_0 + \frac{t - t_0}{t_1 - t_0}\boldsymbol{c}_1, \tag{1.7}$$

$$\boldsymbol{q}_{1,1}(t) = \frac{t_2 - t}{t_2 - t_1}\boldsymbol{c}_1 + \frac{t - t_1}{t_2 - t_1}\boldsymbol{c}_2, \tag{1.8}$$

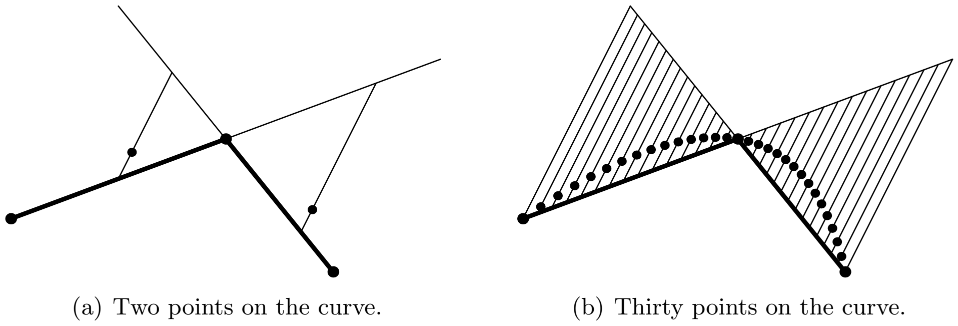

(a) Two points on the curve.
(b) Thirty points on the curve.

Figure 1.9. The geometry of quadratic interpolation.

and from these the quadratic segment

$$\boldsymbol{q}_{0,2}(t) = \frac{t_2 - t}{t_2 - t_0}\boldsymbol{q}_{0,1}(t) + \frac{t - t_0}{t_2 - t_0}\boldsymbol{q}_{1,1}(t). \tag{1.9}$$

The combination in (1.7) is convex as long as t is in $[t_0, t_1]$, the combination in (1.8) is convex when t lies within $[t_1, t_2]$, and the combination in (1.9) is convex when t is restricted to $[t_0, t_2]$. But in computing $\boldsymbol{q}_{0,2}(t)$ we also have to compute $\boldsymbol{q}_{0,1}(t)$ and $\boldsymbol{q}_{1,1}(t)$, 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 1.9.

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.

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*. Bézier curves are also polynomial curves and for that reason not very practical, 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 $\boldsymbol{c}_0$, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$, 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 (1.7) and (1.8) 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,

$$\boldsymbol{p}_{1,1}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1) = (1 - t)\boldsymbol{c}_0 + t\boldsymbol{c}_1, \tag{1.10}$$

$$\boldsymbol{p}_{2,1}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2) = (1 - t)\boldsymbol{c}_1 + t\boldsymbol{c}_2. \tag{1.11}$$

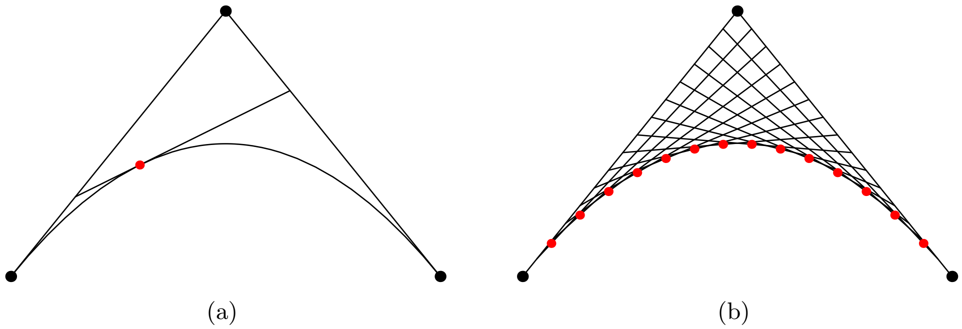


Figure 1.10. A Bézier curve based on three points.

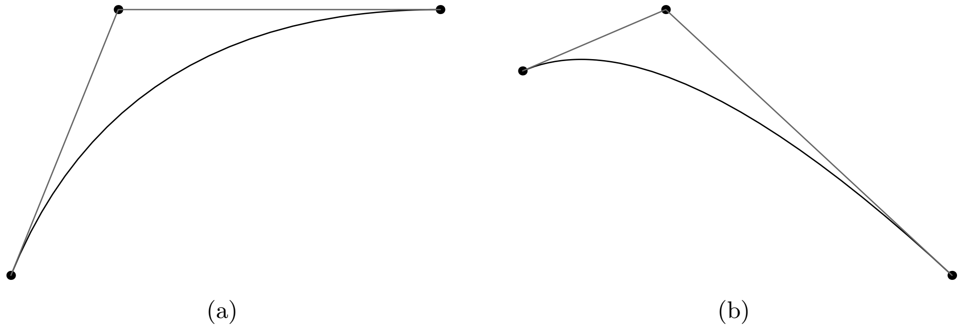


Figure 1.11. Two examples of quadratic Bézier curves.

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

$$\boldsymbol{p}_{2,2}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2) = (1-t)\boldsymbol{p}_{1,1}(t) + t\boldsymbol{p}_{2,1}(t). \tag{1.12}$$

The construction is illustrated in Figure 1.10 (a). In Figure 1.10 (b), where we have repeated the construction for 15 uniformly spaced values of t, the underlying curve is clearly visible.

If we insert the explicit expressions for the two lines in (1.10) and (1.11) in (1.12) we find

$$\boldsymbol{p}_{2,2}(t) = (1-t)^2\boldsymbol{c}_0 + 2t(1-t)\boldsymbol{c}_1 + t^2\boldsymbol{c}_2 = b_{0,2}(t)\boldsymbol{c}_0 + b_{1,2}(t)\boldsymbol{c}_1 + b_{2,2}(t)\boldsymbol{c}_2. \tag{1.13}$$

This is called a quadratic *Bézier curve*; the points $(\boldsymbol{c}_i)_{i=0}^2$ 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. Two examples of quadratic Bézier curves with their control points and control polygons are shown in Figure 1.11 (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 1.10 and 1.11. We note that the curve interpolates $\boldsymbol{c}_0$ at $t = 0$ and $\boldsymbol{c}_2$ 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 $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$ coincides with the direction of the tangent to

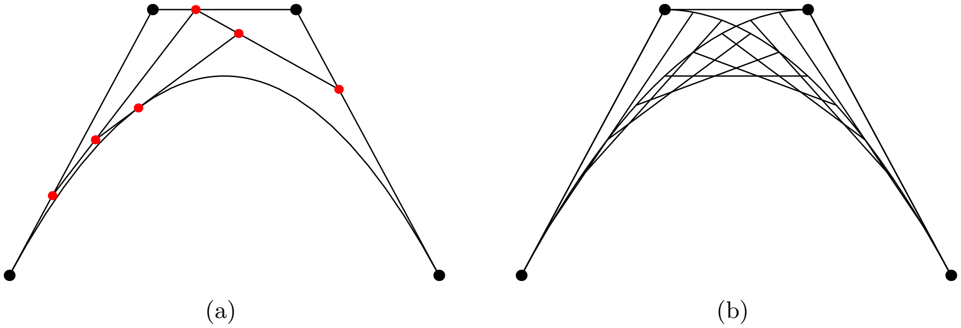


Figure 1.12. Constructing a Bézier curve from four points.

the curve at $t = 0$ while the line from $\boldsymbol{c}_1$ to $\boldsymbol{c}_2$ coincides with the direction of the tangent at $t = 1$. This observation can be confirmed by differentiating equation (1.13). We find

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

The three polynomials in (1.13) add up to 1,

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

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 $\boldsymbol{p}_{2,2}(t)$ is a convex combination of the three points $(\boldsymbol{c}_i)_{i=0}^2$. The geometric interpretation of this is that the curve lies entirely within the triangle formed by the three given points, the *convex hull* of $\boldsymbol{c}_0$, $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$.

1.4.2 Bézier curves based on four and more points

The construction of quadratic Bézier curves generalises naturally to any number of points and any polynomial degree. If we have four points $(\boldsymbol{c}_i)_{i=0}^3$ we can form the cubic Bézier curve $\boldsymbol{p}_{3,3}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$ by taking a weighted average of two quadratic curves,

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

If we insert the explicit expressions for $\boldsymbol{p}_{2,2}(t)$ and $\boldsymbol{p}_{3,2}(t)$, we find

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

The construction is illustrated in Figure 1.12. 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). The data points are the same as those used in Figure 1.6 (a) and (b). Two further examples are shown in Figure 1.13, together with the control points and control polygons which are defined just as in the quadratic case. The data points in Figure 1.13 are the same as those used in Figure 1.6 (c) and (d). In Figure 1.13 (b) the control polygon crosses itself with the result that the underlying Bézier curve does the same.

To construct Bézier curves of degree d, we start with $d + 1$ *control points* $(\boldsymbol{c}_i)_{i=0}^d$, and form a curve $\boldsymbol{p}_{d,d}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_0, \ldots, \boldsymbol{c}_d)$ based on these points by taking a convex

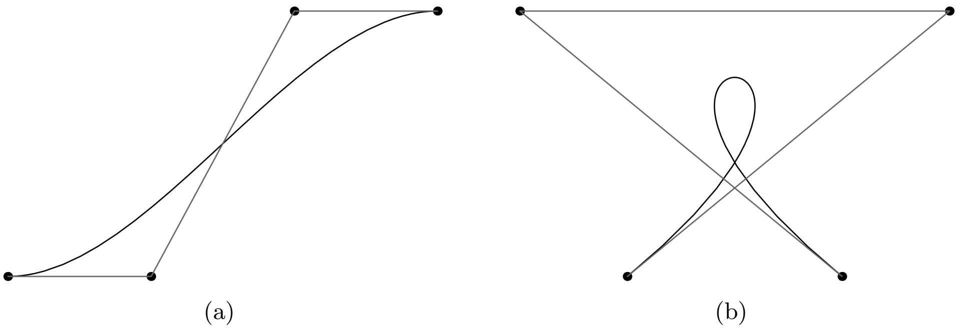


Figure 1.13. Two examples of cubic Bézier curves.

combination of the two Bézier curves $\boldsymbol{p}_{d-1,d-1}$ and $\boldsymbol{p}_{d,d-1}$ of degree $d-1$ which are based on the control points $(\boldsymbol{c}_i)_{i=0}^{d-1}$ and $(\boldsymbol{c}_i)_{i=1}^{d}$ respectively,

$$\boldsymbol{p}_{d,d}(t) = (1-t)\boldsymbol{p}_{d-1,d-1}(t) + t\boldsymbol{p}_{d,d-1}(t).$$

If we expand out we find by an inductive argument that

$$\boldsymbol{p}_{d,d}(t) = b_{0,d}(t)\boldsymbol{c}_0 + \cdots + b_{d,d}(t)\boldsymbol{c}_d, \tag{1.14}$$

where

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

As in the quadratic case we have

$$b_{0,d}(t) + b_{1,d}(t) + \cdots + b_{d,d}(t) = (1-t+t)^d = 1$$

and $0 \le b_{i,d}(t) \le 1$ for any t in $[0,1]$ and $0 \le i \le d$. For any t in $[0,1]$ the point $\boldsymbol{p}_{d,d}(t)$ therefore lies in the convex hull of the points $(\boldsymbol{c}_i)_{i=0}^{d}$. The curve interpolates the first and last control points and the tangent at $t=0$ points in the direction from $\boldsymbol{c}_0$ to $\boldsymbol{c}_1$ and the tangent at $t=1$ points in the direction from $\boldsymbol{c}_{d-1}$ to $\boldsymbol{c}_d$,

$$\boldsymbol{p}'_{d,d}(0) = d(\boldsymbol{c}_1 - \boldsymbol{c}_0), \qquad \boldsymbol{p}'_{d,d}(1) = d(\boldsymbol{c}_d - \boldsymbol{c}_{d-1}). \tag{1.15}$$

As in the quadratic and cubic cases the piecewise linear curve with the control points as vertexes is called the control polygon of the curve.

The complete computations involved in computing a point on a Bézier curve are given in Algorithm 1.2 and depicted graphically in the triangular table in Figure 1.14.

Algorithm 1.2. *Let d be a positive integer and let the $d+1$ points $(\boldsymbol{c}_i)_{i=0}^{d}$ be given. The point $\boldsymbol{p}_{d,d}(t)$ on the Bézier curve $\boldsymbol{p}_{0,d}$ of degree d can be determined by the following computations. First set $\boldsymbol{p}_{i,0}(t) = \boldsymbol{c}_i$ for $i = 0, 1, \ldots, d$ and then compute $\boldsymbol{p}_{d,d}(t)$ by*

$$\boldsymbol{p}_{i,r}(t) = (1-t)\boldsymbol{p}_{i-1,r-1}(t) + t\boldsymbol{p}_{i,r-1}(t)$$

for $i = r, \ldots, d$ and $r = 1, 2, \ldots, d$.

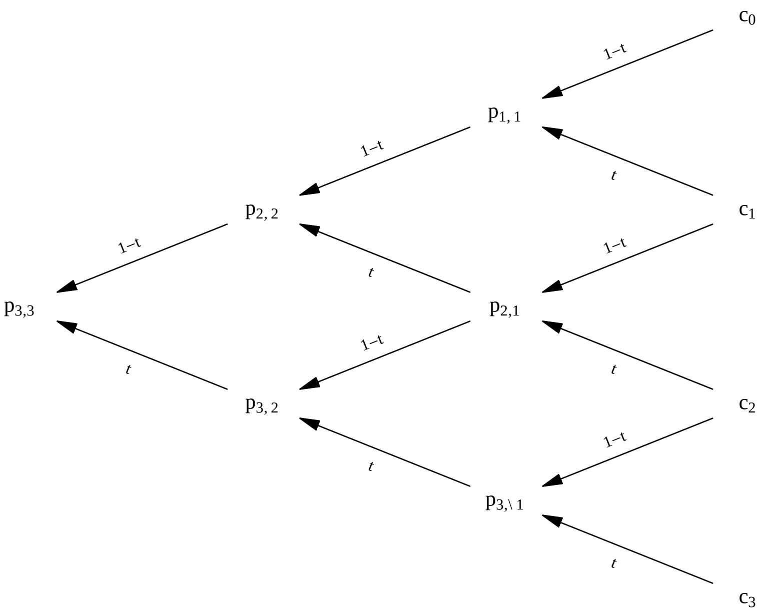


Figure 1.14. Computing a point on a cubic Bézier curve.

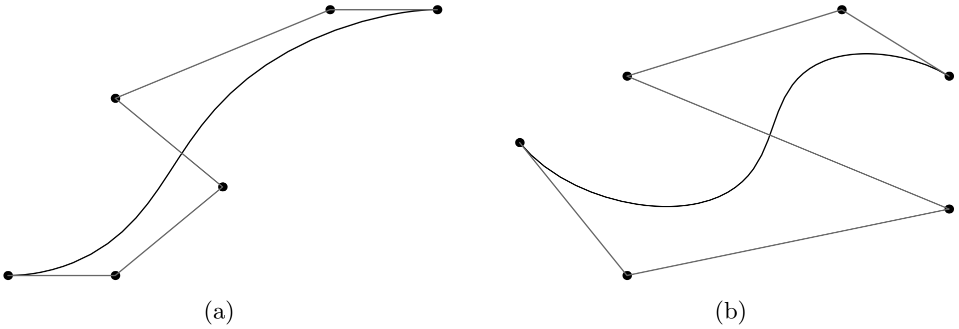


Figure 1.15. Two Bézier curves of degree five.

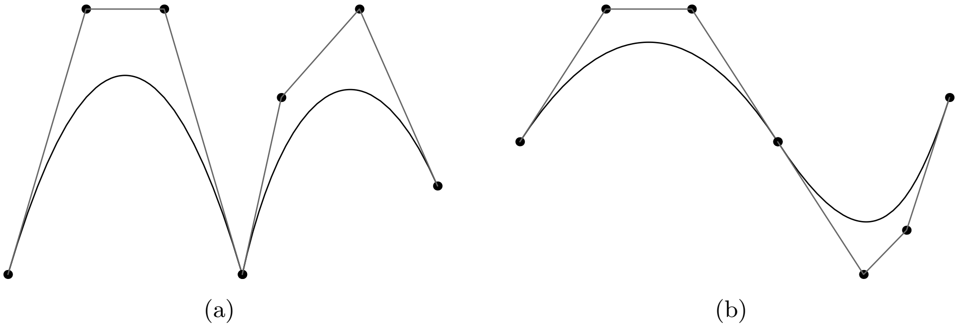


Figure 1.16. Different forms of continuity between two segments of a cubic Bézier curve.

Two examples of Bézier curves of degree five are shown in Figure 1.15. The curve in Figure (a) uses the interpolation points of the two curves in Figure 1.8 as control points.

We have defined Bézier curves on the interval $[0, 1]$, but any nonempty interval would work. If the interval is $[a, b]$ we just have to use convex combinations on the form

$$\boldsymbol{c} = \frac{b-t}{b-a}\boldsymbol{c}_0 + \frac{t-a}{b-a}\boldsymbol{c}_1$$

instead. Equivalently, we can use a linear change of parameter; if $\boldsymbol{p}_{d,d}(t)$ is a Bézier curve on $[0, 1]$ then

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

is a Bézier curve on $[a, b]$.

1.4.3 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 $(\boldsymbol{c}_i)_{i=0}^d$ and the curve itself becomes less intuitive when the degree is large.

Because of these shortcomings it is common 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 $(\boldsymbol{c}_0^i, \ldots, \boldsymbol{c}_d^i)_{i=1}^n$. It is common to let each segment be defined over $[0, 1]$, but it is also possible to form a curve 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 $\boldsymbol{c}_d^{i-1} = \boldsymbol{c}_0^i$ which ensures that segments $i-1$ and i join together continuously as in Figure 1.16 (a). We obtain a smoother join by also letting the tangents be continuous at the join. From (1.15) we see that the tangent at the join between segments $i-1$ and i will be continuous if

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

An example is shown in Figure 1.16 (b).

Quadratic Bézier curves form the basis for the TrueType font technology, while cubic Bézier curves lie at the heart of PostScript and a number of draw programs like Adobe Illustrator. Figure 1.17 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.

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 generalisation 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 $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ we now represent the straight line between them by

$$\boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2; t_2, t_3) = \frac{t_3 - t}{t_3 - t_2}\boldsymbol{c}_1 + \frac{t - t_2}{t_3 - t_2}\boldsymbol{c}_2, \qquad t \in [t_2, t_3], \tag{1.16}$$

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 $(\boldsymbol{c}_i)_{i=1}^n$ 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 $i = 2, 3, \ldots, n$, and define the curve $\boldsymbol{f}$ by

$$\boldsymbol{f}(t) = \begin{cases} \boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2; t_2, t_3), & t \in [t_2, t_3), \\ \boldsymbol{p}(t \mid \boldsymbol{c}_2, \boldsymbol{c}_3; t_3, t_4), & t \in [t_3, t_4), \\ \quad\vdots & \quad\vdots \\ \boldsymbol{p}(t \mid \boldsymbol{c}_{n-1}, \boldsymbol{c}_n; t_n, t_{n+1}), & t \in [t_n, t_{n+1}]. \end{cases} \tag{1.17}$$

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

$$B_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1}, \\ 0, & \text{otherwise}, \end{cases} \tag{1.18}$$

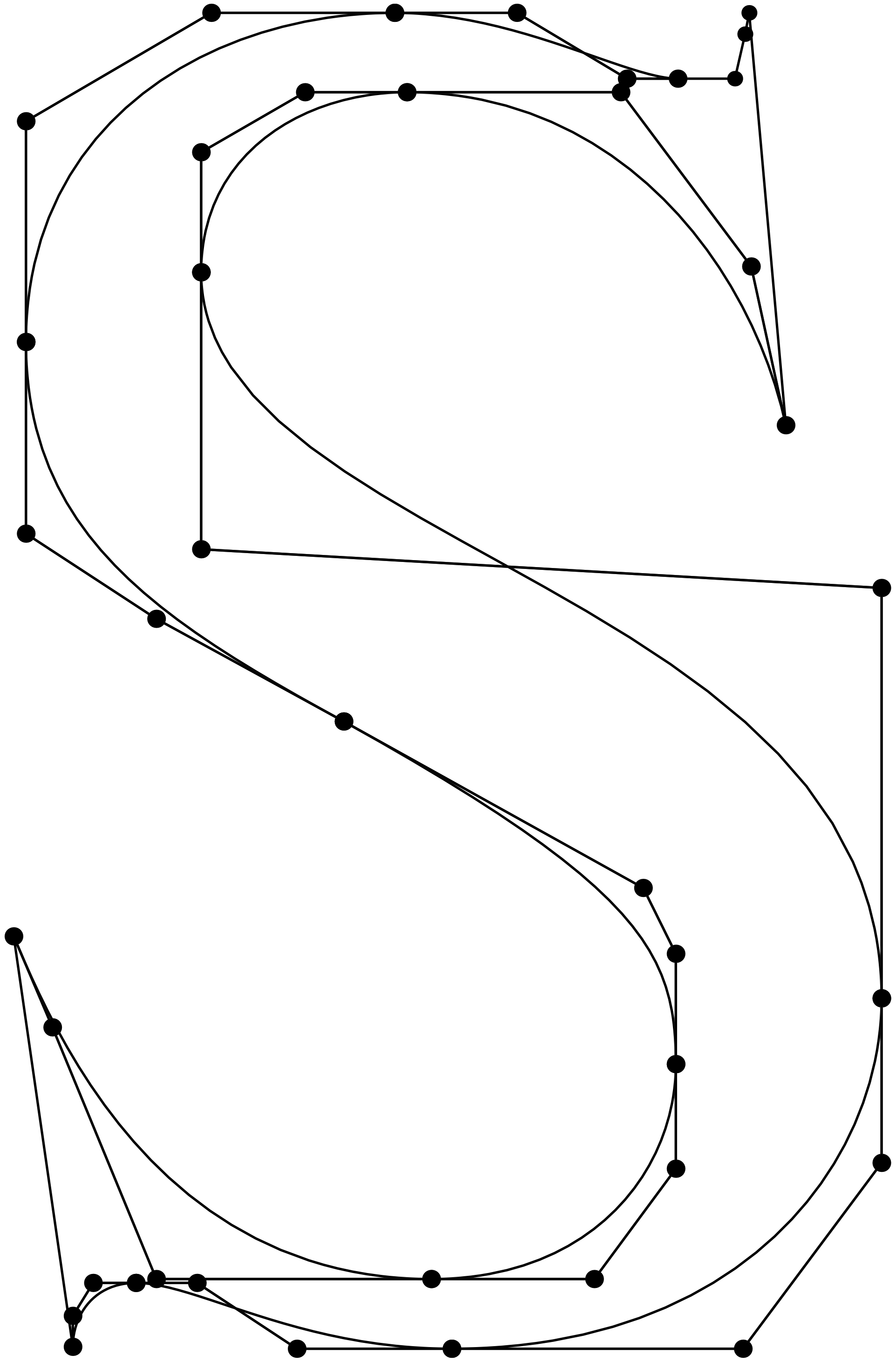


Figure 1.17. The letter S in the Postscript font Times Roman.

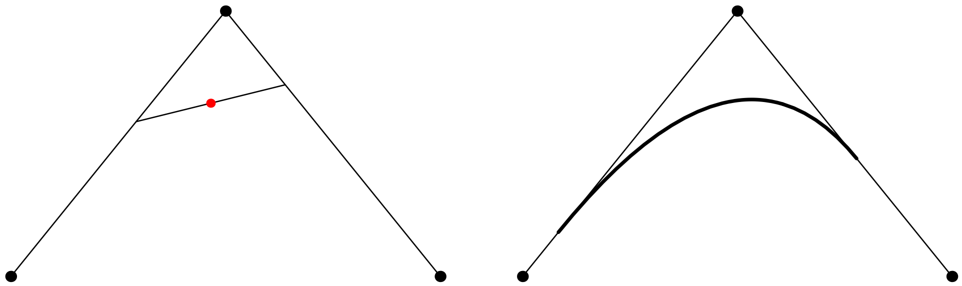


Figure 1.18. Construction of a segment of a quadratic spline curve.

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

$$\boldsymbol{f}(t) = \sum_{i=2}^{n} \boldsymbol{p}_{i,1}(t) B_{i,0}(t). \tag{1.19}$$

This construction can be generalised 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 $\boldsymbol{p}_{2,2}(t)$ is determined by taking three averages, all with weights $1-t$ and t since both the two line segments (1.10) and (1.11), and the quadratic curve itself (1.12), 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 curve of Section 1.4 in that we retain the convex combinations, but use more general weighted averages of the type in (1.16). To construct a spline curve based on the three control points $\boldsymbol{c}_1$, $\boldsymbol{c}_2$, and $\boldsymbol{c}_3$, we introduce 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 $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ by $\boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2; t_2, t_4)$ for $t \in [t_2, t_4]$, and the line connecting $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ by $\boldsymbol{p}(t \mid \boldsymbol{c}_2, \boldsymbol{c}_3; t_3, t_5)$ 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,

$$\boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3; t_2, t_3, t_4, t_5) = \frac{t_4 - t}{t_4 - t_3} \boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2; t_2, t_4) + \frac{t - t_3}{t_4 - t_3} \boldsymbol{p}(t \mid \boldsymbol{c}_2, \boldsymbol{c}_3; t_3, t_5) \tag{1.20}$$

with t varying in $[t_3, t_4]$. Of course we are free to vary t throughout the real line $\mathbb{R}$ since $\boldsymbol{p}$ is a polynomial in t, but then the three combinations involved are no longer all convex. The construction is illustrated in Figure 1.18. Note that if $t_2 = t_3 = 0$ and $t_4 = t_5 = 1$ we are back in the Bézier setting.

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. Suppose we have n control points $(\boldsymbol{c}_i)_{i=1}^n$ 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

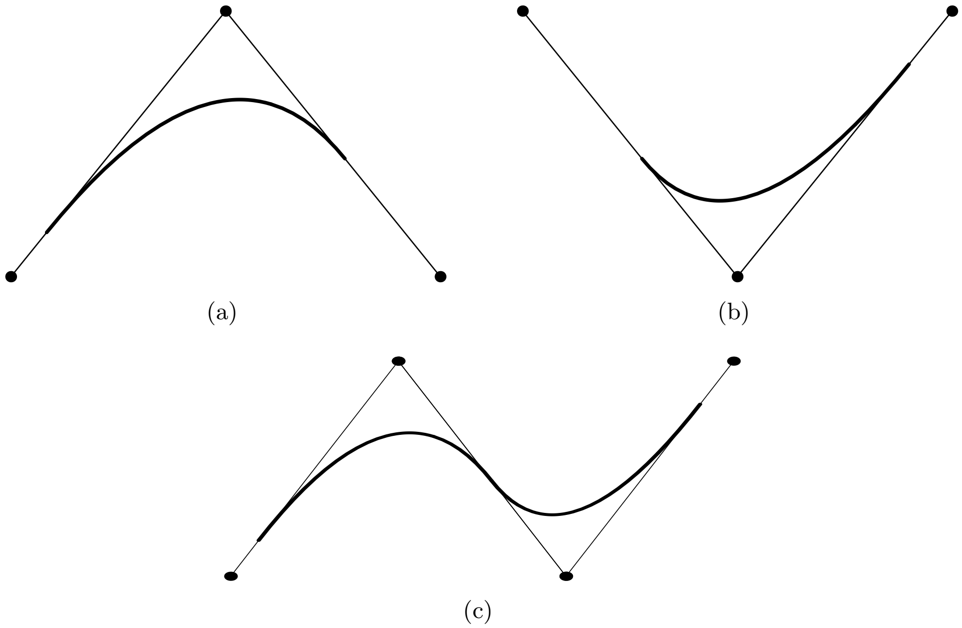


Figure 1.19. A quadratic spline curve (c) and its two polynomial segments (a) and (b).

curve $\boldsymbol{f}(t)$ by

$$\boldsymbol{f}(t) = \begin{cases} \boldsymbol{p}(t \mid \boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3; t_2, t_3, t_4, t_5), & t_3 \leq t \leq t_4, \\ \boldsymbol{p}(t \mid \boldsymbol{c}_2, \boldsymbol{c}_3, \boldsymbol{c}_4; t_3, t_4, t_5, t_6), & t_4 \leq t \leq t_5, \\ \quad\vdots & \quad\vdots \\ \boldsymbol{p}(t \mid \boldsymbol{c}_{n-2}, \boldsymbol{c}_{n-1}, \boldsymbol{c}_n; t_{n-1}, t_n, t_{n+1}, t_{n+2}), & t_n \leq t \leq t_{n+1}. \end{cases} \tag{1.21}$$

An example with $n = 4$ is shown in Figure 1.19. 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 3.2.3 of Chapter 3; see also exercise 6.

By making use of the piecewise constant functions $\{B_{i,0}\}_{i=3}^{n}$ defined in (1.18) and the abbreviation $\boldsymbol{p}_{i,2}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_{i-2}, \boldsymbol{c}_{i-1}, \boldsymbol{c}_i; t_{i-1}, t_i, t_{i+1}, t_{i+2})$, we can write $\boldsymbol{f}(t)$ as

$$\boldsymbol{f}(t) = \sum_{i=3}^{n} \boldsymbol{p}_{i,2}(t) B_{i,0}(t). \tag{1.22}$$

Two examples of quadratic spline curves are shown in Figure 1.20. The control points are the same as those in Figure 1.13. We observe that the curves behave like Bézier curves at the two ends.

1.5.3 Spline curves of higher degrees

The construction of spline curves can be generalised to arbitrary polynomial degrees by forming more averages. A cubic spline segment requires four control points $\boldsymbol{c}_{i-3}$, $\boldsymbol{c}_{i-2}$,

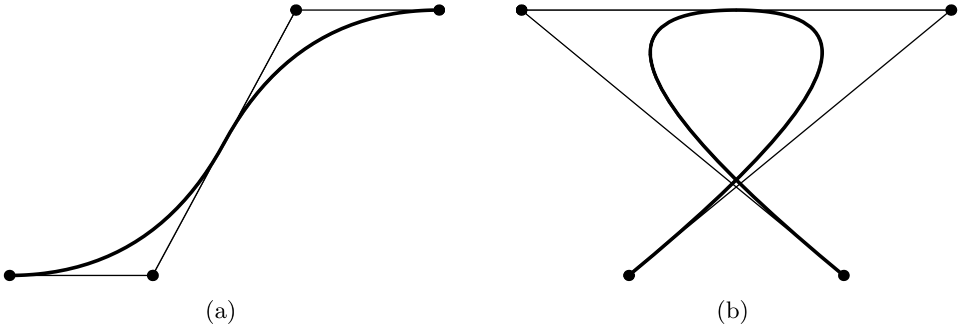


Figure 1.20. Two quadratic spline curves, both with knots $\boldsymbol{t} = (0, 0, 0, 1, 2, 2, 2)$.

$\boldsymbol{c}_{i-1}$, $\boldsymbol{c}_i$, and six knots $(t_j)_{j=i-2}^{i+3}$ which must form a nondecreasing sequence of numbers with $t_i < t_{i+1}$. The curve is the average of two quadratic segments,

$$\begin{aligned}\boldsymbol{p}(t \mid \boldsymbol{c}_{i-3}, \boldsymbol{c}_{i-2}, \boldsymbol{c}_{i-1}, \boldsymbol{c}_i; t_{i-2}, t_{i-1}, t_i, t_{i+1}, t_{i+2}, t_{i+3}) = \\ \frac{t_{i+1}-t}{t_{i+1}-t_i}\boldsymbol{p}(t \mid \boldsymbol{c}_{i-3}, \boldsymbol{c}_{i-2}, \boldsymbol{c}_{i-1}; t_{i-2}, t_{i-1}, t_{i+1}, t_{i+2}) + \\ \frac{t-t_i}{t_{i+1}-t_i}\boldsymbol{p}(t \mid \boldsymbol{c}_{i-2}, \boldsymbol{c}_{i-1}, \boldsymbol{c}_i; t_{i-1}, t_i, t_{i+2}, t_{i+3}),\end{aligned} \tag{1.23}$$

with t varying in $[t_i, t_{i+1}]$. The two quadratic segments are given by convex combinations of linear segments on the two intervals $[t_{i-1}, t_{i+1}]$ and $[t_i, t_{i+2}]$, as in (1.20). The three line segments are in turn given by convex combinations of the given points on the intervals $[t_{i-2}, t_{i+1}]$, $[t_{i-1}, t_{i+2}]$ and $[t_i, t_{i+3}]$. Note that all these intervals contain $[t_i, t_{i+1}]$ so that when t varies in $[t_i, t_{i+1}]$ all the combinations involved in the construction of the cubic curve will be convex. This also shows that we can never get division by zero since we have assumed that $t_i < t_{i+1}$.

The explicit notation in (1.23) is too cumbersome, especially when we consider spline curves of even higher degrees, so we generalise the notation in (1.19) and (1.22) and set

$$\boldsymbol{p}_{i,k,s}(t) = \boldsymbol{p}(t \mid c_{i-k}, \dots, c_i, t_{i-k+1}, \dots, t_i, t_{i+s}, \dots, t_{i+k+s-1}), \tag{1.24}$$

for positive s and r, assuming that the control points and knots in question are given. The first subscript i in $\boldsymbol{p}_{i,k,s}$ indicates which control points and knots are involved (in general we work with many spline segments and therefore long arrays of control points and knots), the second subscript k gives the polynomial degree, and the last subscript s, gives the gap between the knots in the computation of the weight $(t - t_i)/(t_{i+s} - t_i)$. With the abbreviation (1.24), equation (1.23) becomes

$$\boldsymbol{p}_{i,3,1}(t) = \frac{t_{i+1}-t}{t_{i+1}-t_i}\boldsymbol{p}_{i-1,2,2}(t) + \frac{t-t_i}{t_{i+1}-t_i}\boldsymbol{p}_{i,2,2}(t).$$

Note that on both sides of this equation, the last two subscripts sum to four. Similarly, if the construction of quadratic splines given by (1.20) is expressed with the abbreviation given in (1.24), the last two subscripts add to three. The general pattern is that in the recursive formulation of spline curves of degree d, the last two subscripts always add to

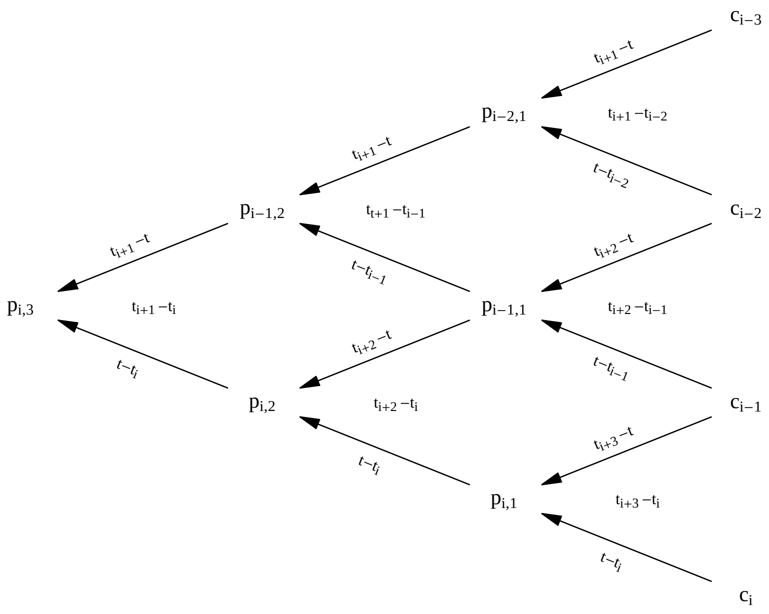


Figure 1.21. Computing a point on a cubic spline curve.

$d+1$. Therefore, when the degree of the spline curves under construction is fixed we can drop the third subscript and write $\boldsymbol{p}_{i,k,s} = \boldsymbol{p}_{i,k}$.

The complete computations involved in computing a point on the cubic segment $\boldsymbol{p}_{i,3}(t)$ can be arranged in the triangular array shown in Figure 1.21 (all arguments to the $\boldsymbol{p}_{i,k}$ have been omitted to conserve space). The labels should be interpreted as in Figure 1.7.

A segment of a general spline curve of degree d requires $d+1$ control points $(\boldsymbol{c}_j)_{j=i-d}^{i}$ and $2d$ knots $(t_j)_{j=i-d+1}^{i+d}$ that form a nondecreasing sequence with $t_i < t_{i+1}$. The curve is a weighted average of two curves of degree $d-1$,

$$\boldsymbol{p}_{i,d}(t) = \frac{t_{i+1}-t}{t_{i+1}-t_i}\boldsymbol{p}_{i-1,d-1}(t) + \frac{t-t_i}{t_{i+1}-t_i}\boldsymbol{p}_{i,d-1}(t). \tag{1.25}$$

Because of the assumption $t_i < t_{i+1}$ we never get division by zero in (1.25). The two curves of degree $d-1$ are obtained by forming similar convex combinations of curves of degree $d-2$. For example,

$$\boldsymbol{p}_{i,d-1}(t) = \frac{t_{i+2}-t}{t_{i+2}-t_i}\boldsymbol{p}_{i-1,d-2}(t) + \frac{t-t_i}{t_{i+2}-t_i}\boldsymbol{p}_{i,d-2}(t),$$

and again the condition $t_i < t_{i+1}$ saves us from dividing by zero. At the lowest level we have d line segments that are determined directly from the control points,

$$\boldsymbol{p}_{j,1}(t) = \frac{t_{j+d}-t}{t_{j+d}-t_j}\boldsymbol{c}_{j-1} + \frac{t-t_j}{t_{j+d}-t_j}\boldsymbol{c}_j$$

for $j = i-d+1, \ldots, i$. The denominators in this case are $t_{i+1}-t_{i-d+1}, \ldots, t_{i+d}-t_i$, all of which are positive since the knots are nondecreasing with $t_i < t_{i+1}$. As long as t is restricted to the interval $[t_i, t_{i+1}]$, all the operations involved in computing $\boldsymbol{p}_{i,d}(t)$

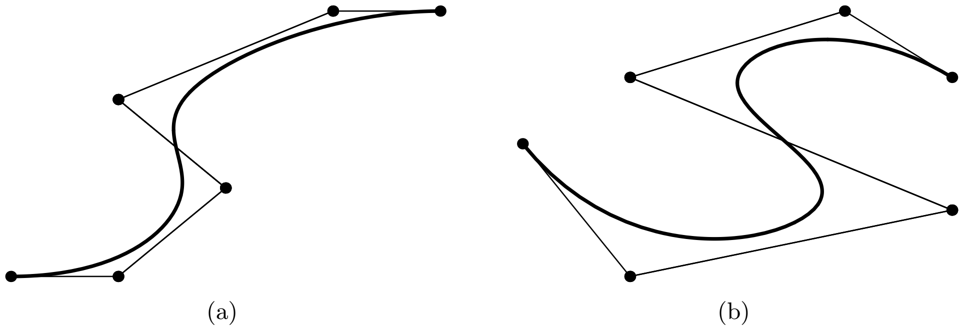


Figure 1.22. Two cubic spline curves, both with knots $\boldsymbol{t} = (0,0,0,0,1,2,3,3,3,3)$.

are convex combinations. The complete computations are summarised in the following algorithm.

Algorithm 1.3. *Let d be a positive integer and let the $d+1$ points $(\boldsymbol{c}_j)_{j=i-d}^{i}$ be given together with the $2d$ knots $\boldsymbol{t} = (t_j)_{j=i-d+1}^{i+d}$. The point $\boldsymbol{p}_{i,d}(t)$ on the spline curve $\boldsymbol{p}_{i,d}$ of degree d is determined by the following computations. First set $\boldsymbol{p}_{j,0}(t) = \boldsymbol{c}_j$ for $j = i-d$, $i-d+1, \ldots, i$ and then compute*

$$\boldsymbol{p}_{j,r}(t) = \frac{t_{j+d-r+1} - t}{t_{j+d-r+1} - t_j}\boldsymbol{p}_{j-1,r-1}(t) + \frac{t - t_j}{t_{j+d-r+1} - t_j}\boldsymbol{p}_{j,r-1}(t) \tag{1.26}$$

for $j = i-d+r, \ldots, i$ and $r = 1, 2, \ldots, d$.

A spline curve of degree d with n control points $(\boldsymbol{c}_i)_{i=1}^{n}$ and knots $(t_i)_{i=2}^{n+d}$ is given by

$$\boldsymbol{f}(t) = \begin{cases} \boldsymbol{p}_{d+1,d}(t) & t \in [t_{d+1}, t_{d+2}], \\ \boldsymbol{p}_{d+2,d}(t), & t \in [t_{d+2}, t_{d+3}]; \\ \quad\vdots & \quad\vdots \\ \boldsymbol{p}_{n,d}(t), & t \in [t_n, t_{n+1}], \end{cases}$$

where as before it is assumed that the knots are nondecreasing and in addition that $t_i < t_{i+1}$ for $i = d+1, \ldots, n$. Again we can express $\boldsymbol{f}$ in terms of the piecewise constant functions given by (1.18),

$$\boldsymbol{f}(t) = \sum_{i=d+1}^{n} \boldsymbol{p}_{i,d}(t) B_{i,0}(t). \tag{1.27}$$

It turns out that spline curves of degree d have continuous derivatives up to order $d-1$, see Section 3.2.3 in Chapter 3.

Figure 1.22 shows two examples of cubic spline curves with control points taken from the two Bézier curves of degree five in Figure 1.15. Again we note that the curves behave like Bézier curves at the ends because there are four identical knots at each end.

1.5.4 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

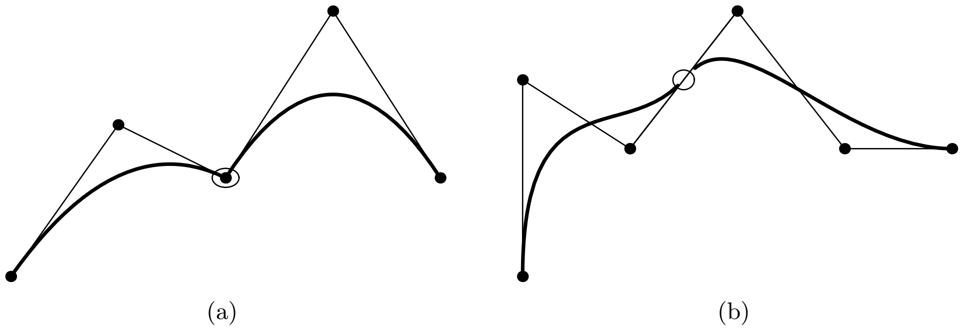


Figure 1.23. A quadratic spline with a double knot at the circled point (a) and a cubic spline with a double knot at the circled point (b).

that it is possible to smoothly join together neighbouring segments. We will study this in much more detail in Chapter 3, but will take the time to state the exact smoothness properties of spline curves here.

Theorem 1.4. *Suppose that the number t_{i+1} occurs m times among the knots $(t_j)_{j=i-d}^{m+d}$, with m some integer bounded by $1 \leq m \leq d+1$, i.e.,*

$$t_i < t_{i+1} = \cdots = t_{i+m} < t_{i+m+1}.$$

Then the spline function $\boldsymbol{f}(t) = \boldsymbol{p}_{i,d,1}(t)B_{i,0}(t) + \boldsymbol{p}_{i+m,d,1}(t)B_{i+m,0}(t)$ has continuous derivatives up to order $d-m$ at the join t_{i+1}.

This theorem introduces a generalisation of our construction of spline curves by permitting $t_{i+1}, \ldots, t_{i+m}$ to coalesce, but if we assume that $m = 1$ the situation corresponds to the construction above. Theorem 1.4 tells us that in this standard case the spline curve $\boldsymbol{f}$ will have d continuous derivatives at the join t_{i+1}: namely $\boldsymbol{f}$, $\boldsymbol{f}'$, …, $\boldsymbol{f}^{d-1}$ will all be continuous at t_{i+1}. This means that if the knots are all distinct, then a linear spline will be continuous, a quadratic spline will also have a continuous first derivative, while for a cubic spline even the second derivative will be continuous. Examples of spline curves with this maximum smoothness can be found above.

What happens when $m > 1$? Theorem 1.4 tells us that each time we add a knot at t_{i+1} the number of continuous derivatives is reduced by one. So a quadratic spline will in general only be continuous at a double knot, whereas a cubic spline will be continuous and have a continuous derivative at a double knot.

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). With a spline curve of degree d this can be done by letting the appropriate knot occur d times. We will see many examples of how the multiplicity of the knots influence the smoothness of a spline in later chapters.

Two examples of spline curves with reduced smoothness are shown in Figure 1.23. 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.

1.6 Representing spline curves in terms of basis functions

In Section 1.4 we saw that a Bézier curve $\boldsymbol{g}$ of degree d with control points $(\boldsymbol{c}_i)_{i=0}^d$ can be written as a linear combination of the Bernstein polynomials $\{b_{i,d}\}_{i=0}^d$ with the control points as coefficients, see (1.14). In this section we want to develop a similar representation for spline curves.

If we have n control points $(\boldsymbol{c}_i)_{i=1}^n$ and the $n+d-1$ knots $\boldsymbol{t} = (t_i)_{i=2}^{n+d}$ for splines of degree d; we have seen that a typical spline can be written

$$\boldsymbol{f}(t) = \sum_{i=d+1}^{n} \boldsymbol{p}_{i,d}(t) B_{i,0}(t), \qquad t \in [t_{d+1}, t_{n+1}], \tag{1.28}$$

where $\{B_{i,0}\}_{i=d+1}^n$ are given by (1.18). When this representation was introduced at the end of Section 1.5.3 we assumed that $t_{d+1} < t_{d+2} < \cdots < t_{n+1}$ (although the end knots were allowed to coincide). To accommodate more general forms of continuity, we know from Theorem 1.4 that we must allow some of the interior knots to coincide as well. If for example $t_i = t_{i+1}$ for some i with $d+1 < i < n+1$, then the corresponding segment $\boldsymbol{p}_{i,d}$ is completely redundant and (1.25) does not make sense since we get division by zero. This is in fact already built into the representation in (1.28), since $B_{i,0}(t)$ is identically zero in this case, see (1.18). A more explicit definition of $B_{i,0}$ makes this even clearer,

$$B_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1}, \\ 0, & t < t_i \text{ or } t \ge t_{i+1}, \\ 0, & t_i = t_{i+1}. \end{cases} \tag{1.29}$$

The representation (1.28) is therefore valid even if some of the knots occur several times. The only complication is that we must be careful when we expand out $\boldsymbol{p}_{i,d}$ according to (1.25) as this will give division by zero if $t_i = t_{i+1}$. One might argue that there should be no need to apply (1.25) if $t_i = t_{i+1}$ since the result is zero anyway. However, in theoretical developments it is convenient to be able to treat all the terms in (1.28) similarly, and this may then lead to division by zero. It turns out though that this problem can be circumvented quite easily by giving an appropriate definition of 'division by zero' in this context, see below.

Let us now see how $\boldsymbol{f}$ can be written more directly in terms of the control points. By making use of (1.25) we obtain

$$\begin{aligned} \boldsymbol{f}(t) &= \sum_{i=d+1}^{n} \Big(\frac{t - t_i}{t_{i+1} - t_i} \boldsymbol{p}_{i,d-1}(t) B_{i,0}(t) + \frac{t_{i+1} - t}{t_{i+1} - t_i} \boldsymbol{p}_{i-1,d-1}(t) B_{i,0}(t) \Big) \\ &= \sum_{i=d+1}^{n-1} \Big(\frac{t - t_i}{t_{i+1} - t_i} B_{i,0}(t) + \frac{t_{i+2} - t}{t_{i+2} - t_{i+1}} B_{i+1,0}(t) \Big) \boldsymbol{p}_{i,d-1}(t) + \\ &\qquad \frac{t_{d+2} - t}{t_{d+2} - t_{d+1}} B_{d+1,0}(t) \boldsymbol{p}_{d,d-1}(t) + \frac{t - t_n}{t_{n+1} - t_n} B_{n,0}(t) \boldsymbol{p}_{n,d-1}(t). \end{aligned} \tag{1.30}$$

This is a typical situation where we face the problem of division by zero if $t_i = t_{i+1}$ for some i. The solution is to declare that 'anything divided by zero is zero' since we know that if $t_i = t_{i+1}$ the answer should be zero anyway.

In (1.30) we have two 'boundary terms' that complicate the expression. But since t is assumed to lie in the interval $[t_{d+1}, t_{n+1}]$ we may add the expression

$$\frac{t-t_d}{t_{d+1}-t_d} B_{d,0}(t)\boldsymbol{p}_{d,d-1}(t) + \frac{t_{n+2}-t}{t_{n+1}-t_{n+1}} B_{n+1,0}(t)\boldsymbol{p}_{n,d-1}(t)$$

which is identically zero as long as t is within this $[t_{d+1}, t_{n+1}]$. By introducing the functions

$$B_{i,1}(t) = \frac{t-t_i}{t_{i+1}-t_i} B_{i,0}(t) + \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} B_{i+1,0}(t) \tag{1.31}$$

for $i = d, \ldots, n$, we can then write $\boldsymbol{f}$ as

$$\boldsymbol{f}(t) = \sum_{i=d}^{n} \boldsymbol{p}_{i,d-1}(t) B_{i,1}(t).$$

This illustrates the general strategy: Successively apply the relations in (1.26) in turn and rearrange the sums until we have an expression where the control points appear explicitly. The functions that emerge are generalisations of $B_{i,1}$ and can be defined recursively by

$$B_{i,r}(t) = \frac{t-t_i}{t_{i+r}-t_i} B_{i,r-1}(t) + \frac{t_{i+r+1}-t}{t_{i+r+1}-t_i} B_{i+1,r-1}(t), \tag{1.32}$$

for $r = 1, 2, \ldots, d$, starting with $B_{i,0}$ as defined in (1.18). Again we use the convention that 'anything divided by zero is zero'. It follows by induction that $B_{i,r}(t)$ is identically zero if $t_i = t_{i+r+1}$ and $B_{i,r}(t) = 0$ if $t < t_i$ or $t > t_{i+r+1}$, see exercise 7.

To prove by induction that the functions defined by the recurrence (1.32) appear in the process of unwrapping all the averaging in (1.26), we consider a general step. Suppose that after $r-1$ applications of (1.26) we have

$$\boldsymbol{f}(t) = \sum_{i=d+2-r}^{n} \boldsymbol{p}_{i,d-r+1}(t) B_{i,r-1}(t).$$

One more application yields

$$\begin{aligned}
\boldsymbol{f}(t) &= \sum_{i=d+2-r}^{n} \Big(\frac{t_{i+r}-t}{t_{i+r}-t_i} \boldsymbol{p}_{i-1,d-r}(t) B_{i,r-1}(t) + \frac{t-t_i}{t_{i+r}-t_i} \boldsymbol{p}_{i,d-r}(t) B_{i,r-1}(t) \Big) \\
&= \sum_{i=d+2-r}^{n-1} \Big(\frac{t-t_i}{t_{i+r}-t_i} B_{i,r-1}(t) + \frac{t_{i+r+1}-t}{t_{i+r+1}-t_{i+1}} B_{i+1,r-1}(t) \Big) \boldsymbol{p}_{i,d-r}(t) + \\
&\qquad \frac{t_{d+2}-t}{t_{d+2}-t_{d+2-r}} B_{d+2-r,r-1}(t)\boldsymbol{p}_{d+1-r,d-r}(t) + \frac{t-t_n}{t_{n+r}-t_n} B_{n,r-1}(t)\boldsymbol{p}_{n,d-r}(t).
\end{aligned}$$

Just as above we can include the boundary terms in the sum by adding

$$\frac{t-t_{d+1-r}}{t_{d+1}-t_{d+1-r}} B_{d+1-r,r-1}(t)\boldsymbol{p}_{d+1-r,d-r}(t) + \frac{t_{n+r+1}-t}{t_{n+r+1}-t_{n+1}} B_{n+1,r-1}(t)\boldsymbol{p}_{n,d-r}(t)$$

which is zero since $B_{i,r-1}(t)$ is zero when $t < t_i$ or $t > t_{i+r}$. The result is that

$$\boldsymbol{f}(t) = \sum_{i=d+1-r}^{n} \boldsymbol{p}_{i,d-r}(t) B_{i,r}(t).$$

After $d-1$ steps we have $\boldsymbol{f}(t) = \sum_{i=2}^{n} \boldsymbol{p}_{i,1,d-1}(t) B_{i,d-1}(t)$. In the last application of (1.26) we recall that $\boldsymbol{p}_{j,0}(t) = \boldsymbol{c}_j$ for $j = i-d, \ldots, i$. After rearranging the sum and adding zero terms as before we obtain

$$\boldsymbol{f}(t) = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d}(t).$$

But note that in this final step we need two extra knots, namely t_1 and t_{n+d+1} which are used by $B_{1,d-1}$ and $B_{n+1,d-1}$, and therefore also by $B_{1,d}$ and $B_{n,d}$. The value of the spline in the interval $[t_{d+1}, t_{n+1}]$ is independent of these knots, but it is customary to demand that $t_1 \le t_2$ and $t_{n+d+1} \ge t_{n+d}$ to ensure that the complete knot vector $\boldsymbol{t} = (t_i)_{i=1}^{n+d+1}$ is a nondecreasing sequence of real numbers.

