Statistical Modeling with Spline Functions
Methodology and Theory

Mark H. Hansen
University of California at Los Angeles

Jianhua Z. Huang
University of Pennsylvania

Charles Kooperberg
Fred Hutchinson Cancer Research Center

Charles J. Stone
University of California at Berkeley

Young K. Truong
University of North Carolina at Chapel Hill

January 5, 2006

2 Preliminaries

This chapter provides a detailed description of the basic elements in numerical analysis that are especially useful for extended linear modeling. These elements consist of approximation spaces, splines, concave functions, and numerical optimization. While various descriptions of these elements are given in the numerical analysis and function approximation literature, we find that they are not specific enough to meet the needs of our treatment. The aim of this chapter is to provide a reasonably self-contained exposition of this background material.

Due to the important role of splines in the model building process throughout the monograph, the chapter starts with a short introduction to univariate splines. The exposition is based on a simple recursive definition for B-splines having distinct knots. Our main objective is to reveal the most important properties of B-splines as quickly as possible. Following this is the description of their properties in function approximation and a motivation for the choice of B splines over other spline basis functions. Section 3 presents multivariate splines based on the tensor product of univariate splines. Section 4 describes the notion of concavity and some of its remarkable consequences in function estimation. Concavity also helps in stabilizing the numerical procedures and otherwise in improving their performance. Section 5 will be important in understanding the numerical implementation of our methodology; it presents numerical optimization based on the Newton (or Newton–Raphson), quasi-Newton, and conjugate gradient algorithms. The final three sections of this chapter provide background material for handling free knot splines. Specificially, following the framework described in this monograph, one task in working with such splines is

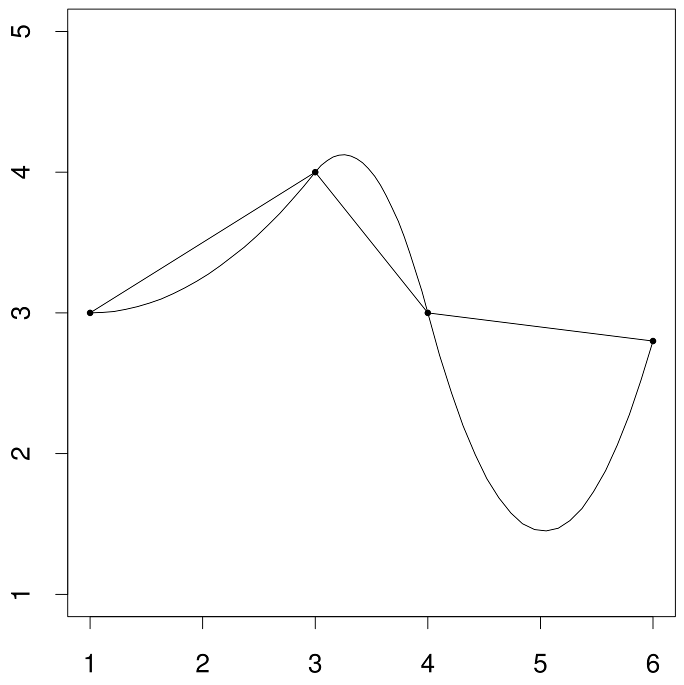

FIGURE 2.1. The linear and quadratic splines.

to compute the partial derivatives of the B-splines with respect to the knot locations. A systematic approach to this topic will call for a rigorous treatment of B-splines involving repeated knots, which has not been considered previously. In this respect, Section 6 is devoted to splines with repeated knots based on divided differences. Interestingly, this approach allows the splines to be constructed with more flexibility in terms of the smoothness of the curve. That is, the spline can have different degree of smoothness at different knot locations. Also, the computation of these splines is based on the recursive nature of B-splines, which was used to define the special case of B-splines with distinct knots. Consequently, there is no significant loss of computational efficiency. Section 7 discusses the continuity property of the divided difference, an essential topic for Section 8 in dealing with the computation of the partial derivatives of the B-splines with respect to the knot locations.

2.1 What is a Spline?

A spline is a curve formed by pasting several smaller pieces of curves together. Two such splines are shown in Figure 2.1. Splines were named by I. J. Schoenberg after the draftman's tool called spline — a thin flexible rod made of metal, plastic or wood held in place by weights for drawing smooth curves that go through certain indicated points. The formal definition of a spline requires

- the description of *knots*, which are the locations where the joining or gluing takes place;
- smaller curves used to form the larger one.

Since polynomials are the simplest among all the curves, it is natural to utilize them as building blocks for splines. For example, in Figure 2.1, the knots are defined by the set $\{1, 3, 4, 6\}$ in the horizontal axis and the smaller curves are constructed by either linear or quadratic polynomials. If the points are joined by the straight lines, the resulting curve is said to be a linear spline. If the points are joined by the quadratic curves, the resulting curve is a quadratic spline. It may be possible to have splines which are constructed by mixing linear and quadratic curves. For instance, one can use linear splines for extending the quadratic spline to the whole real line. Applications like these are plentiful in density and hazard estimation (see Chapters 6 and 7). Most splines in practice are constructed using cubic polynomials as we will see in later chapters. Nevertheless, all of the above are examples of splines, which will be introduced and discussed in greater detail in the next few sections.

2.1.1 Polynomials

Let k be a positive integer. A *polynomial of order* k is a function $p(\cdot)$ defined by

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1}, \qquad x \in \mathbb{R},$$

where $a_0, \ldots, a_{k-1}$ are real numbers. Thus a polynomial of order k has k coefficients, and the term having the highest degree is x^{k-1}.

A polynomial of order one is a constant function. Polynomials of order two, three and four are referred to as linear, quadratic and cubic polynomials, respectively. A polynomial of order k is also referred to as a polynomial of degree $k-1$. Thus, linear, quadratic and cubic polynomials are polynomials of degree one, two and three, respectively.

2.1.2 Piecewise polynomials

For a non-negative integer m and a list of distinct numbers $t_1 < t_2 < \cdots < t_m$, a *piecewise polynomial of order* k (or *of degree* $k-1$) having *knots* $t_1, t_2, \ldots, t_m$ is a function whose restriction to each of the $m-1$ intervals $[t_1, t_2), \ldots, [t_{m-1}, t_m)$is a polynomial of order k (or degree $k-1$) or less. A piecewise polynomial is usually defined on the whole real line or on an interval $[a, b]$ with $a < t_1 < t_2 < \cdots < t_m < b$ by simply extending both pieces at the end. (With this, the t_1 and t_m are not really knots but we continue to keep them in as part of the definition of piecewise polynomials.) Piecewise constant functions are the simplest of all piecewise polynomials. Piecewise polynomials of orders $k = 2, 3, 4$ are called piecewise linear, quadratic, and cubic polynomials, respectively. For example, Figure 2.1 presents the piecewise linear and quadratic polynomials having knots $1, 3, 4$ and 6.

The description of a piecewise polynomial of order k having m knots would require $k(m+1)$ coefficients since there are k coefficients in each of the $m+1$ polynomials.

2.1.3 Splines

A *spline function of order k* (or of degree $k-1$) having simple distinct *knots* t_1, $t_2, \ldots, t_m$ has the appearance shown in Figure 2.1. It is a $k-2$ (for the case $k \geq 2$) times continuously differentiable piecewise polynomial of order k with knots t_1, $t_2, \ldots, t_m$. For example, a spline function of order one is a piecewise constant function, a linear spline ($k = 2$) is a continuous piecewise linear function. A quadratic spline ($k = 3$) is a (continuously) differentiable piecewise quadratic polynomial. (See Figure 2.1.) A cubic spline ($k = 4$) is a piecewise cubic polynomial with continuous second derivatives.

A spline function of order k can be written in the following more convenient form:

$$s(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} + \sum_{i=1}^{m} b_i (x - t_i)_+^{k-1}, \qquad k \geq 2,$$

where $(a)_+$ denotes the nonnegative part of a number a: $(a)_+ = a$ for $a \geq 0$; 0 otherwise. For example, the linear, quadratic and cubic splines having the indicated knots are given by

$$s^1(x) = a_0 + a_1 x + \sum_{i=1}^{m} b_i (x - t_i)_+,$$

$$s^2(x) = a_0 + a_1 x + a_2 x^2 + \sum_{i=1}^{m} b_i (x - t_i)_+^2,$$

$$s^3(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \sum_{i=1}^{m} b_i (x - t_i)_+^3.$$

The coefficients presented in this form can be determined explicitly as a function of s. More specifically, we can express s as

$$s(x) = \sum_{i=0}^{k-1} \frac{s^{(i)}(t_1)}{i!} (x - t_1)^i + \sum_{j=2}^{m} \sum_{i=0}^{k-1} \frac{\Delta s^{(i)}(t_j)}{i!} (x - t_j)^i,$$

where $\Delta g(x) = g(x+) - g(x-)$ with $g(x\pm) = \lim_{\epsilon\to 0} g(x \pm \epsilon)$, $\epsilon > 0$. In fact, let $s(x)$ denote a spline function on $[a, b]$ whose restriction to the first interval $[a, t_2)$ is the polynomial given by

$$p(x) = s(t_1)+s^{(1)}(t_1)(x-t_1)+\cdots+\frac{s^{(k-1)}(t_1)}{(k-1)!}(x-t_1)^{k-1} = \sum_{i=0}^{k-1} \frac{s^{(i)}(t_1)}{i!}(x-t_1)^i.$$

Then $s(x) - p(x)$ vanishes for $x < t_2$ and satisfies, by the definition of splines,

$$s^{(j)}(t_2) - p^{(j)}(t_2) = 0, \qquad j = 0, 1, \ldots, k-2.$$

Since $s - p$ is also a polynomial on the interval $[t_2, t_3)$, the above implies that $s(x) - p(x) = b_1(x - t_1)^{k-1}$ on $[t_2, t_3)$. Here b_1 satisfies $s^{(k-1)}(t_2+) - p^{(k-1)}(t_2+) = (k-1)!b_1$, or $b_1 = [s^{(k-1)}(t_2+) - s^{(k-1)}(t_2-)]/(k-1)!$ since $p^{(k-1)}(t_2+) = p^{(k-1)}(t_2-) = s^{(k-1)}(t_2-)$. Thus

$$s(x) - p(x) = \begin{cases} 0, & x < t_2, \\ \frac{\Delta s^{(k-1)}(t_2)}{(k-1)!}(x - t_2)^{k-1}, & t_2 \le x < t_3, \end{cases}$$

so

$$\begin{aligned} s(x) &= p(x) + \frac{\Delta s^{(k-1)}(t_2)}{(k-1)!}(x - t_2)_+^{k-2} \\ &= \sum_{i=0}^{k-1} \frac{s^{(i)}(t_1)}{i!}(x - t_1)^i + \frac{\Delta s^{(k-1)}(t_2)}{(k-1)!}(x - t_2)_+^{k-1}, \quad x < t_3. \end{aligned}$$

Repeat this argument to obtain

$$\begin{aligned} s(x)-p(x)- \frac{\Delta s^{(k-1)}(t_2)}{(k-1)!}(x-t_2)_+^{k-1} - \cdots - \frac{\Delta s^{(k-1)}(t_{m-2})}{(k-1)!}(x-t_{m-2})_+^{k-1} \\ = \begin{cases} 0, & x < t_{m-1}, \\ \frac{\Delta s^{(k-1)}(t_{m-1})}{(k-1)!}(x - t_{m-1})^{k-1}, & t_{m-1} \le x. \end{cases} \end{aligned}$$

In describing such a spline function, we need $m+k$ coefficients: $a_0, a_1, \ldots,$ $a_{k-1}, b_1, b_2, \ldots, b_m$. This can be confirmed by noting that there are $(m+1)k$ coefficients from the polynomials, less $m(k-1)$ continuity constraints at the interior knots. Thus the splines of order k with the indicated knots form a linear space of dimension $m + k$. A basis for this space is provided by

$$1, x, x^2, \ldots, x^{k-1}, (x - t_1)_+^{k-1}, \ldots, (x - t_m)_+^{k-1}.$$

This is usually referred to as the *truncated power basis*.

Although splines can be expressed directly in terms of the truncated power basis, it is known that such a representation can be highly ill-conditioned. Consider, for example, the truncated power basis

$$1, x, x^2, x^3, x_+^3, (x - 1)_+^3, (x - 2)_+^3$$

for the space of cubic splines having knots at 0, 1, and 2. On the interval $[-1, 3]$, the basis function x^3 can very accurately be approximated by a linear combination of the remaining basis functions. That is, any attempt to fit a linear regression using the above truncated power basis functions is likely to run into the colinearity problem. See also de Boor (1978, p.104). We will present in the next section another basis, consisting of B-splines. In addition to possessing excellent numerical properties, B-splines also provide a powerful tool for establishing theoretical properties.

2.1.4 B-splines: Definition

There are several ways to introduce B-splines. In this section, we consider the approach starting with the recursive relationship as described in Kincaid and Cheney (1996, p. 392). The approach has the advantage of being simpler and making it easier to understand the most important properties of B-splines. The properties to be developed here are justified based on the assumption that the knots are distinct. The numerical procedure, however, can be easily modified to accommodate repeated knots. A more general approach to the construction of B-splines in which repeated knots are allowed will be given in Section 2.6. Repeated knots occur in connection with free knot splines, which are discussed in Chapter 12.

Consider the doubly infinite knot sequence $\cdots < t_{-2} < t_{-1} < t_0 < t_1 < t_2 < \cdots$. We start with the piecewise constant splines and use them as building blocks for higher order splines.

B-splines of order 1

The simplest B-splines have the appearance shown in Figure 2.2 and are defined by

$$B_i^1(x) = 1_{[t_i, t_{i+1})}(x) = \begin{cases} 1, & t_i \le x < t_{i+1}, \\ 0, & \text{otherwise.} \end{cases}$$

Figure 2.2 presents a sequence of order-one B-splines having knots $0, 1, 3, 4, 7, 8, 9, 10$. We observe that they have the following properties:

1. The support of B_i^1 is $[t_i, t_{i+1})$; that is, it takes two knots to construct a B-spline of order one,
2. B_i^1 is non-negative for all x and i,
3. B_i^1 is continuous from the right on the entire line,
4. $\sum_i B_i^1(x) = 1$ for all x.

These splines will be used to construct B-splines of order two: the linear B-splines.

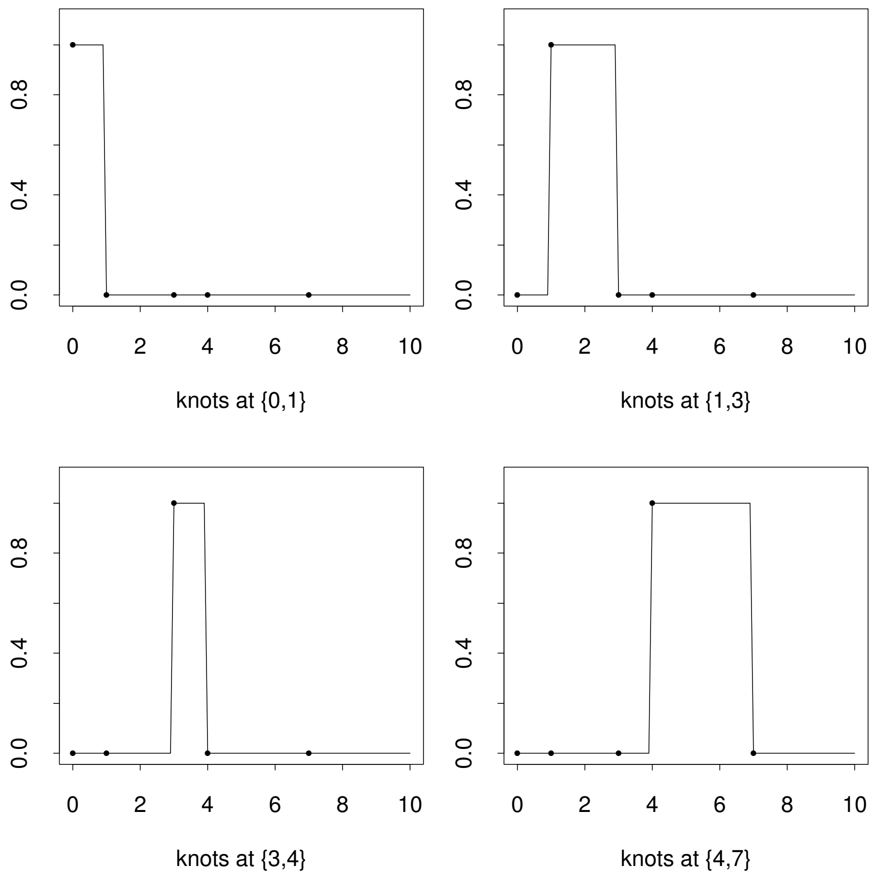

FIGURE 2.2. A sequence of order-one B-splines having knots $\{0, 1, 3, 4, 7\}$.

B-splines of order 2

The linear B-splines, denoted by $B_i^2(\cdot)$, are constructed recursively from those of order one according to

$$B_i^2(x) = \frac{x - t_i}{t_{i+1} - t_i} \cdot B_i^1(x) + \frac{t_{i+2} - x}{t_{i+2} - t_{i+1}} \cdot B_{i+1}^1(x), \quad i = 0, \pm 1, \pm 2, \ldots.$$

That is,

$$B_i^2(x) = \begin{cases} \dfrac{x - t_i}{t_{i+1} - t_i}, & t_i \le x \le t_{i+1}, \\ \dfrac{t_{i+2} - x}{t_{i+2} - t_{i+1}}, & t_{i+1} \le x \le t_{i+2}, \end{cases} \qquad i = 0, \pm 1, \ldots,$$

and $B_i^2(x) = 0$ for $x < t_i$ and for $x > t_{i+2}$.

Figure 2.3 presents a sequence of linear B-splines with knots $0, 1, 3, 4, 7, 8, 9, 10$. Note that it takes three knots to construct a linear B-spline. More

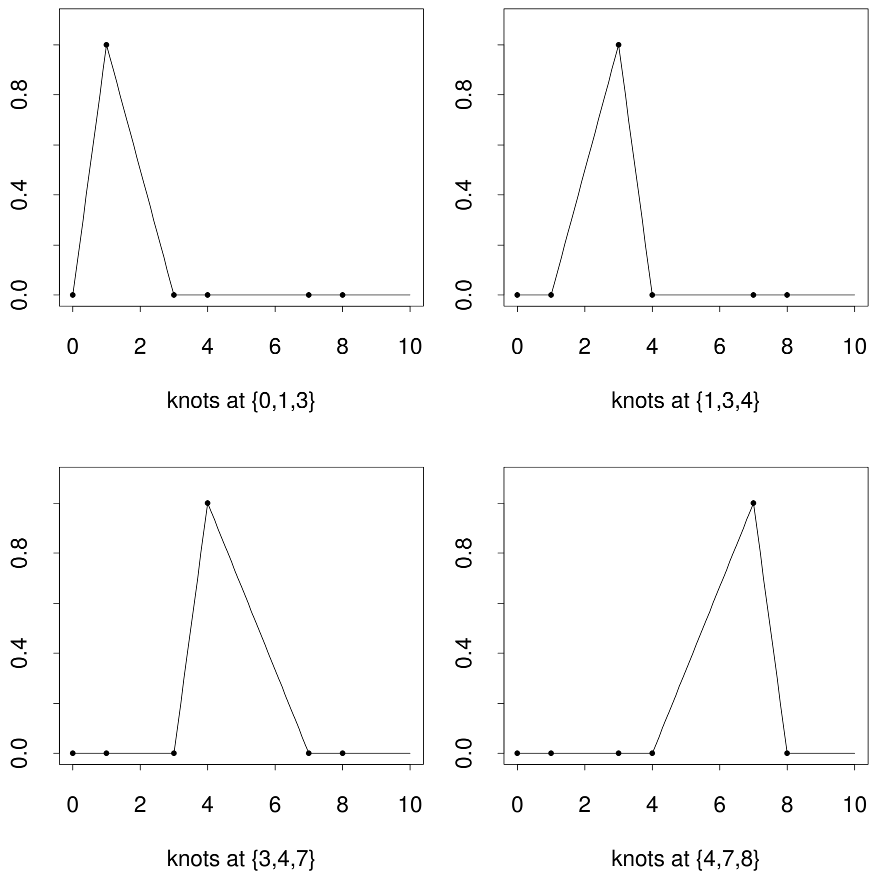

FIGURE 2.3. A sequence of linear B-splines having knots $\{0, 1, 3, 4, 7, 8\}$.

generally, we will show shortly that linear B-splines possess the following properties:

1. The support of B_i^2 is (t_i, t_{i+2});
2. B_i^2 is non-negative for all x and i;
3. B_i^2 is continuous on the entire line;
4. $\sum_i B_i^2(x) = 1$ for all x.

