Spline Methods
Draft

Tom Lyche and Knut Mørken

January 5, 2005

Contents

1 Splines and B-splines an Introduction **3**

1.1 Convex combinations and convex hulls . . . 3
1.1.1 Stable computations . . . 4
1.1.2 The convex hull of a set of points . . . 4
1.2 Some fundamental concepts . . . 7
1.3 Interpolating polynomial curves . . . 8
1.3.1 Quadratic interpolation of three points . . . 9
1.3.2 General polynomial interpolation . . . 10
1.3.3 Interpolation by convex combinations? . . . 13
1.4 Bézier curves . . . 14
1.4.1 Quadratic Bézier curves . . . 14
1.4.2 Bézier curves based on four and more points . . . 16
1.4.3 Composite Bézier curves . . . 19
1.5 A geometric construction of spline curves . . . 20
1.5.1 Linear spline curves . . . 20
1.5.2 Quadratic spline curves . . . 22
1.5.3 Spline curves of higher degrees . . . 23
1.5.4 Smoothness of spline curves . . . 26
1.6 Representing spline curves in terms of basis functions . . . 28
1.7 Conclusion . . . 31

2 Basic properties of splines and B-splines **37**

2.1 Some simple consequences of the recurrence relation . . . 37
2.2 Linear combinations of B-splines . . . 43
2.2.1 Spline functions . . . 43
2.2.2 Spline curves . . . 46
2.3 A matrix representation of B-splines . . . 47
2.4 Algorithms for evaluating a spline . . . 50

3 Further properties of splines and B-splines **57**

3.1 Linear independence and representation of polynomials . . . 57
3.1.1 Some properties of the B-spline matrices . . . 57
3.1.2 Marsden's identity and representation of polynomials . . . 59
3.1.3 Linear independence of B-splines . . . 61

3.2 Differentiation and smoothness of B-splines 62
3.2.1 Derivatives of B-splines 63
3.2.2 Computing derivatives of splines and B-splines 66
3.2.3 Smoothness of B-splines 68
3.3 B-splines as a basis for piecewise polynomials 70

4 Knot insertion **75**
4.1 Convergence of the control polygon for spline functions 75
4.2 Knot insertion 78
4.2.1 Formulas and algorithms for knot insertion 79
4.3 B-spline coefficients as functions of the knots 85
4.3.1 The blossom 85
4.3.2 B-spline coefficients as blossoms 88
4.4 Inserting one knot at a time 90
4.5 Bounding the number of sign changes in a spline 92

5 Spline Approximation of Functions and Data **99**
5.1 Local Approximation Methods 100
5.1.1 Piecewise linear interpolation 100
5.1.2 Cubic Hermite interpolation 102
5.1.3 Estimating the derivatives 105
5.2 Cubic Spline Interpolation 105
5.2.1 Interpretations of cubic spline interpolation 109
5.2.2 Numerical solution and examples 110
5.3 General Spline Approximation 112
5.3.1 Spline interpolation 112
5.3.2 Least squares approximation 113
5.4 The Variation Diminishing Spline Approximation 117
5.4.1 Preservation of bounds on a function 118
5.4.2 Preservation of monotonicity 120
5.4.3 Preservation of convexity 122

6 Parametric Spline Curves **127**
6.1 Definition of Parametric Curves 127
6.1.1 Regular parametric representations 127
6.1.2 Changes of parameter and parametric curves 129
6.1.3 Arc length parametrisation 130
6.2 Approximation by Parametric Spline Curves 131
6.2.1 Definition of parametric spline curves 131
6.2.2 The parametric variation diminishing spline approximation 133
6.2.3 Parametric spline interpolation 134
6.2.4 Assigning parameter values to discrete data 135
6.2.5 General parametric spline approximation 137

7 Tensor Product Spline Surfaces **139**
7.1 Explicit tensor product spline surfaces . . . 139
7.1.1 Definition of the tensor product spline . . . 139
7.1.2 Evaluation of tensor product spline surfaces . . . 142
7.2 Approximation methods for tensor product splines . . . 143
7.2.1 The variation diminishing spline approximation . . . 143
7.2.2 Tensor Product Spline Interpolation . . . 144
7.2.3 Least Squares for Gridded Data . . . 148
7.3 General tensor product methods . . . 151
7.4 Trivariate Tensor Product Methods . . . 154
7.5 Parametric Surfaces . . . 157
7.5.1 Parametric Tensor Product Spline Surfaces . . . 158

8 Quasi-interpolation methods **161**
8.1 A general recipe . . . 161
8.1.1 The basic idea . . . 162
8.1.2 A more detailed description . . . 162
8.2 Some quasi-interpolants . . . 164
8.2.1 Piecewise linear interpolation . . . 164
8.2.2 A 3-point quadratic quasi-interpolant . . . 165
8.2.3 A 5-point cubic quasi-interpolant . . . 166
8.2.4 Some remarks on the constructions . . . 167
8.3 Quasi-interpolants are linear operators . . . 168
8.4 Different kinds of linear functionals and their uses . . . 169
8.4.1 Point functionals . . . 169
8.4.2 Derivative functionals . . . 169
8.4.3 Integral functionals . . . 170
8.4.4 Preservation of moments and interpolation of linear functionals . . . 171
8.4.5 Least squares approximation . . . 172
8.4.6 Computation of integral functionals . . . 173
8.5 Alternative ways to construct coefficient functionals . . . 173
8.5.1 Computation via evaluation of linear functionals . . . 173
8.5.2 Computation via explicit representation of the local approximation . 174
8.6 Two quasi-interpolants based on point functionals . . . 175
8.6.1 A quasi-interpolant based on the Taylor polynomial . . . 175
8.6.2 Quasi-interpolants based on evaluation . . . 177