The above discussion can be summarised in the following theorem.

Theorem 1.5. *Let $(\boldsymbol{c}_i)_{i=1}^{n}$ be a set of control points for a spline curve $\boldsymbol{f}$ of degree d, with nondecreasing knots $(t_i)_{i=1}^{n+d+1}$,*

$$\boldsymbol{f}(t) = \sum_{i=d+1}^{n} \boldsymbol{p}_{i,d}(t) B_{i,0}(t)$$

where $\boldsymbol{p}_{i,d}$ is given recursively by

$$\boldsymbol{p}_{i,d-r+1}(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i} \boldsymbol{p}_{i-1,d-r}(t) + \frac{t - t_i}{t_{i+r} - t_i} \boldsymbol{p}_{i,d-r}(t) \tag{1.33}$$

for $i = d-r+1, \ldots, n$, and $r = d, d-1, \ldots, 1$, while $\boldsymbol{p}_{i,0}(t) = \boldsymbol{c}_i$ for $i = 1, \ldots, n$. The functions $\{B_{i,0}\}_{i=d+1}^{n}$ are given by

$$B_{i,0}(t) = \begin{cases} 1, & t_i \le t < t_{i+1}, \\ 0, & \textit{otherwise}. \end{cases} \tag{1.34}$$

The spline $\boldsymbol{f}$ can also be written

$$\boldsymbol{f}(t) = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d}(t) \tag{1.35}$$

where $B_{i,d}$ is given by the recurrence relation

$$B_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i} B_{i,d-1}(t) + \frac{t_{i+1+d} - t}{t_{i+1+d} - t_{i+1}} B_{i+1,d-1}(t). \tag{1.36}$$

In both (1.33) and (1.36) possible divisions by zero are resolved by the convention that 'anything divided by zero is zero'. The function $B_{i,d} = B_{i,d,\boldsymbol{t}}$ is called a B-spline of degree d (with knots $\boldsymbol{t}$).

B-splines have many interesting and useful properties and in the next chapter we will study these functions in detail.

1.7 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

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

where d is the polynomial degree, $(\boldsymbol{a}_i)_{i=0}^{d}$ are the coefficients and $\{F_i\}_{i=0}^{d}$ are the basis polynomials. The difference 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. Although all three methods are capable of generating any polynomial curve, their differences mean that they lead to different representations of polynomials. For our purposes Bézier and spline curves are preferable since they can be constructed by forming repeated convex combinations. As we argued in Section 1.1, this should ensure that the 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, and as we saw in Section 1.4.3 it 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 be shifted 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.

We were led to splines and B-splines in our search for approximation methods based on convex combinations. The method which uses given points $(\boldsymbol{c}_i)_{i=1}^{n}$ as control points for a spline as in

$$\boldsymbol{f}(t) = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d}(t) \tag{1.37}$$

is often referred to as *Schoenberg's variation diminishing spline approximation*. This is a widely used approximation method that we will study in detail in Section 5.4, and because of the intuitive relation between the spline and its control points the method is often used in interactive design of spline curves. However, there are many other spline approximation methods. For example, we may approximate certain given points $(\boldsymbol{b}_i)_{i=1}^{m}$ by a spline curve that passes through these points, or we may decide that we want a spline curve that approximates these points in such a way that some measure of the error is as small as possible. To solve these kinds of problems, we are faced with three challenges: we must pick a suitable polynomial degree and an appropriate set of knots, and then determine control points so that the resulting spline curve satisfies our chosen criteria. Once this is accomplished we can compute points on the curve by Algorithm 1.3 and store it by storing the degree, the knots and the control points. We are going to study various spline approximation methods of this kind in Chapter 5.

But before turning to approximation with splines, we need to answer some basic questions: Exactly what functions can be represented as linear combinations of B-splines as in (1.37)? Is a representation in terms of B-splines unique, or are there several choices of control points that result in the same spline curve? These and many other questions will be answered in the next two chapters.

Exercises for Chapter 1

1.1 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 $\mathbb{CH}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n)$ to denote the convex hull of the points $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$.

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

$$\tilde{\boldsymbol{c}} = (1-\lambda)\boldsymbol{c}_1 + \lambda \boldsymbol{c}_2 \tag{1.38}$$

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

b) It could be that $\mathbb{CH}(\boldsymbol{c}_1, \boldsymbol{c}_2, \boldsymbol{c}_3)$ 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 $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ gives rise to a convex combination on the form (1.38). Hint: Show that if $\boldsymbol{c}$ is a convex combination of the three points, then we can write

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

where $\tilde{\boldsymbol{c}}$ is some convex combination of $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$. 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 $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ 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 $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$. Use the induction hypothesis and show that $\mathbb{B}$ contains any point on a straight line that connects $\boldsymbol{c}_n$ and an arbitrary point in $\mathbb{CH}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n-1})$.

d) From what we have found in (c) it is not absolutely clear that any convex set $\mathbb{B}$ that contains $\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n$ also contains *all* convex combinations of the points. To settle this show that any point $\boldsymbol{c}$ in $\mathbb{CH}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_n)$ can be written $\boldsymbol{c} = \lambda\tilde{\boldsymbol{c}} + (1 - \lambda)\boldsymbol{c}_n$ for some λ in $[0, 1]$ and some point $\tilde{\boldsymbol{c}}$ in $\mathbb{CH}(\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{n-1})$. Hint: Use a trick similar to that in (b).

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

1.2 Show that the interpolatory polynomial curve $\boldsymbol{q}_{0,d}(t)$ given by (1.4) can be written as in (1.5) with $\ell_{i,d}$ given by (1.6).

1.3 Implement Algorithm 1.1 in a programming language of your choice. Test the code by interpolating points on a semicircle and plot the results. Perform four tests, with 3, 7, 11 and 15 uniformly sampled points. Experiment with the choice of parameter values (t_i) and try to find both some good and some bad approximations.

1.4 Implement Algorithm 1.2 in your favourite programming language. Test the program on the same data as in exercise 3.

1.5 In this exercise we are going to write a program for evaluating spline functions. Use whatever programming language you prefer.

a) Implement Algorithm 1.3 in a procedure that takes as input an integer d (the degree), $d + 1$ control points in the plane, $2d$ knots and a parameter value t.

b) If we have a complete spline curve $\boldsymbol{f} = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d}$ with knots $\boldsymbol{t} = (t_i)_{i=1}^{n+d+1}$ that we want to evaluate at t we must make sure that the correct control points and knots are passed to the routine in (a). If

$$t_\mu \leq t < t_{\mu+1} \tag{1.39}$$

then $(\boldsymbol{c}_i)_{i=\mu-d}^{\mu}$ and $(t_i)_{i=\mu-d+1}^{\mu+d}$ are the control points and knots needed in (a). Write a procedure which takes as input all the knots and a value t and gives as output the integer μ such that (1.39) holds.

c) Write a program that plots a spline function by calling the two routines from (a) and (b). Test your program by picking control points from the upper half of the unit circle and plotting the resulting spline curve. Use cubic splines and try with $n = 4$, $n = 8$ and $n = 16$ control points. Use the knots $\boldsymbol{t} = (0, 0, 0, 0, 1, 1, 1, 1)$ when $n = 4$ and add the appropriate number of knots between 0 and 1 when n is increased. Experiment with the choice of interior knots when $n = 8$ and $n = 16$. Is the resulting curve very dependent on the knots?

1.6 Show that a quadratic spline is continuous and has a continuous derivative at a single knot.

1.7 Show by induction that $B_{i,d}$ depends only on the knots t_i, t_{i+1}, $\ldots$, t_{i+d+1}. Show also that $B_{i,d}(t) = 0$ if $t < t_i$ or $t > t_{i+d+1}$.

CHAPTER 2

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 start with a detailed study of the most basic properties of B-splines, illustrated by examples and figures in Section 2.1, and in Section 2.2 we formally define spline functions and spline curves. In Section 2.3 we give a matrix representation of splines and B-splines, and this representation is the basis for our development of much of the theory in later chapters.

2.1 Some simple consequences of the recurrence relation

We saw in Theorem 1.5 that a degree d spline curve $\boldsymbol{f}$ can be constructed from n control points $(\boldsymbol{c}_i)_{i=1}^n$ and $n+d+1$ knots $(t_i)_{i=1}^{n+d+1}$ and written as

$$\boldsymbol{f} = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d},$$

where $\{B_{i,d}\}_{i=1}^n$ are B-splines. In this section we will explore B-splines by considering a number of examples, and deducing some of their most basic properties. For easy reference we start by recording the definition of B-splines. Since we will mainly be working with functions in this chapter, we use x as the independent variable.

Definition 2.1. *Let d be a nonnegative integer and let $\boldsymbol{t} = (t_j)$, the* knot vector *or* knot sequence, *be a nondecreasing sequence of real numbers of length at least $d+2$. The jth B-spline of degree d with knots $\boldsymbol{t}$ is defined by*

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

for all real numbers x, with

$$B_{j,0,\boldsymbol{t}}(x) = \begin{cases} 1, & \textit{if } t_j \le x < t_{j+1}; \\ 0, & \textit{otherwise.} \end{cases} \tag{2.2}$$

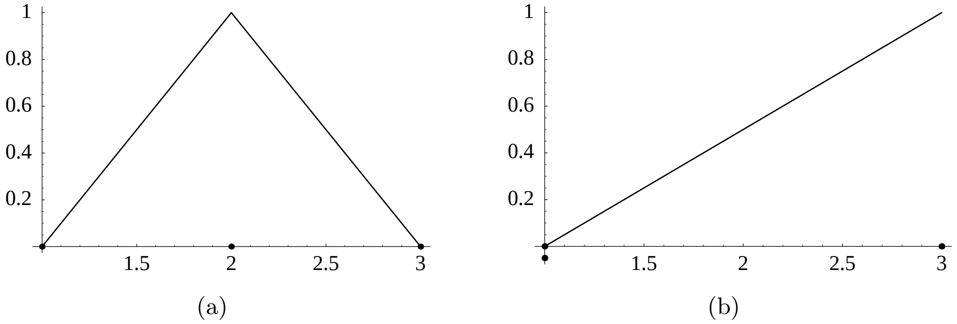


Figure 2.1. A linear B-spline with simple knots (a) and double knots (b).

Here, the convention is assumed that '0/0 = 0'. When there is no chance of ambiguity some of the subscripts will be dropped and the B-spline written as either $B_{j,d}$, $B_{j,\boldsymbol{t}}$, 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, three are also called *simple, double* and *triple* knots.

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

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

which we made use of in Chapter 1. This is true by definition for $d = 0$, and 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

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

where

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

for all x.

The following example shows that linear B-splines are quite simple.

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

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

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 and t_{j+1} are identical; then the first linear piece is identically zero since $B_{j,0} = 0$, 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.

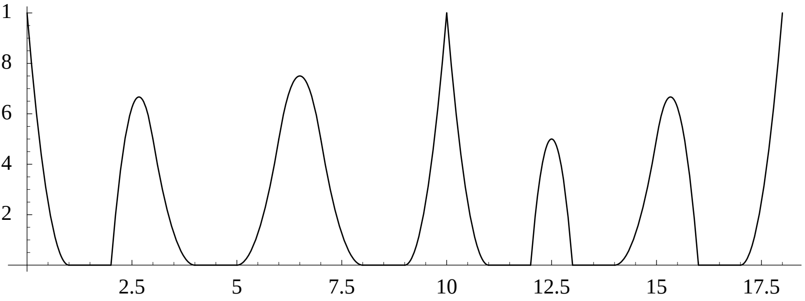


Figure 2.2. From left to right we see the quadratic B-splines $B(x|0,0,0,1)$, $B(x|2,2,3,4)$, $B(x|5,6,7,8)$, $B(x|9,10,10,11)$, $B(x|12,12,13,13)$, $B(x|14,15,16,16)$, and $B(x|17,18,18,18)$.

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.

To emphasise this local nature of B-splines, the notation $B_{j,d}(x) = B(x|t_j,\ldots,t_{j+d+1})$ is sometimes useful. For example, if $d \geq 2$ and if we set $(t_j,t_{j+1},\ldots,t_{j+d},t_{j+d+1}) = (a,b,\ldots,c,d)$, then (2.1) can be written

$$B(x|a,b,\ldots,c,d)(x) = \frac{x-a}{c-a}B(x|a,b,\ldots,c) + \frac{d-x}{d-b}B(x|b,\ldots,c,d). \tag{2.4}$$

Example 2.3 (Quadratic B-splines)**.** Using the zero convention and (2.4) we find

1. $B(x|0,0,0,1) = (1-x)B(x|0,0,1) = (1-x)^2B(x|0,1)$.

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

3. $B(x|0,1,2,3) = \frac{x^2}{2}B(x|0,1) + \left(\frac{3}{4}-(x-\frac{3}{2})^2\right)B(x|1,2) + \frac{(3-x)^2}{2}B(x|2,3)$.

4. $B(x|0,1,1,2) = x^2B(x|0,1) + (2-x)^2B(x|1,2)$.

5. $B(x|0,0,1,1) = 2x(1-x)B(x|0,1)$.

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

7. $B(x|0,1,1,1) = x^2B(x|0,1)$.

Translates (see (2.6)) of these functions are shown in Figure 2.2. Note that the B-spline $B(x|0,1,2,3)$ consists of three nonzero polynomial pieces, but that in general the number of nonzero pieces depends on the multiplicity of the knots. For example, the functions $B(x|0,0,0,1)$ and $B(x|0,0,1,1)$ consist of only one nonzero piece. Figure 2.2 illustrates these smoothness properties of B-splines: At a single knot a quadratic B-spline is continuous and has a continuous derivative, at a double knot it is continuous, while at a triple knot it is discontinuous.

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

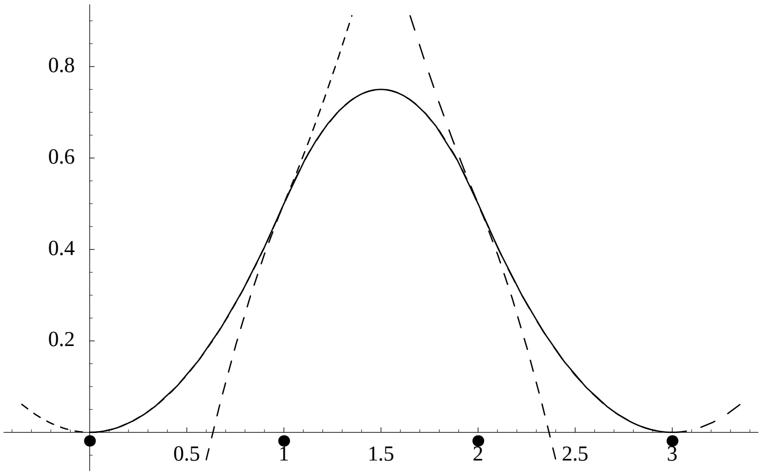


Figure 2.3. The different polynomial pieces of a quadratic B-spline.

By applying the recurrence relation (2.1) twice we obtain an explicit expression for a generic quadratic B-spline,

$$\begin{aligned}B_{j,2}(x) &= \frac{x-t_j}{t_{j+2}-t_j}\Big[\frac{x-t_j}{t_{j+1}-t_j}B_{j,0}(x)+\frac{t_{j+2}-x}{t_{j+2}-t_{j+1}}B_{j+1,0}(x)\Big]\\ &\quad+\frac{t_{j+3}-x}{t_{j+3}-t_{j+1}}\Big[\frac{x-t_{j+1}}{t_{j+2}-t_{j+1}}B_{j+1,0}(x)+\frac{t_{j+3}-x}{t_{j+3}-t_{j+2}}B_{j+2,0}(x)\Big]\\ &=\frac{(x-t_j)^2}{(t_{j+2}-t_j)(t_{j+1}-t_j)}B_{j,0}(x)+\frac{(t_{j+3}-x)^2}{(t_{j+3}-t_{j+1})(t_{j+3}-t_{j+2})}B_{j+2,0}(x)\\ &\quad+\Big(\frac{(x-t_j)(t_{j+2}-x)}{(t_{j+2}-t_j)(t_{j+2}-t_{j+1})}+\frac{(t_{j+3}-x)(x-t_{j+1})}{(t_{j+3}-t_{j+1})(t_{j+2}-t_{j+1})}\Big)B_{j+1,0}(x).\end{aligned} \tag{2.5}$$

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

Figure 2.4 shows some cubic B-splines. The middle B-spline, $B(x|9,10,11,12,13)$, has simple knots and its second derivative is therefore continuous for all real numbers x, including the knots. 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.

Before considering the next example we show that B-splines possess a property called *translation invariance*. Mathematically this is expressed by the formula

$$B(x+y|t_j+y,\dots,t_{j+d+1}+y)=B(x|t_j,\dots,t_{j+d+1})\quad x,y\in\mathbb{R}. \tag{2.6}$$

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

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

so equation (2.6) 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 (2.6) the first coefficient $(x-t_j)/(t_{j+d}-t_j)$ can be written

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

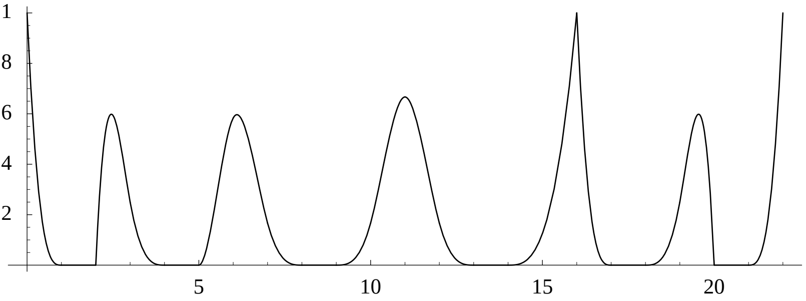


Figure 2.4. From left to right we see the cubic B-splines $B(x|0,0,0,0,1)$, $B(x|2,2,2,3,4)$, $B(x|5,5,6,7,8)$, $B(x|9,10,11,12,13)$, $B(x|14,16,16,16,17)$, $B(x|18,19,20,20,20)$, and $B(x|21,22,22,22,22)$.

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 (2.6) holds for all polynomial degrees.

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

$$M_d(x) = B_{0,d}(x) = B(x|0,1,\cdots,d+1), \quad x \in \mathbb{R}. \tag{2.7}$$

The functions M_d are also called *cardinal B-splines*. On this knot vector all B-splines can be written as translates of the function M_d. Using (2.6) we have

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

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

$$M_d(x) = \frac{x}{d} M_{d-1}(x) + \frac{d+1-x}{d} M_{d-1}(x-1). \tag{2.8}$$

Using this recurrence we can compute the first few uniform B-splines

$$\begin{aligned}
M_1(x) &= xM_0(x) + (2-x)M_0(x-1)\\
M_2(x) &= \frac{x^2}{2}M_0(x) + \Big(\frac{3}{4} - (x-\frac{3}{2})^2\Big)M_0(x-1) + \frac{(3-x)^2}{2}M_0(x-2)\\
M_3(x) &= \frac{x^3}{6}M_0(x) + \Big(\frac{2}{3} - \frac{1}{2}x(x-2)^2\Big)M_0(x-1)\\
&\quad + \Big(\frac{2}{3} - \frac{1}{2}(4-x)(x-2)^2\Big)M_0(x-2) + \frac{(4-x)^3}{6}M_0(x-3)
\end{aligned} \tag{2.9}$$

(compare with Examples 2.2 and 2.3). 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 $[9,13]$.

Example 2.5 (Bernstein polynomials)**.** The Bernstein polynomials that appeared in the representation of Bézier curves in Section 1.4 are special cases of B-splines. In fact it turns out that the jth Bernstein polynomial on the interval $[a,b]$ is (almost) given by

$$B_j^d(x) = B(x|\overbrace{a,\ldots,a}^{d+1-j},\overbrace{b,\ldots,b}^{j+1}), \quad \text{for } j = 0,\ \ldots,\ d.$$

The recurrence relation (2.4) now takes the form

$$\begin{aligned}
B_j^d(x) &= \frac{x-a}{b-a}B(x|\overbrace{a,\ldots,a}^{d+1-j},\overbrace{b,\ldots,b}^{j}) + \frac{b-x}{b-a}B(x|\overbrace{a,\ldots,a}^{d-j},\overbrace{b,\ldots,b}^{j+1})\\
&= \frac{x-a}{b-a}B_{j-1}^{d-1}(x) + \frac{b-x}{b-a}B_j^{d-1}(x).
\end{aligned} \tag{2.10}$$

This is also valid for $j = 0$ and $j = d$ if we define $B_j^{d-1} = 0$ for $j < 0$ and $j \geq d$. Using induction on d one can show the explicit formula

$$B_j^d(x) = \binom{d}{j} \left(\frac{x-a}{b-a}\right)^j \left(\frac{b-x}{b-a}\right)^{d-j} B(x|a,b), \quad \text{for } j = 0, 1, \ldots, d, \tag{2.11}$$

see exercise 5. These are essentially the Bernstein polynomials for the interval $[a, b]$, except that the factor $B(x|a, b)$ causes B_j^d to be zero outside $[a, b]$. To represent Bézier curves, it is most common to use the Bernstein polynomials on the interval $[0, 1]$ as in Section 1.4, i.e., with $a = 0$ and $b = 1$,

$$B_j^d(x) = \binom{d}{j} x^j (1-x)^{d-j} B(x|0,1) = b_{j,d}(x) B(x|0,1), \quad \text{for } j = 0, 1, \ldots, d; \tag{2.12}$$

here b_j^d is the jth Bernstein polynomial of degree d. For example, the quadratic Bernstein basis polynomials are given by

$$b_{0,2}(x) = (1-x)^2, \quad b_{1,2}(x) = 2x(1-x), \quad b_{2,2}(x) = x^2$$

which agrees with what we found in Chapter 1. These functions can also be recognised as the polynomial part of the special quadratic B-splines in (1), (5) and (7) in Example 2.3. For Bernstein polynomials on $[0, 1]$ the recurrence relation (2.10) takes the form

$$b_{j,d}(x) = x b_{j-1,d-1}(x) + (1-x) b_{j,d-1}(x), \quad j = 0, 1, \ldots, d. \tag{2.13}$$

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

1. Local knots. *The jth B-spline $B_{j,d}$ depends only on the knots $t_j, t_{j+1}, \ldots, t_{j+d+1}$.*

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*

$$B_{j,d}(x) = \sum_{k=j}^{j+d} B_{j,d}^k(x) B_{k,0}(x) \tag{2.14}$$

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

5. Special values. *If $z = t_{j+1} = \cdots = t_{j+d} < t_{j+d+1}$ 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 $t_j, \ldots, t_{j+d+1}$ then the derivatives of $B_{j,d}$ of order $0, 1, \ldots, d-m$ are all continuous at z.*

Proof. Properties 1–3 follow directly, by induction, from the recurrence relation, see exercise 3. 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 6 and property 6 will be proved in Chapter 3. ∎

2.2 Linear combinations of B-splines

In Theorem 1.5 we saw that 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.

2.2.1 Spline functions

The B-spline $B_{j,d}$ depends on the knots t_j, …, t_{j+1+d}. This means that if the knot vector is given by $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ 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

$$f = \sum_{j=1}^{n} c_j B_{j,d}, \tag{2.15}$$

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

Definition 2.7 (Spline functions)**.** *Let $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ 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 $\mathbb{S}_{d,\boldsymbol{t}}$ *defined by*

$$\mathbb{S}_{d,\boldsymbol{t}} = \operatorname{span}\{B_{1,d}, \dots, B_{n,d}\} = \Big\{ \sum_{j=1}^{n} c_j B_{j,d} \mid c_j \in \mathbb{R} \text{ for } 1 \leq j \leq n \Big\}.$$

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

As we shall see later, B-splines are linearly independent so $\mathbb{S}_{d,\boldsymbol{t}}$ is a linear space of dimension n.

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 2.8 (A linear spline)**.** Let $(x_i, y_i)_{i=1}^{m}$ be a set of data points with $x_i < x_{i+1}$ for $i = 1$, 2, …, $m-1$. On the knot vector

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

we consider the linear ($d = 1$) spline function

$$s(x) = \sum_{j=1}^{m} y_j B_{j,1}(x), \quad \text{for } x \in [x_1, x_m].$$

From Example 2.2 we see that s satisfies the interpolatory conditions

$$s(x_i) = \sum_{j=1}^{m} y_j B_{j,1}(x_i) = y_i, \quad i = 1, \dots, m-1 \tag{2.16}$$

since $B_{i-1,1}(x_i) = 1$ and all other B-splines are zero at x_i. At $x = x_m$ all the B-splines are zero according to Definition 2.1. But the limit of $B_m(x)$ when x tends to x_m from the left is 1. Equation (2.16) therefore

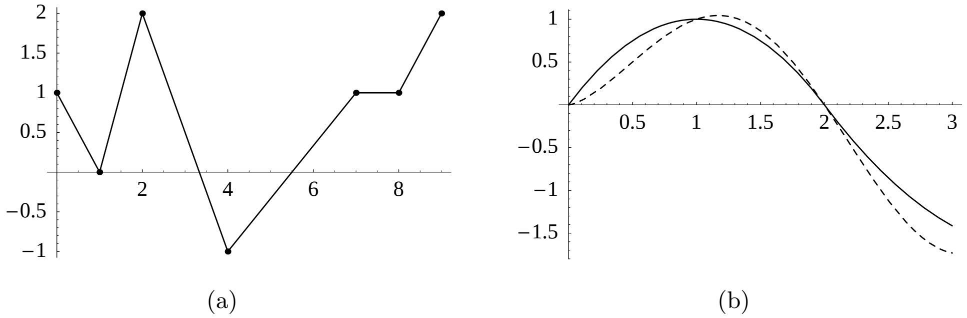


Figure 2.5. A linear spline interpolating data (a), and a quadratic spline (solid) that approximates $\sin(\pi x/2)$ (dashed).

also holds for $i = m$ if we take limits from the left at $x = x_m$. In addition s is linear on each subinterval $[t_\mu, t_{\mu+1})$ since

$$\begin{aligned} s(x) &= y_{\mu-1}B_{\mu-1,1}(x) + y_\mu B_{\mu,1}(x) \\ &= \frac{t_{\mu+1} - x}{t_{\mu+1} - t_\mu} y_{\mu-1} + \frac{x - t_\mu}{t_{\mu+1} - t_\mu} y_\mu. \end{aligned} \tag{2.17}$$

when x is in $[t_\mu, t_{\mu+1})$. It follows that s is the piecewise linear interpolant to the data. An example is shown in Figure 2.5 (a).

Example 2.9 (A quadratic spline)**.** Let $f : [a, b] \to \mathbb{R}$ 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 $\boldsymbol{t} = (t_j)_{j=1}^{n+3}$, where

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

We can then define the quadratic spline function

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

where

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

We note that

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

The function Qf is called the *Variation Diminishing Spline Approximation* to f of degree 2. As a particular instance of this approximation we approximate the function $f(x) = \sqrt{2}\sin\left(\frac{\pi}{2}x\right)$ on the interval $[0, 3]$. With

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

we obtain $(t_j^*)_{j=1}^{5} = (0, 1/2, 3/2, 5/2, 3)$ 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).

Example 2.10 (A cubic polynomial in Bernstein form)**.** On the knot vector

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

we consider the cubic spline function

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

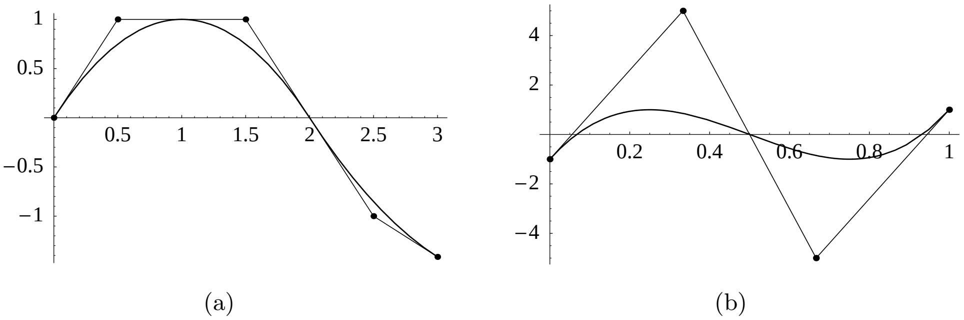


Figure 2.6. The quadratic spline from Example 2.9 with its control polygon (a) and the cubic Chebyshev polynomial with its control polygon (b).

In terms of the cubic Bernstein basis we have

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

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

Note that the knot vectors in the above examples all 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 2.11 (Control polygon for spline functions)**.** *Let $f = \sum_{j=1}^{n} c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{t}}$. The* control points *of f are the points with coordinates (t_j^*, c_j) for $j = 1, \ldots, n$, where*

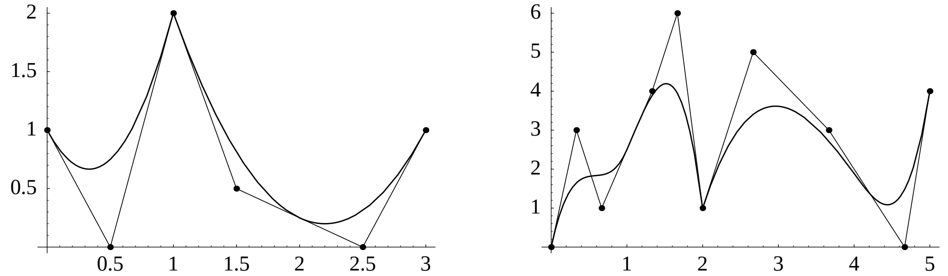
$$\boldsymbol{t}_j^* = \frac{t_{j+1} + \cdots + t_{j+d}}{d}$$

are the knot averages *of $\boldsymbol{t}$. 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 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 2.12. *Let $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for splines of degree d with $n \ge d + 1$, and let $f = \sum_{j=1}^{n} c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{t}}$. Then f has the following properties:*



(a) (b)

Figure 2.7. A quadratic spline function with knots $\boldsymbol{t} = (0,0,0,1,1,2,3,3,3)$ and B-spline coefficients $\boldsymbol{c} = (1,0,2,1/2,0,1)$ is shown in (a) while (b) shows a cubic spline function with knots $\boldsymbol{t} = (0,0,0,0,1,1,2,2,2,4,5,5,5,5)$ and B-spline coefficients $0,3,1,4,6,1,5,3,0,4)$.

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 $z = t_{j+1} = \cdots = t_{j+d} < t_{j+d+1}$ for some j in the range $1 \leq j \leq n$ then $f(z) = c_j$.*

3. *If z occurs m times in $\boldsymbol{t}$ then f has continuous derivatives of order $0, \ldots, d-m$ at z.*

Proof. This follows directly from Lemma 2.6. ∎

2.2.2 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 on one function value. We saw that one way to overcome this restriction was by representing the x- and y-components by two different functions,

$$\boldsymbol{f}(u) = \bigl(f_1(u), f_2(u)\bigr).$$

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

$$\mathbb{S}_{d,\boldsymbol{t}}^q = \Bigl\{\sum_{j=1}^{n} \boldsymbol{c}_j B_{j,d} \mid \boldsymbol{c}_j \in \mathbb{R}^q \text{ for } 1 \leq j \leq n\Bigr\}.$$

More precisely, an element $f = \sum_{j=1}^n c_j B_{j,d}$ of $\mathbb{S}_{d,\boldsymbol{t}}^q$ is called a spline vector function *or a* parametric spline curve *of degree d with knots $\boldsymbol{t}$, and $(\boldsymbol{c}_j)_{j=1}^n$ 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 reference we repeat the definition here.

Definition 2.14 (Control polygon for spline curves). *Let $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for splines of degree d, and let $\boldsymbol{f} = \sum_{j=1}^{n} \boldsymbol{c}_j B_{j,d}$ be a spline curve in $\mathbb{S}_{d,\boldsymbol{t},q}$ for $q \geq 2$. The* control polygon *of f is the piecewise linear function obtained by connecting neighbouring control points 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.

2.3 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 for 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 2.4 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, quadratic and cubic splines in three examples.

Example 2.15 (Vector representation of linear B-splines). Consider the case of linear B-splines with knots $\boldsymbol{t}$, 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

$$\begin{pmatrix} B_{\mu-1,1} & B_{\mu,1} \end{pmatrix} = \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_\mu} & \frac{x-t_\mu}{t_{\mu+1}-t_\mu} \end{pmatrix}. \tag{2.18}$$

Example 2.16 (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,

$$\begin{aligned}\begin{pmatrix} B_{\mu-2,2} & B_{\mu-1,2} & B_{\mu,2} \end{pmatrix} &= \begin{pmatrix} B_{\mu-1,1} & B_{\mu,1} \end{pmatrix} \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+1}-t_{\mu-1}} & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_\mu} & \frac{x-t_\mu}{t_{\mu+2}-t_\mu} \end{pmatrix} \\ &= \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_\mu} & \frac{x-t_\mu}{t_{\mu+1}-t_\mu} \end{pmatrix} \begin{pmatrix} \frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu-1}} & \frac{x-t_{\mu-1}}{t_{\mu+1}-t_{\mu-1}} & 0 \\ 0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_\mu} & \frac{x-t_\mu}{t_{\mu+2}-t_\mu} \end{pmatrix}.\end{aligned} \tag{2.19}$$

If these matrices are multiplied together the result would of course agree with that in Example 2.3. 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 2.4.

Example 2.17 (Matrix representation of cubic B-splines)**.** In the cubic case the only nonzero B-splines on $[t_\mu, t_{\mu+1})$ are $\{B_{j,3}\}_{j=\mu-3}^{\mu}$. Again it can be checked with Definition 2.1 that for x in this interval these B-splines may be written

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

The generalisation of this matrix notation to B-splines of arbitrary degree is straightforward.

Theorem 2.18. *Let $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ be a knot vector for B-splines of degree d, and let μ be an integer such that $t_\mu < t_{\mu+1}$ and $d+1 \le \mu \le n$. For each positive integer k with $k \le d$ define the matrix $\boldsymbol{R}_k^\mu(x) = \boldsymbol{R}_k(x)$ by*

$$
\boldsymbol{R}_k(x) = \begin{pmatrix}
\frac{t_{\mu+1}-x}{t_{\mu+1}-t_{\mu+1-k}} & \frac{x-t_{\mu+1-k}}{t_{\mu+1}-t_{\mu+1-k}} & 0 & \cdots & 0\\
0 & \frac{t_{\mu+2}-x}{t_{\mu+2}-t_{\mu+2-k}} & \frac{x-t_{\mu+2-k}}{t_{\mu+2}-t_{\mu+2-k}} & \cdots & 0\\
\vdots & \vdots & \ddots & \ddots & \vdots\\
0 & 0 & \cdots & \frac{t_{\mu+k}-x}{t_{\mu+k}-t_{\mu}} & \frac{x-t_{\mu}}{t_{\mu+k}-t_{\mu}}
\end{pmatrix}. \tag{2.20}
$$

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

$$
\boldsymbol{B}_d = \big(B_{\mu-d,d} \;\; B_{\mu-d+1,d} \;\; \ldots \;\; B_{\mu,d}\big) = \boldsymbol{R}_1(x)\boldsymbol{R}_2(x)\cdots\boldsymbol{R}_d(x). \tag{2.21}
$$

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

$$
f(x) = \boldsymbol{R}_1(x)\boldsymbol{R}_2(x)\cdots\boldsymbol{R}_d(x)\boldsymbol{c}_d, \tag{2.22}
$$

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

For $d = 0$ the usual convention of interpreting an empty product as 1 is assumed in equations (2.21) and (2.22).

Theorem 2.18 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 (2.22) will also be helpful for designing algorithms for computing $f(x)$. This is the theme of Section 2.4.

It should be emphasised that equation (2.21) 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

$$\begin{pmatrix} B^\mu_{\mu-d,d}(x) & B^\mu_{\mu-d+1,d}(x) & \ldots & B^\mu_{\mu,d}(x) \end{pmatrix} = \boldsymbol{R}^\mu_1(x)\boldsymbol{R}^\mu_2(x)\cdots\boldsymbol{R}^\mu_d(x),$$

see Lemma 2.6.

Likewise, equation (2.22) gives a representation of the polynomial f^μ that agrees with the spline f on the interval $[t_\mu, t_{\mu+1})$,

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

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 2.19 (Matrix representation of a quadratic spline)**.** In Example 2.9 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

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

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 $t_3 \le x < t_4$ so $s(x)$ is determined by (2.22) with $\mu = 3$ and $d = 2$. To determine the matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ 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 (2.22) 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 $t_4 \le x < t_5$ so $s(x)$ is determined by (2.22) with $\mu = 4$ and $d = 2$. To determine the matrices $\boldsymbol{R}_1$ and $\boldsymbol{R}_2$ 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

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

2.4 Algorithms for evaluating a spline

We originally introduced spline curves as the result of the geometric construction given in Algorithm 1.3 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.

Recall from Theorem 2.18 that a spline f of degree d with knots $\boldsymbol{t}$ and B-spline coefficients $\boldsymbol{c}$ can be expressed as

$$f(x) = \boldsymbol{R}_1(x)\cdots\boldsymbol{R}_d(x)\boldsymbol{c}_d \tag{2.23}$$

for any x in the interval $[t_\mu, t_{\mu+1})$. Here $\boldsymbol{c}_d = (c_{\mu-d}, \ldots, c_\mu)$ 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

$$\boldsymbol{c}_{k-1} = \boldsymbol{R}_k\boldsymbol{c}_k, \qquad \text{for } k = d,\, d-1,\, \ldots,\, 1. \tag{2.24}$$

Upon completion of this we have $f(x) = \boldsymbol{c}_0$ (note that $\boldsymbol{c}_0$ is a vector of dimension 1, i.e., a scalar). We see that this algorithm amounts to post-multiplying each matrix $\boldsymbol{R}_k$ by a vector which in component form becomes

$$(\boldsymbol{R}_k(x)\boldsymbol{c}_k)_j = \frac{t_{j+k}-x}{t_{j+k}-t_j}c_{k,j-1} + \frac{x-t_j}{t_{j+k}-t_j}c_{k,j} \tag{2.25}$$

for $j = \mu-k+1, \ldots, \mu$. This we immediately recognise as Algorithm 1.3.

The alternative algorithm accumulates the matrix products in (2.23) 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, and then multiply with the corresponding B-spline coefficients and sum up. Building up the B-splines is accomplished by starting with $\boldsymbol{B}_0(x)^T = 1$ and then performing the multiplications

$$\boldsymbol{B}_k(x)^T = \boldsymbol{B}_{k-1}(x)^T\boldsymbol{R}_k(x), \qquad k = 1, \ldots, d.$$

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

$$\boldsymbol{B}_d(x) = \bigl(B_{\mu-d,d}(x), \ldots, B_{\mu,d}(x)\bigr)^T.$$

We can then multiply with the B-spline coefficients and add up. We see that this algorithm amounts to pre-multiplying each matrix $\boldsymbol{R}_k$ by a row vector which in component form becomes

$$(\boldsymbol{B}_{k-1}(x))^T\boldsymbol{R}_k(x))_j = \frac{x-t_j}{t_{j+k}-t_j}B_{j,k-1}(x) + \frac{t_{j+1+k}-x}{t_{j+1+k}-t_{j+1}}B_{j+1,k-1}(x) \tag{2.26}$$

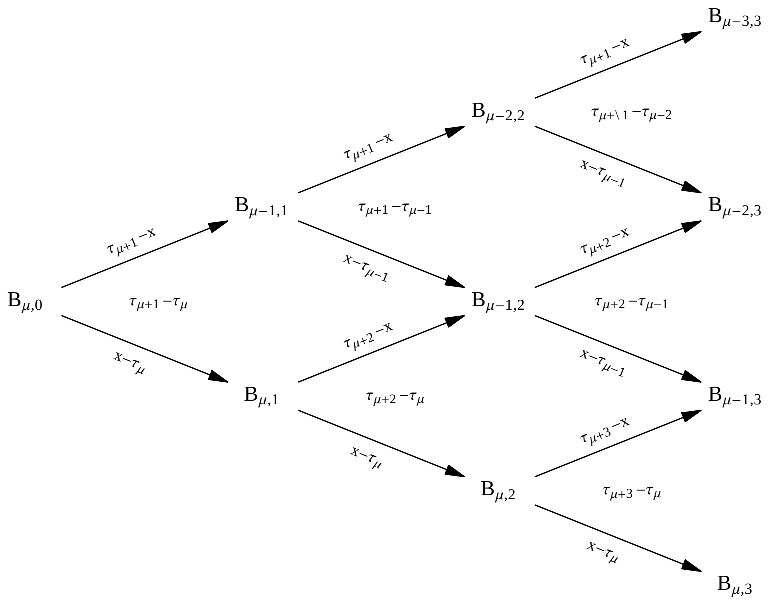


Figure 2.8. A triangular algorithm for computation of all the nonzero cubic B-splines at x.

for $j = \mu - k, \ldots, \mu$. Here it should be noted that $B_{\mu-k,k-1}(x) = B_{\mu+1,k-1}(x) = 0$ when $x \in [t_\mu, t_{\mu+1})$. For $j = \mu - k$, the first term on the right in (2.26) is therefore zero, and similarly, for $j = \mu$, the last term in (2.26) is zero.

Algorithm 2.20 (L). *Let the polynomial degree d, the $2d$ knots $t_{\mu-d+1} \le t_\mu < t_{\mu+1} \le t_{\mu+d}$, the B-spline coefficients $\boldsymbol{c}_d^{(0)} = \boldsymbol{c}_d = (c_j)_{j=\mu-d}^{\mu}$ of a spline f, and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of the products*

$$\boldsymbol{c}_{k-1} = \boldsymbol{R}_k(x)\boldsymbol{c}_k, \qquad k = d, \ldots, 1,$$

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

$$f(x) = \boldsymbol{c}_0.$$

Algorithm 2.21 (R). *Let the polynomial degree d, the knots $t_{\mu-d+1} \le t_\mu < t_{\mu+1} \le t_{\mu+d}$ and a number x in $[t_\mu, t_{\mu+1})$ be given and set $\boldsymbol{B}_0 = 1$. After evaluation of the products*

$$\boldsymbol{B}_k(x)^T = \boldsymbol{B}_{k-1}(x)^T \boldsymbol{R}_k(x), \qquad k = 1, \ldots, d,$$

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

$$\boldsymbol{B}_d(x) = \big(B_{\mu-d,d}(x), \ldots, B_{\mu,d}(x)\big)^T.$$

These algorithms have a simple triangular structure, just like Algorithm 1.3, see figures 2.9–2.8. Figure 2.9 shows how the value of a cubic spline can be computed at a point x, while Figure 2.8 shows the computation of all the nonzero B-splines at a point.

In Algorithms 2.20 and 2.21 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 19 discusses evaluation in such a case.

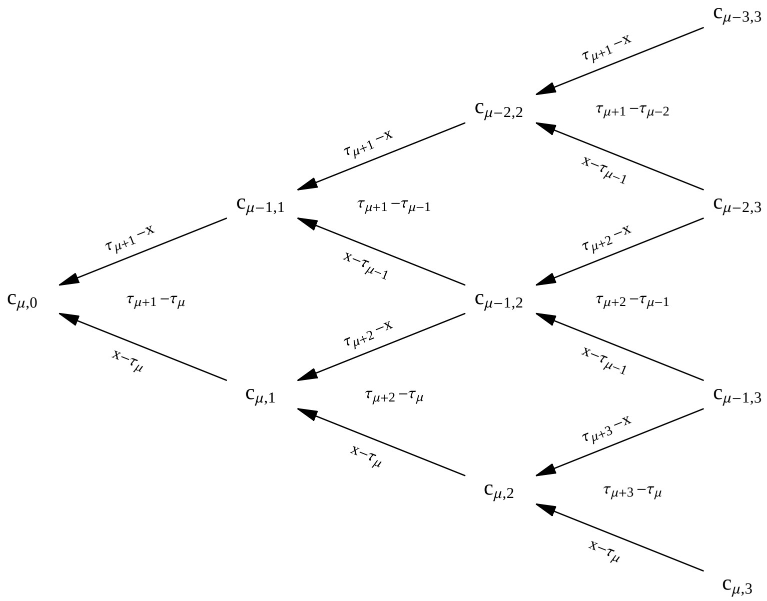


Figure 2.9. A triangular algorithm for computing the value of a cubic spline with B-spline coefficients $\boldsymbol{c}$ at $x \in [t_\mu, t_{\mu+1})$.

Exercises for Chapter 2

2.1 Show that

$$B(x|0,3,4,6) = \frac{1}{12}x^2 B(x|0,3) + \frac{1}{12}(-7x^2+48x-72)B(x|3,4) + \frac{1}{6}(6-x)^2 B(x|4,6).$$

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

a) $B(x|0,0,0,0,1)$ and $B(x|0,1,1,1,1)$

b) $B(x|0,1,1,1,2)$

2.3 Show that the B-spline $B_{j,d}$ satisfies properties 1–3 of Lemma 2.6.

2.4 Show that $B_{j,d}$ is a piecewise polynomial by establishing equation 2.14. Use induction on the degree d.

2.5 In this exercise we are going to establish some properties of the Bernstein polynomials.

a) Prove the differentiation formula

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

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

c) Show that

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

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

$$B(t_i|t_j,\ldots,t_{j+1+d}) = B(t_i|t_j,\ldots,t_{i-1},t_{i+1},\ldots,t_{j+1+d}) \tag{2.27}$$

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

b) Use the formula in (2.27) to compute the following values of a quadratic B-spline at the interior knots:

$$B_{j,2}(t_{j+1}) = \frac{t_{j+1}-t_j}{t_{j+2}-t_j}, \quad B_{j,2}(t_{j+2}) = \frac{t_{j+3}-t_{j+2}}{t_{j+3}-t_{j+1}}. \tag{2.28}$$

c) Prove property (5) of Lemma 2.6.

2.7 Prove the following formula using (2.4) and (2.11)

$$B(x|a,\overbrace{b,\ldots,b}^{d},c) = \frac{(x-a)^d}{(b-a)^d}B(x|a,b) + \frac{(c-x)^d}{(c-b)^d}B(x|b,c).$$

Show that this function is continuous at all real numbers.

2.8 Prove the following formulas by induction on d.

$$\begin{aligned} B(x|\overbrace{a,\ldots,a}^{d},b,c) &= \frac{x-a}{b-a}\sum_{i=0}^{d-1}\frac{(c-x)^i(b-x)^{d-1-i}}{(c-a)^i(b-a)^{d-1-i}}B(x|a,b) \\ &+ \frac{(c-x)^d}{(c-a)^{d-1}(c-b)}B(x|b,c), \\ B(x|a,b,\overbrace{c,\ldots,c}^{d}) &= \frac{(x-a)^d}{(c-a)^{d-1}(b-a)}B(x|a,b) \\ &+ \frac{c-x}{c-b}\sum_{i=0}^{d-1}\frac{(x-a)^i(x-b)^{d-1-i}}{(c-a)^i(c-b)^{d-i}}B(x|b,c). \end{aligned}$$

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

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

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

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

2.11 Given a knot vector $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ 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 2.20 and 2.21 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.

2.12 Implement Algorithm 2.21 in your favourite programming language.

2.13 Implement Algorithm 2.20 in your favourite programming language.

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

2.15 Implement Algorithm 2.21 in your favourite programming language.

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

2.17 Write a program that plots the cubic B-spline $B(x|0, 1, 3, 5, 6)$ and its polynomial pieces. Present the results as in Figure 2.3.

2.18 a) What is computed by Algorithm 2.20 if x does not belong to the interval $[t_\mu, t_{\mu+1})$?

b) Repeat (b) for Algorithm 2.21.

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

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

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 $\boldsymbol{t}$ by

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

Explain how this knot vector can be exploited to compute $B_{1,3}(x)$ by Algorithms 2.20 or 2.21.

CHAPTER 3

Further properties of splines and B-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?” This may seem like a rather theoretical and non interesting issue from a practical point of view, but if our spline spaces contain sufficiently many interesting functions we will gain the flexibility that is required for practical applications.

The answer to the question above is that our spline space contain a large class of piecewise polynomials and this ensures that splines are reasonably flexible, much more flexible 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 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 2.18.

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. We start by associating the polynomial $\rho_{j,0}(y) = 1$ with $B_{j,0}$ and, more generally, the polynomial in y given by

$$\rho_{j,d}(y) = (y - t_{j+1})(y - t_{j+2}) \cdots (y - t_{j+d}), \tag{3.1}$$

is associated with the B-spline $B_{j,d}$ for $d \geq 1$. This polynomial is called the *dual polynomial* of the B-spline $B_{j,d}$. On the interval $[t_\mu, t_{\mu+1})$ we have the $d+1$ nonzero B-splines $\boldsymbol{B}_d =$

$(B_{\mu-d,d},\ldots,B_{\mu,d})^T$. We collect the corresponding dual polynomials in the vector

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

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

Lemma 3.1. *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*

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

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

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

for $j = \mu-d+1, \ldots, \mu$. 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

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

A simple calculation reveals that

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

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 $\boldsymbol{R}$'s. By making use of all the matrices $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_d$ we end up with a scalar.

Corollary 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)*. Then the relation*

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

holds for all real numbers x_1, x_2, …, x_d and y.

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

Lemma 3.3. *For $d \geq 2$ and for any x and z in $\mathbb{R}$, the matrices $\boldsymbol{R}_{d-1}$ and $\boldsymbol{R}_d$ satisfy the relation*

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

Proof. Applying (3.3) twice we obtain

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

and by symmetry we also have

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

Equivalently,

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

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

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

To complete the proof, we must show that $\boldsymbol{B} = \boldsymbol{0}$. Let $\boldsymbol{a}$ be any vector in $\mathbb{R}^{d-1}$. Then we know from (3.8) that $\boldsymbol{a}^T\boldsymbol{B}\boldsymbol{\rho}_d(y) = 0$ for all y. Since the $d+1$ polynomials in $\boldsymbol{\rho}_d$ are linearly independent, see Lemma 3.7, this means that $\boldsymbol{a}^T\boldsymbol{B} = \boldsymbol{0}$. But $\boldsymbol{a}$ was arbitrary, so $\boldsymbol{B}$ maps all vectors to $\boldsymbol{0}$, in other words $\boldsymbol{B} = \boldsymbol{0}$. ∎

3.1.2 Marsden's identity and representation of polynomials

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

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

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

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

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

The power of Theorem 3.4 lies in the fact that the coefficients $\boldsymbol{\rho}_d$ depend on y. Using this result we can show explicitly how the powers 1, x, …, x^d can be written in terms of B-splines.