We observe that linear B-splines have a wider support than the piecewise constant ones. Moreover, they are unimodal.

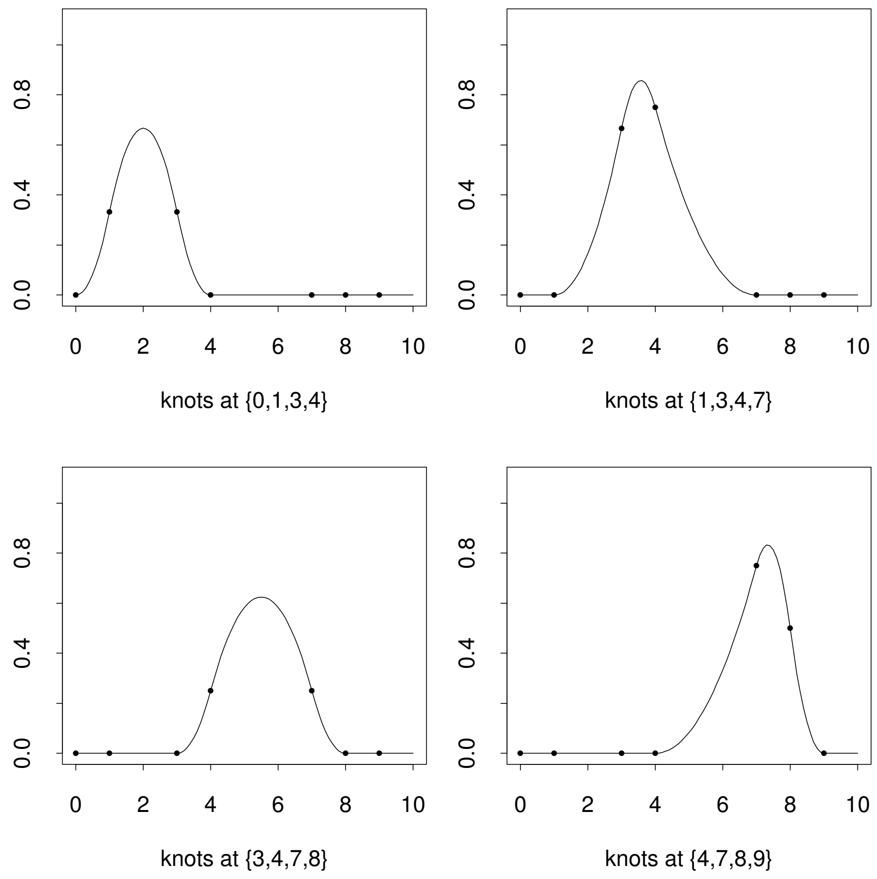

FIGURE 2.4. A sequence of quadratic B-splines having knots $\{0, 1, 3, 4, 7, 8, 9\}$.

B-splines of order k

In general, kth-order B-splines can be constructed recursively according to

$$B_i^k(x) = \frac{x - t_i}{t_{i+k-1} - t_i} \cdot B_i^{k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} \cdot B_{i+1}^{k-1}(x),$$
$$k = 2, 3, \ldots, \quad i = 0, \pm 1, \pm 2, \ldots.$$

For examples, quadratic and cubic B-splines are constructed using the above formula by setting $k = 3, 4$, respectively. Based on the same knots considered previously, Figures 2.4 and 2.5 present a sequence of quadratic and cubic B-splines. In these figures, we see that it takes two, three, four and five knots to construct a B-splines of orders one, two, three and four, respectively.

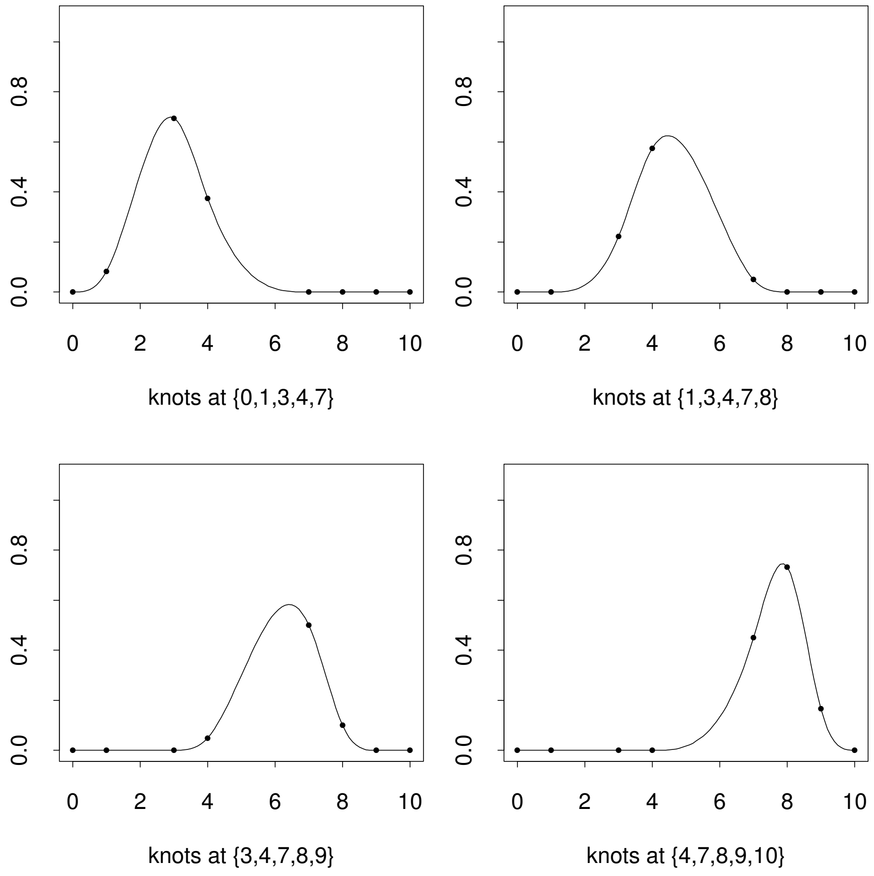

FIGURE 2.5. A sequence of cubic B-splines having knots $\{0, 1, 3, 4, 7, 8, 9, 10\}$.

2.1.5 Important properties of B-splines

Since B-splines will be used throughout the monograph as a basis for building extended linear models, it is essential to understand their properties. This section compiles a list of the most important and relevant properties of B-splines that are used in this monograph. The reader is referred to de Boor (1978), DeVore and Lorentz (1993), and Schumaker (1981) for more in-depth treatments.

1. Support of B-splines: From the above examples, we have observed that the supports of B-splines increase with the order. More formally, starting with $B_i^1 = 1_{[t_i,t_{i+1})}$,

$$B_i^k(x) = 0 \quad \text{for} \quad x \notin (t_i, t_{i+k}), \qquad k > 1, \tag{2.1.1}$$

and

$$B_i^k(x) > 0 \quad \text{for} \quad x \in (t_i, t_{i+k}), \qquad k > 1. \tag{2.1.2}$$

In fact, it is easy to see from the recursive formula that (2.1.1) is valid for $k = 2$. We next show that it holds for the general case by an inductive argument. Suppose it is true for $k > 1$. If x is outside (t_i, t_{i+k+1}), then it is outside (t_i, t_{i+k}), and (t_{i+1}, t_{i+k+1}). By the inductive hypothesis, $B_i^k(x) = B_{i+1}^k(x) = 0$. Hence, $B_i^{k+1}(x) = 0$ according to the recursive definition. The second statement follows similarly.

This property shows that for a given x, there are only finitely many nonzero B-splines at x. More specifically, only the k B-splines B_{i-k+1}^k, $B_{i-k+2}^k, \ldots, B_i^k$ might be nonzero on $[t_i, t_{i+1})$.

2. Given a sequence of numbers $\{c_i^k\}$, define

$$c_i^{k-1} = c_i^k \frac{x - t_i}{t_{i+k-1} - t_i} + c_{i-1}^k \frac{t_{i+k-1} - x}{t_{i+k-1} - t_i}, \qquad k \geq 1, \quad i = 0, \pm 1, \ldots.$$

Set $f(x) = \sum_i c_i^k B_i^k(x)$. Then

$$\begin{aligned} f(x) &= \sum_i c_i^k \left[\frac{x - t_i}{t_{i+k-1} - t_i} \cdot B_i^{k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} \cdot B_{i+1}^{k-1}(x) \right] \\ &= \sum_i \left[c_i^k \frac{x - t_i}{t_{i+k-1} - t_i} + c_{i-1}^k \frac{t_{i+k-1} - x}{t_{i+k-1} - t_i} \right] B_i^{k-1}(x) \\ &= \sum_i c_i^{k-1} B_i^{k-1}(x) \\ &= \cdots \\ &= \sum_i c_i^1 B_i^1(x). \end{aligned}$$

This provides an efficient way to evaluate $f(x)$ since $f(x) = c_i^0$ for $t_i \leq x < t_{i+1}$. The evaluation is usually carried out using the following arrangement:

$$\begin{array}{cccc} c_i^k & c_i^{k-1} & \cdots & c_i^1 \\ c_{i-1}^k & c_{i-1}^{k-1} & \cdots & \\ \vdots & \cdots & & \\ c_{i-k+1}^k & & & \end{array}$$

3. $\sum B_i^k(x) = \sum_{i=j+1-k}^{j} B_i^k(x) = 1$ for $x \in [t_j, t_{j+1}]$. In fact, from the above recursive definition of c_i^{k-1}, with $c_i^k = 1$ for $i = 0, \pm 1, \pm 2, \ldots$,

$$\begin{aligned} c_i^{k-1} &= c_i^k \frac{x - t_i}{t_{i+k-1} - t_i} + c_{i-1}^k \frac{t_{i+k-1} - x}{t_{i+k-1} - t_i} \\ &= 1 \cdot \frac{x - t_i}{t_{i+k-1} - t_i} + 1 \cdot \frac{t_{i+k-1} - x}{t_{i+k-1} - t_i} = 1 \qquad \text{for all } i. \end{aligned}$$

It follows from induction that $c_i^j = 1$ for all i and $j = k, k-1, \ldots, 0$.

4. Derivatives of B-splines:

$$\frac{d}{dx} B_i^k(x) = \frac{k-1}{t_{i+k-1} - t_i} B_i^{k-1}(x) - \frac{k-1}{t_{i+k} - t_{i+1}} B_{i+1}^{k-1}(x).$$

This property can be verified directly by using the recursive definition. [See Kincaid and Cheney (1996, p.398) and Section 2.6.6.] The result will be needed in showing that certain B-splines are linearly independent.

5. The set $\{B_{j-k+1}^k, B_{j-k+2}^k, \ldots, B_j^k\}$ of B-splines is linearly independent on (t_j, t_{j+1}).

The result is clearly true for $k = 1$. Supposing it is true for $j = k$, we will show that it is also true for $j = k + 1$. Set $s = \sum_{i=0}^k c_i B_{j-k+i}^{k+1}$ and suppose that $s = 0$ on (t_j, t_{j+1}). Then

$$0 = s' = k \sum_{i=1}^k \frac{c_i - c_{i-1}}{t_{j+i} - t_{j-k+i}} B_{j-k+i}^k \qquad \text{on } (t_j, t_{j+1}).$$

(Note that $B_{j-k}^k = 0 = B_{j+1}^k$ on (t_j, t_{j+1}).) By the inductive hypothesis, the set $\{B_{j-k+1}^k, B_{j-k+2}^k, \ldots, B_j^k\}$ is linearly independent on the interval (t_j, t_{j+1}), so $c_0 = c_1 = \cdots = c_k$. Denote this common value by c. Then $s(x) = c$ for all x. This implies that $c = 0$ since $s(x) = 0$ for all x.

6. The set $\{B_{-k+1}^k, B_{-k+2}, \ldots, B_m^k\}$ of B-splines is linearly independent on (t_0, t_{m+1}).

Set $s = \sum_{i=-k+1}^m c_i B_i^k$ and suppose that $s = 0$ on (t_0, t_{m+1}). By the above property, $\{B_{-k+1}^k, B_{-k+1}^k, \ldots, B_0^k\}$ is linearly independent on (t_0, t_1), so $c_i = 0$ for $i = -k+1, \ldots, 0$. Suppose there exists the first $j \geq 1$ such that $c_j \neq 0$. Then for any $x \in (t_j, t_{j+1}) \subset (t_0, t_{m+1})$,

$$0 = s(x) = \sum_{i=j}^m c_i B_i^k(x) = c_j B_j^k(x) \neq 0,$$

a contradiction. We conclude that $c_i = 0$ for $i = -k+1, \ldots, m$.

7. On the interval $[t_0, t_{m+1}]$, the set $\{B_{-k+1}^k, B_{-k+2}, \ldots, B_m^k\}$ of B-splines forms a basis for the space of all k-th order splines having knots $t_1, t_2, \ldots, t_m$. That is, every k-th order splines s on $[t_0, t_{m+1}]$ can be written as $s = \sum_{i=1}^{k+m} c_i B_{-k+i}^k$.

We first show that the dimension of the space of k-th order splines on $[t_0, t_{m+1}]$ is at most $k + m$ by exhibiting a set of $k + m$ functions that span that space. Recall that a spline function of order k defined on $[t_0, t_{m+1}]$ can be written as

$$s(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} + \sum_{i=1}^{m} b_i (x - t_i)_+^{k-1}.$$

Thus the truncated power basis

$$1, x, x^2, \ldots, x^{k-1}, (x - t_1)_+^{k-1}, \ldots, (x - t_m)_+^{k-1}$$

spans the space of splines of order k defined on $[t_0, t_{m+1}]$. The desired result follows from Property 6.

Property 7 is often used to motivate the use of the term B-splines, B for "basis". This property is very important, for a basis consisting of B-splines yields more efficient and stable numerical procedures for working with splines than does a truncated power basis.

8. B-splines of order two or higher are unimodal. See, for example, de Boor (1978, p. 157).

2.2 Function Approximation

Even though we deal with noisy data in statistics, it is essential to understand how a function is approximated from noisy-free data and to quantify the properties of this approximation. The later usually enters the picture of estimation through the bias term.

2.2.1 Properties of Polynomial Approximation

If the function being approximated is not continuous, there may be no agreement at any given point except at the interpolating points. For example, the Dirichlet function f is defined to be 1 for irrationals and zero otherwise. If the interpolating points are chosen to be rationals, then the interpolating polynomial is given by $p(x) = 0$ for all x, but $f(x) - p(x) = 1$ for irrational x. Thus continuity is essential in determining the quality of approximation. But strange things can also happen even when $f(\cdot)$ is continuous and possess continuous derivatives of all orders. Consider the Runge function:

$$f(x) = 1/(1 + x^2)$$

on the interval $[-5, 5]$. If we interpolate f using the polynomial p_k of order k based on k equally spaced points. Then

$$\lim_{k \to \infty} \max_{-5 \le x \le 5} |f(x) - p_k(x)| = \infty.$$

This example illustrates the danger of using high order polynomials in approximation: requiring a polynomial to agree with the function at many nodal points can significantly increase the error at non-nodal points.

Nevertheless, the error of interpolation can be described as follows. If p_k interpolates f at the k distinct nodes $x_1, x_2, \ldots, x_k$ in $[a,b]$ and if the k-th derivative $f^{(k)}$ is continuous, then

$$|f(x) - p_k(x)| \leq \frac{1}{4k} \max_{a \leq x \leq b} |f^{(k)}(x)| \Big(\frac{b-a}{k-1}\Big)^k, \qquad k > 1.$$

2.2.2 *Why Splines?*

It is well known that splines have better approximation properties than polynomials. In fact, the above Runge function has been shown to be better approximated by spline functions. To appreciate this statement without going too deeply into the details of function approximation, the most effective tool is Jackson's theorem on the distance of a function to the space of polynomials. We start with a few definitions.

The *modulus of continuity* of a function f on $[a,b]$ is defined by

$$\omega(f;\delta) = \max\{|f(x) - f(y)| : |x-y| \leq \delta, \quad x,y \in [a,b]\}, \qquad \delta > 0.$$

The distance from f to the space of polynomials P_k of order at most k is defined by

$$\operatorname{dist}(f, P_k) = \inf\{\|f-g\|_\infty : g \in P_k\},$$

where

$$\|f\|_\infty = \sup_x |f(x)|.$$

Suppose f possesses r continuous derivatives on the interval $[a,b]$, and $k > r+1$. It can be shown that there is a constant C_r depending on r such that

$$\operatorname{dist}(f, P_k) \leq C_r \Big(\frac{b-a}{k-1}\Big)^r \omega\Big(f^{(r)}, \frac{b-a}{2(k-1-r)}\Big).$$

The above result is referred to as Jackson's theorem. See, for example, Rivlin (1969), p. 23. It is known that this bound is sharp (de Boor, p. 34). The only way to reduce the error is to shrink the factor $(b-a)/(k-1)$ by either making $(b-a)$ smaller or increasing the order k of the approximating polynomial. The latter option is not desirable as discussed in the last section. Since the interval $[a,b]$ is given, the only way to reduce the error is to partition it into smaller intervals and apply polynomial approximation on each of these intervals.

Using m partitioned intervals has the same effect as utilizing the polynomial of order km as far as the bound is concerned. This amounts to a m-fold increase in the degrees of freedom. But the degrees of freedom enter the picture of approximation differently. Evaluating a polynomial of order km requires km coefficients and basis functions whose complexity increases with km, while the evaluation of a piecewise polynomial function of order k and m segments involves only $k+1$ coefficients and a local basis of fixed complexity regardless of the value of m. This structural difference also strongly influences the construction of approximations, which requires the solution of a full system involving polynomials and usually only a banded system in the case of piecewise polynomials.

2.2.3 Why B-splines?

It was demonstrated in Section 2.1.5 that a basis for the space of splines of order k having knots $t_1, t_2, \ldots, t_m$ is provided by B-splines of order k constructed with the indicated knots. That is, any spline of order k can be expressed as a linear combination of B-splines.

In addition to being a useful and powerful tool in spline theory, B-splines have another desirable property in that their support consists of a small fixed, finite number of intervals between knots. The practical implication is that these basis functions provide a stable numerical method for evaluating splines. For example, in the regression context, the B-spline representation yields a banded design matrix which is more efficient in computation than the design matrix corresponding to the truncated power basis.