9 Approximation theory and stability **181**
9.1 The distance to polynomials . . . 181
9.2 The distance to splines . . . 183
9.2.1 The constant and linear cases . . . 184
9.2.2 The quadratic case . . . 184
9.2.3 The general case . . . 186
9.3 Stability of the B-spline basis . . . 189
9.3.1 A general definition of stability . . . 189
9.3.2 The condition number of the B-spline basis. Infinity norm . . . 190

9.3.3 The condition number of the B-spline basis. p-norm 192

10 Shape Preserving Properties of B-splines **199**
10.1 Bounding the number of zeros of a spline 199
10.2 Uniqueness of spline interpolation 202
10.2.1 Lagrange Interpolation 203
10.2.2 Hermite Interpolation 204
10.3 Total positivity 206

A Some Linear Algebra **211**
A.1 Matrices 211
A.1.1 Nonsingular matrices, and inverses. 211
A.1.2 Determinants. 212
A.1.3 Criteria for nonsingularity and singularity. 212
A.2 Vector Norms 213
A.3 Vector spaces of functions 215
A.3.1 Linear independence and bases 216
A.4 Normed Vector Spaces 218

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* .

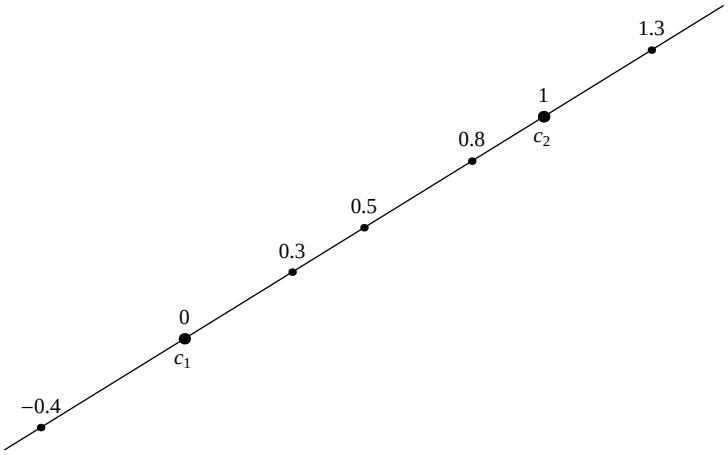


Figure 1.1. Some points on the line $(1-\lambda)\boldsymbol{c}_1 + \lambda\boldsymbol{c}_2$ and the corresponding values of λ.

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 $0 \le \lambda \le 1$. This expression is usually implemented on a computer by expressing it in terms of convex combinations of real numbers,

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

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

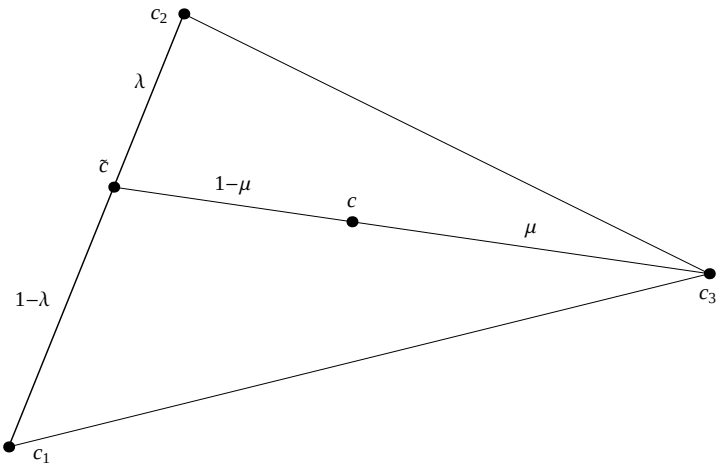


Figure 1.2. Determining the convex hull of three points.

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.