Corollary 3.5. *For any x in 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*

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

$$x = \sum_{j=1}^{n} t^*_{j,d} B_{j,d}(x), \quad \textit{for } d \geq 1, \tag{3.12}$$

$$x^2 = \sum_{j=1}^{n} t^{**}_{j,d} B_{j,d}(x), \quad \textit{for } d \geq 2, \tag{3.13}$$

where

$$t^*_{j,d} = (t_{j+1} + \cdots + t_{j+d})/d \tag{3.14}$$

$$t^{**}_{j,d} = \sum_{i=j+1}^{j+d-1} \sum_{k=i+1}^{j+d} t_i t_k / \binom{d}{2}. \tag{3.15}$$

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

$$x^r = \sum_{j=1}^{n} \sigma^r_{j,d} B_{j,d}(x) \tag{3.16}$$

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

$$\sigma^r_{j,d} = \Bigl(\sum t_{j_1} t_{j_2} \cdots t_{j_r}\Bigr) / \binom{d}{r}, \qquad \textit{for } r = 0, 1, \ldots, d, \tag{3.17}$$

where the sum is over all integers j_1, ..., j_r with $j + 1 \leq j_1 < \cdots < j_r \leq j + d$, 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

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

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

$$\rho_{j,d}(y) = y^d - t^*_{j,d} y^{d-1} + t^{**}_{j,d} y^{d-2} + \text{ lower order terms.} \tag{3.19}$$

From this it follows that

$$D^d \rho_{j,d}(0) = d!, \quad D^{d-1} \rho_{j,d}(0) = -(d-1)! t^*_{j,d}, \quad D^{d-2} \rho_{j,d}(0) = (d-2)! t^{**}_{j,d}. \tag{3.20}$$

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

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

Using the same reasoning as above, we therefore find that

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

so (3.16) follows from (3.18). ∎

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

Example 3.6. In the cubic case, the relations (3.11)–(3.13) are

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

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

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

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

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

3.1.3 Linear independence of B-splines

Recall from Appendix A 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 3.7. *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 3.5 we know that the power basis 1, x, …, x^d, and therefore any polynomial of degree d, can be represented by linear combinations of B-splines. On the interval $[t_\mu, t_{\mu+1})$ the only nonzero B-splines are $\{B_{j,d}\}_{j=\mu-d}^{\mu}$. These B-splines therefore form a basis for polynomials of degree d on $[t_\mu, t_{\mu+1})$, and in particular they are linearly independent on this interval since there are only $d+1$ of them. The symmetry of x and y in (3.10) leads to the same conclusion for the dual polynomials. ∎

From this local result we shall 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 3.8. *A knot vector $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ 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* $j = 1, 2, \ldots, n$.

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.*

In most applications $d+1$-regular knot vectors are used, but linear independence can be proved in the more general situation of a $d+1$-extended knot vector.

Theorem 3.9. *Suppose that* $\boldsymbol{t}$ *is a* $d+1$*-extended knot vector. Then the B-splines in* $\mathbb{S}_{d,\boldsymbol{t}}$ *are linearly independent on the interval* $[t_{d+1}, t_{n+1})$.

Proof. We prove the result in the case where $\boldsymbol{t}$ is $d+1$-regular and leave the proof in the general case to the reader, see Exercise 2. 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 $j = 1, \ldots, n$. 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 $k = \mu - d, \ldots, \mu$. 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 the B-splines 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

In this section we study differentiation of splines with the matrix representation of B-splines as a starting point. We start by defining jumps and derivatives for piecewise continuous functions.

Definition 3.10. *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) *where the* one-sided limits

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

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

$$J_z f = f(z+) - f(z-) \tag{3.26}$$

is called the jump *of* f *at* z.

Definition 3.11. *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* $k = 0, \ldots, r$ *at some* $z \in (a, b)$ *then* f *is said to be* C^r *at* z*. Differentiation for 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*

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

At a point where the rth 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 the same at such a point.

Example 3.12. It is easy to check that the quadratic B-spline

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

is continuous on $\mathbb{R}$. The first derivative

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

is piecewise continuous on $\mathbb{R}$ with a discontinuity at $x = 0$, and the second derivative

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

is piecewise continuous on $\mathbb{R}$ with discontinuities at 0, 1, and 2. The third derivative is identically zero and continuous everywhere. This B-spline is therefore C^0 at $x = 0$, it is C^1 at $x = 1$ and $x = 2$ and at all other real numbers it has infinitely many continuous derivatives.

3.2.1 Derivatives of B-splines

Our next aim is to study differentiation of B-splines. We will start by working with polynomials and considering what happens on one knot interval, and then generalise to splines.

From Definition 3.11 we see that the rth derivative of a B-spline $B_{j,d}$ is given by

$$D^r B_{j,d} = \sum_{k=j}^{j+d} D^r B^k_{j,d} B_{k,0}, \quad r \geq 0, \tag{3.27}$$

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

Recall from Theorem 2.18 that on a knot interval $[t_\mu, t_{\mu+1})$ the row vector of the nonzero B-splines $\boldsymbol{B}_d$ is given by

$$\boldsymbol{B}_d(x) = R_1(x) \cdots R_d(x). \tag{3.28}$$

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

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

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

Proof. Let $(\boldsymbol{AB})_{ij}$ be an arbitrary element in the product $\boldsymbol{AB}$. Then

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

which proves the lemma. ∎

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

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

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

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

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

To simplify equation 3.29, we need the following lemma.

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

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

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

Using equation 3.31 we can move the D in (3.29) from $\boldsymbol{R}_k$ to $\boldsymbol{R}_d$ for each k. The end result is

$$D\boldsymbol{B}_d(x) = d\,\boldsymbol{R}_1(x)\cdots\boldsymbol{R}_{d-1}(x)D\boldsymbol{R}_d = d\,\boldsymbol{B}_{d-1}(x)D\boldsymbol{R}_d. \tag{3.32}$$

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

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

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

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

In general, for the rth derivative we find

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

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

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

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

Suppose that $f(x) = \sum_{j=1}^n c_j B_{j,d}(x)$. The r'th derivative of f at x is given by

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

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

Note that the symmetry property (3.31) gives us a curious freedom in how to represent the rth derivative: It does not matter which of the d matrices $\boldsymbol{R}_k$ 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 3.2.2, and leads to the following differentiation formula for a B-spline.

Theorem 3.16. *The derivative of the jth B-spline of degree d on $\boldsymbol{t}$ is given by*

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

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

$$DB_{j,d}(x) = \frac{d}{d-1}\Big(\frac{x - t_j}{t_{j+d} - t_j} DB_{j,d-1}(x) + \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} DB_{j+1,d-1}(x)\Big) \tag{3.36}$$

for $d \ge 2$ and any x in $\mathbb{R}$.

Using the $'0/0 = 0'$ convention the differentiation formula (3.35) can be expressed more explicitly as

$$DB_{j,d} = d \begin{cases} 0, & \text{if } t_j = t_{j+d+1}; \\ \dfrac{B_{j,d-1}}{t_{j+d} - t_j}, & \text{if } t_j < t_{j+d} \text{ and } t_{j+1} = t_{j+1+d}; \\ -\dfrac{B_{j+1,d-1}}{t_{j+1+d} - t_{j+1}}, & \text{if } t_j = t_{j+d} \text{ and } t_{j+1} < t_{j+1+d}; \\ \dfrac{B_{j,d-1}}{t_{j+d} - t_j} - \dfrac{B_{j+1,d-1}}{t_{j+1+d} - t_{j+1}}, & \text{otherwise.} \end{cases}$$

Proof. Clearly (3.35) holds if $x \notin [t_j, t_{j+1+d})$ so suppose $x \in [t_\mu, t_{\mu+1})$ for some $j \le \mu \le j+d$. By (3.33) for $r = 1$ we have

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

Carrying out the matrix multiplication on the right and comparing the jth component on both sides we obtain (3.35). Since (3.35) is independent of μ, it holds for all $x \in [t_j, t_{j+d+1})$.

To prove (3.36) we make use of Lemma 3.14 and differentiate the matrix $\boldsymbol{R}_1$ instead of $\boldsymbol{R}_d$, see exercise 6. ∎

3.2.2 Computing derivatives of splines and B-splines

As for evaluation, see Section 2.4, there are two closely related algorithms for computing the rth derivative of a spline, both based on the matrix representation from Theorem 3.15,

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

As before, we assume that x lies in the interval $[t_\mu, t_{\mu+1})$ and that the vector $\boldsymbol{c}_d$ contains the B-spline coefficients that multiply the B-splines that are nonzero on $[t_\mu, t_{\mu+1})$ so that $\boldsymbol{c}_d = (c_{\mu-d}, \ldots, c_\mu)^T$. 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), while the DR (Derivative Right) Algorithm computes the rth derivative of all the nonzero B-splines at a point by accumulating matrix products from left to right, then multiplying with the coefficients and summing up.

Algorithm 3.17 (DL)**.** *Let the polynomial degree d, the $2d$ knots $t_{\mu-d+1} \le t_\mu < t_{\mu+1} \le t_{\mu+d}$, the B-spline coefficients $\boldsymbol{c}_d^{(0)} = \boldsymbol{c}_d = (c_j)_{j=\mu-d}^{\mu}$ of a spline f, and a number x in $[t_\mu, t_{\mu+1})$ be given. After evaluation of the products*

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

the rth derivative of f at x is given by

$$D^r f(x) = d!\,\boldsymbol{c}_0^{(r)}/(d-r)!.$$

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

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

the vector $D^r\boldsymbol{B}_d(x)$ will contain the value of the rth derivative of the nonzero B-splines at x,

$$D^r\boldsymbol{B}_d(x) = \big(D^r B_{\mu-d,d}(x), \ldots, D^r B_{\mu,d}(x)\big)^T.$$

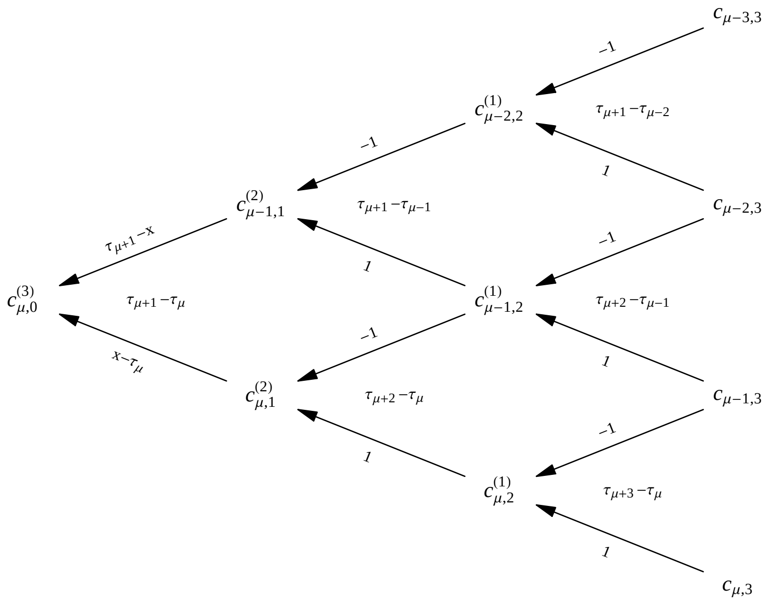


Figure 3.1. A triangular algorithm for computation of the second derivative of a cubic spline at x.

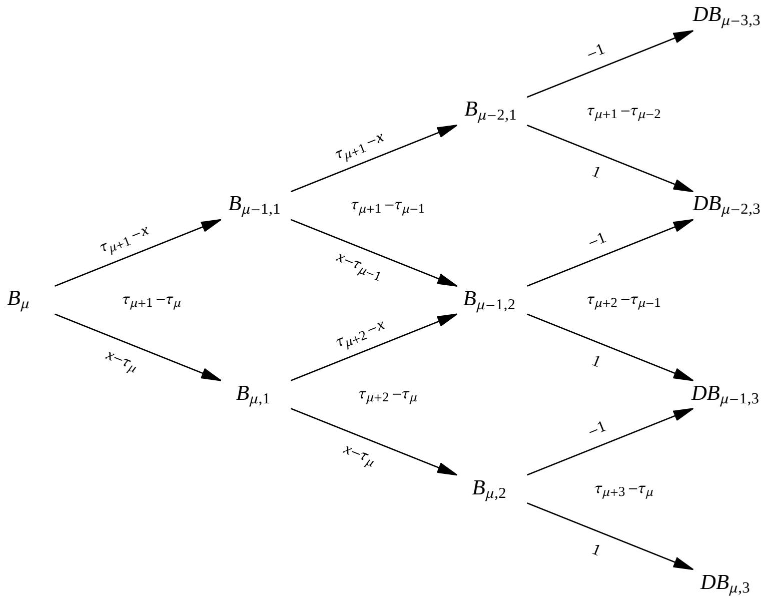


Figure 3.2. A triangular algorithm for computation of the derivative of the nonzero cubic B-splines at x.

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 matrix-vector products on the forms $D\boldsymbol{R}_k\boldsymbol{c}_k$ and $\boldsymbol{R}_k(x)\boldsymbol{c}_k$. The component form of the latter product is given in (2.25), while the component form of the former is obtained by differentiating the linear factors in (2.25) with respect to x. The result is

$$(D\boldsymbol{R}_k\boldsymbol{c}_k)_j = \frac{c_{k,j} - c_{k,j-1}}{t_{j+k} - t_j} \tag{3.38}$$

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

The alternative algorithm accumulates the matrix products in (2.23) from left to right. The component form of the product $\boldsymbol{B}_{k-1}(x)^T\boldsymbol{R}_k$ is given in (2.26) while the component form of the product $\boldsymbol{B}_{k-1}(x))^T D\boldsymbol{R}_k$ is

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

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

3.2.3 Smoothness of B-splines

A characteristic feature of splines is their smoothness properties as stated in Theorem 1.4 in Chapter 1. In this section we discuss the smoothness properties of splines in detail. We start by characterising the smoothness of B-splines.

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

This theorem will proved via a sequence of steps. We first note from the explicit formula (2.11) 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 the general case the proof is based on the following recurrence relations for jumps.

Lemma 3.20. *The jump in a B-spline at a real number x satisfies the recurrence relation*

$$J_x(B_{j,d}) = \frac{x - t_j}{t_{j+d} - t_j} J_x(B_{j,d-1}) + \frac{t_{j+1+d} - x}{t_{j+1+d} - t_{j+1}} J_x(B_{j+1,d-1}), \tag{3.40}$$

with

$$J_x(B_{j,0}) = \begin{cases} 1, & \text{if } x = t_j, \\ -1, & \text{if } x = t_{j+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{3.41}$$

For $r \ge 1$ the jump in the rth derivative is given by

$$J_x(D^r B_{j,d}) = d\Big(\frac{J_x(D^{r-1}B_{j,d-1})}{t_{j+d} - t_j} - \frac{J_x(D^{r-1}B_{j+1,d-1})}{t_{j+1+d} - t_{j+1}}\Big), \quad \text{for } x \in \mathbb{R} \text{ and } r \ge 1. \tag{3.42}$$

Proof. Evaluating the recurrence relation (2.1) at $x+$ and $x-$ and subtracting we obtain (3.40), while (3.41) follows directly from the definition of $B_{j,0}$. Differentiating the differentiation formula (3.35) a total of $r-1$ times leads to

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

for any real number x. The same formula holds if we replace $D = D_+$ by D_-. Taking the difference of the two formulas leads to (3.42). ∎

As usual the $'0/0 = 0'$ convention is used in (3.40) and (3.42).

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

Lemma 3.21. *Suppose that no knot among t_j, $t_{j+1}, \ldots, t_{j+d+1}$ occurs more than d times. Then the B-spline $B_{j,d}$ is continuous for all real numbers x.*

Proof. The proof is by induction on the degree d. For a B-spline of degree 0 the lemma does not apply so the induction starts with $d = 1$. It is easy to see from the explicit representation in Example 2.2 that a linear B-spline with three distinct knots is continuous. For the induction step we assume that the lemma holds for B-splines of degree $d-1$. To prove that it is also true for B-splines of degree d suppose first that no knots occur more than $d-1$ times. Then 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 $t_j, t_{j+1}, \ldots, t_{j+d+1}$. There are three cases. Suppose first that $x = t_j$. Since $t_{j+d-1} < t_{j+d}$ it follows from the induction hypothesis that $J_x(B_{j+1,d-1}) = 0$, while $J_x(B_{j,d-1}) = 1$. From (3.40) we then obtain $J_x(B_{j,d}) = 0$, since $(x-t_j)J_x(B_{j,d-1}) = 0 \cdot 1 = 0$. The proof in the case $x = t_{j+1+d}$ is similar. Finally, if $t_j < x < t_{j+1+d}$ then $x = t_{j+1} = \cdots = t_{j+d}$ so (3.40) yields

$$J_x(B_{j,d}) = \frac{x-t_j}{t_{j+d}-t_j}\cdot 1 + \frac{t_{j+1+d}-x}{t_{j+1+d}-t_{j+1}}(-1) = 0.$$

This completes the proof. ∎

Proof. [The continuity part of Theorem 3.19]
For $r = 0$ the result follows from Lemma 3.21, while for general r it follows from (3.42) and induction on d that $J_z(D^r B_{j,d}) = 0$ for $1 \le r \le d-m$. ∎

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

Lemma 3.22. *Suppose that the number z occurs exactly m times among the knots $t_j, \ldots, t_{j+1+d}$. Then the $d-m+1$th derivative of $B_{j,d}$ has a jump at z given by*

$$J_z(D^{d-m+1}B_{j,d}) = \frac{d!}{(m-1)!}(t_{j+1+d}-t_j)/\prod_{\substack{k=j\\ t_k\neq z}}^{j+1+d}(t_k - z) \neq 0. \tag{3.43}$$

Proof. As usual the proof is by induction of the degree d. We first note that (3.43) holds in the case where $m = d + 2$, so we may assume that $m \leq d + 1$. It is easy to check that equation (3.43) holds when $d = 0$ and $m = 1$. Suppose that (3.43) holds for B-splines of degree $d - 1$. For a B-spline of degree d we apply (3.42) 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 (3.42) 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 (3.43) 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 (3.42) and the induction hypothesis we then obtain

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

which completes the proof. ∎

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 $\mathbb{S}_{d,\boldsymbol{t}}$? 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 $\mathbb{S}_{d,\boldsymbol{t}}$, 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 $\mathbb{S}_{d,\boldsymbol{t}}$ 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 $\boldsymbol{\Delta} = (\xi_i)_{i=1}^N$ be a partition of $[a, b]$,*

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

and let $\boldsymbol{r} = (r_i)_{i=2}^{N-1}$ be a sequence of integers. By $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ 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 $f \in \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ if and only if the restriction of f to (ξ_{i-1}, ξ_i) is a polynomial of degree d for $i = 2, \ldots, N$, and $D^k f$ is continuous at ξ_i for $k = 0, \ldots, r_i$ and $i = 2, \ldots, N - 1$.

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

Lemma 3.24. *The dimension of $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ is $n = (N - 1)d + 1 - \sum_{i=2}^{N-1} r_i$.*

To see why Lemma 3.24 is reasonable we can argue as follows. If there were no smoothness conditions ($r_i = -1$ for all i) we would have a space of dimension $(N-1)(d+1)$ (there are $N - 1$ subintervals and on each we have a space of polynomials of degree d). All together there are $\sum_{i=2}^{N-1}(r_i + 1)$ smoothness conditions so

$$\dim \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta}) \geq (N - 1)(d + 1) - \sum_{i=2}^{N-1}(r_i + 1) = (N - 1)d + 1 - \sum_{i=2}^{N-1} r_i. \tag{3.44}$$

A priori we only get a lower bound since we cannot be sure that each continuity constraint reduces the dimension by one. A more careful investigation reveals that the dimension agrees with this lower bound, see Exercise 7.

There are many ways to represent the piecewise polynomials in $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$. One possibility is to pick one's favourite polynomial basis and represent each piece as a polynomial of degree d and ignore the smoothness conditions. Another possibility is to use the truncated power basis that is employed to prove Lemma 3.24 in Exercise 7. The following theorem shows that $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ can in fact be represented in terms of B-splines on an appropriate knot vector.

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

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

where n is given in Lemma 3.24 and the end knots satisfy $t_1 \leq \cdots \leq t_{d+1} \leq a$ and $b \leq t_{n+1} \leq \cdots \leq t_{n+d+1}$. Then

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

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

Proof. Let $\mathbb{S} = \mathbb{S}_{d,\boldsymbol{t}}|_{[a,b]}$. We note that by the construction of the knot vector, the B-splines in $\mathbb{S}$ satisfy the smoothness conditions of $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ so $\mathbb{S} \subseteq \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$. On the other hand, the length of the knot vector $\boldsymbol{t}$ is $n + d + 1$ so $\dim \mathbb{S} = \dim \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$. But a subspace that has the same dimension as the full space must agree with the full space so $\mathbb{S} = \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$. ∎

Exercises for Chapter 3

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

a) Let $\boldsymbol{t}$ denote the augmented knot vector $\boldsymbol{t} = (0, 0, 0, 1, 3, 4, 5, 5)$. Show that we can associate 4 B-splines $\{B_{i,3}\}_{i=1}^{4}$ with $\boldsymbol{t}$ 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.

3.2 Prove Theorem 3.9.

3.3 Let $\boldsymbol{t} = (t_j)_{j=1}^{n+d+1}$ be 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 $\boldsymbol{A}$ be matrix where each entry is a function of x and let α be a scalar function of x. Prove the formula

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

3.5 a) Count the number of operations (additions/subtractions, multiplications, divisions) involved in computing the matrix $\boldsymbol{R}_k(x)$ defined in (2.20). Do the same for the matrix $D\boldsymbol{R}_k$ defined in (3.30).

b) Recall that in the formula (3.34) for the rth derivative of f, we have the freedom to differentiate any r of the d matrices $\{\boldsymbol{R}_k(x)\}_{k=1}^{d}$. 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

$$(DB_{\mu-d,d}(x), \ldots, DB_{\mu,d}(x)) = dD\boldsymbol{R}_1\boldsymbol{R}_2(x)\cdots\boldsymbol{R}_d(x) \tag{3.45}$$

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

b) Show that (3.45) 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 In this exercise we will provide a proof of Lemma 3.24. Let π_d denote the linear space of polynomials of degree at most d. Recall that the powers $1, x, \ldots, x^d$ is a basis for π_d on any interval $[a, b]$ with $a < b$ and that the dimension of π_d is $d + 1$.

a) Let $\boldsymbol{\Delta} = (\xi_i)_{i=}^{N}$ be a partition of some interval $[a, b]$,

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

and let $\mathbb{S}_d^{-1}(\boldsymbol{\Delta})$ denote the set of functions that are polynomials of degree $d+1$ on each subinterval (ξ_{i-1}, ξ_i) for $i = 2, \ldots, N$ (no continuity is assumed between the different pieces). Show that the dimension of $\mathbb{S}_d^{-1}(\boldsymbol{\Delta})$ is $(N-1)(d+1)$. Hint: Show that the functions $\{\eta_{i,k}\}_{i=1,k=0}^{N-1,d}$ defined by

$$\eta_{i,k}(x) = \begin{cases} (x-\xi_i)^k, & \text{if } \xi_i \le x < \xi_{i-1}; \\ 0, & \text{otherwise}; \end{cases}$$

form a basis for $\mathbb{S}_d^{-1}(\boldsymbol{\Delta})$.

b) Show that a different basis for $\mathbb{S}_d^{-1}(\boldsymbol{\Delta})$ is given by the functions $\{\theta_{i,k}\}_{i=1,k=0}^{N-1,d}$ defined by

$$\theta_{i,k}(x) = (x-\xi_i)_+^k,$$

where

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

except that we use the convention $0^0 = 1$.

c) Let J denote the jump-operator defined in Definition 3.10. Show that

$$J_{\xi_i}(D^\ell \theta_{j,k}) = k!\delta_{\ell,k}\delta_{i,j}$$

where $\delta_{m,n} = 1$ if $m = n$ and zero otherwise.

d) Let $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ be as in Definition 3.23. Show that $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ is a subspace of $\mathbb{S}_d^{-1}(\boldsymbol{\Delta})$. Show also that if $f = \sum_{i=1}^{N-1}\sum_{k=0}^{d} c_{i,k}\eta_{i,k}$ is in $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$ then $c_{i,k} = 0$ for $k = 0$, $1, \ldots, r_i$ and $i = 2, 3, \ldots, N-1$. Hint: Make use of (c). Conclude that $\{\theta_{i,k}\}_{i=1,k=r_i}^{N-1,d}$, where $r_1 = 0$, is a basis for $\mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta})$, and that

$$\dim \mathbb{S}_d^{\boldsymbol{r}}(\boldsymbol{\Delta}) = (N-1)d + 1 - \sum_{i=2}^{N-1} r_i.$$

CHAPTER 4

Knot insertion

We have already seen that the control polygon of a spline provides a rough sketch of the spline itself. In this chapter we will study another aspect of the relationship between a spline and its control polygon. Namely, we shall see that as the distance between the knots of a spline is reduced, the control polygon approaches the spline it represents.

To reduce the knot spacing we perform *knot insertion*. Knot insertion amounts to what the name suggests, namely insertion of knots into an existing knot vector. This results in 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. In fact, by inserting sufficiently many knots, we can make the distance between the spline and its control polygon as small as we wish. This has obvious advantages for practical computations since we can represent a function consisting of infinitely many points by a polygon with a finite number of vertexes. By combining this with other properties of splines like the convex hull property, we obtain a very powerful toolbox for algorithmic manipulation of spline functions.

We start, in Section 4.1, by showing that the control polygon of a spline converges to the spline as the knot spacing goes to zero. To prove this we make use of one property of splines that is proved later, in Chapter 8. The obvious way to reduce the knot spacing is via knot insertion, and in Section 4.2 we develop algorithms for expressing the B-spline coefficients relative to the new refined knot vector in terms of the B-spline coefficients relative to the original knot vector. In Section 4.3 we give a characterisation the B-spline coefficients as functions of the knots. 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.

4.1 Convergence of the control polygon for spline functions

Recall that for a spline function $f(x) = \sum_i c_i B_{i,d,\boldsymbol{t}}$, the control polygon is the piecewise linear interpolant to the points (t_i^*, c_i), where $t_i^* = (t_{i+1} + \cdots + t_{i+d})/d$ is the ith knot average. Lemma 4.1 below shows that this is indeed a 'good' definition of the control polygon since c_i is close to $f(t_i^*)$, 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 $[t_{i+1}, t_{i+d+1}]$. More specifically, we define the size of f on the interval $[a, b]$ in terms of the max-norm

$$\|f\|_{[a,b]} = \max_{x\in[a,b]} \big|f(x)\big|,$$

where we take the limit from the right at a and the limit from the left at b. We will prove in Lemma 9.16 that there exists a constant K_d that depends on d, but *not* on $\boldsymbol{t}$, such that the inequality

$$|c_i| \le K_d \|f\|_{[t_{i+1},t_{i+d}]} \tag{4.1}$$

holds.

Lemma 4.1. *Let f be a spline in $\mathbb{S}_{d,\boldsymbol{t}}$ with coefficients (c_i). Then*

$$|c_i - f(t_i^*)| \le K(t_{i+d} - t_{i+1})^2 \|D^2 f\|_{[t_{i+1},t_{i+d}]}, \tag{4.2}$$

where $t_i^* = (t_{i+1} + \cdots + t_{i+d})/d$, 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 $t_{i+1} = t_{i+d}$ then we know from property 5 in Lemma 2.6 that $B_{i,d}(t_i^*) = 1$ so $c_i = f(t_i^*)$ and there is nothing to prove. Assume for the rest of the proof that the interval $J = (t_{i+1}, t_{i+d})$ 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 on 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. The ith B-spline coefficient of g in $\mathbb{S}_{d,\boldsymbol{t}}$ is given by

$$b_i = c_i - f(x_0) - (t_i^* - x_0)Df(x_0).$$

Choosing $x_0 = t_i^*$ we have $b_i = c_i - f(t_i^*)$ and according to the inequality (4.1) and the error term in first order Taylor expansion we find

$$|c_i - f(t_i^*)| = |b_i| \le K_d \|g\|_J \le \frac{K_d (t_{i+d} - t_{i+1})^2}{2} \|D^2 f\|_J.$$

The inequality (4.2) therefore holds with $K = K_d/2$ and the proof is complete. ∎

Lemma 4.1 shows that the corners of the control polygon converge to the spline as the knot spacing goes to zero, but what does this really mean? So far we have considered the knots of a spline to be given, fixed numbers, but it is in fact possible to represent a spline on many different knot vectors. Suppose for example that the given spline f is a polynomial of degree d on the interval $[a, b] = [t_{d+1}, t_{m+1}]$, and that the knot vector $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ is $d+1$-regular. From Section 3.1.2 we know that f lies in $\mathbb{S}_{d,\boldsymbol{t}}$ regardless of how the interior knots in $[a, b]$ are chosen. We can therefore think of the B-spline coefficients as functions of the knots, and the difference $c_i - f(t_i^*)$ is then also a function of the knots. Lemma 4.1

tells us that this difference is bounded by $(t_{i+d}-t_{i+1})^2$ and therefore tends to zero as t_{i+1} tends to t_{i+d}. It is important to realise that this argument is in general only valid if f is kept fixed. Otherwise it may happen that $\|D^2 f\|_{[t_{i+1},t_{i+d}]}$ increases when $(t_{i+d}-t_{i+1})^2$ decreases with the result that the product remains fixed.

If f is a true piecewise polynomial with jumps in some derivatives at the knots in (a,b), we can introduce some auxiliary knots in (a,b) where the jumps in the derivatives are zero. These knots we can then move around in such a way that the control polygon approaches f.

Since the corners of the control polygon converges to the spline it is not surprising that the control polygon as a whole also converges to the spline.

Theorem 4.2. *Let $f=\sum_{i=1}^m c_i B_{i,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{t}}$, and let $\Gamma_{d,\boldsymbol{t}}(f)$ be its control polygon. Then*

$$\big\|\Gamma_{d,\boldsymbol{t}}(f)-f\big\|_{[t_1^*,t_m^*]} \le Kh^2\|D^2 f\|_{[t_1,t_{m+d+1}]}, \tag{4.3}$$

where $h=\max_i\{t_{i+1}-t_i\}$ and the constant K only depends on d.

Proof. As usual, we assume that $\boldsymbol{t}$ 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 $[t_1^*,t_m^*]$ and let j be such that $t_j^*\le x<t_{j+1}^*$. Observe that since the interval $J^*=(t_j^*,t_{j+1}^*)$ is nonempty we have $t_{j+1}<t_{j+d+1}$ 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 on J^*. Let

$$g(x)=\frac{(t_{j+1}^*-x)f(t_j^*)+(x-t_j^*)f(t_{j+1}^*)}{t_{j+1}^*-t_j^*}$$

be the linear interpolant to f on this interval. We will show that both $\Gamma=\Gamma_{d,\boldsymbol{t}}(f)$ and f are close to g on J^* and then deduce that Γ is close to f because of the triangle inequality

$$\big|\Gamma(x)-f(x)\big|\le\big|\Gamma(x)-g(x)\big|+\big|g(x)-f(x)\big|. \tag{4.4}$$

Let us first consider the difference $\Gamma-g$. Note that

$$\Gamma(x)-g(x)=\frac{(t_{j+1}^*-x)(b_j-f(t_j^*))+(x-t_j^*)(b_{j+1}-f(t_{j+1}^*))}{t_{j+1}^*-t_j^*}$$

for any x in J^*. We therefore have

$$\big|\Gamma(x)-g(x)\big|\le\max\Big\{\big|b_j-f(t_j^*)\big|,\big|b_{j+1}-f(t_{j+1}^*)\big|\Big\},$$

for $x\in J^*$. From Lemma 4.1 we then conclude that

$$|\Gamma(x)-g(x)|\le K_1h^2\|D^2 f\|_J,\quad x\in J^*, \tag{4.5}$$

where $J=[t_1,t_{m+d+1}]$ and K_1 depending only on d.

The second difference $f(x)-g(x)$ in (4.4) is the error in linear interpolation to f at the endpoints of J^*. For this process we have the standard error estimate

$$\big|f(x)-g(x)\big|\le\frac{1}{8}(t_{j+1}^*-t_j^*)^2\|D^2 f\|_{J^*}\le\frac{1}{8}h^2\|D^2 f\|_J,\quad x\in J^*. \tag{4.6}$$

If we now combine (4.5) and (4.6) as indicated in (4.4), we obtain the Theorem with constant $K=K_1+1/8$. ∎

Because of the factor h^2 in Theorem 4.2 we say (somewhat loosely) that the control polygon converges quadratically to the spline.

4.2 Knot insertion

In Section 4.1 we showed that the control polygon converges to the spline it represents when the knot spacing tends to zero. In this section we shall develop two algorithms for reducing the knot spacing by inserting new (artificial) knots into a spline. The two algorithms for knot insertion are closely related to Algorithms 2.20 and 2.21; in fact these two algorithms are special cases of the algorithms we develop here.

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

A simple example of a knot vector and a refinement is given by

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

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

Note that if $\boldsymbol{t}$ is a refinement of $\boldsymbol{\tau}$ then $\boldsymbol{\tau}$ is a subsequence of $\boldsymbol{t}$, and this we will write $\boldsymbol{\tau} \subseteq \boldsymbol{t}$. The term knot insertion is used because in most situations the knot vector $\boldsymbol{\tau}$ is given and $\boldsymbol{t}$ is obtained by 'inserting' knots into $\boldsymbol{\tau}$.

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

Lemma 4.4. *Let d be a positive integer and let $\boldsymbol{\tau}$ be a knot vector with at least $d+2$ knots. If $\boldsymbol{t}$ is a knot vector which contains $\boldsymbol{\tau}$ as a subsequence then $\mathbb{S}_{d,\boldsymbol{\tau}} \subseteq \mathbb{S}_{d,\boldsymbol{t}}$.*

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

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

Suppose that f is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$ with B-spline coefficients $\boldsymbol{c} = (c_j)$ so that $f = \sum_j c_j B_{j,d,\boldsymbol{\tau}}$. If $\boldsymbol{\tau}$ is a subsequence of $\boldsymbol{t}$, we know from Lemma 4.4 that $\mathbb{S}_{d,\boldsymbol{\tau}}$ is a subspace of $\mathbb{S}_{d,\boldsymbol{t}}$ so f must also lie in $\mathbb{S}_{d,\boldsymbol{t}}$. Hence there exist real numbers $\boldsymbol{b} = (b_i)$ with the property that $f = \sum_i b_i B_{i,d,\boldsymbol{t}}$, i.e., the vector $\boldsymbol{b}$ contains the B-spline coefficients of f in $\mathbb{S}_{d,\boldsymbol{t}}$. 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 $\mathbb{S}_{d,\boldsymbol{t}}$.

Since $\mathbb{S}_{d,\boldsymbol{\tau}} \subseteq \mathbb{S}_{d,\boldsymbol{t}}$ all the B-splines in $\mathbb{S}_{d,\boldsymbol{\tau}}$ are also in $\mathbb{S}_{d,\boldsymbol{t}}$ so that

$$B_{j,d,\boldsymbol{\tau}} = \sum_{i=1}^{m} \alpha_{j,d}(i) B_{i,d,\boldsymbol{t}}, \quad j = 1,2,\ldots,n, \tag{4.7}$$

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

$$\boldsymbol{B}_{\boldsymbol{\tau}}^T = \boldsymbol{B}_{\boldsymbol{t}}^T \boldsymbol{A}, \tag{4.8}$$

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

$$f = \boldsymbol{B}_{\boldsymbol{\tau}}^T \boldsymbol{c} = \boldsymbol{B}_{\boldsymbol{t}}^T \boldsymbol{b},$$

where $\boldsymbol{b}$ and $\boldsymbol{c}$ are related by

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c}, \qquad \text{or} \qquad b_i = \sum_{j=1}^{n} a_{i,j} c_j \quad \text{for } i = 1,\, 2,\, \ldots,\, m. \tag{4.9}$$

The basis transformation $\boldsymbol{A}$ is called *the knot insertion matrix of degree d from* $\boldsymbol{\tau}$ *to* $\boldsymbol{t}$ and we will use the notation $\alpha_{j,d}(i) = \alpha_{j,d,\boldsymbol{\tau},\boldsymbol{t}}(i)$ 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* $\boldsymbol{t}$ *with knots* $\boldsymbol{\tau}$.

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

Example 4.5. Let us determine the transformation matrix $\boldsymbol{A}$ for splines with $d = 0$, when the coarse knot vector is given by $\boldsymbol{\tau} = (0, 1, 2)$, and the refined knot vector is $\boldsymbol{t} = (0, 1/2, 1, 3/2, 2) = (t_i)_{i=1}^5$. In this case

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

We clearly have

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

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

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

Example 4.6. Let us also consider an example with linear splines. Let $d = 1$, and let $\boldsymbol{\tau}$ and $\boldsymbol{t}$ be as in the preceding example. In this case $\dim \mathbb{S}_{d,\boldsymbol{\tau}} = 1$ and we find that

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

The situation is shown in Figure 4.1. The linear B-spline on $\boldsymbol{\tau}$ is a weighted sum of the three B-splines (dashed) on $\boldsymbol{t}$. The knot insertion matrix $\boldsymbol{A}$ is therefore the 3×1-matrix, or row vector, given by

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

4.2.1 Formulas and algorithms for knot insertion

To develop algorithms for performing knot insertion we need to study the matrix $\boldsymbol{A}$ in some more detail. Suppose as before that we have two knot vectors $\boldsymbol{\tau}$ and $\boldsymbol{t}$ with $\boldsymbol{\tau} \subseteq \boldsymbol{t}$

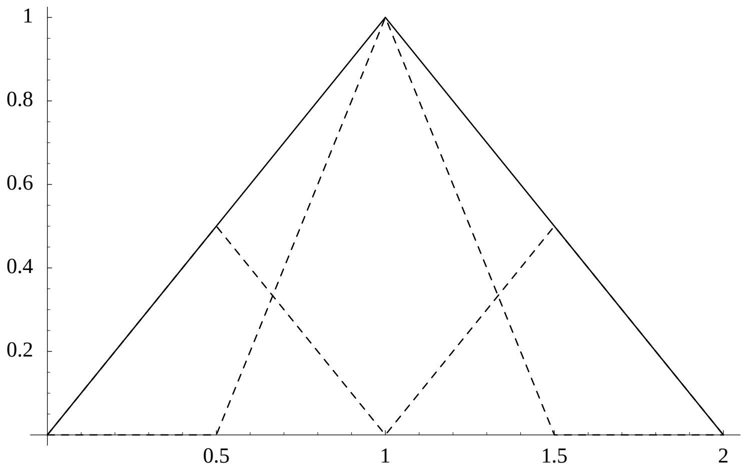


Figure 4.1. Refining a linear B-spline.

and a spline function $f = \sum_j c_j B_{j,d,\boldsymbol{\tau}}$ in $\mathbb{S}_{d,\boldsymbol{\tau}}$. Since the (i,j)-entry of $\boldsymbol{A}$ is $\alpha_{j,d}(i)$, the B-spline coefficients of f relative to $\mathbb{S}_{d,\boldsymbol{t}}$ are given by

$$b_i = \sum_{j=1}^{n} \alpha_{j,d}(i) c_j \tag{4.10}$$

for $i = 1, \ldots, m$, see (4.9). Similarly, equation (4.8) can now be written

$$B_{j,d,\boldsymbol{\tau}} = \sum_{i=1}^{m} \alpha_{j,d}(i) B_{i,d,\boldsymbol{t}} \tag{4.11}$$

for $j = 1, \ldots, n$.

In the following we 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 common knots at the ends so that $\tau_1 = t_1$ and $t_{n+1} = t_{m+1}$. Exercise 6 shows that this causes no loss of generality. The following theorem gives an explicit formula for the knot insertion matrix $\boldsymbol{A}$.

Recall from Theorem 2.18 the the B-spline matrix $\boldsymbol{R}_k(x) = \boldsymbol{R}^{\mu}_{k,\boldsymbol{\tau}}(x)$ is given by

$$\boldsymbol{R}^{\mu}_{k,\boldsymbol{\tau}}(x) = \begin{pmatrix} \frac{\tau_{\mu+1}-x}{\tau_{\mu+1}-\tau_{\mu+1-k}} & \frac{x-\tau_{\mu+1-k}}{\tau_{\mu+1}-\tau_{\mu+1-k}} & 0 & \cdots & 0 \\ 0 & \frac{\tau_{\mu+2}-x}{\tau_{\mu+2}-\tau_{\mu+2-k}} & \frac{x-\tau_{\mu+2-k}}{\tau_{\mu+2}-\tau_{\mu+2-k}} & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{\tau_{\mu+k}-x}{\tau_{\mu+k}-\tau_{\mu}} & \frac{x-\tau_{\mu}}{\tau_{\mu+k}-\tau_{\mu}} \end{pmatrix}.$$

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 $\boldsymbol{A}$ 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

$$\boldsymbol{\alpha}_d(i)^T = \big(\alpha_{\mu-d,d}(i), \ldots, \alpha_{\mu,d}(i)\big) = \begin{cases} 1, & \text{if } d = 0, \\ \boldsymbol{R}^\mu_{1,\boldsymbol{\tau}}(t_{i+1}) \boldsymbol{R}^\mu_{2,\boldsymbol{\tau}}(t_{i+2}) \cdots \boldsymbol{R}^\mu_{d,\boldsymbol{\tau}}(t_{i+d}), & \text{if } d > 0. \end{cases} \tag{4.12}$$

If $f = \sum_j c_j B_{j,d,\boldsymbol{\tau}}$ is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$, with B-spline coefficients $\boldsymbol{b}$ in $\mathbb{S}_{d,\boldsymbol{t}}$, then b_i is given by

$$b_i = \sum_{j=\mu-d}^{\mu} \alpha_{j,d}(i) c_j = \boldsymbol{R}^\mu_{1,\boldsymbol{\tau}}(t_{i+1}) \cdots \boldsymbol{R}^\mu_{d,\boldsymbol{\tau}}(t_{i+d}) \boldsymbol{c}_d, \tag{4.13}$$

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

Proof. We already know that the two B-spline bases are related through the relation (4.11); we will obtain the result by analysing this equation for different values of j.

Let ν be the largest integer such that $t_\nu = t_i$. This means that $\nu - d \leq i \leq \nu$ (note that $\nu = i$ only if $t_i < t_{i+1}$). Since $\boldsymbol{\tau}$ is a subsequence of $\boldsymbol{t}$ we have $[t_\nu, t_{\nu+1}) \subseteq [\tau_\mu, \tau_{\mu+1})$. Restrict x to the interval $[t_\nu, t_{\nu+1})$ so that $B_{\ell,d,\boldsymbol{t}}(x) = 0$ for $\ell < \nu - d$ and $\ell > \nu$. Equation (4.11) can then be written

$$B_{j,d,\boldsymbol{\tau}}(x) = \sum_{\ell=\nu-d}^{\nu} \alpha_{j,d}(\ell) B_{\ell,d,\boldsymbol{t}}(x), \quad \text{for } j = 1, \ldots, n. \tag{4.14}$$

Since $x \in [\tau_\mu, \tau_{\mu+1})$ we have $B_{j,d,\boldsymbol{\tau}}(x) = 0$ for $j < \mu - d$ or $j > \mu$. For these values of j the left-hand side of (4.14) is therefore zero, and since the B-splines $\{B_{\ell,d,\boldsymbol{t}}\}_{\ell=\nu-d}^{\nu}$ are linearly independent we must have $\alpha_{j,d}(\ell) = 0$ for $\nu - d \leq \ell \leq \nu$, and in particular $\alpha_{j,d}(i) = 0$, for $j < \mu - d$ and $j > \mu$.

To establish (4.12) we consider the remaining values of j, namely $j = \mu - d, \ldots, \mu$. On the interval $[t_\nu, t_{\nu+1})$, the nonzero part of the two B-spline bases can be represented by the two vectors $\boldsymbol{B}_{d,\boldsymbol{\tau}} = (B_{k,d,\boldsymbol{\tau}})_{k=\mu-d}^{\mu}$ and $\boldsymbol{B}_{d,\boldsymbol{t}} = (B_{\ell,d,\boldsymbol{t}})_{\ell=\nu-d}^{\nu}$. We also have the vectors of dual polynomials $\boldsymbol{\sigma}_d(y) = \big(\sigma_{k,d}(y)\big)_{k=\mu-d}^{\mu}$ and $\boldsymbol{\rho}_d(y) = \big(\rho_{\ell,d}(y)\big)_{\ell=\nu-d}^{\nu}$ given by

$$\begin{aligned} \sigma_{k,d}(y) &= (y - \tau_{k+1}) \cdots (y - \tau_{k+d}), \\ \rho_{\ell,d}(y) &= (y - t_{\ell+1}) \cdots (y - t_{\ell+d}). \end{aligned}$$

From Corollary 3.2 we have that the two sets of dual polynomials are related by

$$\rho_{\ell,d}(y) = \boldsymbol{R}_1(t_{\ell+1}) \cdots \boldsymbol{R}_d(t_{\ell+d}) \boldsymbol{\sigma}_d(y) \tag{4.15}$$

(in this proof we omit the second subscript and the superscript to the $\boldsymbol{R}$ matrices). Combining this with two versions of Marsden's identity (3.10) then yields

$$\begin{aligned} \boldsymbol{B}_{d,\boldsymbol{\tau}}(x)^T \boldsymbol{\sigma}_d(y) = (y - x)^d &= \sum_{\ell=\nu-d}^{\nu} \rho_{\ell,d}(y) B_{\ell,d,\boldsymbol{t}}(x) \\ &= \sum_{\ell=\nu-d}^{\nu} B_{\ell,d,\boldsymbol{t}}(x) \boldsymbol{R}_1(t_{\ell+1}) \cdots \boldsymbol{R}_d(t_{\ell+d}) \boldsymbol{\sigma}_d(y). \end{aligned}$$

The linear independence of the $d+1$ dual polynomials $\boldsymbol{\sigma}_d(y)$ allows us to conclude that

$$\boldsymbol{B}_{d,\boldsymbol{\tau}}(x)^T = \sum_{\ell=\nu-d}^{\nu} B_{\ell,d,\boldsymbol{t}}(x)\boldsymbol{R}_1(t_{\ell+1})\cdots\boldsymbol{R}_d(t_{\ell+d}) = \boldsymbol{B}_{d,\boldsymbol{t}}(x)^T \begin{pmatrix} \boldsymbol{R}_1(t_{\nu-d+1})\cdots\boldsymbol{R}_d(t_\nu) \\ \boldsymbol{R}_1(t_{\nu-d+2})\cdots\boldsymbol{R}_d(t_{\nu+1}) \\ \vdots \\ \boldsymbol{R}_1(t_{\nu+1})\cdots\boldsymbol{R}_d(t_{\nu+d}) \end{pmatrix}$$

for any x in the interval $[t_\nu, t_{\nu+1})$. Comparing this equation with (4.14) and making use of the linear independence of B-splines shows that the matrix on the right is the submatrix of $\boldsymbol{A}$ given by $(\alpha_{j,d}(i))_{i=\nu-d,j=\mu-d}^{\nu,\mu}$. Since $\nu - d \leq i \leq \nu$ equation (4.12) follows. Equation (4.13) now follows from (4.10). ∎

Note that if no new knots are inserted ($\boldsymbol{\tau} = \boldsymbol{t}$) then the two sets of B-spline coefficients $\boldsymbol{c}$ and $\boldsymbol{b}$ are obviously the same. Equation (4.13) then shows that

$$c_i = \boldsymbol{R}_{1,\boldsymbol{\tau}}^{\mu}(\tau_{i+1})\cdots\boldsymbol{R}_{d,\boldsymbol{\tau}}^{\mu}(\tau_{i+d})\boldsymbol{c}_d. \tag{4.16}$$

This simple observation will be useful later.

An example will illustrate the use of Theorem 4.7.

Example 4.8. We consider quadratic splines ($d=2$) on the knot vector $\boldsymbol{\tau} = (-1,-1,-1,0,1,1,1)$, and insert two new knots, at $-1/2$ and $1/2$ so $\boldsymbol{t} = (-1,-1,-1,-1/2,0,1/2,1,1,1)$. 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 $\boldsymbol{A}$ are given by

$$\boldsymbol{\alpha}_2(i) = \boldsymbol{R}_{1,\boldsymbol{\tau}}^3(t_{i+1})\boldsymbol{R}_{2,\boldsymbol{\tau}}^3(t_{i+2})$$

for $i = 1, \ldots, 4$. Since

$$\boldsymbol{R}_{1,\boldsymbol{\tau}}^3(x) = \begin{pmatrix} -x & 1+x \end{pmatrix}, \qquad \boldsymbol{R}_{2,\boldsymbol{\tau}}^3(x) = \begin{pmatrix} -x & 1+x & 0 \\ 0 & (1-x)/2 & (1+x)/2 \end{pmatrix},$$

we have from (4.12)

$$\boldsymbol{\alpha}_2(i) = \frac{1}{2}\begin{pmatrix} 2t_{i+1}t_{i+2}, & 1 - t_{i+1} - t_{i+2} - 3t_{i+1}t_{i+2}, & (1+t_{i+1})(1+t_{i+2}) \end{pmatrix}.$$

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 $\boldsymbol{A}$ 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 $\boldsymbol{A}$ we have to move to the interval $[\tau_4, \tau_5) = [0,1)$. Here we have

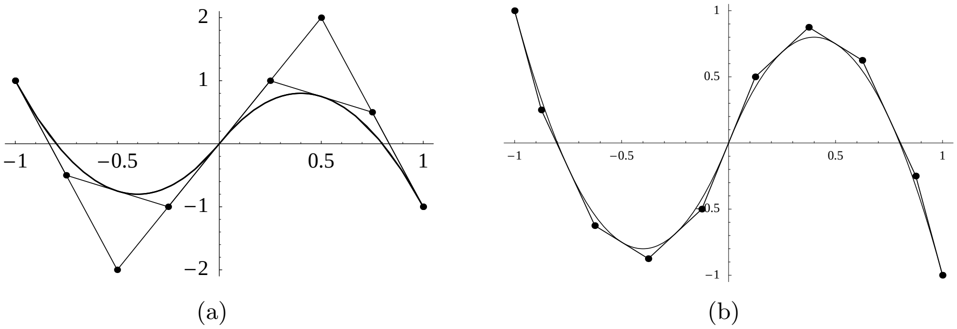
$$\boldsymbol{R}_{1,\boldsymbol{\tau}}^4(x) = \begin{pmatrix} 1-x & x \end{pmatrix} \qquad \boldsymbol{R}_{2,\boldsymbol{\tau}}^4(x) = \begin{pmatrix} (1-x)/2 & (1+x)/2 & 0 \\ 0 & 1-x & x \end{pmatrix},$$

so

$$\boldsymbol{a}_2(i) = \boldsymbol{R}_{1,\boldsymbol{\tau}}^4(t_{i+1})\boldsymbol{R}_{2,\boldsymbol{\tau}}^4(t_{i+2}) = \frac{1}{2}\begin{pmatrix} (1-t_{i+1})(1-t_{i+2}), & 1 + t_{i+1} + t_{i+2} - 3t_{i+1}t_{i+2}, & 2t_{i+1}t_{i+2} \end{pmatrix}.$$

Evaluating this for $i = 5$, 6 and inserting one zero as the first entry, we obtain the last two rows as

$$\begin{pmatrix} 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$



(a)

(b)

Figure 4.2. 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).

