Contents

Chapter 4

Interpolation and Approximation

4.1 The Interpolation Problem

4.1.1 Introduction

Polynomials are used as the basic means of approximation in nearly all areas of numerical analysis. We have previously encountered two types of interpolation

- Piecewise linear interpolation that is commonly used in tables, when the requirements of accuracy are modest. A more modern application is in Computer Graphics.
- Interpolation of the values of a function in n *equidistant* points by a function in $\mathcal{P}_n$. Recall that in Sec. 3.2.2, $\mathcal{P}_n$ was defined as the space of polynomials in one variable of degree *less than* n; n is the number of data required to specify a polynomial in $\mathcal{P}_n$; the dimension of the linear space $\mathcal{P}_n$ is n.[1]

  The formulas with equidistant points, in particular Stirling's interpolation formula, given in Sec. 3.2, is mainly used piecewise for small values of n; see Problem 2.

The first type of interpolation will be generalized in Sec. 4.4, where we shall study interpolation by **piecewise polynomials**. Of particular importance are **splines**, which are piecewise polynomials, where a few derivatives are required to be continuous at the joints of the pieces.

In the first two sections we shall go deeper into the following **polynomial interpolation problem** for *non-equidistant*, distinct points:

Find a polynomial $p \in \mathcal{P}_n$ such that

$$p(x_i) = f(x_i), \quad i = 1 : n, \quad x_i \neq x_j \text{ for } i \neq j. \tag{4.1.1}$$

Recall that, by Theorem 3.2.1, the interpolation polynomial $p(x)$ *is uniquely determined* for a given **grid**, $(x_1, x_2, \ldots, x_n)$. This theorem is general, although the rest

[1] Some authors use similar notations, e.g., $\mathbf{P}_n$ or Π_n, to denote the $n+1$-dimensional space of polynomials of degree *less than or equal to* n.

of Sec. 3.2 dealt with interpolation polynomials in the equidistant case only, and their application to numerical differentiation and integration.[2] Also note that the formulation and the solution of this problem are *independent of the ordering of the points* x_i.

A set of polynomials $\mathbf{p} = \{p_1(x), p_2(x), \ldots, p_n(x)\}$, such that any polynomial $p \in \mathcal{P}_n$ can be expressed as a linear combination

$$p(x) = c_1 p_1(x) + \ldots + c_m p_n(x),$$

is called a **basis** in $\mathcal{P}_n$. The column vector $c = (c_1, c_2, \ldots c_n)^T$ can be viewed as a *coordinate vector* of p in the space $\mathcal{P}_n$, with respect to this basis. The **power basis**, where $p_j(x) = x^{j-1}$, i.e.

$$p(x) = \sum_{j=1}^{n} c_j x^{j-1},$$

is the simplest basis, though not always the best.

The interpolation problem (4.1.1) leads to a linear system of equations

$$c_1 p_1(x_i) + c_2 p_2(x_i) + \ldots + c_m p_n(x_i) = f(x_i), \quad i = 1 : n. \tag{4.1.2}$$

If we introduce the matrix

$$M_{\mathbf{p}} = [p_j(x_i)]_{i,j=1}^{n}, \tag{4.1.3}$$

and the column vector $\tilde{f} = \big(f(x_1), f(x_2), \ldots, f(x_n)\big)^T$,[3] then the linear system becomes

$$M_{\mathbf{p}} c = \tilde{f}. \tag{4.1.4}$$

The proof of Theorem 3.2.1 was based on the fact that this matrix is non-singular in the case of the power basis; in this case $M_{\mathbf{p}} = V^T$, where V is the **Vandermonde matrix**[4]

$$V = [x_j^{i-1}]_{i,j=1}^{n} = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \cdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix}. \tag{4.1.5}$$

In any basis $\mathbf{q} = \{q_1(x), q_2(x), \ldots, q_n(x)\}$ for $\mathcal{P}_n$, the q_j must be linear combinations of the p_k, $k = 1 : n$. This can be expressed in vector-matrix form:

$$\big(q_1(x), q_2(x), \ldots, q_n(x)\big) = \big(p_1(x), p_2(x), \ldots, p_n(x)\big) A, \tag{4.1.6}$$

where A is a constant matrix. *A must be non-singular*; for, if A were singular then there would exist a non-trivial vector v such that $Av = 0$, hence

$$(q_1(x), q_2(x), \ldots, q_n(x))v = (p_1(x), p_2(x), \ldots, p_n(x))Av = 0 \quad \forall x,$$

[2] It is de facto so, although the polynomials are invisible in the derivations of formulas by operator methods.

[3] We try to make a distinction between f that is an element in some function space and $\tilde{f} \in \mathbf{R}^n$.

[4] Alexandre Théophile Vandermonde (1735–1796), member of the French Academy of Sciences, is regarded as the founder of the theory of determinants. What is now referred to as the "Vandermonde matrix" does not seem to appear in his writings!

and $(q_1(x), q_2(x), \ldots, q_n(x))$ would thus not be a basis.

Similarly set $M_{\mathbf{q}} = [q_j(x_i)]_{i,j=1}^n$. By putting $x = x_i$, $i = 1 : m$ into (4.1.6), we see that $M_{\mathbf{q}} = M_{\mathbf{p}}A$, and $M_{\mathbf{q}}$ is non-singular for every basis. If we set $p(x) = \sum d_j q_j(x)$, the system (4.1.2) becomes for this basis $M_{\mathbf{q}}d = \tilde{f}$. Then

$$M_{\mathbf{p}}c = \tilde{f} = M_{\mathbf{q}}d = M_{\mathbf{p}}Ad, \qquad c = M_{\mathbf{p}}^{-1}\tilde{f} = Ad. \tag{4.1.7}$$

the matrix A is thus like a coordinate transformation in Geometry; Gander [25] gives the matrix A for the transformation between various several common representations.

Example 4.1.1 (An application to numerical integration).

We shall find a formula for integrals of the form

$$I = \int_0^1 x^{-1/2} f(x)\, dx$$

that is exact for $f \in \mathcal{P}_m$ and uses the values $f(x_i)$, $i = 1 : m$. Such integrals need a special treatment, due to the integrable singularity at $x = 0$.

Set $\mu_j = \int_0^1 x^{-1/2} p_j(x)\, dx$, and introduce the row vector $\mu^A = (\mu_1, \mu_2, \ldots, \mu_n)$. Then

$$I \approx \int_0^1 x^{-1/2} p(x)\, dx = \sum_{j=1}^n c_j \mu_j = \mu^A c = \mu^A(p) M_{\mathbf{p}}^{-1} \tilde{f}, \tag{4.1.8}$$

where it is emphasized in the last expression that μ^A depends on the basis. In fact $\mu^A(q) = \mu^A(p)A$, and $M_{\mathbf{q}}^{-1} = A^{-1}M_{\mathbf{p}}^{-1}$; we see that the approximation to I is independent of the basis, as it should, in view of Theorem 3.2.1.

Another approach is the **method of undetermined coefficients**, i.e. to seek a formula

$$I \approx \sum_{i=1}^n b_i f(x_i) \equiv b^T \tilde{f},$$

that is exact when $f(x) = p_j(x)$, $j = 1 : m$; then it is exact for all $p \in \mathcal{P}_m$ These conditions lead to the linear system

$$M_{\mathbf{p}}^T b = \mu. \tag{4.1.9}$$

This may be called the *adjoint* or dual to the system $M_{\mathbf{p}}c = \tilde{f}$. Using the standard basis it reads

$$Vb = \mu.$$

From (4.1.9) we get $b^T = \mu^T M_{\mathbf{p}}^{-1}$, and the final result of this approach becomes

$$I \approx b^T \tilde{f} = \mu(p)^T M_{\mathbf{p}}^{-1} \tilde{f},$$

which is *the same as* (4.1.8), although interpolation was not mentioned in this approach. In view of Theorem (3.2.1) this is no surprise, since the same values of f are used, and both formulas are exact for all $f \in \mathcal{P}_m$.

For numerical applications (for the power basis) see Problem 2. Evidently these two approaches can be used for any linear functional of f.

4.1.2 Various Bases for $\mathcal{P}_n$

There are many ways of specifying polynomials. If the purpose is to compute derivatives or integrals of the interpolation polynomial, the power basis or the **the shifted power basis**, where

$$q_j(x) = (x-a)^{j-1},$$

are usually also convenient. If a shifted power basis is to be used for polynomial approximation on a certain interval, it is often best to choose a near the midpoint of the interval.

The power basis has a bad reputation, which is related the ill-conditioning of the corresponding Vandermonde matrices. Many bounds and asymptotic estimates for the condition number of the Vandermonde matrix

$$V = V(x_1, x_2, \ldots, x_n)$$

are known; see [26, Sec. 1.3], [31, Sec. 22.1]. For example, for equidistant points on $[-1.1]$, i.e. $x_i = -1 + 2(i-1)/(n-1)$, it holds that

$$\kappa_\infty(V) = \|V^{-1}\|_\infty \|V\|_\infty \sim \pi^{-1} e^{\pi/4} (3.1)^n.$$

Hence, for $n = 20$, $\kappa_\infty(V) \approx 1.05 \cdot 10^9$. Other point distributions are even worse, e.g., for the harmonic points $x_i = 1/i$, $i = 1:n$,

$$\kappa_\infty(V) > n^{n+1},$$

which is faster than exponential growth! For the **Chebyshev points** on $[-1, 1]$

$$x_i = \cos\Big(\frac{2i-1}{n}\frac{\pi}{2}\Big), \quad i = 1:n, \tag{4.1.10}$$

i.e. the zeros of $T_{n-1}(x)$, the Vandermonde matrix is better conditioned

$$\kappa_\infty(V) \sim 0.253^{3/4}(1+\sqrt{2})^n.$$

It should be stressed that the condition number of the Vandermonde matrix measures the sensitivity of the coefficients c_i in the polynomial $p(x) = \sum_{j=1}^n c_j x^{j-1}$ to perturbations in the given data f_i. It is possible, that even when these coefficients are inaccurately determined, the interpolation polynomial $p(x)$ does still reproduce the true interpolation polynomial well. For further discussion of these points, see Sec. 4.2.4 and Sec. 4.3.4.

Mathematically, the choice of basis (for a finite-dimensional space) makes no difference. Computationally, working with *rounded values of the coefficients*, the choice of basis can make a great difference. Consider a sequence of polynomials $q_1, q_2, q_3, \ldots$

$$\begin{aligned}
q_1(x) &= a_{11} \\
q_2(x) &= a_{12} + a_{22}x \\
q_3(x) &= a_{13} + a_{23}x + a_{33}x^2 \\
&\ldots \\
q_n(x) &= a_{1m} + a_{2m}x + a_{3m}x^2 + \ldots + a_{mm}x^{n-1} \\
&\ldots
\end{aligned}$$

where $a_{jj} \neq 0$ for all j, is defined to be a **triangle family** of polynomials, i.e. $q_j(x)$ is of $(j-1)$'th degree with a non-zero leading coefficient.[5].

Conversely, for any j, $p_j(x) = x^{j-1}$ can be expressed recursively and uniquely as linear combinations of $q_1(x), q_2(x), \ldots, q_j(x)$, so that we obtain a triangular scheme also for the inverse transformation. So *every triangle family* $\{q_1(x), q_2(x), \ldots, q_m(x)\}$ *is a basis for* $\mathcal{P}_m$.

What we has just seen, is indeed a proof of the well known fact that the inverse of a triangular matrix, (with no zeros in the main diagonal) is also triangular. Among interesting triangle families we can mention the families where $q_{j+1}(x)$ is defined by $(x-c)^j$, $T_j(x)$, and many other families of orthogonal polynomials.

Let $x_1, x_2, \ldots, x_n$ be n distinct points and consider the **Newton polynomials**

$$p_1(x) = 1, \quad p_j(x) = (x - x_1)(x - x_2)\ldots(x - x_{j-1}), \quad j = 2 : n,$$

They define a triangle family with unit leading coefficients, and hence form a basis. The representation

$$p(x) = c_1 p_1 + c_2 p_2(x) + c_3 p_3(x) + \cdots + c_n p_n(x) \tag{4.1.11}$$

is often very convenient. Since $p_j(x_k) = 0$, if $k < j$, the coefficients $c_1, c_2, \ldots, c_n$ satisfy the lower triangular system of equations $Lc = f$, where

$$L = \begin{pmatrix} 1 & & & & \\ 1 & (x_2 - x_1) & & & \\ 1 & (x_3 - x_1) & (x_3 - x_1)(x_3 - x_2) & & \\ \vdots & \vdots & \vdots & \ddots & \\ 1 & (x_n - x_1) & (x_n - x_1)(x_n - x_2) & \cdots & \prod_{j=1}^{n-1}(x_n - x_j) \end{pmatrix} \tag{4.1.12}$$

Hence the coefficients can be computed recursively, by forward substitution, with much less work than the linear system (4.1.4) would require for the power basis, by standard methods for linear algebra. In the next section we shall see how this basis leads to Newton's interpolation formula, which is one of the best interpolation formulas, with respect to flexibility, computational economy and numerical stability.

If a polynomial $p(x)$ is given in the form (4.1.11) then it can be evaluated using only n multiplications and $2n$ additions, for a given numerical value x, from the nested form

$$\begin{aligned} p(x) = (\cdots(c_m(x - x_{n-1}) + c_{n-1})(x - x_{n-2}) + \\ \cdots + c_3)(x - x_2) + c_2)(x - x_1) + c_1. \end{aligned}$$

(Notice that the case, where all the $x_i = 0$, gives Horner's rule, see Sec. 1.4.2. We have $p(x) = b_1$, where b_1 is computed using the recursion formula:

$$b_n := c_n, \quad b_{i-1} := b_i(x - x_{i-1}) + c_{i-1}, \quad i = n : -1 : 2. \tag{4.1.13}$$

We leave the proof to Problem 4.

[5]In the terminology used in the previous subsection, this triangular matrix equals A^T; this explains the notation for the elements

Other bases are sometimes more advantageous. A **cardinal basis** of $\mathcal{P}_n$ is generated by the polynomial

$$\Phi_n(x) = (x - x_1)(x - x_2) \cdots (x - x_n), \tag{4.1.14}$$

where x_i, $i = 1 : n$, are distinct. The basis used in Lagrange interpolation formula reads

$$\ell_j(x) = \frac{\Phi_n(x)}{(x - x_j)\Phi_n'(x_j)}, \quad j = 1 : n. \tag{4.1.15}$$

Here ℓ_j is a polynomial of degree $n - 1$. By L'Hospital's rule

$$\ell_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{if } i \neq j. \end{cases} \tag{4.1.16}$$

(This is the property that in a more general context characterizes a cardinal basis.) This basis directly displays the solution of the interpolation problem for n distinct points. **Lagrange's interpolation** formula[6] has been widely regarded as being of mainly theoretical interest. In Sec. 4.2.2 two modified forms of Lagrange interpolation formula will be given, which are more computationally attractive.

The matrix approach described in the previous subsections may sometimes be convenient; a Vandermonde matrix is easily generated, when you work in a matrix-oriented command language. If you deal with a modest number of polynomials of low degree, convenience can be given a larger weight than the optimal number of arithmetical operations and a minimized effect of rounding errors. In the latter respects, the matrix approach is inferior to Newton's interpolation formula and the other formulas and algorithms to be discussed later.

The main reason why we started with such non-optimal procedures, is that they are *easily generalizable to other interpolation problems*, e.g., interpolation in other function spaces than $\mathcal{P}_n$ (see Problem 5), or interpolation with other conditions on the function f in addition to function values (see later sections). For such non-standard interpolation problems—that do occur in practice—the matrix approach is helpful also for finding out under what conditions the problem has a unique solution.

4.1.3 Discrete Least Squares Approximation

Let $p_1(x), p_2(x), \ldots, p_n(x)$ be a basis for $\mathcal{P}_n$. A natural extension of the interpolation problem is to determine a polynomial

$$p(x) = \sum_{j=1}^{n} c_j p_j(x) \in \mathcal{P}_n,$$

that, in some sense, best fits to the data $(x_i, f(x_i))$, $i = 1 : m$, where $m > n$. Since the number of equations is larger than the number of parameters, the corresponding linear system $Mc = f$ is **overdetermined**. and can typically be satisfied only

[6] Lagrange published his interpolation formula in 1794.

approximately; see Example 1.2.5, where a straight line could not be made to pass through the five points.

In **discrete least squares approximation** on determines *the coefficient vector c that minimizes the sum of squared residuals*

$$S(c) = \sum_{i=1}^{m} (p(x_i) - f(x_i))^2. \tag{4.1.17}$$

This can in many applications be motivated by statistical arguments; see Theorem 4.5.19. It also leads to rather simple computations. The conditions for the minimization are

$$\frac{\partial S(c)}{\partial c_k} = 2 \sum_{i=1}^{m} p_k(x_i)\big(p(x_i) - f(x_i)\big) = 0, \quad k = 1 : n.$$

These conditions can be written in the form $M^T(Mc - f) = 0$, or

$$M^T M c = M^T f, \tag{4.1.18}$$

Here $M^T M$ is a symmetric $n \times n$ matrix and (4.1.18) is called the **normal equations**; see Sec. 1.6.5. It can be shown that the matrix $M^T M$ is non-singular, and that the system yields the minimum of $S(c)$, unless the columns of M are linearly dependent.

Overdetermination can be used to attain two different types of **smoothing**:

(a) to reduce the effect of random or other irregular errors in the values of the function;

(b) to give the polynomial a smoother behaviour between the grid points.

Note that interpolation is a special case ($n = m$) of this problem. In this case the normal equations are mathematically equivalent to the system $Mc = f$. Since the condition number of $M^T M$ is the square of the condition number of M, the matrix $M^T M$ often is very ill-conditioned even for moderate n. Therefore forming the normal equations cannot be recommended in general. A stable method for discrete least squares polynomial approximation is obtained by using a basis of orthogonal polynomials; see Sec. 4.5.5. Stable methods for more general least squares problems are treated in Volume II, Chapter 8.

4.1.4 The Runge Phenomenon

Equidistant interpolation can give rise to convergence difficulties when the number of interpolation points becomes large. This difficulty is often referred to as **Runge's phenomenon**[7], and we illustrate it in the following example. A more advanced discussion is given in Sec. 4.4.3, by means of complex analysis.

[7] Carl Runge (1856–1927) German mathematician, who held a chair in Applied Mathematics in Göttingen 1904–1925. Runge's example is from 1901.

Example 4.1.2. The graph of the function

$$f = \frac{1}{1+25x^2} = \frac{i}{2}\Big(\frac{1}{i+5x} + \frac{1}{i-5x}\Big),$$

where $i = \sqrt{-1}$, is the continuous curve shown in Figure 4.2.1, is approximated in two different ways by a polynomial of degree 10 in $[-1, 1]$.

The dashed curve has been determined by interpolation on the equidistant grid with $m = 11$ points

$$x_i = -1 + 2(i-1)/(m-1), \qquad i = 1 : m. \tag{4.1.19}$$

The graph of the polynomial so obtained has—unlike the graph of f—a disturbing course between the grid points. The agreement with f near the ends of the interval is especially bad, while near the center of the interval $[-\frac{1}{5}, \frac{1}{5}]$ the agreement is fairly good. Such behavior is typical of *equidistant interpolation of fairly high degree*, and can be explained theoretically.

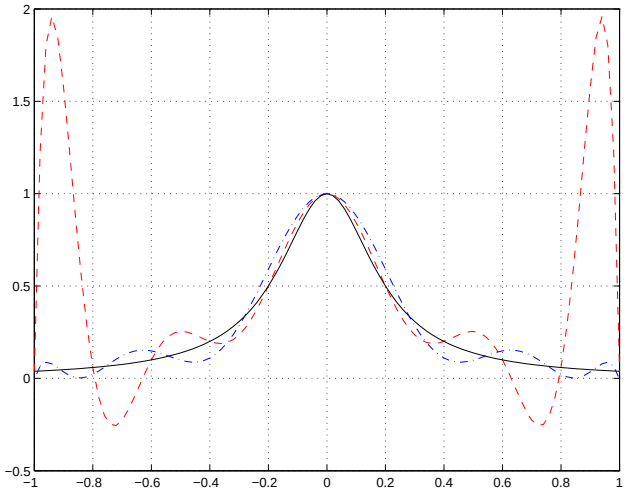

Figure 4.1.1. *Polynomial interpolation of* $1/(1+25x^2)$ *in two ways by the use of 11 points: equidistant points (dashed curve), Chebyshev abscissae (dash-dot curve).*

The *dotted* curve in Figure 4.1.1 has been determined by interpolation at the Chebyshev points

$$x_i = \cos\Big(\frac{2i-1}{m}\frac{\pi}{2}\Big), \quad i = 1 : m, \tag{4.1.20}$$

$(m = 11)$. This procedure is studied more closely in a later section. The agreement with f is now much better than with equidistant interpolation, but still not good. The function is not at all suited for approximation by one polynomial over the

entire interval. Here one would get a much better result using approximation with piecewise polynomials; see Sec. 4.4.

Notice that the difference between the values of the two polynomials is much smaller at the grid points of the equidistant grid than in certain points between the grid points, especially in the outer parts of the interval. This intimates that the values which one gets by equidistant interpolation with a polynomial of high degree can be very sensitive to disturbances in the given values of the function. Put another way, *equidistant interpolation using polynomials of high degree is in some cases an ill-conditioned problem, especially in the outer parts of the interval* $[x_1, x_m]$. The effect is even worse if one extrapolates—i.e. if one computes values of the polynomial outside the grid. However, equidistant interpolation works well near the center of the interval.

Even with equidistant data one can often get a more well-behaved curve by—instead of interpolating—fitting a polynomial of lower degree using the method of least squares. Generally, if one chooses $n < 2\sqrt{m}$, then the polynomial fit is quite well conditioned, but higher values of n should be avoided. In the above example, however, the agreement would still be quite bad, even at the grid points, when the degree is chosen to be so low.

If one intends to approximate a function *in the entire interval* $[-1, 1]$ by a *polynomial* and can choose the points at which the function is computed or measured, then one should choose the Chebyshev points. Using these points, interpolation is a fairly *well-conditioned* problem in the entire interval. The risk of disturbing surprises between the grid points is insignificant. One can also conveniently fit a polynomial of lower degree than $n - 1$, if one wishes to smooth errors in measurement; see Sec. 4.5.5.

Example 4.1.2 shows how important it is to study the course of the approximating curve $p^*(x)$ between the points which are used in the calculation before one accepts the approximation. When one uses procedures for approximation for which one does not have a complete theoretical analysis, one should make an *experimental perturbational calculation*. In the above case such a calculation would very probably reveal that the interpolation polynomial reacts quite strongly if the values of the function are disturbed by small amounts, say $\pm 10^{-3}$. This would give a basis for rejecting the unpleasant dashed curve in the example, even if one knew nothing more about the function than its values at the equidistant grid points.

Review Questions

1. The interpolation problem in $\mathcal{P}_n$ leads to a linear system $V^T c = \tilde{f}$, where V is a Vandermonde matrix. Write down the expression for the element v_{ij}.
2. What is meant by the method of undetermined coefficients? Give an example!
3. What is meant by a triangle family $q_1(x), q_2(x), \ldots, q_n(x)$ of polynomials? Are all such families a basis for $\mathcal{P}_n$?
4. What property characterizes a cardinal basis for $\mathcal{P}_n$?

Problems and Computer Exercises

1. (a) Study experimentally interpolation in $\mathcal{P}_n$, $n = 2 : 2 : 16$ for $f(x) = (3+x)^{-1}$, $x \in [-1, 1]$. Use the linear system $V^T c = \tilde{f}$ and the power basis. Study both equidistant points and Chebyshev points

$$x_i = -1 + 2\frac{i-1}{n-1}, \quad x_i = \cos\Big(\frac{2i-1}{n}\frac{\pi}{2}\Big), \quad i = 1 : n$$

respectively. Plot the error curve, $y = |f(x) - p(x)|$ in semi-logarithmic scale. For the larger values of m, make also experiments to illuminate the effects from random perturbations of the function values to the values of $p(x)$.

(b) Make also a few experiments with a random vector $\tilde{f}$, for $n = 16$ and $n = 8$, in order to compare the grid data and the order of magnitude of $p(x)$ between the grid points.

2. Prove the validity of (4.1.13).

3. *A warning for polynomial extrapolation of empirical functions, or . . . ?*

(a) Write a program $c = \mathit{polyapp}(x, y, n)$ that finds the coefficient vector c for a polynomial in $p \in \mathcal{P}_n$, in a shifted power basis, such that $y_i \approx p(x_i)$, $i = 1 : m$, $m \geq n$, in the least squares sense, or study a program that does almost this.[8]

(b) The following data shows the development of the Swedish GDP, quoted (with permission) from a table made by a group associated with the Swedish Employer's Confederation. (The data are expressed in prices of 1985 and scaled so that the value for 1950 is 100.)

1950	1955	1960	1965	1970	1975	1980	1985	1990
100.0	117.7	139.3	179.3	219.3	249.1	267.5	291.5	326.4
1952	1957	1962	1967	1972	1977	1982	1987	1992
104.5	124.6	153.5	189.2	226.4	247.7	270.2	307.6	316.6

(a) For the upper pairs of data, compute and plot $p(x), x \in [1950, 2000]$ (say). Mark the given data points. Do this for $m = 9$, and for (say) $n = 9$, and then for $n = 8 : -2 : 2$. Store the results, so that comparisons can be made afterwards.

Hint: If you use `polyfit` first subtract 1970 from the years.

(b) Do the same for the lower pairs of data. Organize the plots, so that interesting comparisons become convenient, e.g. how well were the data points of one of the sets interpolated by the results from the other set?

(c) Make forecasts for 1995 and 2000 with both data sets. Then, use a reduced data set, e.g., for the years 1982 and earlier (so that $m = 7$), and and compare the forecasts for 1987 and 1992 with the given data. (Isn't it a reasonable test for every suggested forecast model to study its ability to predict the present

[8] The Matlab command `polyfit` does *almost* this.

from the past?)

(d) See if you obtain better results with the logarithms of the GDP values.

4.2 Interpolation Formulas and Algorithms

4.2.1 Newton's Interpolation Formula

Let $x_1, x_2, \ldots, x_n$ be n distinct points and let p^* in $\mathcal{P}_n$ be the unique solution of the interpolation problem for $f(x)$ with the basis of (4.1.11). Suppose that

$$\begin{aligned} f(x) = c_1 + c_2(x - x_1) + \ldots + c_n(x - x_1)(x - x_2)\cdots(x - x_{n-1}) \\ + A_n(x)(x - x_1)(x - x_2)\cdots(x - x_n), \end{aligned} \tag{4.2.1}$$

If $f \in \mathcal{P}_n$, we know from Sec. 4.1.2 that such a formula holds with $A_n(x) \equiv 0$. We shall see that it is correct in general.

For $x = x_1$ we get $c_1 = f(x_1)$. Set

$$[x]f = f(x), \qquad [x_1, x]f = \frac{f(x) - f(x_1)}{x - x_1}.$$

Then

$$\begin{aligned} [x_1, x]f = c_2 + c_3(x - x_2) + \ldots + c_n(x - x_2)\cdots(x - x_{n-1}) \\ + A_n(x)(x - x_2)\cdots(x - x_n), \end{aligned}$$

and $c_2 = [x_1, x_2]f$. We now *define* recursively, for $k \geq 1$, **divided differences**[9]

$$[x_1, x_2, \ldots, x_{k-1}, x_k, x]f = \frac{[x_1, x_2, \ldots, x_{k-1}, x]f - [x_1, x_2, \ldots, x_{k-1}, x_k]f}{x - x_k}. \tag{4.2.2}$$

We obtain, for $k = 2$,

$$\begin{aligned} [x_1, x_2, x]f = c_3 + c_4(x - x_3) + \ldots + c_n(x - x_3)\cdots(x - x_{n-1}) \\ + A_n(x)(x - x_3)\cdots(x - x_n), \end{aligned}$$

and $c_3 = [x_1, x_2, x_3]f$. By induction it follows that

$$c_k = [x_1, x_2, \ldots, x_{k-1}, x_k]f, \qquad k = 1 : m. \tag{4.2.3}$$

For $k = n$ we obtain, $A_n(x) = [x_1, x_2, \ldots x_n, x]f$.