Moreover, to describe and explain the data better, it is necessary to modify the construction of the basis functions by tailoring them to certain features of the problem being considered. This can be carried out effectively using the B-spline approach, both numerically and methodologically. For examples, to consider the tail behavior of the density estimate as described in Chapters 6, it will be useful to modify the basis by utilizing cubic splines with linear tails. To enhance its flexibility in fitting various hazard functions in survival analysis considered in Chapter 7, special bases are introduced to describe the behavior near the time point at zero as well as the tail. In handling the estimation of line spectra involving stationary time series analysis, special basis functions are needed in estimating the spectral density function of time series, which is discussed in Chapter 8.

2.2.4 Distance from a function to a spline space

One of our objectives is to recover the function of interest from a set of noisy data by using spline approximation. The performance of this procedure is normally quantified in terms of the bias and the variance of the estimate. In this section, we will consider the description of the former by presenting one of the numerical properties of splines in approximating functions. Specifically, we will establish the fact that the distance between the function of interest and the space of polynomial splines is dominated by the size of the inter-distances between knots. This distance will be described in later chapters of this monograph as the bias of the spline based estimate.

Let f denote a real-valued function defined on the interval $[a, b]$. For a non-negative integer m and a list of distinct numbers $a = t_0 < t_1 < t_2 < \cdots < t_m < t_{m+1} = b$, the space S_m^k of kth order polynomial splines on $[a, b]$ with the knot sequence $\{t_1, t_2, \ldots, t_m\}$ will be described in terms of the kth order B-splines $\{B_i^k\}$ as discussed in Section 2.1.4 using the doubly infinite knot sequence $\{t_i, i = 0, \pm 1, \pm 2, \ldots\}$. According to Property 7 of Section 2.1.5, $\{B_{-k+1}^k, B_{-k+2}^k, \ldots, B_m^k\}$ is a basis for the linear space S_m^k.

Let $\delta = \max\{|t_{j+1} - t_j| : j = 0, 1, \ldots, m\}$ denote the size of the inter-distances between knots. In order to find an upper bound to $\operatorname{dist}(f, S_m^k)$, the distance from the function f to the space S_m^k, we need a simple property concerning the modulus of continuity $\omega(f; \delta)$: If f is differentiable and $|f'| \le M$ on $[a, b]$, then by the mean value theorem,

$$\omega(f; \delta) \le \delta \sup_{a \le x \le b} |f'(x)| \le M\delta.$$

Now we proceed to establish such a bound by starting with a key member of S_m^k. Choose any point x_i from the intersection of the support of B_i^k with $[a, b]$, $i = -k + 1, -k + 2, \ldots, m$, and set

$$s = \sum_{i=-k+1}^{m} f(x_i) B_i^k \in S_m^k.$$

To see the role of this element, we take a point z from one of the intervals $[t_j, t_{j+1}]$ in $[a, b]$. Property 1 of Section 2.1.5 implies that z belongs to the supports only of $B_{j-k+1}^k, \ldots, B_j^k$ and $|z - x_i| \le k\delta$ for $i = j - k + 1, \ldots, j$. It follows from Properties 1 and 3 of Section 2.1.5 that

$$\begin{aligned} |s(z) - f(z)| &= \left| \sum [f(x_i) - f(z)] B_i^k(z) \right| \\ &\le \sum_{i=j-k+1}^{j} |f(x_i) - f(z)| B_i^k(z) \\ &\le \omega(f, k\delta). \end{aligned}$$

Since z is arbitrary, we have $\|s - f\|_\infty \le \omega(f, k\delta)$. Therefore,

$$\operatorname{dist}(f, S_m^k) \le \omega(f, k\delta). \tag{2.2.1}$$

For a continuously differentiable function f on $[a, b]$ and $s \in S_m^k$, using the simple property of the modulus of continuity mentioned above and (2.2.1), we get that

$$\operatorname{dist}(f, S_m^k) = \operatorname{dist}(f - s, S_m^k) \le \omega(f - s, k\delta) \le k\delta \|f' - s'\|_\infty.$$

As s ranges over S_m^k, s' ranges over S_m^{k-1}. By taking the infimum over s, we conclude that

$$\operatorname{dist}(f, S_m^k) \le k\delta \cdot \operatorname{dist}(f', S_m^{k-1}).$$

Suppose $r < k < m$. If $f \in C^r[a,b]$, then by repeating the above argument $(r-2)$ times, we have

$$\begin{aligned}\mathrm{dist}(f, S_m^k) &\le k(k-1)\cdots(k-r+2)\delta^{r-1}\mathrm{dist}(f^{(r-1)}, S_m^{k-r+1}) \\ &\le k(k-1)\cdots(k-r+2)\delta^{r-1}\omega(f^{(r-1)}, (k-r+1)\delta) \\ &\le k(k-1)\cdots(k-r+1)\delta^{r}\|f^{(r)}\|_\infty.\end{aligned}$$

This result indicates that the bias of the splines based estimates is $\mathrm{dist}(f, S_m^k)$, and is dominated by the size of the inter-distance δ of the knots.

2.3 Tensor products of splines

Up to now splines have been discussed for one dimensional data. It is possible to define splines in higher dimensions. There are multivariate version of B-splines, which are piecewise polynomials having compact support. Typically, we will use tensor products of B-splines. See Chapter 9 for a different approach.

2.3.1 Tensor products of linear spaces

Given functions g_1 on $\mathcal{X}_1$ and g_2 on $\mathcal{X}_2$, their tensor product $g_1 \otimes g_2$ is the function $g_1(x_1)g_2(x_2)$ on $\mathcal{X}_1 \times \mathcal{X}_2$. Given linear spaces $\mathbb{G}_1$ and $\mathbb{G}_2$ of functions on $\mathcal{X}_1$ and $\mathcal{X}_2$, respectively, their tensor product $\mathbb{G}_1 \otimes \mathbb{G}_2$ is the linear space spanned by $g_1 \otimes g_2$ as g_1 runs over $\mathbb{G}_1$ and g_2 runs over $\mathbb{G}_2$.

More generally, given functions $g_1, g_2, \ldots, g_M$ on $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$ respectively, their tensor $g_1 \otimes \cdots \otimes g_M$ is the function $g_1(x_1)g_2(x_2)\ldots g_M(x_M)$ defined on $\mathcal{X}_1 \times \cdots \times \mathcal{X}_M$. Given linear spaces $\mathbb{G}_1, \mathbb{G}_2, \ldots, \mathbb{G}_M$ of functions on $\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M$, respectively, their tensor product $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_M$ is the linear space spanned by $g_1 \otimes g_2 \otimes \cdots \otimes g_M$ as g_i runs over $\mathbb{G}_i$ for $i = 1, 2, \ldots, M$.

If $\mathbb{G}_i$ has dimension d_i and that $\{B_{i1}, \ldots, B_{id_i}\}$ is a set of basis functions in $\mathbb{G}_i$, $i = 1, 2, \ldots, M$, then

$$\{B_{1i_1} \otimes \cdots \otimes B_{Mi_M} : 1 \le i_j \le d_j, 1 \le j \le M\}$$

is a set of basis functions of $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_M$.

2.3.2 Tensor products of B-splines

Let $\mathbb{G}_j$ denote a d_j-dimensional linear space of splines on the real line, $j = 1, 2, \ldots, M$. Then $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_M$ is the linear space spanned by the tensor products $g_1 \otimes \cdots \otimes g_M$ of functions in $\mathbb{G}_j$, $j = 1, \ldots, M$,

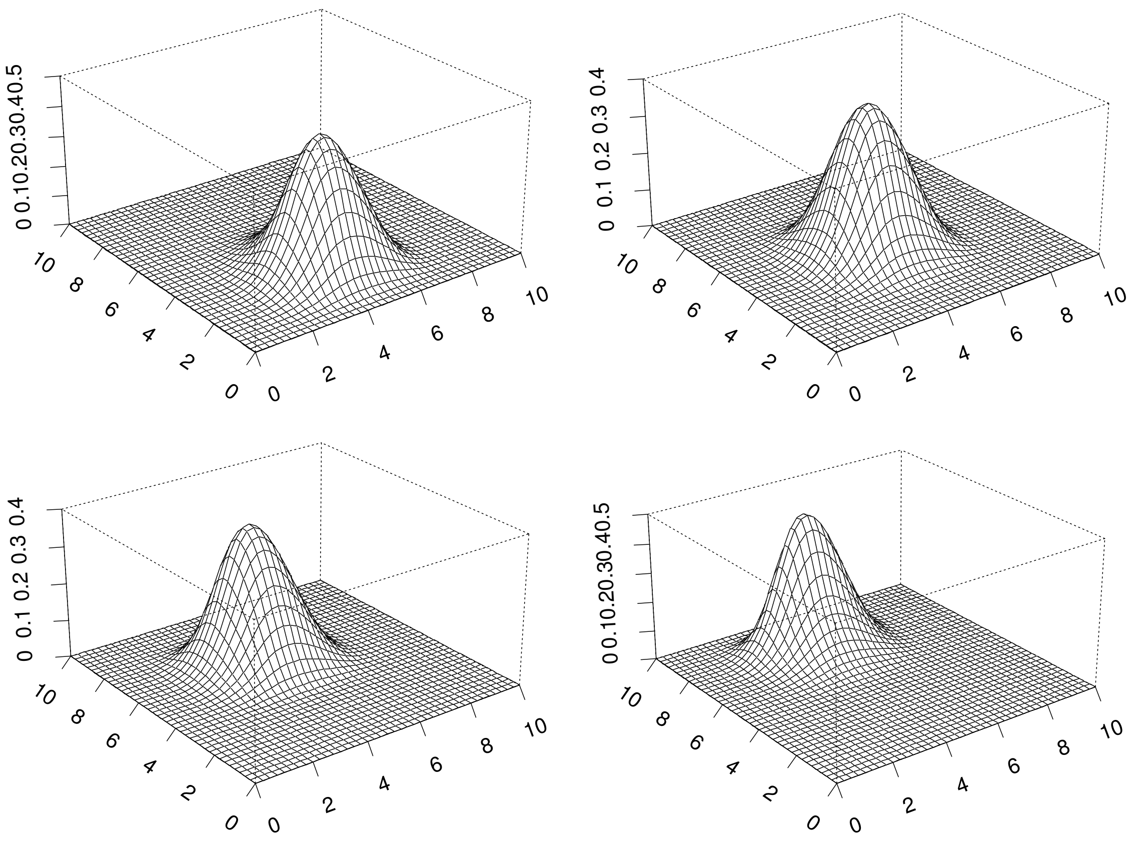

FIGURE 2.6. Tensor products of the cubic B-splines in Figure 2.5 with the cubic B-spline with knots $\{1, 3, 4, 7, 8\}$.

respectively. If $\{B_{i1}, \ldots, B_{id_i}\}$ is the set of B-splines in $\mathbb{G}_i$, $i = 1, 2, \ldots, M$, then

$$\{B_{1i_1} \otimes \cdots \otimes B_{Mi_M} : 1 \le i_j \le d_j, 1 \le j \le M\}$$

is a set of basis functions of $\mathbb{G}_1 \otimes \mathbb{G}_2 \otimes \cdots \otimes \mathbb{G}_M$. We refer to $B_{1i_1} \otimes \cdots \otimes B_{Mi_M}$, where $1 \le i_j \le d_j$ for $1 \le j \le M$, as tensor product B-splines.

2.3.3 Approximation properties

Waiting for Schumaker's book, it's been missing ... the lib has re-ordered.

2.4 Concavity

This section is devoted to the description of concavity, which plays an important role in optimization for maximum likelihood estimation. One of the most striking consequences of imposing concavity in a problem is that a local maximum is also a global maximum. Due to the geometric nature of concavity, we start with the concave function involving one variable and then proceed to higher dimensions.

2.4.1 One-dimensional case

Even though our applications will be in the context of high dimensional parameter spaces, there are several reasons to examine the one-dimensional case.

- Concavity is essentially a one-dimensional concept since it involves a straight line connecting two points.
- For theoretical and methodological purposes, it is necessary to study the one-dimensional trace $t \mapsto f(\boldsymbol{x} + t\mathbf{d})$, $t \in \mathbb{R}$, thoroughly.

Let f denote a real-valued function on $\mathbb{R}$. Then f is said to be *concave* if for every a, b in $\mathbb{R}$, the line segment connecting the points $(a, f(a))$ and $(b, f(b))$ is never above the graph of f. See Figure 2.7. In other words, f is concave if

$$f(ta + (1-t)b) \geq tf(a) + (1-t)f(b), \qquad 0 \leq t \leq 1, \quad a, b \in \mathbb{R}.$$

It is said to be strictly concave if

$$f(ta + (1-t)b) > tf(a) + (1-t)f(b), \quad 0 < t < 1, \quad a, b \in \mathbb{R} \text{ with } a \neq b.$$

Let $a < b < c$. From

$$b = \frac{c-b}{c-a}a + \frac{b-a}{c-a}c$$

and the concavity of f, we see that (Figure 2.7)

$$\frac{f(b)-f(a)}{b-a} \geq \frac{f(c)-f(a)}{c-a} \geq \frac{f(c)-f(b)}{c-b}, \qquad a < b < c. \tag{2.4.1}$$

For a fixed number x, it follows from the above inequality that the quotient function q defined on a neighborhood of 0 by

$$q(h) = (f(x+h) - f(x))/h, \qquad h \neq 0$$

is non-increasing. Thus the derivatives from the left and the right, f'_- and f'_+, exist and

$$\begin{aligned} f'_-(x) = \lim_{h \uparrow 0} q(h) &= \inf\{q(h) : h < 0\} \\ &\geq \sup\{q(h) : h > 0\} = \lim_{h \downarrow 0} q(h) = f'_+(x). \end{aligned}$$

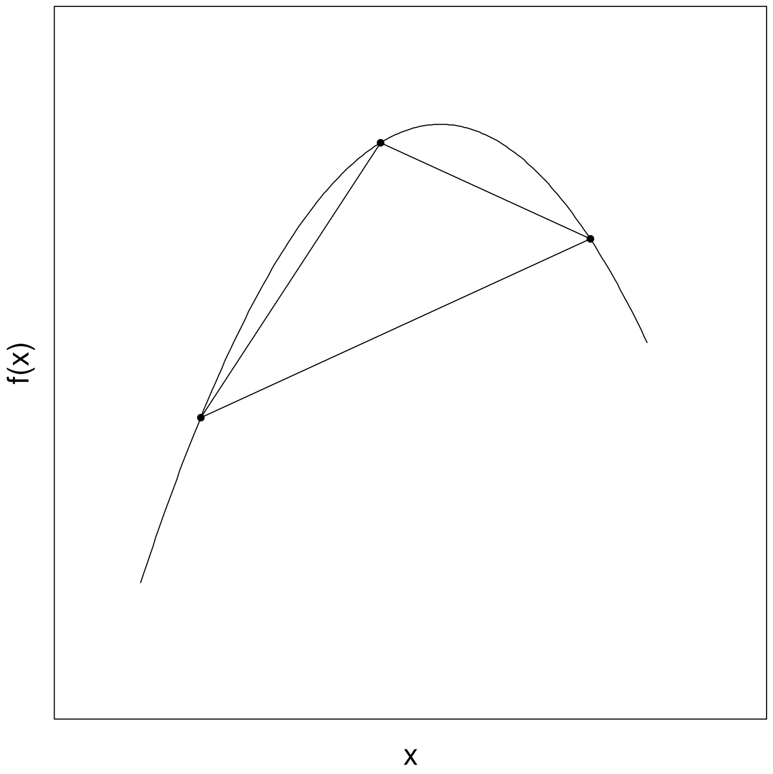

FIGURE 2.7. Concave function.

Moreover, for $a < b$, (2.4.1) shows that

$$f'_+(a) \geq \frac{f(b) - f(a)}{b - a} \geq \lim_{c \downarrow b} \frac{f(c) - f(b)}{c - b} = f'_+(b). \tag{2.4.2}$$

That is, f'_+ and f'_- are non-increasing on $\mathbb{R}$.

Let $a \leq x < y \leq b$ and set $M = \max\{|f'_-(a)|, |f'_+(b)|\}$. Then

$$-M \leq f'_+(b) \leq f'_+(y) \leq \frac{f(x) - f(y)}{x - y} \leq f'_-(x) \leq f'_-(a) \leq M.$$

Thus

$$|f(x) - f(y)| \leq M|x - y|.$$

It follows from this that a concave function also satisfies the Lipschitz condition.

In summary, a concave function is continuous and differentiable from the left and the right.

2.4.2 Multi-dimensional case

All the above results hold analogously in higher dimensional space. Let f denote a real-valued function on $\mathbb{R}^m$. Then f is said to be *concave* if for every $\boldsymbol{a}, \boldsymbol{b}$, the line segment connecting the points $(\boldsymbol{a}, f(\boldsymbol{a}))$ and $(\boldsymbol{b}, f(\boldsymbol{b}))$ is never above the graph of f. In other words, f is concave if

$$f(t\boldsymbol{a} + (1-t)\boldsymbol{b}) \geq tf(\boldsymbol{a}) + (1-t)f(\boldsymbol{b}), \qquad 0 \leq t \leq 1.$$

f is strictly concave if

$$f(t\boldsymbol{a} + (1-t)\boldsymbol{b}) > tf(\boldsymbol{a}) + (1-t)f(\boldsymbol{b}), \qquad 0 < t < 1 \text{ with } \boldsymbol{a} \neq \boldsymbol{b}.$$

Another method for recognizing concavity will be given in Section 2.4.3.

2.4.3 Checking concavity

For the applications to be described in this monograph, we will be considering statistical modeling problems involving twice differentiable likelihood functions. Thus it is natural to consider twice differentiable concave functions.

According to (2.4.2), the derivative of a concave function is non-increasing. It turns out that the converse is also true, which provides an easy way to detect concavity. In fact, if f is differentiable and f' is non-increasing, then f is concave on $\mathbb{R}$. For if f is not concave, there exists $a < x < b$ such that

$$f(x) < \frac{b-x}{b-a}f(a) + \frac{x-a}{b-a}f(b).$$

This is equivalent to

$$\frac{f(x)-f(a)}{x-a} < \frac{f(b)-f(x)}{b-x}.$$

Now apply the Mean Value Theorem to each of the intervals $[a, x]$ and $[x, b]$ to see that there exist $a < \xi < x < \eta < b$ such that $f'(\xi) < f'(\eta)$. Thus f' fails to be non-increasing on $\mathbb{R}$. This result provides a simple way to check concavity. That is, a function f is concave on $\mathbb{R}$ if it has nonpositive second derivative on $\mathbb{R}$.