To see visually the effect of knot insertion, let $f = B_{1,2,\boldsymbol{\tau}} - 2B_{2,2,\boldsymbol{\tau}} + 2B_{3,2,\boldsymbol{\tau}} - B_{4,2,\boldsymbol{\tau}}$ be a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$ with B-spline coefficients $\boldsymbol{c} = (1,-2,2,-1)^T$. Its coefficients $\boldsymbol{b} = (b_i)_{i=1}^6$ are then given by

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 3/4 & 1/4 & 0 \\ 0 & 1/4 & 3/4 & 0 \\ 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ -2 \\ 2 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 \\ -1/2 \\ -1 \\ 1 \\ 1/2 \\ -1 \end{pmatrix}.$$

Figure 4.2 (a) shows a plot of f together with its control polygons relative to $\boldsymbol{\tau}$ and $\boldsymbol{t}$. We note that the control polygon relative to $\boldsymbol{t}$ 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 $\boldsymbol{t}$, we obtain the new knot vector

$$\boldsymbol{t}^1 = (-1,-1,-1,-3/4,-1/2,-1/4,0,1/4,1/2,3/4,1,1,1).$$

A plot of f and its control polygon relative to this knot vector is shown in Figure 4.2 (b).

Example 4.9. Let us again consider quadratic splines on a uniform knot vector with multiple knots at the ends,

$$\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+3} = (3,3,3,4,5,6,\ldots,n,n+1,n+1,n+1),$$

and form $\boldsymbol{t}$ by inserting one knot half way between each pair of old knots,

$$\boldsymbol{t} = (t_i)_{i=1}^{2n+1} = (3,3,3,7/2,4,9/2,5,\ldots,n,(2n+1)/2,n+1,n+1,n+1).$$

Since $\dim \mathbb{S}_{d,\boldsymbol{\tau}} = n$ and $\dim \mathbb{S}_{d,\boldsymbol{t}} = 2n-2$, the knot insertion matrix $\boldsymbol{A}$ 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 $\boldsymbol{A}$ 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 $4 \le \mu \le n-1$, we need the matrices $\boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}$ and $\boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}$ which are given by

$$\boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}(x) = \begin{pmatrix} \mu+1-x & x-\mu \end{pmatrix}, \qquad \boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}(x) = \begin{pmatrix} (\mu+1-x)/2 & (x+1-\mu)/2 & 0 \\ 0 & (\mu+2-x)/2 & (x-\mu)/2 \end{pmatrix}.$$

Observe that $\tau_i = i$ for $i = 3, \ldots, n+1$ and $t_i = (i+3)/2$ for $i = 3, \ldots, 2n-1$. Entries $\mu-2$, $\mu-1$ and μ of row $2\mu-3$ are therefore given by

$$\boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}(t_{2\mu-2})\boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}(t_{2\mu-1}) = \boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}(\mu+1/2)\boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}(\mu+1) = \begin{pmatrix} 1/2 & 1/2 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1/2 & 1/2 \end{pmatrix} = \begin{pmatrix} 0 & 3/4 & 1/4 \end{pmatrix}.$$

Similarly, entries $\mu-3$, $\mu-2$ and μ of row $2\mu-2$ are given by

$$\boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}(t_{2\mu-1})\boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}(t_{2\mu}) = \boldsymbol{R}^{\mu}_{1,\boldsymbol{\tau}}(\mu+1)\boldsymbol{R}^{\mu}_{2,\boldsymbol{\tau}}(\mu+3/2) = \begin{pmatrix}0 & 1\end{pmatrix}\begin{pmatrix}-1/4 & 5/4 & 0\\ 0 & 1/4 & 3/4\end{pmatrix} = \begin{pmatrix}0 & 1/4 & 3/4\end{pmatrix}.$$

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

$$\boldsymbol{A} = \begin{pmatrix}
1 & 0 & 0 & 0 & \dots & 0 & 0 & 0\\
1/2 & 1/2 & 0 & 0 & \dots & 0 & 0 & 0\\
0 & 3/4 & 1/4 & 0 & \dots & 0 & 0 & 0\\
0 & 1/4 & 3/4 & 0 & \dots & 0 & 0 & 0\\
0 & 0 & 3/4 & 1/4 & \dots & 0 & 0 & 0\\
0 & 0 & 1/4 & 3/4 & \dots & 0 & 0 & 0\\
\vdots & \vdots & \vdots & \vdots & \dots & \vdots & \vdots & \vdots\\
0 & 0 & 0 & 0 & \dots & 3/4 & 1/4 & 0\\
0 & 0 & 0 & 0 & \dots & 1/4 & 3/4 & 0\\
0 & 0 & 0 & 0 & \dots & 0 & 1/2 & 1/2\\
0 & 0 & 0 & 0 & \dots & 0 & 0 & 1
\end{pmatrix}.$$

The formula for $\boldsymbol{\alpha}_d(i)$ 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 $B_{j,d}(x)$,

$$\alpha_{j,d}(i) = \frac{t_{i+d}-\tau_j}{\tau_{j+d}-\tau_j}\alpha_{j,d-1}(i) + \frac{\tau_{j+1+d}-t_{i+d}}{\tau_{j+1+d}-\tau_{j+1}}\alpha_{j+1,d-1}(i), \tag{4.17}$$

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 $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ and $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ with common knots at the ends be given. To compute the $m \times n$ knot insertion matrix $\boldsymbol{A} = \big(\alpha_{j,d}(i)\big)_{i,j=1}^{m,n}$ from $\boldsymbol{\tau}$ to $\boldsymbol{t}$ perform the following steps:*

```
1. For $i = 1, \dots, m$.
    1.1 Determine $\mu$ such that $\tau_\mu \le t_i < \tau_{\mu+1}$.
    1.2 Compute entries $\mu-d, \dots, \mu$ of row $i$ by evaluating
        $\boldsymbol{\alpha}_d(i)^T = \big(\alpha_{\mu-d,d}(i), \dots, \alpha_{\mu,d}(i)\big)^T = \begin{cases}1, & \text{if } d = 0.\\ \boldsymbol{R}_1(t_{i+1})\cdots\boldsymbol{R}_d(t_{i+d}), & \text{if } d > 0.\end{cases}$
        All other entries in row $i$ are zero.
```

An algorithm for converting a spline from a B-spline representation in $\mathbb{S}_{d,\boldsymbol{\tau}}$ to $\mathbb{S}_{d,\boldsymbol{t}}$ is as follows.

Algorithm 4.11 (Oslo-Algorithm 2)**.** *Let the polynomial degree d, and the two $d+1$-regular knot vectors $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ and $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ with common knots at the ends be given together with the spline f in $\mathbb{S}_{d,\boldsymbol{\tau}}$ with B-spline coefficients $\boldsymbol{c} = (c_j)_{j=1}^{n}$. To compute the B-spline coefficients $\boldsymbol{b} = (b_i)_{i=1}^{m}$ of f in $\mathbb{S}_{d,\boldsymbol{t}}$ perform the following steps:*

```
1. For $i = 1, \ldots, m$.
    1.1 Determine $\mu$ such that $\tau_\mu \le t_i < \tau_{\mu+1}$.
    1.2 Set $\boldsymbol{c}_d = (c_j)_{j=\mu-d}^{\mu}$ and compute $b_i$ by evaluating
```

$$b_i = \begin{cases} c_\mu, & \text{if } d = 0. \\ \boldsymbol{R}_1(t_{i+1}) \cdots \boldsymbol{R}_d(t_{i+d})\boldsymbol{c}_d, & \text{if } d > 0. \end{cases}$$

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 $f = \sum_{j=1}^{n} c_j B_{j,d,\boldsymbol{\tau}}$ is a polynomial represented on a $d+1$-extended knot vector $\boldsymbol{\tau}$. On the knot interval $[\tau_\mu, \tau_{\mu+1})$ we know that f can be written as

$$f(x) = \boldsymbol{R}_1(x) \cdots \boldsymbol{R}_d(x)\boldsymbol{c}_d, \tag{4.18}$$

where $\boldsymbol{c}_d = (c_{\mu-d}, \ldots, c_\mu)^T$, see Section 2.3. 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

$$F(x_1, \ldots, x_d) = \boldsymbol{R}_1(x_1) \cdots \boldsymbol{R}_d(x_d)\boldsymbol{c}_d. \tag{4.19}$$

We recognise the right-hand side of this expression from equation (4.13) in Theorem 4.7: If we have a knot vector that includes the knots $(x_0, x_1, \ldots, x_d, x_{d+1})$, then $F(x_1, \ldots, x_d)$ gives the B-spline coefficient that multiplies the B-spline $B(x \mid x_0, \ldots, x_{d+1})$ in the representation of the polynomial f on the knot vector $\boldsymbol{x}$. When f is a polynomial, it turns out that the function F is completely independent of the knot vector $\boldsymbol{\tau}$ that underlie the definition of the $\boldsymbol{R}$-matrices in (4.19). 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 (4.19) 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 $f(x_1, \ldots, x_d)$ is said to be affine if it is affine viewed as a function of each x_i for $i = 1, \ldots, d$, 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

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y), \quad x, y \in \mathbb{R} \tag{4.20}$$

for all real numbers α and β, while for an affine function f with $b \neq 0$ equation (4.20) only holds if $\alpha + \beta = 1$. This is in fact a complete characterisation of affine functions: If (4.20) holds with $\alpha + \beta = 1$, then f is affine, see exercise 9.

A general affine function of 2 variables is given by

$$\begin{aligned} f(x_1, x_2) &= ax_2 + b = (a_2x_1 + b_2)x_2 + a_1x_1 + b_1 \\ &= c_0 + c_1x_1 + c_2x_2 + c_{1,2}x_1x_2. \end{aligned} \tag{4.21}$$

Similarly, an affine function of three variables is a function on the form

$$f(x_1, x_2, x_3) = c_0 + c_1x_1 + c_2x_2 + c_3x_3 + c_{1,2}x_1x_2 + c_{1,3}x_1x_3 + c_{2,3}x_2x_3 + c_{1,2,3}x_1x_2x_3.$$

In general, an affine function can be written as a linear combination of 2^d terms. This follows by induction as in (4.21) where we passed from one argument to two.

A symmetric and affine function satisfies the equation

$$f(x_1, x_2, \ldots, x_d) = f(x_{\pi_1}, x_{\pi_2}, \ldots, x_{\pi_d}),$$

for any permutation $(\pi_1, \pi_2, \ldots, \pi_d)$ of the numbers 1, 2, ..., d. We leave it as an exercise to show that symmetric, affine functions of two and three variables can be written in the form

$$\begin{aligned} f(x_1, x_2) &= a_0 + a_1(x_1 + x_2) + a_2x_1x_2, \\ f(x_1, x_2, x_3) &= a_0 + a_1(x_1 + x_2 + x_3) + a_2(x_1x_2 + x_1x_3 + x_2x_3) + a_3x_1x_2x_3. \end{aligned}$$

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 $\mathcal{B}[p](x_1, \ldots, x_d)$ of p is a function of d variables with the properties:*

1. *Symmetry. The blossom is a symmetric function of its arguments,*

   $$\mathcal{B}[p](x_1, \ldots, x_d) = \mathcal{B}[p](x_{\pi_1}, \ldots, x_{\pi_d})$$

   for any permutation π_1, ..., π_d of the integers 1, ..., d.

2. *Affine. The blossom is affine in each of its variables,*

   $$\mathcal{B}[p](\ldots, \alpha x + \beta y, \ldots) = \alpha\mathcal{B}[p](\ldots, x, \ldots) + \beta\mathcal{B}[p](\ldots, y, \ldots)$$

   whenever $\alpha + \beta = 1$.

3. *Diagonal property. The blossom agrees with p on the diagonal,*

$$\mathcal{B}[p](x, \ldots, x) = p(x)$$

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 $\mathcal{B}[p](x_1, \ldots, x_d)$. 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*

$$\mathcal{B}[c_1p_1 + c_2p_2](x_1, \ldots, x_d) = c_1\mathcal{B}[p_1](x_1, \ldots, x_d) + c_2\mathcal{B}[p_2](x_1, \ldots, x_d). \tag{4.22}$$

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

$$F(x_1, \ldots, x_d) = c_0 + \sum_{j=1}^{d} \sum_{1 \leq i_1 < \cdots < i_j \leq d} c_{i_1,\ldots,i_j} x_{i_1} \cdots x_{i_j}.$$

Symmetry forces all the coefficients multiplying terms of the same degree to be identical. To see this we note first that

$$F(1, 0, \ldots, 0) = c_0 + c_1 = F(0, \ldots, 1, \ldots, 0) = c_0 + c_i$$

for all i with $1 \leq i \leq d$. Hence we have $c_1 = \cdots = c_d$. 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

$$p_{j-1} + c_{1,2,\ldots,j} = p_{j-1} + c_{1,2,\ldots,j-1,j+1} = \cdots = p_{j-1} + c_{d-j+1,\ldots,d}.$$

From this we conclude that all the coefficients multiplying terms of degree j must be equal. We can therefore write F as

$$F(x_1, \ldots, x_d) = a_0 + \sum_{j=1}^{d} a_j \sum_{1 \leq i_1 < \cdots < i_j \leq d} x_{i_1} \cdots x_{i_j}, \tag{4.23}$$

for suitable constants $(a_j)_{j=0}^{d}$. From the diagonal property $F(x, \ldots, x) = f(x)$ the coefficients $(a_j)_{j=0}^{d}$ are all uniquely determined (since 1, x, …, 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 (4.22) is affine in each of the x_i, it is symmetric, and it reduces to $c_1p_1(x) + c_2p_2(x)$ on the diagonal $x_1 = \cdots = x_d = x$. ∎

Recall that the elementary symmetric polynomials

$$s_j(x_1, \ldots, x_d) = \Bigl(\sum_{1 \leq i_1 < \cdots < i_j \leq d} x_{i_1} x_{i_2} \cdots x_{i_j}\Bigr) \Big/ \binom{d}{j}$$

that appear in (4.23) (apart from the binomial coefficient) agree with the B-spline coefficients of the polynomial powers,

$$\sigma_{k,d}^{j} = s_j(\tau_{k+1}, \ldots, \tau_{k+d}),$$

see Corollary 3.5. In fact, the elementary symmetric polynomials are the blossoms of the powers,

$$\mathcal{B}[x^j](x_1, \ldots, x_d) = s_j(x_1, \ldots, x_d) \qquad \text{for } j = 0,\, \ldots,\, d.$$

They can also be defined by the relation

$$(x - x_1) \cdots (x - x_d) = \sum_{k=0}^{d} (-1)^{d-k} \binom{d}{k} s_{d-k}(x_1, \ldots, x_d) x^k.$$

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 $\boldsymbol{\tau}$ for B-splines of degree d. On the interval $[\tau_\mu, \tau_{\mu+1})$ the only nonzero B-splines are $\boldsymbol{B}_d = (B_{\mu-d,d}, \ldots, B_{\mu,d})^T$ which can be expressed in terms of matrices as

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

If we consider the polynomial piece $f = \boldsymbol{B}_d^T \boldsymbol{c}_d$ with coefficients $\boldsymbol{c}_d = (c_{\mu-d}, \ldots, c_\mu)^T$ we can define a function F of d variables by

$$F(x_1, \ldots, x_d) = \boldsymbol{R}_1(x_1) \cdots \boldsymbol{R}_d(x_d) \boldsymbol{c}_d. \tag{4.24}$$

From equation(4.13) we recognise $F(x_1, \ldots, x_d)$ as the coefficient multiplying a B-spline with knots x_0, x_1, …, x_{d+1} in the representation of the polynomial f.

Equation (3.7) in Lemma 3.3 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 2.18 that $\boldsymbol{R}_1 = \boldsymbol{R}_{1,\boldsymbol{\tau}}$ is given by

$$\boldsymbol{R}_1(x) = \left(\frac{\tau_{\mu+1} - x}{\tau_{\mu+1} - \tau_\mu}, \quad \frac{x - \tau_\mu}{\tau_{\mu+1} - \tau_\mu} \right)$$

which is obviously an affine function of x.

The function F is also related to the polynomial f in that $F(x, \ldots, x) = f(x)$. 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ on the interval $[\tau_\mu, \tau_{\mu+1})$, with coefficients $\boldsymbol{c}_d = (c_{\mu-d}, \ldots, c_\mu)^T$. Then the function*

$$F(x_1, \ldots, x_d) = \boldsymbol{R}_1(x_1) \cdots \boldsymbol{R}_d(x_d) \boldsymbol{c}_d$$

is symmetric and affine, and agrees with f on the diagonal,

$$F(x, \ldots, x) = f(x).$$

Lemma 4.15 and Theorem 4.14 show that the blossom of f is given by

$$\mathcal{B}[f](x_1,\dots,x_d) = \boldsymbol{R}_1(x_1)\cdots\boldsymbol{R}_1(x_d)\boldsymbol{c}_d.$$

Blossoming can be used to give explicit formulas for the B-spline coefficients of a spline.

Theorem 4.16. *Let $f = \sum_{j=1}^{n} c_j B_{j,d,\boldsymbol{\tau}}$ be a spline on a $d+1$-regular knot vector $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$. It's B-spline coefficients are then given by*

$$c_j = \mathcal{B}[f_k](\tau_{j+1},\dots,\tau_{j+d}), \quad \text{for } k = j,\, j+1,\, \dots,\, j+d, \tag{4.25}$$

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 $\mathcal{B}[f_\mu](x_1,\dots,x_d) = \boldsymbol{R}_1(x_1)\cdots\boldsymbol{R}_d(x_d)\boldsymbol{c}_d$, where $\boldsymbol{c}_d = (c_j)_{j=\mu-d}^{\mu}$ are the B-spline coefficients of f active on the interval $[\tau_\mu,\tau_{\mu+1})$. From (4.16) we then obtain

$$c_j = \mathcal{B}[f_\mu](\tau_{j+1},\dots,\tau_{j+d}) \tag{4.26}$$

which is (4.25) in this special situation.

To prove (4.25) in general, fix j and choose the integer k in the range $j \le k \le j+d$. We then have

$$f_k(x) = \sum_{i=k-d}^{k} c_i B_{i,d}(x), \tag{4.27}$$

By the choice of k we see that the sum in (4.27) includes the term $c_j B_{j,d}$. Equation (4.25) therefore follows by applying (4.26) 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*

$$\mathcal{B}[p](\dots,x,\dots) = \frac{b-x}{b-a}\mathcal{B}[p](\dots,a\dots) + \frac{x-a}{b-a}\mathcal{B}[p](\dots,b,\dots) \tag{4.28}$$

for all real numbers a, b and x with $a \ne b$.

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 (4.28) then follows from the affine property of the blossom. ∎

The next result will be useful later.

Lemma 4.18. *Let $\mathcal{B}_x\big[p(x,y)\big]$ denote the blossom of p with respect to the variable x. Then*

$$\mathcal{B}_x\big[(y-x)^k\big](x_1,\dots,x_d) = \frac{k!}{d!}D^{d-k}\big((y-x_1)\cdots(y-x_d)\big), \tag{4.29}$$

for $k = 0, 1, \ldots, d$, and

$$\mathcal{B}_x\bigl[(y_1 - x)\cdots(y_\ell - x)\bigr](x_1,\ldots,x_d) = \frac{(d-\ell)!}{d!}\sum_{1\le i_1,\ldots,i_\ell\le d}(y_1 - x_{i_1})\cdots(y_\ell - x_{i_\ell}), \quad (4.30)$$

where the sum is over all distinct choices $i_1, \ldots, i_\ell$ of ℓ integers from the d integers $1, \ldots, d$.

Proof. For $k = d$ equation (4.29) 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 (4.30) 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 $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ be a given knot vector and let $\boldsymbol{t} = (t_i)_{i=1}^{n+d+2}$ be the knot vector obtained by inserting a knot z in $\boldsymbol{\tau}$ in the interval $[\tau_\mu, \tau_{\mu+1})$. If*

$$f = \sum_{j=1}^{n} c_j B_{j,d,\boldsymbol{\tau}} = \sum_{i=1}^{n+1} b_i B_{i,d,\boldsymbol{t}},$$

then $(b_i)_{i=1}^{n+1}$ can be expressed in terms of $(c_j)_{j=1}^{n}$ through the formulas

$$b_i = \begin{cases} c_i, & \text{if } 1 \le i \le \mu - d;\\ \dfrac{z - \tau_i}{\tau_{i+d} - \tau_i}c_i + \dfrac{\tau_{i+d} - z}{\tau_{i+d} - \tau_i}c_{i-1}, & \text{if } \mu - d + 1 \le i \le \mu;\\ c_{i-1}, & \text{if } \mu + 1 \le i \le n+1. \end{cases} \quad (4.31)$$

Proof. Observe that for $j \le \mu$ we have $\tau_j = t_j$. For $i \le \mu - d$ and with k an integer such that $i \le k \le i + d$ it therefore follows from (4.25) that

$$b_i = \mathcal{B}[f^k](t_{i+1},\ldots,t_{i+d}) = \mathcal{B}[f^k](\tau_{i+1},\ldots,\tau_{i+d}) = c_i.$$

Similarly, we have $t_i = \tau_{i-1}$ for $i \ge \mu + 1$ so

$$b_i = \mathcal{B}[f^k](t_{i+1},\ldots,t_{i+d}) = \mathcal{B}[f^k](\tau_i,\ldots,\tau_{i+d-1}) = c_{i-1}$$

for such values of i.

When i satisfies $\mu - d + 1 \le i \le \mu$ we note that z will appear in the sequence $(t_{i+1},\ldots,t_{i+d})$. From (4.25) we therefore obtain

$$b_i = \mathcal{B}[f^\mu](t_{i+1},\ldots,z,\ldots,t_{i+d}) = \mathcal{B}[f^\mu](\tau_{i+1},\ldots,z,\ldots,\tau_{i+d-1})$$

since we now may choose $k = \mu$. Applying Lemma 4.17 with $x = z$, $a = \tau_i$ and $b = \tau_{i+d}$ yields

$$b_i = \frac{\tau_{i+d} - z}{\tau_{i+d} - \tau_i} \mathcal{B}[f^\mu](\tau_{i+1}, \ldots, \tau_i, \ldots, \tau_{i+d}) + \frac{z - \tau_i}{\tau_{i+d} - \tau_i} \mathcal{B}[f^\mu](\tau_i, \ldots, \tau_{i+d}, \ldots, \tau_{i+d-1}).$$

Exploiting the symmetry of the blossom and again applying (4.25) leads to the middle formula in (4.31). ■

It is sometimes required to insert the same knot several times; this can of course be accomplished by applying the formulas in (4.31) several times. Since blossoms have the property $\mathcal{B}[f](z, \ldots, z) = f(z)$, 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 $(d = 3)$,

$$\begin{array}{c}
\cdots \quad c^0_{\mu-4} \quad c^0_{\mu-3} \quad c^0_{\mu-2} \quad c^0_{\mu-1} \quad c^0_{\mu} \quad c^0_{\mu+1} \quad \cdots \\
c^1_{\mu-2} \quad c^1_{\mu-1} \quad c^1_{\mu} \\
c^2_{\mu-1} \quad c^2_{\mu} \\
c^3_{\mu}
\end{array}$$

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

$$(\ldots, c^0_{\mu-4}, c^0_{\mu-3}, c^1_{\mu-2}, c^1_{\mu-1}, c^1_{\mu}, c^0_{\mu}, c^0_{\mu+1}, \ldots),$$

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,

$$(\ldots, c^0_{\mu-4}, c^0_{\mu-3}, c^1_{\mu-2}, c^2_{\mu-1}, c^2_{\mu}, c^1_{\mu}, c^0_{\mu}, c^0_{\mu+1}, \ldots).$$

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^3_μ 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 (4.31) indicates that the knot insertion matrix $\boldsymbol{A}$ must have a simple structure when only one knot is inserted. Setting $\boldsymbol{c} = (c_i)_{i=1}^n$ and $\boldsymbol{b} = (b_i)_{i=1}^{n+1}$

and remembering that $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c}$, we see that $\boldsymbol{A}$ is given by the $(n+1) \times n$ matrix

$$\boldsymbol{A} = \begin{pmatrix} 1 & 0 & & & & & & \\ & \ddots & \ddots & & & & & \\ & & 1 & 0 & & & & \\ & & 1-\lambda_{\mu-d+1} & \lambda_{\mu-d+1} & & & & \\ & & & \ddots & \ddots & & & \\ & & & & 1-\lambda_{\mu} & \lambda_{\mu} & & \\ & & & & 0 & 1 & & \\ & & & & & \ddots & \ddots & \\ & & & & & & 0 & 1 \end{pmatrix}, \tag{4.32}$$

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 $\boldsymbol{A}$ are nonnegative. This property generalises to arbitrary knot insertion matrices.

Lemma 4.20. *Let $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+d+1}$ and $\boldsymbol{t} = (t_i)_{i=1}^{m+d+1}$ be two knot vectors for splines of degree d with $\boldsymbol{\tau} \subseteq \boldsymbol{t}$. All the entries of the knot insertion matrix $\boldsymbol{A}$ from $\mathbb{S}_{d,\boldsymbol{\tau}}$ to $\mathbb{S}_{d,\boldsymbol{t}}$ are nonnegative and $\boldsymbol{A}$ can be factored as*

$$\boldsymbol{A} = \boldsymbol{A}_{m-n}\boldsymbol{A}_{m-n-1} \cdots \boldsymbol{A}_1, \tag{4.33}$$

where $\boldsymbol{A}_i$ 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 $\boldsymbol{t}$ but not in $\boldsymbol{\tau}$ by $(z_i)_{i=1}^{m-n}$. Set $\boldsymbol{t}^0 = \boldsymbol{\tau}$ and $\boldsymbol{t}^i = \boldsymbol{t}^{i-1} \cup (z_i)$ for $i = 1, \ldots, m-n$. Denote by $\boldsymbol{A}_i$ the knot insertion matrix from $\boldsymbol{t}^{i-1}$ to $\boldsymbol{t}^i$. By applying Böhm's method $m - n$ times we obtain (4.33). Since all the entries in each of the matrices $\boldsymbol{A}_i$ are nonnegative the same must be true of $\boldsymbol{A}$. ∎

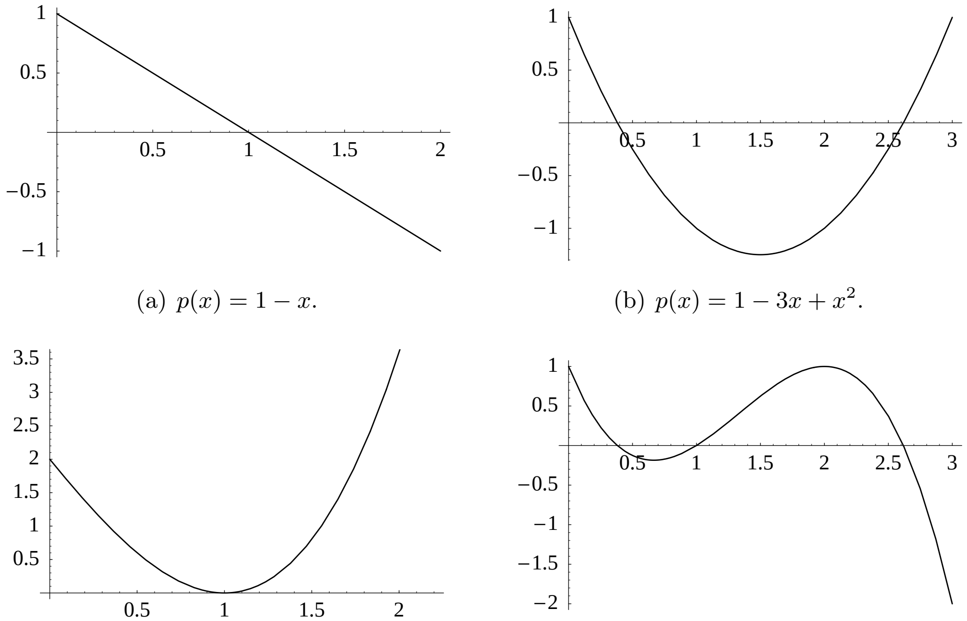
4.5 Bounding the number of sign changes in a spline

In this section we will make use of Böhm's method for knot insertion to prove that the number of spline 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 seldom 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 $\boldsymbol{c} = (c_i)_{i=1}^{n}$ be a vector of real numbers. The number of sign changes in $\boldsymbol{c}$ (zeros are ignored) is denoted $S^-(\boldsymbol{c})$. 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 $S^-\big(f(x_1), \ldots, f(x_{r+1})\big) = r$.*

Example 4.22. Let us consider some simple examples of counting sign changes. It is easily checked that

$$\begin{aligned} S^-(1,-2) &= 1, & S^-(1,0,-1,3) &= 2, \\ S^-(1,0,2) &= 0, & S^-(2,0,0,0,-1) &= 1, \\ S^-(1,-1,2) &= 2, & S^-(2,0,0,0,1) &= 0. \end{aligned}$$



(a) $p(x) = 1 - x$.

(b) $p(x) = 1 - 3x + x^2$.

(c) $p(x) = 2 - 3x + x^2$.

(d) $p(x) = 1 - 4x + 4x^2 - x^3$.

Figure 4.3. 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.

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 $\boldsymbol{c} = (c_0, \ldots, c_d)^T$, and let $Z(p)$ denote the total number of zeros of p in the interval $(0, \infty)$, counted with multiplicities. Then*

$$Z(p) \leq S^-(\boldsymbol{c}),$$

i.e., the number of zeros of p is bounded by the number of sign changes in its coefficients.

Figures 4.3 (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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ and $\mathbb{S}_{d,\boldsymbol{t}}$ be two spline spaces of degree d, with $\mathbb{S}_{d,\boldsymbol{\tau}} \subseteq \mathbb{S}_{d,\boldsymbol{t}}$. Recall from Section 4.4 that to get from the knot vector $\boldsymbol{\tau}$ to the refined knot vector $\boldsymbol{t}$, we can insert one knot at a time. If there are ℓ

more knots in $\boldsymbol{\tau}$ than in $\boldsymbol{t}$, this leads to a factorisation of the knot insertion matrix $\boldsymbol{A}$ as

$$\boldsymbol{A} = \boldsymbol{A}_\ell \boldsymbol{A}_{\ell-1} \cdots \boldsymbol{A}_1, \tag{4.34}$$

where $\boldsymbol{A}_k$ is a $(n+k) \times (n+k-1)$ matrix for $k = 1, \ldots, \ell$, if $\dim \mathbb{S}_{d,\boldsymbol{\tau}} = n$. Each of the matrices $\boldsymbol{A}_k$ corresponds to insertion of only one knot, and all the nonzero entries of the bi-diagonal matrix $\boldsymbol{A}_k$ are found in positions (i,i) and $(i+1,i)$ for $i = 1, \ldots, n+k-1$, 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ and $\mathbb{S}_{d,\boldsymbol{t}}$ be two spline spaces such that $\boldsymbol{t}$ is a refinement of $\boldsymbol{\tau}$. Let $f = \sum_{j=1}^n c_j B_{j,d,\boldsymbol{\tau}} = \sum_{i=1}^m b_i B_{i,d,\boldsymbol{t}}$ be a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$ with B-spline coefficients $\boldsymbol{c}$ in $\mathbb{S}_{d,\boldsymbol{\tau}}$ and $\boldsymbol{b}$ in $\mathbb{S}_{d,\boldsymbol{t}}$. Then $\boldsymbol{b}$ has no more sign changes than $\boldsymbol{c}$, i.e.,*

$$S^-(\boldsymbol{A}\boldsymbol{c}) = S^-(\boldsymbol{b}) \le S^-(\boldsymbol{c}), \tag{4.35}$$

where $\boldsymbol{A}$ is the knot insertion matrix from $\boldsymbol{\tau}$ to $\boldsymbol{t}$.

Proof. Since we can insert the knots one at a time, it clearly suffices to show that (4.35) holds in the case where there is only one more knot in $\boldsymbol{t}$ than in $\boldsymbol{\tau}$. In this case we know from Lemma 4.19 that $\boldsymbol{A}$ is bidiagonal so

$$b_i = \alpha_{i-1}(i)c_{i-1} + \alpha_i(i)c_i, \qquad \text{for } i = 1, \ldots n+1,$$

where $\big(\alpha_j(i)\big)_{i,j=1}^{n+1,n}$ are the entries of $\boldsymbol{A}$ (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

$$S^-(\boldsymbol{c}) = S^-(b_1, c_1, b_2, c_2, \ldots, b_{n-1}, c_{n-1}, b_n, c_n, b_{n+1}) \ge S^-(\boldsymbol{b}).$$

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.

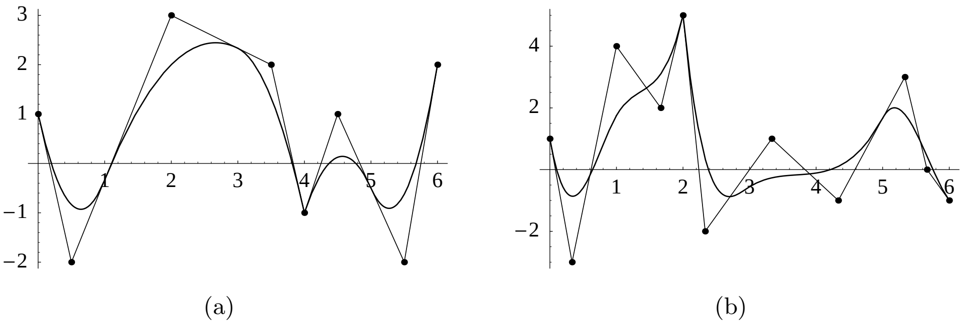
Theorem 4.25. *Let $f = \sum_{j=1}^n c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$. Then*

$$S^-(f) \le S^-(\boldsymbol{c}) \le n-1. \tag{4.36}$$

Proof. Suppose that $S^-(f) = \ell$, and let $(x_i)_{i=1}^{\ell+1}$ be $\ell+1$ points chosen so that $S^-(f) = S^-\big(f(x_1), \ldots, f(x_{\ell+1})\big)$. We form a new knot vector $\boldsymbol{t}$ that includes $\boldsymbol{\tau}$ as a subsequence, but in addition each of the x_i occurs exactly $d+1$ times in $\boldsymbol{t}$. From our study of knot insertion we know that f may be written $f = \sum_j b_j B_{j,d,\boldsymbol{t}}$ for suitable coefficients (b_j), and from Lemma 2.6 we know that each of the function values $f(x_i)$ will appear as a B-spline coefficient in $\boldsymbol{b}$. We therefore have

$$S^-(f) \le S^-(\boldsymbol{b}) \le S^-(\boldsymbol{c}),$$

the last inequality following from Lemma 4.24. The last inequality in (4.36) follows since an n-vector can only have $n-1$ sign changes. ∎



(a) (b)

Figure 4.4. A quadratic spline (a) and a cubic spline (b) with their control polygons.

The validity of Theorem 4.25 can be checked with the two plots in Figure 4.4 as well as all other figures which include both a spline function and its control polygon.

Exercises for Chapter 4

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

$$B_j^d(x) = \binom{d}{j} x^j (1-x)^{d-j}, \qquad \text{for } j = 0,\, 1,\, \dots,\, d. \tag{4.37}$$

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

$$(a+b)^d = \sum_{k=0}^d \binom{d}{k} a^k b^{d-k} \tag{4.38}$$

will be useful.

a) By applying (4.38), show that

$$B_j^d(x) = \sum_{i=j}^d (-1)^{i-j} \binom{d}{j} \binom{d-j}{i-j} x^i, \qquad \text{for } j = 0,\, 1,\, \dots,\, d.$$

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 $\boldsymbol{B}_d = \big(B_0^d(x), \dots, B_d^d(x)\big)^T$ and $\boldsymbol{P}_d = (1, x, \dots, x^d)^T$ are related by $\boldsymbol{B}_d^T = \boldsymbol{P}_d^T \boldsymbol{A}_d$ where $\boldsymbol{A}_d$ is a $(d+1) \times (d+1)$-matrix $\boldsymbol{A}_d$. Show that the entries of $\boldsymbol{A}_d = (a_{i,j})_{i,j=0}^d$ 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 $\boldsymbol{A}_d$ satisfy the recurrence relation

$$a_{i,j} = \beta_i \left(a_{i-1,j-1} - a_{i-1,j}\right), \qquad \text{where } \beta_i = (d-i+1)/i.$$

Give a detailed algorithm for computing $\boldsymbol{A}_d$ based on this formula.

d) Explain how we can find the coefficients of a polynomial relative to the Monomial basis if $\boldsymbol{A}_d$ 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 (4.38), show that for all x and t in $\mathbb{R}$ we have

$$\bigl(tx + (1-x)\bigr)^d = \sum_{k=0}^{d} B_k^d(x) t^k. \tag{4.39}$$

The function $G(t) = \bigl(tx + (1-x)\bigr)^d$ 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 (4.39).

c) Find two different expressions for $G^{(j)}(1)/j!$ and show that this leads to the formulas

$$\binom{d}{j} x^j = \sum_{i=j}^{d} \binom{i}{j} B_k^d(x), \qquad \text{for } j = 0, \ldots, d. \tag{4.40}$$

d) Show that the entries of the matrix $\boldsymbol{B}_d = (b_{i,j})_{i,j=0}^d$ such that $\boldsymbol{P}_d^T = \boldsymbol{B}_d^T \boldsymbol{B}_d$ 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 $\boldsymbol{P}$ denote the cubic Bernstein basis on the interval $[0,1]$ and let $\boldsymbol{Q}$ denote the cubic Bernstein basis on the interval $[2,3]$. Determine the matrix $\boldsymbol{A}_3$ such that $\boldsymbol{P}(x)^T = \boldsymbol{Q}(x)^T \boldsymbol{A}_3$ for all real numbers x.

4.4 Let $\boldsymbol{A}$ denote the knot insertion matrix for the linear ($d = 1$) B-splines on $\boldsymbol{\tau} = (\tau_j)_{j=1}^{n+2}$ to the linear B-splines in $\boldsymbol{t} = (t_i)_{i=1}^{m+2}$. We assume that $\boldsymbol{\tau}$ and $\boldsymbol{t}$ are 2-extended with $\tau_1 = t_1$ and $\tau_{n+2} = t_{m+2}$ and $\boldsymbol{\tau} \subseteq \boldsymbol{t}$.

a) Determine $\boldsymbol{A}$ when $\boldsymbol{\tau} = (0, 0, 1/2, 1, 1)$ and $\boldsymbol{t} = (0, 0, 1/4, 1/2, 3/4, 1, 1)$.

b) Device a detailed algorithm that computes $\boldsymbol{A}$ for general $\boldsymbol{\tau}$ and $\boldsymbol{t}$ and requires $O(m)$ operations.

c) Show that the matrix $\boldsymbol{A}^T \boldsymbol{A}$ is tridiagonal.

4.5 Prove Lemma 4.4 in the general case where $\boldsymbol{\tau}$ and $\boldsymbol{t}$ are not $d+1$-regular. Hint: Augment both $\boldsymbol{\tau}$ and $\boldsymbol{t}$ 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 $\boldsymbol{\tau}$ and $\boldsymbol{t}$ with $d+1$ identical knots at both ends to obtain new knot vectors $\hat{\boldsymbol{\tau}}$ and $\hat{\boldsymbol{t}}$. The knot insertion matrix from $\boldsymbol{\tau}$ to $\boldsymbol{t}$ can then be identified as a sub-matrix of the knot insertion matrix from $\hat{\boldsymbol{\tau}}$ to $\hat{\boldsymbol{t}}$.

4.7 Show that if $\boldsymbol{\tau}$ and $\boldsymbol{t}$ are $d+1$-regular knot vectors with $\boldsymbol{\tau} \subseteq \boldsymbol{t}$ 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

$$f(\alpha x + \beta y) = \alpha f(x) + \beta f(y) \tag{4.41}$$

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 (4.41) 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.

CHAPTER 5

Spline Approximation of Functions and Data

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 important 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 $i = 1, \ldots, m-1$, and construct the 2-regular knot vector $\boldsymbol{t}$ as*

$$\boldsymbol{t} = (t_i)_{i=1}^{m+2} = (x_1, x_1, x_2, x_3, \ldots, x_{m-1}, x_m, x_m).$$

Then the linear spline g given by

$$g(x) = \sum_{i=1}^{m} y_i B_{i,1}(x)$$

satisfies the interpolation conditions

$$g(x_i) = y_i, \quad \text{for } i = 1, \ldots, m-1, \qquad \text{and} \quad \lim_{x \to x_m^-} g(x) = y_m. \tag{5.1}$$

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 $i = 1, \ldots, m$, 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 $1 \le i \le m$ is given by

$$B_{i,1}(x) = \begin{cases} (x - x_{i-1})/(x_i - x_{i-1}), & \text{if } x_{i-1} \le x < x_i, \\ (x_{i+1} - x)/(x_{i+1} - x_i), & \text{if } x_i \le x < x_{i+1}, \\ 0, & \text{otherwise,} \end{cases}$$

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 $\lim_{x \to x_m^-} B_{m,1}(x) = 1$, 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 $\Delta\boldsymbol{x} = \max_{1 \le i \le m-1}\{x_{i+1} - x_i\}$.*

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 $x \in [a,b]$,*

$$|f(x) - (I_1 f)(x)| \le \frac{1}{8}(\Delta\boldsymbol{x})^2 \max_{a \le z \le b} |f''(z)|, \tag{5.2}$$

$$|f'(x) - (I_1 f)'(x)| \le \frac{1}{2}\Delta\boldsymbol{x} \max_{a \le z \le b} |f''(z)|. \tag{5.3}$$

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 $\Delta\boldsymbol{x}$ 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 $\Delta\boldsymbol{x}$ is raised to the power of two in (5.2).

The bounds in Proposition 5.2 depend both on $\Delta\boldsymbol{x}$ 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.,

$$|f''(x)| \le M, \quad \text{for} \quad x \in [a,b], \tag{5.4}$$

we can determine a value of $\Delta\boldsymbol{x}$ such that the error, measured as in (5.2), is smaller than some given tolerance ϵ. We must clearly require $(\Delta\boldsymbol{x})^2 M/8 < \epsilon$. This inequality holds provided $\Delta\boldsymbol{x} < \sqrt{8\epsilon/M}$. We conclude that for any $\epsilon > 0$, we have the implication

$$\Delta\boldsymbol{x} < \sqrt{\frac{8\epsilon}{M}} \quad \Longrightarrow \quad |f(x) - I_1 f(x)| < \epsilon, \quad \text{for } x \in [x_1, x_m]. \tag{5.5}$$

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 $f \in \mathbb{S}_{d,\boldsymbol{t}}$ for some d and $\boldsymbol{t}$ 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 $\boldsymbol{t}$ 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 $\Delta \boldsymbol{x}$ 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* $a = x_1 < x_2 < \cdots < x_m = b$ *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*

$$g(x_i) = f(x_i), \quad \text{and} \quad g'(x_i) = f'(x_i), \quad \text{for } i = 1, \ldots, m. \tag{5.6}$$

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

$$\boldsymbol{\tau} = (\tau_i)_{i=1}^{2m+4} = (x_1, x_1, x_1, x_1, x_2, x_2, \ldots, x_{m-1}, x_{m-1}, x_m, x_m, x_m, x_m), \tag{5.7}$$

which we call *the Cubic Hermite knot vector* on $\boldsymbol{x} = (x_1, \ldots, x_m)$. 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*

$$Hf = \sum_{i=1}^{2m} c_i B_{i,3}, \tag{5.8}$$

where

$$\left.\begin{aligned} c_{2i-1} &= f(x_i) - \frac{1}{3}\Delta x_{i-1} f'(x_i), \\ c_{2i} &= f(x_i) + \frac{1}{3}\Delta x_i f'(x_i), \end{aligned}\right\} \qquad \text{for } i = 1, \ldots, m, \tag{5.9}$$

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* $x_{m+1} = x_m$.

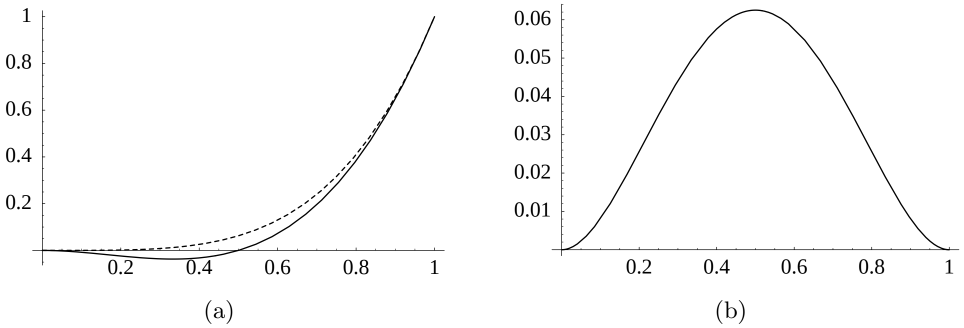


Figure 5.1. Figure (a) shows the cubic Hermite interpolant (solid) to $f(x) = x^4$ (dashed), see Example 5.6, while the error in this approximation is shown in (b).

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 $\boldsymbol{x} = (0, 1/2, 1)$. In this case the cubic Hermite knots are $\boldsymbol{\tau} = (0, 0, 0, 0, 1/2, 1/2, 1, 1, 1, 1)$, and we find the coefficients $\boldsymbol{c} = (0, 0, -1/48, 7/48, 1/3, 1)$. 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*

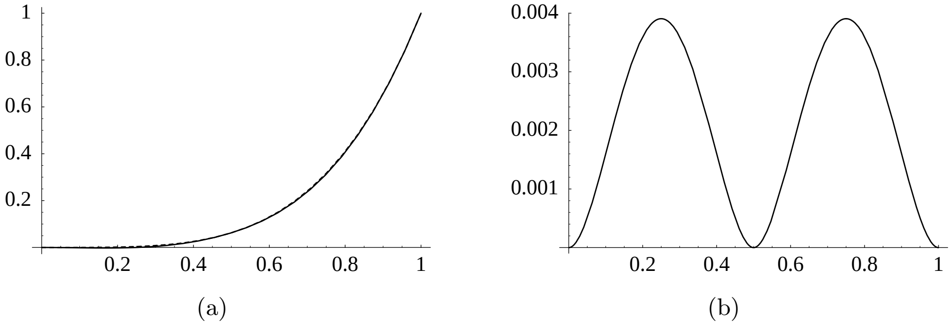


Figure 5.2. Figure (a) shows the cubic Hermite interpolant (solid) to $f(x) = x^4$ (dashed) with two polynomial pieces, see Example 5.7, while the error in the approximation is shown in (b).

interpolation satisfies the conditions

$$\left.\begin{aligned} f(x_i) - \frac{1}{3}\Delta x_{i-1} f'(x_i) &\geq 0, \\ f(x_i) + \frac{1}{3}\Delta x_i f'(x_i) &\geq 0, \end{aligned}\right\} \qquad \textit{for } i = 1, \ldots, m.$$

Then the cubic Hermite interpolant Hf is nonnegative on $[a, b]$.

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 $\boldsymbol{x}$ 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*

$$|f(x) - (Hf)(x)| \leq \frac{1}{384}(\Delta \boldsymbol{x})^4 \max_{a \leq z \leq b} |f^{(iv)}(z)|, \quad \textit{for } x \in [a, b]. \tag{5.10}$$

This estimate also holds whenever f is in some spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\Delta \boldsymbol{x}$ 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

$$\Delta\boldsymbol{x} \le \left(\frac{384\epsilon}{M}\right)^{1/4} \qquad \Longrightarrow \qquad |f(x) - (Hf)(x)| \le \epsilon, \quad \text{for } x \in [a, b].$$

When $\epsilon \to 0$, 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 $\Delta\boldsymbol{x}$ 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

$$\delta_j = \bigl(f(x_{j+1}) - f(x_j)\bigr)/\Delta x_j.$$

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

$$p_i'(x_i) = \frac{\Delta x_{i-1}\delta_i + \Delta x_i\delta_{i-1}}{\Delta x_{i-1} + \Delta x_i}, \quad \text{for } i = 2, \ldots, m-1, \tag{5.11}$$

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 $a = x_1 < x_2 < \cdots < x_m = b$ 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.

$$\begin{array}{rll} \text{(i)} & g'(a) = f'(a) \text{ and } g'(b) = f'(b); & \text{H(ermite)} \\ \text{(ii)} & g''(a) = g''(b) = 0; & \text{N(atural)} \\ \text{(iii)} & g''' \text{ is continuous at } x_2 \text{ and } x_{m-1}. & \text{F(ree)} \\ \text{(iv)} & D^j g(a) = D^j g(b) \text{ for } j = 1,\, 2. & \text{P(eriodic)} \end{array} \tag{5.12}$$

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 F, we therefore define the 4-regular knot vectors

$$\begin{aligned} \boldsymbol{\tau}_H = \boldsymbol{\tau}_N = (\tau_i)_{i=1}^{m+6} &= (x_1, x_1, x_1, x_1, x_2, x_3, \ldots, x_{m-1}, x_m, x_m, x_m, x_m), \\ \boldsymbol{\tau}_F = (\tau_i)_{i=1}^{m+4} &= (x_1, x_1, x_1, x_1, x_3, x_4, \ldots, x_{m-2}, x_m, x_m, x_m, x_m). \end{aligned} \tag{5.13}$$

This leads to three cubic spline spaces $\mathbb{S}_{3,\boldsymbol{\tau}_H}$, $\mathbb{S}_{3,\boldsymbol{\tau}_N}$ and $\mathbb{S}_{3,\boldsymbol{\tau}_F}$, all of which will have two continuous derivatives at each interior knot. Note that x_2 and x_{m-1} are missing in $\boldsymbol{\tau}_F$. This means that any $h \in \mathbb{S}_{3,\boldsymbol{\tau}_F}$ 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 $a = x_1 < x_2 < \cdots < x_m = b$ 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 $\mathbb{S}_{3,\boldsymbol{\tau}_Z}$, such that $g(x_i) = f(x_i)$ for $i = 1, 2, \ldots, m$, 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 $\boldsymbol{u}$ and $\boldsymbol{v}$ are two vectors in $\mathbb{R}^n$, then their inner product is defined by

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \boldsymbol{u} \cdot \boldsymbol{v} = \sum_{i=1}^{n} u_i v_i,$$

and the length or *norm* of $\boldsymbol{u}$ can be defined in terms of the inner product as

$$||\boldsymbol{u}|| = \langle \boldsymbol{u}, \boldsymbol{u} \rangle^{1/2} = \Bigl(\sum_{i=1}^{n} u_i^2\Bigr)^{1/2}.$$

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

$$||u|| = \Big(\int_a^b u(t)^2 dt\Big)^{1/2} = \Big(\int_a^b u^2\Big)^{1/2}.$$

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 $\boldsymbol{\tau}$ be the 1-regular knot vector*

$$\boldsymbol{\tau} = (\tau_i)_{i=1}^{m+2} = (x_1, x_1, x_2, x_3, \ldots, x_{m-1}, x_m, x_m).$$

Then the second derivative of e is orthogonal to the spline space $\mathbb{S}_{1,\boldsymbol{\tau}}$. In other words

$$\int_a^b e''(x)h(x)\,dx = 0, \quad \text{for all} \quad h \in \mathbb{S}_{1,\boldsymbol{\tau}}.$$

Proof. Dividing $[a, b]$ into the subintervals $[x_i, x_{i+1}]$ for $i = 1, \ldots, m-1$, and using integration by parts, we find

$$\int_a^b e''h = \sum_{i=1}^{m-1} \int_{x_i}^{x_{i+1}} e''h = \sum_{i=1}^{m-1} \Big(e'h\Big|_{x_i}^{x_{i+1}} - \int_{x_i}^{x_{i+1}} e'h' \Big).$$

Since $e'(a) = e'(b) = 0$, the first term is zero,

$$\sum_{i=1}^{m-1} e'h\Big|_{x_i}^{x_{i+1}} = e'(b)h(b) - e'(a)h(a) = 0. \tag{5.14}$$

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 minimisation problem. In any minimisation problem, we must specify the space over which we minimise. The space in this case is $\mathbb{E}_H(f)$, which is defined in terms of the related space $\mathbb{E}(f)$

$$\begin{aligned} \mathbb{E}(f) &= \big\{ g \in C^2[a, b] \mid g(x_i) = f(x_i) \text{ for } i = 1, \ldots, m \big\}, \\ \mathbb{E}_H(f) &= \big\{ g \in \mathbb{E}(f) \mid g'(a) = f'(a) \text{ and } g'(b) = f'(b) \big\}. \end{aligned} \tag{5.15}$$

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*

$$\int_a^b \bigl(g''(x)\bigr)^2\,dx \le \int_a^b \bigl(h''(x)\bigr)^2\,dx \quad \textit{for all } h \textit{ in } \mathbb{E}_H(f), \tag{5.16}$$

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

$$\int_a^b h''^2 = \int_a^b \bigl(e'' + g''\bigr)^2 = \int_a^b e''^2 + 2\int_a^b e''g'' + \int_a^b g''^2. \tag{5.17}$$

Since $g \in \mathbb{S}_{3,\boldsymbol{\tau}_H}$ we have $g'' \in \mathbb{S}_{1,\boldsymbol{\tau}}$, where $\boldsymbol{\tau}$ 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 $\int_a^b h''^2 \ge \int_a^b g''^2$.