We now introduce a notation that is convenient in the following. Set $\Phi_0(x) = 1$, and for $k = 1 : n$,

$$\Phi_k(x) = \Phi_{k-1}(x)(x - x_k) = (x - x_1)(x - x_2)\cdots(x - x_k). \tag{4.2.4}$$

[9] We prefer the modern notation $[\ldots]f$ to the older notations $f[\ldots]$ or $f(\ldots)$, since it emphasizes that $[\ldots]$ is an operator. Note that the interpretation $[x]f = f(x)$ is consistent with this.

For $f \in \mathcal{P}_n$ we know that (4.2.1) is correct, hence we can trust the coefficients c_k. Moreover, since $p^*(x_j) = f(x_j)$, $j = 1 : n$, it follows that $[x_1, x_2, \dots x_j]p^* = [x_1, x_2, \dots x_j]f$. Hence

$$p(x) = \sum_{j=1}^{n} [x_1, \dots, x_j]p \, \Phi_{j-1}(x), \quad \forall p \in \mathcal{P}_n,$$
$$p^*(x) = \sum_{j=1}^{n} [x_1, \dots, x_j]f \, \Phi_{j-1}(x).$$

For a general function f we do not yet know that (4.2.1) is correct, but after inserting the only possible values of c_k and $A_n(x)$ in (4.2.1), we can conjecture that the following is an identity:

$$f(x) = \sum_{j=1}^{n} [x_1, \dots, x_j]f \, \Phi_{j-1}(x) + [x_1, x_2, \dots x_n, x]f \, \Phi_n(x).$$

We prove this by induction. For $n = 1$, it is true, because the right hand side becomes $f(x_1) + [x_1, x]f \cdot (x - x_1) = f(x) =$ the left hand side. Next suppose that it is true for $n = m$. The difference between the right hand side for $n = m + 1$ and $n = m$ reads

$$\begin{aligned}
&([x_1, \dots, x_{m+1}]f - [x_1, x_2, \dots x_m, x]f) \, \Phi_m(x) + [x_1, x_2, \dots x_{m+1}, x]f \, \Phi_{m+1}(x) \\
&= \big([x_1, \dots, x_{m+1}]f - [x_1, x_2, \dots x_m, x]f + [x_1, x_2, \dots x_{m+1}, x]f \ (x - x_{m+1})\big)\Phi_m(x) \\
&= \big([x_1, \dots, x_{m+1}, x]f \ (x_{m+1} - x) + [x_1, x_2, \dots x_{m+1}, x]f \ (x - x_{m+1})\big)\Phi_m(x) \\
&= 0.
\end{aligned}$$

Hence the conjecture is true for $n = m+1$. We summarize the results in a theorem.

Theorem 4.2.1. *Newton's Interpolation Formula with exact remainder.*

The interpolation problem of determining the polynomial $p \in \mathcal{P}_n$ such that $p(x_i) = f(x_i)$, $i = 1 : n$, where the x_i are distinct points, has the solution

$$p(x) = \sum_{j=1}^{n} [x_1, \dots, x_j]f \, \Phi_{j-1}(x). \tag{4.2.5}$$

where $\Phi_k(x)$ is defined by (4.2.4)*. The formula*

$$f(x) = \sum_{j=1}^{n} [x_1, \dots, x_j]f \, \Phi_{j-1}(x) + [x_1, x_2, \dots x_n, x]f \, \Phi_n(x) \tag{4.2.6}$$

is an identity, i.e. the exact remainder equals

$$f(x) - p(x) = [x_1, x_2, \dots x_n, x]f \, \Phi_n(x). \tag{4.2.7}$$

These formulas are valid also for complex x_i and x.

Note that to obtain the interpolation polynomial of the next higher degree with Newton's formula, we need only add a term similar to the last term, but involving a new divided difference of one higher order.

In particular, if $f \in \mathcal{P}_n$ then it follows from (4.2.7) that

$$[x_1, x_2, \ldots, x_n, x]f = 0, \quad \forall x.$$

For $x = x_{n+1}$, this equation is, by Theorem 3.2.1, the only non-trivial relation of the form $\sum_{j=1}^{n+1} a_j f(x_j) = 0$ that holds for all $f \in \mathcal{P}_n$,.

Theorem 4.2.2.

For every n, the divided difference $[x_1, x_2, \ldots, x_n]f$ is the coefficient of x^{n-1} in the interpolation polynomial $p^* \in \mathcal{P}_n$. A divided difference is a symmetric function of its arguments.

Proof. The first statement follows from (4.2.3). The second statement then holds, because the interpolation polynomial is uniquely determined and independent of how the points are ordered. □

Assume $k > i$. By the definition of divided differences,

$$[x_{i+1}, \ldots, x_{k-1}, x_k, x]f = \frac{[x_{i+1}, \ldots, x_{k-1}, x]f - [x_{i+1}, \ldots, x_{k-1}, x_k]f}{x - x_k}.$$

Now set $x = x_i$ and use the symmetry property (Theorem 4.2.2). We obtain the formula

$$[x_i, x_{i+1}, \ldots, x_{k-1}, x_k]f = \frac{[x_i, x_{i+1}, \ldots, x_{k-1}]f - [x_{i+1}, \ldots, x_{k-1}, x_k]f}{x_i - x_k}. \quad (4.2.8)$$

This formula can be used recursively to compute the divided differences. The computation is conveniently arranged in a table shown below for $n = 5$ (recall that $[x_i]f = f(x_i)$).

$$\begin{array}{lllll}
x_1 & [x_1]f & & & \\
 & & [x_1, x_2]f & & \\
x_2 & [x_2]f & & [x_1, x_2, x_3]f & \\
 & & [x_2, x_3]f & & [x_1, x_2, x_3, x_4]f \\
x_3 & [x_3]f & & [x_2, x_3, x_4]f & \\
 & & [x_3, x_4]f & & [x_2, x_3, x_4, x_5]f \\
x_4 & [x_4]f & & [x_3, x_4, x_5]f & \\
 & & [x_4, x_5]f & & \\
x_5 & [x_5]f & & &
\end{array} \qquad [x_1, x_2, x_3, x_4, x_5]f$$

This table is called a **divided-difference table**. Note that the points $x_1, x_2, x_3 \ldots$ need not be arranged in increasing order of magnitude. *Each entry in the table is computed from the two entries above and below in the previous column.* Hence the complete table can be constructed, e.g., column by column or diagonal by diagonal.

The divided differences which occur in Newton's interpolation formula (4.2.5) are those in the downward diagonal of the table. However, since the points $x_1, \ldots, x_n$ can be arbitrarily ordered, we can, for example, also introduce the points in Newton's interpolation formula in *backward* order $x_n, \ldots, x_1$. This gives the **backward form** for the interpolating polynomial

$$p^*(x) = f(x_n) + \sum_{j=1}^{n-1} [x_n, x_{n-1} \ldots, x_{n-j}] f \, (x - x_{n-j+1}) \cdots (x - x_n). \tag{4.2.9}$$

The divided differences in this formula lie on the *upward* diagonal starting at f_n in the table.

Theorem 4.2.3. *The Remainder Term for Interpolation*

Let f be a given real function, with $f^{(n)}(x)$ continuous in $\text{int}(x, x_1, x_2, \ldots, x_n)$. *Denote by p^* the polynomial of degree $n-1$ for which $p(x_i) = f(x_i)$, $i = 1 : n$. Then*

$$f(x) - p^*(x) = [x_1, x_2, \ldots x_n, x] f \, \Phi_n(x) = \frac{f^{(n)}(\xi_x)}{n!} \Phi_n(x), \tag{4.2.10}$$

$\Phi_n(x) = \prod_{i=1}^{n} (x - x_i)$, *for some point* $\xi_x \in \text{int}(x, x_1, x_2, \ldots, x_n)$, *and*

$$[x_1, x_2, \ldots x_n, x_{n+1}] f = \frac{f^{(n)}(\xi)}{n!}, \quad \xi \in \text{int}(x_1, \ldots, x_{n+1}). \tag{4.2.11}$$

Proof. Following a proof due to Cauchy, we introduce a new variable z, and set

$$G(z) = f(z) - p^*(z) - [x_1, x_2, \ldots x_n, x] f \, \Phi_n(z).$$

We know by Theorem 4.2.1 that

$$f(x) - p^*(x) = [x_1, x_2, \ldots x_n, x] f \, \Phi_n(x). \tag{4.2.12}$$

hence $G(x) = 0$. Then $G(z) = 0$ for $z = x, x_1, x_2 \ldots, x_n$. From repeated use of Rolle's theorem it follows that there exists a point $\xi_x \in \text{int}(x, x_1, x_2, \ldots, x_n)$, such that $G^{(n)}(\xi_x) = 0$. Since $p^{*(n)}(z) = 0$ and $\Phi_n^{(n)}(z) = n!$ for all z, $G^{(n)}(z) = f^{(n)}(z) - [x_1, x_2, \ldots x_n, x] f \, n!$. If we now put $z = \xi_x$, we obtain

$$[x_1, x_2, \ldots x_n, x] f = \frac{f^{(n)}(\xi_x)}{n!}. \tag{4.2.13}$$

Put this into the definition of $G(z)$, and set $z = x$. Since $G(x) = 0$, the first statement follows. The second statement follows from (4.2.13) for $x = x_{n+1}$. □

In this theorem x_i, x, $f(x)$, etc. must be *real*, while (4.2.12), i.e. Newton's interpolation formula with the exact remainder term, is valid also in the complex plane. Notice the similarity to the remainder term in Taylor's formula. We shall see that this can be considered as a limiting case when all the points x_i coincide. Notice also that the right hand side of (4.2.10) is zero at the grid points—as it should be.

We are now in a position to give a short proof of the important formula (3.3.6) that we now formulate as a theorem.

Theorem 4.2.4. *Assume that $f \in C^k$. Then*

$$\Delta^k f(x) = h^k f^{(k)}(\zeta), \qquad \zeta \in [x, x + kh]. \tag{4.2.14}$$

If $f \in \mathcal{P}_k$ then $\Delta^k f(x) = 0$. Analogous results hold, mutatis mutandis, for backward and central differences.

Proof. Combine the result in Theorem 4.2.5 with (4.2.11), after appropriate substitutions. □

To form the Newton interpolation polynomial we only need one diagonal of the divided-difference table, and it is not necessary to store the entire table.

Algorithm 4.2.1 *Computing the Newton Coefficients*

The following program replaces (overwrites) the function values $f_1, f_2, \ldots, f_n$, where $f_i = f(x_i)$, $i = 1 : n$. by the downward diagonal of divided differences

$$f_i = [x_1, x_2, \ldots, x_i]f, \quad i = 1 : n$$

of the divided difference table. At step j the jth column of the table is computed. Note that it is necessary to proceed from the bottom of each column to avoid overwriting data needed later! The algorithm uses $n(n-1)/2$ divisions and $n(n-1)$ subtractions.

```
for j = 1 : n − 1
  for i = n : −1 : j + 1
    f_i := (f_i − f_{i−1})/(x_i − x_{i−j});
  end
end
```

The Newton interpolation polynomial is then given by (4.2.9).

If it is not known in advance how many interpolation points that are needed to achieve the required accuracy one interpolation point can be added at a time:

Algorithm 4.2.2 *Divided Difference Table*

The following algorithm computes the difference table one diagonal at a time. In the ith step the entries $f_i, [x_{i-1}, x_i]f, \ldots, [x_1, x_2, \ldots, x_i]f$ on the upward diagonal of the divided-difference table overwrites the function values $f_i, f_{i-1}, \ldots, f_1$.

```
for i = 2 : n
  for j = i : −1 : 2
    f_j := (f_j − f_{j−1})/(x_i − x_{j−1});
  end
end
```

For the evaluation of the Newton polynomial at a point $x = z$, we use the simple Horner-like scheme (4.1.13)

$$p(x) = c_1 + \sum_{j=2}^{n} c_j \Phi_{j-1}(x), \quad \phi_{j-1}(x) = \prod_{i=1}^{j-1}(x - x_i).$$

We have $p(x) = b_1$, where b_1 is computed using the recursion formula:

$$b_n := c_n, \quad b_i := b_{i+1}(z - x_i) + c_i, \quad i = n-1 : -1 : 1. \tag{4.2.15}$$

It is straightforward to show that the computed result is the exact value corresponding to slightly perturbed divided differences; cf. Problem 2.3.6.

The auxiliary quantities $b_n, \ldots, b_2$ are of independent interest, since we have

$$p(x) = b_1 + (x - z)\left(b_2 + \sum_{j=2}^{n-1} b_{j+1}\phi_{j-1}(x)\right). \tag{4.2.16}$$

Repeated applications of the Horner scheme are useful in the evaluation of derivatives of a Newton polynomial.

Example 4.2.1.

Compute the interpolation polynomial for the following table:

$$\begin{array}{lllll} x_1 = 1 & \mathbf{0} & & & \\ & & \mathbf{2} & & \\ x_2 = 2 & 2 & & \mathbf{1} & \\ & & 5 & & \mathbf{0} \\ x_3 = 4 & 12 & & 1 & \\ & & 8 & & \\ x_4 = 5 & 20 & & & \end{array}$$

(The entries appearing in the Newton forward interpolation formula are boldface.) We get two alternative representations

$$\begin{aligned} p(x) &= 0 + 2(x-1) + 1(x-1)(x-2) + 0(x-1)(x-2)(x-4) \\ &= 20 + 8(x-5) + 1(x-5)(x-4) + 0(x-5)(x-4)(x-2) \\ &= x^2 - x, \end{aligned}$$

where the second is obtained from (4.2.9). (Note that for these particular data the unique interpolation polynomial in $\mathcal{P}_4$ actually belongs to the subspace $\mathcal{P}_3$.)

The remainder term in interpolation is according to Theorem 4.2.3 equal to

$$\prod_{i=1}^{n}(x - x_i) f^{(n)}(\xi_x)/n!.$$

Here ξ_x depends on x, but one can say that the error curve behaves for the most part like a polynomial curve $y = c\prod_{i=1}^{n}(x - x_i)$. A similar curve is also typical for error curves arising from least squares approximation. This contrasts sharply with the error curve for Taylor approximation, whose behavior is described approximatively by $y = c(x - x_0)^n$. It is natural to ask what the optimal placing of the interpolation points $x_1, \ldots, x_n$ should be in order to minimize the maximum magnitude of $\Phi_n(x) = \prod_{i=1}^{n}(x - x_i)$ in the interval the formula is to be used. For the interval $[-1, 1]$ the answer is given directly by the minimax property (Lemma 3.2.3) of the Chebyshev polynomials—choose $\Phi_n(x) = T_n(x)/2^{n-1}$. Thus the interpolation points should be taken as the zeros of $T_n(x)$. (In case of an interval $[a, b]$ one makes the linear substitution $x = \frac{1}{2}(a + b) + \frac{1}{2}(b - a)t$.) We have already seen examples of the use of Chebyshev interpolation in the discussion of the Runge phenomenon in Sec. 4.1.4.

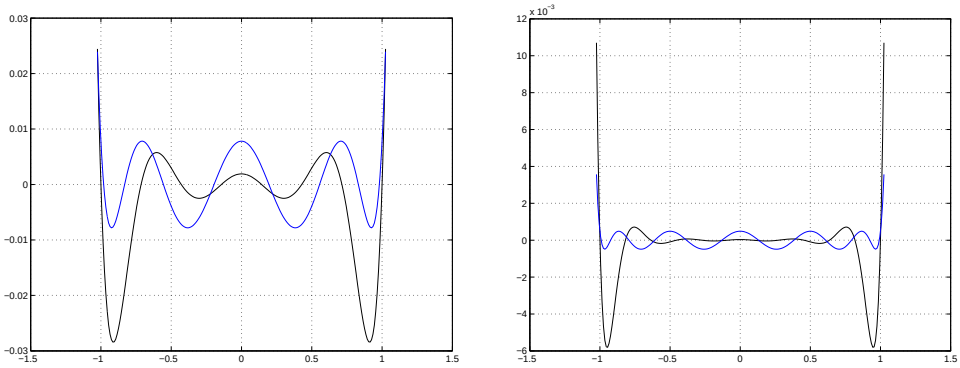

Figure 4.2.1. *Error of interpolation in $\mathcal{P}_n$ for $f(x) = x^n$, using equidistant points and Chebyshev points; $n = 8$ (left) $n = 12$ (right).*

Example 4.2.2.

Use the same notations as before. For $f(x) = x^n$ the interpolation error becomes $f(x) - p^*(x) = \Phi_n(x)$, because $f^{(n)}(x)/n! \equiv 1$. Figure 4.2.1 shows the interpolation error with n equidistant points on $[-1, 1]$ and with n Chebyshev points on the same interval, i.e.

$$x_i = -1 + 2\frac{i-1}{n-1}, \quad x_i = \cos\Big(\frac{2i-1}{n}\frac{\pi}{2}\Big),$$

respectively, for $n = 6$ and $n = 12$. The behaviour of the error curves are rather typical for functions where $f^{(n)}(x)$ is slowly varying. Also note that the error increases rapidly, when x leaves the interval $\text{int}(x_1, x_2, \ldots, x_n)$. In the equidistant case, the error is quite large also in the outer parts of the interval.

Example 4.2.3.

Set $f(x;z) = 1/(z-x)$; x is the variable, z is a parameter; both may be complex. The following elementary, though remarkable, expansion can be proved directly by induction (Problem 3a).

$$\begin{aligned}\frac{1}{z-x} &= \frac{1}{z-x_1} + \frac{x-x_1}{(z-x_1)(z-x_2)} + \ldots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(z-x_1)(z-x_2)\cdots(z-x_n)} \\ &\quad + \frac{(x-x_1)\cdots(x-x_n)}{(z-x_1)\cdots(z-x_n)(z-x)} \\ &= \sum_{j=1}^{n} \frac{\Phi_{j-1}(x)}{\Phi_j(z)} + \frac{\Phi_n(x)}{\Phi_n(z)(z-x)}. \end{aligned} \tag{4.2.17}$$

When we match this with Newton's interpolation formula we find that

$$[x_1, x_2, \ldots, x_j] f(x;z) = \frac{1}{\Phi_j(z)}, \qquad [x_1, x_2, \ldots, x_j, x] f(x;z) = \frac{1}{\Phi_j(z)(z-x)}. \tag{4.2.18}$$

These formulas can also be proved by induction, by working algebraically with $1/(z-x)$ in the divided difference table (Problem 3). See also Problem 3.2.2a for the equidistant case.

An interesting feature is that these formulas do *not* require that the points x_i are distinct. (They are consistent with the extension to non-distinct points that will be made in Sec. 4.3.) Everything is continuous except if $z = x_i$, $i = 1:n$, or, of course if $z = x$, see Sec. 4.3. If all the x_i coincide, we obtain a geometric series with a remainder.

This is more than a particular example. Since $1/(z-x)$ is the kernel of Cauchy's integral (and several other integral representations), this expansion is easily generalized to arbitrary analytic functions.

For given interpolation points the divided differences in Newton's interpolation formula depends on the ordering in which the points x_i are introduced. Mathematically all orderings give the same unique interpolation polynomial. However, *the condition number for the coefficients c in the Newton polynomial can depend strongly on the ordering of the interpolation points.* For simple everyday interpolation problems the increasing order $x_1 < x_2 < \cdots < x_n$ will give satisfactory results. If the point $\tilde{x}$ where the polynomial is to be evaluated is known, then an ordering such that

$$|\tilde{x} - x_1| \le |\tilde{x} - x_2| \le \cdots \le |\tilde{x} - x_n|$$

can be recommended. (In the equidistant case this corresponds to using Stirling's or Bessel's formula.)

Another suitable choice in case convergence is slow and an interpolation polynomial of high order is used, is the **Leja ordering** defined by

$$x_1 = \max_{1\le i\le n} |x_i|, \qquad \prod_{k=1}^{j-1} |x_j - x_k| = \max_{i\ge j} \prod_{k=0}^{j-1} |x_i - x_k|, \quad j = 2:n-1. \tag{4.2.19}$$

Note also that the barycentric Lagrange interpolation formula, to be introduced in Sec. 4.2.2, has very good stability properties.

Let $\mathcal{K}$ be a compact set in the complex plane with a connected complement. Any sequence of points $\xi_1, \xi_2, \ldots$ which satisfies the conditions

$$|\xi_1| = \max_{\xi\in\mathcal{K}} |\xi|, \qquad \prod_{k=1}^{j-1} |\xi_j - \xi_k| = \max_{\xi\in\mathcal{K}} \prod_{k=0}^{j-1} |\xi - \xi_k|, \quad j = 2, 3, \ldots. \tag{4.2.20}$$

are **Leja points** for $\mathcal{K}$. The points may not be uniquely defined by (4.2.20). For a real interval $[a, b]$ the Leja points are distributed similarly to the Chebyshev points. The main advantage of the Leja points is that it is easy to add new Leja points successively to an already computed sequence of Leja points.

Theorem 4.2.5. *For equidistant points $x_i = x_1 + (i-1)h$, $f_i = f(x_i)$, it holds that*

$$[x_i, x_{i+1}, \ldots, x_{i+k}]f = \frac{\Delta^k f_i}{h^k k!}. \tag{4.2.21}$$

Proof. By induction, with the use of equation (4.2.8). The details are left to the reader. □

We have noted above that, in the notation for the equidistant case, $\nabla^k f_n \approx h^k f^{(k)}$, while in the divided difference notation $f[x_n, x_{n-1}, \ldots, x_{n-k}] \approx f^{(k)}/k!$. For the basis functions of the interpolation formulas, we have, respectively,

$$\binom{x}{k} = O(1), \quad (x - x_n)(x - x_{n-1}) \cdots (x - x_{n-k+1}) = O(h^k),$$

provided that $x - x_{n-j} = O(h)$, $j = 0 : k - 1$.

For many applications the quantities used in the equidistant case have a more appropriate order of magnitude. In some applications to differential equations, there may even be a risk for overflow or underflow, when divided differences are used. F. Krogh [35] introduced a scaling for the divided differences with the same advantage; in the equidistant case these scaled divided differences are identical to $\nabla^k f_n$. The main application so far has been to multistep methods for ordinary differential equations.

4.2.2 Lagrange's Interpolation Formula

A basis that is often advantageous to use is the cardinal basis of $\mathcal{P}_n$ generated by the polynomial

$$\Phi_n(x) = (x - x_1)(x - x_2) \cdots (x - x_n), \tag{4.2.22}$$

where x_i, $i = 1 : n$ are n distinct real numbers. The basis reads,

$$\ell_j(x) = \frac{\Phi_n(x)}{(x - x_j)\Phi_n'(x_j)} = \prod_{\substack{i=1 \\ i\neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)}, \quad j = 1 : n \tag{4.2.23}$$

Here ℓ_j, the **Lagrange polynomials** of degree $n-1$, satisfy

$$\ell_j(x_i) = \delta_{ij} = \begin{cases} 1 & \text{if } i = j; \\ 0 & \text{if } i \neq j. \end{cases}$$

Theorem 4.2.6 (Lagrange's interpolation formula).
The unique interpolation polynomial $p \in \mathcal{P}_n$ interpolating the function $f(x)$ at the distinct points x_i, $i = 1 : n$, can be written

$$p(x) = \sum_{j=1}^{n} f(x_j)\ell_j(x), \tag{4.2.24}$$

where

$$\ell_j(x) = \prod_{\substack{i=1 \\ i\neq j}}^{n} \frac{(x - x_i)}{(x_j - x_i)}, \qquad j = 1 : n, \tag{4.2.25}$$

Quite often it is asserted that the Lagrange form is a bad choice for practical computations[10], since for each new value of x the functions $\ell_i(x)$ have to be recomputed at a cost $\mathcal{O}(n^2)$. Further, adding a new data point x_{n+1}, f_{n+1} will require a new computation from scratch. It is concluded that the expression (4.2.24) is not as efficient as the Newton formula.

The above assertions are, however, not well-founded. *The Lagrange representation can easily be rewritten in two more attractive forms, which both are eminently suitable for computation*; see Berrut and Trefethen [3]. Taking out the common factor $\Phi_n(x)$ in (4.2.24) and introducing the **support coefficients**

$$w_j = 1 \Big/ \prod_{\substack{i=1 \\ i\neq j}}^{m} (x_j - x_i), \quad j = 1 : n, \tag{4.2.26}$$

Lagrange interpolation formula can be written in the **modified form**

$$p(x) = \Phi_n(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j} f(x_j), \tag{4.2.27}$$

Here w_j depend only on the given points x_j, $j = 1 : n$, and can be computed in $n(n-1)$ operations. This is twice the work required to compute the divided differences for Newton's interpolation formula. Then, to evaluate $p(x)$ from (4.2.27) for a new value of x we only need to compute $\Phi_n(x)$ and $w_j/(x - x_j)$, $j = 1 : n$, which requires $\mathcal{O}(n)$ operations.

[10] Steffensen [51, p. 25] "For actual numerical interpolation Lagrange's formula is, however, as a rule not very suitable.

The product factor $\Phi_n(x)$ in (4.2.27) can be eliminated as follows. Since the interpolation formula is exact for $f(x) \equiv 1$, we have

$$1 = \Phi_n(x) \sum_{j=1}^{n} \frac{w_j}{x - x_j}.$$

Substituting this in (4.2.27)

$$p(x) = \frac{\displaystyle\sum_{j=1}^{n} \frac{w_j}{x - x_j} f(x_j)}{\displaystyle\sum_{j=1}^{n} \frac{w_j}{x - x_j}}, \quad \text{if} \quad x \neq x_j, \quad j = 1 : n, \tag{4.2.28}$$

which is the **barycentric form** of Lagrange's interpolation formula. This expresses the value $p(x)$ as a weighted mean of the values f_i. (Note that the coefficients $w_j/(x-x_j)$ need not be positive, so the term "barycentric" is not quite appropriate.)

The barycentric formula (4.2.28) has a beautiful symmetric form and is "eminently suitable for machine computation" (Henrici [29, p. 237]) Unlike Newton's interpolation formula, it does not depend on the order in which the nodes are ordered. The numerical stability of the two modified Lagrange interpolation formulas is, contrary to what is often stated, very good. Note that interpolation property of $p(x)$ is preserved even if the coefficients w_i are perturbed, but then $p(x)$ is usually no longer a polynomial but a rational function.

There seems to be a stability problem for the formula (4.2.28) when x is very close to one of the interpolation points x_i. In this case $w_i/(x - x_i)$ will be very large and not accurately computed because of the cancellation in the denominator. However, this is in fact no problem, since there will be exactly the same error in the denominator. Further, in case $\Delta_i = \mathrm{fl}\,(x - x_i)$ is exactly zero, we simply put $\Delta_i = u$ (the unit roundoff).

The Lagrange representation of the interpolation formula can be as efficiently updated. as as Newton's formula. Suppose the support coefficients $w_i^{(k-1)}$, $i = 1 : k-1$ for the points $x_1, \ldots, x_{k-1}$ are known. Adding the point x_k the first $k-1$ new support coefficients can be calculated from

$$w_i^{(k)} = w_i^{(k-1)}/(x_i - x_k), \quad i = 1 : k-1,$$

using $(k-1)$ divisions and subtractions. Finally we have $w_k^{(k)} = 1/\prod_{i=1}^{k-1}(x_k - x_i)$. The computation of the support coefficients is summarized in the following program:

```
w_1 = 1;
for k = 2 : n
   w_k = 1;
   for i = 1 : k - 1
      w_i := w_i/(x_i - x_k);
      w_k = w_k/(x_k - x_i);
   end
end
```

Note that the support coefficients w_i do not depend on the function to be interpolated. Once they are known interpolating a new function f only requires $\mathcal{O}(n)$ operations. This contrasts with Newton's interpolation formula, which requires the calculation of a new table of divided differences for each new function.