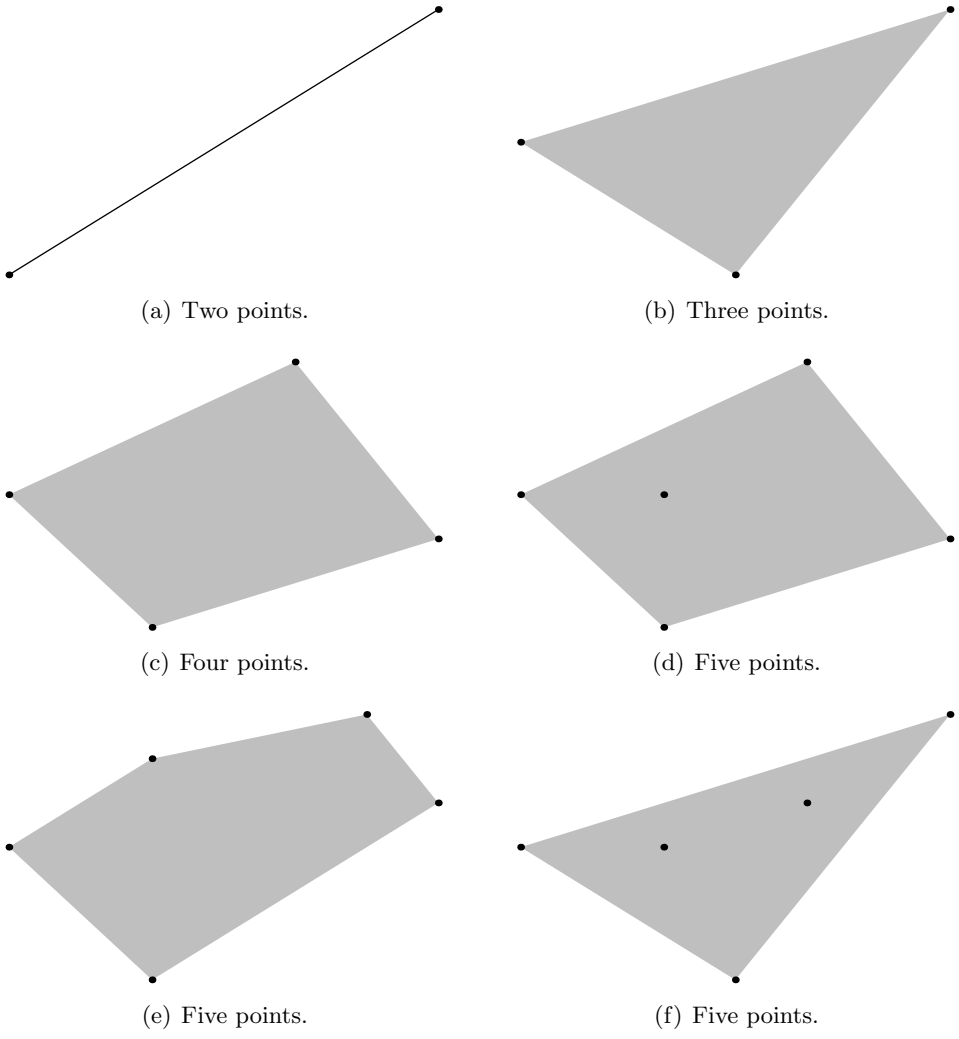
We can form convex and affine combinations in any space dimension, we just let $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ be points in the appropriate space. If we are working in $\mathbb{R}^n$ for instance, then $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ have n components. In our examples we will mostly 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 $0 \leq \lambda_i \leq 1$ for $i = 1$, $2, \ldots, n$. 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.

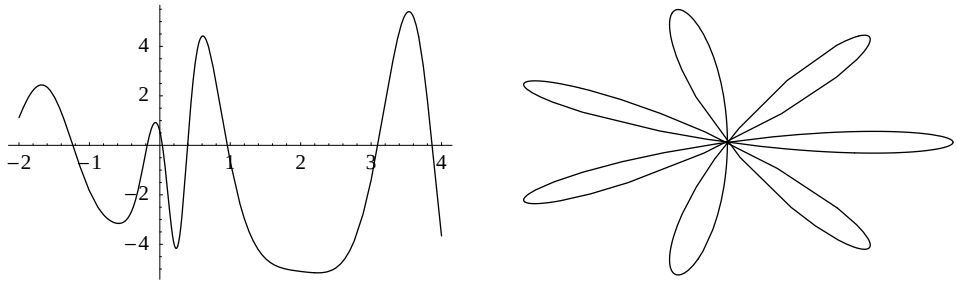


(a) Two points. (b) Three points.

(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

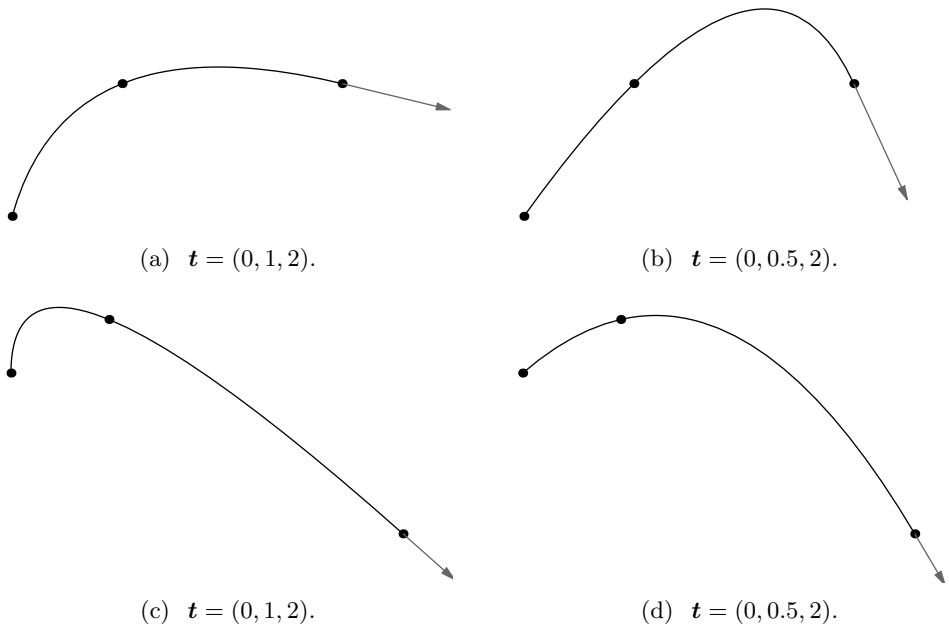
$$\left|\boldsymbol{q}'(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1; t_0, t_1)\right| = \frac{|\boldsymbol{c}_1 - \boldsymbol{c}_0|}{t_1 - t_0} = \frac{\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}}{t_1 - t_0}.$$