The multi-dimensional case can be handled similarly. Recall that the gradient of a function f is the m-dimensional vector:

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_1}, \; \frac{\partial f}{\partial x_2}, \; \cdots, \; \frac{\partial f}{\partial x_m} \right)^T, \qquad \boldsymbol{x} = (x_1, \ldots, x_m)^T \in \mathbb{R}^m.$$

The Hessian of f at $\boldsymbol{x}$ is the $m \times m$ matrix:

$$\mathbf{H}(\boldsymbol{x}) = \begin{pmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_m} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_m} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial^2 f}{\partial x_m \partial x_1} & \dfrac{\partial^2 f}{\partial x_m \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_m \partial x_m} \end{pmatrix}.$$

If f is concave, then $\mathbf{H}$ is non-positive definite; that is,

$$\boldsymbol{x}^T \mathbf{H}(\boldsymbol{x}) \boldsymbol{x} \leq 0, \qquad \boldsymbol{x} \in \mathbb{R}^m.$$

Conversely, if $\mathbf{H}$ is negative definite in the sense that

$$\boldsymbol{x}^T \mathbf{H}(\boldsymbol{x}) \boldsymbol{x} < 0, \qquad \boldsymbol{x} \in \mathbb{R}^m \text{ with } \boldsymbol{x} \neq \mathbf{0},$$

then f is strictly concave. These results follow because if a function is concave on $\mathbb{R}^m$, then its restriction to any line is also concave. The above condition for checking concavity turns out to be very convenient in verifying the concavity of the log-likelihood functions for various applications to be discussed later in this monograph.

2.4.4 Existence and the uniqueness of the maximum

Concavity is a remarkable property for working with a class of problems in which the maximum of a function is used for the determination of certain properties, either in estimation or in model formulation. It will be seen below that a simple condition will ensure the existence of the maximum of a concave function. But it is not necessarily true that the maximum of a concave function exists. The logarithm function is such an example. See Figure 2.8.

A necessary and sufficient condition for the existence of the maximum of a concave function is that its gradient vector vanishes at some point in the domain of that function. That is, $\nabla f(\boldsymbol{x}) = \mathbf{0}$ for some $\boldsymbol{x}$. This can be seen as follows. According to the definition of the directional derivative and concavity:

$$\begin{aligned} \nabla f(\boldsymbol{x})^T(\boldsymbol{y} - \boldsymbol{x}) &= \lim_{t \downarrow 0} \frac{f(\boldsymbol{x} + t(\boldsymbol{y} - \boldsymbol{x})) - f(\boldsymbol{x})}{t} \\ &\geq \lim_{t \downarrow 0} \frac{t f(\boldsymbol{y}) + (1-t) f(\boldsymbol{x}) - f(\boldsymbol{x})}{t} \\ &= f(\boldsymbol{y}) - f(\boldsymbol{x}). \end{aligned}$$

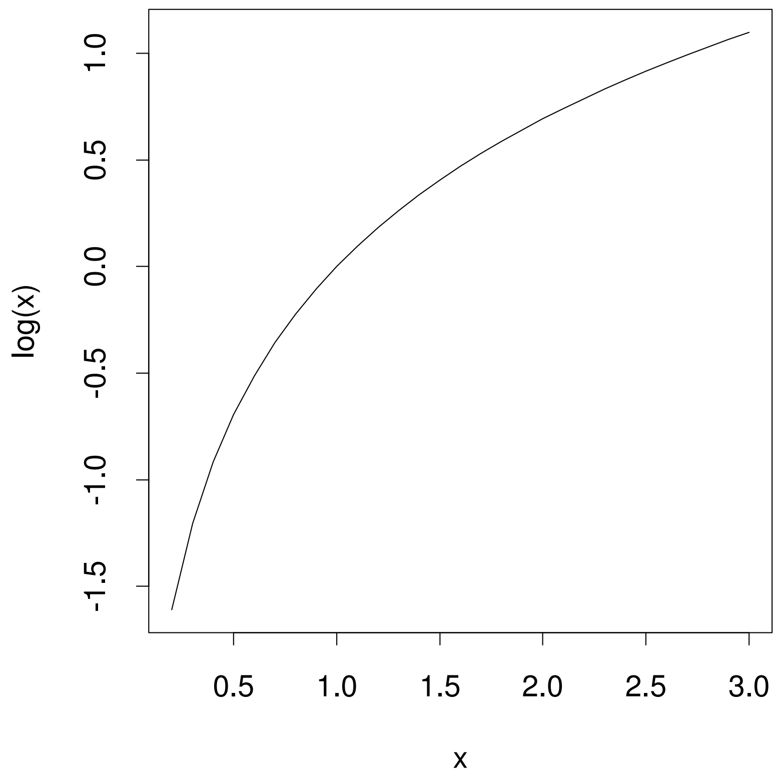

FIGURE 2.8. Concave function.

Or,

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T(\boldsymbol{y}-\boldsymbol{x}), \qquad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^m.$$

This implies that if f is concave and $\nabla f(\boldsymbol{x}) = \boldsymbol{0}$, then $\boldsymbol{x}$ is a maximum point of f.

A strictly concave function f has at most one maximum point; that is, there is at most one point $\boldsymbol{x}_0 \in \mathbb{R}^m$ such that $f(\boldsymbol{x}_0) = \max_{\boldsymbol{x}\in\mathbb{R}^m} f(\boldsymbol{x})$. Suppose f has a maximum point $\boldsymbol{x}_0$. Then there is a positive number ϵ such that

$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0) - \epsilon, \qquad \boldsymbol{x} \in \mathbb{R}^m \text{ with } |\boldsymbol{x}-\boldsymbol{x}_0| = 1;$$

here

$$|\boldsymbol{x}| = \sqrt{x_1^2 + \cdots + x_m^2}, \qquad \boldsymbol{x} = (x_1, \ldots, x_m)^T.$$

It now follows from the concavity of f that

$$f(\boldsymbol{x}) \leq f(\boldsymbol{x}_0) - \epsilon|\boldsymbol{x}-\boldsymbol{x}_0|, \qquad \boldsymbol{x} \in \mathbb{R}^m \text{ with } |\boldsymbol{x}-\boldsymbol{x}_0| \geq 1,$$

and hence that

$$\lim_{|\boldsymbol{x}|\to\infty} f(\boldsymbol{x}) = -\infty. \tag{2.4.3}$$

Conversely, if f is concave and satisfies (2.4.3), then it has a maximum point.

Let f now be a concave function such that

$$\lim_{s\to\infty} f(s\boldsymbol{x}) = -\infty, \qquad \boldsymbol{x} \in \mathbb{R}^m \text{ with } \boldsymbol{x} \neq \mathbf{0}. \tag{2.4.4}$$

Then f has a maximum point. To verify this result, set $B = \{\, \boldsymbol{x} \in \mathbb{R}^m : |\boldsymbol{x}| = 1 \,\}$ and $B_n = \{\, \boldsymbol{x} \in B : f(n\boldsymbol{x}) < f(\mathbf{0}) - 1 \,\}$. It follows from the concavity of f that $B_1 \subset B_2 \subset \ldots$. Since f is continuous, B_n is an open subset of B for $n \geq 1$. It follows from (2.4.4) that $\cup_{n\geq 1} B_n = B$. Since B is compact, there is a positive integer n such that $B_n = B_1 \cup \ldots \cup B_n = B$ and hence $f(n\boldsymbol{x}) < f(\mathbf{0}) - 1$ for $\boldsymbol{x} \in B$. Arguing as in the previous paragraph, we conclude that (2.4.3) holds and hence that f has a maximum point.

We will see that (2.4.4) is easily satisfied in a wide variety of applications described in this monograph.

2.5 Optimization

In previous sections we discussed properties of a concave function and some of their useful consequences. The most important feature is the uniqueness and the existence of the maximum of the concave likelihood function. Our theoretical and methodological developments rely on this to address the identifiability issue in estimation. In practical applications, however, the existence is not enough as the likelihood function may be so flat that locating the optimal solution can be immensely difficult, especially in high dimensional situations. This section is devoted to address the problem of finding the maximum of a function. Here is a brief overview. The Newton–Raphson procedure with good starting values is quite efficient in most of our applications. Occasionally step halving in the search direction is needed to improve its efficiency. One exception is the polychotomous regression to be discussed in Chapter 5.

The function that we want to maximize is called the *objective function*. In our applications to be discussed later, this will be described in the form of a log-likelihood function. Maximum likelihood estimates are obtained by maximizing this function. This problem can be described formally as follows. Let f denote a concave function on $\mathbb{R}^m$ and suppose that the maximum of the function exists. We want to locate $\widehat{\boldsymbol{\beta}} \in \mathbb{R}^m$ so that

$$f(\widehat{\boldsymbol{\beta}}) = \max\{f(\boldsymbol{\beta}) : \boldsymbol{\beta} \in \mathbb{R}^m\}.$$

The literature for optimization is huge and here we give a brief account of the subject to help our readers understand the methodology discussed in later chapters. A point of entry to numerical optimization in statistics is Kennedy and Gentle (1980). Recent accounts can be found in Polak (1997), Nocedal and Wright (1999).

2.5.1 Preview of the methods

In general, the optimal solution can be only approximated by iterative method. Since our applications will be based on a smooth objective function (that is, the log-likelihood function), we will focus on methods that utilize the function itself, the first and second derivatives. Within this class, there are several algorithms that are distinguished by their strategy in moving one iteration to the next. Some procedures accumulate information collected at previous steps, while others use only local information from the current iteration. Neverthless all reliable algorithms should possess the following properties:

- *Robustness.* They should perform well on a wide variety of situations, under reasonable choices of the starting values.
- *Efficiency.* The required computing time should be as little as possible, without sacrificing a lot of storage space.
- *Accuracy.* They should be able to produce a solution with high precision, without being overly sensitive to noise in the data or to the computer rounding errors.

An approach that fits these descriptions well is the line search method. Typically, we start with an initial guess $\boldsymbol{\beta}_0$ and proceed to search for a direction $\mathbf{d}_0$ that yields a reasonable ascent or climb, and then determine how far we should go along this direction. This distance, λ_0, will be selected to maximize or close to maximize the function at the new location given by $\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 + \lambda_0 \mathbf{d}_0$. Upon arriving at $\boldsymbol{\beta}_1$, a new direction $\mathbf{d}_1$ and step size λ_1 will be chosen to optimize the ascent. The new location is then computed according to $\boldsymbol{\beta}_2 = \boldsymbol{\beta}_1 + \lambda_1 \mathbf{d}_1$. The iteration process, which is summarized by

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j + \lambda_j \mathbf{d}_j, \qquad j \geq 0,$$

repeats until no more ascent can be achieved. (This is governed by a stopping rule to be discussed later.) Here $\mathbf{d}_j$ is the *step direction* at the jth step, the *step size* λ_j is the distance along this direction before stopping. Computation at the jth step consists of the determination of these two quantities. That is, any optimization involving dimensionality greater than two will consist of a one-dimensional optimization problem and the search for good directions in the high dimensional space. A good procedure will ensure that the final location is in the vicinity of the optimal solution $\widehat{\boldsymbol{\beta}}$.

How are we going to search for the step direction? We address this issue by finding a simple condition that will ensure that the function increases. To motivate this, we say that a direction $\mathbf{d}_j$ is *acceptable* if the function climbs up along that direction. That is, $f(\boldsymbol{\beta}_j + \lambda_j \mathbf{d}_j) > f(\boldsymbol{\beta}_j)$ for some $\lambda_j > 0$. According to a Taylor's expansion (up to the linear term),

$$f(\boldsymbol{\beta}_j + \lambda_j \mathbf{d}_j) = f(\boldsymbol{\beta}_j) + \nabla f(\boldsymbol{\beta}_j + \bar{\lambda}_j \mathbf{d}_j)^T \lambda_j \mathbf{d}_j, \qquad \text{for some } 0 < \bar{\lambda}_j < \lambda_j;$$

the acceptability condition amounts to requiring that $\nabla f(\boldsymbol{\beta}_j + \bar{\lambda}_j \mathbf{d}_j)^T \mathbf{d}_j > 0$. If f is continuously differentiable and λ_j is sufficiently small, then $\nabla f(\boldsymbol{\beta}_j + \bar{\lambda}_j \mathbf{d}_j)^T \mathbf{d}_j \approx \nabla f(\boldsymbol{\beta}_j)^T \mathbf{d}_j$. Thus $\mathbf{d}_j$ is acceptable if

$$\nabla f(\boldsymbol{\beta}_j)^T \mathbf{d}_j > 0. \tag{2.5.1}$$

Incidentally, $\nabla f(\boldsymbol{\beta})^T \mathbf{d}$ is the directional derivative of f at $\boldsymbol{\beta}$ with respect to the direction $\mathbf{d}$:

$$\frac{d}{d\lambda} f(\boldsymbol{\beta} + \lambda \mathbf{d}) \bigg|_{\lambda=0} = \nabla f(\boldsymbol{\beta})^T \mathbf{d}.$$

In other words, any direction that yields a positive directional derivative is acceptable.

The above analysis provides a simple way to determine a search direction at a given iteration. That is, to find a direction that satisfies (2.5.1), it is sufficient to search it in the form given by

$$\mathbf{d} = \mathbf{B} \nabla f(\boldsymbol{\beta}), \qquad \text{where } \mathbf{B} \text{ is a positive definite matrix.} \tag{2.5.2}$$

It turns out that such a condition is not only sufficient, it is also necessary. In fact, if (2.5.1) holds, then $\mathbf{B} = \boldsymbol{I} - \nabla f(\boldsymbol{\beta}) \nabla f(\boldsymbol{\beta})^T / \nabla f(\boldsymbol{\beta})^T \nabla f(\boldsymbol{\beta}) + \mathbf{d}\mathbf{d}^T / \mathbf{d}^T \nabla f(\boldsymbol{\beta})$ is positive definite and satisfies $\mathbf{d} = \mathbf{B} \nabla f(\boldsymbol{\beta})$.

The choice of the matrix $\mathbf{B}$ in practice has led to the following popular methods.

- The gradient or the steepest ascent method is obtained by setting $\mathbf{B} = \boldsymbol{I}$.
- The Newton–Raphson (NR) method is obtained by using $\mathbf{B} = -\mathbf{H}^{-1}$, where $\mathbf{H}$ is the Hessian matrix of f.
- Quasi-Newton method is obtained when $\mathbf{B}$ is generated by the adding a rank one or rank two symmetric matrix. This popular method is also known as the Davidon–Fletcher–Powell (DFP) when the update is carried out by using a rank-one matrix, while it is called the Fletcher-Powell-Gofarb-Shanno (FPGS) method when a rank-two matrix is used.
- A related popular method that does not belong to this group is called the *conjugate gradient method*, which was originally developed to solve a large system of linear equations.

Here is a quick comparison of these methods: The steepest ascent is slowest method among the competitors. NR is effective when there is good initial guess and the computation of the Hessian matrix is easy. Quasi-Newton uses the secant method to approximate the Hessian used in the

NR method, and it is useful when the latter is difficult to compute. The conjugate gradient method is faster than steepest ascent, though it is not as fast as the Newton-Raphson method. It can be useful when the dimensionality of the problem is large or the storage of the Hessian matrix becomes an issue.

We should remark that the provision of a decent initial value is problem specific. It is very important that such initial value be as accurate as possible. A bad initial value may not lead to the optimal solution.

Before describing these methods in greater detail, it is helpful to review what efficiency means.

Efficiency.

It will be helpful to discuss the efficiency of the above methods in terms of rates of convergence. A sequence $\boldsymbol{\beta}_j$ is said to converge to the optimal solution $\widehat{\boldsymbol{\beta}}$ if $\lim_j \|\boldsymbol{\beta}_j - \widehat{\boldsymbol{\beta}}\| = 0$. It is said to converge super-linearly if

$$\lim_{j\to\infty} \frac{\|\boldsymbol{\beta}_{j+1} - \widehat{\boldsymbol{\beta}}\|}{\|\boldsymbol{\beta}_j - \widehat{\boldsymbol{\beta}}\|} = 0.$$

It is said to converge quadratically iff

$$\lim_{j\to\infty} \frac{\|\boldsymbol{\beta}_{j+1} - \widehat{\boldsymbol{\beta}}\|}{\|\boldsymbol{\beta}_j - \widehat{\boldsymbol{\beta}}\|^2} = c \in \mathbb{R}.$$

Quadratically convergence is a nice property in the sense that the number of significant digits is doubled at each iteration. For example, suppose the error in the current iteration is 10^{-4}, a quadratic convergence method would produce an error of 10^{-8} in the next iteration. Quadratic convergence is currently the benchmark for optimization algorithms, but achieving it is not always feasible. Consequently, most accounts for large scale problems would view super-linear convergence as acceptable.

2.5.2 Gradient Methods – steepest ascent

It was observed in the last section that the function f climbs if $\nabla f(\boldsymbol{\beta}_j + \bar{\lambda}_j \mathbf{d}_j)^T \mathbf{d}_j \approx \nabla f(\boldsymbol{\beta}_j)^T \mathbf{d}_j > 0$. By the Cauchy-Schwarz inequality, the amount of ascent (ignoring the step size) is given by

$$|\nabla f(\boldsymbol{\beta}_j)^T \mathbf{d}_j| \leq \|\nabla f(\boldsymbol{\beta}_j)\| \cdot \|\mathbf{d}_j\|$$

with equality if and only if $\mathbf{d}_j$ is a multiple of $\nabla f(\boldsymbol{\beta}_j)$. This implies that the *steepest ascent* direction is given by $\mathbf{d}_j = \nabla f(\boldsymbol{\beta}_j)$, which is of the form (2.5.2) with $\mathbf{B} = \boldsymbol{I}$, the identity matrix. This search direction method is referred to as steepest ascent in the literature.

The method is normally coupled for the search of optimal step size. Specifically, suppose we start with the initial position $\boldsymbol{\beta}_0$ and the direction $\boldsymbol{g}_0 = \nabla f(\boldsymbol{\beta}_0)$, and iterate according to

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j + \lambda_j \boldsymbol{g}_j, \qquad \text{where } \boldsymbol{g}_j = \nabla f(\boldsymbol{\beta}_j), \quad j \geq 0,$$

with λ_j satisfying

$$\phi'(\lambda_j) = \frac{d}{d\lambda} f(\boldsymbol{\beta}_j + \lambda \boldsymbol{g}_j)\Big|_{\lambda=\lambda_j} = \nabla f(\boldsymbol{\beta}_j + \lambda_j \boldsymbol{g}_j)^T \boldsymbol{g}_j = 0.$$

That is, the step size is maximized during each iteration. In this case, steepest ascent method is said to be exact. Note that $\boldsymbol{g}_{j+1} := \nabla f(\boldsymbol{\beta}_{j+1}) = \nabla f(\boldsymbol{\beta}_j + \lambda_j \boldsymbol{g}_j)$ is perpendicular to $\boldsymbol{g}_j$.

Unfortunately, steepest ascent method is sensitive to small deviations in direction and step size, so these quantities must be computed with relatively high accuracy. See Kennedy and Gentle (1980, p.440). Moreover, the method is very inefficient, requiring a large number of steps which produces a zigzag pattern as indicated in trying to find the maximum of the function $f(\beta_1, \beta_2) = -\beta_1^2 - c\beta_2^2$, $c > 0$. It can be easily seen that if the current iteration is of the form $\boldsymbol{\beta}_j = a(c, \pm 1)^T$, then the next one is given by

$$\boldsymbol{\beta}_{j+1} = a\frac{c-1}{c+1}(c, \mp 1)^T,$$

that is, the error is reduced by the factor $(c-1)/(c+1)$. For example, if $c = 100$ and the starting value is $\boldsymbol{\beta}_0 = (1, 0.01)^T$, then after 100 iterations,

$$\boldsymbol{\beta}_{100} = \left(\frac{c-1}{c+1}\right)^{100} (1, 0.01)^T = (0.135..., 0.00135...).$$

The steepest ascent is usually used to supply initial values since it tends to make good progress initially and then slows down as the iteration approach a solution.