Suppose that we use interpolation points in an interval $[a, b]$ of length $2C$. Then as $n \to \infty$ the scale of the weights will grow or decay exponentially at the rate C^{-n}. If n is large or C is far from 1, the result may underflow or overflow even in IEEE double precision. In such cases there may be a need to rescale the support coefficients.

The computation of the support coefficients can be done in only $\frac{1}{2}n(n-1)$ by using the relation (see Problem 5 and [46, Sec. 3.2.1])

$$\sum_{i=1}^{n} w_i = 0, \quad n > 1;$$

to compute $w_n = \sum_{i=1}^{n-1} w_i$. However, using this identity destroys the symmetry and can lead to stability problems for large n. Serious cancellation in the sum will occur whenever $\max_i |w_i|$ is much larger than $|w_n|$. Hence the use of this identity is not recommended in general.

For various important distributions of interpolating points, it is possible to compute the support coefficients w_i analytically.

Example 4.2.4.

For interpolation at the equidistant points x_1, $x_i = x_1 + (i-1)h$, $i = 2 : n$, the support coefficients are

$$\begin{aligned} w_i &= 1 \big/ \left((x_i - x_1) \cdots (x_i - x_{i-1})(x_i - x_{i+1}) \cdots (x_i - x_n)\right) \\ &= (-1)^{n-i} \big/ \left(h^{n-1}(i-1)!\,(n-i)!\right) = \frac{(-1)^{n-i}}{h^{n-1}(n-1)!} \binom{n-1}{i} \end{aligned}$$

In the barycentric formula (4.2.28) a common factor in the coefficients w_i cancels and we may use instead the modified support coefficients

$$w_i^* = (-1)^{i+1} \binom{n-1}{i}. \tag{4.2.29}$$

For a given n these can be evaluated in only $2n$ operations using the recursion

$$w_1^* = n - 1, \qquad w_i^* = w_{i-1}^* \frac{n-i}{i}, \quad i = 2 : n.$$

Example 4.2.5.

Explicit support coefficients are also known for the Chebyshev points of the first and second kind on $[-1, 1]$. For the Chebyshev points

$$x_i = \cos \frac{(2i-1)}{n} \frac{\pi}{2}, \quad i = 1 : n,$$

they are

$$w_i = (-1)^i \sin \frac{(2i-1)}{n} \frac{\pi}{2}. \tag{4.2.30}$$

For the Chebyshev points of the second kind,

$$x_i = \cos \frac{(i-1)}{(n-1)}\pi, \quad i = 1:n$$

they are

$$w_i = (-1)^i \delta_j, \quad \delta_j = \begin{cases} 1/2 & \text{if } i = 1 \text{ or } i = n, \\ 1, & \text{otherwise} \end{cases}. \tag{4.2.31}$$

Note that all but two weights are equal! This will be considered from another point of view in Sec. 4.6.

For an interval $[a, b]$ the Chebyshev points can be generated by a linear transformation. The corresponding weights w_i then gets multiplied by $2^n(b-a)^n$. However, this factor cancels out in the barycentric formula, and there is no need to include it. Indeed, by *not* doing the risk of overflow or underflow, when $|b-a|$ is far from 1 and n is large, is avoided.

The two examples above show that with equidistant or Chebyshev points only $\mathcal{O}(n)$ operations are needed to get the weights w_i. For these cases the barycentric formula seems superior to all other interpolation formulas.

Lagrange interpolation formula can be used to compute the inverse of the Vandermonde matrix V in (4.1.5) in $\mathcal{O}(n^2)$ operations. If we set $V^{-1} = W = (w_{ij})_{i,j=1}^n$, then $WV = I$, the ith row of which can be written

$$\sum_{j=1}^{n} w_{ij} x_k^j = \delta_{ik}, \quad k = 1:n.$$

This is an interpolation problem that is solved by the Lagrange basis polynomial

$$\ell_i(x) = \prod_{\substack{k=1 \\ k \neq i}}^{n} \frac{(x - x_k)}{(x_i - x_k)} = \sum_{j=1}^{n} w_{ij} x^j, \quad j = 1:n. \tag{4.2.32}$$

This shows that V is nonsingular if and only if the points x_i are distinct.

The elements w_{ij} can be computed as follows. First we compute the coefficients of the polynomial

$$\Phi_n(x) = (x - x_1)(x - x_2) \cdots (x - x_n) = \sum_{j=1}^{n+1} a_j x^{j-1}.$$

This can be done by the recursion:

$$a_1 = -x_1; \quad a_2 = 1;$$
$$\textbf{for } k = 2:n$$

```
    a_{k+1} = 1;
    for i = k : -1 : 2
      a_i = a_{i-1} - x_k a_i;
    end
    a_1 = -x_k a_1;
  end
```

Next the coefficients of polynomials

$$q_i(x) = \Phi_n(x)/(x - x_i), \quad i = 1 : n.$$

are computed by synthetic division. Finally, the Lagrange polynomials are obtained from $\ell_i(x) = q_i(x)/q_i(x_i)$, where the scalars $q_i(x_i)$ are computed by Horner's rule. The cost of computing the n^2 elements in W by this algorithm is only $6n^2$ operations.

4.2.3 Iterative Linear Interpolation

There are other recursive algorithms for interpolation. Of interest are those based on *successive linear interpolations*. The basic formula is given in the following theorem.

Theorem 4.2.7.

Assume that the two polynomials $p_{n-1}(x)$ and $q_{n-1}(x)$, both in $\mathcal{P}_{n-1}$ interpolate $f(x)$ at the points $x_1, \ldots, x_{n-1}$, and $x_2, \ldots, x_n$, respectively. If the n points $x_1, x_2, \ldots, x_{n-1}, x_n$ are distinct then

$$p_n(x) = \frac{(x - x_1)q_{n-1}(x) - (x - x_n)p_{n-1}(x)}{x_n - x_1}.$$

is the unique polynomial in $\mathcal{P}_n$ that interpolates $f(x)$ at the m points $x_1, x_2, \ldots, x_{n-1}, x_n$.

Proof. Since $q_{n-1}(x)$ and $p_{n-1}(x)$ both interpolate $f(x)$ at the points $x_2, \ldots, x_{n-1}$ and

$$\frac{(x - x_1) - (x - x_n)}{x_n - x_1} = 1,$$

it follows that also $p_n(x)$ interpolates $f(x)$ at these points. Further, $p_n(x_1) = p_{n-1}(x_1)$ and hence interpolates $f(x)$ at x_1. A similar argument shows that, $p_n(x)$ interpolates $f(x)$ at $x = x_n$. Hence $p_n(x)$ is the unique polynomial interpolating $f(x)$ at the distinct points $x_1, x_2, \ldots, x_n$. □

Neville's and **Aitken's** algorithms both use Theorem 4.2.7 repeatedly to construct successively higher order interpolation polynomials. Let $p_{i,k}$ denote the polynomial interpolating at the k points $x_{i-k+1}, \ldots, x_i$. In Neville's interpolation algorithm one puts

$$p_{i,1} = f(x_i), \quad i = 1 : n$$

and compute for $i = 2 : n$

$$p_{i,k+1} = \frac{(x - x_{i-k})p_{i,k} - (x - x_i)p_{i-1,k}}{x_i - x_{i-k}}, \qquad k = 1 : i - 1, \tag{4.2.33}$$

where $p_{i,k+1}$ interpolates at the points $x_{i-k}, \ldots, x_{i-1}, x_i$. The calculations can be arranged in a table, which for $n = 4$ has the form

$$\begin{array}{lllll} x_1 & f(x_1) = p_{1,1} & & & \\ & & p_{2,2} & & \\ x_2 & f(x_2) = p_{2,1} & & p_{3,3} & \\ & & p_{3,2} & & p_{4,4} \\ x_3 & f(x_3) = p_{3,1} & & p_{4,3} & \\ & & p_{4,2} & & \\ x_4 & f(x_4) = p_{4,1} & & & \end{array}$$

Here any entry is obtained as a linear combination of the nearest two entries in the preceding column. Note that it is easy to add a new interpolation point in this scheme. To proceed only the last lower diagonal needs to be retained. If it is known in advance that a fixed number k of points are to be used, then one can instead generate the table column by column. When one column has been evaluated then the preceding may be discarded.

These formulas are better than Newton's only in the case that $f(x)$ is to be evaluated for the same values of $x - x_i$ for several functions (sequences) f. In this case one should compute

$$t_{ik} = \frac{x - x_{i-k}}{x_i - x_{i-k}}$$

once and for all.

We saw in Sec. 3.3.5 its application to the extrapolation to $x = 0$ of a polynomial given at a few positive arguments, a typical example, where it is efficient and widely used.

Aitken's scheme is similar to Neville's, but uses another sequence of interpolants. Let $p_{i,k}$, $i \geq k$, denote the polynomial interpolating at the k points $x_1, \ldots, x_{k-1}$ and x_i. Set $p_{i,1} = f(x_i)$, as above, and compute for $i = 2 : n$

$$p_{i,k+1} = \frac{(x - x_{i-1})p_{i,k} - (x - x_i)p_{k,k}}{x_k - x_{i-1}}, \qquad k = 1 : i - 1, \tag{4.2.34}$$

The table can again be generated column by column. To be able to add a new point the whole upper diagonal $p_{i,i}$, $i = 1 : k$, must be saved.

4.2.4 Conditioning of the Interpolation Problem

Consider the problem of finding a polynomial $p_f = p_n(x) \in \mathcal{P}_n$ that interpolates given values f_j at distinct points x_j, $j = 1 : n$. With the terminology of Sec. 2.4.3 the input data are f_j, $j = 1 : n$, and the output data is the value of the polynomial p_f evaluated at some fixed point $\tilde{x}$.

Definition 4.2.8.

The condition number of p_f at fixed $\tilde{x}$ and fixed interpolation points but varying data f_j, $j = 1 : n$ is

$$cond\,(\tilde{x}, f) = \lim_{\epsilon \to 0} \sup \left\{ \frac{|p_{f+\Delta f}(\tilde{x}) - p_f(\tilde{x})|}{\epsilon |p_f(\tilde{x})|} : |\Delta f| \le \epsilon |f| \right\} \tag{4.2.35}$$

Lemma 4.2.9. N. J. Higham [32]

Let $\ell_j(x)$ be the Lagrange basis functions. Then the condition number in Definition 4.2.8 *is, for $p_f(\tilde{x}) \neq 0$,*

$$cond\,(\tilde{x}, f) = \frac{\sum_{j=1}^{n} |\ell_j(\tilde{x}) f_j|}{|p_f(\tilde{x})|} \ge 1, \tag{4.2.36}$$

and for any Δf with $|\Delta f| \le \epsilon |\Delta f|$ we have

$$\frac{|p_{f+\Delta f}(\tilde{x}) - p_f(\tilde{x})|}{\epsilon |p_f(\tilde{x})|} \le cond\,(\tilde{x}, f)\epsilon.$$

Proof. Using the Lagrange basis,

$$p_{f+\Delta f}(\tilde{x}) - p_f(\tilde{x}) = \sum_{j=1}^{n} \ell_j(\tilde{x}) \Delta f_j,$$

It follows immediately that the expression in (4.2.36) is an upper bound for the condition number and it is clearly at least 1. Equality is attained for $\Delta f_j = \epsilon\, sign\,(\ell_j(\tilde{x}))|f_j|$. The inequality follows trivially. □

Assume that the interpolation points x_j, $j = 1 : n$ lie in $[-1, 1]$. Consider $cond\,(x, 1)$, the condition number of interpolating the function $f(x) = 1$ at these points. By Lemma 4.2.9 we have

$$cond\,(x, 1) = \sum_{j=1}^{n} |\ell_j(x)|.$$

This quantity is related to the so called **Lebesgue constant** defined by

$$\Lambda_n = \sup_{x \in [-1,1]} \sum_{j=1}^{n} |\ell_j(x)| \ge cond\,(x, 1). \tag{4.2.37}$$

For equally spaced point, however, Λ_n grows at a rate proportional to $2^n/(n \log n)$; see Cheney and Light [12, Chap. 3].

Theorem 4.2.10 (N. Higham [32]).

Assume that x_i, f_i and x are floating point numbers. Then the computed value $\bar{p}(x)$ of the interpolation polynomial using the modified Lagrange formula (4.2.27) *satisfies*

$$\bar{p}(x) = \Phi_n(x) \sum_{i=1}^{n} \frac{w_i}{x - x_i} f(x_i) \prod_{j=1}^{5(n+1)} (1 + \delta_{ij})^{\pm 1}, \tag{4.2.38}$$

where $|\delta_{ij}| \leq u$.

Thus the formula (4.2.27) *computes the exact value of an interpolating polynomial corresponding to slightly perturbed function values $f(x_i)$. Hence this formula is* backward stable *in the sense of Definition* 2.4.9.

From this theorem and Lemma 4.2.9 the forward error bound

$$\frac{|p_n(\tilde{x}) - \bar{p}_n(\tilde{x})|}{|p_n(\tilde{x})|} \leq \gamma_{5n+5} cond\,(\tilde{x}, f). \tag{4.2.39}$$

can be obtained.

The barycentric formula is not backward stable. A forward error bound similar to (4.2.39) but containing an extra term proportional to $cond\,(\tilde{x}, 1) = \sum_{j=1}^{n} |\ell_j(x)|$ can be shown. Hence the barycentric formula can be significantly less accurate than the modified Lagrange formula (4.2.27) only for a poor choice of interpolation points.

4.2.5 Interpolation by Rational Functions

Rational approximation is often superior to polynomial approximation in the neighborhood of a point at which the function has a singularity. The rational interpolation problem is to determine a rational function

$$f_{m,n}(z) = \frac{P_m(z)}{Q_n(z)} \equiv \frac{\sum_{j=0}^{m} p_j z^j}{\sum_{j=0}^{n} q_j z^j}, \tag{4.2.40}$$

so that

$$f_{m,n}(x_i) = f_i, \quad i = 0 : m + n. \tag{4.2.41}$$

A *necessary condition* for (4.2.41) to hold clearly is that

$$P_m(x_i) - f_i Q_n(x_i) = 0, \quad i = 0 : m + n. \tag{4.2.42}$$

or for $i = 0 : m + n$,

$$p_0 x_i + p_1 x_i + \cdots + p_m x_i^m - f_i(q_0 x_i + q_1 x_i + \cdots + q_n x_i^n) = 0, \tag{4.2.43}$$

This is a homogeneous linear system of $(m + n + 1)$ equations for the $(m + n + 2)$ coefficients in $P_{m,n}$ and $Q_{m,n}$. Such a system always has a nontrivial solution. The coefficients are determined only up to a common factor $\rho \neq 0$.

In contrast to polynomial interpolation a solution to the rational interpolation problem may not exist as shown in the following example:

Example 4.2.6.

Assume that we want to interpolate the first four of the points

x	0	1	2	3	4
y	2	3/2	4/5	1/2	6/17

by a rational function

$$f_{2,1} = \frac{p_0 + p_1 x + p_2 x^2}{q_0 + q_1 x}.$$

Then we must solve the homogeneous linear system

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \end{pmatrix} - \begin{pmatrix} 2 & 0 \\ 3/2 & 3/2 \\ 4/5 & 8/5 \\ 1/2 & 3/2 \end{pmatrix} \begin{pmatrix} q_0 \\ q_1 \end{pmatrix} = 0.$$

Setting $p_2 = 1$ we find the solution $p_0 = 8$, $p_1 = -6$, $q_0 = 4$, $q_1 = -2$. The corresponding rational function

$$f_{2,1} = \frac{8 - 6x + x^2}{4 - 2x} = \frac{(4-x)(2-x)}{2(2-x)}$$

has the common factor $(2 - x)$ and is *reducible* to $f_{2,1} = (4 - x)/2$. The original form is indeterminate $0/0$ at $x = 2$ while the reduced form does not take on the prescribed value at $x = 2$.

An algorithm similar to Newton's algorithm can be used for finding rational interpolants in continued fraction form. Set $v_0(x) = f(x)$, and use a sequence of substitutions

$$v_k(x) = v_k(x_k) + \frac{x - x_k}{v_{k+1}(x)}, \quad k = 0, 1, 2, \ldots. \tag{4.2.44}$$

The first two substitutions give

$$f(x) = v_0(x) = v_0(x_1) + \frac{x - x_0}{v_1(x)} = v_0(x_0) + \cfrac{x - x_0}{v_1(x_1) + \cfrac{x - x_1}{v_2(x)}}.$$

In general this gives a continued fraction

$$f(x) = a_0 + \cfrac{x - x_0}{a_1 + \cfrac{x - x_1}{a_2 + \cfrac{x - x_2}{a_3 +}}} \ldots = a_0 + \frac{x - x_1}{a_1 +} \, \frac{x - x_2}{a_2 +} \, \frac{x - x_3}{a_3 +} \ldots, \tag{4.2.45}$$

where $a_k = v_k(x_k)$, and we have used the compact notation introduced in Sec. 3.5.1. This becomes an identity if the expansion is terminated by replacing a_n in the last denominator by $a_n + (x - x_n)/v_{n+1}(x)$. If we set $x = x_k$, $k \le n$, then the fraction terminates before the the residual $(x - x_n)/v_{n+1}(x)$ is introduced. This means that

setting $1/v_{k+1} = 0$ will give a rational function which agrees with $f(x)$ at the points x_i, $i = 0 : k \le n$, assuming that the constants $a_0, \ldots, a_k$ exist. These continued fractions give a sequence of rational approximations $f_{k,k}$, $f_{k+1,k}$, $k = 0, 1, 2, \ldots$.

Introducing the notation

$$v_k(x) = [x_0, x_1, \ldots, x_{k-1}, x]\phi \tag{4.2.46}$$

we have $a_k = [x_0, x_1, \ldots, x_{k-1}, x_k]\phi$. Then by (4.2.44) we have

$$[x]\phi = f(x), \qquad [x_0, x]\phi = \frac{x - x_0}{[x]\phi - [x_0]\phi} = \frac{x - x_0}{f(x) - f(x_0)},$$
$$[x_0, x_1, x]\phi = \frac{x - x_1}{[x_0, x]\phi - [x_0, x_1]\phi},$$

and in general

$$[x_0, x_1, \ldots, x_{k-1}, x]\phi = \frac{x - x_{k-1}}{[x_0, \ldots, x_{k-2}, x]\phi - [x_0, \ldots, x_{k-2}, x_{k-1}]\phi}. \tag{4.2.47}$$

Therefore we also have

$$a_k = \frac{x - x_{k-1}}{[x_0, \ldots, x_{k-2}, x_k]\phi - [x_0, \ldots, x_{k-2}, x_{k-1}]\phi}. \tag{4.2.48}$$

We call the quantity defined by (4.2.48) the kth **inverse divided difference** of $f(x)$. Note that certain inverse differences can become infinite if the denominator vanishes. They are, in general, *symmetrical only in their last two arguments*[11]

The inverse divided differences of a function $f(x)$ can conveniently be computed recursively and arrange in a table similar to the divided difference table.

$$\begin{array}{llllll}
x_1 & f(x_1) & [x_1]\phi & & & \\
 & & & [x_1, x_2]\phi & & \\
x_2 & f(x_2) & [x_2]\phi & & [x_1, x_2, x_3]\phi & \\
 & & & [x_2, x_3]\phi & & [x_1, x_2, x_3, x_4]\phi \\
x_3 & f(x_3) & [x_3]\phi & & [x_2, x_3, x_4]\phi & \\
 & & & [x_3, x_4]\phi & & \\
x_4 & f(x_4) & [x_4]\phi & & &
\end{array}$$

Here the upper diagonal elements are the desired coefficients in the expansion (4.2.45).

Example 4.2.7.

Assume that we want to interpolate the points given in Example 4.2.7. Form-

[11] The **reciprocal differences** of Thiele [55] are symmetric functions of all their arguments; see Hildebrand [33, pp. 406ff].

ing the inverse differences we get the table

x_i	f_i	ϕ_1	ϕ_2	ϕ_3	ϕ_4
0	2				
		-2			
1	3/2		3		
		5/3		0	
2	4/5		∞		-5
		-2		$-1/5$	
3	1/2		-7		
		$-17/7$			
4	6/17				

This gives a sequence of rational approximations. If we terminate the expansion

$$f_{2,2} = 2 + \frac{x}{-2+}\,\frac{x-1}{3+}\,\frac{x-2}{0+}\,\frac{x-3}{-5}.$$

after a_3 we recover the solution of the previous example. Note that the the degeneracy of the approximation is shown by the entry $a_3 = 0$. Adding the last fraction gives the (degenerate) approximation

$$f_{2,2} = \frac{2+x}{1+x^2}.$$

It is verified directly that this rational function interpolates all the given points.

The formulas using inverse or reciprocal differences are are useful if one wants to determine the coefficients of the rational approximation, and use it for to compute approximations for several arguments. If one is only wants the value of the rational interpolating function for a single argument, then it is more convenient to use an alternative algorithm of Neville-type. If we consider the sequence of rational approximations of degrees (m, n)

$$(0,0),\quad (0,1),\quad (1,1),\quad (1,2),\quad (2,2),$$

the following recursive algorithm results (Stoer and Bulirsch [52, Sec. 2.2]):
For $i = 0, 1, 2, \ldots$, set $T_{i,-1} = 0$, $T_{i,0} = f_i$, and

$$T_{ik} = T_{i,k-1} + \frac{T_{i,k-1} - T_{i-1,k-1}}{\dfrac{x - x_{i-k}}{x - x_i}\left[1 - \dfrac{T_{i,k-1} - T_{i-1,k-1}}{T_{i,k-1} - T_{i-1,k-2}}\right] - 1}, \quad 1 \le k \le i. \tag{4.2.49}$$

As in Neville interpolation the calculations can be arranged in a table of the form

$$
\begin{array}{cccccc}
(m,n) = & (0,0) & (0,1) & (1,1) & (1,2) & \cdots \\
\hline
 & f_1 = T_{1,0} & & & & \\
0 & & T_{2,1} & & & \\
 & f_2 = T_{2,0} & & T_{3,2} & & \\
0 & & T_{3,1} & \longrightarrow & T_{4,3} & \\
 & f_3 = T_{3,0} & & T_{4,2} & \vdots & \ddots \\
0 & & T_{4,1} & \vdots & & \\
 & f_4 = T_{4,0} & \vdots & & & \\
\vdots & \vdots & & & &
\end{array}
$$

Here any entry is determined by a rhombus rule from three entries in the preceding two columns. Note that it is easy to add a new interpolation point in this scheme.

Review Questions

1. Prove the theorem which says that the interpolation problem for polynomials has a unique solution.
2. When is linear interpolation sufficient?
3. Derive Newton's interpolation formula.
4. Derive Newton's interpolation formula for the equidistant case, starting from Newton's general interpolation formula. How is this formula easily remembered?
5. Discuss how various sources of error influence the choice of step length in numerical differentiation.
6. Derive the Lagrange interpolation formula. Show how it can be rewritten in barycentric form. When is the latter form more efficient to use?

Problems and Computer Exercises

1. (a) Compute $f(3)$ by quadratic interpolation in the following table:

x	1	2	4	5
$f(x)$	0	2	12	21

   Use the points $1, 2$, and 4, and the points $2, 4$, and 5, and compare the results.
   (b) Compute $f(3)$ by cubic interpolation.

2. Compute $f(0)$ using one of the interpolation formulas treated above on the following table:

x	0.1	0.2	0.4	0.8
$f(x)$	0.64987	0.62055	0.56074	0.43609

The interpolation formula is here used for *extrapolation*. Use also Richardson extrapolation and compare the results.

3. *Error in linear interpolation*

(a) Suppose we want to compute by linear interpolation the value $y(x)$ at a point $x = x_0 + \theta h$, $h = x_1 - x_0$. Using (4.2.10) show that for $0 \le \theta \le 1$ the remainder $R(x) = f(x) - p(x)$ satisfies

$$|R(x)| \le \frac{h^2}{8} M_2, \tag{4.2.50}$$

(b) Show that if the values f_0 and f_1 are given to t correct decimal digits then the round-off error R_T in linear interpolation $p(x) = (1-\theta)f_0 + \theta f_1$, for $0 \le \theta \le 1$ satisfies $|R_T| =\le \frac{1}{2}10^{-t}$. Show further that if $h^2 M_2 \le 4 \cdot 10^{-t}$, then the total error in $p(x)$ is bounded by 10^{-t} twice the round-off error in the given values of f.

(c) Motivate the rule of thumb that linear interpolation suffices if $|\Delta^2 f_n|/8$ is a tolerable truncation error.

4. Work out the details of Example 4.2.3 (about divided differences etc. for $1/(z-x)$).

5. (a) Consider the two polynomials $p(x)$ and $q(x)$, both in $\mathcal{P}_n$, which interpolate $f(x)$ at the points $x_1, \ldots, x_n$, and $x_2, \ldots, x_{n+1}$, respectively. Assume that $\{x_i\}_{i=1}^{n+1}$ is an increasing sequence, and that $f^{(n)}(x)$ has constant sign in the interval $[x_1, x_{n+1}]$. Show that $f(x)$ is contained between $p(x)$ and $q(x)$ for all $x \in [x_1, x_{n+1}]$.

(b) Suppose that $f(x) = f_1(x) - f_2(x)$, where both $f_1^{(n)}(x)$ and $f_2^{(n)}(x)$ have the same constant sign in $[x_1, x_{n+1}]$. Formulate and prove a kind of generalization of the result in (a).

6. Using the barycentric formula (4.2.27) the interpolation polynomial can be written

$$p(x) = \sum_{i=1}^{n} w_i f(x_i) \prod_{\substack{j=1 \\ j \ne i}}^{m} (x - x_j).$$

Show by taking $f(x) \equiv 1$ and equating the coefficients for x^{n-1} on both sides that the support coefficients satisfy $\sum_{i=1}^{n} w_i = 0$.

7. Show that, if the points x_i are distinct,

$$[x_1, x_2, \ldots, x_m] f = \sum_{i=0}^{m} \frac{f(x_i)}{\Phi_m'(x_i)},$$

where $\Phi_m(x)$ is defined in (4.2.26).

Hint: Compare the coefficients of x^{n-1} in Newton's and Lagrange's expressions for the interpolation polynomial.

8. (a) Check Table 4.2.1 and the conclusions about the optimal step length in the text, Investigate how the attainable accuracy varies with u.

(b) Study the analogous question for $f''(x_0)$ using the formula

$$f''(x_0) \approx \left(1 - \frac{\delta^2}{12} + \frac{\delta^4}{90} - \frac{\delta^6}{560} + \frac{\delta^8}{3,150} - \dots\right) \frac{\delta^2 f_0}{h^2}.$$

9. Prove the validity of Algorithm 4.2.3

10. Given a backwards (upwards) diagonal in the table of divided differences (scaled or unscaled), $\langle X; m, i\rangle Y$, $i = 1 : k$. Find a recurrence formula for the computation of the next diagonal of the difference scheme for the interpolation polynomial $P(x; m, k)Y$, i.e. if $\langle X; m+1, k\rangle Y = \langle X; m, k\rangle Y$ (why?), find $\langle X; m+1, i\rangle Y$, $i = k-1, k-2, \dots, 0$.

Hint: Look up the equidistant case in Example 3.2.6.

11. (Bulirsch and Rutishauser (1968))

(a) The function $\cot x$ has a singularity at $x = 0$. Use values of $cotx$ for $x = 1°$, $2°$, …, $5°$. and rational interpolation of order (2,2) to determine an approximate value of $\cot x$ for $x = 2.5°$, and its error.

(b) Use polynomial interpolation for the same problem. Compare the result with that in (a).

4.3 Generalizations and Applications

4.3.1 Interpolation using Values of Derivatives

The general **Hermite interpolation** problem is the following: Given n distinct points $\{x_i\}_{i=1}^n$, and numbers $r_i \geq 1$, Find a polynomial $p(x)$ of degree $m-1$, where $\sum_{i=1}^n r_i = m$, so that $p(x)$ and its first $r_i - 1$ derivatives agree with those of $f(x)$ at x_i, i.e.

$$p(x)^{(j)}(x_i) = f(x)^{(j)}(x_i), \quad j = 0 : r_i - 1, \quad \sum_{i=1}^{n} r_i = m, \tag{4.3.1}$$

$i = 1 : n$. (We use here the notation $f^{(0)}(x)$ for $f(x)$.)