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 $\mathbb{S}_{3,\boldsymbol{\tau}_H}$ such that

$$\begin{aligned} \sum_{j=1}^{m+2} c_j B_{j,3}(x_i) &= f(x_i), \qquad \text{for } i = 1, \ldots, m,\\ \sum_{j=1}^{m+2} c_j B'_{j,3}(x_i) &= f'(x_i), \qquad \text{for } i = 1 \text{ and } m. \end{aligned} \tag{5.18}$$

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 $g \in \mathbb{S}_{3,\boldsymbol{\tau}_H}$ solves Problem 5.10 H with zero data. Clearly $g = 0$ is a solution. According to Theorem 5.12, any other solution must also minimise 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 $\boldsymbol{\tau}$ the knot vector*

$$\boldsymbol{\tau} = (\tau_i)_{i=1}^m = (x_1, x_2, x_3, \ldots, x_{m-1}, x_m),$$

the second derivative of e is orthogonal to $\mathbb{S}_{1,\boldsymbol{\tau}}$,

$$\int_a^b e''(x)h(x)\,dx = 0, \quad \textit{for all } h \textit{ in } \mathbb{S}_{1,\boldsymbol{\tau}}.$$

Proof. The proof is similar to Lemma 5.11. The relation in (5.14) holds since every $h \in \mathbb{S}_{1,\boldsymbol{\tau}}$ 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*

$$\int_a^b \big(g''(x)\big)^2\,dx \le \int_a^b \big(h''(x)\big)^2\,dx, \quad \textit{for all} \quad h \in \mathbb{E}(f),$$

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 $g'' \in \mathbb{S}_{1,\boldsymbol{\tau}}$. ∎

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 minimised. 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 minimisation 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 minimised 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 minimises 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

$$\kappa(x) = \frac{g''(x)}{\Bigl(1 + \bigl(g'(x)\bigr)^2\Bigr)^{3/2}}.$$

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.

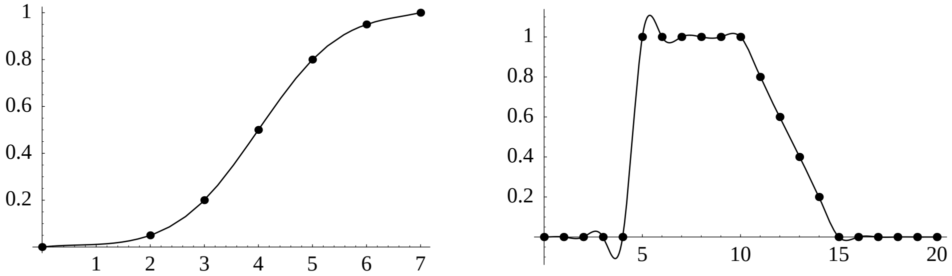
5.2.2 Numerical solution and examples

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

$$\boldsymbol{A}\boldsymbol{c} = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ \beta_2 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{m+1} & \alpha_{m+1} & \gamma_{m+1} \\ & & & \beta_{m+2} & \alpha_{m+2} \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_{m+1} \\ c_{m+2} \end{pmatrix} = \begin{pmatrix} f'(x_1) \\ f(x_1) \\ \vdots \\ f(x_m) \\ f'(x_m) \end{pmatrix} = \boldsymbol{f}, \tag{5.19}$$

where the elements of $\boldsymbol{A}$ are given by

$$\begin{aligned} \alpha_1 &= B'_{1,3}(x_1), \quad \alpha_{m+2} = B'_{m+2,3}(x_m), \\ \gamma_1 &= B'_{2,3}(x_1), \quad \beta_{m+2} = B'_{m+1,3}(x_m), \\ \beta_{i+1} &= B_{i,3}(x_i), \quad \alpha_{i+1} = B_{i+1,3}(x_i), \quad \gamma_{i+1} = B_{i+2,3}(x_i). \end{aligned} \tag{5.20}$$



(a) (b)

Figure 5.3. Cubic spline interpolation to smoothly varying data (a) and data with sharp corners (b).

The elements of $\boldsymbol{A}$ 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 $\boldsymbol{A}$ 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 .

In most cases, the underlying function f is only known through the data $y_i = f(x_i)$, for $i = 1, \ldots, m$. 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

$$d_1 = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \quad \text{and} \quad d_2 = \frac{f(x_m) - f(x_{m-1})}{x_m - x_{m-1}}. \tag{5.21}$$

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 generalised 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 1.6 (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

$$g(x_i) = y_i, \qquad i = 1, \ldots, m.$$

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 $\bigl(x_i, y_i\bigr)_{i=1}^m$ and a spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ whose knot vector $\boldsymbol{\tau} = (\tau_i)_{i=1}^{m+d+1}$ satisfies $\tau_{i+d+1} > \tau_i$, for $i = 1, \ldots, m$. Find a spline g in $\mathbb{S}_{d,\boldsymbol{\tau}}$ such that*

$$g(x_i) = \sum_{j=1}^{m} c_j B_{j,d}(x_i) = y_i, \qquad \text{for } i = 1, \ldots, m. \tag{5.22}$$

The equations in (5.22) form a system of m equations in m unknowns. In matrix form these equations can be written

$$\boldsymbol{A}\boldsymbol{c} = \begin{pmatrix} B_{1,d}(x_1) & \ldots & B_{m,d}(x_1) \\ \vdots & \ddots & \vdots \\ B_{1,d}(x_m) & \ldots & B_{m,d}(x_m) \end{pmatrix} \begin{pmatrix} c_1 \\ \vdots \\ c_m \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_m \end{pmatrix} = \boldsymbol{y}. \tag{5.23}$$

Theorem 5.18 gives necessary and sufficient conditions for this system to have a unique solution, in other words for $\boldsymbol{A}$ to be nonsingular.

Theorem 5.18. *The matrix $\boldsymbol{A}$ in (5.23) is nonsingular if and only if the diagonal elements $a_{i,i} = B_{i,d}(x_i)$ are positive for $i = 1, \ldots m$.*

Proof. See Theorem 10.6 in Chapter 10. ∎

The condition that the diagonal elements of $\boldsymbol{A}$ should be nonzero can be written

$$\tau_i < x_{i+1} < \tau_{i+d+1}, \quad i = 1, 2, \ldots, m, \tag{5.24}$$

provided we allow $x_i = \tau_i$ if $\tau_i = \cdots = \tau_{i+d}$. 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 $\boldsymbol{\tau} = \boldsymbol{\tau}_F$ is given by

$$\boldsymbol{\tau} = (\tau_i)_{i=1}^{m+4} = (x_1, x_1, x_1, x_1, x_3, x_4, \ldots, x_{m-2}, x_m, x_m, x_m, x_m).$$

From this we note that $B_1(x_1)$ and $B_2(x_2)$ are both positive. Since $\tau_{i+2} = x_i$ for $i = 3$, $\ldots$, $m-2$, 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 $\boldsymbol{A}$ in (5.22), at most $d+1$ entries are nonzero, and those entries are consecutive. This gives $\boldsymbol{A}$ 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 minimised. 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 $x_1 < \cdots < x_m$, positive real numbers w_i for $i = 1$, $\ldots$, m, and an n-dimensional spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$, find a spline g in $\mathbb{S}_{d,\boldsymbol{\tau}}$ which solves the minimization problem*

$$\min_{h \in \mathbb{S}_{d,\boldsymbol{\tau}}} \sum_{i=1}^{m} w_i \left(y_i - h(x_i)\right)^2. \tag{5.25}$$

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\boldsymbol{c} = (c_1, \ldots, c_n)$ 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*

$$\min_{\boldsymbol{c}\in\mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{c} - \boldsymbol{b}\|^2,$$

where $\boldsymbol{A} \in \mathbb{R}^{m,n}$ and $\boldsymbol{b} \in \mathbb{R}^m$ have components

$$a_{i,j} = \sqrt{w_i} B_j(x_i) \qquad \text{and} \qquad b_i = \sqrt{w_i} y_i, \tag{5.26}$$

and for any $\boldsymbol{u} = (u_1, \ldots, u_m)$,

$$\|\boldsymbol{u}\| = \sqrt{u_1^2 + \cdots + u_m^2},$$

is the usual Euclidean length of a vector in $\mathbb{R}^m$.

Proof. Suppose $\boldsymbol{c} = (c_1, \ldots, c_n)$ are the B-spline coefficients of some $h \in \mathbb{S}_{d,\boldsymbol{\tau}}$. Then

$$\begin{aligned}\|\boldsymbol{Ac} - \boldsymbol{b}\|_2^2 &= \sum_{i=1}^{m}\Bigl(\sum_{j=1}^{n} a_{i,j}c_j - b_i\Bigr)^2 \\ &= \sum_{i=1}^{m}\Bigl(\sum_{j=1}^{n} \sqrt{w_i}B_j(x_i)c_j - \sqrt{w_i}y_i\Bigr)^2 \\ &= \sum_{i=1}^{m} w_i\Bigl(h(x_i) - y_i\Bigr)^2.\end{aligned}$$

This shows that the two minimisation problems are equivalent. ∎

In the next lemma, we collect some facts about the general linear least squares problem. Recall that a symmetric matrix $\boldsymbol{N}$ is positive semidefinite if $\boldsymbol{c}^T\boldsymbol{Nc} \geq 0$ for all $\boldsymbol{c} \in \mathbb{R}^n$, and positive definite if in addition $\boldsymbol{c}^T\boldsymbol{Nc} > 0$ for all nonzero $\boldsymbol{c} \in \mathbb{R}^n$.

Lemma 5.22. *Suppose m and n are positive integers with $m \geq n$, and let the matrix $\boldsymbol{A}$ in $\mathbb{R}^{m,n}$ and the vector $\boldsymbol{b}$ in $\mathbb{R}^m$ be given. The linear least squares problem*

$$\min_{\boldsymbol{c}\in\mathbb{R}^n} \|\boldsymbol{Ac} - \boldsymbol{b}\|^2 \tag{5.27}$$

always has a solution $\boldsymbol{c}^*$ which can be found by solving the linear set of equations

$$\boldsymbol{A}^T\boldsymbol{Ac}^* = \boldsymbol{A}^T\boldsymbol{b}. \tag{5.28}$$

The coefficient matrix $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ is symmetric and positive semidefinite. It is positive definite, and therefore nonsingular, and the solution of (5.27) is unique if and only if $\boldsymbol{A}$ has linearly independent columns.

Proof. Let span($\boldsymbol{A}$) denote the n-dimensional linear subspace of $\mathbb{R}^m$ spanned by the columns of $\boldsymbol{A}$,

$$\operatorname{span}(\boldsymbol{A}) = \{\boldsymbol{Ac} \mid \boldsymbol{c} \in \mathbb{R}^n\}.$$

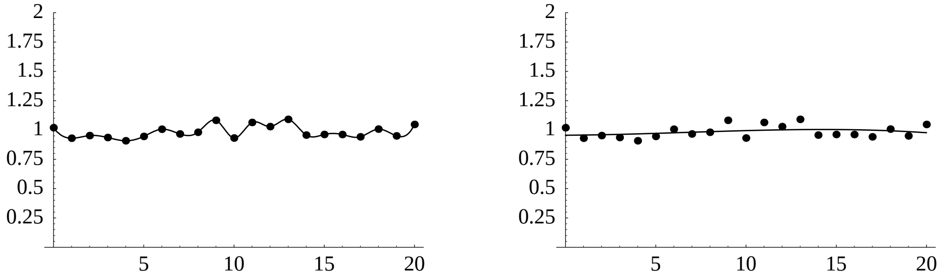
From basic linear algebra we know that a vector $\boldsymbol{b} \in \mathbb{R}^m$ can be written uniquely as a sum $\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2$, where $\boldsymbol{b}_1$ is a linear combination of the columns of $\boldsymbol{A}$ so that $\boldsymbol{b}_1 \in \operatorname{span}(\boldsymbol{A})$, and $\boldsymbol{b}_2$ is orthogonal to span($\boldsymbol{A}$), i.e., we have $\boldsymbol{b}_2^T\boldsymbol{d} = 0$ for all $\boldsymbol{d}$ in span($\boldsymbol{A}$). Using this decomposition of $\boldsymbol{b}$, and the Pythagorean theorem, we have for any $\boldsymbol{c} \in \mathbb{R}^n$,

$$\|\boldsymbol{Ac} - \boldsymbol{b}\|^2 = \|\boldsymbol{Ac} - \boldsymbol{b}_1 - \boldsymbol{b}_2\|^2 = \|\boldsymbol{Ac} - \boldsymbol{b}_1\|^2 + \|\boldsymbol{b}_2\|^2.$$

It follows that $\|\boldsymbol{Ac} - \boldsymbol{b}\|_2^2 \geq \|\boldsymbol{b}_2\|_2^2$ for any $\boldsymbol{c} \in \mathbb{R}^n$, with equality if $\boldsymbol{Ac} = \boldsymbol{b}_1$. A $\boldsymbol{c} = \boldsymbol{c}^*$ such that $\boldsymbol{Ac}^* = \boldsymbol{b}_1$ clearly exists since $\boldsymbol{b}_1$ is in span($\boldsymbol{A}$), and $\boldsymbol{c}^*$ is unique if and only if $\boldsymbol{A}$ has linearly independent columns. To derive the linear system for $\boldsymbol{c}^*$, we note that any $\boldsymbol{c}$ that is minimising satisfies $\boldsymbol{Ac} - \boldsymbol{b} = -\boldsymbol{b}_2$. Since we also know that $\boldsymbol{b}_2$ is orthogonal to span($\boldsymbol{A}$), we must have

$$\boldsymbol{d}^T(\boldsymbol{Ac} - \boldsymbol{b}) = \boldsymbol{c}_1^T\boldsymbol{A}^T(\boldsymbol{Ac} - \boldsymbol{b}) = 0$$

for all $\boldsymbol{d} = \boldsymbol{Ac}_1$ in span($\boldsymbol{A}$), i.e., for all $\boldsymbol{c}_1$ in $\mathbb{R}^n$. But this is only possible if $\boldsymbol{A}^T(\boldsymbol{Ac}-\boldsymbol{b}) = 0$. This proves (5.28).



(a)

(b)

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 $\boldsymbol{N} = \boldsymbol{A}^T\boldsymbol{A}$ is clearly symmetric and

$$\boldsymbol{c}^T\boldsymbol{N}\boldsymbol{c} = \|\boldsymbol{A}\boldsymbol{c}\|_2^2 \geq 0, \tag{5.29}$$

for all $\boldsymbol{c} \in \mathbb{R}^n$, so that $\boldsymbol{N}$ is positive semi-definite. From (5.29) we see that we can find a nonzero $\boldsymbol{c}$ such that $\boldsymbol{c}^T\boldsymbol{N}\boldsymbol{c} = 0$ if and only if $\boldsymbol{A}\boldsymbol{c} = 0$, i.e., if and only if $\boldsymbol{A}$ has linearly dependent columns . We conclude that $\boldsymbol{N}$ is positive definite if and only if $\boldsymbol{A}$ 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 abscissa such that*

$$B_\ell(x_{i_\ell}) \neq 0 \qquad \textit{for } \ell = 1, \ldots, n.$$

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 $\boldsymbol{A}$ in Lemma 5.21 has linearly independent columns. Now $\boldsymbol{A}$ has linearly independent columns if and only if we can find a subset of n rows of $\boldsymbol{A}$ such that the square submatrix consisting of these rows and all columns of $\boldsymbol{A}$ 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 factorisation of a banded matrix followed by back substitution. The least squares problem can also be solved by computing a QR-factorisation of the matrix $\boldsymbol{A}$; 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 generalisation 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 $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_{n+d+1})$ be a $d+1$-regular knot vector with boundary knots $\tau_{d+1} = a$ and $\tau_{n+1} = b$. The spline given by*

$$(Vf)(x) = \sum_{j=1}^{n} f(\tau_j^*) B_{j,d}(x) \tag{5.30}$$

where $\tau_j^* = (\tau_{j+1} + \cdots + \tau_{j+d})/d$ are the knot averages, is called the Variation Diminishing Spline Approximation *of degree d to f on the knot vector $\boldsymbol{\tau}$.*

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 $\boldsymbol{\tau}$, then

$$a = \tau_1^* < \tau_2^* < \ldots < \tau_{n-1}^* < \tau_n^* = b. \tag{5.31}$$

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 $\boldsymbol{\tau}$ are uniform in the interval $[0, 1]$, say

$$\boldsymbol{\tau} = (0, 0, 0, 0, 1/m, 2/m, \ldots, 1 - 1/m, 1, 1, 1, 1). \tag{5.32}$$

For $m \geq 2$ we then have

$$\boldsymbol{\tau}^* = (0, 1/(3m), 1/m, 2/m, \ldots, 1 - 1/m, 1 - 1/(3m), 1). \tag{5.33}$$

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).

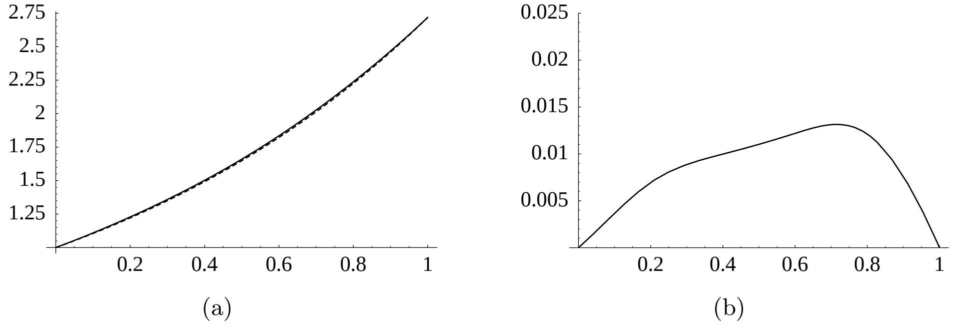


Figure 5.5. The exponential function together with the cubic variation diminishing approximation of Example 5.26 in the special case $m = 5$ is shown in (a). The error in the approximation is shown in (b).

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

$$f(x) = 1 - e^{-50|x|}, \qquad x \in [-1, 1], \tag{5.34}$$

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

$$\boldsymbol{\tau} = (-1, -1, -1, -1, -1 + 1/m, \ldots, -1/m, 0, 0, 0, 1/m, \ldots, 1 - 1/m, 1, 1, 1, 1). \tag{5.35}$$

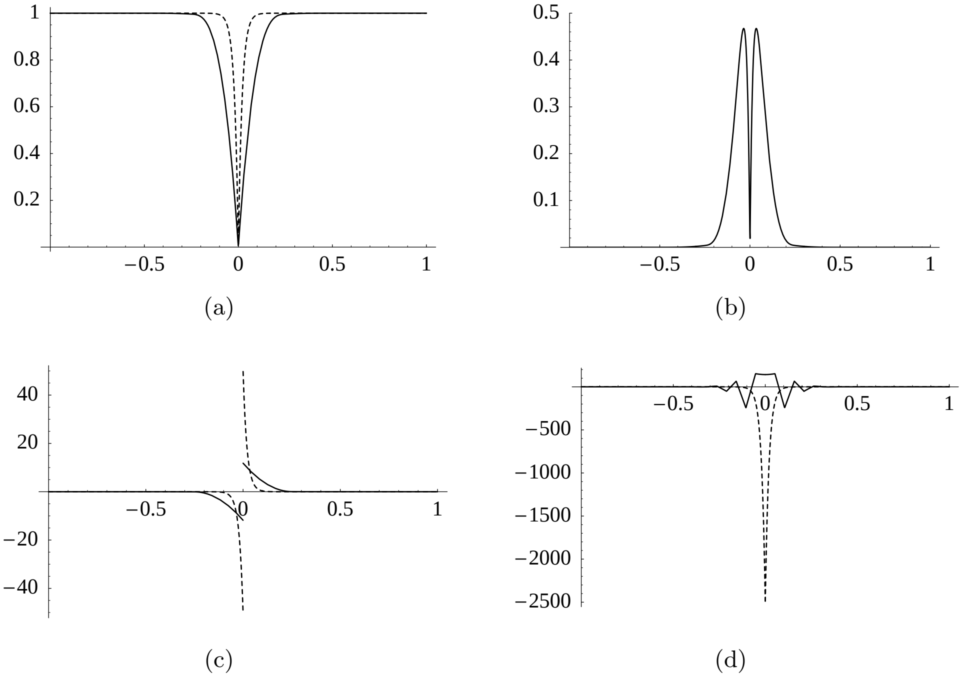
The integer m is a parameter which is used to control the number of knots and thereby the accuracy of the approximation. The spline Vf is shown in Figure 5.6 (a) for $m = 4$ together with the function f itself. The error is shown in Figure 5.6 (b), and we note that the error is zero at $x = 0$, but quite 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 nonnegative 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 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.



(a)

(b)

(c)

(d)

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).

Lemma 5.28. *Let g be a spline in some spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$. Then g is bounded by its smallest and largest B-spline coefficients,*

$$\min_i \{c_i\} \leq \sum_i c_i B_i(x) \leq \max_i \{c_i\}, \qquad \textit{for all } x \in \mathbb{R}. \tag{5.36}$$

Proof. Let $c_{\max}$ be the largest coefficient. Then we have

$$\sum_i c_i B_i(x) \leq \sum_i c_{\max} B_i(x) = c_{\max} \sum_i B_i(x) \leq c_{\max},$$

since $\sum_i B_i(x) \leq 1$. This proves the second inequality in (5.36). The proof of the first inequality is similar. ∎

Any plot of a spline with its control polygon will confirm the inequalities (5.36), see for example Figure 5.7.

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*

$$f_{\min} \leq f(x) \leq f_{\max} \qquad \textit{for all } x \in \mathbb{R}.$$

Then the variation diminishing spline approximation to f from some spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ has the same bounds,

$$f_{\min} \leq (Vf)(x) \leq f_{\max} \qquad \textit{for all } x \in \mathbb{R}. \tag{5.37}$$

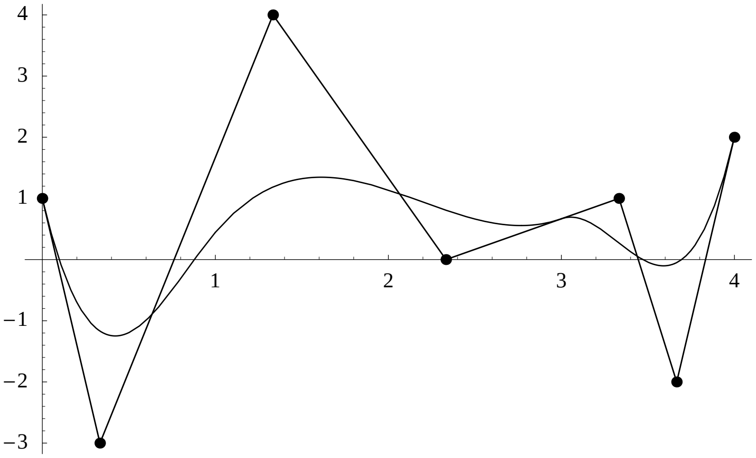


Figure 5.7. A cubic spline with its control polygon. Note how the extrema of the control polygon bound the extrema of the spline.

Proof. Recall that the B-spline coefficients c_i of Vf are given by

$$c_i = f(\tau_i^*) \qquad \text{where } \tau_i^* = (\tau_{i+1} + \cdots + \tau_{i+d})/d.$$

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 characterised 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 nonnegative.*

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 $\boldsymbol{\tau}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$. If the coefficients are increasing, that is $c_i \leq c_{i+1}$ for $i = 1, \ldots, n-1$, then g is increasing. If the coefficients are decreasing then g is decreasing.*

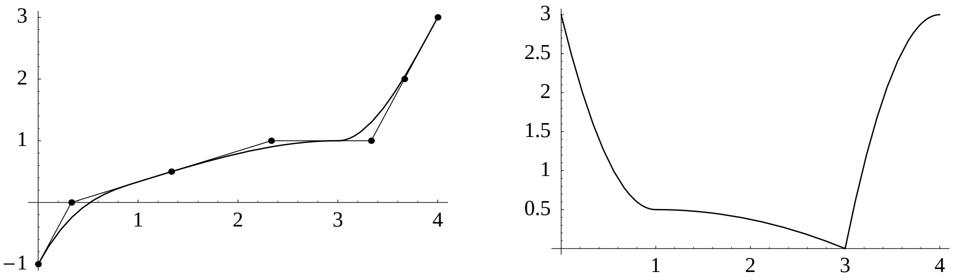
Proof. The proposition is certainly true for $d = 0$, so we can assume that $d \geq 1$. Suppose first that there are no interior knots in $\boldsymbol{\tau}$ of multiplicity $d+1$. If we differentiate g we find $g'(x) = \sum_{i=1}^{n} \Delta c_i B_{i,d-1}(x)$ for $x \in [a, b]$, where

$$\Delta c_i = d \frac{c_i - c_{i-1}}{\tau_{i+d} - \tau_i}.$$

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.



(a)

(b)

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.

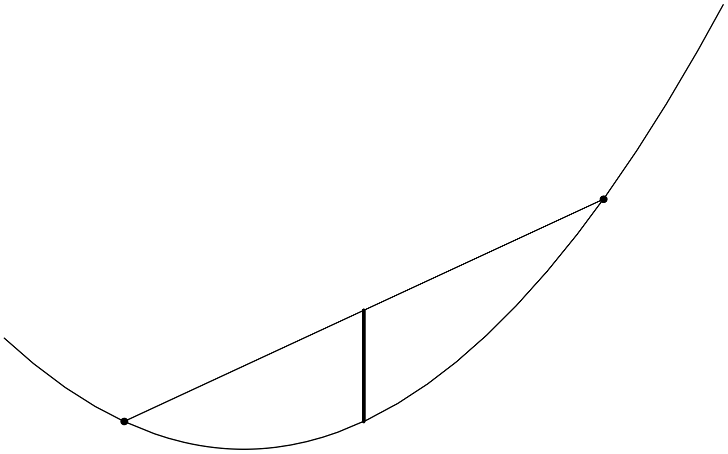


Figure 5.9. A convex function and the cord connecting two points on the graph.

Proposition 5.33. *Let f be function defined on the interval $[a, b]$ and let $\boldsymbol{\tau}$ be a $d+1$-extended knot vector with $\tau_{d+1} = a$ and $\tau_{n+1} = b$. 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.

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 downwardsupward, 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*

$$f\big((1-\lambda)x_0 + \lambda x_2\big) \le (1-\lambda)f(x_0) + \lambda f(x_2) \tag{5.38}$$

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 characterised 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*

$$\frac{f(x_1)-f(x_0)}{x_1-x_0} \leq \frac{f(x_2)-f(x_1)}{x_2-x_1} \tag{5.39}$$

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

$$(1-\lambda)\bigl(f(x_1)-f(x_0)\bigr) \leq \lambda\bigl(f(x_2)-f(x_1)\bigr).$$

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

$$\frac{f(x_1)-f(x_0)}{x_1-x_0} = f'(\theta_0) \qquad \text{and} \qquad \frac{f(x_2)-f(x_1)}{x_2-x_1} = f'(\theta_1),$$

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

$$\frac{f(c)-f(a)}{c-a} \leq \frac{f(y)-f(x)}{y-x} \leq \frac{f(b)-f(d)}{b-d}. \tag{5.40}$$

Set $M = \max\{(f(c) - f(a))/(c - a), (f(b) - f(d))/(b - d)\}$. Then (5.40) is equivalent to

$$|f(y) - f(x)| \le M|y - x|.$$

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 $\boldsymbol{\tau}$ be a $d + 1$-extended knot vector for some $d \ge 1$, and let $g = \sum_{i=1}^{n} c_i B_{i,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$. 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 $i = 2, \ldots, n$. Then g is convex on $[\tau_{d+1}, \tau_{n+1}]$ if it is continuous and

$$\Delta c_{i-1} \le \Delta c_i \qquad \text{for } i = 2, \ldots, n. \tag{5.41}$$

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

$$\delta(x_0, x_1) \le \delta(x_1, x_2). \tag{5.42}$$

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 $x_0 < z \le x_1 < x_2$. 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 $\delta(x_0, z) \le \delta(z, x_2)$, 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

$$\delta(x_0, x_1) \le \delta(z, x_1) \le \delta(x_1, x_2),$$

the first inequality being valid since $\delta(x_0, z) \le \delta(z, x_1)$ 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. ∎

An example of a convex spline is shown in Figure 5.10, together with its first and second derivatives in.

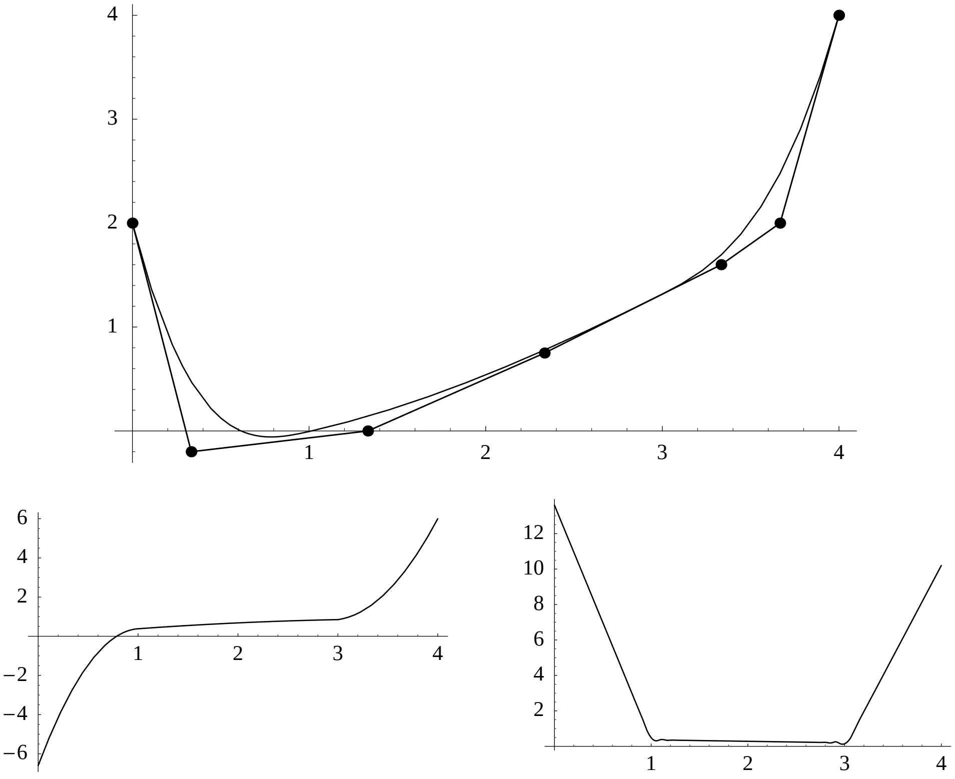


Figure 5.10. A convex spline with its control polygon (a), its first derivative (b) and its second derivative (c).

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 $d \ge 1$ be an integer, and let $\boldsymbol{\tau}$ 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 $\big(f(\tau_i^*)\big)_{i=1}^n$ 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

CHAPTER 6

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 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.

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 $\boldsymbol{f} : [a,b] \mapsto \mathbb{R}^s$ 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 $\boldsymbol{f}$ has continuous derivatives up to order m. If, in addition, the first derivative of $\boldsymbol{f}$ does not vanish in $[a,b]$,*

$$D\boldsymbol{f}(t) = \boldsymbol{f}'(t) \neq 0, \qquad \text{for } t \in [a,b],$$

then $\boldsymbol{f}$ 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 $\boldsymbol{f}(u) = \big(u, h(u)\big)$.

If we imagine travelling along the curve and let u denote the elapsed time of our journey, then the length of $\boldsymbol{f}'(u)$ which we denote by $||\boldsymbol{f}'(u)||$, gives the velocity with which we travel at time u, while the direction of $\boldsymbol{f}'(u)$ 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

$$\boldsymbol{f}(u) = (1-u)\boldsymbol{p}_0 + u\boldsymbol{p}_1, \quad \text{for } u \in [0,1],$$

where $\boldsymbol{p}_0$ and $\boldsymbol{p}_1$ are points in the plane, is a simple example of a parametric representation. Since $\boldsymbol{f}'(u) = \boldsymbol{p}_1 - \boldsymbol{p}_0$ for all u, we have in fact that $\boldsymbol{f}$ is a regular parametric representation, provided that $\boldsymbol{p}_0 \neq \boldsymbol{p}_1$. The tangent vector is, as expected, parallell 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

$$\boldsymbol{f}(u) = \bigl(x(u), y(u)\bigr) = (\cos u, \sin u)$$

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 $||\boldsymbol{f}'(u)|| = 1$ for all u, so that $\boldsymbol{f}$ is a regular parametric representation.

One may wonder what the significance of the regularity condition $\boldsymbol{f}'(u) \neq 0$ is. Let us consider the parametric representation given by

$$\boldsymbol{f}(u) = \begin{cases} (0, u^2), & \text{for } u < 0; \\ (u^2, 0), & \text{for } u \geq 0; \end{cases}$$

in other words, for $u < 0$ the image of $\boldsymbol{f}$ is the positive y-axis and for $u > 0$, the image is the positive x-axis. A plot of $\boldsymbol{f}$ for $u \in [-1, 1]$ is shown in Figure 6.1 (a). The geometric figure traced out by $\boldsymbol{f}$ clearly has a right angle corner at the origin, but $\boldsymbol{f}'$ which is given by

$$\boldsymbol{f}'(u) = \begin{cases} (0, 2u), & \text{for } u < 0; \\ (2u, 0), & \text{for } u > 0; \end{cases}$$

is still continuous for all u. The source of the problem is the fact that $\boldsymbol{f}'(0) = 0$. For this means that as we travel along the curve, the velocity 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

$$\boldsymbol{\theta}(u) = \boldsymbol{f}'(u)/||\boldsymbol{f}'(u)||,$$

we see that

$$\boldsymbol{\theta}(u) = \begin{cases} (0, -1), & \text{for } u < 0; \\ (1, 0), & \text{for } u > 0. \end{cases}$$

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 $\boldsymbol{f}(u) = (u^2, u^3)$ which clearly has a continuous tangent, but again we have $\boldsymbol{f}'(0) = (0, 0)$ 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.

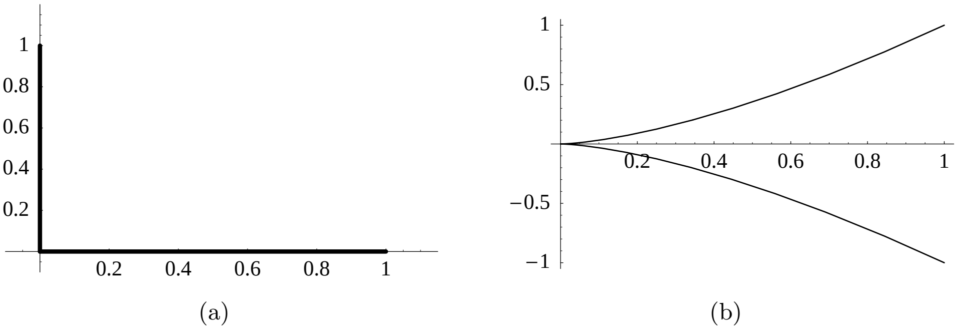


Figure 6.1. A parametric representation with continuous first derivative but discontinuous unit tangent (a), and the parametric representation $\boldsymbol{f}(u) = (u^2, u^3)$ (b).

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. Consider for example again the unit circle $\boldsymbol{f}(u) = (\cos u, \sin u)$. If we substitute $u = 2\pi v$, we obtain the parametric representation

$$\hat{\boldsymbol{r}}(v) = (\cos 2\pi v, \sin 2\pi v).$$

As v varies in the interval $[0, 1]$, the original parameter u will vary in the interval $[0, 2\pi]$ so that $\hat{\boldsymbol{r}}(v)$ will trace out the same set of points in $\mathbb{R}^2$ and therefore yield the same graph as $\boldsymbol{f}(u)$. 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. If $\boldsymbol{f}(u)$ is a parametric representation, then $\boldsymbol{g}(v) = \boldsymbol{f}\big(u(v)\big)$ is called a parametrisation of $\boldsymbol{f}$.*

From the chain rule we see that

$$\boldsymbol{g}'(v) = u'(v)\boldsymbol{f}'\big(u(v)\big).$$

From this it is clear that even if $\boldsymbol{f}$ is a regular parametric representation, we can still have $\boldsymbol{g}'(v) = 0$ for some v if $u'(v)$ becomes zero for some v. 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 $\boldsymbol{f}$ and $\boldsymbol{g}$, the two representations have the same orientation. If $u'(v) < 0$ for all v, then $\boldsymbol{f}$ and $\boldsymbol{g}$ 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.

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 (in this situation we will often say that one representation is a reparametrization 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 representation $\boldsymbol{f}$ 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 velocity 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 now 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 velocity, 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 give 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 velocity. 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 would equal 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 $\boldsymbol{g}(\sigma)$ in $\mathbb{R}^s$ is said to be parametrised by arc length if $||\boldsymbol{g}'(\sigma)|| = 1$ for all σ.*

Let $\boldsymbol{f}(u)$ be a given regular curve with $u \in [a,b]$, and let $\boldsymbol{g}(\sigma) = \boldsymbol{f}(u(\sigma))$ be a reparametrisation such that $||\boldsymbol{g}'(\sigma)|| = 1$ for all σ. Since $\boldsymbol{g}'(\sigma) = u'(\sigma)\boldsymbol{f}'(u(\sigma))$, we see that we must have $|u'(\sigma)| = 1/||\boldsymbol{f}'(u(\sigma))||$ or $|\sigma'(u)| = ||\boldsymbol{f}'(u)||$ (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

$$\sigma(u) = \int_a^u ||\boldsymbol{f}'(v)||\,dv. \tag{6.1}$$

We sum this up in a proposition.

Proposition 6.5. *Let $\boldsymbol{f}(u)$ be a given regular parametric curve. The change of parameter given by (6.1) reparametrises the curve by arc length, so that if $\boldsymbol{g}(\sigma) = \boldsymbol{f}\big(u(\sigma)\big)$ then $||\boldsymbol{g}'(\sigma)|| = 1$.*

Note that $\sigma(u)$ as given by (6.1) gives the length of the curve from the starting point $\boldsymbol{f}(a)$ to the point $\boldsymbol{f}(u)$. This can be seen by sampling $\boldsymbol{f}$ 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 $\boldsymbol{f}$ defined on an interval $[a, b]$ is given by*

$$L(\boldsymbol{f}) = \int_a^b \big\| \boldsymbol{f}'(u) \big\| \, du$$

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 $\boldsymbol{r}(u) = (\cos u, \sin u)$ 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 in for spline functions can be generalised to curves.

6.2.1 Definition of parametric spline curves

A spline curve $\boldsymbol{f}$ 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 $\boldsymbol{c} = (\boldsymbol{c}_i)_{i=1}^n$ is given by*

$$\boldsymbol{g}(u) = \sum_{i=1}^{n} \boldsymbol{c}_i B_{i,d,\boldsymbol{\tau}}(u),$$

where each $\boldsymbol{c}_i = (c_i^1, c_i^2, \ldots, c_i^s)$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}^s$.

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

$$\begin{aligned} \boldsymbol{g}(u) &= \sum_i \boldsymbol{c}_i B_i(u) = \sum_i (c_i^1, \ldots, c_i^s) B_i(u) \\ &= \Big(\sum_i c_i^1 B_i(u), \ldots, \sum_i c_i^s B_i(u) \Big) \\ &= \big(g^1(u), \ldots, g^s(u) \big), \end{aligned} \tag{6.2}$$

so that $\boldsymbol{g}$ 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.

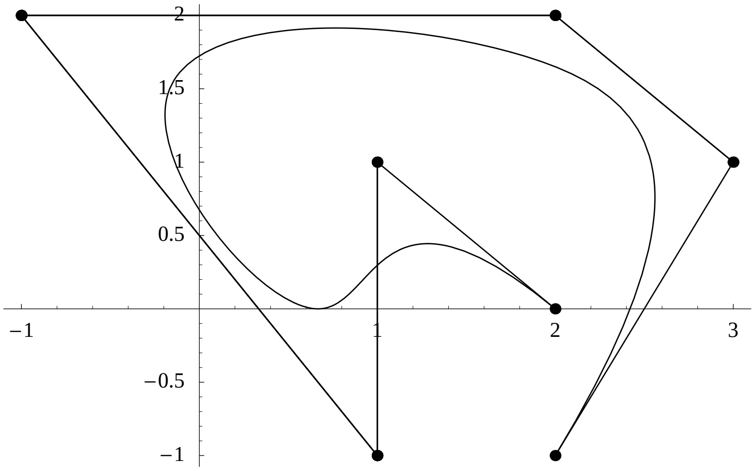


Figure 6.2. A cubic parametric spline curve with its control polygon.

Since a spline curve is nothing but a vector of spline functions as in (6.2), it is simple to compute $\boldsymbol{f}(u)$. Just apply a routine like Algorithm 2.20 to each of the component spline functions g^1, …, g^s. If the algorithm has been implemented in a language that supports vector arithmetic, then evaluation is even simpler. Just apply Algorithm 2.20 directly to $\boldsymbol{g}$, with vector coefficients. The result will be the vector $\boldsymbol{g}(u) = \big(g^1(u), \dots, g^s(u)\big)$.

Example 6.8. As an example of a spline curve, suppose that we are given n points $\boldsymbol{p} = (\boldsymbol{p}_i)_{i=1}^n$ in the plane with $\boldsymbol{p}_i = (x_i, y_i)$, 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

$$\boldsymbol{g}(u) = \sum_{i=1}^n \boldsymbol{p}_i B_{i,1,\boldsymbol{\tau}}(u) = \Big(\sum_{i=1}^n x_i B_{i,1,\boldsymbol{\tau}}(u), \sum_{i=1}^n y_i B_{i,1,\boldsymbol{\tau}}(u)\Big)$$

is a representation of the piecewise linear interpolant to the points $\boldsymbol{p}$.

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 $\boldsymbol{g} = \sum_{i=1}^n \boldsymbol{c}_i B_{i,d,\boldsymbol{\tau}}$ defined on a $d+1$-extended knot vector $\boldsymbol{\tau}$ is a subset of the convex hull of its coefficients,*

$$\boldsymbol{g}(u) \in \mathbb{CH}(\boldsymbol{c}_1, \dots, \boldsymbol{c}_n), \qquad \text{for any } u \in [\tau_{d+1}, \tau_{n+1}].$$

If u is restricted to the interval $[\tau_\mu, \tau_{\mu+1}]$ then

$$\boldsymbol{g}(u) \in \mathbb{CH}(\boldsymbol{c}_{\mu-d}, \dots, \boldsymbol{c}_\mu).$$

To create a spline curve, we only have to be able to create spline functions, since a spline curve is only 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 $\boldsymbol{f}$ 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 $V\boldsymbol{f}$ is defined by*

$$(V\boldsymbol{f})(u)=\sum_{i=1}^{n}\boldsymbol{f}(\tau_i^*)B_{i,d,\boldsymbol{\tau}}(u),$$

where $\tau_i^*=(\tau_{i+1}+\cdots\tau_{i+d})/d$.

Note that the definition of $V\boldsymbol{f}$ means that

$$V\boldsymbol{f}=(Vf^1,\dots,Vf^s).$$

If we have implemented a routine for determining the variation diminishing approximation to a scalar function ($s=1$), we can therefore determine $V\boldsymbol{f}$ 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 half circle is shown in Figure 6.3.

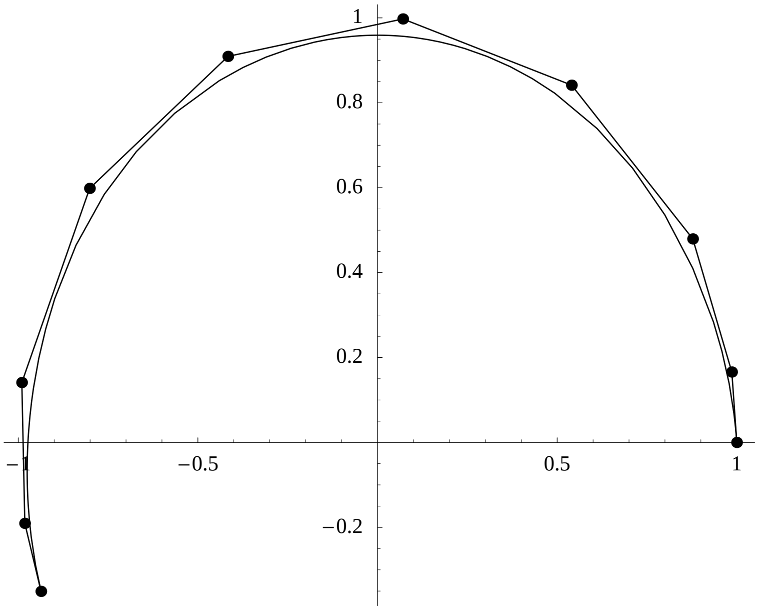


Figure 6.3. A cubic variation diminishing approximation to part of a circle.

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.

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 $\big(u_i, \boldsymbol{f}(u_i)\big)_{i=1}^m$ be given data sampled from the curve $\boldsymbol{f}$ in $\mathbb{R}^s$, and form the knot vector*

$$\boldsymbol{\tau} = (u_1, u_1, u_1, u_1, u_2, \ldots, u_{m-1}, u_m, u_m, u_m, u_m).$$

Then there is a unique spline curve $\boldsymbol{g} = I_N \boldsymbol{f}$ in $\mathbb{S}^s_{3,\boldsymbol{\tau}}$ that satisfies

$$\boldsymbol{g}(u_i) = \boldsymbol{f}(u_i), \qquad \text{for } i = 1, \ldots, m, \tag{6.3}$$

with the natural end conditions $\boldsymbol{g}''(u_1) = \boldsymbol{g}''(u_m) = \boldsymbol{0}$, and this spline curve $\boldsymbol{g}$ uniquely minimises

$$\Big\| \int_{u_1}^{u_m} \boldsymbol{h}''(u)\, du \Big\|$$

when $\boldsymbol{h}$ varies over the class of C^2 parametric representations that satisfies 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

$$I_N \boldsymbol{f} = (I_N f^1, \ldots, I_N f^s).$$

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 $I_N \boldsymbol{f}$, we see that it is only the right hand side that differs; the coefficient matrix $\boldsymbol{A}$ remains the same. This is known to be of great advantage 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 $(\boldsymbol{p}_i)_{i=1}^m$ be a set of points in $\mathbb{R}^s$. Find a cubic spline $\boldsymbol{g}$ in some spline space $\mathbb{S}_{3,\boldsymbol{\tau}}$ such that*

$$\boldsymbol{g}(u_i) = \boldsymbol{p}_i, \qquad \text{for } i = 1, \ldots, m,$$

for some parameter values $(u_i)_{i=1}^m$ with $u_1 < u_2 < \cdots < u_m$.

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. An example may be illuminating.

Example 6.13. Suppose that m points in the plane $\boldsymbol{p} = (\boldsymbol{p}_i)_{i=1}^m$ with $\boldsymbol{p}_i = (x_i, y_i)$ are given. We seek a cubic spline curve that interpolates the points $\boldsymbol{p}$. We can proceed as follows. Associate with each data point $\boldsymbol{p}_i$ 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 $(i, x_i)_{i=1}^n$ and $(i, y_i)_{i=1}^n$. 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 $\boldsymbol{c}^1 = (c_i^1)_{i=1}^{m+2}$ 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 $\boldsymbol{c}^2 = (c_i^2)_{i=1}^{m+2}$ 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

$$\boldsymbol{g}(u) = \sum_{i=1}^{m+2} \boldsymbol{c}_i B_{i,3,\boldsymbol{\tau}}(u)$$

that satisfies $\boldsymbol{g}(i) = \boldsymbol{p}_i$ for $i = 1, 2, \dots, m$.

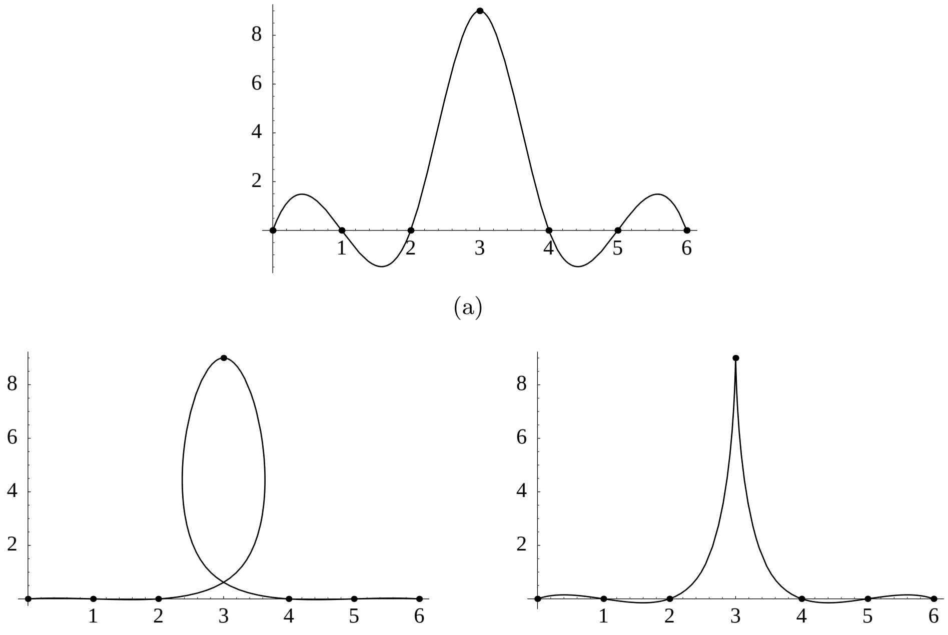
The only part of the construction of the cubic interpolant in Example 6.13 that is different from the corresponding construction for spline functions is the assignment of the parameter value i to the point $\boldsymbol{f}_i = (x_i, y_i)$ for $i = 1, 2, \dots, n$, and therefore also the construction of 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 $\boldsymbol{p}_i$) 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 be 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 $(\boldsymbol{p}_i)_{i=1}^m$ 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.



(a)

(b)

(c)

Figure 6.4. Parametric, cubic spline interpolation with uniform parametrisation (a), cord length parametrisation (b), and centripetal parametrisation (c).

1. **Uniform parametrisation** which amounts to $u_i = i$ for $i = 1, 2, \ldots, m$. This has the shortcomings discussed above.

2. **Cord length parametrisation** which is given by

$$u_1 = 0 \qquad \text{and} \qquad u_i = u_{i-1} + ||\boldsymbol{p}_i - \boldsymbol{p}_{i-1}|| \quad \text{for } i = 2, 3, \ldots, m.$$

   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

$$u_1 = 0 \qquad \text{and} \qquad u_i = u_{i-1} + ||\boldsymbol{p}_i - \boldsymbol{p}_{i-1}||^{1/2} \quad \text{for } i = 2, 3, \ldots, m.$$

   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.

Figures 6.4 (a)–(c) show examples of how the three methods of parametrisation described above perform on a difficult set of data.