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

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

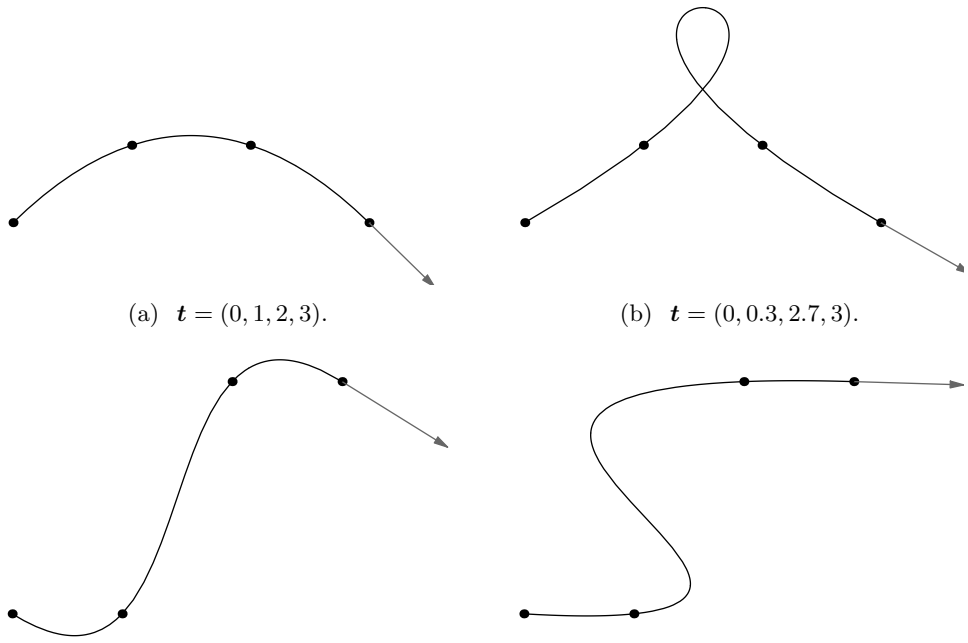
and combine these to obtain the cubic interpolant $\boldsymbol{q}_{0,3}(t)$,

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

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

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

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.

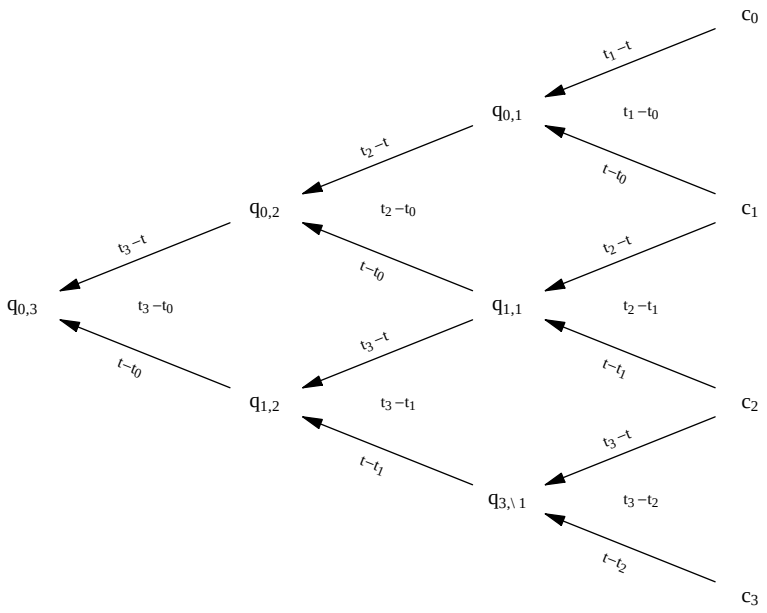


Figure 1.7. Computing a point on a cubic interpolating curve.

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

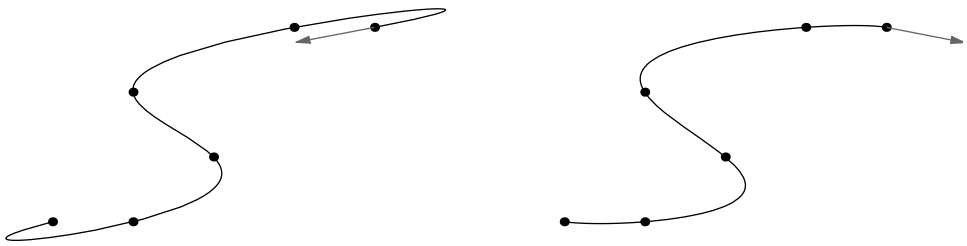
$$\boldsymbol{q}_{i,r}(t) = \frac{t_{i+r} - t}{t_{i+r} - t_i}\boldsymbol{q}_{i,r-1}(t) + \frac{t - t_i}{t_{i+r} - t_i}\boldsymbol{q}_{i+1,r-1}(t)$$

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?

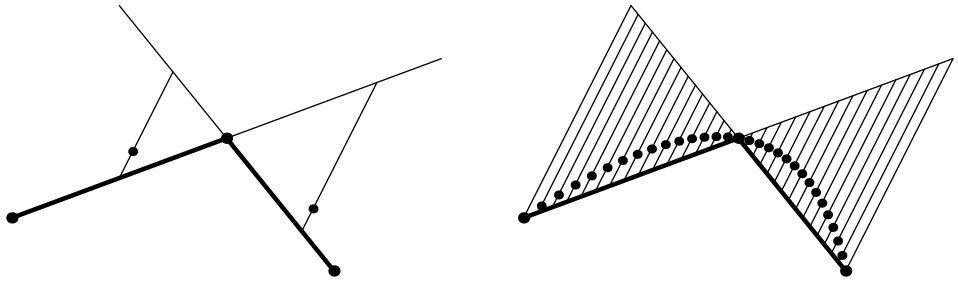
In the interpolation algorithm for polynomials of degree d, Algorithm 1.1, the last step is to form a convex combination between two polynomials of degree $d - 1$,