Hermite interpolation can be viewed as the result of passages to the limit in interpolation at m points, where for $i = 1 : n$ r_i interpolation points coalesce into the point x_i. We say that the point x_i has **multiplicity** r_i. For example, the Taylor polynomial in $\mathcal{P}_m$

$$p(x) = \sum_{j=0}^{m-1} \frac{f^{(j)}(x_1)}{j!}(x - x_0)^j \tag{4.3.2}$$

interpolates $f(x)$ at the point x_1 with multiplicity m (or x_1 is repeated m times).

Note that (4.3.1) are precisely m conditions on $p(x)$, so we can expect that the Hermite interpolation problem is uniquely solvable.

Theorem 4.3.1.
The problem of finding a polynomial $p \in \mathcal{P}_m$ that satisfies the Hermite interpolation conditions

$$p^{(j)}(x_i) = f^{(j)}(x_i), \quad i = 1:n, \quad j = 0:r_i - 1, \tag{4.3.3}$$

where $r_i \geq 1$, $\sum r_i = m$, has a unique solution.

Proof. The conditions are expressed by a system of m linear equations for the coefficients of $\hat{p}$, with respect to some basis. This has a unique solution for any right hand side, unless the corresponding homogeneous problem has a non-trivial solution. Suppose that a polynomial $p \in \mathcal{P}_m$ comes from such a solution of the homogeneous problem, i.e.

$$p^{(j)}(x_i) = 0, \quad i = 1:n, \quad j = 0:r_i - 1.$$

Then, x_i must be a zero of multiplicity r_i of $p(x)$, hence $p(x)$ must have at least $\sum r_i = m$ zeros (counting the multiplicities). But this is impossible, because the degree of p is less than m. This contradiction proves the theorem. □

Since Hermite interpolation is a boundary case of ordinary interpolation the remainder term for interpolation given in Theorem 4.2.3 applies. Hence, assuming that f is a real function, with continuous derivatives of order at least m, the error in Hermitian interpolation is given by

$$f(x) - p(x) = \frac{f^{(m)}(\xi_x)}{m!}\Phi_n(x), \quad \Phi_n(x) = \prod_{i=1}^{n}(x - x_i)^{r_i}. \tag{4.3.4}$$

for some point $\xi_x \in \text{int}(x, x_1, x_2, \ldots, x_n)$.

Example 4.3.1.
Consider the problem of finding a polynomial $p(x) \in \mathcal{P}_4$ that interpolates the function f and its first derivative f' at the two points x_0 and x_1, and also its second derivative at x_0. In the notations of Sec. 4.1.1 the linear system for the coefficient vector c becomes $V^T c = \tilde{f}$, where $\tilde{f} = (f(x_1), f'(x_1), f''(x_1), f(x_2), f'(x_2))^T$, and

$$V = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ x_1 & 1 & 0 & x_2 & 1 \\ x_1^2 & 2x_1 & 2 & x_2^2 & 2x_2 \\ x_1^3 & 3x_1^2 & 6x_1 & x_2^3 & 3x_2^2 \\ x_1^4 & 4x_1^3 & 12x_1^2 & x_2^4 & 4x_2^3 \end{pmatrix} \tag{4.3.5}$$

is a **confluent Vandermonde matrix**. Note that the second, third, and fifth column of V is obtained by "differentiating" the previous column. From Theorem 4.3.1 we conclude that such confluent Vandermonde matrices are nonsingular.

There are explicit formulas, analogous to Lagrange's formula, for Hermite interpolation; see [52, Sec. 2.1.5]. The Hermite interpolation polynomial is written in the form

$$p(x) = \sum_{i=1}^{n} \sum_{k=0}^{r_i-1} f^{(k)}(x_i) L_{ik}(x), \tag{4.3.6}$$

where $L_{ik}(x)$ are **generalized Lagrange polynomials**. These can be defined starting from the auxiliary polynomials

$$l_{ik}(x) = \frac{(x-x_i)^k}{k!} \prod_{\substack{j=1 \\ j\neq i}}^{n} \left(\frac{x-x_j}{x_i-x_j} \right)^{r_j}, \quad i = 1:n, \quad k = 0: r_j - 1.$$

Next, put

$$L_{i,r_i-1} = l_{i,r_i-1}, \quad i = 1:n,$$

and form recursively,

$$L_{ik}(x) = l_{ik}(x) - \sum_{\nu=k+1}^{r_i-1} l_{ik}^{(\nu)}(x_i) L_{i,\nu}(x), \quad k = r_i - 2 : -1 : 0.$$

It can be showed by induction that

$$L_{ik}^{(\sigma)}(x_i) = \begin{cases} 1, & \text{if } i = j \text{ and } k = \sigma; \\ 0, & \text{otherwise.} \end{cases}$$

Hence the L_{ik} are indeed the appropriate polynomials.

Example 4.3.2.

An important special case is when $r_i = 2$, $i = 1 : n$. Then the Hermite interpolating polynomial is the **osculating polynomial**, which agrees with $f(x)$ and $f'(x)$ at $x = x_i$, $i = 1 : n$. In this case we can write

$$p(x) = \sum_{i=1}^{n} (f(x_i) L_{i0}(x) + f'(x_i) L_{i1}(x)).$$

Here $L_{ik}(x)$ can be written in the form

$$L_{i1}(x) = (x - x_i) l_i(x)^2, \qquad L_{i0}(x) = (1 - 2l_i'(x_i)(x - x_i)) l_i(x)^2,$$

where $l_i(x)$, $i = 1 : n$, are the elementary Lagrange polynomials.

An interpolation problem that contains points of multiplicity greater than one can be obtained from the case with distinct points by a passage to the limit. Newton's interpolation formula is suitable for handling this case. Since by Theorem 4.2.2 a divided difference is a symmetric function of its arguments these can be permuted before taking the limit. We can therefore assume, without loss of generality, that

equal arguments are placed together, and that the values x_i for different groups of arguments are different.

Assume that $f^{(r)}(x)$ is continuous. Then by Theorem 4.2.3,

$$[x_1, x_2, \dots x_{r+1}]f = f^{(r)}(\xi)/r!, \quad \xi \in \text{int}(x_1, x_2, \dots, x_{r+1}).$$

If we let $x_i \to x$, $i = 1 : r+1$, then $[x_1, x_2, \dots x_{r+1}]f \to f^{(r)}(x)/r!$. Hence the natural definition of a divided difference with r equal arguments reads

$$[x, x, \dots, x]f = f^{(r)}(x)/r!, \quad r+1 \text{ equal arguments.} \tag{4.3.7}$$

One may also establish the following more general formula

$$[x_1, \dots, x_k, x \dots, x]f = \frac{1}{r!}\frac{d^r}{dx^r}[x_1, \dots, x_k, x]f \quad r+1 \text{ equal arguments.} \tag{4.3.8}$$

We now give a representation of the divided differences which allows several multiplicities.

Theorem 4.3.2.

Assume that $f^{(n-1)}(x)$ is continuous in $[a, b]$, $x_1, \dots, x_n \in [a, b]$ and x is distinct from any x_i. Then

$$[x, x_1, x_2, \dots, x_n]f = \frac{[x, x_2, \dots, x_n]f - [x_1, x_2, \dots, x_n]f}{x - x_1}, \tag{4.3.9}$$

gives the unique continuous extension of divided differences no matter what multiplicities occur in $x_1, \dots, x_n$.

Proof.

□

This definition and the usual recurrence formula for the divided differences are, under the above assumptions, sufficient for the construction of a table of divided differences in the case of multiple points, e.g.,

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

It can be shown that if $f \in C^k$, the divided differences belong to $C^{k+1-\max r_i}$, and that the interpolation polynomial has this kind of differentiability with respect to the x_i, nota bene if the "groups" do not coalesce further.

Example 4.3.3. Consider the interpolation problem in Example 4.3.1. For this we construct the generalized divided-difference table, where $x_1 \neq x_0$.

$$\begin{array}{llllll}
x_0 & f_0 & & & & \\
 & & f_0' & & & \\
x_0 & f_0 & & \frac{1}{2}f_0'' & & \\
 & & f_0' & & [x_0,x_0,x_0,x_1]f & \\
x_0 & f_0 & & [x_0,x_0,x_1]f & & [x_0,x_0,x_0,x_1,x_1]f \\
 & & [x_0,x_1]f & & [x_0,x_0,x_1,x_1]f & \\
x_1 & f_1 & & [x_0,x_1,x_1]f & & \\
 & & f_1' & & & \\
x_1 & f_1 & & & &
\end{array}$$

The interpolating polynomial now reads

$$\begin{aligned}
p(x) &= f_0 + (x-x_0)f_0' + (x-x_0)^2\frac{1}{2}f_0'' + (x-x_0)^3[x_0,x_0,x_0,x_1]f \\
&\quad + (x-x_0)^3(x-x_1)[x_0,x_0,x_0,x_1,x_1]f.
\end{aligned}$$

$$\begin{aligned}
f(x) - p(x) &= [x_0,x_0,x_0,x_1,x_1,x]f(x-x_0)^3(x-x_1)^2 \\
&= f^{(5)}(\xi_x)(x-x_0)^3(x-x_1)^2/5!
\end{aligned}$$

For the simplest Hermite interpolation problem, i.e. **cubic interpolation**, the given data are $f_i = f(x_i)$, $f_i' = f'(x_i)$, $i = 0, 1$. We can write the interpolation polynomial

$$\begin{aligned}
p(x) &= f_0 + (x-x_0)[x_0,x_1]f + (x-x_0)(x-x_1)[x_0,x_0,x_1]f \\
&\quad + (x-x_0)^2(x-x_1)[x_0,x_0,x_1,x_1]f.
\end{aligned}$$

Set $x_1 = x_0 + h$ and $x = x_0 + \theta h$, and denote the remainder $f(x) - p(x)$ by R_T. Then one can show (Problem 1) that

$$\begin{aligned}
p(x) &= f_0 + \theta\Delta f_0 + \theta(1-\theta)(hf_0' - \Delta f_0) \\
&\quad - \theta^2(1-\theta)\Big[(hf_0' - \Delta f_0) + (hf_1' - \Delta f_0)\Big] \\
&= (1-\theta)f_0 + \theta f_1 + \theta(1-\theta)\Big[(1-\theta)(hf_0' - \Delta f_0) - \theta(hf_1' - \Delta f_0)\Big] \qquad (4.3.10)
\end{aligned}$$

For $x \in [x_0, x_1]$ we get the error bound

$$|R_T| \leq \frac{h^4}{384}\max_{x\in[x_0,x_1]}|f^{(4)}(x)|. \qquad (4.3.11)$$

In particular, putting $t = 1/2$, we get the useful formula

$$f_{1/2} = \frac{1}{2}(f_0 + f_1) + \frac{1}{8}h(f_0' - f_1') + R_T. \qquad (4.3.12)$$

Sometimes there are gaps in the sequence of derivatives that are numerically known at a point. The problem is then called **Birkhoff interpolation** or **lacunary** interpolation. We illustrate by two examples that such problems can either have a unique solution or lead to a singular system of linear equations. See also Problems. We use the notation of Sec. 4.1.1.

Example 4.3.4. Given $\tilde{f} = \big(f(-1),\, f'(0),\, f(1)\big)^T$. Try to find a polynomial $p \in \mathcal{P}_3$ that satisfies such data. The new feature is that $f(0)$ is missing.

Set up (4.1.4), i.e. $M_{\mathbf{p}}c = \tilde{f}$, in the power basis.

$$M_{\mathbf{p}} = \begin{pmatrix} 1 & -1 & 1 \\ 0 & 1 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

The determinant is evidently zero, so *there is no solution* for most data. An explanation is that $hf' = \mu\delta f$ for all $f \in \mathcal{P}_3$.

Example 4.3.5. Given $\tilde{f} = \big(f(1),\, f(-1),\, f''(1),\, f''(-1)\big)^T$. Try to find a polynomial $p^* \in \mathcal{P}_4$ that satisfies such data. The new feature is that there are no first derivatives. In this case, we obtain for the power basis,

$$M_{\mathbf{p}} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 2 & 6 \\ 0 & 0 & 2 & -6 \end{pmatrix}.$$

The determinant is 48, and *this interpolation problem is uniquely solvable.* The coefficient vector of p is $c = M_{\mathbf{p}}^{-1}\tilde{f}$.

The first coordinate of c, i.e. $e_1^T c$, is an approximation to $f(0)$; this is also a linear functional of f. Denote by Rf the remainder functional for this approximation, i.e.

$$Rf = f(0) - e_1^T M_{\mathbf{p}}^{-1}\tilde{f}. \tag{4.3.13}$$

4.3.2 Inverse interpolation

It often happens that one has a sequence of pairs $\{(x_i, y_i)\}$ and want to determine a point where $y(x) = c$. We saw an example as early as in the simulation of the motion of a ball (Sec. 1.4), when we computed the landing point. We there used linear interpolation.

In general a natural approach is to reverse the roles of x and y, i.e. to compute the inverse function $x(y)$ for $y = c$, by means of Newton's interpolation formula with the divided differences $[y_i, y_{i+1}, \ldots y_{i+j}]x$ (unscaled or scaled). This is called **inverse interpolation** It is convenient to order the points so that $\ldots < y_5 < y_3 < y_1 < c < y_2 < y_4 < \ldots$. This approach is successful if the function $x(y)$ is suitable for local approximation by a polynomial.

Sometimes, however, the function $y(x)$ is much better suited for local approximation by a polynomial than the inverse function $x(y)$. Then we can instead, for some m, solve the following equation,

$$y_1 + [x_1, x_2]y \cdot (x - x_1) + \sum_{j=2}^{n-1} [x_1, x_2, \ldots x_{j+1}]y\, \Phi_j(x) = c.$$

Again it is convenient to order the points so that the root α comes in the middle, e.g., so that $\ldots < x_5 < x_3 < x_1 < \alpha < x_2 < x_4 < \ldots$.

We write the equation in the form $x = x_1 + F(x)$, where

$$F(x) \equiv \frac{(c - y_1) - \sum_{j=2}^{n-1} [x_1, x_2, \ldots x_{j+1}]y\, \Phi_j(x)}{[x_1, x_2]y}.$$

Then we can use *iteration*. We ignore the sum to get the first guess x^0; this means the same as linear inverse interpolation. We then iterate, $x^i = x_1 + F(x^{i-1})$, until x^i and x^{i-1} are close enough. A more careful termination criterion will be suggested in Chapter 6, where the effect on the result of errors like the interpolation error is also discussed.

Suppose that $x_i - x_1 = O(h)$, $i > 1$, where h is some small parameter in the context (usually some step size), then $\Phi_j(x) = O(h^j)$, $\Phi_j'(x) = O(h^{j-1})$. The divided differences are $O(1)$, and we assume that $[x_1, x_2]y$ is bounded away from zero. Then the terms of the sum decrease like h^j.

By the discussion of iteration in Sec. 1.2, the convergence ratio is $F'(x)$, and this is here approximately

$$\frac{\Phi_2'(x)[x_1, x_2, x_3]y}{[x_1, x_2]y} = O(h).$$

So, if h is small enough, the iterations converge rapidly. If more than two iterations are needed, Aitken acceleration (Sec. 3.3.2) may be practical.

4.3.3 Numerical differentiation

An important problem in many applications is to approximate the derivative of a function using only given function values. A straightforward solution to this problem is to use the derivative of the corresponding interpolation polynomial as the approximation to the derivative of the function. This can also be done for higher order derivatives.

We shall first study the computation of $f'(x_0)$. By the operator expansion (3.3.50) derived in Sec. 3.3.4 we have

$$f'(x_0) = \Big(1 - \frac{\delta^2}{6} + \frac{\delta^4}{30} - \frac{\delta^6}{140} + \frac{\delta^8}{630} - \ldots\Big)\frac{1}{h}\mu\delta f_0, \quad \mu\delta f_0 = \frac{f_1 - f_{-1}}{2}. \tag{4.3.14}$$

By squaring this we obtain

$$f''(x_0) \approx \left(1 - \frac{\delta^2}{12} + \frac{\delta^4}{90} - \frac{\delta^6}{560} + \frac{\delta^8}{3,150} - \frac{\delta^{10}}{16,632} \pm \ldots\right)\frac{\delta^2 f_0}{h^2}. \tag{4.3.15}$$

Suppose that the function values have errors whose magnitude does not exceed $\frac{1}{2}U$. Then the error bound on $\mu\delta f_0 = \frac{1}{2}(f_1 - f_{-1})$ is also equal to $\frac{1}{2}U$. Similarly one can show that the error bounds in $\mu\delta^{(2k+1)} f_0$, for $k = 1:3$ are $1.5U, 5U, 17.5U$, respectively. Thus one gets the upper bounds $U/(2h)$, $3U/(4h)$, and $11U/(12h)$ for the round-off error R_{XF} if one, two, and three terms in (4.3.14).

The truncation error (called R_T) can be estimated by the first neglected term, where

$$\frac{1}{h}\mu\delta^{2k+1} f_0 \approx h^{2k} f^{(2k+1)}(x_0).$$

It has been mentioned several times (see, e.g., Example 3.3.15 in connection with the use of Richardson extrapolation for numerical differentiation) that irregular errors in the values of $f(x)$ are of much greater importance in numerical differentiation than in interpolation and integration.

Example 4.3.6.

Assume that k terms in the formula above is used to approximate $f'(x_0)$, where $f(x) = \ln x$, $x_0 = 3$, and $U = 10^{-6}$. Then $f^{(2k+1)}(3) = (2k)!/3^{2k+1}$, and for the truncation and round-off errors we get:

k	1	2	3
R_T	$0.0123h^2$	$0.00329h^4$	$0.00235h^6$
R_{XF}	$(1/2h)10^{-6}$	$(3/4h)10^{-6}$	$(11/12h)10^{-6}$

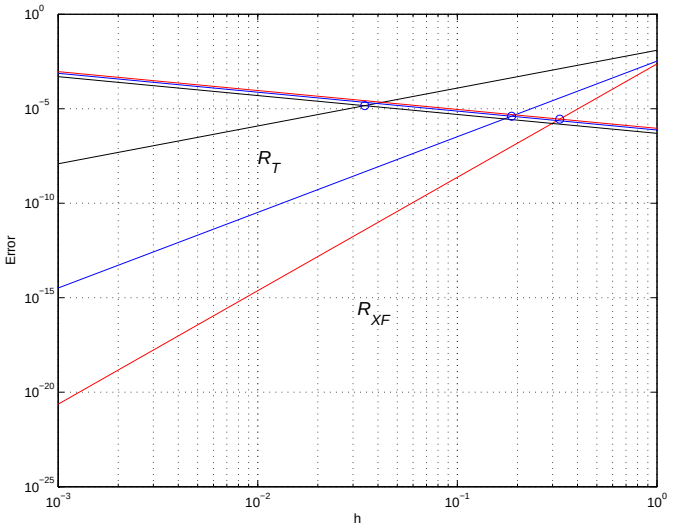

Figure 4.3.1. *Bounds for Truncation error R_T and roundoff error R_{XF} as functions of h for $u = 0.5 \cdot 10^{-6}$.*

In a log-log diagram the plots of R_T and R_{XF} versus h in Figure 4.2.2 are straight lines that illustrate quantitatively the Scylla and Charybdis situation (see explanation in Sec. 3.1.4); the truncation error increases, and the effect of the irregular error decreases with h. One sees how the choice of h, which minimizes the sum of the bounds for the two types of error, depends on u and k, and tells what accuracy can be obtained. The optimal step-lengths for $k = 1, 2, 3$ are $h = 0.0344$, $h = 0.1869$, and $h = 0.3260$, giving error bounds $2.91 \cdot 10^{-5}$, $8.03 \cdot 10^{-6}$, and $5.64 \cdot 10^{-6}$. Note that the optimal error bound with $k = 3$ is not much better than that for $k = 2$.

The effect of the pure rounding errors is important, though it should not be exaggerated. Using IEEE double precision with $u = 1.1 \cdot 10^{-16}$, one can obtain the first two derivatives very accurately by the optimal choice of h. The corresponding figures are $h = 2.08 \cdot 10^{-5}$, $h = 2.19 \cdot 10^{-3}$, and $h = 1.36 \cdot 10^{-2}$, giving the optimal errors bounds $1.07 \cdot 10^{-11}$, $1.52 \cdot 10^{-13}$, and $3.00 \cdot 10^{-14}$, respectively.

It is left to the user (Problem 12) to check and modify the experiments and conclusions indicated in this example. See also the appendix of Chapter 12, where similar questions are discussed in a more general context, namely differentiation for vector-valued functions of vector-valued arguments.

4.3.4 Fast Algorithms for Vandermonde Systems

Given distinct scalars $x_1, x_2, \ldots, x_n$, let V be the Vandermonde matrix

$$V = V(x_1, x_2, \ldots, x_n) = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \cdots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{pmatrix}. \tag{4.3.16}$$

As shown in Sec. 4.1.1 the solution $a = V^{-T} f$ of the **dual Vandermonde system**

$$V^T a = f \tag{4.3.17}$$

gives the coefficients for the interpolating polynomial in the power basis. This polynomial can be computed, e.g., by Newton's interpolation formula in $O(n^2)$ operations. The related **primal Vandermonde systems**

$$Vy = b \tag{4.3.18}$$

arises in problems of determining approximation of linear functionals (see Example 4.1.1). We would like to have a stable and efficient method also for solving the primal system. One possibility would be to use the algorithm given in Sec. 4.2.2, which computes the inverse V^{-1} in about $6n^2$ operations and then form the product $V^{-1}b = y$.

We shall now derive a more efficient and accurate algorithm for solving primal Vandermonde systems. We start by expressing the solution of the dual problem

in terms of a matrix factorization. Using the power basis the unique polynomial satisfying the interpolation conditions $p(x_i) = f_i$, $i = 1 : n$, is

$$p(x) = (1, x, \ldots, x^{n-1})a,$$

where the coefficient vector a satisfies the linear system $V^T a = f$,

One of the most efficient ways to compute $p(x)$ is by Newton's interpolation formula, which uses the basis polynomials

$$p_1(x) = 1, \qquad p_k(x) = (x - x_1) \cdots (x - x_{k-1}), \quad k = 2 : n.$$

We write the polynomial in the form

$$p(x) = c_1 + c_2 p_2(x) + \cdots + c_n p_n(x).$$

where $c_j = [x_1, \ldots, x_{j-1}]f$. These divided differences can be recursively computed as described in Sec. 4.2.1. Then the coefficient vector a of $p(x)$ in the power basis

$$p(x) = a_1 + a_2 x + \cdots + a_n x^{n-1},$$

can be computed by Horner's rule. This is implemented in the algorithm below. Note that the matrix V^T is never formed and we only need storage for a few vectors. The operation count is $\frac{5}{2}n(n+1)$ flops.

Algorithm 4.3.1 Fast Dual Vandermonde Solver

```
function a = dvand(x,f)
% Newton's method for solving a dual Vandermonde system
% V^T(x_1,x_2,...,x_n)a = f.
  n = length(x);
  a = f;
  for k = 1:n-1
     for j = n:(-1):k+1
        a(j) = (a(j) - a(j-1))/(x(j) - x(j-k));
     end
  end
  for k = n-1:(-1):1
     for j = k:n-1
        a(j) = a(j) - x(k)*a(j+1);
      end
  end
```

To derive a corresponding algorithm for solving primal Vandermonde systems the above algorithm can be interpreted as a factorization of the matrix $(V^T)^{-1}$ into a product of diagonal and lower bidiagonal matrices. Let

$$D_k = \mathrm{d}iag\,(1, \ldots, 1, (x_{k+1} - x_1), \ldots, (x_n - x_{n-k})).$$

and define the matrices

$$L_k(x) = \begin{pmatrix} I_{k-1} & 0 \\ 0 & B_{n-k+1}(x) \end{pmatrix}, \quad k = 1 : n-1, \tag{4.3.19}$$

where

$$B_p(x) = \begin{pmatrix} 1 & & & \\ -x & 1 & & \\ & \ddots & \ddots & \\ & & -x & 1 \end{pmatrix} \in \mathbf{R}^{p\times p}, \tag{4.3.20}$$

Then the dual Vandermonde algorithm can be written in matrix terms as $c = U^T f$, $a = L^T c$, where

$$U^T = D_{n-1}^{-1} L_{n-1}(1) \cdots D_1^{-1} L_1(1), \tag{4.3.21}$$

$$L^T = L_1^T(x_1) L_2^T(x_2) \cdots L_{n-1}^T(x_{n-1}). \tag{4.3.22}$$

Since $a = V^{-T} f = L^T U^T f$, we have $V^{-T} = L^T U^T$.

We can now obtain a fast algorithm for solving a primal Vandermonde system $Vy = b$ as follows. Transposing the matrix factorization of V^{-T} gives $V^{-1} = UL$. Hence $y = V^{-1}b = U(Lb)$ and the solution to the primal system can be computed from $d = Lb$, $y = Ud$. Transposing (4.3.21)–(4.3.22) this gives

$$L = L_{n-1}(x_{n-1}) \cdots L_2(x_2) L_1(x_1)$$

$$U = L_1^T(1) D_1^{-1} \cdots L_{n-1}^T(1) D_{n-1}^{-1}.$$

This leads to an algorithm for solving primal Vandermonde systems. The operation count and storage requirement of this are the same as for dual system algorithm.

Algorithm 4.3.2 Fast Primal Vandermonde Solver

```
function y = pvand(x,b)
% Newton's method for solving a primal Vandermonde system
% V(x_1,x_2,...,x_n)y = b.
  n = length(x);
  y = b;
  for k = 1:n-1
     for j = n:(-1):k+1
        y(j) = y(j) - x(k)*y(j-1);
     end
  end
  for k = n-1:(-1):1
     for j = k+1:n
        y(j) = y(j)/(x(j) - x(j-k));
      end
    for j = k:n -1
```

```
        y(j) = y(j) - y(j+1);
    end
end
```

The above two algorithms are not only fast. Also they can give almost full relative accuracy in the solution of some Vandermonde systems, which are so ill-conditioned that Gaussain elimination with complete pivoting fails to produce a single correct digit. This was first observed by Björck and Pereyra [4], from which the following example is taken.

Example 4.3.7.

Consider a primal Vandermonde system $V_n y = b$, with

$$x_i = 1/(i+2), \qquad b_i = 1/2^{i-1}, \quad i = 1:n.$$

The exact solution can be shown to be

$$y_i = (-1)^{i-1}\binom{n}{i}(1+i/2)^{n-1}.$$

Let $\bar{y}_i$ be the solution computed by the primal Vandermonde algorithm and take as a measure of the relative error

$$e_n = \max_{1\le i\le n} |y_i - \bar{y}_i|/|y_i|.$$

Using a hexadecimal floating point arithmetic with $u = 16^{-13} = 2.22 \cdot 10^{-16}$ the following results were obtained:

n	5	10	15	20	25
e_n/u	4	5	10	54	81

The computed solution has small componentwise relative error, which is remarkable since, e.g., $\kappa(V_{10}) = 9 \cdot 10^{13}$.

A forward error analysis given by Higham [30], explains the surprisingly favorable results. If the points are positive and monotonically ordered

$$0 < x_1 < x_2 \cdots < x_n, \tag{4.3.23}$$

then the error in the solution $\bar{a}$ of a Vandermonde system $Vy = b$ computed by the primal algorithm can be bounded as

$$|\bar{a} - a| \le 5u|V^{-1}|\,|b| + O(u^2). \tag{4.3.24}$$

If the components of the right hand side satisfy $(-1)^n b_i \ge 0$, then $|V^{-1}|\,|b| = |V^{-1}b|$, and this bound reduces to

$$|\bar{a} - a| \le 5u|a| + O(u^2), \tag{4.3.25}$$

i.e. the solution is computed with small relative error indendent of the conditioning of V. A similar result holds for the dual algorithm. These good results can be shown to be related to the fact that when (4.3.23) holds, the matrix $V(x_1, x_2, \ldots, x_n)$ is **totally positive**, i.e. the determinant of every squares submatrix of V is positive; see [8].