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 $(\boldsymbol{p}_i)_{i=1}^m$, 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 $\boldsymbol{f}$, so that $\boldsymbol{f}_i = (u_i, \boldsymbol{p}_i)$ for $i = 1, \ldots, m$, the parametric spline approximation satisfies

$$A\boldsymbol{f} = (Af^1, \ldots, Af^s), \tag{6.4}$$

where f^j denotes the data set $(u_i, p_i^j)_{i=1}^m$ which we think of as $\bigl(u_i, f^j(u_i)\bigr)_{i=1}^m$. 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.

CHAPTER 7

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 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 visualised 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 generalise very easily as we shall see below. The methods also generalise 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 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

$$f(x, y) = c_{0,0}(1 - x)(1 - y) + c_{0,1}(1 - x)y + c_{1,0}x(1 - y) + c_{1,1}xy,$$

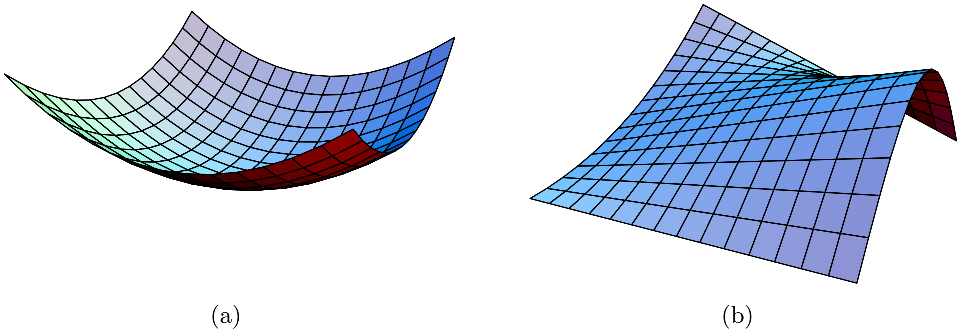


Figure 7.1. A piece of the circular paraboloid $z = x^2 + y^2$ is shown in (a), while the surface $(1 - x)y^2 + x\sin(\pi y)$ is shown in (b).

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 $\boldsymbol{\sigma}$ we have the spline space

$$\mathbb{S}_1 = \mathbb{S}_{d,\boldsymbol{\sigma}} = \operatorname{span}\{\phi_1, \ldots, \phi_{n_1}\}.$$

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,

$$f(x, y) = \sum_{i=1}^{n_1} c_i(y)\phi_i(x). \qquad (7.1)$$

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 $\boldsymbol{\tau}$,

$$\mathbb{S}_2 = \mathbb{S}_{d_2,\boldsymbol{\tau}_2} = \operatorname{span}\{\psi_1, \ldots, \psi_{n_2}\}$$

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

$$c_i(y) = \sum_{j=1}^{n_2} c_{i,j}\psi_j(y) \qquad (7.2)$$

for suitable numbers $(c_{i,j})_{i,j=1}^{n_1,n_2}$. Combining (7.1) and (7.2) we obtain

$$f(x, y) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} c_{i,j}\phi_i(x)\psi_j(y). \qquad (7.3)$$

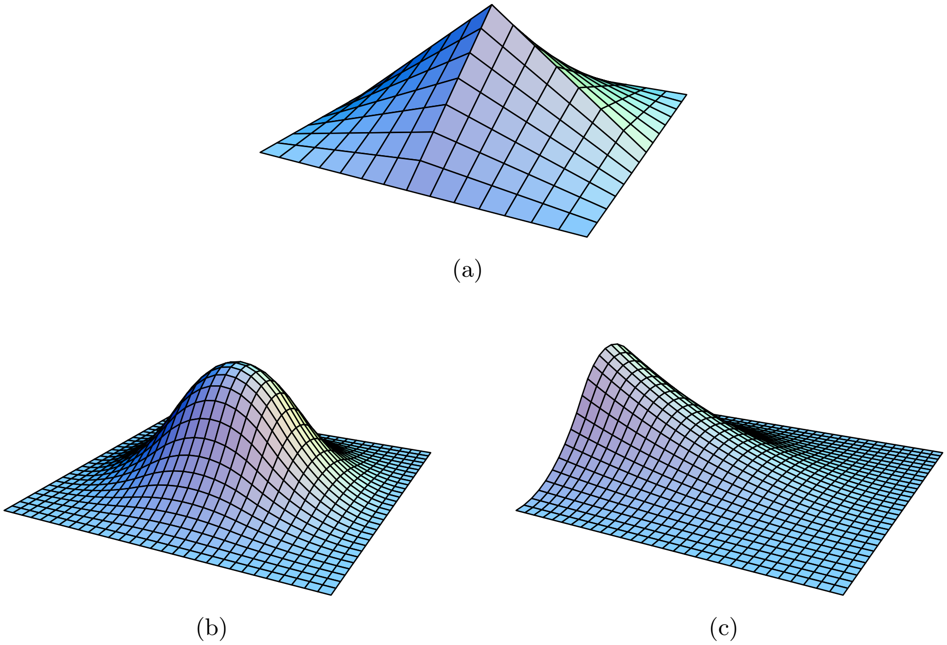


Figure 7.2. A bilinear B-spline (a), a biquadratic B-spline (b) and biquadratic B-spline with a triple knot in one direction (c).

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

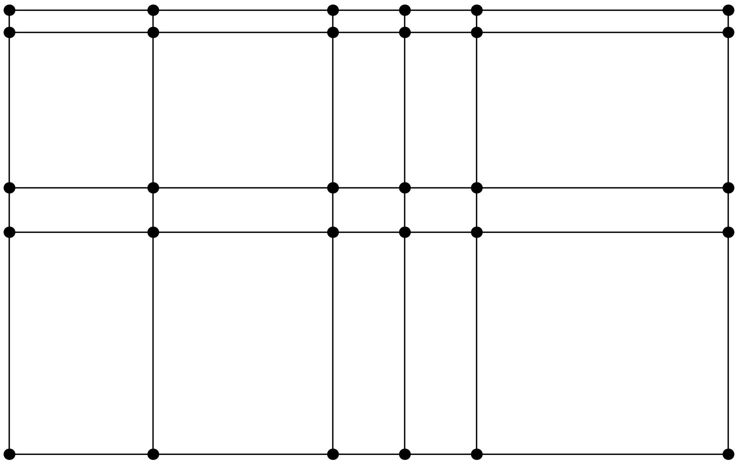


Figure 7.3. The knot lines for a tensor product spline surface.

involve sums of terms of the form

$$\frac{\partial}{\partial x}\left(c_{i,j}\phi_i(x)\psi_j(y)\right) = c_{i,j}\phi_i'(x)\psi_j(y).$$

A tensor product surface can be written conveniently in matrix-vector form. If $f(x,y)$ is given by (7.3) then

$$f(x,y) = \boldsymbol{\phi}(x)^T \boldsymbol{C} \boldsymbol{\psi}(y), \tag{7.4}$$

where

$$\boldsymbol{\phi} = (\phi_1, \ldots, \phi_{n_1})^T, \qquad \boldsymbol{\psi} = (\psi_1, \ldots, \psi_{n_2})^T,$$

and $\boldsymbol{C} = (c_{i,j})$ is the matrix of coefficients. This can be verified quite easily by expanding the multiplications.

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 generalisations 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 $f(x,y) = \boldsymbol{\phi}(x)^T \boldsymbol{C} \boldsymbol{\psi}(y)^T$, 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 $\sigma_\nu \le x < \sigma_{\nu+1}$ and $\tau_\mu \le y < \tau_{\mu+1}$, then we know that only $(\phi_i(x))_{i=\nu-d}^{\nu}$ and $(\psi_j(y))_{j=\mu-\ell}^{\mu}$ can be nonzero at (x,y). To compute

$$f(x,y) = \boldsymbol{\phi}(x)^T \boldsymbol{C} \boldsymbol{\psi}(y) \tag{7.5}$$

we therefore first make use of Algorithm 2.21 to compute the $d+1$ nonzero B-splines at x and the $\ell+1$ nonzero B-splines at y with the triangular down algorithm. We can then pick out that part of the coefficient matrix $\boldsymbol{C}$ 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 $\boldsymbol{C}$ 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 generalised 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

$$\Omega = \Big\{(x,y) \mid a_1 \leq x \leq b_1 \;\&\; a_2 \leq y \leq b_2\Big\} = [a_1,b_1]\times[a_2,b_2].$$

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 $\tau_{n_2} = b_2$. As above we let $\phi_i = B_{i,d,\boldsymbol{\sigma}}$ and $\psi_j = B_{j,\ell,\boldsymbol{\tau}}$ be the B-splines on $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$ respectively. The spline

$$Vf(x,y) = \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f(\sigma_i^*,\tau_j^*)\phi_i(x)\psi_j(y) \tag{7.6}$$

where

$$\begin{aligned}\sigma_i^* &= \sigma_{i,d}^* = (\sigma_{i+1}+\ldots+\sigma_{i+d})/d\\ \tau_j^* &= \tau_{j,\ell}^* = (\tau_{j+1}+\ldots+\tau_{j+\ell})/\ell,\end{aligned} \tag{7.7}$$

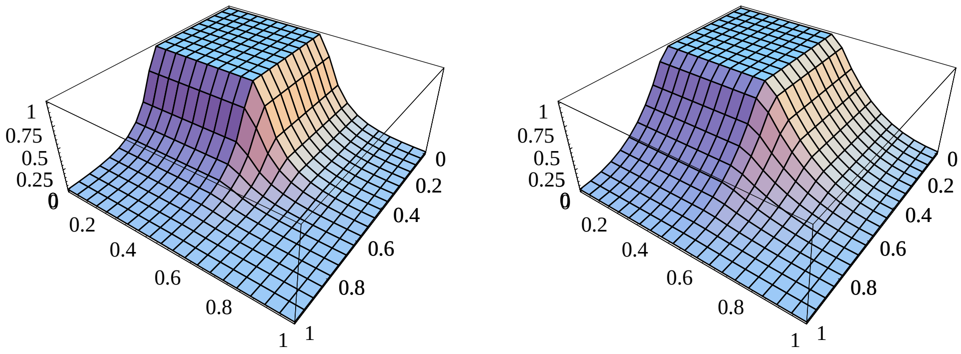
is called the *variation diminishing spline approximation on* $(\boldsymbol{\sigma},\boldsymbol{\tau})$ *of degree* (d,ℓ). If no interior knots in $\boldsymbol{\sigma}$ has multiplicity $d+1$ then

$$a_1 = \sigma_1^* < \sigma_2^* < \ldots < \sigma_{n_1}^* = b_1,$$

and similarly, if no interior knots in $\boldsymbol{\tau}$ has multiplicity $\ell+1$ then

$$a_2 = \tau_1^* < \tau_2^* < \ldots < \tau_{n_2}^* = b_2.$$

This means that the nodes $(\sigma_i^*,\tau_j^*)_{i,j=1}^{n_1,n_2}$ divides the domain Ω into a rectangular grid.



(a)

(b)

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).

Example 7.2. Suppose we want to approximate the function

$$f(x, y) = g(x)g(y), \tag{7.8}$$

where

$$g(x) = \begin{cases} 1, & 0 \le x \le 1/2, \\ e^{-10(x-1/2)}, & 1/2 < x \le 1, \end{cases}$$

on the unit square

$$\Omega = \Big\{(x, y) \ \mid \ 0 \le x \le 1 \ \& \ 0 \le y \le 1\Big\} = [0, 1]^2.$$

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 $\boldsymbol{\sigma}$ and $\boldsymbol{\tau}$. For an integer q with $q \ge 2$ we define the knot vectors by

$$\begin{aligned} \boldsymbol{\sigma} = \boldsymbol{\tau} = (&0, 0, 0, 0, 1/(2q), \ldots, 1/2 - 1/(2q), 1/2, 1/2, 1/2, \\ &1/2 + 1/(2q), \ldots 1 - 1/(2q), 1, 1, 1, 1). \end{aligned}$$

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

$$(x_i, y_j, f_{i,j})_{i=1,j=1}^{m_1,m_2}, \tag{7.9}$$

where

$$a_1 = x_1 < x_2 < \cdots < x_{m_1} = b_1, \qquad a_2 = y_1 < y_2 < \cdots < y_{m_2} = b_2.$$

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

$$g(x_i, y_j) = f_{i,j}, \quad i = 1, \ldots, m_1, \quad j = 1, \ldots, m_2. \tag{7.10}$$

We think of $\mathbb{S}_1$ and $\mathbb{S}_2$ as two univariate spline spaces

$$\mathbb{S}_1 = \operatorname{span}\{\phi_1, \ldots, \phi_{m_1}\}, \qquad \mathbb{S}_2 = \operatorname{span}\{\psi_1, \ldots, \psi_{m_2}\}, \tag{7.11}$$

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 in the form

$$g(x, y) = \sum_{p=1}^{m_1} \sum_{q=1}^{m_2} c_{p,q} \psi_q(y) \phi_p(x) \tag{7.12}$$

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

$$\sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) = f_{i,j}, \tag{7.13}$$

$$\sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) = d_{p,j}. \tag{7.14}$$

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

$$\begin{aligned}
&\boldsymbol{\Phi} = (\phi_{i,p}) \in \mathbb{R}^{m_1,m_1}, \quad \phi_{i,p} = \phi_p(x_i), \\
&\boldsymbol{\Psi} = (\psi_{j,q}) \in \mathbb{R}^{m_2,m_2}, \quad \psi_{j,q} = \psi_q(y_j), \\
&\boldsymbol{D} = (d_{p,j}) \in \mathbb{R}^{m_1,m_2},\ \boldsymbol{F} = (f_{i,j}) \in \mathbb{R}^{m_1,m_2},\ \boldsymbol{C} = (c_{p,q}) \in \mathbb{R}^{m_1,m_2}.
\end{aligned} \tag{7.15}$$

We then see that in (7.13) and (7.14)

$$\sum_{p=1}^{m_1} d_{p,j} \phi_p(x_i) = \sum_{p=1}^{m_1} \phi_{i,p} d_{p,j} = (\boldsymbol{\Phi} \boldsymbol{D})_{i,j} = (\boldsymbol{F})_{i,j},$$

$$\sum_{q=1}^{m_2} c_{p,q} \psi_q(y_j) = \sum_{q=1}^{m_2} \psi_{j,q} c_{p,q} = (\boldsymbol{\Psi} \boldsymbol{C}^T)_{j,p} = (\boldsymbol{D}^T)_{j,p}.$$

It follows that (7.13) and (7.14) can be written in the following matrix form

$$\boldsymbol{\Phi} \boldsymbol{D} = \boldsymbol{F} \quad \text{and} \quad \boldsymbol{C} \boldsymbol{\Psi}^T = \boldsymbol{D}. \tag{7.16}$$

From these equations we obtain the following proposition.

Proposition 7.3. *Suppose the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ 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 $\boldsymbol{C} = (c_{p,q})$ satisfies the matrix equation*

$$\mathbf{\Phi} \boldsymbol{C} \mathbf{\Psi}^T = \boldsymbol{F}.$$

Proof. The above derivation shows that there is a unique $g \in \mathbb{S}_1 \otimes \mathbb{S}_2$ such that (7.10) holds if and only if the matrix equations in (7.16) have unique solutions $\boldsymbol{D}$ and $\boldsymbol{C}$. But this is the case if and only if the matrices $\mathbf{\Phi}$ and $\mathbf{\Psi}$ 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

$$X_j(x) = \sum_{p=1}^{m_1} d_{p,j} \phi_p(x), \quad j = 1, 2, \ldots, m_2.$$

Here the $d_{p,j}$ are taken from (7.13). Then for each j we have

$$X_j(x_i) = f_{i,j}, \quad i = 1, 2, \ldots, m_1.$$

We see that X_j is a curve which interpolates the data $\boldsymbol{f}_j = (f_{1,j}, \ldots, f_{m_1,j})$ at the y-level y_j. Moreover, by using (7.10) we see that for all x

$$X_j(x) = g(x, y_j), \quad j = 1, 2, \ldots, m_2.$$

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 $\boldsymbol{f}_j$.

(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

$$\sum_{q=1}^{m_2} e_{i,q} \psi_q(y_j) = f_{i,j}, \tag{7.17}$$

$$\sum_{p=1}^{m_1} c_{p,q} \phi_p(x_i) = e_{i,q}. \tag{7.18}$$

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 $F_i = (f_{i,1}, \ldots, f_{i,m_2})$. 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 $\boldsymbol{x}, \boldsymbol{f}$ into a spline interpolating this data. We can write such a map as

$$g = I[\boldsymbol{x}, \boldsymbol{f}] = \sum_{p=1}^{m_1} c_p \phi_p.$$

The coefficients $\boldsymbol{c} = (c_p)$ are determined from the interpolation requirements $g(x_i) = f_i$ for $i = 1, 2, \ldots, m_1$. We also have a related map $\tilde{I}$ which maps the data into the coefficients

$$\boldsymbol{c} = \tilde{I}[\boldsymbol{x}, \boldsymbol{f}].$$

Given m_2 data sets $(x_i, f_{i,j})_{i=1}^{m_1}$ for $j = 1, 2, \ldots, m_2$, we combine the function values into a matrix

$$\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_n) = (f_{i,j}) \in \mathbb{R}^{m_1, m_2}$$

and define

$$\boldsymbol{C} = \tilde{I}[\boldsymbol{x}, \boldsymbol{F}] = (\tilde{I}[\boldsymbol{x}, \boldsymbol{f}_1], \ldots, \tilde{I}[\boldsymbol{x}, \boldsymbol{f}_n]). \tag{7.19}$$

With this notation the equations in (7.16) correspond to

$$\boldsymbol{D} = \tilde{I}_1[\boldsymbol{x}, \boldsymbol{F}], \quad \boldsymbol{C}^T = \tilde{I}_2[\boldsymbol{y}, \boldsymbol{D}^T],$$

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

$$\boldsymbol{C} = (\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = \tilde{I}_2[\boldsymbol{y}, \tilde{I}_1[\boldsymbol{x}, \boldsymbol{F}]^T]^T. \tag{7.20}$$

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 $(I_1 \otimes I_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}]$ as the spline in $\mathbb{S}_1 \otimes \mathbb{S}_2$ with coefficients $(\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}]$.

These operators can be applied in any order. We can apply I_1 on each of the data vectors $\boldsymbol{f}_j$ 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

$$(\tilde{I}_1 \otimes \tilde{I}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = (\tilde{I}_2 \otimes \tilde{I}_1)[\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{F}^T].$$

Tensor product interpolation is quite easy to program on a computer. In order to implement the $\tilde{I}[\boldsymbol{x}, \boldsymbol{F}]$ 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 $I[\boldsymbol{x}, \boldsymbol{f}]$ we would have a program

$$I^P[\boldsymbol{x}, \boldsymbol{f}, d, \boldsymbol{\tau}, \boldsymbol{c}],$$

which to given data $\boldsymbol{x}$ and $\boldsymbol{f}$ will return a spline space represented by the degree d and the knot vector $\boldsymbol{\tau}$, and the coefficients $\boldsymbol{c}$ 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 $\mathbb{S}_1 = \mathbb{S}_{q,\boldsymbol{\sigma}}$ and $\mathbb{S}_2 = \mathbb{S}_{\ell,\boldsymbol{\tau}}$. Assuming that these programs can handle data of the form $\boldsymbol{x}, \boldsymbol{F}$, a program to carry out the process in (7.20) would be

1. $I_1^P[\boldsymbol{x}, \boldsymbol{F}, d, \boldsymbol{\sigma}, \boldsymbol{D}]$;
2. $I_2^P[\boldsymbol{y}, \boldsymbol{D}^T, \ell, \boldsymbol{\tau}, \boldsymbol{G}]$;
3. $\boldsymbol{C} = \boldsymbol{G}^T$;

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*

$$\min_{g \in \mathbb{S}_1 \otimes \mathbb{S}_2} \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} w_i v_j \left[g(x_i, y_j) - f_{i,j}\right]^2.$$

We assume that the vectors of data abscissas $\boldsymbol{x} = (x_i)_{i=1}^{m_1}$ and $\boldsymbol{y} = (y_j)_{j=1}^{m_2}$ 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

$$\mathbb{S}_1 = \operatorname{span}\{\phi_1, \ldots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \operatorname{span}\{\psi_1, \ldots, \psi_{n_2}\},$$

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*

$$\min_{\boldsymbol{C} \in \mathbb{R}^{n_1,n_2}} \|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T - \boldsymbol{G}\|^2, \tag{7.21}$$

where

$$\begin{aligned} \boldsymbol{A} &= (a_{i,p}) \in \mathbb{R}^{m_1,n_1}, \quad a_{i,p} = \sqrt{w_i}\phi_p(x_i), \\ \boldsymbol{B} &= (b_{j,q}) \in \mathbb{R}^{m_2,n_2}, \quad b_{j,q} = \sqrt{v_j}\psi_q(y_j), \\ \boldsymbol{G} &= (\sqrt{w_i}\sqrt{v_j} f_{i,j}) \in \mathbb{R}^{m_1,m_2}, \ \boldsymbol{C} = (c_{p,q}) \in \mathbb{R}^{n_1,n_2}. \end{aligned} \tag{7.22}$$

Here, the norm $\|\cdot\|$ is the Frobenius norm,

$$\|\boldsymbol{E}\| = \Big(\sum_{i=1}^{m}\sum_{j=1}^{n}|e_{i,j}|^2\Big)^{1/2} \tag{7.23}$$

for any rectangular $m\times n$ matrix $\boldsymbol{E}=(e_{i,j})$.

Proof. Suppose $\boldsymbol{C}=(c_{p,q})$ are the B-spline coefficients of some $g\in\mathbb{S}_1\otimes\mathbb{S}_2$. Then

$$\begin{aligned}\|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T-\boldsymbol{G}\|^2 &= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\Big(\sum_{p=1}^{n_1}\sum_{q=1}^{n_2}a_{i,p}c_{p,q}b_{j,q}-g_{i,j}\Big)^2\\ &= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\Big(\sum_{p=1}^{n_1}\sum_{q=1}^{n_2}\sqrt{w_i}\phi_p(x_i)c_{p,q}\sqrt{v_j}\psi_q(y_j)-\sqrt{w_i}\sqrt{v_j}f_{i,j}\Big)^2\\ &= \sum_{i=1}^{m_1}\sum_{j=1}^{m_2}w_iv_j\left[g(x_i,y_j)-f_{i,j}\right]^2.\end{aligned}$$

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 $\boldsymbol{C}=\boldsymbol{C}^*$, and the solution is unique if and only if both matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ have linearly independent columns. The solution $\boldsymbol{C}^*$ can be found by solving the matrix equation*

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T\boldsymbol{B}=\boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B}. \tag{7.24}$$

Proof. By arranging the entries of $\boldsymbol{C}$ 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 $\boldsymbol{H}=(h_{i,j})$ and $\boldsymbol{K}=(k_{i,j})$ in $\mathbb{R}^{m,n}$ we define the scalar product

$$(\boldsymbol{H},\boldsymbol{K})=\sum_{i=1}^{m}\sum_{j=1}^{n}h_{i,j}q_{i,j}.$$

This is a scalar product of the matrices $\boldsymbol{H}$ and $\boldsymbol{K}$ regarded as vectors. We have $(\boldsymbol{H},\boldsymbol{H})=\|\boldsymbol{H}\|^2$, the Frobenius norm of $\boldsymbol{H}$, squared. We also observe that for any $m\times n$ matrices $\boldsymbol{H}$ and $\boldsymbol{K}$, we have

$$\|\boldsymbol{H}+\boldsymbol{K}\|^2=\|\boldsymbol{H}\|^2+2(\boldsymbol{H},\boldsymbol{K})+\|\boldsymbol{K}\|^2.$$

Moreover,

$$(\boldsymbol{E},\boldsymbol{H}\boldsymbol{K})=(\boldsymbol{H}^T\boldsymbol{E},\boldsymbol{K})=(\boldsymbol{E}\boldsymbol{K}^T,\boldsymbol{H}), \tag{7.25}$$

for any matrices $\boldsymbol{E},\boldsymbol{H},\boldsymbol{K}$ such that the matrix operations make sense. For any $\boldsymbol{C}\in\mathbb{R}^{n_1,n_2}$ we let

$$q(\boldsymbol{C})=\|\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T-\boldsymbol{G}\|^2.$$

This is the function we want to minimise. Suppose $\boldsymbol{C}^*$ is the solution of (7.24). We want to show that $q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) \geq q(\boldsymbol{C}^*)$ for any real ϵ and any $\boldsymbol{D} \in \mathbb{R}^{n_1 \times n_2}$. This will follow from the relation

$$q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) = q(\boldsymbol{C}^*) + 2\epsilon(\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B}, \boldsymbol{D}) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2. \tag{7.26}$$

For if $\boldsymbol{C}^*$ satisfies (7.24) then the complicated middle term vanishes and

$$q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) = q(\boldsymbol{C}^*) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2 \geq q(\boldsymbol{C}^*).$$

To establish (7.26) we have to expand $q(\boldsymbol{C}^* + \epsilon \boldsymbol{D})$,

$$\begin{aligned} q(\boldsymbol{C}^* + \epsilon \boldsymbol{D}) &= \|(\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T - \boldsymbol{G}) + \epsilon\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2 \\ &= q(\boldsymbol{C}^*) + 2\epsilon(\boldsymbol{A}\boldsymbol{C}^*\boldsymbol{B}^T - \boldsymbol{G}, \boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T) + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2. \end{aligned}$$

Using (7.25) on the middle term, we can move $\boldsymbol{A}$ and $\boldsymbol{B}^T$ 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 $\boldsymbol{C}$ does not satisfy (7.24). We need to show that $\boldsymbol{C}$ does not minimise q. Now, for at least one matrix component i, j we have

$$z = (\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{C}\boldsymbol{B}^T\boldsymbol{B} - \boldsymbol{A}^T\boldsymbol{G}\boldsymbol{B})_{i,j} \neq 0.$$

We choose $\boldsymbol{D}$ as the matrix where the i, j element is equal to 1 and all other entries are 0. Then (7.26) takes the form

$$q(\boldsymbol{C} + \epsilon\boldsymbol{D}) = q(\boldsymbol{C}) + 2\epsilon z + \epsilon^2\|\boldsymbol{A}\boldsymbol{D}\boldsymbol{B}^T\|^2,$$

and this implies that $q(\boldsymbol{C} + \epsilon\boldsymbol{D}) < q(\boldsymbol{C})$ for $\epsilon z < 0$ and $|\epsilon|$ sufficiently small. But then $\boldsymbol{C}$ 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 $\boldsymbol{D}$ from the system $\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{D} = \boldsymbol{A}^T\boldsymbol{G}$.
2. Find $\boldsymbol{C}$ from the system $\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{C}^T = \boldsymbol{B}^T\boldsymbol{D}^T$.

The matrix $\boldsymbol{C}$ is then the solution of (7.24). The first step is equivalent to

$$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{d}_j = \boldsymbol{A}^T\boldsymbol{g}_j, \quad j = 1, 2, \ldots, m_2,$$

where $\boldsymbol{D} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_{m_2})$ and $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{m_2})$. This means that we need to solve m_2 linear least squares problems

$$\min \|\boldsymbol{A}\boldsymbol{d}_j - \boldsymbol{g}_j\|_2^2, \quad j = 1, 2, \ldots, m_2.$$

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

$$\min \|\boldsymbol{B}\boldsymbol{h}_i - \boldsymbol{e}_i\|_2^2, \quad i = 1, 2, \ldots, n_1,$$

where the $\boldsymbol{e}_i$ are the rows of $\boldsymbol{D}$, and the $\boldsymbol{h}_i$ are the rows of $\boldsymbol{C}$

$$\boldsymbol{D} = \begin{pmatrix} \boldsymbol{e}_1^T \\ \vdots \\ \boldsymbol{e}_{n_1}^T \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} \boldsymbol{h}_1^T \\ \vdots \\ \boldsymbol{h}_{n_1}^T \end{pmatrix}.$$

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 $L[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}]$ which to given univariate data $\boldsymbol{x} = (x_i)_{i=1}^m$ and $\boldsymbol{f} = (f_i)_{i=1}^m$, and positive weights $\boldsymbol{w} = (w_i)_{i=1}^m$, assigns a spline

$$g = L[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}] = \sum_{p=1}^{n} c_p \phi_p,$$

in a spline space $\mathbb{S} = \operatorname{span}\{\phi_1, \ldots, \phi_n\}$. We also have the operator $\tilde{L}[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{f}]$ which maps the data into the B-spline coefficients and is defined analagously 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

$$\boldsymbol{C} = (\tilde{L}_1 \otimes \tilde{L}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}, \boldsymbol{w}, \boldsymbol{v}] = \tilde{L}_2[\boldsymbol{y}, \boldsymbol{v}, \tilde{L}_1[\boldsymbol{x}, \boldsymbol{w}, \boldsymbol{F}]^T]^T, \tag{7.27}$$

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 $\tilde{Q}_1[\boldsymbol{x}, \boldsymbol{f}]$ and $\tilde{Q}_2[\boldsymbol{y}, \boldsymbol{g}]$ define univariate approximation methods then we define their tensor product as

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = \tilde{Q}_2[\boldsymbol{y}, \tilde{Q}_1[\boldsymbol{x}, \boldsymbol{F}]^T]^T. \tag{7.28}$$

In this section we want to show that

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[x, y, \boldsymbol{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[y, x, \boldsymbol{F}^T]$$

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 $Q[\boldsymbol{x}, \boldsymbol{f}]$ is a univariate approximation operator mapping the data into a spline in a univariate spline space

$$\mathbb{S} = \operatorname{span}\{\phi_1, \ldots, \phi_n\}.$$

Thus

$$Q[\boldsymbol{x}, \boldsymbol{f}] = \sum_{p=1}^{n} a_p(\boldsymbol{f})\phi_p(x). \tag{7.29}$$

The coefficients $a_p(\boldsymbol{f})$ of the spline are functions of both $\boldsymbol{x}$ and $\boldsymbol{f}$, but here we are mostly interested in the dependence of $\boldsymbol{f}$. We also let $(a_p(\boldsymbol{f})) = \tilde{Q}[\boldsymbol{x}, \boldsymbol{f}]$ be the coefficients of $Q[\boldsymbol{x}, \boldsymbol{f}]$. We are interested in the following class of operators Q.

Definition 7.7. *The operator $Q : \mathbb{R}^m \to \mathbb{S}$ given by (7.29) is linear if*

$$a_p(\boldsymbol{f}) = \sum_{i=1}^{m} a_{p,i} f_i, \tag{7.30}$$

for suitable numbers $a_{p,i}$ independent of $\boldsymbol{f}$.

If Q is linear then

$$Q[\boldsymbol{x}, \alpha\boldsymbol{g} + \beta\boldsymbol{h}] = \alpha Q[\boldsymbol{x}, \boldsymbol{g}] + \beta Q[\boldsymbol{x}, \boldsymbol{h}]$$

for all $\alpha, \beta \in \mathbb{R}$ and all $\boldsymbol{g}, \boldsymbol{h} \in \mathbb{R}^m$.

Example 7.8. All methods in Chapter 5 are linear approximation schemes.

1. For the Schoenberg Variation Diminishing Spline Approximation we have $\boldsymbol{f} = (f_1, \ldots, f_m) = (f(\tau_1^*), \ldots, f(\tau_m^*))$. Thus Vf is of the form (7.29) with $a_p(\boldsymbol{f}) = f_p$, 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 $\boldsymbol{c} = (c_p)$ are found by solving a linear system $\boldsymbol{\Phi c} = \boldsymbol{f}$. Thus $\boldsymbol{c} = \boldsymbol{\Phi}^{-1}\boldsymbol{f}$, and c_p is of the form (7.30) with $a_{p,i}$ being the (p,i)-element of $\boldsymbol{\Phi}^{-1}$. 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

   $$\min_{\boldsymbol{c}} \sum_{i=1}^{m} w_i \left[\sum_{p=1}^{n} c_p\phi_p(x_i) - f_i\right]^2.$$

   The vector $\boldsymbol{c}$ is determined as the solution of a linear system

   $$\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{c} = \boldsymbol{A}^T\boldsymbol{f}.$$

   Thus $a_{p,i}$ is the (p,i)-element of the matrix $(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$.

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*

$$(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}) = (x_i, y_j, f_{i,j})_{i=1,j=1}^{m_1,m_2}, \tag{7.31}$$

we have

$$(\tilde{Q}_1 \otimes \tilde{Q}_2)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{F}] = (\tilde{Q}_2 \otimes \tilde{Q}_1)[\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{F}^T].$$

Proof. To see this we go through the constructions in detail. Suppose that

$$Q_1[\boldsymbol{x},\boldsymbol{f}] = \sum_{p=1}^{n_1} a_p(\boldsymbol{f})\phi_p, \quad a_p(\boldsymbol{f}) = \sum_{i=1}^{m_1} a_{p,i} f_i,$$

$$Q_2[\boldsymbol{y},\boldsymbol{g}] = \sum_{q=1}^{n_2} b_p(\boldsymbol{g})\psi_p, \quad b_q(\boldsymbol{g}) = \sum_{j=1}^{m_2} b_{q,j} g_j.$$

The matrix $\boldsymbol{F} = (f_{i,j})) \in \mathbb{R}^{m_1,m_2}$ can be partitioned either by rows or by columns.

$$\boldsymbol{F} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_{m_2}) = \begin{pmatrix} \boldsymbol{g}_1 \\ \vdots \\ \boldsymbol{g}_{m_1} \end{pmatrix}.$$

If we use Q_1 first then we obtain a family of x-curves from the columns $\boldsymbol{f}_j$ of the data $\boldsymbol{F}$

$$Q_1[\boldsymbol{x},\boldsymbol{f}_j] = \sum_{p=1}^{n_1} a_p(\boldsymbol{f}_j)\phi_p(x), \quad j = 1, 2, \ldots, m_2.$$

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

$$c_{p,q} = b_q\left(a_p(\boldsymbol{f}_1), \ldots, a_p(\boldsymbol{f}_{m_2})\right).$$

Using the linearity we obtain

$$c_{p,q} = \sum_{j=1}^{m_2} b_{q,j} a_p(\boldsymbol{f}_j) = \sum_{j=1}^{m_2}\sum_{i=1}^{m_1} b_{q,j} a_{p,i} f_{i,j}. \tag{7.32}$$

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

$$d_{p,q} = a_p\left(b_q(\boldsymbol{g}_1), \ldots, b_q(\boldsymbol{g}_{m_1})\right).$$

Thus,

$$d_{p,q} = \sum_{i=1}^{m_1} a_{p,i} b_q(\boldsymbol{g}_i) = \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} a_{p,i} b_{q,j} f_{i,j}.$$

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. ∎

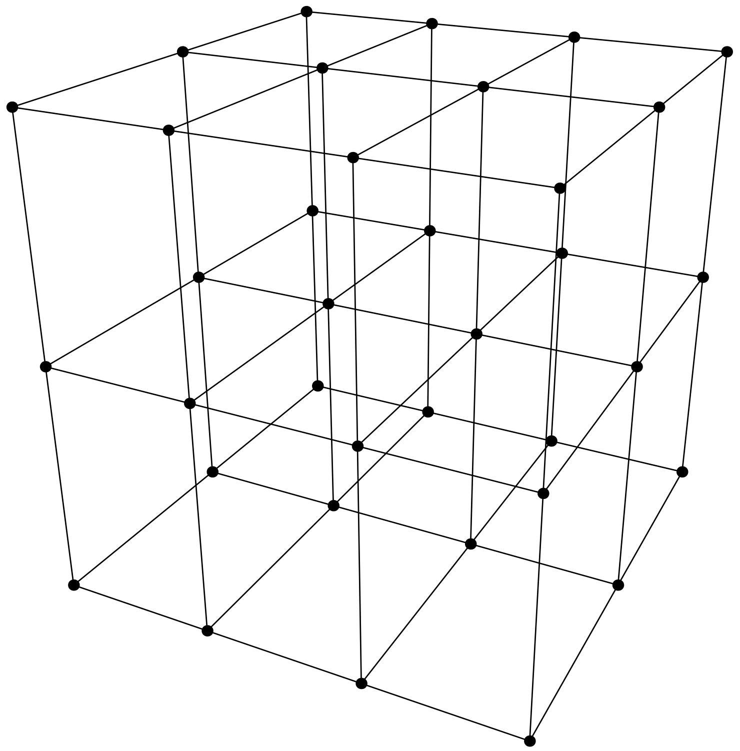


Figure 7.5. A cubical gridded region in space.

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

$$(x_i, y_j, z_k, f_{i,j,k})_{i=1,j=1,k=1}^{m_1,\ m_2,\ m_3}. \tag{7.33}$$

Here the f's are function values of an unknown trivariate function

$$f = f(x, y, z).$$

The data is given on a cubical region determined from the grid points (x_i, y_j, z_k) in space. We write

$$\boldsymbol{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1,m_2,m_3}$$

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

$$g(x, y, z) = \sum_{p=1}^{n_1} \sum_{q=1}^{n_2} \sum_{r=1}^{n_3} c_{p,q,r} \omega_r(z) \psi_q(y) \phi_p(x). \tag{7.34}$$

Here

$$\mathbb{S}_1 = \operatorname{span}\{\phi_1, \ldots, \phi_{n_1}\}, \quad \mathbb{S}_2 = \operatorname{span}\{\psi_1, \ldots, \psi_{n_2}\}, \quad \mathbb{S}_3 = \operatorname{span}\{\omega_1, \ldots, \omega_{n_3}\},$$

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

$$\begin{aligned} \sum_{p=1}^{n_1} d_p(y_j,z_k)\phi_p(x_i) &= f_{i,j,k}, \quad i = 1,2,\ldots,m_1, \\ \sum_{q=1}^{n_2} e_{p,q}(z_k)\psi_q(y_j) &= d_p(y_j,z_k), \quad j = 1,2,\ldots,m_2, \\ \sum_{r=1}^{n_3} c_{p,q,r}\omega_r(z_k) &= e_{p,q}(z_k). \quad k = 1,2,\ldots,m_3, \end{aligned} \tag{7.36}$$

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

$$\boldsymbol{A} = (a_{i_1,i_2,\ldots,i_s})_{i_1=1,i_2=1,\ldots,i_s=1}^{m_1,\ m_2,\ \ldots\ ,m_s}.$$

We write

$$\boldsymbol{A} \in \mathbb{R}^{m_1,m_2,\ldots,m_s} = \mathbb{R}^{\boldsymbol{m}},$$

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 $\boldsymbol{A} \in \mathbb{R}^{m_1,m_2,\ldots,m_s}$ and $\boldsymbol{B} \in \mathbb{R}^{n_1,n_2,\ldots,n_e}$ can be defined if the last dimension of $\boldsymbol{A}$ equals the first dimension of $\boldsymbol{B}$. Indeed, with $m = m_s = n_1$, we define the product $\boldsymbol{AB}$ as the tensor

$$\boldsymbol{C} = \boldsymbol{AB} \in \mathbb{R}^{m_1,m_2,\ldots,m_{s-1},n_2,\ldots,n_s}$$

with elements

$$c_{i_1,\ldots,i_{s-1},j_2,\ldots,j_e} = \sum_{i=1}^{m} a_{i_1,\ldots,i_{s-1},i} b_{i,j_2,\ldots,j_e}.$$

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 $\boldsymbol{A}$ and the first index of $\boldsymbol{B}$. 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

$$\boldsymbol{\Phi D} = \boldsymbol{F}. \tag{7.37}$$

Here

$$\begin{aligned}&\boldsymbol{\Phi} = (\phi_{i,p}) = (\phi_p(x_i)) \in \mathbb{R}^{m_1,n_1},\\ &\boldsymbol{D} = (d_{p,j,k}) = d_p(y_j, z_k) \in \mathbb{R}^{n_1,m_2,m_3}, \quad \boldsymbol{F} = (f_{i,j,k}) \in \mathbb{R}^{m_1,m_2,m_3}.\end{aligned}$$

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

$$\begin{aligned}&\boldsymbol{\Psi} = (\psi_{j,q}) = (\psi_q(y_j)) \in \mathbb{R}^{m_2,n_2},\\ &\boldsymbol{E} = (e_{q,k,p}) = (e_{p,q}(z_k)) \in \mathbb{R}^{n_2,m_3,n_1}, \quad \boldsymbol{D}' = (d_{j,k,p}) \in \mathbb{R}^{m_2,m_3,n_1}.\end{aligned}$$

The next equation can then be written

$$\boldsymbol{\Psi E} = \boldsymbol{D}'. \tag{7.38}$$

The construction of $\boldsymbol{D}'$ from $\boldsymbol{D}$ 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 $\boldsymbol{E}$ for the last equation in (7.36). We obtain

$$\boldsymbol{\Omega G} = \boldsymbol{E}', \tag{7.39}$$

where

$$\begin{aligned}&\boldsymbol{\Omega} = (\omega_{k,r}) = (\omega_r(z_k)) \in \mathbb{R}^{m_3,n_3},\\ &\boldsymbol{E}' = (e_{k,p,q}) = (e_{p,q}(z_k)) \in \mathbb{R}^{m_3,n_1,n_2}, \quad \boldsymbol{G} = (g_{r,p,q}) \in \mathbb{R}^{n_3,n_1,n_2}.\end{aligned}$$

The coefficients $\boldsymbol{C}'$ are obtained by a final cyclic rotation of the dimensions

$$\boldsymbol{C} = \boldsymbol{G}'. \tag{7.40}$$

The systems (7.37), (7.38), and (7.39) corresponds to three univariate operators of the form $Q[\boldsymbol{x}, \boldsymbol{f}]$. 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

$$(Q_1 \otimes Q_2 \otimes Q_3)[\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{F}] = Q_3[\boldsymbol{z}, Q_2[\boldsymbol{y}, Q_1[\boldsymbol{x}, F]']']'. \tag{7.41}$$

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 $\boldsymbol{p}_0$, $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$. We define the function $\boldsymbol{f} : \mathbb{R}^2 \mapsto P$ by

$$\boldsymbol{f}(u,v) = \boldsymbol{p}_0 + (\boldsymbol{p}_1 - \boldsymbol{p}_0)u + (\boldsymbol{p}_2 - \boldsymbol{p}_0)v. \tag{7.42}$$

We see that $\boldsymbol{f}(0,0) = \boldsymbol{p}_0$, while $\boldsymbol{f}(1,0) = \boldsymbol{p}_1$ and $\boldsymbol{f}(0,1) = \boldsymbol{p}_2$, so that $\boldsymbol{f}$ interpolates the three points. Since $\boldsymbol{f}$ 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 $\boldsymbol{f}$ of an open set $\Omega \subseteq \mathbb{R}^2$ onto S such that*

1. *$\boldsymbol{f}$ has continuous derivatives up to order m.*

Suppose that $\boldsymbol{f}(u,v) = \big(f^1(u,v), f^2(u,v), f^3(u,v)\big)$ and let $D_1\boldsymbol{f}$ and $D_2\boldsymbol{f}$ denote differentiation with respect to the first and second variables of $\boldsymbol{f}$, respectively. The parametric representation $\boldsymbol{f}$ is said to be regular if in addition

(ii) the Jacobian matrix of $\boldsymbol{f}$ given by

$$J(\boldsymbol{f}) = \begin{pmatrix} D_1f^1(u,v) & D_2f^1(u,v) \\ D_1f^2(u,v) & D_2f^2(u,v) \\ D_1f^3(u,v) & D_2f^3(u,v) \end{pmatrix}$$

has full rank for all (u,v) in Ω.

That $J(\boldsymbol{f})$ 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 $J(\boldsymbol{f})$.

A function of two variables $z = h(x,y)$ can always be considered as a parametric surface through the representation $\boldsymbol{f}(u,v) = \big(u, v, h(u,v)\big)$.

In the following we will always assume that $\boldsymbol{f}$ is sufficiently smooth for all operations on $\boldsymbol{f}$ 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

$$J(\boldsymbol{f}) = \big(\boldsymbol{p}_1 - \boldsymbol{p}_0, \boldsymbol{p}_2 - \boldsymbol{p}_0\big)\,,$$

(the two vectors $\boldsymbol{p}_1 - \boldsymbol{p}_0$ and $\boldsymbol{p}_2 - \boldsymbol{p}_0$ give the columns of $J(\boldsymbol{f})$). We see that $J(\boldsymbol{f})$ has full rank unless $\boldsymbol{p}_1 - \boldsymbol{p}_0 = \lambda(\boldsymbol{p}_2 - \boldsymbol{p}_0)$ 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 $\boldsymbol{f}(u, v_0)$ for fixed v_0 is called a u-curve, while a curve of the form $\boldsymbol{f}(u_0, v)$ 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 $D_1\boldsymbol{f}(u, v)$ and $D_2\boldsymbol{f}(u, v)$ 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 $D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)$ is nonzero and normal to the two tangent vectors.

Definition 7.11. *The unit normal of the regular parametric representation $\boldsymbol{f}$ is the vector*

$$\boldsymbol{N}(u, v) = \frac{D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)}{\|D_1\boldsymbol{f}(u, v) \times D_2\boldsymbol{f}(u, v)\|}.$$

The normal vector will play an important role when we start analysing the curvature of surfaces.

Let $\big(u(\sigma), v(\sigma)\big)$ be a regular curve in the domain Ω of a parametric representation $\boldsymbol{f}$. This curve is mapped to a curve $\boldsymbol{g}(\sigma)$ on the surface,

$$\boldsymbol{g}(\sigma) = \boldsymbol{f}\big(u(\sigma), v(\sigma)\big).$$

The tangent of $\boldsymbol{g}$ is given by

$$\boldsymbol{g}'(\sigma) = u'(\sigma)D_1\boldsymbol{f}\big(u(\sigma), v(\sigma)\big) + v'(\sigma)D_2\boldsymbol{f}\big(u(\sigma), v(\sigma)\big),$$

in other words, a linear combination of the two tangent vectors $D_1\boldsymbol{f}\big(u(\sigma), v(\sigma)\big)$ and $D_2\boldsymbol{f}\big(u(\sigma), v(\sigma)\big)$. Note that $\boldsymbol{g}$ is regular since $\boldsymbol{g}'(\sigma) = 0$ implies $u'(\sigma) = v'(\sigma) = 0$.

All regular curves on S through the point $\boldsymbol{f}(u, v)$ has a tangent vector on the form $\delta_1 D_1\boldsymbol{f} + \delta_2 D_2\boldsymbol{f}$, 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 $\boldsymbol{f}(u, v)$.

Definition 7.12. *Let S be a surface with a regular parametric representation $\boldsymbol{f}$. The tangent space or tangent plane $T\boldsymbol{f}(u, v)$ of S at $\boldsymbol{f}(u, v)$ is the plane in $\mathbb{R}^3$ spanned by the two vectors $D_1\boldsymbol{f}(u, v)$ and $D_2\boldsymbol{f}(u, v)$, i.e., all vectors on the form $\delta_1 D_1\boldsymbol{f}(u, v) + \delta_2 D_2\boldsymbol{f}(u, v)$.*

Note that the normal of the tangent plane $T\boldsymbol{f}(u, v)$ is the normal vector $\boldsymbol{N}(u, v)$.

7.5.1 Parametric Tensor Product Spline Surfaces

Given how we generalised from spline functions to parametric spline curves, the definition of parametric tensor product spline surfaces is the obvious generalisation of tensor product spline functions.

Definition 7.13. *A parametric tensor product spline surface is given by a parametric representation on the form*

$$\boldsymbol{f}(u, v) = \sum_{i=1}^{m} \sum_{j=1}^{n} \boldsymbol{c}_{i,j} B_{i,d,\boldsymbol{\sigma}}(u) B_{j,\ell,\boldsymbol{\tau}}(v),$$

where the coefficients $(\boldsymbol{c}_{i,j})_{i,j=1}^{m,n}$ are points in space,

$$\boldsymbol{c}_{i,j} = (c^1_{i,j}, c^2_{i,j}, c^3_{i,j}),$$

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 parametrisation, but we will not consider this in more detail here.

CHAPTER 8

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ that are nonzero so a spline g in $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\mu = j, j, \ldots, j+d$ (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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$. 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 $i = \mu-d, \ldots, \mu$. 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 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ of dimension n and the real function f defined on the interval $[\tau_{d+1}, \tau_{n+1}]$ be given, and suppose that $\boldsymbol{\tau}$ is a $d+1$-regular knot vector. To approximate f from the space $\mathbb{S}_{d,\boldsymbol{\tau}}$ perform the following steps for $j = 1, 2, \ldots, n$:*

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,*

$$g^I = P^I f^I = \sum_{i=\nu-d}^{\mu} b_i B_i, \tag{8.1}$$

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 $\mathbb{S}_{d,\boldsymbol{\tau},I}$ of the spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ to the interval I can be written as a linear combination of the B-splines $\{B_i\}_{i=\nu-d}^{\mu}$. 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 generalises 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 $d_1 \le d$. Then the global approximation method P will also reproduce polynomials of degree d_1. If all the local methods reproduce all the splines in $\mathbb{S}_{d,\boldsymbol{\tau},I}$ then P will reproduce the whole spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$.*

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 $\mathbb{S}_{d,\boldsymbol{\tau},I}$ and f is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$, then $Pf = f$. If f is in $\mathbb{S}_{d,\boldsymbol{\tau}}$ 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 $f^I = \sum_{i=\nu-d}^{\mu} \hat{b}_i B_i$. Since P^I reproduces $\mathbb{S}_{d,\boldsymbol{\tau},I}$ we will have $P^I f^I = f^I$ or

$$\sum_{i=\nu-d}^{\mu} b_i B_i = \sum_{i=\nu-d}^{\mu} \hat{b}_i B_i.$$

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 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 $\mathbb{S}_{d,\boldsymbol{\tau},I}$ 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 $\mathbb{S}_{1,\boldsymbol{\tau}}$ is given and that $\boldsymbol{\tau}$ 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 $\mathbb{S}_{1,\boldsymbol{\tau},I}$ 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 andτ_{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

$$P_1 f(x) = \sum_{i=1}^{n} f(\tau_{j+1}) B_j(x).$$

Since a straight line is completely characterised 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 $\mathbb{S}_{1,\boldsymbol{\tau}}$.

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 $\mathbb{S}_{2,\boldsymbol{\tau}}$ 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

$$b_j = \frac{1}{2}\bigl(-f(\tau_{j+1}) + 4f(\tau_{j+3/2}) - f(\tau_{j+2})\bigr),$$

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 2.12 we know that any spline g in $\mathbb{S}_{3,\boldsymbol{\tau}}$ 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 $\mathbb{S}_{2,\boldsymbol{\tau}}$.