2.5.3 Newton–Raphson Method

This is probably the most popular method for numerical optimization. It requires the computation of the gradient vector and the Hessian matrix, which may be important for likelihood based estimation, where the standard errors of the estimates are computed through the Hessian matrix. While steepest ascent is using the Taylor's formula only up to the linear term, the Newton–Raphson method is based on the quadratic approximation given by

$$f(\boldsymbol{\beta} + \mathbf{d}) \approx f(\boldsymbol{\beta}) + \nabla f(\boldsymbol{\beta})^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{H}(\boldsymbol{\beta})\mathbf{d},$$

where $\mathbf{H}$ is negative definite. Maximize the above function to yield

$$\mathbf{0} = \nabla f(\boldsymbol{\beta}) + \mathbf{H}(\boldsymbol{\beta})\mathbf{d}.$$

The optimal direction, called the Newton direction, is given by

$$\mathbf{d} = -[\mathbf{H}(\boldsymbol{\beta})]^{-1}\nabla f(\boldsymbol{\beta}),$$

which is (2.5.2) with $\mathbf{B} = -\mathbf{H}^{-1}$ and $\lambda = 1$. Thus the iteration process can be described by

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j - \mathbf{H}(\boldsymbol{\beta}_j)^{-1}\nabla f(\boldsymbol{\beta}_j), \qquad j \geq 0.$$

Under appropriate conditions, if the sequence $\{\boldsymbol{\beta}_j\}$ converges, it converges to $\widehat{\boldsymbol{\beta}}$. This is usually summarized as follows. Suppose f is continuously differentiable in an open convex set of $\mathbb{R}^m$, its Hessian is Lipschitz and the inverse is also bounded. Suppsose ∇f has a zero over that set. Then the sequence is well defined, converges to the zero of ∇f quadratically. In the following discussion, it will be helpful to examine the practical implications of this important result.

1. It is known that this method is very sensitive to the initial values. In order to ensure convergence, it may be helpful to modify the Newton–Raphson method using the step-halving procedure by shrinking back the step-search appropriately. See Section 2.5.7 for more detail.

2. The computation of the Hessian matrix is required. This is one of the weaknesses of the method. Namely, in some problem such a computation can be tedious and is error-prone. For large scale problems, the computational time and the storage of the Hessian can be serious obstacles.

3. It requires m^2 function evaluations of the Hessian matrix and m evaluations of the gradient vector.

4. The next direction requires $O(m^3)$ operations in solving the Newton equation

5. This method is very efficient if one can overcome the above issues. This efficiency is described in terms of quadratic convergence: the iterates converge to the true maximum quadratically. In other words, the number of significant digits in $\boldsymbol{\beta}_k$ as an approximation to $\widehat{\boldsymbol{\beta}}$ doubles at each iteration once it is near the optimal value. We illustrate this using the algorithm for the one-dimensional case:

   $$\lambda_{j+1} = \lambda_j - \phi'(\lambda_j)/\phi''(\lambda_j).$$

   Set $f = \phi' \in C^2$ so that $f(\lambda) = 0$. Then there is a neighborhood of λ and a constant c such that if Newton's method is started in that neighborhood, the successive points become steadily closer to λ and satisfy

$$|\lambda_{j+1} - \lambda| \le c|\lambda_j - \lambda|^2.$$

To see this, set $e_j = \lambda_j - \lambda$. Then

$$e_{j+1} = \lambda_{j+1} - \lambda = \lambda_j - f(\lambda_j)/f'(\lambda_j) - \lambda = e_j - f(\lambda_j)/f'(\lambda_j).$$

By Taylor's Theorem,

$$0 = f(\lambda) = f(\lambda_j - e_j) = f(\lambda_j) - e_j f'(\lambda_j) + \frac{1}{2}e_j^2 f''(\xi_j),$$

where ξ_j is between λ_j and λ. Rearranging

$$e_j - f(\lambda_j)/f'(\lambda_j) = \frac{1}{2}\frac{f''(\xi_j)}{f'(\lambda_j)}e_j^2.$$

Thus

$$e_{j+1} = \frac{1}{2}\frac{f''(\xi_j)}{f'(\lambda_j)}e_j^2 = \text{const.} \times e_j^2,$$

which is a quadratic convergence.

Most of our applications have the Hessian matrix computed explicitly. With moderate m and appropriately chosen starting values, the Newton–Raphson method has been very effective. However, the efficiency decreases with increasing dimensionality m due to the computation of the of the Hessian matrix and the cost for solving the system of linear equations. See Chapter 5 for details. In this situation, it may be helpful to consider the modified Newton methods, which will be discussed next.

2.5.4 Quasi-Newton Method

In an attempt to perform a large optimization computation, a physicist W. C. Davidon in the mid 1950s developed an idea to approximate the error-prone Hessian matrix used in the Newton–Raphson method. His idea can be described as follows. At the ith iteration, the objective function is approximated by the quadratic model:

$$m_i(\mathbf{d}) = f_i + \boldsymbol{g}_i^T\mathbf{d} + \frac{1}{2}\mathbf{d}^T\mathbf{B}_i\mathbf{d},$$

where f_i and $\boldsymbol{g}_i$ are the function and the gradient values at $\boldsymbol{\beta}_i$, and $\mathbf{B}_i$ is an $m \times m$ symmetric negative definite matrix that will be updated during the iteration process. Maximize the function m_i to obtain the updated search direction:

$$\mathbf{d}_i = -\mathbf{B}_i^{-1}\boldsymbol{g}_i.$$

The new iterate is

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \lambda_i \mathbf{d}_i,$$

where the step size λ_i is along the direction $\mathbf{d}_i$ given by

$$\phi'(\lambda_i) = \frac{d}{d\lambda} f(\boldsymbol{\beta}_i + \lambda \mathbf{d}_i) = \nabla f(\boldsymbol{\beta}_i + \lambda \mathbf{d}_i)^T \mathbf{d}_i = 0.$$

To obtain the update $\mathbf{B}_{i+1}$ from $\mathbf{B}_i$, construct a quadratic model of the form

$$m_{i+1}(\mathbf{d}) = f_{i+1} + \boldsymbol{g}_{i+1}^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \mathbf{B}_{i+1} \mathbf{d}$$

so that its gradient matches the gradient of f at $\boldsymbol{\beta}_i$ and $\boldsymbol{\beta}_{i+1}$. The second condition is clearly satisfied since $\nabla m_{i+1}(\mathbf{0}) = \boldsymbol{g}_{i+1}$. The first condition is satisfied if

$$\nabla m_{i+1}(-\lambda_i \mathbf{d}_i) = \boldsymbol{g}_{i+1} - \lambda_i \mathbf{B}_{i+1} \mathbf{d}_i = \boldsymbol{g}_i.$$

That is,

$$\lambda_i \mathbf{B}_{i+1} \mathbf{d}_i = \mathbf{B}_{i+1}(\boldsymbol{x}_{i+1} - \boldsymbol{x}_i) = \boldsymbol{g}_{i+1} - \boldsymbol{g}_i.$$

Set

$$\boldsymbol{r}_i = \boldsymbol{\beta}_{i+1} - \boldsymbol{\beta}_i, \qquad \boldsymbol{y}_i = \boldsymbol{g}_{i+1} - \boldsymbol{g}_i.$$

Then we obtain the so-called *secant equation*:

$$\mathbf{B}_{i+1} \boldsymbol{r}_i = \boldsymbol{y}_i. \tag{2.5.3}$$

This relationship indicates that the quasi-Newton method actually employs a discrete approximation of the Hessian matrix using the secant method.

We now derive the popular Davidon–Fletcher–Powell (DFP) and Broyden–Fletcher–Goldfarb–Shanno (BFGS) updated formulas. We start with the symmetric rank-one update method, follow with the rank-two method, which then yields the desired updated formulas for the Hessian and its inverse.

Symmetric rank-one update:

Start with a symmetric matrix $\mathbf{B}_0$ (say, the identity matrix), then update it according to

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \boldsymbol{z}_i \boldsymbol{z}_i^T, \qquad i \geq 1,$$

where $\boldsymbol{z}_i$ is selected to satisfy (2.5.3). Now multiply the above by $\boldsymbol{r}_i$ and use (2.5.3) to get

$$\boldsymbol{z}_i(\boldsymbol{z}_i^T \boldsymbol{r}_i) = \boldsymbol{y}_i - \mathbf{B}_i \boldsymbol{r}_i,$$

so that

$$(\boldsymbol{z}_i^T \boldsymbol{r}_i)^2 = (\boldsymbol{y}_i - \mathbf{B}_i \boldsymbol{r}_i)^T \boldsymbol{r}_i.$$

Consequently,

$$\mathbf{B}_{i+1} = \mathbf{B}_i + \frac{(\boldsymbol{y}_i - \mathbf{B}_i \boldsymbol{r}_i)(\boldsymbol{y}_i - \mathbf{B}_i \boldsymbol{r}_i)^T}{(\boldsymbol{y}_i - \mathbf{B}_i \boldsymbol{r}_i)^T \boldsymbol{r}_i}.$$

Although one can obtain the inverse of the Hessian matrix using the Sherman–Morrison–Woodbury formula, we prefer to derive it using the above idea since it will provide insight for the methods to be discussed later. Start with a symmetric $\mathbf{G}_0$, then update it according to

$$\mathbf{G}_{i+1} = \mathbf{G}_i + \boldsymbol{z}_i \boldsymbol{z}_i^T, \qquad j \geq 1,$$

where $\boldsymbol{z}_i$ is chosen to satisfy

$$\mathbf{G}_{i+1} \boldsymbol{y}_i = \boldsymbol{r}_i.$$

By an argument similar to the one given above, we obtain the updated formula for the inverse of the Hessian matrix:

$$\mathbf{G}_{i+1} = \mathbf{G}_i + \frac{(\boldsymbol{r}_i - \mathbf{G}_i \boldsymbol{y}_i)(\boldsymbol{r}_i - \mathbf{G}_i \boldsymbol{y}_i)^T}{(\boldsymbol{r}_i - \mathbf{G}_i \boldsymbol{y}_i)^T \boldsymbol{y}_i}.$$

Symmetric rank-two update:

This method is developed by expanding the updated formula for $\mathbf{G}_{i+1}$:

$$\mathbf{G}_{i+1} = \mathbf{G}_i + a_1 \boldsymbol{r}_i \boldsymbol{r}_i^T + a_2 (\mathbf{G}_i \boldsymbol{y}_i)(\mathbf{G}_i \boldsymbol{y}_i)^T + a_3 [\boldsymbol{r}_i (\mathbf{G}_i \boldsymbol{y}_i)^T + (\mathbf{G}_i \boldsymbol{y}_i) \boldsymbol{r}_i^T].$$

Simplify by dropping the asymmetric terms, yielding

$$\mathbf{G}_{i+1} = \mathbf{G}_i + a_1 \boldsymbol{r}_i \boldsymbol{r}_i^T + a_2 (\mathbf{G}_i \boldsymbol{y}_i)(\mathbf{G}_i \boldsymbol{y}_i)^T.$$

Now determine the coefficients a_1 and a_2 to ensure that $\mathbf{G}_{i+1} \boldsymbol{y}_i = \boldsymbol{r}_i$. From

$$\boldsymbol{r}_i = \mathbf{G}_i \boldsymbol{y}_i + a_1 (\boldsymbol{r}_i^T \boldsymbol{y}_i) \boldsymbol{r}_i + a_2 [(\mathbf{G}_i \boldsymbol{y}_i)^T \boldsymbol{y}_i](\mathbf{G}_i \boldsymbol{y}_i)$$

we see that setting $a_1 = 1/\boldsymbol{r}_i^T \boldsymbol{y}_i$ and $a_2 = -1/(\mathbf{G}_i \boldsymbol{y}_i)^T \boldsymbol{y}_i$ would meet the requirement. Thereby we obtain the well-known Davidon–Fletcher–Powell (DFP) updated formula for the inverse of the Hessian matrix:

$$\mathbf{G}_{i+1} = \mathbf{G}_i + \frac{\boldsymbol{r}_i \boldsymbol{r}_i^T}{\boldsymbol{y}_i^T \boldsymbol{r}_i} - \frac{(\mathbf{G}_i \boldsymbol{y}_i)(\mathbf{G}_i \boldsymbol{y}_i)^T}{\boldsymbol{y}_i^T \mathbf{G}_i \boldsymbol{y}_i}.$$

Using the same reasoning, the (BFGS) updated formula for the Hessian matrix is given by

$$\mathbf{H}_{i+1} = \mathbf{H}_i + \frac{\boldsymbol{y}_i \boldsymbol{y}_i^T}{\boldsymbol{y}_i^T \boldsymbol{r}_i} - \frac{(\mathbf{H}_i \boldsymbol{r}_i)(\mathbf{H}_i \boldsymbol{r}_i)^T}{\boldsymbol{r}_i^T \mathbf{H}_i \boldsymbol{r}_i}.$$

We summarize these in the following algorithms.

DFP Algorithm:

1. Starting values: $\boldsymbol{\beta}_0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{\beta}_0)$, $\mathbf{G}_0$, $i = 0$.
2. If $\boldsymbol{g}_i = \mathbf{0}$, stop. Else, compute
$$\lambda_i = \operatorname*{argmin}_{\lambda \geq 0} f(\boldsymbol{\beta}_i - \lambda \mathbf{G}_i \boldsymbol{g}_i).$$
3. Compute
$$\begin{aligned}
\boldsymbol{\beta}_{i+1} &= \boldsymbol{\beta}_i - \lambda_i \mathbf{G}_i \boldsymbol{g}_i,\\
\boldsymbol{g}_{i+1} &= \nabla f(\boldsymbol{\beta}_{i+1}),\\
\boldsymbol{r}_i &= \boldsymbol{\beta}_{i+1} - \boldsymbol{\beta}_i,\\
\boldsymbol{y}_i &= \boldsymbol{g}_{i+1} - \boldsymbol{g}_i,\\
\mathbf{G}_{i+1} &= \mathbf{G}_i + \frac{\boldsymbol{r}_i \boldsymbol{r}_i^T}{\boldsymbol{y}_i^T \boldsymbol{r}_i} - \frac{(\mathbf{G}_i \boldsymbol{y}_i)(\mathbf{G}_i \boldsymbol{y}_i)^T}{\boldsymbol{y}_i^T \mathbf{G}_i \boldsymbol{y}_i}.
\end{aligned}$$
4. Set $i = i + 1$, goto step 2.

BFGS Algorithm:

1. Starting values: $\boldsymbol{\beta}_0$, $\boldsymbol{g}_0 = \nabla f(\boldsymbol{\beta}_0)$, $\mathbf{H}_0$, $i = 0$.
2. If $\boldsymbol{g}_i = \mathbf{0}$, stop. Else, compute
$$\lambda_i = \operatorname*{argmin}_{\lambda \geq 0} f(\boldsymbol{\beta}_i - \lambda \mathbf{H}_i^{-1} \boldsymbol{g}_i).$$
3. Compute
$$\begin{aligned}
\boldsymbol{\beta}_{i+1} &= \boldsymbol{\beta}_i - \lambda_i \mathbf{H}_i^{-1} \boldsymbol{g}_i,\\
\boldsymbol{g}_{i+1} &= \nabla f(\boldsymbol{\beta}_{i+1}),\\
\boldsymbol{r}_i &= \boldsymbol{\beta}_{i+1} - \boldsymbol{\beta}_i,\\
\boldsymbol{y}_i &= \boldsymbol{g}_{i+1} - \boldsymbol{g}_i,\\
\mathbf{H}_{i+1} &= \mathbf{H}_i + \frac{\boldsymbol{y}_i \boldsymbol{y}_i^T}{\boldsymbol{y}_i^T \boldsymbol{r}_i} - \frac{(\mathbf{H}_i \boldsymbol{r}_i)(\mathbf{H}_i \boldsymbol{r}_i)^T}{\boldsymbol{r}_i^T \mathbf{H}_i \boldsymbol{r}_i}.
\end{aligned}$$
4. Set $i = i + 1$, goto step 2.

Remark.

The quasi-Newton method, also known as the variable metric method, was introduced by Davidon (1959) and was further investigated by Fletcher and Powell (1963). Subsequently, it has been referred to as the Davidon–Fletcher–Powell (DFP) method. For a survey of this work, see Dennis and

Schnabel (1983), which also contains a long list of references. Broyden (1967) used the symmetric rank-one update method. More efficient methods have been introduced by Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970), these can be abbreviated as the BFGS methods. The quasi-Newton method is known to have a super-linear performance [see, for example, Nocedal and Wright (1999), p. 216)].

2.5.5 Conjugate Directions

This method was proposed by Hestenes and Stiefel in 1952 in solving large systems of linear equations of the form $\mathbf{H}\boldsymbol{v} = \boldsymbol{g}$. The method was subsequently developed to handle optimization problems by Fletcher and Reeves (1964) and Polak and Ribière (1969) based on the idea that the local behavior of the function to be optimized is quite similar to that of a quadratic function.

For the purpose of motivation, we start with the linear conjugate problem in which the objective function is exactly quadratic. Suppose the function being maximized is given by

$$f(\boldsymbol{\beta}) = -\frac{1}{2}\boldsymbol{\beta}^T\mathbf{H}\boldsymbol{\beta} + \boldsymbol{b}^T\boldsymbol{\beta},$$

where $\mathbf{H}$ denotes an $m \times m$, symmetric, positive-definite matrix. If $\mathbf{H}$ is diagonal, then the optimization problem is reduced to m one-dimensional (coordinate-wise) optimization problems. If $\mathbf{H}$ is not diagonal, the m one-dimensional problems can still be achieved by changing the coordinates. Suppose this is done through

$$\boldsymbol{\beta} = \boldsymbol{\beta}_0 + \sum_{j=0}^{m-1} \lambda_j \boldsymbol{v}_j, \qquad \boldsymbol{\beta} \in \mathbb{R}^m,$$

where the set of nonzero vectors $\{\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}\}$ satisfies

$$\boldsymbol{v}_i^T\mathbf{H}\boldsymbol{v}_j = 0 \qquad \text{for all } i \neq j. \tag{2.5.4}$$

Then

$$\begin{aligned} f(\boldsymbol{\beta}) &= f\Big(\boldsymbol{\beta}_0 + \sum_{j=0}^{m-1} \lambda_j \boldsymbol{v}_j\Big) \\ &= f(\boldsymbol{\beta}_0) + \sum_{j=0}^{m-1} \Big(-\frac{1}{2}(\boldsymbol{v}_j^T\mathbf{H}\boldsymbol{v}_j)\lambda_j^2 + (\boldsymbol{b} - \mathbf{H}\boldsymbol{\beta}_0)^T\boldsymbol{v}_j\lambda_j\Big), \end{aligned}$$

which can be seen as decomposing the original optimization problem in $\mathbb{R}^m$ into m one-dimensional problems involving λ_j, $j = 0, 1, \ldots, m-1$.