The given algorithms has been generalized to confluent Vandermonde matrices (see Example 4.3.1) and other classes of Vandermonde-like matrices.

4.3.5 Multidimensional Interpolation

Much of the theory of the introduction can be generalized to other interpolation problems than problems with polynomials in one variable, but one cannot be sure that there is unconditionally a unique solution to the problem. It may not be enough to require that the points are distinct.

Example 4.3.8.

The interpolation by a linear function in two variables,

$$p(x_i, y_i; c) = c_1 + c_2 x_i + c_3 y_i = f_i, \quad i = 1:3,$$

leads to the linear system $Vc = f$, where

$$V = \begin{pmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \end{pmatrix}, \quad c = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}, \quad f = \begin{pmatrix} f_1 \\ f_2 \\ f_3 \end{pmatrix}.$$

This interpolation problem has exactly one solution if V is nonsingular, i.e. when $\det(V) \neq 0$. But $\frac{1}{2}\det(V)$ is just the area of the triangle with vertices (x_i, y_i), $i = 1:3$. If this area is zero then the three points lie on a line and the problem has either infinitely many solutions, or no solution.

The simplest way to generalize interpolation to functions of several variables is to use repeated one-dimensional interpolation, i.e. to work with one variable at a time. The following formula for **bilinear interpolation**,

$$\begin{aligned} f(x_0 + ph, y_0 + qh) &\approx (1-q)\varphi(y_0) + q\varphi(y_0 + k), \\ \varphi(y) &= (1-p)f(x_0, y) + pf(x_0 + h, y). \end{aligned}$$

is the simplest example. After simplification it can written as

$$\begin{aligned} f(x_0 + ph, y_0 + qh) \approx{}& (1-p)(1-q)f_{0,0} + p(1-q)f_{1,0} \\ &+ (1-p)qf_{0,1} + pqf_{1,1}, \end{aligned} \tag{4.3.26}$$

where we have used the notation $f_{ij} = f(x_0 + ih, y_0 + jk)$, $i, j \in \{0, 1\}$. This formula is exact for functions of the form $f(x, y) = a + bx + cy + dxy$, and from equation (4.2.50) we obtain the error bound,

$$\max_{(x,y)\in R} \frac{1}{2}\big(p(1-p)h^2|f_{xx}| + q(1-q)h^2|f_{yy}|\big), \qquad 0 \leq p, q \leq 1,$$

where $R = \{(x, y) : x_0 \le x \le x_0 + h, y_0 \le y \le y_0 + k\}$. The formula for bilinear interpolation can easily be generalized by using higher order interpolation in the x and/or y direction.

In the following we consider explicitly only the case of two dimension, since corresponding formulas for three and more dimensions are analogous.

A **rectangular grid** in the (x, y)-plane with grid spacings h, k in the x and y directions, respectively, consists of points $x_i = x_0 + ih$, $y_i = y_0 + ik$. In the following we use the notation $f(x_i, y_j) = f_{ij}$.

Central difference approximations for partial derivatives using function values can be obtained by working with one variable at a time,

$$\frac{\partial f}{\partial x} = \frac{1}{2h}(f_{i+1,j} - f_{i-1,j}) + O(h^2), \quad \frac{\partial f}{\partial y} = \frac{1}{2k}(f_{i,j+1} - f_{i,j-1}) + O(k^2).$$

For second order derivatives

$$\frac{\partial^2 f}{\partial x^2} = \frac{1}{h^2}(f_{i+1,j} - 2f_{ij} + f_{i-1,j}),$$

and a similar formula holds for $\partial^2 f/\partial y^2$.

Formulas of higher accuracy can also be obtained by operator techniques, based on an operator formulation of Taylor's expansion (see Theorem 4.6.6,

$$f(x_0 + h, y_0 + k) = \exp\left(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\right) f(x_0, y_0) \tag{4.3.27}$$

From this we obtain

$$\begin{aligned} f(x_0 + h, y_0 + k) = f_{0,0} &+ \Big(h\frac{\partial}{\partial x} + k\frac{\partial}{\partial y}\Big) f_{0,0} \\ &+ \Big(h^2\frac{\partial^2}{\partial x^2} + 2hk\frac{\partial^2}{\partial x \partial y} + k^2\frac{\partial^2}{\partial y^2}\Big) f_{0,0} + O(h^2 + k^2). \end{aligned}$$

An interpolation formula valid for all quadratic functions can be obtained by replacing in Taylor's formula the derivatives by difference approximations valid for quadratic polynomials,

$$\begin{aligned} f(x_0 + ph, y_0 + qh) \approx f_{0,0} &+ \frac{1}{2}p(f_{1,0} - f_{-1,0}) + \frac{1}{2}q(f_{0,1} - f_{0,-1}) \\ &+ \frac{1}{2}p^2(f_{1,0} - 2f_{0,0} + f_{-1,0}) \\ &+ \frac{1}{4}pq(f_{1,1} - f_{1,-1} - f_{-1,1} + f_{-1,-1}) \\ &+ \frac{1}{2}q^2(f_{0,1} - 2f_{0,0} + f_{0,-1}). \end{aligned} \tag{4.3.28}$$

This formula uses function values in nine points. (The proof of the expression for approximating the mixed derivative $\dfrac{\partial^2}{\partial x \partial y} f_{0,0}$ is left as an exercise, Problem 2.

Review Questions

1. What is meant by Hermite interpolation (osculatory interpolation)? Prove the uniqueness result for the Hermite interpolation problem.

2. (a) Write down the confluent Vandermonde matrix for the Hermite cubic interpolation problem.

(b) Express the divided difference $[x_0, x_0, x_1, x_1]f$ in terms of f_0, f_0',and f_1,f_1'.

3. How is bilinear interpolation performed? What is the order of accuracy?

Problems and Computer Exercises

1. (a) Construct the divided difference scheme (unscaled or scaled) for the simplest Hermite interpolation problem, where the given data are $f(x_i)$, $f'(x_i)$, $i = 0, 1$; $x_1 = x_0 + h$. Prove all the formulas concerning this problem that are stated at the end of Sec. 4.3.2.

(b) For $f(x) = (1 + x)^{-1}$, $x_0 = 1$, $x_1 = 1.5$, compute $f(1.25)$ by Hermite interpolation. Compare the error bound and the actual error.

(c) Show that for Hermite interpolation

$$|f'(x) - p'(x)| \leq \frac{h^3}{72\sqrt{3}}\Big(\max_{x\in[x_0,x_1]} |f^{(iv)}(x)| + O(h|f^{(v)}(x)|)\Big).$$

Hint: $\frac{d}{dx}[x_0, x_0, x_1, x_1, x]f = [x_0, x_0, x_1, x_1, x, x]f \leq \ldots$.

2. Given $x_i, y(x_i), y'(x_i)$, $x_i = x_0 + ih$, $i = 1, 2, 3$. Let $p \in \mathcal{P}_6$ be the Hermite interpolation polynomial to these data.

(a) Find the remainder term, and show that the interpolation error for $x \in [x_1, x_3]$ does not exceed $h^6 \max |f^{(6)}(x)|/4860$ in magnitude.

(b) Write a program that computes $p(x_1 + 2jh/k)$, $j = 0 : k$.

COMMENT: This is one of several possible procedures for starting a multistep method for an ordinary differential equation $y' = f(x, y)$. Two steps with an accurate one-step method, provide values of y, y', and this program then produces starting values (y only) for the multistep method.

3. Give a short and complete proof of the uniqueness of the interpolation polynomial for distinct points, by the use of the ideas of the proof of Theorem 4.3.1.

4. Derive an approximate formula for $f'(x_0)$ when the values $f(x_{-1}), f(x_0), f(x_1)$ are given at three *non-equidistant* points. Give an approximate remainder term. Check the formula and the error estimate on an example of your own choice.

5. (a) Given a sequence of function values $f_1, f_2, f_3 \ldots$ at equidistant points $x_j = x_0 + jh$. Assume that $\min f_j = f_n$, and let $p(x)$ be the quadratic interpolation

polynomial determined by f_{n-1}, f_n, f_{n+1}. Show that

$$\min p(x) = f_n - \frac{(\mu\delta f_n)^2}{2\delta^2 f_n}, \quad \text{at} \quad x = x_n - h\frac{\mu\delta f_n}{\delta^2 f_n},$$

and that the error of the minimum value can be bounded by $\max|\Delta^3 f_j|/\sqrt{243}$, where j is in some neighborhood of n. Why and how is the estimate of x less accurate?

(b) Write a handy program that includes the search all local maxima and minima. Sketch or work out improvements of this algorithm, perhaps with ideas of inverse interpolation and with cubic interpolation. And perhaps for non-equidistant data.

6. (a) Compute by bilinear interpolation $f(0.5, 0.25)$ when

$$f(0,0) = 1, \quad f(1,0) = 2, \quad f(0,1) = 3, \quad f(1,1) = 5.$$

(b) Set $c = (c_1, c_2, c_3, c_4, c_5, c_6)^T$,

$$p(x, y; c) = c_1 + c_2 x + c_3 y + c_4 x^2 + c_5 xy + c_6 y^2.$$

Consider the interpolation problem: Given x_i, y_i, f_i, $i = 1:6$; try to find c, so that $p(x_i, y_i; c) = f_i$, $i = 1:6$.
Choose x_i, y_i, f_i, $i = 1:6$ by 18 independent random numbers, solve the linear system $p(x_i, y_i; c) = f_i$, $i = 1:6$, look at $\max|c_i|$. Repeat this (say) 25 times. You have a fair chance to avoid singular cases, or cases where $\max|c_i|$ is very large.

(c) Now choose (x_i, y_i) as 6 distinct points on some circle in $\mathbf{R}^2$, and choose f_i at random. This should theoretically lead to a singular matrix. Explain why, and find experimentally the rank (if your software has convenient commands or routines for that). Find a general geometric characterization of the sextuples of points (x_i, y_i), $i = 1:6$, that lead to singular interpolation problems.

Hint: Brush up your knowledge of conic sections.

7. Derive a formula for $f''_{xy}(0,0)$ using f_{ij}, $|i| \le 1, |j| \le 1$, which is exact for all quadratic functions.

4.4 Piecewise Polynomial Interpolation

4.4.1 Bernstein Polynomials

Parametric curves are often used to find a functional form of a curve (or surface) given geometrically by a set of points. Let $c(t)$, $t \in [0,1]$ a parametric curve connecting two points p_0 and p_1 in $\mathbf{R}^{\mathbf{d}}$, so that $p_0 = c(0)$ and $p_1 = c(1)$. In the simplest case we can take $c(t)$ to be linear and write

$$c(t) = (1-t)p_0 + tp_1.$$

If extended to a set of points $p_0, \ldots, p_n$, $n > 1$ this will not give a smooth curve and is therefore of limited interest. We now generalize this approach and take $c(t)$ to be a polynomial of degree n.

The **Bernstein polynomials**[12] are defined by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \quad i = 0 : n. \tag{4.4.1}$$

Using the binomial theorem we have

$$1 = ((1-t)+t)^n = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} = \sum_{i=0}^{n} B_i^n(t),$$

that is, the Bernstein polynomials of degree n are nonnegative and give a partition of unity. The Bernstein polynomials of degree n form a basis for the space of polynomials of degree $\le n$.

For $n = 3$ the four cubic Bernstein polynomials are

$$B_0^3 = (1-t)^3, \quad B_1^3 = 3t(1-t)^2, \quad B_2^3 = 3t^2(1-t), \quad B_3^3 = t^3. \tag{4.4.2}$$

are plotted in Figure 4.4.1.

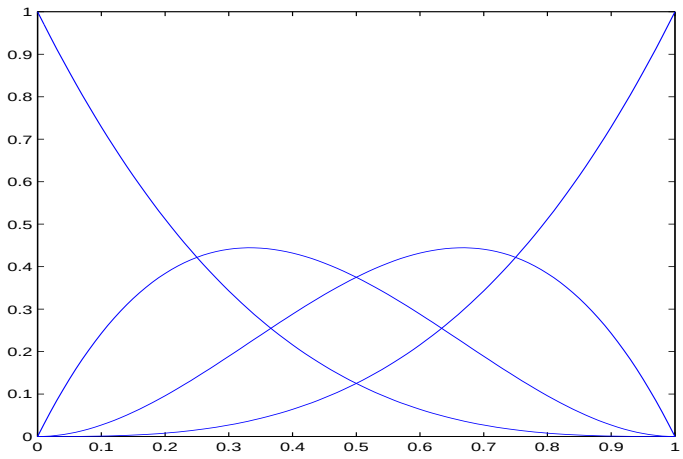

Figure 4.4.1. *Bernstein polynomials.*

Some important properties of the Bernstein polynomials are given in the following theorem.

[12] Sergi NatanovičBernštein (1880–1968) Russian mathematician, who made major contributions to polynomial approximation. In 1911 he introduced the polynomials named after him.

Theorem 4.4.1. *The Bernstein polynomials $B_i^n(t)$ have the following properties:*

1. $B_i^n(t) > 0, \quad t \in (0,1)$ *(nonnegativity);*
2. $B_i^n(t) = B_{n-i}^n(1-t)$ *(symmetry);*
3. *The Bernstein polynomials $B_i^n(t)$ have a unique maximum value at $t = i/n$ on $[0,1]$;*
4. *The Bernstein polynomials satisfy the following recursion formula*

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t), \quad i = 0:n. \tag{4.4.3}$$

Proof. The first three properties follows directly from the definition (4.4.1). The recursion formula is a consequence of the relation

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$$

between the binomial coefficients. □

Starting with $B_0^0(t) = 1$, and setting $B_{-1}^n(t) = B_{n+1}^n(t) = 0$ this recursion can be used to evaluate the Bernstein polynomials at a given point t.

4.4.2 Parametric Bézier Curves

Given a set of $n+1$ **control points** p_i, $i = 0:n$, the Bézier curve is given by

$$c(t) = \sum_{i=0}^{n} p_i B_i^n(t), \quad t \in [0,1]. \tag{4.4.4}$$

The Bézier curve interpolates the first and last control points p_0 and p_1. It follows directly from the form of (4.4.4) that applying an affine transformation to $c(t)$ can be performed simply by applying the same transformation to the control points. Hence the Bézier curve has the desirable property that it is invariant under translations and rotations.

Bézier curves are a major tool in computer graphics, where usually $p_i \in \mathbf{R}^2$ or $\mathbf{R}^3$. One important application is in computer aided design (CAD) systems, used, e.g., in the auto industry. Often a curve is constructed by smoothly patching together several Bézier curves of lower order.

Example 4.4.1. A quadratic Bézier curve is given by

$$c(t) = (1-t)^2 p_0 + 2t(1-t)p_1 + t^2 p_2, \quad t \in [0,1].$$

Clearly $c(0) = p_0$ and $c(1) = p_2$. For $t = 1/2$ we get

$$c(1/2) = \frac{1}{2}\Big(\frac{p_0 + p_2}{2} + p_1\Big).$$

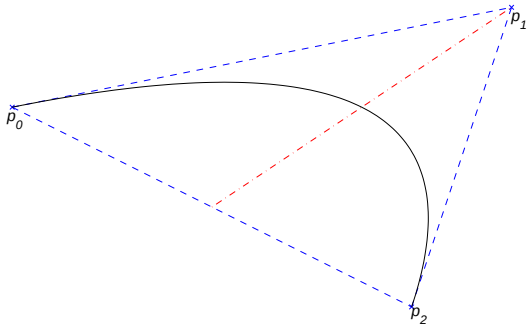

Figure 4.4.2. *Quadratic Bézier curve with control points.*

Hence we can construct the point $c(1/2)$ geometrically as the intersection between the the midpoint of the line between p_0 and p_2 and the point p_1; see Figure 4.4.2.

The **Bézier polygon** is the closed piecewise linear curve connecting the control points p_i and p_{i+1}, $i = 0 : n-1$ and finally p_n and back to p_0. In Figure 4.4.2 this is the polygon formed by the dashed lines. This polygon provides a rough idea about the shape of the Bézier curve.

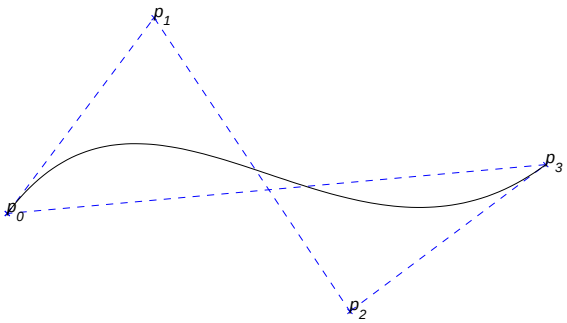

Figure 4.4.3. *Cubic Bézier curve with control points* $P_0, \ldots, P_3$.

Definition 4.4.2. *A set S in $\mathbf{R}^d$ is called convex if for any points $x, y \in S$, the straight line*

$$\{tx + (1-t)y \mid t \in (0,1)\}$$

is also contained in S; The **convex hull** *of a set S in $\mathbf{R}^d$ is the smallest convex subset of $\mathbf{R}^d$, which contains S.*

From the definition (4.4.4) of the Bézier curve it follows that for all $t \in [0,1]$, the curve $c(t)$ is a convex combination of the control points. Therefore $c(t)$ lies within the convex hull of the control points. Often, but not always, the convex hull is the region enclosed Bézier polygon; cf. Figures 4.4.2–3.

The variation of a function in an interval $[a,b]$ is the least upper bound on the sum of the oscillations in the closed subintervals $[a,x_1]$, $[x_1,x_2]$, $\ldots$, $[x_n,b]$, for all possible such subdivisions. The Bézier curve is variation diminishing. In particular if the control points p_i are monotonic, so is $c(t)$. Further, if p_i are convex (concave) so is $c(t)$.

Usually all control points are not known in advance but the curves shape is controlled by moving the control points until the curve has the desired shape. For example, in the quadratic case moving p_1 has a direct and intuitive effect on the curve $c(t)$. An advantage of the Bernstein basis for representing polynomials is that the coefficients (control points) is closely related to the shape of the curve. This is not the case when using a monomial or Chebyshev basis.

Theorem 4.4.3. *The Bézier curve $c(t)$ is tangent to $p_1 - p_0$ and $p_n - p_{n-1}$ for $t = 0$ and $t = 1$, respectively.*

Proof. To show this we compute the derivative of the Bernstein polynomial (4.4.1)

$$\frac{d}{dt}B_i^n(t) = \begin{cases} -nB_0^{n-1}(t), & \text{if } i = 0 \\ n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right), & \text{if } 0 < i < n; \\ nB_{n-1}^{n-1}(t), & \text{if } i = n. \end{cases}$$

This follows from

$$\frac{d}{dt}B_i^n(t) = \binom{n}{i}\left(it^{i-1}(1-t)^{n-i} - (n-i)t^i(1-t)^{n-i-1}\right),$$

and using the definition of the Bernstein polynomials. Setting $t = 0$ we find that $\frac{d}{dt}B_i^n(0) = 0$, $i > 1$, and therefore from (4.4.4)

$$\frac{d}{dt}c(t) = n(p_1 - p_0),$$

which shows the statement for $t = 0$. The result for $t = 1$ follows from symmetry. □

More generally, at a boundary point the kth derivative of the Bézier curve depends only on the k closest control points. This fact is useful for smoothly joining together several pieces of Bézier curves.

To evaluate the Bézier curve at $t \in [0,1]$ we use the recursion formula (4.4.3) to obtain

$$c(t) = \sum_{i=0}^{n} p_i B_i^n(t)$$

$$= (1-t)\sum_{i=0}^{n-1} p_i B_i^{n-1}(t) + t\sum_{i=1}^{n} p_i B_{i-1}^{n-1}(t)$$
$$= \sum_{i=0}^{n-1} \big((1-t)p_i + tp_{i+1}\big) B_i^{n-1}(t) = \sum_{i=0}^{n-1} p_i^{(1)}(t) B_i^{n-1}(t)$$

where we have introduced the new auxiliary control points

$$p_i^{(1)}(t) = (1-t)p_i + tp_{i+1}, \quad i = 0 : n-1,$$

as convex combinations (depending on t) of the original control points. Using this result we can successively lower the grade of the Bernstein polynomial until we arrive at $B_0^0 = 1$. This gives a recursion scheme for the auxiliary control points due to de Casteljau:

$$\begin{aligned} p_i^{(0)}(t) &= p_i, \quad i = 0 : n \\ p_i^{(r)}(t) &= (1-t)p_i^{(r-1)}(t) + tp_{i+1}^{(r-1)}(t), \quad i = 0 : n-r. \end{aligned} \tag{4.4.5}$$

It follows

$$c(t) = \sum_{i=0}^{n-r} p_i^{(r)}(t) B_i^{n-r}(t), \quad r = 0 : n \tag{4.4.6}$$

and in particular $c(t) = p_0^{(n)}$.

De Casteljau's algorithm can be arranged in a triangular array

$$\begin{array}{lllll} p_0 = p_0^{(0)} & & & & \\ & p_0^{(1)} & & & \\ p_1 = p_1^{(0)} & & p_0^{(2)} & & \\ & p_1^{(1)} & & & \\ p_2 = p_2^{(0)} & \vdots & p_1^{(2)} & \ddots & \\ \vdots & \vdots & \vdots & \vdots & p_0^{(n)} \\ \vdots & p_{n-2}^{(1)} & \vdots & & \\ p_{n-1} = p_{n-1}^{(0)} & & p_{n-2}^{(2)} & & \\ & p_{n-1}^{(1)} & & & \\ p_n = p_n^{(0)} & & & & \end{array} \tag{4.4.7}$$

Since at each step the new control points are convex combinations of the previous control points, de Casteljau's algorithm is very stable. It uses about n^2 operations and so is less efficient than Horner's algorithm for evaluating a polynomial in the monomial basis.

The kth derivative of $c(t)$ is also available from the de Casteljau scheme. It holds that

$$c'(t) = n(p_1^{n-1} - p_0^{n-1}),$$

$$c''(t) = n(n-1)(p_2^{n-2} - 2p_1^{n-2} + p_0^{n-2}), \ldots,$$

and in general

$$c^{(k)}(t) = \frac{n!}{(n-k)!}\Delta^k p_0^{n-k}, \quad 0 \le k \le n, \tag{4.4.8}$$

where the difference operates on the lower index i.

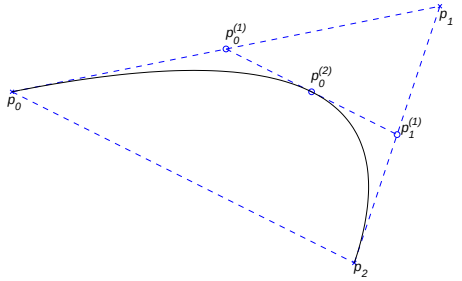

Figure 4.4.4. *Casteljau's algorithm for* $n = 2$, $t = \frac{1}{2}$.

De Casteljau's algorithm is illustrated for the quadratic case in Figure 4.4.4, where the following geometric interpretation can be observed. In the interval $[0, t]$ the Bézier curve is represented by a quadratic spline with control points $p_0, p_0^{(1)}, p_0^{(2)}$. In the remaining interval $[t, 1]$ it is represented by a quadratic spline with control points $p_0^{(2)}, p_1^{(1)}, p_2$. Note that these two sets of control points lies closer to the curve $c(t)$. After a few more subdivisions it will be hard to distinguish the polygon joining the control points form the curve.

4.4.3 Splines

The name **spline** comes from a very old technique in drawing smooth curves in which a thin strip of wood, called a draftsman's spline, is bent so that it passes trough a given set of points, see Figure 4.5.5. The points of interpolation are called **knots** and the spline is secured at the knots by means of lead weights called **ducks**. Before the computer age splines were used in ship building and other engineering designs.

A mathematical model of a spline was given by Daniel Bernoulli (1742) and Euler (1744)[13]. By Hamilton's principle the shape the spline will take is such that its elastic strain energy is minimized. The strain energy of a spline $y = s(x)$, $x \in [a, b]$ in the plane is given by

$$E(s) = \int_a^b \kappa(x)^2 \, dx$$

[13]Euler derived the differential equation satisfied by the spline using techniques now known as calculus of variation and Lagrange multipliers. When Euler did this work Lagrange was still a small child!

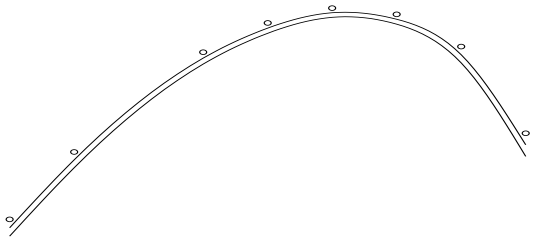

Figure 4.4.5. *The original spline.*

where

$$\kappa(x) = \frac{s''(x)}{(1 + (s'(x))^2)^{3/2}}.$$

is the curvature of the spline. For slowly varying deflections, i.e. when $(s'(x))^2$ is approximately constant, the approximation

$$E(s) =\approx const \cdot \int_a^b s''(x)^2 \, dx,$$

Under this assumption, according to elasticity theory, $s(x)$ is built up of **piecewise** third degree polynomials (cubic polynomials) in such a way that $s(x)$ *and its two first derivatives are everywhere continuous.* Let x_i, $i = 0 : m$ be the points the spline is forced to interpolate. Then the third derivative can have discontinuities at the points x_i. Such a function is called a **cubic spline function**, or shorter, a **cubic spline** The points x_i, $i = 0 : m$, are called **breakpoints** or **knots**.

The mathematical concept of spline functions was introduced in 1946 by Schoenberg in the seminal paper [44]. The importance of the B-spline basis for spline approximation (see Sec. sec4.4.6) was also first appreciated by Schoenberg. These were not used in practical calculations for general knot sequences until the early seventies, when a stable recurrence relation was established independently by de Boor [6] and Cox [17].

Spline functions are now used extensively in computer aided design (CAD), where curves and surfaces have to be represented mathematically, so that they can be manipulated and visualized easily. Important applications occur in computer-aided design, analysis and manufacturing as well as in aircraft and automotive industries. Spline functions can also be used in the numerical treatment of boundary-value problems for differential equations.

With the use of splines, there is no reason to fear equidistant data, as opposed to the situation with higher-degree polynomials. Also, if the function to be approximated is badly behaved somewhere then, using spline approximation with properly chosen knots, the effect of this can be confined locally, allowing good approximation elsewhere in the interval. In the following we restrict ourself to consider curves in the plane. For more information on spline approximations of curves and surfaces the reader is referred to de Boor [7], where also FORTRAN programs for computations with spline functions can be found, and Dierckx [22].

We have seen that it is often not efficient to approximate a given function by a single polynomial over its entire range. On the other hand, polynomials of low degree can give good approximations *locally* in a small interval. Therefore it is natural to consider approximations by piecewise polynomials of different degrees of global continuity.

We first consider the simplest curve interpolating given values $y_i = f(x_i) \in [a, b$ on a grid

$$\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$$

is by a broken line $s(x)$, where

$$s(x) = q_i(x) = y_{i-1} + d_i(x - x_i), \quad x \in [x_{i-1}, x_i), \quad i = 1 : m \tag{4.4.9}$$

Here

$$h_i = x_i - x_{i-1}, \qquad d_i = [x_{i-1}, x_i]f(x) = (y_i - y_{i-1})/h_i. \tag{4.4.10}$$

i.e. d_i is the divided difference of f at $[x_{i-1}, x_i]$. If $f \in C^2[a, b]$ then the error satisfies (see (4.2.50)).

$$|f(x) - s(x)| \le \frac{1}{8} \max_i \Big(h_i^2 \max_{x \in [x_{i-1}, x_i]} |f''(x)| \Big). \tag{4.4.11}$$

Hence, we can make the error arbitrary small by decreasing $\max_i h_i$. An important property of interpolation with a piecewise affine function is that it preserves monotonicity and convexity of the interpolated function.

The broken line interpolating function has a discontinuous first derivative at the knots, which makes it unsuitable for many applications. To get better smoothness piecewise polynomials of higher degree need to be used. Although piecewise quadratic approximation is sometimes useful, piecewise *cubic* polynomials with continuous second derivatives are by far the more important (see Figure 4.4.6.