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

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

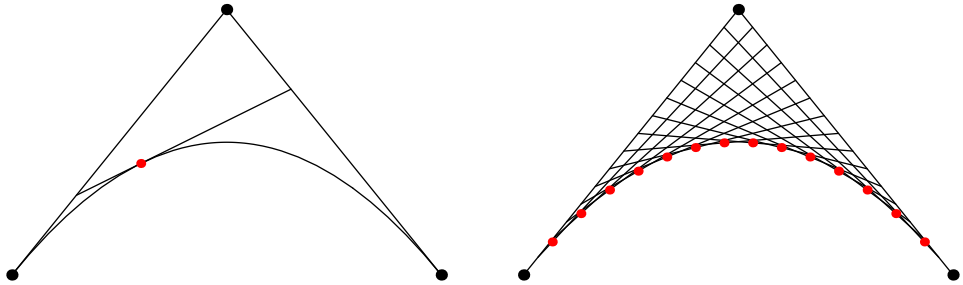
The curve construction method that we consider in this section is an alternative to polynomial interpolation and produces what we call *Bézier curves*. Bézier curves are also polynomial curves and for that reason not very practical, but they avoid the problem of wiggles and bulges because all computations 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,

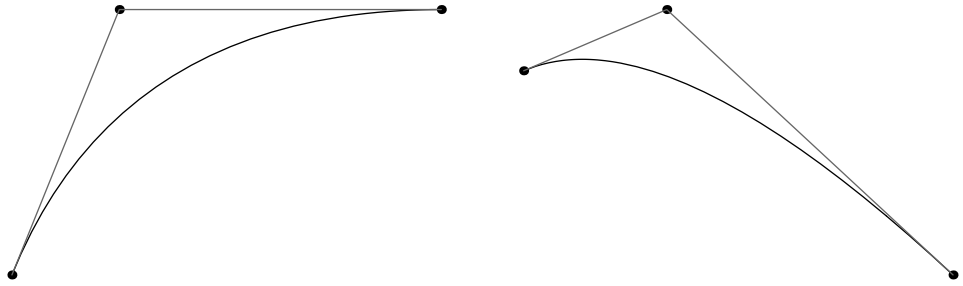
$$\boldsymbol{p}_{1,1}(t) = \boldsymbol{p}(t \mid \boldsymbol{c}_0, \boldsymbol{c}_1) = (1 - t)\boldsymbol{c}_0 + t\boldsymbol{c}_1, \tag{1.10}$$

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



(a) (b)

Figure 1.10. A Bézier curve based on three points.



(a) (b)

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

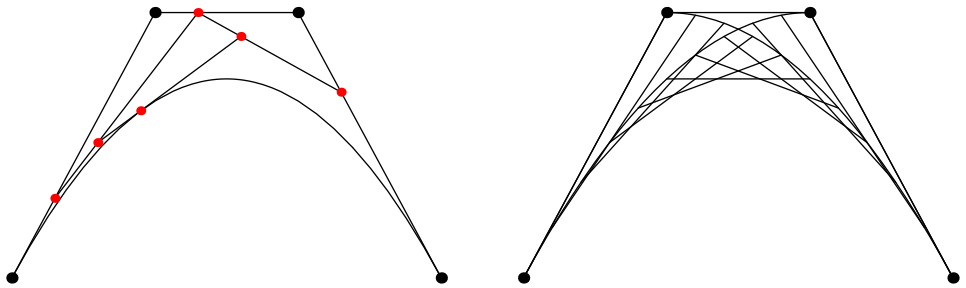


Figure 1.12. Constructing a Bézier curve from four points.

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

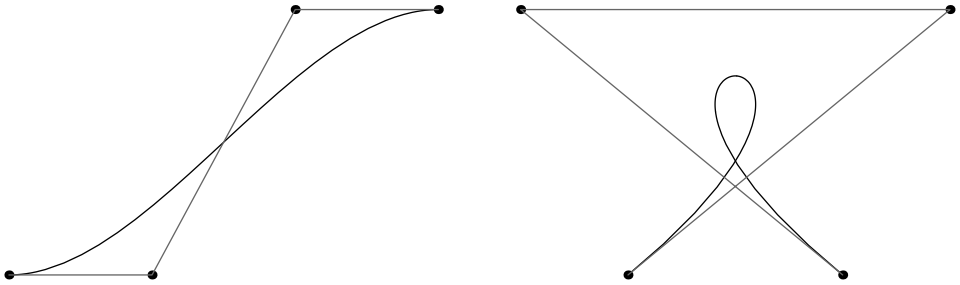


Figure 1.13. Two examples of cubic Bézier curves.

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 $r = 1, 2, \ldots, d$.