The vectors $\boldsymbol{v}_0, \boldsymbol{v}_1, \ldots, \boldsymbol{v}_{m-1}$ satisfying (2.5.4) are called the *conjugate directions.* It can be easily verified that the set of conjugate directions is linearly independent, thereby forming a basis for $\mathbb{R}^m$. There are many ways to select conjugate directions. For example, one can select the eigenvectors of $\mathbf{H}$. But this is not efficient for large scale problems. Alternatively, one can modify the Gram–Schmidt orthogonalization process to produce such a set, but this is also not practical because of the storage requirement. In practice, the conjugate directions are generated by using an updating method, that is, the new direction $\boldsymbol{v}_{i+1}$ is produced by using only the previous direction $\boldsymbol{v}_i$. Before giving the details, we write the intermediate solution as (with $\boldsymbol{\beta}_0$ being the starting value)

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j + \lambda_j \boldsymbol{v}_j = \boldsymbol{\beta}_0 + \sum_{i=0}^{j} \lambda_i \boldsymbol{v}_i, \qquad j = 0, 1, \ldots, m-2,$$

where

$$\lambda_j = -\frac{(\boldsymbol{b} - \mathbf{H}\boldsymbol{\beta}_0)^T \boldsymbol{v}_j}{\boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j}$$

maximizes f along the direction $\boldsymbol{v}_j$. Set $\boldsymbol{g}_j = \nabla f(\boldsymbol{\beta}_j)$ for $j = 0, 1, \ldots, m$. Then by the definition of λ_j,

$$\boldsymbol{g}_j^T \boldsymbol{v}_{j-1} = 0 \qquad j = 1, \ldots, m. \tag{2.5.5}$$

Moreover,

$$\boldsymbol{g}_{j+1} = \boldsymbol{g}_j - \lambda_j \mathbf{H} \boldsymbol{v}_j, \qquad j = 0, 1, \ldots, m-1.$$

Here is the detail for generating the conjugate directions recursively. Start with $\boldsymbol{v}_0 = -\boldsymbol{g}_0$ and update it via

$$\boldsymbol{v}_{j+1} = -\boldsymbol{g}_{j+1} + \alpha_{j+1} \boldsymbol{v}_j, \qquad j \geq 0, \tag{2.5.6}$$

with α_{j+1} selected so that $\boldsymbol{v}_j$ and $\boldsymbol{v}_{j+1}$ are conjugate with respect to $\mathbf{H}$; that is,

$$\alpha_{j+1} = \frac{\boldsymbol{g}_{j+1}^T \mathbf{H} \boldsymbol{v}_j}{\boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j}.$$

Then the vectors $\boldsymbol{v}_0, \ldots, \boldsymbol{v}_{m-1}$ are conjugate directions. The method is efficient and requires less storage in that the new direction is produced by the previous one, not the whole past.

We summarize this process as follows.

1. Start with $\boldsymbol{\beta}_0$ as initial value, evaluate $\boldsymbol{g}_0 = \nabla f(\boldsymbol{\beta}_0)$ and $\boldsymbol{v}_0 = -\boldsymbol{g}_0$.

2. For $j = 0, 1, \ldots, m-1$, evaluate

$$\begin{aligned}
\lambda_j &= \boldsymbol{g}_0^T \boldsymbol{v}_j / \boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j, \\
\boldsymbol{\beta}_{j+1} &= \boldsymbol{\beta}_j + \lambda_j \boldsymbol{v}_j, \\
\boldsymbol{g}_{j+1} &= \boldsymbol{g}_j - \lambda_j \mathbf{H} \boldsymbol{v}_j, \\
\alpha_{j+1} &= \boldsymbol{g}_{j+1}^T \mathbf{H} \boldsymbol{v}_j / \boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j, \\
\boldsymbol{v}_{j+1} &= -\boldsymbol{g}_{j+1} + \lambda_{j+1} \alpha_j \boldsymbol{v}_j.
\end{aligned}$$

Note that the solutions, gradients and the conjugate directions are updated automatically. It can be shown that it takes m steps to solve the quadratic problem where Newton's method obtains the solution in one iteration. In general the one-dimensional problem is very computing intensive, so the use of the conjugate approach can be justified only if the calculation of the Hessian matrix can be avoided.

From (2.5.5) and (2.5.6),

$$\begin{aligned}
\lambda_j &= \boldsymbol{g}_j^T \boldsymbol{v}_j / \boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j, \\
\boldsymbol{g}_j^T \boldsymbol{v}_j &= \boldsymbol{g}_j^T (-\boldsymbol{g}_j + \alpha_j \boldsymbol{v}_{j-1}) = -\boldsymbol{g}_j^T \boldsymbol{g}_j, \\
\boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j &= (-\boldsymbol{g}_j + \alpha_j \boldsymbol{v}_{j-1})^T \mathbf{H} \boldsymbol{v}_j = -\boldsymbol{g}_j^T \mathbf{H} \boldsymbol{v}_j, \\
\boldsymbol{g}_j^T \boldsymbol{g}_{j+1} &= \boldsymbol{g}_j^T \boldsymbol{g}_j - (\boldsymbol{g}_j^T \boldsymbol{v}_j / \boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j) \boldsymbol{g}_j^T \mathbf{H} \boldsymbol{v}_j = 0.
\end{aligned}$$

Thus,

$$\begin{aligned}
\alpha_{j+1} = \frac{\boldsymbol{g}_{j+1}^T \mathbf{H} \boldsymbol{v}_j}{\boldsymbol{v}_j^T \mathbf{H} \boldsymbol{v}_j} &= \frac{\boldsymbol{g}_{j+1}^T (\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)}{\boldsymbol{v}_j^T (\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)} \\
&= -\frac{\boldsymbol{g}_{j+1}^T (\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)}{\boldsymbol{v}_j^T \boldsymbol{g}_j} \\
&= \frac{\boldsymbol{g}_{j+1}^T (\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)}{\boldsymbol{g}_j^T \boldsymbol{g}_j} \\
&= \frac{\boldsymbol{g}_{j+1}^T \boldsymbol{g}_{j+1}}{\boldsymbol{g}_j^T \boldsymbol{g}_j}.
\end{aligned}$$

For quadratic functions, there are many equivalent ways to compute the coefficient α_j. However, these methods yield quite different results for general function f. In this case, the solution to the optimization is still given by the line search method:

$$\boldsymbol{\beta}_{j+1} = \boldsymbol{\beta}_j + \lambda_j \boldsymbol{v}_j = \boldsymbol{\beta}_0 + \sum_{i=0}^{j} \lambda_i \boldsymbol{v}_i, \qquad j = 0, 1, \ldots, m-2,$$

where

$$\lambda_j = \underset{\lambda}{\operatorname{argmax}}\, f(\boldsymbol{\beta}_j + \lambda \boldsymbol{v}_j),$$

with the conjugate directions generated by

$$\boldsymbol{v}_{j+1} = -\boldsymbol{g}_{j+1} + \alpha_{j+1}\boldsymbol{v}_j, \qquad j \geq 0.$$

Typical choices for α_j are

1. the Polak–Ribière formula:
$$\alpha_{j+1} = \frac{(\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)^T \boldsymbol{g}_{j+1}}{\|\boldsymbol{g}_j\|^2},$$
2. the Fletcher–Reeves formula:
$$\alpha_{j+1} = \frac{\|\boldsymbol{g}_{j+1}\|^2}{\|\boldsymbol{g}_j\|^2},$$
3. the Hestenes–Stiefel formula:
$$\alpha_{j+1} = \frac{(\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)^T \boldsymbol{g}_{j+1}}{(\boldsymbol{g}_{j+1} - \boldsymbol{g}_j)^T \boldsymbol{v}_j}.$$

Among these methods, the Fletcher–Reeves is the least efficient. The methods by Polak–Ribière and Hestenes–Stiefel are considered to be more practical. See Gilbet and Nocedal (1992) for an extensive comparative numerical study. For high dimensional problems such as the speech problem in Chapter 5, these methods are more efficient than Newton's method since the Hessian matrix is not required. The above paper also contains some convergent results. See also the illuminating discussion in Section 5.2 of Nocedal and Wright (1999).

2.5.6 One-Dimensional Optimization — step length search

In the previous discussion, we were led to consider the one-dimensional search problem:

$$\max \phi(\lambda), \qquad \phi(\lambda) = f(\boldsymbol{\beta} + \lambda \mathbf{d}).$$

Or equivalently,

$$\phi'(\lambda) = \frac{d}{d\lambda} f(\boldsymbol{\beta}_j + \lambda \boldsymbol{g}_j) = \langle \nabla f(\boldsymbol{\beta}_j + \lambda \boldsymbol{g}_j), \boldsymbol{g}_j \rangle = 0.$$

Sophisticated line search can be quite complicated. Here is preview of the popular methods. If the computation of the second derivative is not an issue, then by far the Newton's method is the most efficient as it has the quadratically convergence property. The secant method uses a discrete method to approximate the derivative of the Newton method. The tradeoff is the efficiency, it has the super-linear convergence rate. The bisection method is the slowest among the three. The success of these methods depends on the initial value.

2.5.7 Step-halving

Suppose the Hessian is everywhere negative definite and that the objective function has a (necessarily unique) maximum $\widehat{\boldsymbol{\beta}}$. The Newton–Raphson method for computing $\widehat{\boldsymbol{\beta}}$ is to start with an initial approximation $\boldsymbol{\beta}_0$ and iteratively determine $\boldsymbol{\beta}_{m+1}$ from $\boldsymbol{\beta}_m$ according to

$$\boldsymbol{\beta}_{m+1} = \boldsymbol{\beta}_m - [\mathbf{H}(\boldsymbol{\beta}_m)]^{-1}\nabla f(\boldsymbol{\beta}_m), \qquad m \geq 0.$$

If the initial approximation $\boldsymbol{\beta}_0$ to $\widehat{\boldsymbol{\beta}}$ is sufficiently accurate, then $\boldsymbol{\beta}_m$ converges to $\widehat{\boldsymbol{\beta}}$ as $m \to \infty$ and, in fact, the convergence is of second order; that is, $|\boldsymbol{\beta}_{m+1} - \widehat{\boldsymbol{\beta}}| = O(|\boldsymbol{\beta}_m - \widehat{\boldsymbol{\beta}}|^2)$. Thus, roughly speaking, the number of accurate digits in the approximation $\boldsymbol{\beta}_m$ to $\widehat{\boldsymbol{\beta}}$ doubles at each iteration. We stop the interations when $f(\boldsymbol{\beta}_{m+1}) - f(\boldsymbol{\beta}_m) \leq \epsilon$, where (say) $\epsilon = 10^{-6}$

If the initial approximation is not sufficiently accurate, however, then $\boldsymbol{\beta}_m$ can diverge as $m \to \infty$. In order to guarantee convergence from any starting value, we use the Newton–Raphson method with step-halving, in which $\boldsymbol{\beta}_{m+1}$ is determined from $\boldsymbol{\beta}_m$ according to the formula

$$\boldsymbol{\beta}_{m+1} = \boldsymbol{\beta}_m - 2^{-\nu_m}[\mathbf{H}(\boldsymbol{\beta}_m)]^{-1}\nabla f(\boldsymbol{\beta}_m);$$

here ν_m is the smallest nonnegative integer such that

$$\begin{aligned} f(\boldsymbol{\beta}_m - 2^{-\nu_m}[\mathbf{H}(\boldsymbol{\beta}_m)]^{-1}\nabla f(\boldsymbol{\beta}_m)) &\geq f(\boldsymbol{\beta}_m - 2^{-\nu_m-1}[\mathbf{H}(\boldsymbol{\beta}_m)]^{-1}\nabla f(\boldsymbol{\beta}_m)) \\ &\geq f(\boldsymbol{\beta}_m). \end{aligned}$$

(Here, the second inequality is redundant, being a consequence of the first inequality and the concavity of the function.)

In verifying the global convergence of the Newton–Raphson method with step-halving, we can assume that $\nabla f(\boldsymbol{\beta}_m) \neq \mathbf{0}$ for $m \geq 0$ (otherwise the stopping rule will lead to stopping at some $\boldsymbol{\beta}_m = \widehat{\boldsymbol{\beta}}$). Then ν_m is well defined for $m \geq 0$ and, by the strict concavity of the function, $f(\boldsymbol{\beta}_{m+1}) > f(\boldsymbol{\beta}_m)$ for $m \geq 0$. Consequently, $f(\boldsymbol{\beta}_m)$ increases to a finite limit $\hat{f}$ as $m \to \infty$. Also, the points $\boldsymbol{\beta}_m$, $m \geq 0$, are distinct and lie in a compact set and hence they have an accumulation point $\boldsymbol{\beta}'$. Moreover, the integers ν_m, $m \geq 0$, are bounded. Letting m increase along a sequence such that $\boldsymbol{\beta}_m \to \boldsymbol{\beta}'$ and $\nu_m = \nu$, we conclude that

$$\begin{aligned} \hat{f} &\geq f(\boldsymbol{\beta}' - 2^{-\nu}[\mathbf{H}(\boldsymbol{\beta}')]^{-1}\nabla f(\boldsymbol{\beta}')) \\ &\geq f(\boldsymbol{\beta}' - 2^{-\nu-1}[\mathbf{H}(\boldsymbol{\beta}')]^{-1}\nabla f(\boldsymbol{\beta}')) \\ &\geq f(\boldsymbol{\beta}') \\ &\geq \hat{f} \end{aligned}$$

and hence that

$$f(\boldsymbol{\beta}' - 2^{-\nu}[\mathbf{H}(\boldsymbol{\beta}')]^{-1}\nabla f(\boldsymbol{\beta}')) = f(\boldsymbol{\beta}' - 2^{-\nu-1}[\mathbf{H}(\boldsymbol{\beta}')]^{-1}\nabla f(\boldsymbol{\beta}')) = f(\boldsymbol{\beta}').$$

Since the objective function is concave and its Hessian is negative definite, we see that $\nabla f(\boldsymbol{\beta}') = \mathbf{0}$ and hence that $\boldsymbol{\beta}' = \widehat{\boldsymbol{\beta}}$. Therefore, $\boldsymbol{\beta}_m \to \widehat{\boldsymbol{\beta}}$ as $m \to \infty$.

2.5.8 How to terminate an iteration

To guard against a particular method from running into an infinite loop, some rule would have be adopted to halt the iterations. In fact, at some step of the iteration where the improvement or displacement is less than a prespecified small constant, the iteration will be forced to stop. The stopping rule is problem specific, more details and examples will be described in later chapters.

2.6 B-splines with repeated knots

There are several ways to introduce B-splines. One could start with the recursive relationship as described in Kincaid and Cheney (1996, p.392), or the determinant approach as given in Schumaker (1981), or the divided difference in de Boor (1978). The first approach is very elegant, but it suffers the drawback of not being able to handle duplicated or repeated knots. The second approach demands a working knowledge of matrices and determinants and many of their properties in addition to the divided difference. For all of our practical purposes, the approach considered by de Boor seems to be the most appropriate since it strikes the balance between numerical properties and the theoretical development having repeated knots. The easiest way to understand divided differences is through polynomial interpolation, which will be discussed next.

2.6.1 Polynomial interpolation

For a given function $f(\cdot)$ and a list of distinct real numbers $x_0, x_1, \ldots, x_n$, there exists a unique polynomial p of order at most n such that

$$p(x_i) = f(x_i), \qquad i = 0, 1, \ldots, n.$$

Such a polynomial p is said to interpolate f at $x_0, x_1, \ldots, x_n$ and can be easily constructed via the *Newton form*:

$$\begin{aligned} &p_0(x_0) = f(x_0),\\ &\quad p_k(x) = p_{k-1}(x) + c_k(x - x_0)(x - x_1)\cdots(x - x_{k-1}), \quad k = 1, \ldots, n, \end{aligned}$$

where c_k is chosen such that $p_k(x_k) = f(x_k)$, $k = 1, 2, \ldots, n$. Note that p_k interpolates f at $x_0, x_1, \ldots, x_{k-1}$, and the desired interpolating polynomial is $p = p_n$.

There is only one such polynomial. In fact, let q be another polynomial that interpolates f at $x_0, x_1, \ldots, x_n$. Then $p-q$ is a polynomial of order at most $n+1$ having $n+1$ distinct zeros, so it must be the zero polynomial. Hence $p = q$.

2.6.2 Divided difference – efficient way to evaluate the coefficients

In practice, the coefficients c_k are computed by a more efficient algorithm. We observe that each leading coefficient c_k of p_k depends on f and $x_0, x_1, \ldots, x_{k-1}$. It is convenient to denote it by $c_k = [x_0, x_1, \cdots, x_{k-1}]f$ and refer it to as the *divided difference* of order k. To see the reason behind this terminology, it is instructive to consider the case when $k = 2$. We have

$$f(x_1) = p_1(x_1) = p_0(x_1) + c_1(x_1 - x_0)$$

with

$$c_1 = \frac{f(x_1) - f(x_0)}{x_1 - x_0},$$

which is in a form of divided difference. It is even more instructive to examine the case when x_1 is approaching x_0. In this case,

$$c_1 = \lim_{x_1 \to x_0} \frac{f(x_1) - f(x_0)}{x_1 - x_0} = f'(x_0).$$

This suggests a way to extend the notion of interpolating polynomials to a list of repeated numbers. This will be discussed next.

2.6.3 Divided differences with repetition

The polynomial p is said to interpolate f at $x_0, x_1, \ldots, x_n$ if

$$p^{(j)}(x) = f^{(j)}(x), \qquad j = 0, 1, \ldots, k-1,$$

for each x that occurs k times in the list $x_0, x_1, \ldots, x_n$.

Let $x_0, x_1, \ldots, x_n$ be a list of real numbers in which no element is repeated more than k times. Let f belong to C^{k-1} on an interval containing these numbers. Then there exists a unique polynomial of order at most $n+1$ that interpolates f at these points.

The desired polynomial has $n+1$ coefficients. These and the $n+1$ interpolating conditions set up a square system of $n+1$ linear equations with $n+1$ unknowns. This system is solvable if and only if the solution to the homogeneous equation $Au = 0$ is $u = 0$. In the current interpolation problem, the homogeneous problem is to find a polynomial p of order at most $n+1$ interpolating 0 at the given points. Such a polynomial has the form

$$p(x) = c\prod_{i=0}^{n}(x - x_i).$$

But p has order at most $n+1$, so $p=0$ as claimed.

In the general case, we define $[x_0, x_1, \ldots, x_n]f$ as the coefficient of x^n of a polynomial p that interpolates f at $x_0, x_1, \ldots, x_n$. If there are k repetitions in the list, then this definition requires the existence $f^{(k-1)}$. Otherwise, the divided difference is not well defined.

With this definition in mind, we claim that the interpolating polynomial takes the form

$$p(x) = f(x_0) + \sum_{j=1}^{n} [x_0, \ldots, x_j]f \prod_{i=0}^{j-1} (x - x_i). \tag{2.6.1}$$

In fact, $n=0$ asserts that $p(x) = f(x_0)$ interpolates f at x_0. Suppose now that the polynomial

$$q(x) = f(x_0) + \sum_{j=1}^{n-1} [x_0, \ldots, x_j]f \prod_{i=0}^{j-1} (x - x_i)$$

interpolates f at $x_0, x_1, \ldots, x_{n-1}$. Then

$$p(x) - [x_0, \ldots, x_n]f \prod_{i=0}^{n-1} (x - x_i)$$

is of order at most n, which also interpolates f at $x_0, x_1, \ldots, x_{n-1}$. It follows from the uniqueness of q that

$$p(x) - [x_0, \ldots, x_n]f \prod_{i=0}^{n-1} (x - x_i) = q(x),$$

or,

$$p(x) = f(x_0) + \sum_{j=1}^{n} [x_0, \ldots, x_j]f \prod_{i=0}^{j-1} (x - x_i)$$

as desired.

2.6.4 Properties of the divided difference

This section describes some of the important properties of the divided difference. The list is not an exhausted account but it serves to provide enough information for us to establish the important results for B-splines. More detailed discussions of this topic can be found in de Boor (1978), DeVore and Lorentz (1993), and Schumaker (1981).