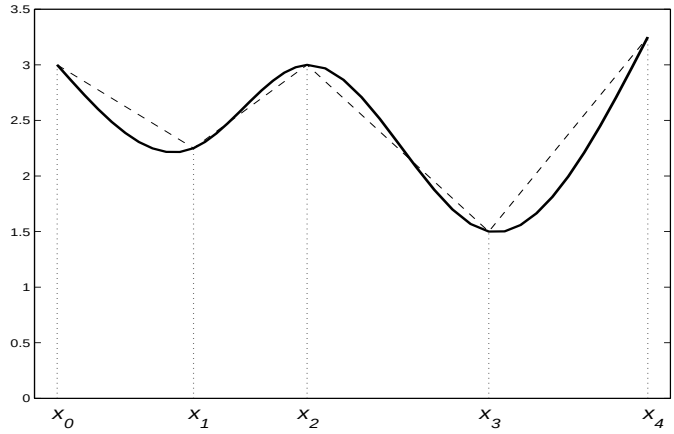

Figure 4.4.6. *Broken line and cubic spline interpolation.*

A cubic polynomial $q_i(x)$ on the interval $[x_{i-1}, x_i)$ is uniquely determined by the values of the function and its first derivative at the end points of the interval. This follows from the more general result on Hermite interpolation in Theorem 4.3.1. By (4.3.11), translated to the notation in (4.4.10), the cubic $q_i(x)$ can be written in the form

$$q_i(x) = \theta y_i + (1-\theta)y_{i-1} + h_i\theta(1-\theta)\left[(k_{i-1}-d_i)(1-\theta) - (k_i - d_i)\theta\right], \quad (4.4.12)$$

where h_i, d_i, $i = 1:m$, are as in (4.4.10),

$$\theta = \frac{x - x_{i-1}}{h_i} \in [0,1), \quad x \in [x_{i-1}, x_i), \quad (4.4.13)$$

is a local variable and $k_i = q_i'(x_i)$.

If the interpolating spline $s(x)$ is to be evaluated at many points, a form that is more efficient to use than (4.4.12) is the piecewise polynomial form (**pp form**)

$$q_i(x) = y_{i-1} + a_{1i}(x - x_{i-1}) + a_{2i}(x - x_{i-1})^2 + a_{3i}(x - x_{i-1})^3, \quad (4.4.14)$$

$i = 1:m$. From (4.4.12) we obtain after some calculation

$$\begin{aligned} a_{1i} &= q_i'(x_{i-1}) = k_{i-1}, \\ a_{2i} &= \tfrac{1}{2}q_i''(x_{i-1}) = (3d_i - 2k_{i-1} - k_i)/h_i, \\ a_{3i} &= \tfrac{1}{6}q_i'''(x_{i-1}) = (k_{i-1} + k_i - 2d_i)/h_i^2. \end{aligned} \quad (4.4.15)$$

Using Horner's scheme $q_i(x)$ can be evaluated from (4.4.14) using only four multiplication.

With piecewise cubic polynomials we can interpolate given function values and first derivatives on the grid Δ. By construction the interpolating piecewise cubic function $s(x)$ will have continuous first derivatives. If $f \in C^4[a,b]$ then it follows from the remainder term (4.3.11) that the error satisfies

$$|f(x) - s(x)| \le \frac{1}{384}\max_i \Big(h_i^4 \max_{x\in[x_{i-1},x_i]} |f^{(iv)}(x)|\Big). \quad (4.4.16)$$

It can be shown (Problem 1c of Sec. 4.3) that also the first derivative of $s(x)$ is a good approximation to $f'(x)$. If $f \in C^5[a,b]$ we have

$$|f'(x) - s'(x)| \le \frac{1}{72\sqrt{3}}\max_i \Big(h_i^3 \max_{x\in[x_i,x_{i+1}]} |f^{(iv)}(x) + O(h_i f^{(5)}(x))|\Big). \quad (4.4.17)$$

Sometimes it is useful to consider the values k_i, $i = 0:m$, as parameters which are used to give the interpolating function the desired shape. In the next section we show that it is possible to choose these parameters such that the interpolating function $s(x)$ also has a continuous *second* derivative.

We shall now formally define a spline function of order $k \ge 1$.

Definition 4.4.4.

Let $\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$ be a subdivision of the interval $[a,b]$. A spline function on Δ of order $k \ge 1$ (degree $k-1 \ge 0$), is a real function s with properties:

(a) *For $x \in [x_i, x_{i+1}]$, $i = 0 : m-1$, $s(x)$ is a polynomial of degree $< k$.*

(b) *For $k = 1$, $s(x)$ is a piecewise constant function. For $k \geq 2$, $s(x)$ and its first $k-2$ derivatives are continuous on $[a, b]$, i.e. $s(x) \in C^{k-2}[a, b]$.*

We denote by $S_{\Delta,k}$ the set of all spline functions of order k on Δ. From the definition it follows that if $s_1(x)$ and $s_2(x)$ are spline functions of the same degree, so is $c_1 s_1(x) + c_2 s_2(x)$. Thus $S_{\Delta,k}$ is a *linear space*.

Examples of elements of $S_{\Delta,k}$ are the **truncated power** functions

$$(x - x_j)_+^{k-1}, \quad j = 1 : m-1,$$

and all their linear combinations. Moreover, $\mathcal{P}_k$ is a linear subspace of $S_{\Delta,k}$.[14] Conversely, together these functions span $S_{\Delta,k}$. All we need for the first subinterval is a basis of $\mathcal{P}_k$, e.g., the power basis $\{1, x, \ldots, x^{k-1}\}$. Further, all we need for each additional subinterval $[x_j, x_{j+1})$, $j = 1 : m-1$, is the new basis function $(x - x_j)_+^{k-1}$. One can show that these $k + m - 1$ functions are linearly independent. The dimension of the linear space thus is $k + m - 1$.

4.4.4 Cubic Spline Interpolation

In the following we shall first study cubic spline functions which *interpolate* a given function $f(x)$ at the grid Δ, i.e. the space $S_{\Delta,4}$. By definition a cubic spline consists of cubic polynomials pieced together in such a way that their values and first *two* derivatives coincide at the knots. In contrast to Hermite interpolation, the cubic polynomial in each subinterval will now depend on *all data points*.

Theorem 4.4.5.

Every cubic spline function, with knots $a = x_0 < x_1 < \cdots < x_m = b$, which interpolates the function $y = f(x)$,

$$s(x_i) = f(x_i) = y_i, \quad i = 0 : m,$$

equals for $x \in [x_{i-1}, x_i)$, $i = 1 : m$ a third degree polynomial of the form (4.4.12)*. The $m + 1$ parameters k_i, $i = 0 : m$, satisfy $m - 1$ linear equations*

$$h_{i+1}k_{i-1} + 2(h_i + h_{i+1})k_i + h_i k_{i+1} = 3(h_i d_{i+1} + h_{i+1} d_i), \quad i = 1 : m-1, \tag{4.4.18}$$

where $h_i = x_i - x_{i-1}$, $d_i = (y_i - y_{i-1})/h_i$.

Proof. We require the second derivative of the spline $s(x)$ to be continuous at x_i, $i = 1 : m-1$. We have

$$s(x) = \begin{cases} q_i(x), & x \in [x_{i-1}, x_i), \\ q_{i+1}(x), & x \in [x_i, x_{i+1}), \end{cases}$$

[14] Recall the notation $(x-u)_+^j = \max\{x-u, 0\}$ that was introduced in Sec. 3.2.3 in connection with the Peano kernel.

where $q_i(x)$ is given by (4.4.14)–(4.4.15). Differentiating $q_i(x)$ twice we get $\frac{1}{2}q_i''(x) = a_{2,i} + 3a_{3,i}(x - x_{i-1})$, and putting $x = x_i$

$$\tfrac{1}{2}q_i''(x_i) = a_{2,i} + 3a_{3,i}h_i = (k_{i-1} + 2k_i - 3d_i)/h_i. \tag{4.4.19}$$

Replacing i by $i+1$ we get $\frac{1}{2}q_{i+1}''(x) = a_{2,i+1} + 3a_{3,i+1}(x - x_i)$, and hence

$$\tfrac{1}{2}q_{i+1}''(x_i) = a_{2,i+1} = (3d_{i+1} - 2k_i - k_{i+1})h_{i+1}. \tag{4.4.20}$$

These last two expressions must be equal, which gives the conditions

$$\frac{1}{h_i}(k_{i-1} + 2k_i - 3d_i) = \frac{1}{h_{i+1}}(3d_{i+1} - 2k_i - k_{i+1}), \quad i = 1 : m-1. \tag{4.4.21}$$

Multiplying both sides by $h_i h_{i+1}$ we get (4.4.18). □

The conditions (4.4.18) are $(m-1)$ linearly independent[15] equations for the $(m+1)$ unknowns k_i, $i = 0 : m$. Two additional conditions are therefore needed to uniquely determine the interpolating spline. The four most important choices are discussed below.

(i) If the derivatives at the end points are known we can take

$$k_0 = f'(a), \qquad k_m = f'(b). \tag{4.4.22}$$

The corresponding spline function $s(x)$ is called the **complete cubic spline interpolant**. If k_0 and k_m are determined by numerical differentiation with a truncation error that is $O(h^4)$, we call the spline interpolant **almost complete**. For example, k_0 and k_m may be the sum of (at least) four terms of the expansions

$$Df(x_0) = \frac{1}{h}\ln(1+\Delta)y_0, \qquad Df(x_m) = -\frac{1}{h}\ln(1-\nabla)y_m,$$

into powers of the operators Δ and ∇, respectively.[16]

(ii) A physical spline is straight outside the interval $[a, b]$, i.e. $s''(x) = 0$ for $x \le a$ or $x \ge b$. Thus $q_1''(x_0) = q_m''(x_m) = 0$. From (4.4.20) and (4.4.19) we have

$$\tfrac{1}{2}q_i''(x_{i-1}) = (3d_i - 2k_{i-1} - k_i)/h_i, \quad \tfrac{1}{2}q_i''(x_i) = -(3d_i - k_{i-1} - 2k_i)/h_i.$$

Setting $i = 1$ in the first equation and $i = m$ in the second gives the two conditions

$$\begin{aligned} 2k_0 + k_1 &= 3d_1 \\ k_{m-1} + 2k_m &= 3d_m. \end{aligned} \tag{4.4.23}$$

[15] The equations are strictly row diagonally dominant (see Sec. 7.4.1) and therefore linearly independent

[16] Two terms of the *central* difference expansion in (3.3.45) or one *Richardson* extrapolation, see Example 3.4.16, give higher accuracy, but need extra function values outside the grid Δ.

The corresponding approximating spline is called the **natural spline interpolant**. It should be stressed that when a cubic spline is used for the approximation of a smooth function, these boundary conditions are *not natural*!

(iii) If the end point derivatives are not known, a convenient condition is to require that $s'''(x)$ be continuous across the first and last interior knots x_1 and x_{m-1}. Hence $q_1(x) = q_2(x)$ and $q_{m-1}(x) = q_m(x)$. Then x_1 and x_{m-1} are no longer knots, and these conditions are known as "**not a knot**" conditions. From (4.4.15) we obtain,

$$\tfrac{1}{6}q_i'''(x) = a_{3i} = (k_{i-1} + k_i - 2d_i)/h_i^2, \quad x \in [x_{i-1}, x_i), \quad i = 1:m.$$

Hence the condition $q_1''' = q_2'''$ gives $(k_0 + k_1 - 2d_1)/h_1^2 = (k_1 + k_2 - 2d_2)/h_2^2$, or

$$h_2^2 k_0 + (h_2^2 - h_1^2)k_1 - h_1^2 k_2 = 2(h_2^2 d_1 - h_1^2 d_2).$$

Since this equation would destroy the tridiagonal form of the system, we use (4.4.18), with $i = 1$ to eliminate k_2. This gives the equation

$$h_2 k_0 + (h_2 + h_1)k_1 = 2h_2 d_1 + \frac{h_1(h_2 d_1 + h_1 d_2)}{h_2 + h_1}. \tag{4.4.24}$$

If the right boundary condition is treated similarly we get

$$(h_{m-1} + h_m)k_{m-1} + h_{m-1}k_m = 2h_{m-1}d_m + \frac{h_m(h_{m-1}d_m + h_m d_{m-1})}{h_{m-1} + h_m}. \tag{4.4.25}$$

(iv) If the spline is used to represent a periodic function, then $y_0 = y_m$ and the boundary conditions

$$s'(a) = s'(b), \qquad s''(a) = s''(b), \tag{4.4.26}$$

suffice to determine the spline uniquely. From the first condition it follows that $k_0 = k_m$, which can be used to eliminate k_0 in the equation (4.4.18) for $k = 1$. The second condition in (4.4.26) gives using (4.4.21) $(k_0 + 2k_1 - 3d_1)/h_1 = -(2k_{m-1} + k_m - 3d_m)/h_m$, or after eliminating k_0,

$$2h_m k_1 + 2h_1 k_{m-1} + (h_1 + h_m)k_m = 3(h_m d_1 + h_1 d_m),$$

The spline interpolant has the following best approximation property.

Theorem 4.4.6.

Among all functions g that are twice continuously differentiable on $[a, b]$ and which interpolate f at the points $a = x_0 < x_1 < \cdots < x_m = b$, the natural spline function minimizes

$$\int_a^b \big(s''(t)\big)^2 dt.$$

The same minimum property holds for the complete spline interpolant, if the functions g satisfy $g'(a) = f'(a)$, and $g'(b) = f'(b)$.

Proof. See de Boor [7, 1978, Chapter 5]. □

Due to this property spline functions yield smooth interpolation curves, except for rather thin oscillatory layers near the boundaries if the "natural" boundary conditions $s''(a) = s''(b) = 0$ are far from being satisfied. For the complete or almost complete cubic spline and for cubic splines determined by the "not-a-knot" conditions, these oscillations are much smaller; see Sec. 4.4.4.[17]

Equations (4.4.18) together with any of these boundary conditions give rise to a well-conditioned system of linear equations for determining the derivatives k_i. For the first three boundary conditions the system is **tridiagonal**. As demonstrated in Example 1.3.5 such systems can be solved by Gaussian elimination in $\mathcal{O}(n)$ flops. It can be proved that a sufficient condition for the algorithm given there to be stable is that A is **diagonally dominant**, i.e.

$$|b_1| > |c_1|, \quad |b_k| > |a_{k-1}| + |c_k|, \quad k = 2 : m-1, \quad |b_n| > |a_{m-1}|.$$

It is also stable for the system resulting from the not-a-knot boundary condition although this is not diagonally dominant in the first and last row; see Problem 2b. (Methods for solving general banded linear systems will be studied in more detail in Volume II, Sec. 7.4).

Example 4.4.2. In the case of spline interpolation with constant stepsize $h_i = h$ equation (4.4.18) becomes

$$k_{i-1} + 4k_i + k_{i+1} = 3(d_i + d_{i+1}), \quad i = 1 : m-1. \tag{4.4.27}$$

The "not a knot" boundary conditions (4.4.24)–(4.4.25) become

$$k_0 + 2k_1 = \tfrac{1}{2}(5d_1 + d_2), \qquad 2k_{m-1} + k_m = \tfrac{1}{2}(d_{m-1} + 5d_m). \tag{4.4.28}$$

We obtain a tridiagonal system $Tk = g$, where,

$$\begin{pmatrix} 1 & 2 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 2 & 1 \end{pmatrix} \begin{pmatrix} k_0 \\ k_1 \\ \vdots \\ k_{m-1} \\ k_m \end{pmatrix} = 3 \begin{pmatrix} (5d_1 + d_2)/6 \\ d_1 + d_2 \\ \vdots \\ d_{m-1} + d_m \\ (d_{m-1} + 5d_m)/6 \end{pmatrix}.$$

except for the first and last row, the elements of T are constant along the diagonals. The condition number of T increases very slowly with m; for example, $\kappa(T) < 16$ for $m = 100$.

Consider now the periodic boundary conditions in (iv). Setting $k_m = k_0$ in the last equation we obtain a linear system of equations $Tk = g$ for $k_1, \ldots, k_{m-1}$

[17] When a spline is to be used for the approximate representation of a smooth function, the natural spline is *not* a natural choice.

where

$$T = \left(\begin{array}{ccccc|c} b_1 & c_1 & & & & a_m \\ a_1 & b_2 & c_2 & & & 0 \\ & \ddots & \ddots & \ddots & & \vdots \\ & & a_{m-3} & b_{m-2} & c_{m-2} & 0 \\ & & & a_{m-2} & b_{m-1} & c_{m-1} \\ \hline c_m & 0 & \cdots & 0 & a_{m-1} & b_m \end{array}\right). \tag{4.4.29}$$

Here T is tridiagonal except for its last row and last column, where an extra nonzero element occurs. Such systems, called **arrowhead system**, can be solved with about twice the work of a tridiagonal system; see further Chapter 7.

In some applications one wants to smoothly interpolate given points (x_j, y_j), $j = 0 : m$, where a representation of the form $y = f(x)$ is not suitable. Then we can use a **parametric spline** representation $x = x(\theta)$, $y = y(\theta)$, where the parameter values $0 = \theta_0 \le \theta_1 \le \cdots \le \theta_m$ correspond to the given points. Using the approach described previously two spline functions $s_x(t)$ and $s_y(t)$ can then be determined, that interpolate the points (θ_i, x_i) and (θ_i, y_i), $i = 0 : m$, respectively. The parametrization is usually chosen as $\theta_i = d_i/d$, $i = 1 : m$, where $d_0 = 0$,

$$d_i = d_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}, \quad i = 1 : m.$$

are the the cumulative distance and $d = \sum_{j=1}^{m} d_j$.

For boundary conditions we have the same choices as mentioned previously. In particular, using periodic boundary conditions for $s_x(t)$ and $s_y(t)$ allows the representation of *closed curves* (see Problem 5).

We will now derive estimates of the error in cubic spline interpolation of a function with good smoothness properties, $f \in C^5$ (say). Let $x \in I_i = [x_{i-1}, x_i]$, and set

$$t = (x - x_{i-1})/h_i, \quad y_i = f(x_i), \quad y_i' = f'(x_i).$$

The error can be expressed as the sum of two components:

i. The error $E_H(x)$ due to Hermite interpolation with correct values of $f'(x_{i-1})$, $f'(x_i)$.

ii. The error $E_S(x)$ due to the errors of the slopes $e_i = k_i - y_i'$, $i = 0 : m$.

We shall see that the first part is typically the dominant part. For the error $E_H(x)$ we have from equations (4.4.16)–(4.4.17)

$$\max_{x \in I_i} |E_H(x)| \le \frac{1}{384} \max_{x \in I_i} |h_i^4 f^{(iv)}(x)|, \tag{4.4.30}$$

By (4.4.12) the second part of the error is

$$E_S(x) = h_i t(1-t)\big[e_{i-1}(1-t) - e_i t\big], \quad x = x_{i-1} + t h_i, \quad t \in [0, 1).$$

Since $|1 - t| + |t| = 1$, it follows easily that

$$|E_S(x)| \le \frac{1}{4} \max_{1 \le i \le m} |h_i e_j|, \quad j = i - 1, i. \tag{4.4.31}$$

We shall estimate $|e_j|$ in the case of *constant step size*. Set

$$l_i = 3(d_i + d_{i+1}) - (y'_{i-1} + 4y'_i + y'_{i+1}), \qquad i = 1 : m-1.$$

Then by (4.4.18) $(e_1, \ldots, e_{m-1})$ satisfies

$$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & 4 & 1 \\ & & & 1 & 4 \end{pmatrix} \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_{m-2} \\ e_{m-1} \end{pmatrix} = \begin{pmatrix} l_1 \\ l_2 \\ \vdots \\ l_{m-2} \\ l_{m-1} \end{pmatrix} - \begin{pmatrix} e_0 \\ 0 \\ \vdots \\ 0 \\ e_m \end{pmatrix},$$

or $Ae = l - b$. We write $e = e_I - e_B$, where $Ae_I = l$, $Ae_B = b$. These two systems will be treated differently.

We first estimate e_I and note that, since the matrix A is diagonally dominant, we can use Lemma 6.4.1 to obtain[18]

$$\max_{1\le i<m} |e_{I,i}| \le \frac{1}{\alpha} \max_{1\le i<m} |l_i|, \quad \text{where} \quad \alpha = \min_i \left(|a_{ii}| - \sum_{j\ne i} |a_{ij}| \right) = 2.$$

In order to estimate $\max_{1\le i<m} |l_i|$, note that the defining relation for l_i can be rewritten as

$$\frac{h}{3} l_i = y_{i+1} - y_{i-1} - \frac{h}{3}(y'_{i-1} + 4y'_i + y'_{i+1}).$$

The right hand side here equals the *local error of Simpson's formula* for computing the integral of y' over the interval $[x_{i-1}, x_{i+1}]$, which according to Problem 3.2.12 approximately is $h^5 f^{(5)}(x_i)/90$. It follows that[19]

$$\max_{1\le i<m} |e_{I,i}| \le \frac{1}{60} \max_i h_i^4 |f^{(5)}(x_i)|.$$

By (4.4.31) this shows that the contribution of e_I to E_S is $O(h^5)$ if $f \in C^5$, while $E_H(x)$ is $O(h^4)$. For complete splines $e_0 = e_m = 0$, and for almost complete splines $e_0 = O(h^4)$, $e_m = O(h^4)$. Hence $e_{B,i} = O(h^4)$, and its contribution to E_S is $O(h^5)$. So if h is sufficiently small, *the Hermite interpolation error is, in the whole interval* $[a, b]$*, asymptotically, the dominant source of error for complete and almost complete splines.*[20]

Similar conclusions seem to hold also in the case of variable step size, under the reasonable assumption that $h_{n+1} - h_n = O(h_n^2)$, see Sec. 13.1 (in particular Problem 11), where variable step size is discussed in the context of ordinary differential equations.

Finally we discuss the *effect of the boundary slope errors* for other boundary conditions. The equation $Ae_B = b$ can be written as a difference equation

$$e_{B,i+1} + 4e_{B,i} + e_{B,i-1} = 0, \quad i = 1 : m-1.$$

[18] This is typically an overestimate, almost by a factor of 3, see Problem 3.3.37.

[19] Notice that, if $f \in \mathcal{P}_5$, the slopes k_i becomes exact in complete cubic splines interpolation.

[20] In the literature the usual (rigorous) error bound for a perfect spline, due to Hall and Meyer, is five times as large as the bound for the Hermite error. It is valid with $h = \max h_i$, independent of the position of the knots, for all $f \in C^4$, while we require $f \in C^5$.

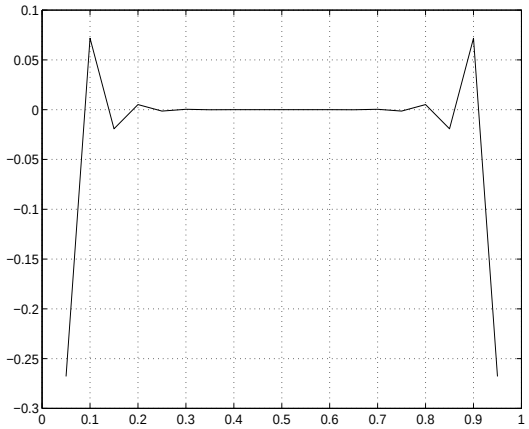

Figure 4.4.7. *Boundary slope errors $e_{B,i}$ for a cubic spline, $e_0 = e_m = -1$; $m = 20$.*

see Sec. 3.4. One can show (Problem 3.4.5) that, for any boundary condition,

$$e_{B,i} \approx u^i e_0 + u^{m-i} e_m, \quad u = \sqrt{3} - 2 \approx -0.268,$$

if u^m is negligible. (Here u and u^{-1} are the roots of the characteristic equation $u^2 + 4u + 1 = 0$.)

Figure 4.4.7 shows (for $m = 20$, $e_0 = e_m = -1$) how rapidly this error component dies out, e.g., $u^4 = 0.005$. At the midpoint $x = 0.5$ the error is $0.3816 \cdot 10^{-5}$.

If $m \gg 1$, $e_0 \neq 0$, and $e_m \neq 0$ it follows that e_B is negligible outside thin *oscillatory boundary layers* near $x = x_0$ and $x = x_m$. The height and thickness of the layers depend on e_0 and e_m. We discuss the left boundary; the right one is analogous. Assume that

$$e_0 = e_{B,0} \neq 0, \qquad e_1 \approx e_{B,1} \approx u e_{B,0} \approx u e_0.$$

We then estimate e_0 by putting

$$k_0 = y_0' + e_0, \qquad k_1 = y_1' + e_1 \approx y_1' + u e_0,$$

into the boundary condition at x_0, i.e. the first equation of (4.4.23) for the natural splines and (4.4.24) for the "not a knot" splines. (Complete splines have no oscillatory boundary layers; $e_B = 0$.) The peak of the contribution of e_B to the spline interpolation error is then obtained by (4.4.12) for $i = 1$, and equals

$$h e_0 \max_{0 \le t \le 1} |t(1 - t(1 - t - ut)| \approx 0.17 h e_0. \tag{4.4.32}$$

For the *natural splines*, this procedure leads to

$$(2 + u)e_0 = 3\frac{1}{h}(y_1 - y_0) - 2y_0' - y_1' - O(h^4)$$

$$= 3\Big(y_0' + \frac{1}{2}hy_0'' + \dots\Big) - 3y_0' - hy_0'' + \dots \sim \frac{1}{2}hy_0''.$$

Since $2 + u = \sqrt{3}$, we obtain $e_0 \approx 0.29hy_0''$, and, by (4.4.32), the peak near $x = x_0$ becomes approximately $0.049h^2|y''|$, i.e. 40% of the *linear* interpolation error (instead of cubic), often clearly visible in a graph of $s(x)$.

For the "not a knot"-splines the procedure leads to

$$(1+2u)e_0 = \frac{5}{2h}(y_1 - y_0) + \frac{1}{2h}(y_2 - y_1) - y_0' - 2y_1' - O(h^4) \approx \frac{h^3}{12}y^{(4)};$$

see Problem 3.2.10. We thus obtain $e_0 \sim 0.180h^3y^{(4)}$, and hence by (4.4.32) the peak near x_0 becomes $0.031h^4y^{(4)}$, typically very much smaller than we found for natural splines. Still it is about 11.5 times as large as the Hermite interpolation error, but since the oscillations die out by the factor $u = 0.29$ in each step, we conclude that *the Hermite interpolation is the dominant error source in cubic "not a knot"-spline interpolation in (say) the interval* $[a + 3h, b - 3h]$.

For natural splines the boundary layers are much thicker, because the peaks are much higher.

Example 4.4.3.

For the function $f(x) = 1/(1 + 25x^2)$, $x \in [-1, 1]$, the maximum norm of the error is 0.022, in interpolation with a natural cubic spline function at the eleven equidistant points $x_i = -1 + 0.2i$, $i = 0 : 10$. This good result contrasts sharply with the unpleasant experience near the boundaries of interpolation with a tenth-degree polynomial shown in Figure 4.2.1. An (almost) perfect cubic spline or a "not a knot"-spline gives even better results near the boundaries.

4.4.5 Computing with B-Splines

It was shown in Sec. 4.4.3 that the set of spline functions of order k, $S_{\Delta,k}$, on the grid

$$\Delta = \{a = x_0 < x_1 < \cdots < x_m = b\}$$

is a linear space of dimension $k + m - 1$. A basis was shown to be

$$\big\{1, x, \dots, x^{k-1}\big\} \cup \big\{(x - x_1)_+^{k-1}, (x - x_2)_+^{k-1}, \dots, (x - x_{m-1})_+^{k-1}\big\}, \qquad (4.4.33)$$

which is the **truncated power basis**.