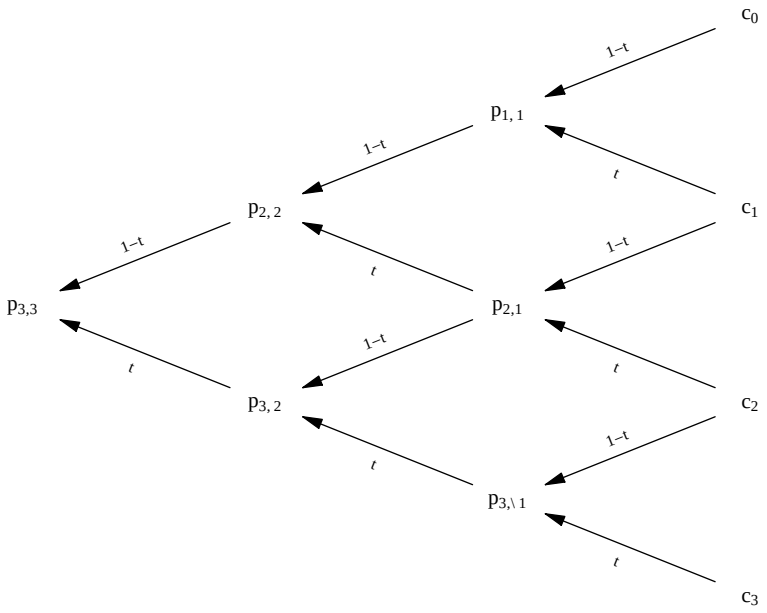


Figure 1.14. Computing a point on a cubic Bézier curve.

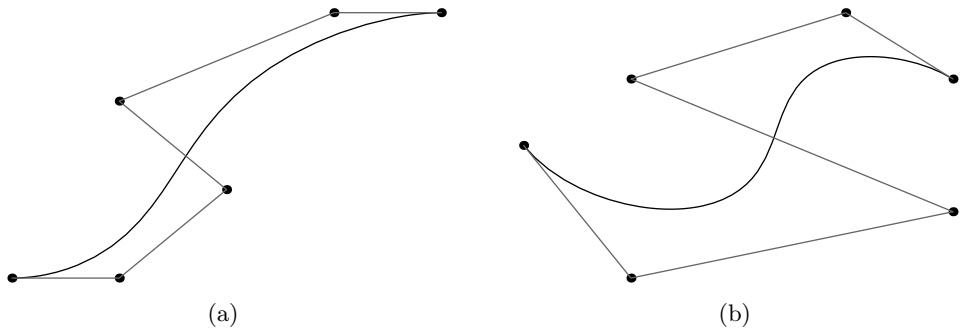


Figure 1.15. Two Bézier curves of degree five.

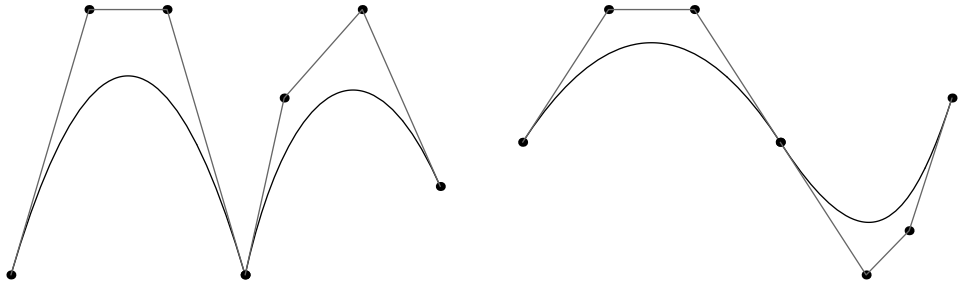


Figure 1.16. Different forms of continuity between two segments of a cubic Bézier curve.