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 $\mathbb{S}_{3,\boldsymbol{\tau},I}$ 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 $\mathbb{S}_{3,\boldsymbol{\tau},I}$. 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 $\mathbb{S}_{3,\boldsymbol{\tau},I}$ 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

$$\sum_{i=j-2}^{j+2} b_i B_i(x_{j,k}) = f(x_{j,k}), \quad \text{for } k = 0,\ 1,\ \ldots,\ 4.$$

In matrix-vector form this becomes

$$\begin{pmatrix} B_{j-2}(x_{j,0}) & B_{j-1}(x_{j,0}) & 0 & 0 & 0 \\ B_{j-2}(x_{j,1}) & B_{j-1}(x_{j,1}) & B_j(x_{j,1}) & B_j(x_{j,1}) & 0 \\ B_{j-2}(x_{j,2}) & B_{j-1}(x_{j,2}) & B_j(x_{j,2}) & B_j(x_{j,2}) & B_j(x_{j,2}) \\ 0 & B_{j-1}(x_{j,3}) & B_j(x_{j,3}) & B_j(x_{j,3}) & B_j(x_{j,3}) \\ 0 & 0 & 0 & B_j(x_{j,4}) & B_j(x_{j,4}) \end{pmatrix} \begin{pmatrix} b_{j-2} \\ b_{j-1} \\ b_j \\ b_{j+1} \\ b_{j+2} \end{pmatrix} = \begin{pmatrix} f(x_{j,0}) \\ f(x_{j,1}) \\ f(x_{j,2}) \\ f(x_{j,3}) \\ f(x_{j,4}) \end{pmatrix}$$

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 $\mathbb{S}_{3,\boldsymbol{\tau},I}$. The expression for $\lambda_j f$ is in general rather complicated, but in the special case where the width of the two knot intervals is equal and $x_{j,2}$ and $x_{j,4}$ are chosen as the midpoints of the two intervals we end up with

$$\lambda_j f = \frac{1}{6}\big(f(\tau_{j+1}) - 8f(\tau_{j+3/2}) + 20f(\tau_{j+2}) - 8f(\tau_{j+5/2}) + f(\tau_{j+3})\big)$$

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

$$\begin{aligned}\lambda_1 f &= f(\tau_4),\\ \lambda_2 f &= \frac{1}{18}\bigl(-5f(\tau_4) + 40f(\tau_{9/2}) - 36f(\tau_5) + 18f(\tau_{11/2}) - f(\tau_6)\bigr).\end{aligned}$$

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

$$\begin{aligned}\lambda_{n-1} f &= \frac{1}{18}\bigl(-f(\tau_{n-1}) + 18f(\tau_{n-1/2}) - 36f(\tau_n) + 40f(\tau_{n+1/2}) - 5f(\tau_{n+1})\bigr),\\ \lambda_n f &= f(\tau_{n+1}).\end{aligned}$$

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 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 are not linear, 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,*

$$P(\alpha f + \beta g) = \alpha Pf + \beta Pg.$$

Proof. Suppose that the linear coefficient functionals are $(\lambda_j)_{j=1}^n$. Then we have

$$P(\alpha f + \beta g) = \sum_{i=1}^{n} \lambda_j(\alpha f + \beta g)B_i = \alpha \sum_{i=1}^{n} \lambda_j f B_i + \beta \sum_{i=1}^{n} \lambda_j g B_i = \alpha Pf + \beta Pg$$

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 $\mathbb{S}_{2,\boldsymbol{\tau}}$ 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 $\mathbb{S}_{2,\boldsymbol{\tau},I}$ 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,

$$\min_{g \in \mathbb{S}_{2,\boldsymbol{\tau},I}} \sum_{k=0}^{9} \bigl(g(x_{j,k}) - f(x_{j,k})\bigr)^2.$$

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, …, 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

$$\lambda f = \int_a^b f(x)\phi(x)\,dx \tag{8.3}$$

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,

$$\lambda f = w_0 \int_a^b f(x)\phi_0(x)\,dx + w_1 \int_a^b f(x)\phi_1(x)\,dx + \cdots + w_\ell \int_a^b f(x)\phi_\ell(x)\,dx,$$

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

$$\phi(x) = w_0\phi_0(x) + w_1\phi_1(x) + \cdots + w_\ell\phi_\ell(x).$$

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

$$\lim_{\epsilon\to 0} \int_{I_\epsilon} f(x)\phi_\epsilon(x)\,dx = f(x_i), \tag{8.4}$$

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 $\bigl(x_i, f(x_i)\bigr)_{i=1}^m$ 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

$$\int_a^b g(x)x^i\,dx = \int_a^b f(x)x^i\,dx, \quad i = 0, 1, \ldots, n-1$$

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 $(\rho_i)_{i=1}^n$,

$$\rho_i g = \rho_i f, \quad \text{for } i = 1, 2, \ldots, n. \tag{8.5}$$

When $\rho_i f = \int_a^b f(x)x^{i-1}\,dx$ for $i = 1, \ldots, n$ we preserve moments, while if $\rho_i f = f(x_i)$ for $i = 1, \ldots, n$ 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

$$c_1\rho_i(\psi_1) + c_2\rho_i(\psi_2) + \cdots + c_n\rho_i(\psi_n) = \rho_i(f), \quad i = 1, \ldots, n$$

in the n unknown coefficients $(c_i)_{i=1}^n$. In matrix-vector form this becomes

$$\begin{pmatrix} \rho_1(\psi_1) & \rho_1(\psi_2) & \cdots & \rho_1(\psi_n) \\ \rho_2(\psi_1) & \rho_2(\psi_2) & \cdots & \rho_1(\psi_n) \\ \vdots & \vdots & \ddots & \vdots \\ \rho_n(\psi_1) & \rho_n(\psi_2) & \cdots & \rho_n(\psi_n) \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix} = \begin{pmatrix} \rho_1(f) \\ \rho_2(f) \\ \vdots \\ \rho_n(f) \end{pmatrix}. \tag{8.6}$$

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 $i = 0, \ldots, d$, 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 $i = 0, \ldots, d$.

If the equations (8.6) can be solved, each coefficient will be a linear combination of the entries on the right-hand side,

$$c_j = \lambda_j f = w_{j,1}\rho_1(f) + w_{j,2}\rho_2(f) + \cdots + w_{j,n}\rho_n(f).$$

We recognise this as (8.2) when the ρ_i correspond to point functionals, whereas we have

$$\begin{aligned} c_j = \lambda_j f &= w_{j,1}\int_a^b f(x)\,dx + w_{j,2}\int_a^b f(x)x\,dx + \cdots + w_{j,n}\int_a^b f(x)x^{n-1}\,dx \\ &= \int_a^b f(x)\bigl(w_{j,1} + w_{j,2}x + \cdots + w_{j,n}x^{n-1}\bigr)\,dx \end{aligned}$$

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

$$\min_{g\in\mathbb{S}} \int_a^b \bigl(f(x) - g(x)\bigr)^2\,dx.$$

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

$$\min_{(c_1,\ldots,c_n)\in\mathbb{R}^n} \int_a^b \Bigl(f(x) - \sum_{i=1}^n c_i\psi(x)\Bigr)^2\,dx.$$

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*

$$\sum_{i=1}^n c_i \int_a^b \psi_i(x)\psi_j(x)\,dx = \int_a^b \psi_j(x)f(x)\,dx, \quad \text{for } j = 1, \ldots, n.$$

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,

$$\int_a^b f(x)\,dx \approx w_0 f(x_0) + w_1 f(x_1) + \cdots + w_\ell f(x_\ell).$$

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 $\mathbb{S}_{2,\boldsymbol{\tau},I}$ 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

$$\lambda_j f = w_1 f(\tau_{j+1}) + w_2 f(\tau_{j+3/2}) + w_3 f(\tau_{j+2}). \tag{8.7}$$

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 $\mathbb{S}_{2,\boldsymbol{\tau},I}$. 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 jth 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 jth 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 jth B-spline of this polynomial, we make use of Corollary 3.5 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 jth B-spline coefficient of the first polynomial is therefore

$$\gamma_j\Big(\frac{ac - (b+c)b + bc}{(a-b)(a-c)}\Big) = \frac{ac - b^2}{(a-b)(a-c)} \tag{8.8}$$

which simplifies to $-1/2$ since $b = (a+c)/2$. Similarly, we find that the jth B-spline coefficient of the second and third polynomials are 2 and $-1/2$, respectively. The complete jth 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 $0 \le r \le d$ and let x_j be a number in $[\tau_j, \tau_{j+d+1}]$ for $j = 1, \ldots, n$. Consider the quasi-interpolant*

$$Q_{d,r} f = \sum_{j=1}^{n} \lambda_j(f) B_{j,d}, \quad \text{where} \quad \lambda_j(f) = \frac{1}{d!} \sum_{k=0}^{r} (-1)^k D^{d-k} \rho_{j,d}(x_j) D^k f(x_j), \tag{8.9}$$

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$.

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 jth 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 $y = x_i$ and rearranging, we obtain the jth B-spline coefficient of $(x - x_j)^k/k!$ as

$$\gamma_j\big((x - x_j)^k/k!\big) = (-1)^k D^{d-k} \rho_{j,d}(x_j)/d! \qquad \text{for } k = 0, \ldots, r.$$

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 agree with the local spline space $\mathbb{S}_{d,\boldsymbol{\tau},I}$ since I is a single knot interval. The quasi-interpolant therefore reproduces the whole spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ in this case. ∎

Example 8.6. We find

$$D^d \rho_{j,d}(y)/d! = 1, \quad D^{d-1} \rho_{j,d}(y)/d! = y - \tau_j^*, \quad \text{where} \quad \tau_j^* = \frac{\tau_{j+1} + \cdots + \tau_{j+d}}{d}. \tag{8.10}$$

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

$$Q_{2,2} f = \sum_{j=1}^{n} \big[f(x_j) - (x_j - \tau_{j+3/2}) Df(x_j) + \frac{1}{2}(x_j - \tau_{j+1})(x_j - \tau_{j+2}) D^2 f(x_j)\big] B_{j,2}. \tag{8.11}$$

while for $d = r = 3$ and $x_j = \tau_{j+2}$ we obtain

$$Q_{3,3} f = \sum_{j=1}^{n} \big[f(\tau_{j+2}) + \frac{1}{3}(\tau_{j+3} - 2\tau_{j+2} + \tau_{j+1}) Df(\tau_{j+2}) - \frac{1}{6}(\tau_{j+3} - \tau_{j+2})(\tau_{j+2} - \tau_{j+1}) D^2 f(\tau_{j+2})\big] B_{j,3}. \tag{8.12}$$

We leave the detailed derivation as a problem for the reader.

Since $Q_{d,d}f = f$ for all $f \in \mathbb{S}_{d,\boldsymbol{\tau}}$ it follows that the coefficients of a spline $f = \sum_{j=1}^{n} c_j B_{j,d}$ can be written in the form

$$c_j = \frac{1}{d!}\sum_{k=0}^{d}(-1)^k D^{d-k}\rho_{j,d}(x_j)D^k f(x_j), \qquad \text{for } j = 1, \ldots, n, \tag{8.13}$$

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

$$\tau_j \le x_{j,0} < x_{j,1} < \cdots < x_{j,r} \le \tau_{j+d+1} \tag{8.14}$$

located in the support $[\tau_j, \tau_{j+d+1}]$ of the B-spline $B_{j,d}$ for $j = 1, \ldots, n$. We consider the quasi-interpolant

$$P_{d,r}f = \sum_{j=1}^{n} \lambda_{j,r}(f)B_{j,d}, \tag{8.15}$$

where

$$\lambda_{j,r}(f) = \sum_{k=0}^{r} w_{j,k}f(x_{j,k}). \tag{8.16}$$

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ be a spline space with a $d+1$-regular knot vector $\boldsymbol{\tau} = (\tau_i)_{i=1}^{n+d+1}$. Let $(x_{j,k})_{k=0}^{r}$ be $\ell+1$ distinct points in $[\tau_j, \tau_{j+d+1}]$ for $j = 1, \ldots, n$, and let $w_{j,k}$ be the jth B-spline coefficient of the polynomial*

$$p_{j,k}(x) = \prod_{\substack{r=0\\ r\ne k}}^{r} \frac{x - x_{j,r}}{x_{j,k} - x_{j,r}}.$$

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

$$\tau_j \le \tau_{\ell_j} \le x_{j,0} < x_{j,1} < \cdots < x_{j,r} \le \tau_{\ell_j+1} \le \tau_{j+d+1} \tag{8.17}$$

then $P_{d,d}f = f$ for all $f \in \mathbb{S}_{d,\boldsymbol{\tau}}$.

Proof. It is not hard to see that

$$p_{j,k}(x_{j,i}) = \delta_{k,i}, \quad k, i = 0, \ldots, r$$

so that the polynomial

$$P^I_{d,r}f(x) = \sum_{k=0}^{r} p_{j,k}(x)f(x_{j,k})$$

satisfies the interpolation conditions $P^I_{d,r}f(x_{j,r}) = f(x_{j,r})$ 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 (9.14) 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

$$P_{d,1} f = \sum_{j=1}^{n} \left[\frac{x_{j,1} - \tau_j^*}{x_{j,1} - x_{j,0}} f(x_{j,0}) + \frac{\tau_j^* - x_{j,0}}{x_{j,1} - x_{j,0}} f(x_{j,1}) \right] B_{j,d}. \tag{8.18}$$

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

$$\tilde{P}_{d,1} f = \sum_{j=1}^{n} f(\tau_j^*) B_{j,d}. \tag{8.19}$$

This is again the *Variation diminishing method of Schoenberg.*

Exercises for Chapter 8

8.1 In this exercise we assume that the points $(x_{i,k})$ and the spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ are as in Theorem 8.7.

a) Show that for $r = d = 2$

$$\begin{aligned} P_{2,2} f = \sum_{j=1}^{n} \Bigg[&\frac{(\tau_{j+1} - x_{j,1})(\tau_{j+2} - x_{j,2}) + (\tau_{j+2} - x_{j,1})(\tau_{j+1} - x_{j,2})}{2(x_{j,0} - x_{j,1})(x_{j,0} - x_{j,2})} f(x_{j,0}) \\ &+ \frac{(\tau_{j+1} - x_{j,0})(\tau_{j+2} - x_{j,2}) + (\tau_{j+2} - x_{j,0})(\tau_{j+1} - x_{j,2})}{2(x_{j,1} - x_{j,0})(x_{j,1} - x_{j,2})} f(x_{j,1}) \\ &+ \frac{(\tau_{j+1} - x_{j,0})(\tau_{j+2} - x_{j,1}) + (\tau_{j+2} - x_{j,0})(\tau_{j+1} - x_{j,1})}{2(x_{j,2} - x_{j,0})(x_{j,2} - x_{j,1})} f(x_{j,2}) \Bigg] B_{j,2} \end{aligned} \tag{8.20}$$

b) Show that (8.20) reduces to the operator (9.4) 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ are is 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) $Q_{d,1} f = \sum_{j=1}^{n} \big[f(x_j) + (\tau_j - x_j) Df(x_j) \big] B_{j,d}, \quad (r = 1).$

c) $\tilde{Q}_{d,1} f = \sum_{j=1}^{n} f(\tau_j^*) B_{j,d}, \quad (r = 1).$

d)

$$\begin{aligned} Q_{2,2} f = \sum_{j=1}^{n} \big[&f(x_j) - (x_j - \tau_{j+3/2}) Df(x_j) \\ &+ \frac{1}{2}(x_j - \tau_{j+1})(x_j - \tau_{j+2}) D^2 f(x_j) \big] B_{j,2}, \text{(r=2)}. \end{aligned}$$

e) $\tilde{Q}_{2,2}f = \sum_{j=1}^{n} \bigl[f(\tau_{j+3/2}) - \frac{1}{2}(\tau_{j+2} - \tau_{j+1})^2 D^2 f(\tau_{j+3/2})\bigr] B_{j,2}, \quad (r = 2).$

f)

$$\begin{aligned} Q_{3,3}f = \sum_{j=1}^{n} \bigl[f(\tau_{j+2}) &+ \frac{1}{3}(\tau_{j+3} - 2\tau_{j+2} + \tau_{j+1}) Df(\tau_{j+2}) \\ &- \frac{1}{6}(\tau_{j+3} - \tau_{j+2})(\tau_{j+2} - \tau_{j+1}) D^2 f(\tau_{j+2})\bigr] B_{j,3}, \quad (r = 3). \end{aligned}$$

CHAPTER 9

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. We have already seen that 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 was proved in Section 4.1.

9.1 The distance to polynomials

We start by determining how well a given a real valued function f defined on an interval $[a,b]$ can be approximated by a polynomial of degree d. To measure the error in the approximation we will use the uniform norm which for a bounded function g defined on an interval $[a,b]$ is defined by

$$||g||_{\infty,[a,b]} = \sup_{a \le x \le b} \bigl|g(x)\bigr|.$$

Whenever we have an approximation p to f we can then 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 error is referred to as the *distance* from f to the space π_d of polynomials of degree $\leq d$ and is defined formally as

$$\operatorname{dist}_{\infty,[a,b]}(f, \pi_d) = \inf_{p \in \pi_d} ||f - p||_{\infty,[a,b]}.$$

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 $C^k_{\boldsymbol{\Delta}}[a, b]$ 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 kth derivative $D^k f$ is continuous everywhere except for a finite number of points in the interior (a, b), given by $\boldsymbol{\Delta} = (z_j)$. At the points of discontinuity $\boldsymbol{\Delta}$ 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 $C_{\boldsymbol{\Delta}}[a, b] = C^0_{\boldsymbol{\Delta}}[a, b]$. Note that we will often refer to these spaces without stating explicitly what the singularities $\boldsymbol{\Delta}$ are.

An upper bound for the distance of f to polynomials of degree d is fairly simple to give by choosing a particular approximation, namely Taylor expansion.

Theorem 9.1. *Given a polynomial degree d and a function f in $C^{d+1}_{\boldsymbol{\Delta}}[a, b]$, then*

$$\operatorname{dist}_{\infty,[a,b]}(f, \pi_d) \leq K_d h^{d+1} ||D^{d+1} f||_{\infty,[a,b]},$$

where $h = b - a$ and

$$K_d = \frac{1}{2^{d+1}(d + 1)!}$$

depends only on d.

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

$$\operatorname{dist}_{\infty,[a,b]}(f, \pi_d) \leq ||f - T_d f||_{\infty,[a,b]}. \tag{9.1}$$

To study the error we use 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 $x \leq m$ and combining it with (9.1) completes the proof of the theorem. ∎

We remark that the best possible constant K_d can actually be computed. In fact, for each $f \in C^{d+1}[a,b]$ there is a point $\xi \in [a,b]$ such that

$$\operatorname{dist}_{\infty,[a,b]}(f, \pi_d) = \frac{2}{4^{d+1}(d+1)!} h^{d+1} |D^{d+1} f(\xi)|$$

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 9.11. 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 a family of simple approximation methods called quasi-interpolants. As well as leading to good estimates of the distance between f and a spline space, many of the quasi-interpolants are good, practical approximation methods.

We consider a spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ where d is a nonnegative integer and $\boldsymbol{\tau} = (\tau_i)_{i=1}^{n+d+1}$ is a $d+1$ regular knot vector. We 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ defined by

$$\operatorname{dist}_{\infty,[a,b]}(f, \mathbb{S}_{d,\boldsymbol{\tau}}) = \inf_{g \in \mathbb{S}_{d,\boldsymbol{\tau}}} ||f - g||_{\infty,[a,b]}.$$

We want to show the following.

Theorem 9.2. *Let the polynomial degree d and the function f in $C^{d+1}_{\boldsymbol{\Delta}}[a,b]$ be given. Then for any spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$*

$$\operatorname{dist}_{\infty,[a,b]}(f, \mathbb{S}_{d,\boldsymbol{\tau}}) \leq K_d h^{d+1} ||D^{d+1} f||_{\infty,[a,b]}, \tag{9.2}$$

where the constant K_d depends on d, but not on f, h or $\boldsymbol{\tau}$.

We will prove this theorem by constructing a spline $P_d f$ such that

$$|f(x) - P_d f(x)| \leq K_d h^{d+1} ||D^{d+1} f||_{\infty,[a,b]}, \quad x \in [a,b] \tag{9.3}$$

for a constant K_d depending only on d. The approximation $P_d f$ will be 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 ith 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. These kinds of approximation methods are called *quasi-interpolants*.

9.2.1 The constant and linear cases

We first prove Theorem 9.2 in the low degree cases $d = 0$ and $d = 1$. For $d = 0$ the knots form a partition $a = \tau_1 < \cdots < \tau_{n+1} = b$ of $[a, b]$ and the B-spline $B_{i,0}$ is the characteristic function of the interval $[\tau_i, \tau_{i+1})$ for $i = 1, \ldots, n-1$, while $B_{n,0}$ is the characteristic function of the closed interval $[\tau_n, \tau_{n+1}]$. We consider the step function

$$g = P_0 f = \sum_{i=1}^{n} f(\tau_{i+1/2}) B_{i,0},$$

where $\tau_{i+1/2} = (\tau_i + \tau_{i+1})/2$. Fix $x \in [a, b]$ and let l be an integer such that $\tau_l \leq x < \tau_{l+1}$. We then have

$$f(x) - P_0 f(x) = f(x) - f(\tau_{l+1/2}) = \int_{\tau_{l+1/2}}^{x} Df(y)dy$$

so

$$|f(x) - P_0 f(x)| \leq |x - \tau_{l+1/2}| \; ||Df||_{\infty,[\tau_l,\tau_{l+1}]} \leq \frac{h}{2} ||Df||_{\infty,[a,b]}.$$

In this way we obtain (9.2) with $K_0 = 1/2$.

In the linear case $d = 1$ we define $P_1 f$ to be the piecewise linear interpolant to f on $\boldsymbol{\tau}$

$$g = P_1 f = \sum_{i=1}^{n} f(\tau_{i+1}) B_{i,1}.$$

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 $K_1 = 1/8$.

9.2.2 The quadratic case

Consider next the quadratic case $d = 2$. We shall approximate f by the quasi-interpolant $P_2 f$ that we constructed in Section 8.2.2. Its properties is summarised in the following lemma.

Lemma 9.3. *Suppose $\boldsymbol{\tau} = (\tau_i)_{i=1}^{n+3}$ is a knot vector with $\tau_{i+3} > \tau_i$ for $i = 1, \ldots, n$. The operator*

$$P_2 f = \sum_{i=1}^{n} \lambda_i(f) B_{i,2,\boldsymbol{\tau}}, \quad \textit{with} \quad \lambda_i(f) = -\frac{1}{2} f(\tau_{i+1}) + 2f(\tau_{i+3/2}) - \frac{1}{2} f(\tau_{i+2}) \tag{9.4}$$

satisfies $P_2 p = p$ for all $p \in \pi_2$.

To show that (9.3) holds for $d = 2$ we now give a sequence of small lemmas.

Lemma 9.4. *Let $P_2(f)$ be as in (9.4). Then*

$$|\lambda_i(f)| \le 3||f||_{\infty,[\tau_{i+1},\tau_{i+2}]}, \quad i = 1, \ldots, n. \tag{9.5}$$

Proof. Fix an integer i. Then

$$|\lambda_i(f)| = |-\frac{1}{2} f(\tau_{i+1}) + 2f(\tau_{i+3/2}) - \frac{1}{2} f(\tau_{i+2})| \le (\frac{1}{2} + 2 + \frac{1}{2})||f||_{\infty,[\tau_{i+1},\tau_{i+2}]}$$

from which the result follows. ∎

Lemma 9.5. *For $\ell = 3, \ldots, n$ we can bound $P_2 f$ on a subinterval $[\tau_\ell, \tau_{\ell+1}]$ by*

$$||P_2 f||_{\infty,[\tau_\ell,\tau_{\ell+1}]} \le 3||f||_{\infty,[\tau_{\ell-1},\tau_{\ell+2}]}. \tag{9.6}$$

Proof. Fix $x \in [\tau_\ell, \tau_{\ell+1}]$. Since the B-splines are nonnegative and form a partition of unity we have

$$\begin{aligned} |P_2 f(x)| &= \Big| \sum_{i=\ell-2}^{\ell} \lambda_i(f) B_{i,2,\boldsymbol{\tau}}(x) \Big| \le \max_{\ell-2 \le i \le \ell} |\lambda_i(f)| \\ &\le 3 \max_{\ell-2 \le i \le \ell} ||f||_{\infty,[\tau_{i+1},\tau_{i+2}]} = 3||f||_{\infty,[\tau_{\ell-1},\tau_{\ell+2}]}, \end{aligned}$$

where we used Lemma 9.4. This completes the proof. ∎

The following lemma shows that locally, the spline $P_2 f$ approximates f essentially as well as the best quadratic polynomial.

Lemma 9.6. *For $\ell = 3, \ldots, n$ the error $f - P_2 f$ on the interval $[\tau_\ell, \tau_{\ell+1}]$ is bounded by*

$$||f - P_2 f||_{\infty,[\tau_\ell,\tau_{\ell+1}]} \le 4 \operatorname{dist}_{\infty,[\tau_{\ell-1},\tau_{\ell+2}]}(f, \pi_2). \tag{9.7}$$

Proof. Let $p \in \pi_2$ be any quadratic polynomial. Since $P_2 p = p$ and P_2 is a linear operator, application of (9.6) to $f - p$ yields

$$\begin{aligned} \big|f(x) - (P_2 f)(x)\big| &= \big|f(x) - p(x) - \big((P_2 f)(x) - p(x)\big)\big| \\ &\le \big|f(x) - p(x)\big| + \big|P_2(f - p)(x)\big| \\ &\le (1 + 3)||f - p||_{\infty,[\tau_{\ell-1},\tau_{\ell+2}]}. \end{aligned} \tag{9.8}$$

Since p is arbitrary we obtain (9.7). ∎

We can now prove (9.2) for $d = 2$. For any interval $[a, b]$ Theorem 9.1 with $d = 2$ gives

$$\operatorname{dist}_{\infty,[a,b]}(f, \pi_2) \le K_2 h^3 ||D^3 f||_{\infty,[a,b]},$$

where $h = b - a$ and $K_2 = 1/(2^3 3!)$. Combining this estimate on $[a, b] = [\tau_{\ell-1}, \tau_{\ell+2}]$ with (9.7) we obtain (9.3) and hence (9.2).

9.2.3 The general case

The general case is analogous to the quadratic case, but the details are more complicated. Recall that for $d = 2$ we picked three points $x_{i,k} = \tau_{i+1} + k(\tau_{i+2} - \tau_{i+1})/2$ for $k = 0, 1, 2$ in each subinterval $[\tau_{i+1}, \tau_{i+2}]$ and then chose constants $w_{i,k}$ for $k = 0$, 1, 2 such that the operator

$$P_2 f = \sum_{i=1}^{n} \lambda_i(f) B_{i,2,\boldsymbol{\tau}}, \quad \text{with} \quad \lambda_i(f) = w_{i,0} f(x_{i,0}) + w_{i,1} f(x_{i,1}) + w_{i,2} f(x_{i,2}),$$

reproduced quadratic polynomials. We will follow the same strategy for general degree. The resulting quasi-interpolant is a special case of the one given in Theorem 8.7.

Suppose that $d \geq 2$ and fix an integer i such that $\tau_{i+d} > \tau_{i+1}$. We pick the largest subinterval $[a_i, b_i] = [\tau_l, \tau_{l+1}]$ of $[\tau_{i+1}, \tau_{i+d}]$ and define the uniformly spaced points

$$x_{i,k} = a_i + \frac{k}{d}(b_i - a_i), \quad \text{for } k = 0,\ 1,\ \ldots,\ d \tag{9.9}$$

in this interval. Given $f \in C_{\boldsymbol{\Delta}}[a, b]$ we define $P_d f \in \mathbb{S}_{d,\boldsymbol{\tau}}$ by

$$P_d f(x) = \sum_{i=1}^{n} \lambda_i(f) B_{i,d}(x), \quad \text{where} \quad \lambda_i(f) = \sum_{k=0}^{d} w_{i,k} f(x_{i,k}). \tag{9.10}$$

The following lemma shows how the coefficients $(w_{i,k})_{k=0}^{d}$ should be chosen so that $P_d p = p$ for all $p \in \pi_d$.

Lemma 9.7. *Suppose that in* (9.10) *the functionals λ_i are given by $\lambda_i(f) = f(\tau_{i+1})$ if $\tau_{i+d} = \tau_{i+1}$, while if $\tau_{i+d} > \tau_{i+1}$ we set*

$$w_{i,k} = \gamma_i(p_{i,k}), \quad k = 0, 1, \ldots, d, \tag{9.11}$$

where $\gamma_i(p_{i,k})$ is the ith B-spline coefficient of the polynomial

$$p_{i,k}(x) = \prod_{\substack{j=0 \\ j \neq k}}^{d} \frac{x - x_{i,j}}{x_{i,k} - x_{i,j}}. \tag{9.12}$$

Then the operator P_d in (9.10) *satisfies $P_d p = p$ for all $p \in \pi_d$.*

Proof. Suppose first that $t_{i+d} > t_{i+1}$. Any $p \in \pi_d$ can be written in the form

$$p(x) = \sum_{k=0}^{d} p(x_{i,k}) p_{i,k}(x). \tag{9.13}$$

For if we denote the function on the right by $q(x)$ then $q(x_{i,k}) = p(x_{i,k})$ for $k = 0$, 1, …, d, and since $q \in \pi_d$ it follows by the uniqueness of the interpolating polynomial that $p = q$. Now, by linearity of γ_i we have

$$\begin{aligned} \lambda_i(p) &= \sum_{k=0}^{d} w_{i,k} p(x_{i,k}) = \sum_{k=0}^{d} \gamma_i(p_{i,k}) p(x_{i,k}) \\ &= \gamma_i\bigl(\sum_{k=0}^{d} p_{i,k} p(x_{i,k})\bigr) = \gamma_i(p). \end{aligned}$$

If $t_{i+1} = t_{i+d}$ we know that a spline of degree d with knots $\boldsymbol{t}$ agrees with its $i+1$st coefficient at t_{i+1}. In particular, for any polynomial p we have $\lambda_i(p) = f(t_{i+1}) = \gamma_i(p)$. Alltogether this means that

$$P_d(p) = \sum_{i=1}^{n} \lambda_i(p) B_{i,d}(x) = \sum_{i=1}^{n} \gamma_i(p) B_{i,d}(x) = p$$

which confirms the lemma. ∎

The B-spline coefficients of $p_{i,k}$ can be found from the following lemma.

Lemma 9.8. *Given a spline space $\mathbb{S}_{d,\boldsymbol{\tau}}$ and numbers $v_1, \ldots, v_d$. The ith B-spline coefficient of the polynomial $p(x) = (x - v_1) \ldots (x - v_d)$ can be written*

$$\gamma_i(p) = \frac{1}{d!} \sum_{(j_1,\ldots,j_d)\in\Pi_d} (t_{i+j_1} - v_1) \cdots (t_{i+j_d} - v_d), \tag{9.14}$$

where Π_d is the set of all permutations of the integers $1, 2, \ldots, d$.

Proof. By Theorem 4.16 we have

$$\gamma_i(p) = \mathcal{B}[p](\tau_{i+1}, \ldots, \tau_{i+d}),$$

where $\mathcal{B}[p]$ is the blossom of p. It therefore suffices to verify that the expression (9.14) for $\gamma_i(p)$ satisfies the three properties of the blossom, but this is immediate. ∎

As an example, for $d = 2$ the set of all permutations of $1, 2$ are $\Pi_2 = \{(1,2), (2,1)\}$ and therefore

$$\gamma_i\big((x - v_1)(x - v_2)\big) = \frac{1}{2}\bigg((\tau_{i+1} - v_1)(\tau_{i+2} - v_2) + (\tau_{i+2} - v_1)(\tau_{i+1} - v_2)\bigg).$$

We can now give 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*

$$|\lambda_i(f)| \le K_d ||f||_{\infty,[\tau_{i+1},\tau_{i+d}]}, \quad i = 1, \ldots, n, \tag{9.15}$$

where

$$K_d = \frac{2^d}{d!}[d(d-1)]^d \tag{9.16}$$

depends only on d.

Proof. Fix an integer i. From Lemma 9.8 we have

$$w_{i,k} = \sum_{(j_1,\ldots,j_d)\in\Pi_d} \prod_{r=1}^{d} \left(\frac{\tau_{i+j_r} - v_r}{x_{i,k} - v_r}\right)/d!, \tag{9.17}$$

where $(v_r)_{r=1}^d = (x_{i,0}, \ldots, x_{i,k-1}, x_{i,k+1}, \ldots, x_{i,d})$. and Π_d denotes the set of all permutations of the integers 1, 2, …, 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

$$\prod_{r=1}^{d} (\tau_{i+j_r} - v_r) \le (\tau_{i+d} - \tau_{i+1})^d.$$

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

$$\prod_{r=1}^{d} |x_{i,k} - v_r| = \prod_{\substack{q=0\\q\neq k}}^{d} \frac{|k-q|}{d}(b_i - a_i) \geq k!(d-k)! \left(\frac{b_i - a_i}{d}\right)^d \geq k!(d-k)! \left(\frac{\tau_{i+d} - \tau_{i+1}}{d(d-1)}\right)^d$$

for all values of k and r since $[a_i, b_i]$ is the largest subinterval of $[\tau_{i+1}, \tau_{i+d}]$. Since the sum in (9.17) contains $d!$ terms, we find

$$\sum_{k=0}^{d} |w_{i,k}| \leq \frac{[d(d-1)]^d}{d!} \sum_{k=0}^{d} \binom{d}{k} = \frac{2^d}{d!}[d(d-1)]^d = K_d$$

and hence

$$|\lambda_i(f)| \leq ||f||_{\infty,[\tau_{i+1},\tau_{i+d}]} \sum_{k=0}^{d} |w_{i,k}| \leq K_d ||f||_{\infty,[\tau_{i+1},\tau_{i+d}]} \tag{9.18}$$

which is the required inequality. ∎

From the bound for $\lambda_i(f)$ we easily obtain a bound for the norm of $P_d f$.

Theorem 9.10. *For $d+1 \leq l \leq n$ and $f \in C_{\boldsymbol{\Delta}}[a,b]$ we have the bound*

$$||P_d f||_{\infty,[\tau_l,\tau_{l+1}]} \leq K_d ||f||_{\infty,[\tau_{l-d+1},\tau_{l+d}]}, \tag{9.19}$$

where K_d is the constant in Theorem 9.9.

Proof. Fix $x \in [\tau_l, \tau_{l+1}]$. Since the B-splines are nonnegative and form a partition of unity we have by Theorem 9.9

$$\begin{aligned} |P_d f(x)| = |\sum_{i=l-d}^{l} \lambda_i(f) B_{i,d,\boldsymbol{\tau}}(x)| &\leq \max_{l-d \leq i \leq l} |\lambda_i(f)| \\ &\leq K_d \max_{l-d \leq i \leq l} ||f||_{\infty,[\tau_{i+1},\tau_{i+d}]} = K_d ||f||_{\infty,[\tau_{l-d+1},\tau_{l+d}]} \end{aligned}$$

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 9.11. *For $l = d+1, \ldots, n$ the error $f - P_d f$ on the interval $[\tau_l, \tau_{l+1}]$ is bounded by*

$$||f - P_d f||_{\infty,[\tau_l,\tau_{l+1}]} \leq (1 + K_d)\,\mathrm{dist}_{\infty,[\tau_{l-d+1},\tau_{l+d}]}(f, \pi_d), \tag{9.20}$$

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 therefore have

$$\begin{aligned} \big|f(x) - (P_d f)(x)\big| &= \big|f(x) - p(x) - \big((P_d f)(x) - p(x)\big)\big| \\ &\leq \big|f(x) - p(x)\big| + \big|P_d(f-p)(x)\big| \\ &\leq (1 + K_d)||f - p||_{\infty,[\tau_{l-d+1},\tau_{l+d}]}. \end{aligned}$$

Since p is arbitrary we obtain (9.20). ∎

We can now prove (9.2) for general d. By Theorem 9.1 we have for any interval $[a,b]$

$$\mathrm{dist}_{\infty,[a,b]}(f,\pi_d) \le K_d h^{d+1} ||D^{d+1} f||_{\infty,[a,b]},$$

where $h = b-a$ and K_d only depends on d. Combining this estimate on $[a,b] = [\tau_{l-d+1}, \tau_{l+d}]$ with (9.20) we obtain (9.3) and hence (9.2).

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 then 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*. In this section we will study the relationship between a spline and its coefficients quantitatively by introducing the *condition number* of a basis.

We have already seen that the size of a spline is bounded by its B-spline coefficients. There is also a reverse inequality, i.e., a bound on the B-spline coefficients in terms of the size of f. There are several reasons why such inequalities are important. In Section 4.1 we made use of this fact to estimate how fast the control polygon converges to the spline as more and more knots are inserted. A more direct consequence is that small relative perturbations in the coefficients can only lead to small changes in the function values. Both properties reflect the fact that the B-spline basis is well conditioned.

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 9.12. *Let $\mathbb{U}$ be a normed linear space. A basis (ϕ_j) for $\mathbb{U}$ 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*

$$C_1^{-1}\big\|(c_j)\big\| \le \Big\|\sum_j c_j\phi_j\Big\| \le C_2\big\|(c_j)\big\|, \tag{9.21}$$

for all sets of coefficients $\boldsymbol{c} = (c_j)$. Let C_1^* and C_2^* denote the smallest possible values of C_1 and C_2 such that (9.21) *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 $\mathbb{U}$ and the vector norm of the coefficients. In our case $\mathbb{U}$ will of course be some spline space $\mathbb{S}_{d,\boldsymbol{t}}$ and the basis (ϕ_j) will be the B-spline basis. The norms we will consider are the p-norms which are defined by

$$||f||_p = ||f||_{p,[a,b]} = \left(\int_a^b |f(x)|^p dx\right)^{1/p}, \quad \text{and} \quad ||\boldsymbol{c}||_p = \left(\sum_j |c_j|^p\right)^{1/p}$$

where f is a function on the interval $[a,b]$ and $\boldsymbol{c} = (c_j)$ is a real vector, and p is a real number in the range $1 \le p < \infty$ for any real number. For $p = \infty$ the norms are defined by

$$||f||_\infty = ||f||_{\infty,[a,b]} = \max_{a\le x\le b} |f(x)|, \quad \text{and} \quad ||\boldsymbol{c}||_\infty = \big\|(c_j)\big\|_\infty = \max_j |c_j|,$$

In practice, the most important norms are the 1-, 2- and ∞-norms.

In Definition 9.12 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 $\mathbb{U}$, 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 (9.21).

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 9.13. *Suppose (ϕ_j) is a stable basis for $\mathbb{U}$. If $f = \sum_j c_j\phi_j$ and $g = \sum_j b_j\phi_j$ are two elements in $\mathbb{U}$ with $f \neq 0$, then*

$$\frac{||f - g||}{||f||} \leq \kappa \frac{||\boldsymbol{c} - \boldsymbol{b}||}{||\boldsymbol{c}||}, \tag{9.22}$$

where κ is the condition number of the basis as in Definition 9.12.

Proof. From (9.21), 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 (9.22) 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, then 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 a computer. This round-off error means that the computed

spline, here g, will differ from the exact f. Proposition 9.13 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 9.13 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, …, 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 The condition number of the B-spline basis. Infinity norm

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. We will now prove that the condition number of the B-spline basis can be bounded independently of the knot vector so it cannot grow beyond all bounds when the knots vary.

The best constant C_2^* in Definition 9.12 can be found quite easily for the B-spline basis.

Lemma 9.14. *In all spline spaces $\mathbb{S}_{d,\boldsymbol{t}}$ the bound*

$$\Big\|\sum_{i=1}^{m} b_i B_{i,d}\Big\|_{\infty,[t_1,t_{m+1+d}]} \leq ||\boldsymbol{b}||_\infty$$

holds. Equality holds if $b_i = 1$ for all i and the knot vector $\boldsymbol{t} = (t_i)_{i=0}^{n+d}$ is $d+1$-extended; in this case $C_2^* = 1$.

Proof. This follows since the B-splines are nonnegative and sum to one. ∎

To find a bound for the constant C_1 we shall use the operator P_d given by (9.3). We recall that P_d reproduces polynomials of degree d, i.e., $P_d p = p$ for all $p \in \pi_d$. We now show that more is true; we have in fact that P_d reproduces all splines in $\mathbb{S}_{d,\boldsymbol{\tau}}$.

Theorem 9.15. *The operator*

$$P_d f = \sum_{i=1}^{n} \lambda_i(f) B_{i,d}$$

given by (9.3) *reproduces all splines in $\mathbb{S}_{d,\boldsymbol{\tau}}$, $P_d f = f$ for all $f \in \mathbb{S}_{d,\boldsymbol{\tau}}$.*

Proof. We first show that

$$\lambda_j(B_{k,d}) = \delta_{j,k}, \qquad \text{for } j,k = 1,\, \ldots,\, n. \tag{9.23}$$

Fix i and let

$$I_i = [a_i, b_j] = [\tau_{l_i}, \tau_{l_i+1}]$$

be the interval used to define $\lambda_i(f)$. We consider the polynomials

$$\phi_k = B_{k,d}|_{I_i} \qquad \text{for } l_i - d \leq k \leq l_i$$

obtained by restricting the B-splines $\{B_{k,d}\}_{k=l_i-d}^{l_i}$ to the interval I_i. Since P_d reproduces π_d we have

$$\phi_k(x) = (P_d\phi_k)(x) = \sum_{j=l_i-d}^{l_i} \lambda_j(\phi_k)\phi_j(x)$$

for x in the interval I_i. By the linear independence of the the polynomials (ϕ_k) we therefore obtain

$$\lambda_j(B_{k,d}) = \lambda_j(\phi_k) = \delta_{j,k}, \qquad \text{for } j,k = l_i - d,\, \ldots,\, l_i.$$

In particular we have $\lambda_i B_{i,d} = 1$ since $l_i - d \leq i \leq l_i$. For $k < l_i - d$ or $k > l_i$ the support of $B_{k,d}$ has empty intersection with I_i so $\lambda_i(B_{k,d}) = 0$ for these values of k. Thus (9.23) holds for all k.

To complete the proof suppose $f = \sum_{k=1}^{n} c_k B_{k,d}$ is a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$. From (9.23) we then have

$$Qf = \sum_{j=1}^{n}\Big(\sum_{k=1}^{n} c_k \lambda_j(B_{k,d})\Big)B_{j,d} = \sum_{j=1}^{n} c_j B_{j,d} = f. \quad ∎$$

To obtain an upper bound for C_1^* we note that the leftmost inequality in (9.21) is equivalent to

$$|b_i| \leq C_1 ||f||, \quad i = 1, \ldots, m.$$

Lemma 9.16. *There is a constant K_d, depending only on the polynomial degree d, such that for all splines $f = \sum_{i=1}^m b_i B_{i,d}$ in some given spline space $\mathbb{S}_{d,\boldsymbol{t}}$ the inequality*

$$|b_i| \leq K_d ||f||_{[t_{i+1}, t_{i+d}]} \tag{9.24}$$

holds for all i.

Proof. Consider the operator P_d given in Lemma 9.7. Since $P_d f = f$ we have $b_i = \lambda_i(f)$. The result now follows from (9.15) ∎

Note that if $[a, b] \subseteq [c, d]$, then $||f||_{\infty,[a,b]} \leq ||f||_{\infty,[c,d]}$. From (9.24), we therefore conclude that $|b_i| \leq K_d \|f\|_{\infty,[t_1,t_{m+1+d}]}$ for all i or briefly $||\boldsymbol{b}|| \leq K_d \|f\|$. The constant K_d can therefore be used as C_1 in Definition 9.12 in the case where the norm is the ∞-norm. Combining the two lemmas we obtain the following theorem.

Theorem 9.17. *There is a constant K_1, depending only on the polynomial degree d, such that for all spline spaces $\mathbb{S}_{d,\boldsymbol{t}}$ and all splines $f = \sum_{i=1}^m b_i B_{i,d} \in \mathbb{S}_{d,\boldsymbol{t}}$ with B-spline coefficients $\boldsymbol{b} = (b_i)_{i=1}^m$ the inequalities*

$$K_1^{-1} ||\boldsymbol{b}||_\infty \leq ||f||_{\infty,[t_1,t_{m+d}]} \leq ||\boldsymbol{b}||_\infty \tag{9.25}$$

hold.

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 9.17 shows that $\kappa_{d,\infty}$ is bounded above by K_1.

The estimate K_d for C_1^* given by (9.16) is a number which grows quite rapidly with d and does not indicate that the B-spline basis is stable . However, it is possible to find better estimates for the condition number, and it is known that the B-spline basis is very stable , at least for moderate values of d. To determine the condition number is relatively simple 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 numerically to be $\kappa_{3,\infty} \approx 5.5680$ and $\kappa_{4,\infty} \approx 12.088$.

9.3.3 The condition number of the B-spline basis. p-norm

With $1 \leq p \leq \infty$ and q such that $1/p + 1/q = 1$ we recall the Hölder inequality for functions

$$\int_a^b |f(x)g(x)|dx \leq ||f||_p ||g||_q,$$

and the Hölder inequality for sums

$$\sum_{i=1}^m |b_i c_i| \leq ||(b_i)_{i=1}^m||_p ||(c_i)_{i=1}^m||_q.$$

We also note that for any polynomial $g \in \pi_d$ and any interval $[a, b]$ we have

$$|g(x)| \leq \frac{C}{b-a} \int_a^b |g(x)|\, dx, \quad x \in [a, b], \tag{9.26}$$

where the constant C only depends on the degree d. This follows on $[a, b] = [0, 1]$ since all norms on a finite dimensional vector space are equivalent, and then on an arbitrary interval $[a, b]$ by a change of variable.

In order to generalise the stability result (9.25) to arbitrary p-norms we need to scale the B-splines differently. We define the p-norm B-splines to be identically zero if $\tau_{i+d+1} = \tau_i$ and

$$B^p_{i,d,\boldsymbol{t}} = \left(\frac{d+1}{\tau_{i+d+1} - \tau_i}\right)^{1/p} B_{i,d,\boldsymbol{t}}, \tag{9.27}$$

otherwise.

Theorem 9.18. *There is a constant K, depending only on the polynomial degree d, such that for all $1 \leq p \leq \infty$, all spline spaces $\mathbb{S}_{d,\boldsymbol{t}}$ and all splines $f = \sum_{i=1}^m b_i B^p_{i,d} \in \mathbb{S}_{d,\boldsymbol{t}}$ with p-norm B-spline coefficients $\boldsymbol{b} = (b_i)_{i=1}^m$ the inequalities*

$$K^{-1}||\boldsymbol{b}||_p \leq ||f||_{p,[\tau_1, \tau_{m+d}]} \leq ||\boldsymbol{b}||_p \tag{9.28}$$

hold.

Proof. We first prove the upper inequality. Let $\gamma_i = (d+1)/(\tau_{i+d+1} - \tau_i)$ for $i = 1, \ldots, m$ and set $[a, b] = [\tau_1, \tau_{m+d+1}]$. Using the Hölder inequality for sums we have

$$\sum_i |b_i B^p_{i,d}| = \sum_i |b_i \gamma_i^{1/p} B^{1/p}_{i,d}| B^{1/q}_{i,d} \leq \Big(\sum_i |b_i|^p \gamma_i B_{i,d}\Big)^{1/p} \Big(\sum_i B_{i,d}\Big)^{1/q}.$$

Raising this to the pth power and using the partition of unity property we obtain the inequality

$$\big|\sum_i b_i B^p_{i,d}(x)\big|^p \leq \sum_i |b_i|^p \gamma_i B_{i,d}(x), \quad x \in \mathbb{R}.$$

Therefore, recalling that $\int B_{i,d}(x) dx = 1/\gamma_i$ we find

$$||f||^p_{p,[a,b]} = \int_a^b \big|\sum_i b_i B^p_{i,d}(x)\big|^p\, dx \leq \sum_i |b_i|^p \gamma_i \int_a^b B_{i,d}(x)\, dx = \sum_i |b_i|^p.$$

Taking pth roots proves the upper inequality.

Consider now the lower inequality. Recall from (9.24) that we can bound the B-spline coefficients in terms of the infinity norm of the function. In terms of the coefficients b_i of the p-norm B-splines we obtain from (9.24) for all i

$$\left(\frac{d+1}{\tau_{i+d+1} - \tau_i}\right)^{1/p} |b_i| \leq K_1 \max_{\tau_{i+1} \leq x \leq \tau_{i+d}} |f(x)|,$$

where the constant K_1 only depends on d. Taking max over a larger subinterval, using (9.26), and then Hölder for integrals we find

$$\begin{aligned}|b_i| &\le K_1(d+1)^{-1/p}\big(\tau_{i+d+1}-\tau_i\big)^{1/p}|\max_{\tau_i\le x\le\tau_{i+d+1}}|f(x)| \\ &\le CK_1(d+1)^{-1/p}\big(\tau_{i+d+1}-\tau_i\big)^{-1+1/p}\int_{\tau_i}^{\tau_{i+d+1}}|f(y)|\,dy \\ &\le CK_1(d+1)^{-1/p}\Big(\int_{\tau_i}^{\tau_{i+d+1}}|f(y)|^p\,dy\Big)^{1/p}\end{aligned}$$

Raising both sides to the pth power and summing over i we obtain

$$\sum_i |b_i|^p \le C^pK_1^p(d+1)^{-1}\sum_i\int_{\tau_i}^{\tau_{i+d+1}}|f(y)|^p\,dy \le C^pK_1^p||f||^p_{p,[a,b]}.$$

Taking pth roots we obtain the lower inequality in (9.28) with $K = CK_1$. ∎

Exercises for Chapter 9

9.1 In this exercise we will study the order of approximation by the Schoenberg Variation Diminishing Spline Approximation of degree $d\ge 2$. This approximation is given by

$$V_df=\sum_{i=1}^n f(\tau_i^*)B_{i,d},\quad \text{with}\quad \tau_i^*=\frac{\tau_{i+1}+\cdots\tau_{i+d}}{d}.$$

Here $B_{i,d}$ is the ith 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 $i=2,\ldots,n$. Moreover we define the quantities

$$a=\tau_1,\quad b=\tau_{n+d+1},\quad h=\max_{1\le i\le n}\tau_{i+1}-\tau_i.$$

We want to show that V_df is an $O(h^2)$ approximation to a sufficiently smooth f.

We first consider the more general spline approximation

$$\tilde{V}_df=\sum_{i=1}^n\lambda_i(f)B_{i,d},\quad\text{with}\quad\lambda_i(f)=w_{i,0}f(x_{i,0})+w_{i,1}f(x_{i,1}).$$

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, $i=1,\ldots,n$.

Before attempting to solve this exercise the reader might find it helpful to review Section 9.2.2

a) Suppose for $i=1,\ldots,n$ 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 $\tilde{V}_dp=p$ 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 $i = 1, \ldots, n$, i.e. we consider $V_d f$. We define the usual uniform norm on an interval $[c, d]$ by

$$||f||_{[c,d]} = \sup_{c \le x \le d} |f(x)|, \quad f \in C_{\mathbf{\Delta}}[c, d].$$

c) Show that for $d + 1 \le l \le n$

$$||V_d f||_{[\tau_l, \tau_{l+1}]} \le ||f||_{[\tau^*_{l-d}, \tau^*_l]}, \quad f \in C_{\mathbf{\Delta}}[a, b].$$

d) Show that for $f \in C_{\mathbf{\Delta}}[\tau^*_{l-d}, \tau^*_l]$ and $d + 1 \le l \le n$

$$||f - V_d f||_{[\tau_l, \tau_{l+1}]} \le 2 \operatorname{dist}_{[\tau^*_{l-d}, \tau^*_l]}(f, \pi_1).$$

e) Explain why the following holds for $d + 1 \le l \le n$

$$\operatorname{dist}_{[\tau^*_{l-d}, \tau^*_l]}(f, \pi_1) \le \frac{(\tau^*_l - \tau^*_{l-d})^2}{8} ||D^2 f||_{[\tau^*_{l-d}, \tau^*_l]}.$$

f) Show that the following $O(h^2)$ estimate holds

$$||f - V_d f||_{[a,b]} \le \frac{d^2}{4} h^2 ||D^2 f||_{[a,b]}.$$

(Hint: Verify that $\tau^*_l - \tau^*_{l-d} \le hd$.)

9.2 In this exercise we want to perform a numerical simulation experiment to determine the order of approximation by the quadratic spline approximations

$$V_2 f = \sum_{i=1}^{n} f(\tau_i^*) B_{i,2}, \quad \text{with} \quad \tau_i^* = \frac{\tau_{i+1} + \tau_{i+2}}{2},$$
$$P_2 f = \sum_{i=1}^{n} \big(-\frac{1}{2} f(\tau_{i+1}) + 2 f(\tau_i^*) - \frac{1}{2} f(\tau_{i+2}) \big) B_{i,2}.$$

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 $m \ge 0$ 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

$$V_2^m f = \sum_{i=1}^{n} f(\tau^m_{i+3/2}) B^m_{i,2}, \quad \text{with} \quad \tau_i^m = \frac{\tau^m_{i+1} + \tau^m_{i+2}}{2},$$
$$P_2^m f = \sum_{i=1}^{n} \big(-\frac{1}{2} f(\tau^m_{i+1}) + 2 f(\tau^m_{i+3/2}) - \frac{1}{2} f(\tau^m_{i+2}) \big) B^m_{i,2},$$

and $B^m_{i,2}$ is the ith quadratic B-spline on $\boldsymbol{\tau}^m$. As approximations to the norms $||f - V^m_2 f||_{[0,\pi]}$ and $||f - P^m_2 f||_{[0,\pi]}$ we use

$$E^m_V = \max_{0\leq j\leq 100} |f(j\pi/100) - V^m_2 f(j\pi/100)|,$$
$$E^m_P = \max_{0\leq j\leq 100} |f(j\pi/100) - P^m_2 f(j\pi/100)|.$$

Write a computer program to compute numerically the values of E^m_V and E^m_P for $m = 0, 1, 2, 3, 4, 5$, and the ratios E^m_V/E^{m-1}_V and E^m_P/E^{m-1}_P for $1 \leq m \leq 5$. What can you deduce about the approximation order of the two methods?