Example 4.4.4. For $k = 2$ the space $S_{\Delta,k}$ consists of continuous piecewise affine (linear) functions also called linear splines. Then a basis is

$$\{1, x\} \cup \{l_1(x), \dots, l_{m-1}(x)\}, \quad l_i(x) = (x - x_i)_+.$$

Another basis for $S_{\Delta,2}$ is obtained by introducing an artificial exterior knot $x_{-1} \le x_0$. Then it is easy to see that using the functions $l_i(x)$, $i = -1 : m - 1$ every linear

spline on $[x_0, x_m]$ can also be written as

$$s(x) = \sum_{i=-1}^{m-1} c_i l_i(x).$$

The truncated power basis has several disadvantages. The basis functions are not local; e.g., the monomial basis functions $\{1, x, \ldots, x^{k-1}\}$ are nonzero on the whole interval $[a, b]$. Also the basis functions (4.4.33) are almost linearly dependent when the knots are close. Therefore this basis yields an ill-conditioned linear systems for various tasks and is not suited for numerical computations. In the following we will construct a more satisfactory basis for $S_{\Delta,k}$.

In anticipation of the fact that it may be desirable to interpolate at other points than the knots we consider from now on the sequence of knots

$$\Delta = \{\tau_0 \le \tau_1 \le \cdots \le \tau_m\}. \tag{4.4.34}$$

where $\tau_i \le \tau_{i+k}$, $i = 0 : m - k$, i.e. at most k successive knots are allowed to coincide.

We start by considering $k = 1$. The space $S_{\Delta,1}$ consists of piecewise constant functions. As a basis for $S_{\Delta,2}$ we can simply take the functions

$$N_{i,1}(x) = \begin{cases} 1 & x \in [\tau_i, \tau_{i+1}); \\ 0 & \text{otherwise.} \end{cases}, \quad i = 0 : m - 1. \tag{4.4.35}$$

The functions $N_{i,1}(x)$ are arbitrarily chosen to be continuous from the right.

For $k = 2$ we define the **hat functions**[21] by

$$N_{i,2}(x) = \begin{cases} (x - \tau_i)/(\tau_{i+1} - \tau_i), & x \in [\tau_i, \tau_{i+1}], \\ (\tau_{i+2} - x)/(\tau_{i+2} - \tau_{i+1}), & x \in [\tau_{i+1}, \tau_{i+2}), \\ 0, & x \notin (\tau_i, \tau_{i+2}), \end{cases} \quad i = -1 : m - 1. \tag{4.4.36}$$

where we have introduced two **exterior knots** $\tau_{-1} \le \tau_0$ and $\tau_{m+1} \ge \tau_m$ at the boundaries. (In the following we refer to the knots $\tau_0, \ldots, \tau_m$ as **interior knots**.) Note that for $x \in (\tau_i, \tau_{i+1})$ we have $N_{j,2}(x) = 0$, $j \ne i-1, i$. Hence, for a fixed value of x *at most two* hat functions will be nonzero. The exterior knots are usually taken to coincide with the boundary so that $\tau_{-1} = \tau_0$ and $\tau_{m+1} = \tau_m$; see Figure 4.4.8. In this case $N_{-1,1}$ and $N_{m-1,1}$ become "half-hats" with a singularity at τ_0 and τ_m, respectively.

The $(m + 1)$ functions $N_{i,2}(x)$, $i = -1 : m - 1$, are **B-splines** of order two (degree one). At a distinct knot τ_i just one hat function is nonzero, $N_{i+1}(x) = 1$. It follows that the spline function of order $k = 2$ interpolating the points (τ_i, y_i), $i = 0 : m$, can uniquely be written as

$$s(x) = \sum_{i=-1}^{m-1} c_i N_{i,2}(x). \tag{4.4.37}$$

[21] The generalization of hat function to two dimensions is often called tent function. This concept is very important in, e.g., in finite element methods;, see Chap. 14.

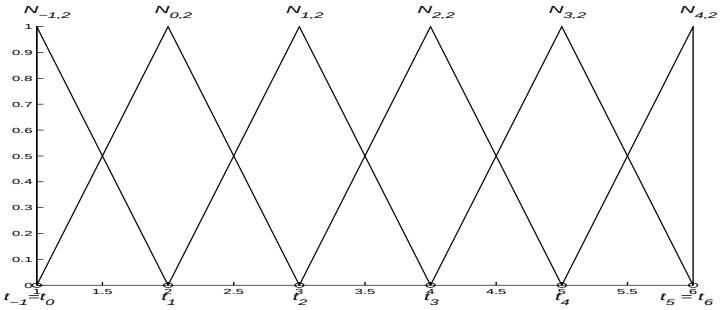

Figure 4.4.8. *The six hat functions $N_{i,2}(x)$, $i = -1 : 4$ $(m + k - 1 = 6)$ with knots $t_{-1} = t_0, t_1, \ldots, t_4, t_5 = t_6$.*

with $c_i = y_{i+1}$. This shows that the restriction of the functions $N_{i,2}(x)$, $i = -1 : m - 1$, to the interval $[\tau_0, \tau_m]$ are $(m + 1)$ linearly independent functions in $S_{\Delta,2}$ and form a basis for $S_{\Delta,2}$.

If we allow two interior knots coalesce, $\tau_i = \tau_{i+1}$, $0 < i < m - 1$, then $N_{i-1,2}(x)$ and $N_{i,2}(x)$ will have a discontinuity at τ_i. This generalizes the concept of a B-spline of order 2 given in Definition 4.4.4 and allows us to model functions with discontinuities at certain knots.

It is easily verified that the functions $N_{i,2}(x)$ can be written as a linear combination of the basis function

$$l_i(x) = (x - \tau_i)_+, \quad i = 1 : m + 1,$$

and it holds that

$$\begin{aligned} N_{i,2}(x) &= \big((x - \tau_{i+2})_+ - (x - \tau_{i+1})_+\big)/(\tau_{i+2} - \tau_{i+1}) \\ &\quad -\big((x - \tau_{i+1})_+ - (x - \tau_i)_+\big)/(\tau_{i+1} - \tau_i) \\ &= \big([\tau_{i+1}, \tau_{i+2}]_t(t - x)_+ - [\tau_i, \tau_{i+1}]_t(t - x)_+ \\ &= (\tau_{i+2} - \tau_i)[\tau_i, \tau_{i+1}, \tau_{i+2}]_t(t - x)_+, \quad i = 1 : m. \end{aligned} \tag{4.4.38}$$

Here $[\tau_i, \tau_{i+1}, \tau_{i+2}]_t$ means the second order divided difference functional[22] operating on a function of t, i.e. the values τ_i etc. are to be substituted for t not for x. Recall that divided differences are defined also for *coincident values* of the argument; see Sec. 4.3.1.

From the definition of the Peano kernel and its basic properties, given in Sec. 3.2.3 it follows that the last expression in (4.4.38) tells us that $N_{i,2}$ is the Peano kernel of a second order divided difference functional multiplied by the constant $\tau_{i+2} - \tau_i$. This observation suggests a definition of B-splines of arbitrary order k and a B-spline basis for the space $S_{\Delta,k}$.

Definition 4.4.7.

[22]The notation is defined in Sec. 4.2.1

Let $\Delta = \{\tau_0 \le \tau_1 \le \cdots \le \tau_m\}$ be an arbitrary sequence of knots such that $\tau_i < \tau_{i+k}$, $i = 0 : m-k$. Then a B-spline of order k equals (apart from a stepsize factor) the Peano kernel of a k-th order divided difference functional; more precisely we define (with the notations used in this chapter)

$$N_{i,k}(x) = (\tau_{i+k} - \tau_i)[\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}]_t (t-x)_+^{k-1}, \tag{4.4.39}$$

where $[\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}]l_x^{k-1}$ denotes the k-th divided difference of the function $l_x^{k-1}(\cdot)$ with respect to the set of points $\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}$.

Since divided differences are defined also for coalescing points (see Sec. 4.4.3), Definition 4.4.7 remains valid for knots that are not distinct.

Example 4.4.5. For $k = 1$ (4.4.38) gives ($\tau_i \ne \tau_{i+1}$)

$$N_{i,1}(x) = (\tau_{i+1} - \tau_i)[\tau_i, \tau_{i+1}]_t (t-x)_+^0.$$

If $\tau_i < x < \tau_{i+1}$, then $(\tau_{i+1} - x)_+^0 = 1$ and $(\tau_i - x)_+^0 = 0$ and hence $N_{i,1} = 1$; otherwise $N_{i,1} = 0$. This coincides with the piecewise constant functions in (4.4.35).

It can be shown that $N_{i,k}(x)$ is defined for all x and is a linear combination of functions $(\tau_j - x)_+^{k-1}$. If the knots are distinct then by Problem 4.2.11,

$$N_{i,k}(x) = (\tau_{i+k} - \tau_i) \sum_{j=i}^{i+k} \frac{(\tau_j - x)_+^{k-1}}{\Phi'_{i,k}(\tau_j)}, \quad \Phi_{i,k}(x) = \prod_{j=i}^{i+k} (x - \tau_j). \tag{4.4.40}$$

This shows that $N_{i,k}$ is a linear combination of functions $(\tau_j - x)_+^{k-1}$, $j = i : i+k$, and thus a spline of order k (as anticipated in the terminology).

The B-spline for equidistant knots is related to the probability density of the sum of k uniformly distributed random variables on $[-\frac{1}{2}, \frac{1}{2}]$. This was known already to Laplace.[23]

Theorem 4.4.8. *The B-splines of order k has the following properties:*

(i) Positivity: $N_{i,k}(x) > 0, \quad x \in (\tau_i, \tau_{i+k})$.

(ii) Compact support: $N_{i,k}(x) = 0, \quad x \notin [\tau_i, \tau_{i+k}]$.

(iii) Summation property: $\sum_i N_{i,k}(x) = 1, \quad \forall x \in [\tau_0, \tau_m]$.

Proof. A proof can be based on the general facts concerning Peano kernels found in Sec. 3.2.3, where also an expression for the B-spline ($k = 3$) is calculated for the equidistant case. (Unfortunately the symbol x means opposite things here and in Sec. 3.2.3.)

[23] Pierre Simon Laplace (1749–1827), French mathematician and astronomer, has also given important contributions to mathematical physics and probability theory.

(i) By (4.2.11) $Rf = [\tau_i, \tau_{i+1}, \ldots, \tau_{i+k}]f = f^{(k)}(\xi)/k!$, $\xi \in (\tau_i, \tau_{i+k})$, and $Rp = 0$, for $p \in \mathcal{P}_k$. It then follows from the corollary of Peano's remainder theorem that the Peano kernel does not change sign in $[\tau_i, \tau_{i+k}]$. It must then have the same sign as $\int K(u)\,du = R(x-a)^k/k! = 1$. This proves a somewhat weaker statement than (i) ($N_{i,k}(x) \geq 0$ instead of $N_{i,k}(x) > 0$).

(ii) This property follows since a Peano kernel always vanishes outside its interval of support of the functional; in this case $[\tau_i, \tau_{i+k}]$. (A more general result concerning the number of zeros is found, e.g., in Powell [38, Theorem 19.1]. Among other things this theorem implies that the jth derivative of a B-spline, $j \leq k-2$, changes sign exactly j times. This explains the "bell-shape" of B-splines.)

(iii) For a sketch of a proof of the summation property [24], see Problem 8. □

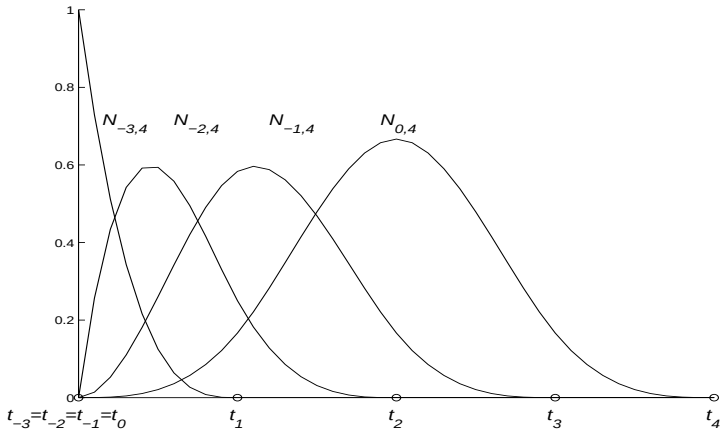

Figure 4.4.9. *The four cubic B-splines nonzero for $x \in (t_0, t_1)$ with coalescing exterior knots $t_{-3} = t_{-2} = t_{-1} = t_0$.*

To get a basis of B-splines for the space $S_{\Delta,k}$, $\Delta = \{\tau_0 \leq \tau_1 \leq \cdots \leq \tau_m\}$, $(m+k-1)$ B-splines of order k are needed. We therefore choose $2(k-1)$ additional knots $\tau_{-k+1} \leq \cdots \leq \tau_{-1} \leq \tau_0$, and $\tau_{m+k-1} \geq \cdots \geq \tau_{m+1} \geq \tau_m$, and B-splines $N_{i,k}(x)$, $i = -k+1 : m-1$.

It is convenient to let the exterior knots coincide with the end points,

$$\tau_{-k+1} = \cdots = \tau_{-1} = \tau_0, \qquad \tau_m = \tau_{m+1} = \cdots = \tau_{m+k-1}.$$

It can be shown that this choice tends to optimize the conditioning of the B-spline basis. Figure 4.4.9 shows the first four cubic B-splines for $k = 4$ (the four last B-splines are a mirror image of these). We note that $N_{-3,4}$ is discontinuous, $N_{-2,4}$ has a non-zero first derivative, and $N_{-2,4}$ a non-zero second derivative at the left boundary.

[24] The B-splines $M_{i,k}$ originally introduced by Curry and Schoenberg were normalized so that $\int_{-\infty}^{\infty} M_{i,k}\,dx = 1$.

Interior knots of multiplicity $r > 1$ are useful when we want to model a function, which has less than $k-2$ continuous derivatives at a particular knot. If $r \leq k$ interior knots coalesce then the spline will only have $k-1-r$ continuous derivatives at this knot.

Lemma 4.4.9. *Let τ_i be a knot of multiplicity $r \leq k$, i.e.*

$$\tau_{i-1} < \tau_i = \cdots = \tau_{i+r-1} < \tau_{i+r}.$$

Then $N_{i,k}$ is at least $(k-r-1)$ times continuously differentiable at τ_i. For $r = k$, the B-spline becomes discontinuous.

Proof. The truncated power $(t-\tau_i)_+^{k-1}$ is $(k-2)$ times continuously differentiable and $[\tau_i, \ldots, \tau_{i+k}]g$ contains at most the $(r-1)$st derivative of g. Hence the lemma follows. □

Consider the spline function

$$s(x) = \sum_{i=-k+1}^{m-1} c_i N_{i,k}(x). \tag{4.4.41}$$

If $s(x) = 0$, $x \in [\tau_0, \tau_m]$, then $s(\tau_0) = s'(\tau_0) = \cdots = s^{(k-1)}(\tau_0) = 0$, and $s(\tau_i) = 0$, $i = 1 : m-1$. From this it can be deduced by induction that in (4.4.41) $c_i = 0$, $i = -k+1 : m-1$. This shows that the $(m+k-1)$ B-splines $N_{i,k}(x)$, $i = -k+1 : m-1$, are linearly independent and form a basis for the space $S_{\Delta,k}$. (A more general result is given in de Boor [7, Theorem IX.1].) Thus any spline function $s(x)$ of order k (degree $k-1$) on Δ can be uniquely written in the form (4.4.41). Note that from the compact support property it follows that for any fixed value of $x \in [\tau_0, \tau_m]$ at most k terms will be nonzero in the sum in (4.4.41), so we have

$$s(x) = \sum_{i=j-k+1}^{j} c_i N_{i,k}(x), \qquad x \in [\tau_j, \tau_{j+1}). \tag{4.4.42}$$

For developing a recurrence relation for B-splines we need the following difference analogue of **Leibniz'**[25] **formula**.

Theorem 4.4.10.
Let $f(x) = g(x)h(x)$, and $x_i \leq x_{i+1} \leq \ldots \leq x_{i+k}$. Then

$$[x_i, \ldots, x_{i+k}]f = \sum_{r=i}^{i+k} [x_i, \ldots, x_r]g \cdot [x_r, \ldots, x_{i+k}]h, \tag{4.4.43}$$

provided that $g(x)$ and $f(x)$ are sufficiently many times differentiable so that the divided differences on the right hand side are defined for any coinciding points x_j.

[25] Gottfried Wilhelm von Leibniz (1646–1716). Leibniz developed his version of calculus at the same time as Newton. Many of the notations he introduced are still used today.

Proof. Note that the product polynomial

$$P(x) = \sum_{r=i}^{i+k} (x - x_i) \cdots (x - x_{r-1})[x_i, \ldots, x_r]g \cdot \sum_{s=i}^{i+k} (x - x_{s+1}) \cdots (x - x_{i+k})[x_s, \ldots, x_{i+k}]h$$

agrees with $f(x)$ at $x_i, \ldots, x_{i+k}$ since by Newton's interpolation formula the first factor agrees with $g(x)$ and the second with $h(x)$ there. If we multiply out we can write $P(x)$ as a sum of two polynomials

$$P(x) = \sum_{r,s=i}^{i+k} \ldots = \sum_{r \le s} \ldots + \sum_{r > s} \ldots = P_1(x) + P_2(x).$$

Since in $P_2(x)$ each term in the sum has $\prod_{j=i}^{i+k}(x - x_j)$ as a factor it follows that $P_1(x)$ will also interpolate $f(x)$ at $x_i, \ldots, x_{i+k}$. The theorem now follows since the leading coefficient of $P_1(x)$, which equals $\sum_{r=i}^{i+k}[x_i, \ldots, x_r]g \cdots [x_r, \ldots, x_{i+k}]h$, must equal the leading coefficient of the unique interpolation polynomial of degree k, which is $[x_i, \ldots, x_{i+k}]f$. □

Theorem 4.4.11. *The B-splines satisfy the recurrence relation*

$$N_{i,k}(x) = \frac{x - \tau_i}{\tau_{i+k-1} - \tau_i} N_{i,k-1}(x) + \frac{\tau_{i+k} - x}{\tau_{i+k} - \tau_{i+1}} N_{i+1,k-1}(x). \tag{4.4.44}$$

Proof. (de Boor [7, pp. 130–131]) The recurrence is derived by applying Leibniz' formula for the k-th divided difference to the product

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

This gives

$$[\tau_i, \ldots, \tau_{i+k}]_t (t - x)_+^{k-1} = (\tau_i - x)[\tau_i, \ldots, \tau_{i+k}]_t (t - x)_+^{k-2} + 1 \cdot [\tau_{i+1}, \ldots, \tau_{i+k}]_t (t - x)_+^{k-2}. \tag{4.4.45}$$

since $[\tau_i]_t(t - x) = (\tau_i - x)$, $[\tau_i, \tau_{i+1}]_t(t - x) = 1$, and $[\tau_i, \ldots, \tau_j]_t(t - x) = 0$ for $j > i + 1$. By the definition of a divided difference

$$(\tau_i - x)[\tau_i, \ldots, \tau_{i+k}]_t = \frac{\tau_i - x}{\tau_{i+k} - \tau_i} \left([\tau_{i+1}, \ldots, \tau_{i+k}]_t - [\tau_i, \ldots, \tau_{i+k-1}]_t \right).$$

Substitute this in (4.4.45), simplify and apply the definition of B-splines. This yields (4.4.44). □

Note that with k multiple knots at the boundaries the denominators in (4.4.44) can become zero. In this case the corresponding nominator also is zero and the term should be set equal to zero.

From Property (ii) in Theorem 4.4.8 we conclude that only k B-splines of order k may be nonzero on a particular interval $[\tau_j, \tau_{j+1}]$. Starting from $N_{i,1}(x) = 1$, $x \in [\tau_i, \tau_{i+1})$ and 0 otherwise, cf. (4.4.35), these B-splines of order k can be *simultaneously* evaluated using this recurrence by forming successively their values for order $1 : k$ in only about $\frac{3}{2}k^2$ flops. This recurrence is extremely stable, since it consists of taking positive (nonnegative) combinations of positive (nonnegative) numbers.

Suppose that $x \in [\tau_i, \tau_{i+1}]$, and $\tau_i \neq \tau_{i+1}$. Then the B-splines of order $k = 1, 2, 3, \ldots$, nonzero at x can be simultaneously evaluated by computing the triangular array

$$\begin{array}{ccccc}
 & & 0 & & \\
 & 0 & & & \cdots \\
 & & N_{i-3,4} & & \\
0 & & N_{i-2,3} & & \cdots \\
 & N_{i-1,2} & & N_{i-2,4} & \\
N_{i,1} & & N_{i-1,3} & & \cdots \\
 & N_{i,2} & & N_{i-1,4} & \\
0 & & N_{i,3} & & \cdots \\
 & 0 & & N_{i,4} & \\
 & & 0 & & \cdots \\
 & & & 0 &
\end{array} \tag{4.4.46}$$

The boundary of zeros in the array is due to the fact that all other B-splines not mentioned explicitly vanish at x. This array can be generated column by column. The first column is known from (4.4.35), and each entry in a subsequent column can be computed as a linear combination with nonnegative coefficients of its two neighbors using (4.4.44). Note that if this is arranged in a suitable order the elements in the new column can overwrite the elements in the old column.

To evaluate $s(x)$, we first determine the index i such that $x \in [\tau_i, \tau_{i+1})$ using, e.g., a linear search or bisection (see Sec. 6.1). The recurrence above is then used to generate the triangular array (4.4.46), which provides $N_{j,k}(x)$, $j = i - k + 1 : i$. in the sum (4.4.42).

Using the B-spline basis we can formulate a more general interpolation problem, where the $n = m + k - 1$ interpolation points, or nodes, x_j do not necessarily coincide with the knots τ_i. We consider determining a spline function $s(x) \in S_{\Delta,k}$, such that

$$s(x_j) = f_j, \quad j = 1 : m + k - 1.$$

Since any spline $s(x) \in S_{\Delta,k}$ can be written as a linear combination of B-splines, the interpolation problem can equivalently be written

$$\sum_{i=-k+1}^{m-1} c_i N_{i,k}(x_j) = f_j, \quad j = 1 : m + k - 1. \tag{4.4.47}$$

These equations form a linear system $Ac = f$ for the coefficients, where

$$a_{ij} = N_{i-k,k}(x_j), \quad i, j = 1 : m + k - 1, \tag{4.4.48}$$

and

$$c = (c_{-k+1}, \ldots, c_{m-1})^T, \qquad f = (f_1, \ldots, f_{m+k-1})^T.$$

The elements $a_{ij} = N_{i-k,k}(x_j)$ of the matrix A can be evaluated by the recurrence (4.4.44). The matrix A will have a banded structure since $a_{ij} = 0$ unless $x_j \in [\tau_i, \tau_{i+k}]$. Hence at most k elements are nonzero in each row of A. (Note that if $x_j = \tau_i$ for some i only $k-1$ elements will be nonzero, which explains why tridiagonal systems were encountered in cubic spline interpolation in earlier sections.)

Schoenberg and Whitney [45, 1953] showed that *the matrix A is nonsingular if and only if its diagonal elements are nonzero*,

$$a_{jj} = N_{j-k,k}(x_j) \neq 0, \quad j = 1 : n,$$

or equivalently if the nodes x_j satisfy

$$\tau_{j-k} < x_j < \tau_j, \quad j = 1 : n. \tag{4.4.49}$$

Further, the matrix can be shown to be **totally nonnegative**, i.e. the determinant of every submatrix is nonnegative. For such systems, if Gaussian elimination is carried out *without pivoting*, the error bound is particularly favorable. This will also preserve the banded structure of A during the elimination.

When the B-spline representation (4.4.41) of the interpolant has been determined it can be evaluated at a given point using the recursion formula (4.4.44). If it has to be evaluated at many points it is more efficient to first convert the spline to its polynomial representation (4.4.14). For hints on how to do that see Problem 9 (b) and (c).

Unless the Schoenberg–Whitney condition (4.4.49) is well-satisfied the system may become ill-conditioned. For splines of even order k the interior nodes

$$\tau_0 = x_0, \quad \tau_{j+1} = x_{j+k/2}, \quad j = 0 : n - k - 1, \quad \tau_m = x_n,$$

is a good choice in this respect. In the important case of cubic splines this means that knots are positioned at each data point except the second and next last (cf. the "not a knot" condition in Sec. 4.4.4.

In some application we are given function values $f_j = f(x_j)$, $j = 1 : n$, that we want to approximate with a spline functions with *much fewer knots* so that $m + k - 1 \leq n$. Then (4.4.47) is an *overdetermined linear system* and the interpolation conditions cannot be satisfied exactly. We therefore consider the linear **least squares spline approximation** problem

$$\min \sum_{j=1}^{n} \left(\sum_{i=-k+1}^{m-1} c_i N_{i,k}(x_j) - f_j \right)^2. \tag{4.4.50}$$

Using the same notation as above this can be written in matrix form

$$\min_c \|Ac - f\|_2^2. \tag{4.4.51}$$

The matrix A will have full column rank equal to $m + k - 1$ if and only if there is a subset of points τ_j satisfying the Schoenberg–Whitney conditions (4.4.49).

If A has full column rank then the least squares solution c is unique and is uniquely determined by the normal equations $A^T Ac = A^T f$. The matrix $A^T A$ is symmetric and positive definite and hence the normal equations can be solved using Cholesky factorization of $A^T A$. A will have at most k nonzero elements in each row and advantage should be taken of the banded form of the matrix $A^T A$; see Figure 4.4.10. More stable methods for solving linear least squares problems (4.4.51) will be introduced in Sec. 8.5.7.

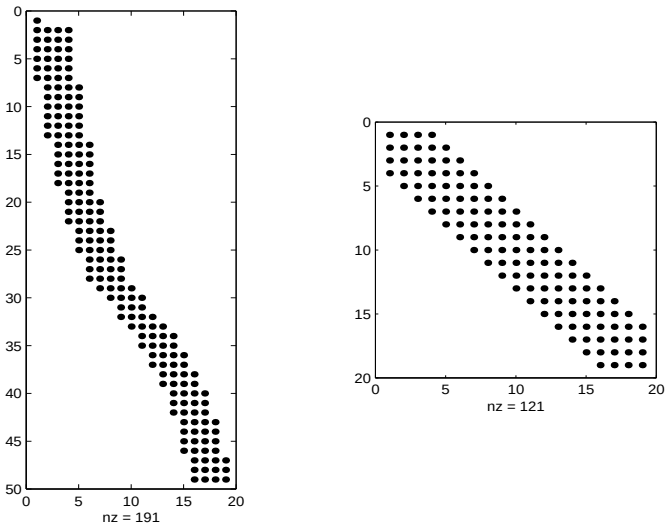

Figure 4.4.10. *Structure of the matrices A and $A^T A$ arising in cubic spline approximation of Titanium data.(nonzero elements showed).*

Example 4.4.6. (de Boor [7]) Consider experimental data describing a property of titanium as a function of temperature. Experimental values for $t_i = 585 + 10i$, $i = 1 : 49$, are given. We want to fit this data using a least squares cubic spline Figure 4.4.11 shows results from using a least squares fitted cubic spline with 9 and 17 knots, respectively. The spline with 9 knots shows oscillations near the points where the curve flattens out and the top of the peak is not well matched. Increasing the number of knots to 17 we get a very good fit.

We have in the treatment above assumed that the set of (interior) knots $\{\tau_0 \leq \tau_1 \leq \cdots \leq \tau_m\}$ is given. In many spline approximation problems it is more realistic to consider the location of knots to be free and try to determine a small set of knots such that the given data can be approximated to a some preassigned accuracy. Several schemes have been developed to treat this problem.