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

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

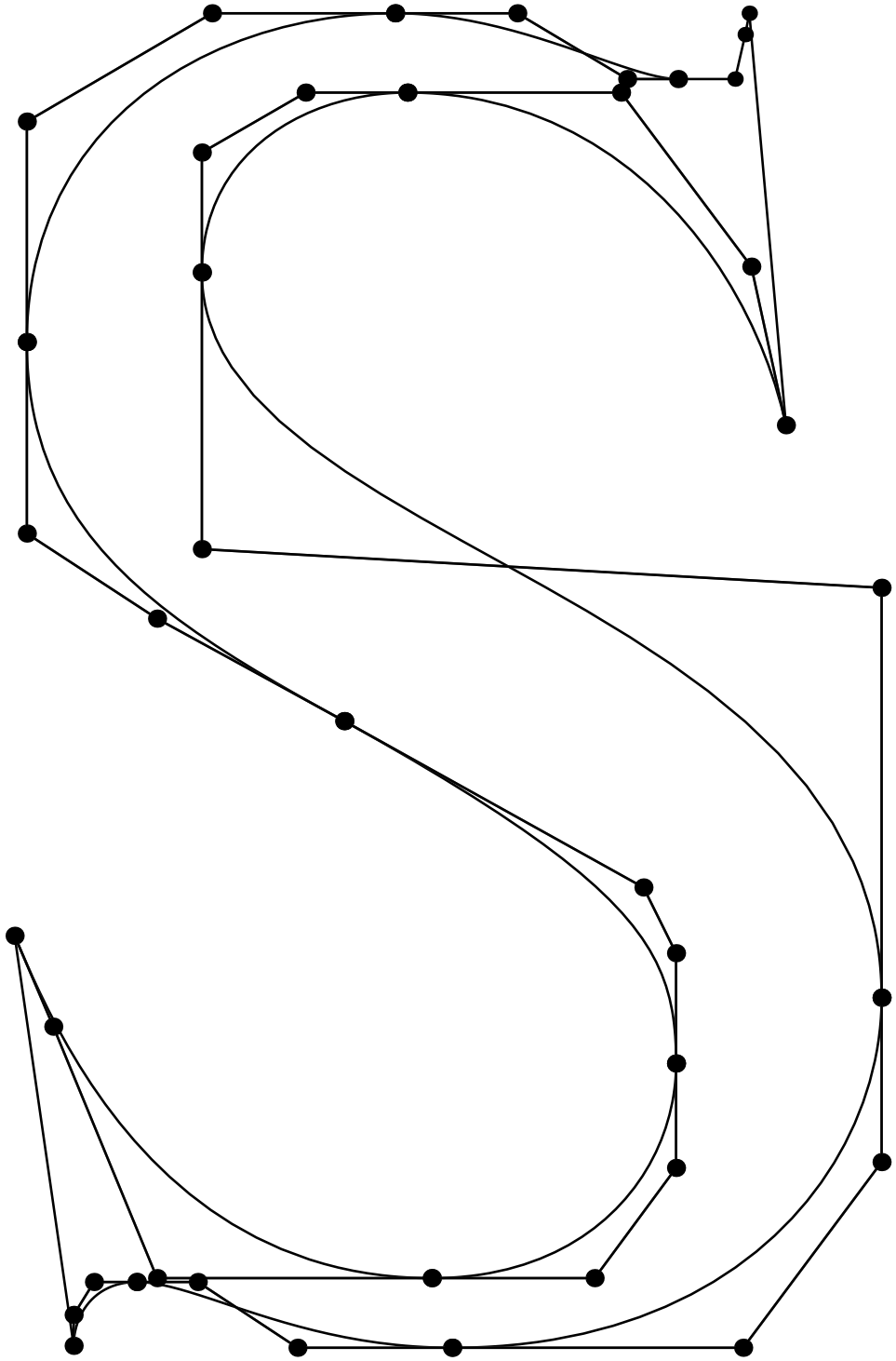


Figure 1.17. The letter S in the Postscript font Times Roman.

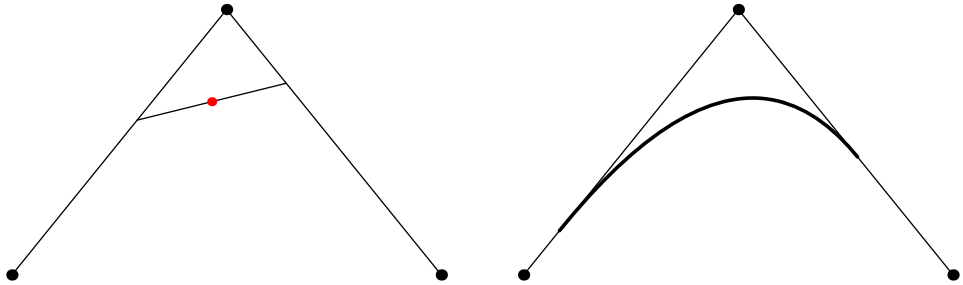


Figure 1.18. Construction of a segment of a quadratic spline curve.

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 $t_2 \le t_3 < t_4 \le t_5$. 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

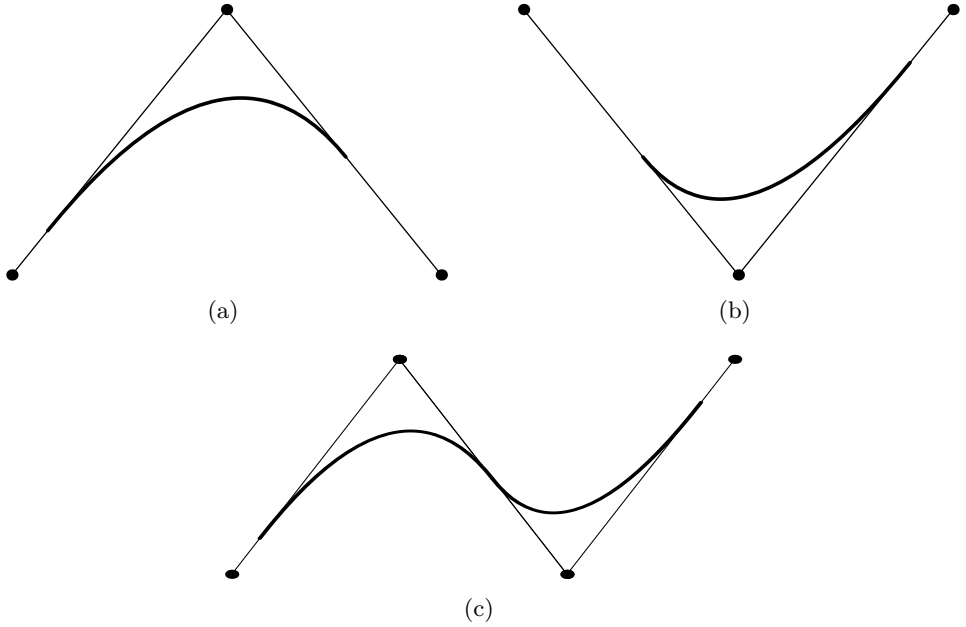


Figure 1.19. A quadratic spline curve (c) and its two polynomial segments (a) and (b).

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

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

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 $\boldsymbol{c}_{i-3}$, $\boldsymbol{c}_{i-2}$,

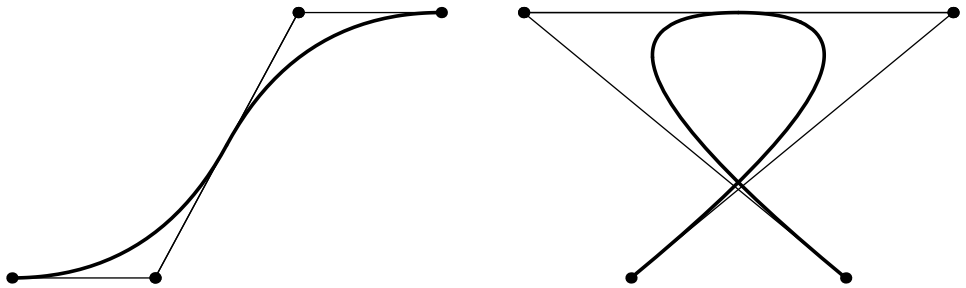


Figure 1.20. Two quadratic spline curves, both with knots $\boldsymbol{t} = (0, 0, 0, 1, 2, 2, 2)$.