Make plots of $V^m_2 f$, $P^m_2 f$, $f - V^m_2 f$, and $f - P^m_2 f$ for some values of m.

9.3 Suppose we have $m \geq 3$ data points $\big(x_i, f(x_i)\big)^m_{i=1}$ sampled from a function f, where the abscissas $\boldsymbol{x} = (x_i)^m_{i=1}$ satisfy $x_1 < \cdots < x_m$. 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 $\boldsymbol{x}$ we form a 3-regular knot vector by using every second data point as a knot

$$\boldsymbol{\tau} = (\tau_j)^{n+3}_{j=1} = (x_1, x_1, x_1, x_3, x_5, \ldots, x_{m-2}, x_m, x_m, x_m), \tag{9.29}$$

where $n = (m + 3)/2$. In the quadratic spline space $\mathbb{S}_{2,\boldsymbol{\tau}}$ we can then construct the spline

$$Q_2 f = \sum_{j=1}^{n} \lambda_j(f) B_{j,2}, \tag{9.30}$$

where the B-spline coefficients $\lambda_j(f)^n_{j=1}$ are defined by the rule

$$\lambda_j(f) = \frac{1}{2}\Big(- \theta^{-1}_j f(x_{2j-3}) + \theta^{-1}_j (1 + \theta_j)^2 f(x_{2j-2}) - \theta_j f(x_{2j-1})\Big), \tag{9.31}$$

for $j = 1, \ldots, n$. Here $\theta_1 = \theta_n = 1$ and

$$\theta_j = \frac{x_{2j-2} - x_{2j-3}}{x_{2j-1} - x_{2j-2}}$$

for $j = 2, \ldots, n - 1$.

a) Show that Q_2 simplifies to P_2 given by (9.4) 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 $j = 1, \ldots, n$ and $f \in C_{\boldsymbol{\Delta}}[x_1, x_m]$

$$|\lambda_j(f)| \leq (2\theta + 1)||f||_{\infty,[\tau_{j+1},\tau_{j+2}]},$$

where

$$\theta = \max_{1\leq j\leq n} \{\theta^{-1}_j, \theta_j\}.$$

d) Show that for $l = 3, \ldots, n$, $f \in C_{\mathbf{\Delta}}[x_1, x_m]$, 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 $l = 3, \ldots, n$ and $f \in C_{\mathbf{\Delta}}[x_1, x_m]$

$$||f - Q_2 f||_{\infty,[\tau_l,\tau_{l+1}]} \leq (2\theta + 2)\,\mathrm{dist}_{[\tau_{l-1},\tau_{l+2}]}(f, \pi_2).$$

f) Show that for $f \in C^3_{\mathbf{\Delta}}[x_1, x_m]$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ is said to be* connected *if for each x in (τ_1, τ_{n+d+1}) there is some j such that $\tau_j < x < \tau_{j+d+1}$ 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 $\tau_\mu < x < \tau_{\mu+1}$, then $\tau_j < x < \tau_{j+d+1}$ for $j = \mu - d, \ldots, \mu$ (assuming the knot vector is long enough) so we must have $c_{\mu-d} = \cdots = c_\mu = 0$. In other words f must be identically zero on $(\tau_\mu, \tau_{\mu+1})$. In this case f splits into two spline functions f_1 and f_2 with knot vectors $\boldsymbol{\tau}^1 = (\tau_j)_{j=1}^{\mu}$ and

$\boldsymbol{\tau}^2 = (\tau_j)_{j=\mu+1}^{n+d+1}$. We clearly have

$$f_1 = \sum_{j=1}^{\mu-d-1} c_j B_{j,d}, \qquad f_2 = \sum_{j=\mu+1}^{n} c_j B_{j,d}.$$

(ii) The splitting point x is a knot of multiplicity m, say

$$\tau_\mu < x = \tau_{\mu+1} = \cdots = \tau_{\mu+m} < \tau_{\mu+m+1}.$$

In this case we have $\tau_j < x < \tau_{j+1+d}$ for $j = \mu + m - d, \ldots, \mu$. 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 $\boldsymbol{\tau}^1 = (\tau_j)_{j=1}^{\mu+m}$ and the knot vector of f_2 is $\boldsymbol{\tau}^2 = (\tau_j)_{j=\mu+1}^{n+d+1}$, while

$$f_1 = \sum_{j=1}^{\mu+m-d-1} c_j B_{j,d}, \qquad f_2 = \sum_{j=\mu+1}^{n} c_j B_{j,d}.$$

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 $\boldsymbol{\tau}$,*

$$\tau_i < z = \tau_{i+1} = \cdots = \tau_{i+m} < \tau_{i+m+1}$$

for some i. Let $f = \sum_j c_j B_{j,d}$ be a spline in $\mathbb{S}_{d,\boldsymbol{\tau}}$. Then

$$c_j = \frac{1}{d!} \sum_{k=0}^{d-m} (-1)^k D^{d-k} \rho_{j,d}(z) D^k f(z) \tag{10.1}$$

for all j such that $\tau_j < z < \tau_{j+d+1}$, where $\rho_{j,d}(y) = (y - \tau_{j+1}) \cdots (y - \tau_{j+d})$.

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 $j = i + m - d, \ldots, i$ so

$$c_j = \lambda_j f = \frac{1}{d!} \sum_{k=0}^{d} (-1)^k D^{d-k} \rho_{j,d}(z) D^k f(z), \tag{10.2}$$

for these values of j. But in this case $\rho_{j,d}(y)$ contains the factor $(y - \tau_{i+1}) \cdots (y - \tau_{i+m}) = (y - z)^m$ so $D^{d-k} \rho_{j,d}(z) = 0$ for $k > d - m$ and $j = i + m - d, \ldots, i$, i.e., for all values of j such that $\tau_j < z < \tau_{j+d+1}$. The formula (10.1) therefore follows from (10.2). ∎

In the situation of Lemma 10.2, we know from Lemma 2.6 that $D^k f$ is continuous at z for $k = 0, \ldots, d-m$, 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 (τ_1, τ_{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 $D^j f(x) = 0$ for $j = 0, \ldots, d-m$. But then we see from Lemma 10.2 that $c_l = \lambda_l f = 0$ for all l such that $\tau_l < x < \tau_{l+d+1}$. 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 $f(z) = Df(z) = \cdots = D^{r-1} f(z) = 0$ 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 $S^-(\boldsymbol{c})$ denotes the number of sign changes in the vector $\boldsymbol{c}$ (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.

$$\begin{aligned}
Z(x) &= 1, & S^-(x) &= 1, & Z\big(x(1-x)^2\big) &= 3, & S^-\big(x(1-x)^2\big) &= 1,\\
Z(x^2) &= 2, & S^-(x^2) &= 0, & Z\big(x^3(1-x)^2\big) &= 5, & S^-\big(x^3(1-x)^2\big) &= 1,\\
Z(x^7) &= 7, & S^-(x^7) &= 1, & Z(-1-x^2+\cos x) &= 2, & S^-(-1-x^2+\cos x) &= 0.
\end{aligned}$$

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ that is connected. Then*

$$Z_{(\tau_1,\tau_{n+d+1})}(f) \leq S^-(\boldsymbol{c}) \leq n-1.$$

Proof. Let $z_1 < z_2 < \cdots < z_\ell$ be the zeros of f in the interval (τ_1, τ_{n+d+1}), each of multiplicity r_i; Lemma 10.2 shows that z_i occurs at most $d - r_i$ times in $\boldsymbol{\tau}$. For if z_i occured $m > d - r_i$ times in $\boldsymbol{\tau}$ then $d - m < r_i$, and hence $c_j = 0$ by (10.1) for all j such that $\tau_j < z < \tau_{j+d+1}$, which means that z is a splitting point for f. But this is impossible since f is connected.

Now we form a new knot vector $\hat{\boldsymbol{\tau}}$ 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 $\boldsymbol{\tau}$ other than z_i in $[z_i - h, z_i + h]$ for $1 \leq i \leq \ell$. Let $\hat{\boldsymbol{c}}$ be the B-spline coefficients of f relative to $\hat{\boldsymbol{\tau}}$. By Lemma 4.24 we then have $S^-(\hat{\boldsymbol{c}}) \leq S^-(\boldsymbol{c})$ so it suffices to prove that $Z_{(\tau_1,\tau_{n+d+1})}(f) \leq S^-(\hat{\boldsymbol{c}})$. But since

$$Z_{(\tau_1,\tau_{n+d+1})}(f) = \sum_{i=1}^{\ell} Z_{(z_i-h,z_i+h)}(f),$$

it suffices to establish the theorem in the following situation: The knot vector is given by

$$\boldsymbol{\tau} = (\overbrace{z-h,\dots,z-h}^{d+1},\overbrace{z,\dots,z}^{d-r},\overbrace{z+h,\dots,z+h}^{d+1})$$

and z is a zero of f of multiplicity r. We want to show that

$$c_j = \frac{(d-r)!}{d!}(-1)^{d+1-j}h^r D^r f(z), \quad j = d+1-r,\dots,d+1, \tag{10.3}$$

so that the $r+1$ coefficients $(c_j)_{j=d+1-r}^{d+1}$ alternate in sign. For then $S^-(\boldsymbol{c}) \geq r = Z_{(z-h,z+h)}(f)$. 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

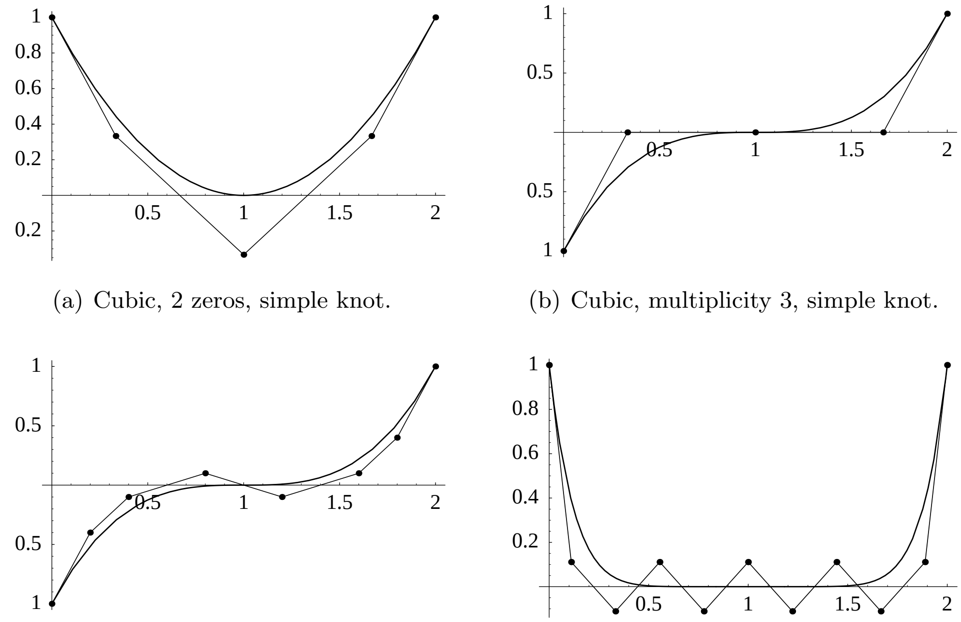
$$D^{d-r}\rho_{j,d}(z) = (d-r)!h^{d+1-j}(-h)^{r-d-1+j} = (d-r)!(-1)^{r-d-1+j}h^r$$

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.



(a) Cubic, 2 zeros, simple knot.

(b) Cubic, multiplicity 3, simple knot.

(c) Degree 5, multiplicity 3, double knot.

(d) Degree 9, multiplicity 8, simple knot.

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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ of dimension n and data $(y_i)_{i=1}^n$ given at n distinct points $x_1 < x_2 < \cdots < x_n$, the aim is to determine a spline $g = \sum_{i=1}^n c_i B_{i,d}$ in $\mathbb{S}_{d,\boldsymbol{\tau}}$ such that

$$g(x_i) = y_i, \qquad \text{for } i = 1, \ldots, n. \tag{10.4}$$

This leads to the linear system of equations

$$\boldsymbol{A}\boldsymbol{c} = \boldsymbol{y},$$

where

$$\boldsymbol{A} = \begin{pmatrix} B_{1,d}(x_1) & B_{2,d}(x_1) & \ldots & B_{n,d}(x_1) \\ B_{1,d}(x_2) & B_{2,d}(x_2) & \ldots & B_{n,d}(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ B_{1,d}(x_n) & B_{2,d}(x_n) & \ldots & B_{n,d}(x_n) \end{pmatrix}, \qquad \boldsymbol{c} = \begin{pmatrix} c_1 \\ c_2 \\ \vdots \\ c_n \end{pmatrix}, \qquad \boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}.$$

The matrix $\boldsymbol{A}$ is often referred to as the *B-spline collocation matrix.* Since $B_{i,d}(x)$ is nonzero only if $\tau_i < x < \tau_{i+d+1}$ (we may allow $\tau_i = x$ if $\tau_i = \tau_{i+d} < \tau_{i+d+1}$), the matrix $\boldsymbol{A}$ will in general be sparse. The following theorem tells us exactly when $\boldsymbol{A}$ is nonsingular.

Theorem 10.6. *Let $\mathbb{S}_{d,\boldsymbol{\tau}}$ be a given spline space, and let $x_1 < x_2 < \cdots < x_n$ be n distinct numbers. The collocation matrix $\boldsymbol{A}$ with entries $\bigl(B_{j,d}(x_i)\bigr)_{i,j=1}^n$ is nonsingular if and only if its diagonal is positive, i.e.,*

$$B_{i,d}(x_i) > 0 \qquad \text{for } i = 1, \ldots, n. \tag{10.5}$$

Proof. We start by showing that $\boldsymbol{A}$ is singular if a diagonal entry is zero. Suppose that $x_q \leq \tau_q$ (strict inequality if $\tau_q = \tau_{q+d} < \tau_{q+d+1}$) 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 $i = 1, \ldots, q$ and $j = q, \ldots, n$. But this means that only the $n - q$ last entries of each of the last $n - q + 1$ columns of $\boldsymbol{A}$ can be nonzero; these columns must therefore be linearly dependent and $\boldsymbol{A}$ must be singular. A similar argument shows that $\boldsymbol{A}$ is also singular if $x_q \geq \tau_{q+d+1}$.

To show the converse, suppose that (10.5) holds but $\boldsymbol{A}$ is singular. Then there is a nonzero vector $\boldsymbol{c}$ such that $\boldsymbol{A}\boldsymbol{c} = 0$. Let $f = \sum_{i=1}^{n} c_i B_{i,d}$ denote the spline with B-spline coefficients $\boldsymbol{c}$. We clearly have $f(x_q) = 0$ for $q = 1, \ldots, n$. Let G denote the set

$$G = \cup_i \big\{(\tau_i, \tau_{i+d+1}) \mid c_i \neq 0\big\}.$$

Since each x in G must be in (τ_i, τ_{i+d+1}) for some i with $c_i \neq 0$, we note that G contains no splitting points of f. Note that if $x_i = \tau_i = \tau_{i+d} < \tau_{i+d+1}$ 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 $\boldsymbol{c}$. Since we also have $S^-(\boldsymbol{c}) < \ell \leq Z_G(f)$, 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 $\boldsymbol{A}$ 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 $\tau_i < x_i < \tau_{i+d+1}$ for $i = 1, \ldots, n$ (note again that if $\tau_i = \tau_{i+d} < \tau_{i+d+1}$, then $x_i = \tau_i$ 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

$$\lambda_{\boldsymbol{x}}(i) = \max_j \{j \mid x_{i-j} = x_i\}$$

and assume that the interpolation points are given in nondecreasing order as $x_1 \leq x_2 \leq \cdots \leq x_n$, then the interpolation conditions are

$$D^{\lambda_{\boldsymbol{x}}(i)} g(x_i) = D^{\lambda_{\boldsymbol{x}}(i)} f(x_i) \tag{10.6}$$

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 $i = 1, \ldots, n - d - 1$. 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 $\boldsymbol{A}$ is now given by

$$a_{i,j} = D^{\lambda_{\boldsymbol{x}}(i)} B_{j,d}(x_i),$$

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 $\tau_i \leq x_i < \tau_{i+d+1}$, where equality is allowed in the first inequality only if $D^{\lambda_{\boldsymbol{x}}(i)} B_{i,d}(x_i) \neq 0$. 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 $\boldsymbol{A}$. Any columns may be removed, or equivalently, we consider a subset $\{B_{j_1,d}, \ldots, B_{j_\ell,d}\}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ be a spline space and let $\{B_{j_1,d}, \ldots, B_{j_\ell,d}\}$ be a subsequence of its B-splines. Let $x_1 \leq \cdots \leq x_\ell$ be a sequence of interpolation points with $x_i \leq x_{i+d+1}$ for $i = 1, \ldots, \ell - d - 1$. Then the $\ell \times \ell$ matrix $\boldsymbol{A}(\boldsymbol{j})$ with entries given by*

$$a_{i,q} = D^{\lambda_{\boldsymbol{x}}(i)} B_{j_q,d}(x_i)$$

for $i = 1, \ldots, \ell$ and $q = 1, \ldots, \ell$ is nonsingular if and only if

$$\tau_{j_i} \leq x_i < \tau_{j_i+d+1}, \qquad \text{for } i = 1, \ldots, \ell, \tag{10.7}$$

where equality is allowed in the first inequality if $D^{\lambda_{\boldsymbol{x}}(i)} B_{j_i,d}(x_i) \neq 0$.

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 $\boldsymbol{A}(\boldsymbol{j})$ is singular. Then we can find a nonzero vector $\boldsymbol{c}$ such that $\boldsymbol{A}(\boldsymbol{j})\boldsymbol{c} = \boldsymbol{0}$. Let $f = \sum_{i=1}^{\ell} c_i B_{j_i,d}$ denote the spline with $\boldsymbol{c}$ as its B-spline coefficients, and let G denote the set

$$G = \cup_{i=1}^{\ell} \{(\tau_{j_i}, \tau_{j_i+d+1}) \mid c_i \neq 0\}.$$

To carry through the argument of Theorem 10.6 we need to verify that in the exceptional case where $x_i = \tau_{j_i}$ then $c_i = 0$.

Set $r = \lambda_{\boldsymbol{x}}(i)$ and suppose that the knot τ_{j_i} occurs m times in $\boldsymbol{\tau}$ and that $\tau_{j_i} = x_i$ so $D^r B_{j_i,d}(x_i) \neq 0$. In other words

$$\tau_\mu < x_i = \tau_{\mu+1} = \cdots = \tau_{\mu+m} < \tau_{\mu+m+1}$$

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

$$f(x_i) = Df(x_i) = \cdots = D^r f(x_i) = 0.$$

(Remember that if a derivative is discontinuous at x_i we take limits from the right.) Recall from Lemma 2.6 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 $f(x_i) = Df(x_i) = \cdots = D^{d-m} f(x_i) = 0$ and hence $c_{\mu+m-d} = \cdots = c_\mu = 0$ by Lemma 10.2. The remaining interpolation conditions at x_i are $D^{d-m+1} f(x_i) = D^{d-m+2} f(x_i) = \cdots = D^r f(x_i) = 0$. 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 $D^{d-m+1}B_{\mu+\nu} = 0$ for $\nu > 1$. 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 $x_i = \tau_{j_i}$, 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 $\boldsymbol{A}$ in $\mathbb{R}^{m,n}$ is said to be totally positive if all its square submatrices have nonnegative determinant. More formally, let $\boldsymbol{i} = (i_1, i_2, \ldots, i_\ell)$ and $\boldsymbol{j} = (j_1, j_2, \ldots, j_\ell)$ be two integer sequences such that*

$$1 \leq i_1 < i_2 < \cdots < i_\ell \leq m, \tag{10.8}$$

$$1 \leq i_1 < i_2 < \cdots < i_\ell \leq n, \tag{10.9}$$

and let $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ denote the submatrix of $\boldsymbol{A}$ with entries $(a_{i_p,j_q})_{p,q=1}^{\ell}$. Then $\boldsymbol{A}$ is totally positive if $\det \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j}) \geq 0$ *for all sequences $\boldsymbol{i}$ and $\boldsymbol{j}$ 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 $\boldsymbol{\tau}$ and $\boldsymbol{t}$ be two knot vectors with $\boldsymbol{\tau} \subseteq \boldsymbol{t}$. Then the knot insertion matrix from $\mathbb{S}_{d,\boldsymbol{\tau}}$ to $\mathbb{S}_{d,\boldsymbol{t}}$ is totally positive.*

Proof. Suppose that there are k more knots in $\boldsymbol{t}$ than in $\boldsymbol{\tau}$; our proof is by induction on k. We first note that if $k = 0$, then $\boldsymbol{A} = I$, the identity matrix, while if $k = 1$, then $\boldsymbol{A}$ is a bi-diagonal matrix with one more rows than columns. Let us denote the entries of $\boldsymbol{A}$ by $\big(\alpha_j(i)\big)_{i,j=1}^{n+1,n}$ (if $k = 0$ the range of i is $1, \ldots, n$). 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 $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$. If $j_\ell \geq i_\ell$ then $j_\ell > i_q$ for $q = 1, \ldots, \ell - 1$ so $\alpha_{j_\ell}(i_q) = 0$ for $q < \ell$. This means that only the last entry of the last column of $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ is nonzero. The other possibility is that

$j_\ell \le i_\ell - 1$ 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 $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ is nonzero. Expanding the determinant either by the last column or last row we therefore have $\det \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j}) = \alpha_{j_\ell}(i_\ell) \det \boldsymbol{A}(\boldsymbol{i}', \boldsymbol{j}')$ where $\boldsymbol{i}' = (i_1, \ldots, i_{\ell-1})$ and $\boldsymbol{j}' = (j_1, \ldots, j_{\ell-1})$. Continuing this process we find that

$$\det \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j}) = \alpha_{j_1}(i_1)\alpha_{j_2}(i_2) \cdots \alpha_{j_\ell}(i_\ell)$$

which clearly is nonnegative.

For $k \ge 2$, we make use of the factorization

$$\boldsymbol{A} = \boldsymbol{A}_k \cdots \boldsymbol{A}_1 = \boldsymbol{A}_k \boldsymbol{B}, \tag{10.10}$$

where each $\boldsymbol{A}_r$ corresponds to insertion of one knot and $\boldsymbol{B} = \boldsymbol{A}_{k-1} \cdots \boldsymbol{A}_1$ is the knot insertion matrix for inserting $k-1$ of the knots. By the induction hypothesis we know that both $\boldsymbol{A}_k$ and $\boldsymbol{B}$ are totally positive; we must show that $\boldsymbol{A}$ is totally positive. Let $(\boldsymbol{a}_i)$ and $(\boldsymbol{b}_i)$ denote the rows of $\boldsymbol{A}$ and $\boldsymbol{B}$, and let $\big(\alpha_j(i)\big)_{i,j=1}^{m,m-1}$ denote the entries of $\boldsymbol{A}_k$. From (10.10) we have

$$\boldsymbol{a}_i = \alpha_{i-1}(i)\boldsymbol{b}_{i-1} + \alpha_i(i)\boldsymbol{b}_i \qquad \text{for } i = 1, \ldots, m,$$

where $\alpha_0(1) = \alpha_m(m) = 0$. Let $\boldsymbol{a}_i(\boldsymbol{j})$ and $\boldsymbol{b}_i(\boldsymbol{j})$ denote the vectors obtained by keeping only entries $(j_q)_{q=1}^{\ell}$ of $\boldsymbol{a}_i$ and $\boldsymbol{b}_i$ respectively. Row q of $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ of $\boldsymbol{A}$ is then given by

$$\boldsymbol{a}_{i_q}(\boldsymbol{j}) = \alpha_{i_q-1}(i_q)\boldsymbol{b}_{i_q-1}(\boldsymbol{j}) + \alpha_{i_q}(i_q)\boldsymbol{b}_{i_q}(\boldsymbol{j}).$$

Using the linearity of the determinant in row q we therefore have

$$\begin{aligned}\det\begin{pmatrix}\boldsymbol{a}_{i_1}(\boldsymbol{j})\\ \vdots \\ \boldsymbol{a}_{i_q}(\boldsymbol{j}) \\ \vdots \\ \boldsymbol{a}_{i_\ell}(\boldsymbol{j})\end{pmatrix} &= \det\begin{pmatrix}\boldsymbol{a}_{i_1}(\boldsymbol{j})\\ \vdots \\ \alpha_{i_q-1}(i_q)\boldsymbol{b}_{i_q-1}(\boldsymbol{j}) + \alpha_{i_q}(i_q)\boldsymbol{b}_{i_q}(\boldsymbol{j}) \\ \vdots \\ \boldsymbol{a}_{i_\ell}(\boldsymbol{j})\end{pmatrix} \\ &= \alpha_{i_q-1}(i_q)\det\begin{pmatrix}\boldsymbol{a}_{i_1}(\boldsymbol{j})\\ \vdots \\ \boldsymbol{b}_{i_q-1}(\boldsymbol{j}) \\ \vdots \\ \boldsymbol{a}_{i_\ell}(\boldsymbol{j})\end{pmatrix} + \alpha_{i_q}(i_q)\det\begin{pmatrix}\boldsymbol{a}_{i_1}(\boldsymbol{j})\\ \vdots \\ \boldsymbol{b}_{i_q}(\boldsymbol{j}) \\ \vdots \\ \boldsymbol{a}_{i_\ell}(\boldsymbol{j})\end{pmatrix}.\end{aligned}$$

By expanding the other rows similarly we find that $\det \boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ can be written as a sum of determinants of submatrices of $\boldsymbol{B}$, multiplied by products of $\alpha_j(i)$'s. By the induction hypothesis all these quantities are nonnegative, so the determinant of $\boldsymbol{A}(\boldsymbol{i}, \boldsymbol{j})$ must also be nonnegative. Hence $\boldsymbol{A}$ 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 $\mathbb{S}_{d,\boldsymbol{\tau}}$ be a spline space and let $\{B_{j_1,d},\ldots,B_{j_\ell,d}\}$ be a subsequence of its B-splines. Let $x_1 \leq \cdots \leq x_\ell$ be a sequence of interpolation points with $x_i \leq x_{i+d+1}$ for $i = 1, \ldots, \ell-d-1$, and denote by $\boldsymbol{A}(\boldsymbol{j})$ the $\ell \times \ell$ matrix with entries given by*

$$a_{i,q} = D^{\lambda_{\boldsymbol{x}}(i)} B_{j_q,d}(x_i)$$

for $i = 1, \ldots, \ell$ and $q = 1, \ldots, \ell$. Then

$$\det \boldsymbol{A}(\boldsymbol{j}) \geq 0.$$

Proof. We first prove the claim in the case $x_1 < x_2 < \cdots < x_\ell$. By inserting knots of multiplicity $d+1$ at each of $(x_i)_{i=1}^{\ell}$ we obtain a knot vector $\boldsymbol{t}$ that contains $\boldsymbol{\tau}$ as a subsequence. If $t_{i-1} < t_i = t_{i+d} < t_{i+d+1}$ we know from Lemma 2.6 that $B_{j,d,\boldsymbol{\tau}}(t_i) = \alpha_{j,d}(i)$. This means that the matrix $\boldsymbol{A}(\boldsymbol{j})$ appears as a submatrix of the knot insertion matrix from $\boldsymbol{\tau}$ to $\boldsymbol{t}$. It therefore follows from Theorem 10.9 that $\det \boldsymbol{A}(\boldsymbol{j}) \geq 0$ in this case.

To prove the theorem in the general case we consider a set of distinct collocation points $y_1 < \cdots < y_\ell$ and let $\boldsymbol{A}(\boldsymbol{j},\boldsymbol{y})$ denote the corresponding collocation matrix. Set $\lambda^i = \lambda_{\boldsymbol{x}}(i)$ and let ρ_i denote the linear functional given by

$$\rho_i f = \lambda^i!\,[y_{i-\lambda^i},\ldots,y_i]f \tag{10.11}$$

for $i = 1, \ldots, \ell$. Here $[\cdot,\ldots,\cdot]f$ 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 $\boldsymbol{D}$ the matrix with (i,j)-entry $\rho_i B_{j,d}$, we find by properties of determinants and (10.11) that

$$\det \boldsymbol{D} = \gamma_{1,1} \cdots \gamma_{\ell,\ell} \det \boldsymbol{A}(\boldsymbol{j},\boldsymbol{y}).$$

If we now let $\boldsymbol{y}$ tend to $\boldsymbol{x}$ 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 $\boldsymbol{D}$ tends to $\boldsymbol{A}(\boldsymbol{j})$ so $\det \boldsymbol{A}(\boldsymbol{j}) \geq 0$. ∎

APPENDIX A

Some Linear Algebra

A.1 Matrices

The collection of m, n matrices

$$\boldsymbol{A} = \begin{pmatrix} a_{1,1}, & \cdots & , a_{1,n} \\ \cdots & & \cdots \\ a_{m,1}, & \cdots & , a_{m,n} \end{pmatrix}$$

with real elements $a_{i,j}$ is denoted by $\mathbb{R}^{m,n}$. If $n = 1$ then $\boldsymbol{A}$ is called a column vector. Similarly, if $m = 1$ then $\boldsymbol{A}$ 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 $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n \in \mathbb{R}^m$ is* linearly independent *if $x_1\boldsymbol{a}_1 + \cdots + x_n\boldsymbol{a}_n = \boldsymbol{0}$ for some real numbers $x_1, \ldots, x_n$, implies that $x_1 = \cdots = x_n = 0$.*

Suppose $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ are the columns of a matrix $\boldsymbol{A} \in \mathbb{R}^{m,n}$. For a vector $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \mathbb{R}^n$ we have $\boldsymbol{Ax} = \sum_{j=1}^n x_j\boldsymbol{a}_j$. It follows that the collection $\boldsymbol{a}_1, \ldots, \boldsymbol{a}_n$ is linearly independent if and only if $\boldsymbol{Ax} = \boldsymbol{0}$ implies $\boldsymbol{x} = \boldsymbol{0}$.

Definition A.2. *A square matrix $\boldsymbol{A}$ such that $\boldsymbol{Ax} = \boldsymbol{0}$ implies $\boldsymbol{x} = \boldsymbol{0}$ is said to be nonsingular.*

Definition A.3. *A square matrix $\boldsymbol{A} \in \mathbb{R}^{n,n}$ is said to be invertible if for some $\boldsymbol{B} \in \mathbb{R}^{n,n}$*

$$\boldsymbol{BA} = \boldsymbol{AB} = \boldsymbol{I},$$

where $\boldsymbol{I} \in \mathbb{R}^{n,n}$ is the identity matrix.

An invertible matrix $\boldsymbol{A}$ has a unique inverse $\boldsymbol{B} = \boldsymbol{A}^{-1}$. If $\boldsymbol{A}, \boldsymbol{B}$, and $\boldsymbol{C}$ are square matrices, and $\boldsymbol{A} = \boldsymbol{BC}$, then $\boldsymbol{A}$ is invertible if and only if both $\boldsymbol{B}$ and $\boldsymbol{C}$ are also invertible. Moreover, the inverse of $\boldsymbol{A}$ is the product of the inverses of $\boldsymbol{B}$ and $\boldsymbol{C}$ in reverse order, $\boldsymbol{A}^{-1} = \boldsymbol{C}^{-1}\boldsymbol{B}^{-1}$.

A.1.2 Determinants.

The determinant of a square matrix $\boldsymbol{A}$ will be denoted $\det(\boldsymbol{A})$ or

$$\begin{vmatrix} a_{1,1}, & \dots & , a_{1,n} \\ \vdots & & \vdots \\ a_{n,1}, & \dots & , a_{n,n} \end{vmatrix}.$$

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 $\boldsymbol{A} \in \mathbb{R}^{n,n}$.*

1. *$\boldsymbol{A}$ is nonsingular.*

2. *$\boldsymbol{A}$ is invertible.*

3. *$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has a unique solution $\boldsymbol{x} = \boldsymbol{A}^{-1}b$ for any $\boldsymbol{b} \in \mathbb{R}^n$.*

4. *$\boldsymbol{A}$ has linearly independent columns.*

5. *$\boldsymbol{A}^T$ is nonsingular.*

6. *$\boldsymbol{A}$ has linearly independent rows.*

7. $\det(\boldsymbol{A}) \neq \boldsymbol{0}$.

We also have a number of criteria for a matrix to be singular.

Theorem A.5. *The following is equivalent for a square matrix $\boldsymbol{A} \in \mathbb{R}^{n,n}$.*

1. *There is a nonzero $\boldsymbol{x} \in \mathbb{R}^n$ so that $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$.*

2. *$\boldsymbol{A}$ has no inverse.*

3. *$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ has either no solution or an infinite number of solutions.*

4. *$\boldsymbol{A}$ has linearly dependent columns.*

5. *There is a nonzero $\boldsymbol{x}$ so that $\boldsymbol{x}^T\boldsymbol{A} = \boldsymbol{0}$.*

6. *$\boldsymbol{A}$ has linearly dependent rows.*

7. $\det(\boldsymbol{A}) = \boldsymbol{0}$.

Corollary A.6. *A matrix with more columns than rows has linearly dependent columns.*

Proof. Suppose $\boldsymbol{A} \in \mathbb{R}^{m,n}$ with $n > m$. By adding $n - m$ rows of zeros to $\boldsymbol{A}$ we obtain a square matrix $\boldsymbol{B} \in \mathbb{R}^{n,n}$. This matrix has linearly dependent rows. By Theorem A.4 the matrix $\boldsymbol{B}$ has linearly dependent columns. But then the columns of $\boldsymbol{A}$ are also linearly dependent. ∎

A.2 Vector Norms

Formally, a *vector norm* $||\ || = ||\boldsymbol{x}||$, is a function $\|\ \| : \mathbb{R}^n \to [0, \infty)$ that satisfies for $\boldsymbol{x}, \boldsymbol{y}, \in \mathbb{R}^n$, and $\alpha \in \mathbb{R}$ the following properties

$$\begin{array}{ll} 1. & ||\boldsymbol{x}|| = 0 \quad \text{implies} \quad \boldsymbol{x} = \boldsymbol{0}. \\ 2. & ||\alpha \boldsymbol{x}|| = |\alpha| ||\boldsymbol{x}||. \\ 3. & ||\boldsymbol{x} + \boldsymbol{y}|| \le ||\boldsymbol{x}|| + ||\boldsymbol{y}||. \end{array} \tag{A.1}$$

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 $p \ge 1$ and $\boldsymbol{x} = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^n$ by

$$\begin{aligned} ||\boldsymbol{x}||_p &= (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{1/p}. \\ ||\boldsymbol{x}||_\infty &= \max_i |x_i|. \end{aligned} \tag{A.2}$$

Since

$$||\boldsymbol{x}||_\infty \le ||\boldsymbol{x}||_p \le n^{1/p} ||\boldsymbol{x}||_\infty, \qquad p \ge 1 \tag{A.3}$$

and $\lim_{p\to\infty} n^{1/p} = 1$ for any $n \in \mathbb{N}$ we see that $\lim_{p\to\infty} ||\boldsymbol{x}||_p = ||\boldsymbol{x}||_\infty$.

The 1,2, and ∞ norms are the most important. We have

$$||\boldsymbol{x}||_2^2 = x_1^2 + \dots + x_n^2 = \boldsymbol{x}^T \boldsymbol{x}. \tag{A.4}$$

Lemma A.7 (The Hölder inequality)**.** *We have for* $1 \le p \le \infty$ *and* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$

$$\sum_{i=1}^n |x_i y_i| \le ||\boldsymbol{x}||_p ||\boldsymbol{y}||_q, \quad \textit{where} \quad \frac{1}{p} + \frac{1}{q} = 1. \tag{A.5}$$

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

$$f(\lambda x + (1-\lambda) y) \le \lambda f(x) + (1-\lambda) f(y) \tag{A.6}$$

for every $\lambda \in [0, 1]$ and $x, y \in \mathbb{R}$.

If $\boldsymbol{x} = \boldsymbol{0}$ or $\boldsymbol{y} = \boldsymbol{0}$, there is nothing to prove. Suppose $\boldsymbol{x}, \boldsymbol{y} \ne \boldsymbol{0}$. Define $\boldsymbol{u} = \boldsymbol{x}/||\boldsymbol{x}||_p$ and $\boldsymbol{v} = \boldsymbol{y}/||\boldsymbol{y}||_q$. Then $||\boldsymbol{u}||_p = ||\boldsymbol{v}||_q = 1$. If we can prove that $\sum_i |u_i v_i| \le 1$, we are done because then $\sum_i |x_i y_i| = ||\boldsymbol{x}||_p ||\boldsymbol{y}||_q \sum_i |u_i v_i| \le ||\boldsymbol{x}||_p ||\boldsymbol{y}||_q$. Since $|u_i v_i| = |u_i||v_i|$, we can assume that $u_i \ge 0$ and $v_i \ge 0$. 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

$$e^{s_i/p + t_i/q} \le \frac{1}{p} e^{s_i} + \frac{1}{q} e^{t_i}.$$

But then

$$\sum_i |u_i v_i| = \sum_i e^{s_i/p + t_i/q} \le \frac{1}{p} \sum_i e^{s_i} + \frac{1}{q} \sum_i e^{t_i} = \frac{1}{p} \sum_i u_i^p + \frac{1}{q} \sum_i v_i^q = \frac{1}{p} + \frac{1}{q} = 1.$$

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* $1 \le p \le \infty$ *and* $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}$

$$||\boldsymbol{x} + \boldsymbol{y}||_p \le ||\boldsymbol{x}||_p + ||\boldsymbol{y}||_p. \tag{A.7}$$

Proof. Let $\boldsymbol{u} = (u_1, \dots, u_n)$ with $u_i = |x_i + y_i|^{p-1}$. Since $q(p-1) = p$ and $p/q = p-1$, we find

$$||\boldsymbol{u}||_q = (\sum_i |x_i + y_i|^{q(p-1)})^{1/q} = (\sum_i |x_i + y_i|^p)^{1/q} = ||\boldsymbol{x} + \boldsymbol{y}||_p^{p/q} = ||\boldsymbol{x} + \boldsymbol{y}||_p^{p-1}.$$

Using this and the Hölder inequality we obtain

$$\begin{aligned} ||\boldsymbol{x} + \boldsymbol{y}||_p^p = \sum_i |x_i + y_i|^p \le \sum_i |u_i||x_i| + \sum_i |u_i||y_i| &\le (||\boldsymbol{x}||_p + ||\boldsymbol{y}||_p)||\boldsymbol{u}||_q \\ &\le (||\boldsymbol{x}||_p + ||\boldsymbol{y}||_p)||\boldsymbol{x} + \boldsymbol{y}||_p^{p-1}. \end{aligned}$$

Dividing by $||\boldsymbol{x} + \boldsymbol{y}||_p^{p-1}$ 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

$$||\boldsymbol{x}||_\infty \le ||\boldsymbol{x}||_p \le n^{1/p}||\boldsymbol{x}||_\infty, \quad p \ge 1.$$

More generally, we say that two vector norms $||\ ||$ and $||\ ||'$ are *equivalent* if there exists positive constants μ and M such that

$$\mu||\boldsymbol{x}|| \le ||\boldsymbol{x}||' \le M||\boldsymbol{x}|| \tag{A.8}$$

for all $\boldsymbol{x} \in \mathbb{R}^n$.

Theorem A.9. *All vector norms on* $\mathbb{R}^n$ *are equivalent.*

Proof. It is enough to show that a vector norm $||\ ||$ is equivalent to the l_∞ norm, $||\ ||_\infty$. Let $\boldsymbol{x} \in \mathbb{R}^n$ and let $\boldsymbol{e}_i, i = 1, \dots, n$ be the unit vectors in $\mathbb{R}^n$. Writing $\boldsymbol{x} = x_1\boldsymbol{e}_1 + \cdots + x_n\boldsymbol{e}_n$ we have

$$||\boldsymbol{x}|| \le \sum_i |x_i|||\boldsymbol{e}_i|| \le ||\boldsymbol{x}||_\infty M, \quad M = \sum_i ||\boldsymbol{e}_i||.$$

To find $\mu > 0$ such that $||\boldsymbol{x}|| \ge \mu||\boldsymbol{x}||_\infty$ for all $\boldsymbol{x} \in \mathbb{R}^n$ is less elementary. Consider the function f given by $f(x) = ||\boldsymbol{x}||$ defined on the l_∞ "unit ball"

$$S = \{\boldsymbol{x} \in \mathbb{R}^n : ||\boldsymbol{x}||_\infty = 1\}.$$

S is a closed and bounded set. From the inverse triangle inequality

$$|\,||\boldsymbol{x}|| - ||\boldsymbol{y}||\,| \le ||\boldsymbol{x} - \boldsymbol{y}||, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n.$$

it follows that f is continuous on S. But then f attains its maximum and minimum on S, i.e. there is a point $\boldsymbol{x}^* \in S$ such that

$$||\boldsymbol{x}^*|| = \min_{\boldsymbol{x}\in S} ||\boldsymbol{x}||.$$

Moreover, since $\boldsymbol{x}^*$ is nonzero we have $\mu := ||\boldsymbol{x}^*|| > 0$. If $\boldsymbol{x} \in \mathbb{R}^n$ is nonzero then $\boldsymbol{x} = \boldsymbol{x}/||\boldsymbol{x}||_\infty \in S$. Thus

$$\mu \le ||\boldsymbol{x}|| = ||\frac{\boldsymbol{x}}{||\boldsymbol{x}||_\infty}|| = \frac{1}{||\boldsymbol{x}||_\infty}||\boldsymbol{x}||,$$

and this establishes the lower inequality. ∎

It can be shown that for the p norms we have for any q with $1 \le q \le p \le \infty$

$$||\boldsymbol{x}||_p \le ||\boldsymbol{x}||_q \le n^{1/q-1/p}||\boldsymbol{x}||_p, \quad \boldsymbol{x} \in \mathbb{R}^n. \tag{A.9}$$

<

A.3 Vector spaces of functions

In $\mathbb{R}^m$ we have the operations $\boldsymbol{x} + \boldsymbol{y}$ and $a\boldsymbol{x}$ 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

$$(f + g)(x) = f(x) + g(x),$$
$$(af(x) = af(x).$$

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 $j = 0, 1, \ldots, r$.

Definition A.12. *A vector space S is said to be* finite dimesional *if*

$$S = \operatorname{span}(\phi_1, \ldots, \phi_n) = \{\sum_{j=1}^{n} c_j \phi_j : c_j \in \mathbb{R}\},$$

for a finite number of functions $\phi_1, \ldots, \phi_n$ in S. The functions $\phi_1, \ldots, \phi_n$ are said to span *or* generate *S.*

Of the examples above the space $\pi_d = \operatorname{span}(1, x, x^2, \ldots x^d)$ generated by the monomials $1, x, x^2, \ldots x^d$ 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 $f \in S = \operatorname{span}(\phi_1, \ldots, \phi_n)$ then $f = \sum_{j=1}^{n} c_j \phi_j$ for some $\boldsymbol{c} = (c_1, \ldots, c_n)$. With $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$ we will often use the vector notation

$$f(x) = \boldsymbol{\phi}(x)^T \boldsymbol{c} \tag{A.10}$$

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 $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$ in a vector space S is said to be* linearly independent *on a subset J of I if $\boldsymbol{\phi}(x)^T \boldsymbol{c} = c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0$ for all $x \in J$ implies that $\boldsymbol{c} = \boldsymbol{0}$. 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 $f = \boldsymbol{\phi}^T \boldsymbol{c} = \boldsymbol{\phi}^T \boldsymbol{b}$ for some $\boldsymbol{c}, \boldsymbol{b} \in \mathbb{R}^n$ then $f = \boldsymbol{\phi}^T(\boldsymbol{c} - \boldsymbol{b}) = \boldsymbol{0}$. Since $\boldsymbol{\phi}$ is linearly independent we have $\boldsymbol{c} - \boldsymbol{b} = \boldsymbol{0}$, or $\boldsymbol{c} = \boldsymbol{b}$.

Definition A.14. *A set of functions $\boldsymbol{\phi}^T = (\phi_1, \ldots, \phi_n)$ in a vector space S is* a basis for *S if the following two conditions hold*

1. *$\boldsymbol{\phi}$ is linearly independent.*
2. *$S = \operatorname{span}(\boldsymbol{\phi})$.*

Theorem A.15. *The monomials $1, x, x^2, \ldots x^d$ are linearly independent on any set $J \subset \mathbb{R}$ containing at least $d+1$ distinct points. In particular these functions form as basis for π_d.*

Proof. Let $x_0, \ldots, x_d$ be $d+1$ distinct points in J, and let $p(x) = c_0 + c_1 x + \cdots + c_d x^d = 0$ for all $x \in J$. Then $p(x_i) = 0$, for $i = 0, 1, \ldots, d$. 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 $k = 0, 1, \ldots, d$. 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 $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$ be a basis for S and $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_m)^T$ a basis for T. Then*

$$\boldsymbol{\phi} = \boldsymbol{A}^T \boldsymbol{\psi}, \tag{A.11}$$

for some matrix $\boldsymbol{A} \in \mathbb{R}^{m,n}$. If $f = \boldsymbol{\phi}^T \boldsymbol{c} \in S$ is given then $f = \boldsymbol{\psi}^T \boldsymbol{b}$ with

$$\boldsymbol{b} = \boldsymbol{A}\boldsymbol{c}. \tag{A.12}$$

Moreover $\boldsymbol{A}$ has linearly independent columns.

Proof. Since $\phi_j \in T$ there are real numbers $a_{i,j}$ such that

$$\phi_j = \sum_{i=1}^{m} a_{i,j} \psi_i, \quad \text{for} \quad j = 1, \ldots, n,$$

This equation is simply the component version of (A.11). If $f \in S$ then $f \in T$ and $f = \boldsymbol{\psi}^T \boldsymbol{b}$ for some $\boldsymbol{b}$. By (A.11) we have $\boldsymbol{\phi}^T = \boldsymbol{\psi}^T \boldsymbol{A}$ and $f = \boldsymbol{\phi}^T \boldsymbol{c} = \boldsymbol{\psi}^T \boldsymbol{A}\boldsymbol{c}$ or $\boldsymbol{\psi}^T \boldsymbol{b} = \boldsymbol{\psi}^T \boldsymbol{A}\boldsymbol{c}$. Since $\boldsymbol{\psi}$ is linearly independent we get (A.12). Finally, to show that $\boldsymbol{A}$ has linearly independent columns suppose $\boldsymbol{A}\boldsymbol{c} = \boldsymbol{0}$. Define $f \in S$ by $f = \boldsymbol{\phi}^T \boldsymbol{c}$. By (A.11) we have $f = \boldsymbol{\psi}^T \boldsymbol{A}\boldsymbol{c} = \boldsymbol{0}$. But then $f = \boldsymbol{\phi}^T \boldsymbol{c} = \boldsymbol{0}$. Since $\boldsymbol{\phi}$ is linearly independent we conclude that $\boldsymbol{c} = \boldsymbol{0}$. ∎

The matrix $\boldsymbol{A}$ 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 $\boldsymbol{\psi} = (\psi_1 \ldots, \psi_k)^T$ is a linearly independent set in a vector space $S = \operatorname{span}(\phi_1, \ldots, \phi_n)$, then $k \le n$.*

Proof. With $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$ we have

$$\boldsymbol{\psi} = \boldsymbol{A}^T \boldsymbol{\phi}, \quad \text{for some} \quad \boldsymbol{A} \in \mathbb{R}^{n,k}.$$

If $k > n$ then $\boldsymbol{A}$ 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 $\boldsymbol{c} \in \mathbb{R}^k$ such that $\boldsymbol{A}\boldsymbol{c} = \boldsymbol{0}$. But then $\boldsymbol{\psi}^T \boldsymbol{c} = \boldsymbol{\phi}^T \boldsymbol{A}\boldsymbol{c} = \boldsymbol{0}$, for some nonzero $\boldsymbol{c}$. This implies that $\boldsymbol{\psi}$ is linearly dependent, a contradiction. We conclude that $k \le n$. ∎

Lemma A.18. *Every basis for a vector space must have the same number of elements.*

Proof. Suppose $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_n)^T$ and $\boldsymbol{\psi} = (\psi_1, \ldots, \psi_m)^T$ are two bases for the vector space. We need to show that $m = n$. Now

$$\boldsymbol{\phi} = \boldsymbol{A}^T \boldsymbol{\psi}, \quad \text{for some} \quad \boldsymbol{A} \in \mathbb{R}^{m,n},$$

$$\boldsymbol{\psi} = \boldsymbol{B}^T \boldsymbol{\phi}, \quad \text{for some} \quad \boldsymbol{B} \in \mathbb{R}^{n,m}.$$

By Lemma A.16 we know that both $\boldsymbol{A}$ and $\boldsymbol{B}$ 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 $S_k = \mathrm{span}(\psi_1, \dots, \psi_k)$ 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 $S_{k+1} = \mathrm{span}(\psi_1, \dots, \psi_{k+1})$. If $S_{k+1} = S$ then we set $m = k + 1$ and stop the process. Otherwise we continue to generate vector spaces $S_{k+2}, S_{k+3}, \cdots$. 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 $S = \mathrm{span}(\boldsymbol{\phi})$. If $\boldsymbol{\phi}$ contains $\dim(S)$ elements then $\boldsymbol{\phi}$ is a basis for S.*

Proof. Let $n = \dim(S)$ and suppose $\boldsymbol{\phi} = (\phi_1, \dots, \phi_n)$ 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 $S = \mathrm{span}(\phi_1, \dots, \phi_{n-1})$ and $\dim(S) < n$ by Lemma A.17, a contradiction to the assumption that $\boldsymbol{\phi}$ is linearly dependent. ∎

A.4 Normed Vector Spaces

Suppose S is a vector space of functions. A *norm* $||\ || = ||f||$, is a function $||\ || : S \to [0, \infty)$ that satisfies for $f, g, \in S$, and $\alpha \in \mathbb{R}$ the following properties

$$\begin{array}{ll} 1. & ||f|| = 0 \quad \text{implies} \quad f = 0. \\ 2. & ||\alpha f|| = |\alpha| ||f||. \\ 3. & ||f + g|| \le ||f|| + ||g||. \end{array} \tag{A.13}$$

Property 3 is known as the *Triangle Inequality.* The pair $(S, ||\ ||)$ 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 $1 \le p \le \infty$ and $f \in S$ by

$$\begin{aligned} ||f||_p = ||f||_{L^p[a,b]} &= \left(\int_a^b |f(x)|^p dx\right)^{1/p}, \quad p \ge 1, \\ ||f||_\infty = ||f||_{L^\infty[a,b]} &= \max_{a \le x \le b} |f(x)|. \end{aligned} \tag{A.14}$$

The 1,2, and ∞ norms are the most important.

We have for $1 \le p \le \infty$ and $f, g \in S$ the Hölder inequality

$$\int_a^b |f(x)g(x)|dx \le ||f||_p ||g||_q, \quad \textit{where} \quad \frac{1}{p} + \frac{1}{q} = 1, \tag{A.15}$$

and the Minkowski inequality

$$||f+g||_p \leq ||f||_p + ||g||_p. \tag{A.16}$$

For $p = 2$ (A.15) is known as the Schwarz inequality, the Cauchy-Schwarz inequality, or the Buniakowski-Cauchy- Schwarz inequality.

Index