1. Symmetry: $[x_0, x_1, \ldots, x_n]f$ depends only on the numbers $x_0, \ldots, x_n$ and not on the order in which they appear. This is because the interpolating polynomial depends only on the points of interpolation.

2. Divided difference for the product of two functions: Let g and h be two functions, then

$$[x_0, x_1, \ldots, x_n](gh) = \sum_{r=0}^{n}([x_0, x_1, \ldots, x_r]g)([x_r, x_{r+1}, \ldots, x_n]h).$$

This is called the Leibniz's formula which will be used to derive the recursive relationship for B-splines. According to (2.6.1), the function

$$F(x) = \sum_{r=0}^{n}[x_0, x_1, \ldots, x_r]g \prod_{i=0}^{r-1}(x - x_i) \times \sum_{s=0}^{n}[x_s, x_{s+1}, \ldots, x_n]h \prod_{j=s+1}^{n}(x - x_j)$$

agrees with $f = gh$ at $x_0, x_1, \ldots, x_n$. To write it as

$$F(x) = \sum_{r,s=0}^{n} = \sum_{r \le s} + \sum_{r>s}.$$

With $\sum_{r>s}$ vanishing at $x_0, \ldots, x_n$, the first sum $\sum_{r\le s}$ is a polynomial of order $n+1$ which agrees with $f = gh$. Its leading coefficient

$$\sum_{r=s}([x_0, x_1, \ldots, x_r]g)([x_r, x_1, \ldots, x_n]h)$$

must be $[x_0, x_1, \ldots, x_n]f$.

3. $[x_0, x_1, \ldots, x_n]g = 0$ for polynomial g of order less than $n+1$. This will be used to show that B-splines have compact support.

4. If f is n times differentiable in (a, b) and if $x_0, x_1, \ldots, x_n$ are distinct points in $[a, b]$, then there exists $\xi \in (a, b)$ such that

$$[x_0, x_1, \ldots, x_n]f = \frac{f^{(n)}(\xi)}{n!} \tag{2.6.2}$$

In fact, let p_{n+1} be the polynomial of order $n+1$ that agrees with f at the indicated points. Then the function $e^{(n)} = f - p_n$ is n times differentiable and has $n+1$ zeros in (a, b). Apply Rolle's theorem n times to conclude that $e^{(n)}$ has a zero ξ in (a, b):

$$0 = e^{(n)}(\xi) = f^{(n)}(\xi) - p_n^{(n)}(\xi).$$

Now the desired results from the definition of the divided difference:

$$p_n^{(n)}(\xi) = n![x_0, x_1, \ldots, x_n]f.$$

2.6.5 Computing the divided differences recursively

Let $x_0 \leq x_1 \leq \cdots \leq x_n$. Then

$$[x_0, x_1, \ldots, x_n]f = \begin{cases} \dfrac{[x_1, x_2, \ldots, x_n]f - [x_0, x_1, \ldots, x_{n-1}]f}{x_n - x_0}, & \text{if } x_n \neq x_0, \\ f^{(n)}(x_0)/n!, & \text{otherwise.} \end{cases} \tag{2.6.3}$$

In fact, if $x_n = x_0$, then it is easy to see that $f^{(n)}(x_0)/n!$ is the leading coefficient of the polynomial

$$p(x) = \sum_{k=0}^{n} \frac{1}{k!} f^{(k)}(x_0)(x - x_0)^k,$$

which interpolates f at $x_0 = x_1 = \cdots = x_n$. Suppose now that $x_0 \neq x_n$. Let p_{k+1} denote the polynomial of order $k+1$ that interpolates f at $x_0, x_1, \ldots, x_k$. Let q be the polynomial interpolating f at $x_1, \ldots, x_n$. Then by the uniqueness of the interpolating polynomials:

$$p_n(x) = q(x) + \frac{x - x_n}{x_n - x_0}[q(x) - p_{n-1}(x)], \quad x_n \neq x_0.$$

Equating the leading coefficients, we have

$$[x_0, x_1, \ldots, x_n]f = \frac{[x_1, x_2, \ldots, x_n]f - [x_0, x_1, \ldots, x_{n-1}]f}{x_n - x_0}, \qquad x_n \neq x_0.$$

Alternatively, the recursive formula can also be written as

$$[x_0, x_1, \ldots, x_{n-1}, x_n]f = [x_0, x_n][x_1, x_2, \ldots, x_{n-1}, \cdot]f. \tag{2.6.4}$$

The computation of the divided differences is most conveniently carried out using the following table.

x	$f[]$	$f[,]$	$f[,,]$	$f[,,,]$
x_0	$f[x_0]$			
		$f[x_0, x_1]$		
x_1	$f[x_1]$		$f[x_0, x_1, x_2]$	
		$f[x_1, x_2]$		$f[x_0, x_1, x_2, x_3]$
x_2	$f[x_2]$		$f[x_1, x_2, x_3]$	
		$f[x_2, x_3]$		
x_3	$f[x_3]$			

To save storage space in a computer, the following version is preferred in numerical implementation of the procedure:

x_0	$f[x_0]$	$f[x_0, x_1]$	$f[x_0, x_1, x_2]$	$f[x_0, x_1, x_2, x_3]$
x_1	$f[x_1]$	$f[x_1, x_2]$	$f[x_1, x_2, x_3]$	
x_2	$f[x_2]$	$f[x_2, x_3]$		
x_3	$f[x_3]$			

Example. Compute the divided differences based on the following function values.

x	3	1	5	6
$f(x)$	1	-3	2	4

Divided differences:

3	1	2	$-3/8$	$7/40$
1	-3	$5/4$	$3/20$	
5	2	2		
6	$4/3$			

2.6.6 B-Splines with repeated knots

Let (t_i) denote a non-decreasing sequence of numbers. The i-th B-spline of order k (or degree $k-1$) is defined by

$$B_i(x) = B_{i,k}(x) = (t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1}. \tag{2.6.5}$$

Here $(y-x)_+^0 = \text{ind}(y \geq x) = 1$ if $y \geq x$, and zero otherwise; $(y-x)_+^{k-1} = (y-x)^{k-1}\text{ind}(y \geq x)$ for $k \geq 2$.

We should remark that the current definition includes the previous one with distinct knots as a special case. For example, suppose $t_1 < t_2 < t_3$. Then by the first part of (2.6.3), or simply apply (2.6.4),

$$B_{1,1}(x) = (t_2 - x)_+^0 - (t_1 - x)_+^0 = \text{ind}(t_1 \leq x < t_2),$$

and

$$\begin{aligned} B_{i,2}(x) &= (t_3 - t_1)[t_1, t_2, t_3](\cdot - x)_+ \\ &= (t_3 - t_1)[t_1, t_3]\,[t_2, \cdot](\cdot - x)_+ \\ &= [t_2, t_3](\cdot - x)_+ - [t_2, t_1](\cdot - x)_+ \\ &= \begin{cases} \dfrac{x - t_1}{t_2 - t_1}, & t_1 \leq x < t_2; \\ \dfrac{t_3 - x}{t_3 - t_2}, & t_2 \leq x < t_3. \end{cases} \end{aligned}$$

These agree with the those given in Section 2.1.5. In fact, one can also write the linear B-splines according to

$$B_{i,2}(x) = \frac{x - t_1}{t_2 - t_1} B_{i,1}(x) + \frac{t_3 - x}{t_3 - t_2} B_{i+1,1}(x),$$

which reminds us the recursive formula for computing the B-splines described in Section 2.1.5. As we will see shortly, the important properties developed previously for distinct knots in fact hold for the B-splines having repeated knots. Some of these are easy consequences of the properties of the divided differences; others are more complicated to establish. We begin with the easy ones.

1. Support of B-splines:

$$B_{i,k}(x) = 0 \quad \text{for} \quad x \notin [t_i, t_{i+k}].$$

For if x is outside $[t_i, t_{i+k}]$, $(\cdot - x)_+^{k-1}$ is either zero or a polynomial of order at most k. In either case, $[t_i, \ldots, t_{i+k}](\cdot - x)_+^{k-1} = 0$ by Property 3 of Section 2.6.4. That is, there are only k B-splines having any particular interval $[t_i, t_{i+1}]$ in their support.

2. $\sum B_i(x) = \sum_{i=u+1-k}^{v} B_i(x) = 1$ for $t_u < x < t_v$. This follows from (2.6.3). The result indicates that B-splines form a *partition of unity.*

3. Recursive formula. Write

$$(y - x)_+^{k-1} = (y - x)(y - x)_+^{k-2}.$$

By the Leibniz's formula,

$$[t_i, \ldots, t_{i+k}](y - x)_+^{k-1} = (t_i - x)[t_i, \ldots, t_{i+k}](y - x)_+^{k-2} \\ + 1 \cdot [t_{i+1}, \ldots, t_{i+k}](y - x)_+^{k-2}.$$

Apply (2.6.3) to the first term on the right hand side to obtain,

$$[t_i, \ldots, t_{i+k}](y - x)_+^{k-1} = \frac{x - t_i}{t_{i+k} - t_i}[t_i, \ldots, t_{i+k-1}](y - x)_+^{k-2} \\ + \frac{t_{i+k} - x}{t_{i+k} - t_i}[t_{i+1}, \ldots, t_{i+k}](y - x)_+^{k-2}.$$

Therefore,

$$B_{i,k}(x) = \frac{x - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(x) + \frac{t_{i+k} - x}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(x), \qquad k \geq 2.$$

Here we adopt the convention that zero divided by zero is zero. This recursive formula is very important for computing the B-splines in practice. It is numerically stable and efficient. See de Boor (1978) for more details. Figures 2.9–2.12 present a sequence of B-splines of orders 1, 2, 3 and 4 computed using this formula.

4. It follows from the recursive relationship that $B_i(\cdot) \geq 0$ for all x. In fact, by the inductive argument given in Property 2.1.2 of Section 2.1.5, $B_{i,k}(x) > 0$ on (t_i, t_{i+k}).

5. Derivatives of B-splines:

$$\frac{d}{dx} B_i^k(x) = \frac{k - 1}{t_{i+k-1} - t_i} B_i^{k-1}(x) - \frac{k - 1}{t_{i+k} - t_{i+1}} B_{i+1}^{k-1}(x).$$

In fact, this follows from

$$\frac{d}{dx}(\cdot - x)_+^j = j(\cdot - x)_+^{j-1}$$

and

$$\frac{d}{dx}B_i^k(x) = -(k-1)([t_{i+1}\cdots t_{i+k}] - [t_i \cdots t_{i+k-1}])(\cdot - x)_+^{k-2}.$$

6. According to Theorem 4.2 of DeVore and Lorentz (1993), there is a positive constant M_0 such that

$$M_0^{-1}J^{-1}|\boldsymbol{\beta}|^2 \le \int \Big|\sum_j \beta_j B_j\Big|^2 \le M_0 J^{-1}|\boldsymbol{\beta}|^2, \qquad \boldsymbol{\beta} \in \mathbb{R}^J.$$

Remark.

Our definition of B-spline is adapted from de Boor (1978, p.108), which is referred to as the normalized B-splines in Schumaker (1981, p.124) with a slightly different appearance given by

$$N_i(x) = N_i^k(x) = (-1)^k(t_{i+k} - t_i)[t_i, \ldots, t_{i+k}](x - \cdot)_+^{k-1}.$$

It is easy to see that this is identical to (2.6.5).

2.6.7 Basis: Curry–Schoenberg Theorem

It was established in Section 2.1.5 that the B-splines of order k with distinct knots form a basis for the space of splines of order k. In establishing that result, we employed $k-1$ times differentiable splines by requiring knots to be distinct. We now consider a generalization based on B-splines with repeated knots in which a variable differentiability condition at each knot (but up to the order k) is allowed. This result is known as the *Curry–Schoenberg Theorem.*

Let m denote a positive integer. Let $\boldsymbol{\tau} = (\tau_0, \tau_1, \ldots, \tau_m, \tau_{m+1})$ denote a sequence of strictly increasing numbers with $-\infty \le \tau_0 < \tau_{m+1} \le \infty$ and let $\boldsymbol{\nu} = (\nu_1, \nu_2, \ldots, \nu_m)$ denote a sequence of integers satisfying $\nu_j \le k$, $j = 1, \ldots, m$. Let $P_{k,\boldsymbol{\tau},\boldsymbol{\nu}}$ denote the space of piecewise polynomials having order k, knot sequence $\boldsymbol{\tau}$ and satisfying

$$\text{jump}_{\tau_i} D^{j-1}f := D^{j-1}f(\tau_i^+) - D^{j-1}f(\tau_i^-) = 0, \qquad j = 1, \ldots, \nu_i, \quad i = 1, \ldots, m.$$

Set

$$d \equiv d_{k,\boldsymbol{\nu}} = k + \sum_{i=1}^m (k - \nu_i) = k(m+1) - (\nu_1 + \cdots + \nu_m) = \text{dim}P_{k,\boldsymbol{\tau},\boldsymbol{\nu}},$$

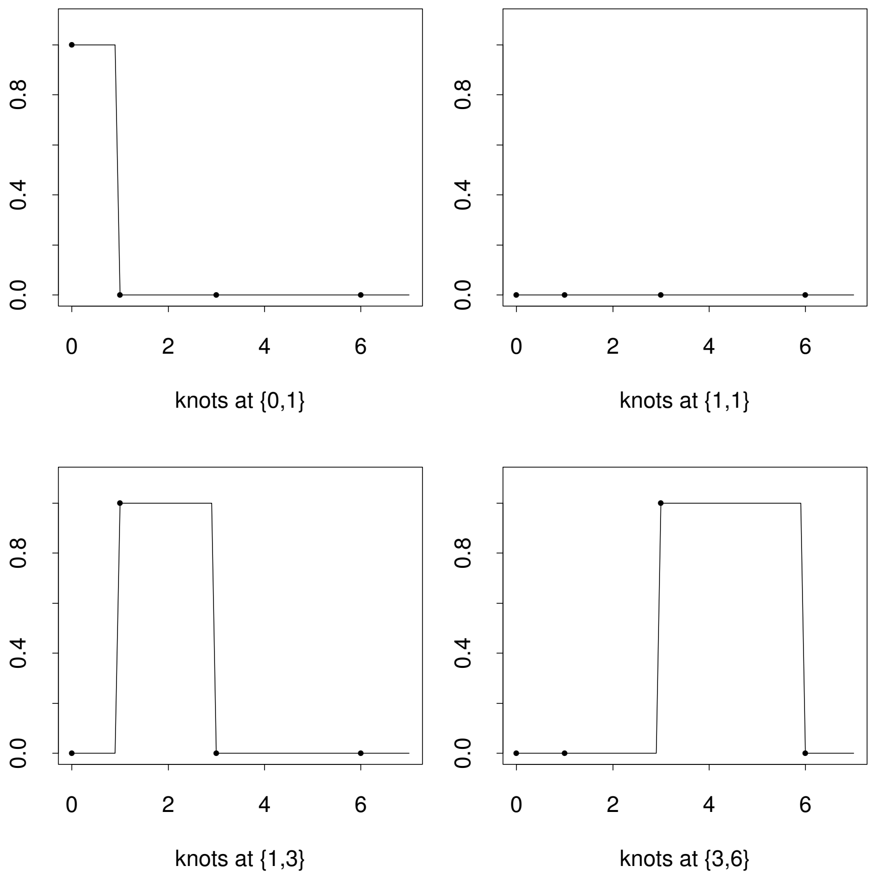

FIGURE 2.9. A sequence of piecewise constant B-splines having knots $\{0, 1, 1, 3, 6\}$.

and let $\boldsymbol{t} = (t_1, t_2, \ldots, t_{d+k})$ denote a sequence of non-decreasing numbers such that

1. $\tau_0 \le t_1 \le t_2 \le \cdots \le t_k \le \tau_1$ and $\tau_m \le t_{d+1} \le t_{d+2} \le \cdots \le t_{d+k} \le \tau_{m+1}$;
2. the number τ_i occur exactly $k - \nu_i$ times in $\boldsymbol{t}$, $i = 1, 2, \ldots, m$.

Then, the sequence $B_{1,k}, \ldots, B_{d,k}$ of B-splines of order k having the knot sequence $\boldsymbol{t}$ is a basis for $P_{k,\boldsymbol{\tau},\boldsymbol{\nu}}$.

A proof of this important result is given in Schumaker (1981, pp 123–124) and de Boor (1978, pp 113–118). In associating with this result, it is always helpful to remember that

$$\underbrace{\text{number of continuity conditions at } \tau}_{\nu} + \underbrace{\text{number of knots at } \tau}_{(k-\nu)} = k.$$

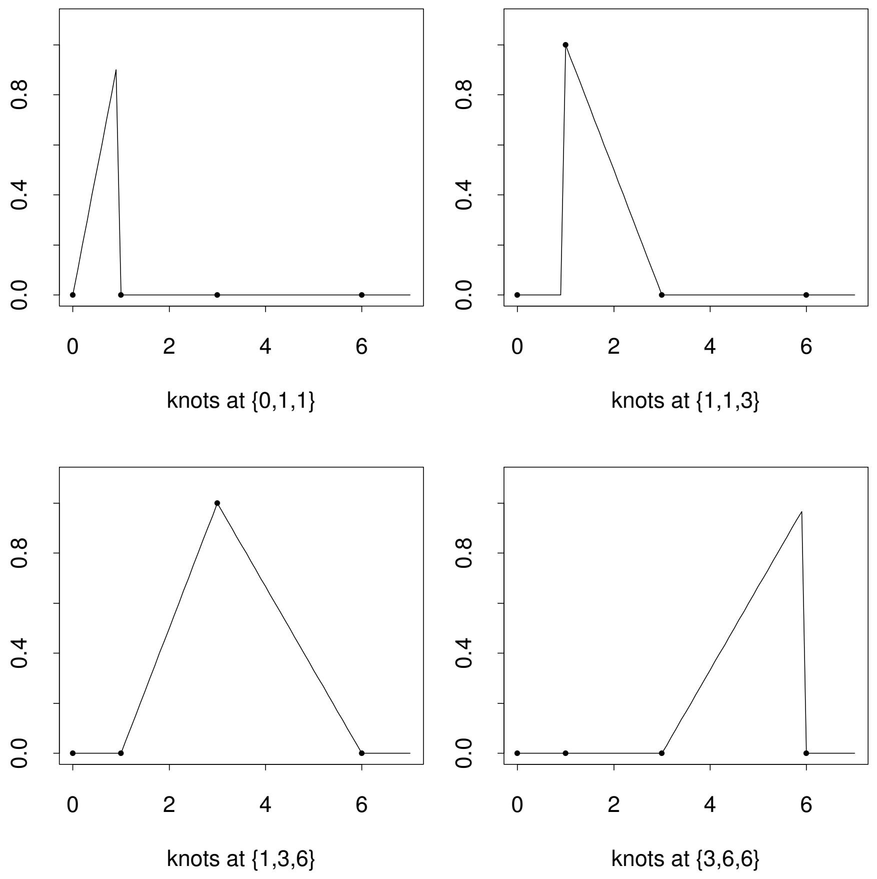

FIGURE 2.10. A sequence of linear B-splines having knots $\{0, 1, 1, 3, 6, 6\}$.

2.6.8 Examples

Figure 2.9 – 2.12 present a sequence of piecewise constant, linear, quadratic and cubic B-splines with knots $\{0, 1, 1, 3, 4, 6, 6, 7\}$. The computation of these splines was carried out using the recursive formula and the additional routine for identifying repeated knots.