$\boldsymbol{c}_{i-1}$, $\boldsymbol{c}_i$, 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,

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

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

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

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

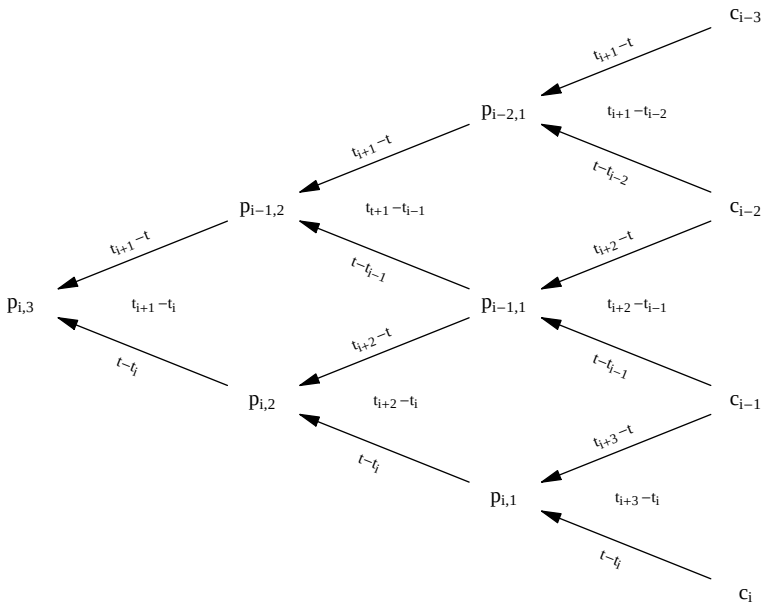


Figure 1.21. Computing a point on a cubic spline curve.

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

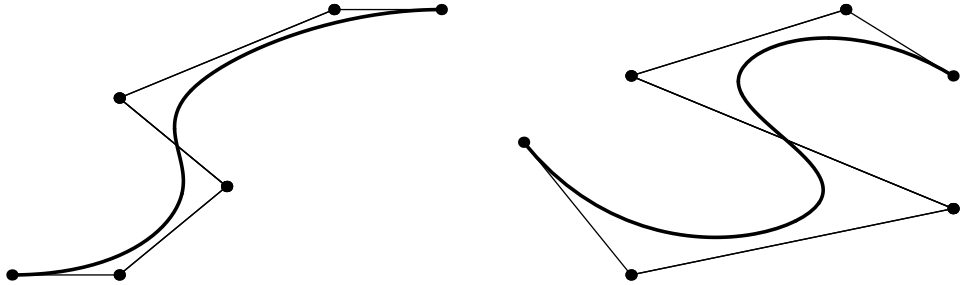


Figure 1.22. Two cubic spline curves, both with knots $\boldsymbol{t} = (0, 0, 0, 0, 1, 2, 3, 3, 3, 3)$.

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 $j = i-d$, $i-d+1$, ..., 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 $j = i-d+r$, ..., 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

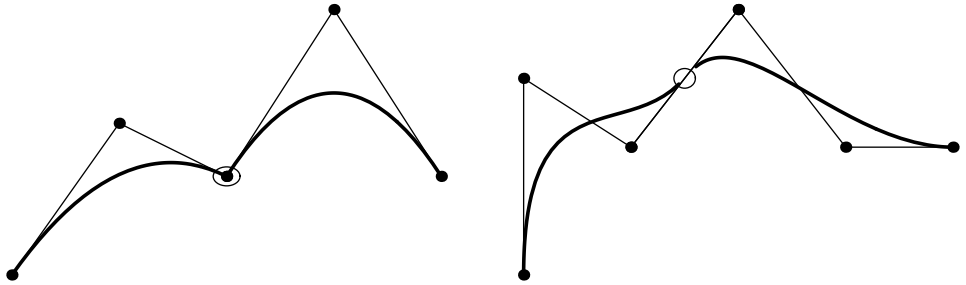


Figure 1.23. A quadratic spline with a double knot at the circled point (a) and a cubic spline with a double knot at the circled point (b).

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 $\boldsymbol{f}$, $\boldsymbol{f}'$, $\ldots$, $\boldsymbol{f}^{d-1}$ 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}, ..., t_{i+d+1}. Show also that $B_{i,d}(t) = 0$ if $t < t_i$ or $t > t_{i+d+1}$.

Index

combination
    affine, 4, 12, 90
    convex, 3–5, 7, 10, 13, 14, 16, 17, 22, 24, 25, 31–33, 92
convex hull, 3–6, 13, 14, 16, 17, 31–33, 75, 134
convex set, 5, 32, 33

interpolation
    cubic, 10
    general degree, 11
    Neville-Aitken algorithm, 12
    quadratic, 9

numerical stability, 3, 4, 26, 31, 76, 111, 163, 183, 191–196, 201

proof by induction, 29, 33, 34, 39–43, 53, 54, 69, 70, 86, 87, 209

round-off error, 3, 13, 19, 31, 114, 115, 164, 191, 193