Note that the B-spline of order one over the repeated knots is defined as zero. This is illustrated in the second picture of Figure 2.9. Also, at the repeated knot location, the basis function is less smooth according to the above remark. This feature appears in the remaining figures.

2.6.9 Interpolation Errors via divided difference

Note that the error bound in approximating the function f can also be expressed in terms of divided differences.

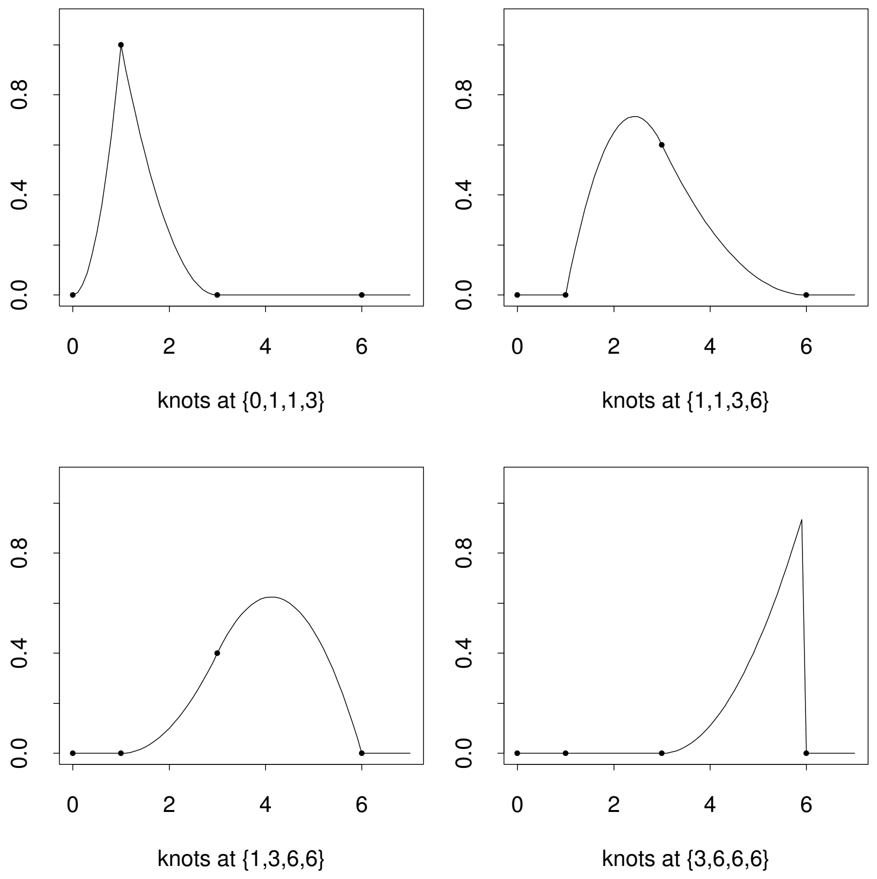

FIGURE 2.11. A sequence of quadratic B-splines having knots $\{0, 1, 1, 3, 6, 6, 6\}$.

If p interpolates f at the $n+1$ distinct nodes $x_0, x_1, \ldots, x_n$ in $[a, b]$, then for x not a node

$$f(x) - p(x) = f[x_0, x_1, \ldots, x_n, x] \prod_{i=0}^{n} (x - x_i).$$

In fact, let p and q denote the polynomials interpolating f at $x_0, \ldots, x_n$ and $x_0, \ldots, x_n, t$, respectively. Then by the Newton form:

$$q(x) = p(x) + [x_0, \ldots, x_n, t]f \prod_{i=0}^{n} (x - x_i).$$

Thus

$$f(t) = q(t) = p(t) + [x_0, \ldots, x_n, t]f \prod_{i=0}^{n} (t - x_i).$$

This property is known as the *osculatory interpolation.*

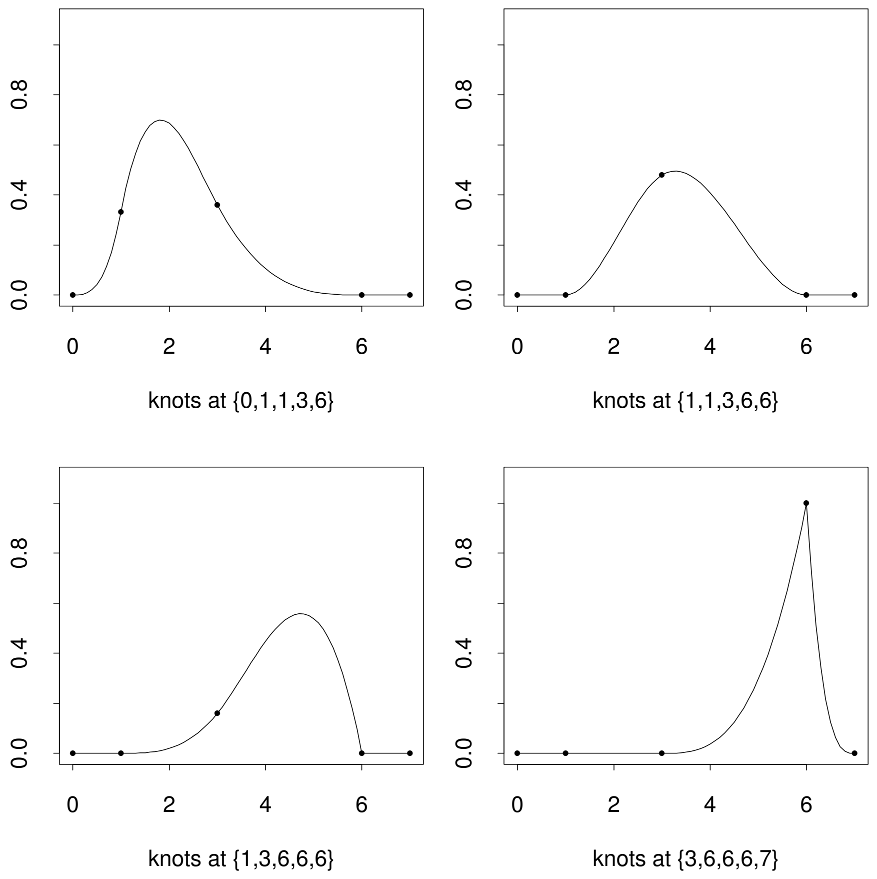

FIGURE 2.12. A sequence of cubic B-splines having knots $\{0, 1, 1, 3, 6, 6, 6, 7\}$.

2.7 Continuity of divided differences

When view as a function of the knots, the divided difference has the same smoothness as the function being interpolated. To see this, suppose f is n times continuously differentiable on $[a, b]$, and let $t_0, t_1, \ldots, t_n$ be points in $[a, b]$, distinct or not. Then

1. There exists $\xi \in [\min t_i, \max t_i]$ such that
$$[t_0, t_1, \ldots, t_n]f = f^{(n)}(\xi)/n!.$$

2. If, for each m, $t_{0,m}, t_{1,m}, \ldots, t_{n,m}$ are $n+1$ points, and $t_{i,m} \to t_i$ as $m \to \infty$, $i = 0, 1, \ldots, n$, then
$$[t_{0,m}, t_{1,m}, \ldots, t_{n,m}]f \to [t_0, t_1, \ldots, t_n]f.$$

These can be verified as follows. One first shows that the second result holds for $t_0 < t_n$. This follows easily from (2.6.4) and induction. Using this result, the first statement will be proven by considering either: (i) $t_0 = \cdots = t_n$, or (ii) $t_0 < t_n$. Case (i) is just the definition of the divided difference. To see case (ii), let $t_{0,m} < t_{1,m} < \cdots < t_{n,m}$ so that $t_{i,m} \to t_i$ as $m \to \infty$, $i = 0, 1, \ldots, n$. By (2.6.2), there exist $\xi_m \in [t_0, t_n]$ such that

$$[t_{0,m}, t_{1,m}, \ldots, t_{n,m}]f = f^{(n)}(\xi_m)/n!, \qquad m \geq 1.$$

From the result just proven, the continuity of $f^{(n)}$ and the compactness of $[t_0, t_m]$,

$$\begin{aligned}[t_0, t_1, \ldots, t_n]f = \lim_{m\to\infty}[t_{0,m}, t_{1,m}, \ldots, t_{n,m}]f &= \lim_{m\to\infty} f^{(n)}(\xi_m)/n! \\ &= f^{(n)}(\xi)/n!\end{aligned}$$

for some $\xi \in [\lim t_{0,m}, \lim t_{n,m}] = [t_0, t_n]$. This proves the first assertion. To complete the second, let $t_0 = \cdots = t_n$. Then

$$\begin{aligned}[t_0, t_1, \ldots, t_n]f = f^{(n)}(t_0)/n! &= \lim_{m\to\infty} f^{(n)}(\xi_m)/n! \\ &= \lim_{m\to\infty}[t_{0,m}, t_{1,m}, \ldots, t_{n,m}]f.\end{aligned}$$

As an important application of the above result, consider

$$g_n(t) = [t_0, t_1, \ldots, t_n, t]f$$

for some sufficiently often differentiable function f. By (2.6.4) and the continuity of the divided difference,

$$[t, t+h]g_n = [t_0, t_1, \ldots, t_n, t, t+h]f \to [t_0, t_1, \ldots, t_n, t, t]f.$$

Thus

$$\frac{\partial}{\partial t}[t_0, \ldots, t_n, t]f = g_n'(t) = \lim_{h\to 0}[t, t+h]g_n = [t_0, \ldots, t_n, t, t]f. \qquad (2.7.1)$$

This result is very important for the development of the free knot spline procedures to be discussed later in the monograph.

2.8 Partial derivatives of B-splines with respect to knot locations

Let m denote a nonnegative integer; let $y_1, \ldots, y_{m+1}$ be not necessarily distinct real numbers; let $\tau_1, \ldots, \tau_d$ be the distinct values of $y_1, \ldots, y_{m+1}$; and let $l_j = \sum_k \operatorname{ind}(y_k = \tau_j) \geq 1$ denote the *multiplicity* of τ_j among

$y_1, \dots y_{m+1}$. Then $l_1 + \cdots + l_d = m+1$ and hence $l_j \le m+2-d \le m+1$. In particular, if $y_1 < y_{m+1}$ or, equivalently, if $d \ge 2$, then $l_j \le m$. The number l_j is related to the number of continuity condition at the jth knot ν_j according to $l_j = k - \nu_j$. See Section 2.6.7.

Let f be a function such that $f^{(0)}, \dots, f^{(l_j-1)}$ (that is, f and its first $l_j - 1$ derivatives) are well defined at τ_j. If the numbers $y_1, \dots, y_{m+1}$ are in nondecreasing order and their distinct values $\tau_1, \dots, \tau_d$ are in strictly increasing order, we can write

$$[y_1, \dots, y_{m+1}] = \begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix}$$

and we can rewrite (2.6.3) or (2.6.4) as

$$\begin{aligned} &\begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix} f \\ &= \frac{\begin{bmatrix} l_1 - 1 & l_2 & \cdots & l_d \\ \tau_1 & \tau_2 & \cdots & \tau_d \end{bmatrix} f - \begin{bmatrix} l_1 & \cdots & l_{d-1} & l_d - 1 \\ \tau_1 & \cdots & \tau_{d-1} & \tau_d \end{bmatrix} f}{\tau_d - \tau_1} \qquad \text{if } d \ge 2. \end{aligned} \tag{2.8.1}$$

(In the numerator of the right side of (2.8.1) we ignore the first column of the first array if $l_1 = 1$ and the dth column of the second array if $l_d = 1$.)

The next result is taken from Theorem 2.55 of Schumaker (1981), which follows by repeatedly applying (2.7.1).

Lemma 2.8.1.

$$\frac{\partial}{\partial \tau_j} \begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix} f = l_j \begin{bmatrix} l_1 & \cdots & l_j + 1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_j & \cdots & \tau_d \end{bmatrix} f.$$

Let m now be a positive integer and suppose $y_1 < y_{m+1}$. We can rewrite the mth order (normalized) B-spline associated with the knots $y_1, \dots, y_{m+1}$ as [see (2.6.5)]

$$\begin{aligned} B_m(x) &= (y_{m+1} - y_1)[y_1, \dots, y_{m+1}](\cdot - x)_+^{m-1} \\ &= (\tau_d - \tau_1) \begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1}. \end{aligned} \tag{2.8.2}$$

In particular,

$$B_1(x) = \operatorname{ind}(\tau_2 \ge x) - \operatorname{ind}(\tau_1 \ge x) = \operatorname{ind}(\tau_1 \le x < \tau_2). \tag{2.8.3}$$

If $d = 2$, $l_1 = 1$ and $l_2 = 2$, then for $\tau_1 \le x < \tau_2$,

$$\begin{aligned} B_2(x) &= (\tau_2 - \tau_1)[\tau_1, \tau_2, \tau_2](\cdot - x)_+ \\ &= (\tau_2 - \tau_1)[\tau_1, \tau_2][\tau_2, \cdot](\cdot - x)_+ \\ &= [\tau_2, \tau_2](\cdot - x)_+ - [\tau_2, \tau_1](\cdot - x). \\ &= 1 - \frac{\tau_2 - x}{\tau_2 - \tau_1} \\ &= \frac{x - \tau_1}{\tau_2 - \tau_1}. \end{aligned}$$

[Note that $[\tau_2, \tau_2](\cdot - x)_+$ is undefined at $x = \tau_2$.] Similarly, for $\tau_1 \le x < \tau_2$, $d = 2$, $l_1 = 1$ and $l_2 = 3$,

$$\begin{aligned} B_3(x) &= (\tau_2 - \tau_1)[\tau_1, \tau_2, \tau_2, \tau_2](\cdot - x)_+^2 \\ &= (\tau_2 - \tau_1)[\tau_1, \tau_2][\tau_2, \tau_2, \cdot](\cdot - x)_+ \\ &= \Big(\frac{x - \tau_1}{\tau_2 - \tau_1}\Big)^2. \end{aligned}$$

By induction, if $d = 2$ and $l_1 = 1$, then $m = l_2$ and

$$B_m(x) = \Big(\frac{x - \tau_1}{\tau_2 - \tau_1}\Big)^{l_2 - 1} \operatorname{ind}(\tau_1 \le x < \tau_2). \tag{2.8.4}$$

Similarly, if $d = 2$ and $l_2 = 1$, then $m = l_1$ and

$$B_m(x) = \Big(\frac{\tau_2 - x}{\tau_2 - \tau_1}\Big)^{l_1 - 1} \operatorname{ind}(\tau_1 \le x < \tau_2). \tag{2.8.5}$$

The next result is taken from Theorem 4.27 of Schumaker (1981), which is an easy consequence of (2.8.1).

Lemma 2.8.2. *Fix $1 \le j \le d$ and suppose that $l_j \le m - 2$.*
(a) ($j = 1$)

$$\begin{aligned} \frac{\partial}{\partial \tau_1} B(x) = (l_1 - 1) &\begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1} \\ &- l_1 \begin{bmatrix} l_1 + 1 & \cdots & l_d - 1 \\ \tau_1 & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1}; \end{aligned}$$

(b) ($j = d$)

$$\begin{aligned} \frac{\partial}{\partial \tau_d} B(x) = l_d &\begin{bmatrix} l_1 - 1 & \cdots & l_d + 1 \\ \tau_1 & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1} \\ &- (l_d - 1) \begin{bmatrix} l_1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1}; \end{aligned}$$

(c) *if* $2 \leq j \leq d-1$, *then*

$$\frac{\partial}{\partial \tau_j} B(x) = l_j \begin{bmatrix} l_1 - 1 & \cdots & l_j + 1 & \cdots & l_d \\ \tau_1 & \cdots & \tau_j & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1}$$
$$- l_j \begin{bmatrix} l_1 & \cdots & l_j + 1 & \cdots & l_d - 1 \\ \tau_1 & \cdots & \tau_j & \cdots & \tau_d \end{bmatrix} (\cdot - x)_+^{m-1}.$$

The same formulas are valid when l_j *is* $m-1$ *or* m *for all* x *excluding* $x = \tau_j$.

For $1 \leq j_1, j_2 \leq d$ with $j_1 \neq j_2$ and either $l_{j_2} \geq 2$ or $d \geq 3$, let $B_{j_1 j_2}$ be the B-spline obtained from B by increasing the multiplicity l_{j_1} of τ_{j_1} and decreasing the multiplicity of l_{j_2} of τ_{j_2} each by 1. The next result follows from (2.8.1), (2.8.2), (2.8.4), (2.8.5), and Lemma 2.8.2.

Lemma 2.8.3. *Fix* $1 \leq j \leq d$ *and suppose that* $l_j \leq m-2$.
(a) $(j = 1)$

$$\frac{\partial}{\partial \tau_1} B(x) = (l_1 - 1) \frac{B(x)}{\tau_d - \tau_1} - l_1 \frac{B_{1d}(x)}{\tau_d' - \tau_1},$$

where the second term is replaced by zero when its denominator equals zero (*that is, when* $d = 2$ *and* $l_2 = 1$);
(b) $(j = d)$

$$\frac{\partial}{\partial \tau_d} B(x) = l_d \frac{B_{d1}(x)}{\tau_d - \tau_1'} - (l_d - 1) \frac{B(x)}{\tau_d - \tau_1},$$

where the first term is replaced by zero when its denominator equals zero (*that is, when* $d = 2$ *and* $l_1 = 1$);
(c) *if* $2 \leq j \leq d-1$, *then*

$$\frac{\partial}{\partial \tau_j} B(x) = l_j \Big(\frac{B_{j1}(x)}{\tau_d - \tau_1'} - \frac{B_{jd}(x)}{\tau_d' - \tau_1} \Big).$$

The same formulas for $\partial B(x)/\partial \tau_j$ *are valid when* l_j *is* $m-1$ *or* m *for* $x \neq \tau_j$. *Here* $\tau_1' = \tau_1$ *if* τ_1 *is a knot of the B-spline in the corresponding numerator, and* $\tau_1' = \tau_2$ *otherwise* (*that is, when* $l_1 = 1$). *Similarly,* $\tau_d' = \tau_d$ *if* τ_d *is a knot of the B-spline in the numerator, and* $\tau_d' = \tau_{d-1}$ *otherwise* (*that is, when* $l_d = 1$).

Notes

Section 2.1. By browsing through any book on numerical analysis, one can easily conclude that polynomials have played an essential role in approximation theory. See, for example, Conte and de Boor (1972), Kincaid and Cheney (1996), DeVore and Lorentz (1993) and Schumaker (1981).

It is not easy to trace the complete history of piecewise polynomials, but there is no doubt that they have been very useful in numerical analysis for a long time. See DeVore and Lorentz (1993) and Schumaker (1981) for more details.

Spline function was introduced by Schoenberg (1946). Our exposition to B-splines involving non-multiple knots is based on the recursive retionship described in Kincaid and Cheney (1996). According to Schumaker (1981), B-spline was discovered by at least three different authors.

Section 2.2. The main references are de Boor (1978), DeVore and Lorentz (1993) and Schumaker (1981). For more information on moduli of smoothness and approximation properties, see DeVore and Lorentz (1993).

Section 2.3. See Chapter 12 of Schumaker (1981) for more information on tensor-product splines.

Section 2.4. The main reference is Rocketfellow (1980)???

Section 2.5.

Section 2.6. We follow the exposition for B-spline with possibly multivariate types. described in ...

Section 2.7. de Boor ...

Section 2.8. Diff wrt knots.