One class of algorithms start with only a few knots and iteratively add more

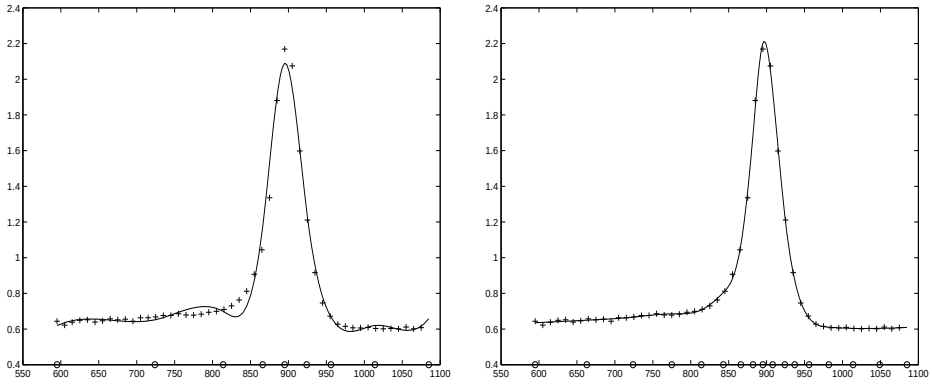

Figure 4.4.11. *Least squares cubic spline approximation of Titanium data; the knots are marked on the axes by a "o"; left: 9 knots;left: 17 knots.*

knots guided by some measure of the error; see de Boor [6, Chapter XII]. The placement of the knots are chosen so that the Schoenberg–Whitney conditions are always satisfied. The iterations are stopped when the approximation is deemed satisfactory. If a node $\tilde{\tau} \in [\tau_j, \tau_{j+1})$ is inserted then the B-spline series with respect to the enlarged set of nodes can cheaply and stably be computed from the old one (see Dierckx [22]).

Other algorithms starts with many knots and successively *remove* knots, which are not contributing much to the quality of the approximation. In these two classes of algorithms one does not seek an optimal knot placement at each step. This is done in a more recent algorithms; see Schwetlick and Schütze [47].

Review Questions

1. What is meant by a cubic spline function? Give an example where such a function is better suited than a polynomial for approximation over the whole interval.

2. (a) What is the dimension of the space $S_{\Delta,k}$ of spline functions of order k on a grid $\Delta = \{x_0, x_1, \ldots, x_m\}$? Give a basis for this space.

(b) Set up the linear system for cubic spline interpolation in the equidistant case for some common boundary conditions. What does the unknown quantities mean, and what conditions are expressed by the equations? About how many operations are required to interpolate a cubic spline function to $m+1$, $m \gg 1$, given values of a function?

3. What error sources have influence on the results of cubic spline interpolation? How fast do the boundary errors die out? How do the results in the interior of the interval depend on the step size (asymptotically)? One of the common types of boundary conditions yield much larger error than the others. Which

one? Compare it quantitatively with one of the others.

4. Approximately how many arithmetic operations are required to evaluate the function values of all cubic B-splines that are nonzero at a given point?

5. Express the restrictions of $f(x) = 1$ and $f(x) = x$ to the interval $[x_0, x_m]$ as linear combinations of the hat functions defined by (4.4.36).

6. The Schoenberg–Whitney conditions give necessary and sufficient conditions for a certain interpolation problem with B-splines of order k. What is the interpolation problem and what are the conditions?

Problems and Computer Exercises

1. Consider a cubic Bézier curve defined by the four control points p_0, p_1, p_2 and p_3. Show that at $t = 1/2$

$$c(1/2) = \frac{1}{4}\frac{p_0 + p_3}{2} + \frac{3}{4}\frac{p_1 + p_2}{2}$$

and interpret this formula geometrically.

2. (G. Eriksson) Approximate the function $y = \cos x$ on $[-\pi/2, \pi/2]$ by a cubic Bézier curve. Determine the four control points in such a way that it interpolates $\cos x$ and its derivative at $-\pi/2, 0$ and $\pi/2$.

Hint Use symmetry and the result of Problem 1 to find the y-coordinate of p_1 and p_2.

3. Suppose that $f(x)$ and the grid Δ are symmetric around the midpoint of the interval $[a, b]$. You can then considerably reduce the amount of computation needed for the construction of the cubic spline interpolant by replacing the boundary condition at $x = b$ by an adequate condition at the midpoint. Which?

(a) Set up the matrix and right hand side for this in the case of constant step size h.

(b) Do the same for a general case of variable step size.

4. (a) Write a program for solving a tridiagonal linear system by Gaussian elimination without pivoting. Assume that the nonzero diagonals are stored in three vectors. Adapt it to cubic spline interpolation with equidistant knots with several types of boundary conditions.

(b) Consider the tridiagonal system resulting from the not-a-knot boundary conditions. Show that after eliminating k_0 between the first two equations and k_m between the last two equations the remaining tridiagonal system for $k_1, \ldots, k_{m-1}$ is diagonally dominant.

(c) Interpolate a cubic spline $s(x)$ through the points $(x_i, f(x_i))$, where

$$f(x) = (1 + 25x^2)^{-1}, \quad x_i = -1 + \frac{2}{10}(i-1), \quad i = 1:11.$$

Compute a natural spline, a complete spline (here $f'(x_1)$ and $f'(x_{11})$ are

needed) and a "not a knot" spline. Compute and compare error curves (natural and logarithmic).

(c) Similar runs as in (b), though for $f(x) = 1/x$, $1 \le x \le 2$, with $h = 0.1$ and $h = 0.05$. Compare the "almost complete", as described in the text, with the complete and the natural boundary condition.

5. If f'' is known at the boundary points, then the boundary conditions can be chosen so that $f'' = s''$ at the boundary points. Show that this leads to the conditions

$$2k_0 + k_1 = 3d_1 - h_1 f''(x_0),$$
$$k_{m-1} + 2k_m = 3d_m + h_m f''(x_m).$$

6. Show that the formula

$$\int_{x_0}^{x_m} s(x)dx = \sum_{i=1}^{m} \Big(\frac{1}{2} h_i (y_{i-1} + y_i) + \frac{1}{12} (k_{i-1} - k_i) h_i^2 \Big),$$

is exact for all cubic spline functions $s(x)$. How does the formula simplify if all $h_i = h$?

Hint: Integrate (4.4.12) from x_{i-1} to x_i.

7. In (4.4.12) the cubic spline $q_i(x)$ on the interval $[x_{i-1}, x_i)$ is expressed in terms of function values y_{i-1}, y_i, and the first derivatives k_{i-1}, k_i.

(a) Show that if $M_i = s''(x_i)$, $i = 0 : m$, are the *second derivatives* (also called **moments**) of the spline function then

$$k_i - d_i = \frac{h_i}{6}(2M_i + M_{i-1}), \qquad k_{i-1} - d_i = -\frac{h_i}{6}(M_i + 2M_{i-1}).$$

Hence $q_i(x)$ can also be uniquely expressed in terms of y_{i-1}, y_i and M_{i-1}, M_i.

(b) Show that, using the parametrization in (a), the continuity of the *first* derivative of the spline function at an interior point x_i gives the equation

$$h_i M_{i-1} + 2(h_i + h_{i+1})M_i + h_{i+1} M_{i+1} = 6(d_{i+1} - d_i).$$

8. (a) Develop an algorithm for solving the arrowhead linear system $Tk = g$ (4.4.29), using Gaussian elimination without pivoting. Show that about twice the number of arithmetic operations are needed compared to a tridiagonal system.

(b) At the end of Sec. 4.4.4 parametric spline interpolation to given points (x_i, y_i), $i = 0 : m$, is briefly mentioned. Work out the details on how to use this to represent a closed curve. Try it out on a boomerang, an elephant, or what have you?

9. (a) Compute and plot a B-spline basis of order $k = 3$ (locally quadratic) and $m = 6$ subintervals of equal length.

Hint: In the equidistant case there is some translation invariance and symmetry, so you do not really need more than essentially three different B-splines.

You need one spline with triple knot at x_0 and a single knot at x_1 (very easy to construct), and two more splines.

(b) Set up a scheme to determine a locally quadratic B-spline, which interpolates given values at the *midpoints* $x_i = (\tau_{i+1}+\tau_i)/2$ ($\tau_{i+1} \neq \tau_i$), $i = 0 : m-1$, and the boundary points τ_0, τ_m. Show that the spline is uniquely determined by these interpolation conditions.

10. Derive the usual formula of Leibniz for the kth derivative from (4.4.43) by a passage to the limit.

11. Use the recurrence (4.4.44)

$$N_{i,k}(x) = \frac{x-\tau_i}{\tau_{i+k-1}-\tau_i} N_{i,k-1}(x) + \frac{\tau_{i+k}-x}{\tau_{i+k}-\tau_{i+1}} N_{i+1,k-1}(x)$$

to show that

$$\sum_i N_{i,k}(x) = \sum_i N_{i,k-1}(x), \quad \tau_0 \le x \le \tau_m,$$

where the sum is taken over all nonzero values. Use this to give an induction proof of the summation property in Theorem 4.4.8.

12. (a) Using the result $\dfrac{d}{dx}(t-x)_+^{k-1} = -(k-1)(t-k)_+^{k-2}$, $k \ge 1$, show the formula for differentiating a B-spline

$$\frac{d}{dx} N_{i,k}(x) = (k-1)\left(\frac{N_{i,k-1}(x)}{\tau_{i+k-1}-\tau_i} - \frac{N_{i+1,k-1}(x)}{\tau_{i+k}-\tau_{i+1}} \right).$$

Then use the relation (4.4.44) to show

$$\frac{d}{dx} \sum_{i=r}^{s} c_i N_{i,k}(x) = (k-1) \sum_{i=r}^{s+1} \frac{c_i - c_{i-1}}{\tau_{i+k-1}-\tau_i} N_{i,k-1}(x),$$

where $c_{r-1} := c_{s+1} := 0$.

(b) Given the B-spline representation of a cubic spline function $s(x)$. Show how to find its polynomial representation (4.4.14) by computing the function values and first derivatives $s(\tau_i), s'(\tau_i)$, $i = 0 : m$.

(c) Apply the idea in (a) recursively to show how to compute all derivatives of $s(x)$ up to order $k-1$. Use this to develop a method for computing the polynomial representation of a spline of arbitrary order k from its B-spline representation.

13. Three different bases for the space of cubic polynomials of degree ≤ 3 on the interval $[0,1]$ are the monomial basis $\{1, t, t^2, t^3\}$, the Bernstein basis $\{B_0^3(t), B_1^3(t), B_2^3(t), B_3^3(t)\}$, and the Hermite basis. Determine the matrices for these basis changes.

4.5 Approximation and Function Spaces

Function space concepts have been introduced successively in this book. Recall, e.g., the discussion of operators and functionals in Sec. 3.2.1, where also the linear space

$\mathcal{P}_n$, the n-dimensional space of polynomials of degree less than n was introduced. This terminology was used and extended in Sec. 4.1, in the discussion of various bases and coordinate transformations in $\mathcal{P}_n$.

For coming applications of functional analysis to interpolation and approximation it is now time for a digression about:

- distances and norms in function spaces;
- a general error bound that we call *the norm and distance formula*;
- inner-product spaces and orthogonal systems.

4.5.1 Distance and Norm

For the study of accuracy and convergence of methods of interpolation and approximation we now introduce the concept of a **metric space**. By this we understand a set of points $\mathcal{S}$, and a real-valued function d, a **distance** defined for pairs of points in $\mathcal{S}$ in such a way that the following axioms are satisfied for all x, y, z in $\mathcal{S}$. (Draw a triangle with vertices at the points x, y, z.)

1. $d(x, x) = 0$, (reflexivity)
2. $d(x, y) > 0$ if $x \neq y$, (posivity)
3. $d(x, y) = d(y, x)$, (symmetry)
4. $d(x, y) \leq d(x, z) + d(z, y)$ (triangle inequality).

The axioms reflect familiar features of distance concepts used in mathematics and everyday life, such as the absolute value of complex numbers, the shortest distance along a geodesic on the surface of the earth, or the shortest distance along a given road system.[26].

Many other natural and useful relations can be derived from these axioms, e.g.

$$d(x, y) \geq |d(x, z) - d(y, z)|, \qquad d(x_1, x_n) \leq \sum_{i=1}^{n-1} d(x_i, x_{i+1}), \tag{4.5.1}$$

where $x_1, x_2, \ldots, x_n$ is a sequence of points in $\mathcal{S}$; see Problem 1.

Definition 4.5.1.

A sequence of points $\{x_n\}$ in a metric space $\mathcal{S}$ is said to **converge** *to a limit $x^* \in \mathcal{S}$ if $d(x_n, x^*) \to 0$. As $n \to \infty$, we write $x_n \to x^*$ or $\lim_{n\to\infty} x_n = x^*$. A sequence $\{x_n\}$ in $\mathcal{S}$ is called a* **Cauchy sequence**, *if for every $\epsilon > 0$, there is an integer $N(\epsilon)$ such that $d(x_m, x_n) < \epsilon$, for all $m, n \geq N(\epsilon)$. Every convergent sequence is a Cauchy sequence, but the converse is not necessarily true. $\mathcal{S}$ is called a* **complete space** *if every Cauchy sequence in $\mathcal{S}$ converges to a limit in $\mathcal{S}$.*

[26] If $\mathcal{S}$ is a functions space, the points of $\mathcal{S}$ are functions with operands in some other space, e.g., in $\mathbf{R}$ or $\mathbf{R}^n$

It is well known that $\mathbf{R}$ satisfies the characterization of a complete space, but the set of *rational* numbers is not complete. For example, the iteration $x_1 = 1$, $x_{n+1} = \frac{1}{2}(x_n + 2/x_n)$, studied in Example 1.2.1, defines a sequence of rational numbers that converges to $\sqrt{2}$, which is not a rational number.

The distance of a point $x \in \mathcal{S}$ from a subset (subspace) $\mathcal{S}' \subset \mathcal{S}$ is defined by[27]

$$dist\,(x, \mathcal{S}') = \inf_{f \in \mathcal{S}'} d(f, x). \tag{4.5.2}$$

Many important problems in Pure and Applied Mathematics can be formulated as minimization problems. The function space terminology often makes proofs and algorithms less abstract.

Most spaces that we shall encounter in this book are **linear spaces**. Their elements are called vectors, why these spaces also are called **vector spaces** Two operations are defined in these spaces, namely the addition of vectors and the multiplication of a vector by a scalar. They obey the usual rules of algebra.[28] The set of scalars can be either $\mathbf{R}$ or $\mathbf{C}$; the vector space is then called *real* or *complex*, respectively.

We shall be concerned with the problem of **linear approximation**, i.e. a function f is to be approximated using a function f^* that can be expressed as a linear combination

$$f^* = c_1\phi_1(x) + c_2\phi_2(x) + \cdots + c_n\phi_n(x), \tag{4.5.3}$$

of n given linearly independent functions $\phi_1(x), \phi_2(x), \ldots, \phi_n(x)$, where $c_1, c_2, \ldots, c_n$ are parameters to be determined.[29] They may be considered as coordinates of f^* in the functions space spanned by $\phi_1(x), \phi_2(x), \ldots, \phi_n(x)$.

In a vector space the distance of the point f from the origin is called the **norm** of f and denoted by $\|f\|$, typically with some subscript that more or less cryptically indicates the relevant space. The definition of the norm depends namely on the space. The following axioms must be satisfied.

Definition 4.5.2.

A real valued function $\|f\|$ is called a norm on a vector space $\mathcal{S}$, if it satisfies the conditions 1–3 below for all $f, g \in \mathcal{S}$, and for all scalars λ

1. $\|f\| > 0$, *unless* $f = 0$, *(positivity)*
2. $\|\lambda f\| = |\lambda|\|f\|$, *(homogeneity)*
3. $\|f + g\| \leq \|f\| + \|g\|$ *(triangle inequality)*

[27] $\inf x$ denotes the infimum of x, i.e. the greatest lower bound of x. Similarly $\sup x$ is short for supremum, i.e. the least upper bound.

[28] See Appendix A.1 for a summary about vector spaces. In larger texts on linear algebra or functional analysis you find a collection of eight axioms (commutativity, associativity, etc.) required by a linear vector space.

[29] The functions ϕ_j, however, are typically *not* linear. The term "linear interpolation" is from our present point of view rather misleading.

A normed vector space is a metric space; the distance reads

$$d(x, y) = \|x - y\|.$$

If it is also a complete space, it is called a **Banach space**.[30]

The most common norms in spaces of (real and complex) infinite sequences $x = (\xi_1, \xi_2, \xi_3, \ldots)^T$ or spaces of functions on a bounded and closed interval $[a, b]$ and in the spaces $\mathbf{R}^n$ and $\mathbf{C}^n$ are

$$\|x\|_\infty = \max_j |\xi_j|, \qquad \|f\|_\infty = \max_{x\in[a,b]} |f(x)|,$$

$$\|x\|_2 = \Big(\sum_{j=1}^{\infty} |\xi_j|^2\Big)^{1/2}, \qquad \|f\|_2 = \|f\|_{2,[a,b]} = \Big(\int_a^b |f(x)|^2\,dx\Big)^{1/2},$$

$$\|x\|_{2,\omega} = \Big(\sum_{j=1}^{\infty} \omega_j|\xi_j|^2\Big)^{1/2}, \qquad \|f\|_{2,\omega} = \Big(\int_a^b |f(x)|^2\omega(x)\,dx\Big)^{1/2},$$

These norms are called

- the **max(imum) norm** (or the uniform norm);
- the **Euclidean norm** (or the L_2 norm for integrals and l_2 norm for coordinate sequences);
- the **weighted Euclidean norm**. Here $\omega(x)$ is a weight function, assumed to be continuous and strictly positive on the open interval (a, b).

We assume that the integrals

$$\int_a^b |x|^k\omega(x)\,dx$$

exist for all k. Integrable singularities at the end points are permitted; an important example is $\omega(x) = (1 - x^2)^{-1/2}$ on the interval $[-1, 1]$.

The above norms are special cases or limiting cases ($p \to \infty$ gives the max norm) of the l_p or L_p norms and weighted variants of these. They are defined for $p \geq 1$, as follows[31]

$$\|x\|_p = \Big(\sum_{j=1}^{\infty} |\xi_j|^p\Big)^{1/p}, \qquad \|f\|_p = \Big(\int_a^b |f(x)|^p\,dx\Big)^{1/p}. \tag{4.5.4}$$

(The sum in the l_p norm has a finite number of terms, if the space is finite dimensional.)

[30] Stefan Banach (1892–1945) Polish mathematician. professor at the University in Lvov. Banach founded modern Functional Analysis and gave major contributions to the theory of topological vector spaces, measure theory and related topics. In 1939 he was elected President of the Polish Mathematical Society.

[31] The triangle inequality for $\|x\|_p$ is derived from two classical inequalities due to Hölder and Minkowski. Elegant proofs of these are presented, e.g., in Hairer and Wanner [28, p. 327].

From the minimax property of Chebyshev polynomials (Lemma 3.2.3) it follows that the best approximation in the maximum norm to the function $f(x) = x^n$ on $[-1, 1]$ by a polynomial of lower degree is given by $x^n - 2^{1-n}T_n(x)$. The error assumes its extrema in a sequence of $n + 1$ points $x_i = \cos(i\pi/n)$. The sign of the error alternates at these points.

The above property is generalized in the following theorem, which we give without proof. It is the basis of many algorithm for computing approximations in the maximum norm.

Theorem 4.5.3.

Let f be a continuous function on $[a, b]$ and let $\hat{p}$ be the nth degree polynomial which best approximates f in the maximum norm. Then $\hat{p}$ is characterized by the fact that there exists at least $n + 2$ points

$$a \le \zeta_0 < \zeta_1 < \zeta_2 < \cdots < \zeta_{n+1} \le b,$$

where the error $r = \hat{p} - f$ takes on its maximal magnitude with alternating signs; i.e. $|r(\zeta_i)| = \|r\|_\infty$ and

$$r(\zeta_{i+1}) = -r(\zeta_i), \quad i = 0 : n.$$

This characterization constitutes both a necessary and sufficient condition. If $f^{(n+1)}(x)$ has constant sign in $[a, b]$ then $\zeta_0 = a$, $\zeta_{n+1} = b$.

Convergence in a space, equipped with the max norm, means **uniform convergence**. Therefore, *the completeness of the space $C[a, b]$ follows from a classical theorem of Analysis* that tells that the limit of a uniformly convergent sequence is a continuous function. The generalization of this theorem to several dimensions implies the completeness of the space of continuous functions, equipped with the max norm on a closed bounded region in $\mathbf{R}^n$.

Other classes of functions can be normed with the max norm $\max_{x\in[a,b]} |f(x)|$, e.g., $C^1[a, b]$, but this space is not complete; one can subtract a sequence of functions in this space with a limit that is not continuous, but one can often live well with incompleteness.

The notation L_2 norm comes from the function space $L_2[a, b]$, which is the class of functions for which the integral $\int_a^b |f(x)|^2\, dx$ exists, in the sense of Lebesgue;[32] the Lebesgue integral was needed in order to make the space complete. No knowledge of Lebesgue integration is needed for the study of this book, but this particular fact can be interesting as a background. One can apply this norm also to the (smaller) class of continuous functions on $[a, b]$. In this case the Riemannn[33] integral is equivalent. This also yields a normed linear space but *it is not complete*.

[32] Henri Léon Lebesgue (1875–1941) French mathematician. His definition of the Lesbegue integral greatly extended the scope of Fourier analysis.

[33] George Friedrich Bernhard Riemann (1826–1866) German mathematician. He got his Ph. D. 1951 at Göttingen, supervised by Gauss. In his habilitation lecture on Geometry Riemann introduced the curvature tensor and laid the groundwork for Einstein's theory of relativity.

A modification of the L_2 norm that also includes higher derivatives of f is used in the Sobolev spaces, which is a theoretical framework for the study of the practically very important Finite Element Methods (FEM), used in particular for the numerical treatment of partial differential equations; see Vol III. Although $C[0,1]$ is an infinite-dimensional space, the restriction of the continuous functions f to the equidistant grid defined by $x_i = ih$, $h = 1/n$, $i = 0 : n$, constitute an $n+1$ dimensional space, with the function values on the grid as coordinates. If we choose the norm

$$\|f\|_{2,G_h} = \Big(\sum_{i=0}^{1/h} h|f(x_i)|^2\Big)^{1/2},$$

then

$$\lim_{h\to 0} \|f\|_{2,G_h} = \big(\int_0^1 |f(x)|^2\,dx\big)^{1/2} = \|f\|_{2,[0,1]}.$$

Limit processes of this type are common in Numerical Analysis.

Notice that even if $n+1$ functions $\phi_1(x), \phi_1(x), \ldots, \phi_{n+1}(x)$ are linearly independent on the interval $[0,1]$ (say), their restrictions to a grid with n points must be linearly dependent; but if a number of functions are linearly independent on a set $\mathcal{M}$ (a discrete set or continuum), any extensions of these functions to a set $\mathcal{M}' \supset \mathcal{M}$ will also be linearly independent.

The class of functions, analytic in a simply connected domain $\mathcal{D} \subset \mathbf{C}$, normed with $\|f\|_{\mathcal{D}} = \max_{z\in\partial\mathcal{D}} |f(z)|$, is a Banach space denoted by $\mathbf{Hol}(\mathcal{D})$. (The explanation to this term is that analytic functions are also called **holomorphic**.) By the maximum principle for analytic functions $|f(z)| \le \|f\|_{\mathcal{D}}$ for $z \in \mathcal{D}$.

4.5.2 Operator Norms and the Distance Formula

The concepts of **linear operator** and **linear functional** were introduced in Sec. 3.2.2. We here extend to a general vector space $\mathcal{B}$ some definitions for a finite dimensional vector space given in Appendix A.

Next we shall generalize the concept **operator norm** that we have previously used for matrices. Consider an arbitrary bounded linear operator $A : S_1 \mapsto S_2$ in a normed vector space S.

$$\|A\| = \sup_{\|f\|_{S_1}} \|Af\|_{S_2} \tag{4.5.5}$$

Note that $\|A\|$ depends on the vector norm in both S_1 and S_2. It follows that $\|Af\| \le \|A\|\|f\|$. Moreover, whenever the ranges of the operators A_1, A_2 are such that $A_1 + A_2$ and A_1A_2 are defined

$$\|\lambda A\| \le |\lambda|\|A\|, \qquad \|A_1+A_2\| \le \|A_1\|+\|A_2\|, \qquad \|A_1\cdot A_2\| \le \|A_1\|\cdot\|A_2\|. \tag{4.5.6}$$

Similarly for sums with an arbitrary number of terms and for integrals, etc. It follows that $\|A^n\| \le \|A\|^n$, $n = 2, 3, \ldots$.

Example 4.5.1.

Let $f \in C[0,1]$, $\|f\| = \|f\|_\infty$,

$$Af = \sum_{i=1}^{m} a_i f(x_i) \;\Rightarrow\; \|A\| = \sum_{i=1}^{m} |a_i|,$$

$$Af = \int_0^1 e^{-x} f(x)\,dx \;\Rightarrow\; \|A\| = \int_0^1 e^{-x}\,dx = 1 - e^{-1},$$

$$B = \sum_{i=1}^{m} a_i A^i \;\Rightarrow\; \|B\| \le \sum_{i=1}^{m} |a_i| \|A\|^i, \quad (a_i \in \mathbf{C}),$$

$$Kf = \int_0^1 k(x,t) f(t)\,dt \;\Rightarrow\; \|K\|_\infty \le \sup_{x\in[0,1]} \int_0^1 |k(x,t)|\,dt.$$

The proofs of these results are left as a problem. In the last example, approximate the unit square by a uniform grid $(x_i, t_j)_{i,j=1}^{m}$, $h = 1/m$, and approximate the integrals by Riemann sums. Then approximate $\|K\|_\infty$ by the max norm for the matrix with the elements $k_{i,j} = hk(x_i, t_j)$, see Appendix A.8.

The **integral operator** K can thus be considered as an analogue of a matrix. This is a useful point of view also for the numerical treatment of linear integral equations, e.g., equations of the form $f - \lambda K f = g$, g given, see Volume III.

Example 4.5.2.

For the forward difference operator Δ we obtain $\|\Delta\|_\infty = 2$, hence $\|\Delta^k\|_\infty \le 2^k$. In fact $\|\Delta^k\|_\infty = 2^k$, because the upper bound 2^k is attained by the sequence $\{(-1)^n\}_0^\infty$. The same holds for ∇^k, δ^k, and $\mu\delta^k$.

Example 4.5.3.

Let $\mathcal{D}$ be a domain in $\mathbf{C}$, the interior of which contains the closed interval $[a,b]$. Define the mapping D^k: $\mathbf{Hol}\mathcal{D} \Rightarrow C[a,b]$ (with maxnorm), by

$$D^k f(x) = \frac{\partial^k}{\partial x^k} \frac{1}{2\pi i} \int_\partial \mathcal{D} f(z) \frac{1}{(z-x)}\,dz = \frac{1}{2\pi i} \int_\partial \mathcal{D} \frac{k! f(z)}{(z-x)^{k+1}}\,dz.$$

$$sup_{x\in[a,b]} \|D^k f(x)\| \le \max_{z\in\partial\mathcal{D}} |f(z)| \cdot \sup_{x\in[a,b]} \frac{k!}{2\pi} \int_{\partial\mathcal{D}} \frac{|dz|}{|z-x|^{k+1}} < \infty.$$

Note that D^k is in this setting a *bounded* operator, while if we had considered D^k to be a mapping from $C^k[a,b]$ to $C[a,b]$, where both spaces are normed with the max norm in $[a,b]$, D^k would have been an *unbounded* operator.

Many of the procedures for the approximate computation of integrals, derivatives, etc. that encounter in this book, may be characterized as follows. Let A be a linear functional, such that Af cannot be easily computed for an arbitrary function

f, but it can be approximated by another linear functional $\tilde{A}_k$, more easily computable, such that $\tilde{A}_k f = Af \ \forall f \in \mathcal{P}_k$. A general error bound to such procedures was given in Sec. 3.2.3 by the Peano Remainder Theorem, in terms of an integral,

$$\int f^{(k)}(u)K(u)\,du$$

where the Peano kernel $K(u)$ is determined by the functional $R = \tilde{A} - A$.

Now we shall give a different type of error bound for more general approximation problems, where other classes of functions and operators may be involved. Furthermore, no estimate of $f^{(k)}(u)$ is required. It is based on the following almost trivial theorem. It yields, however, often less sharp bounds than the Peano formula, in situations when the latter can be applied.

Theorem 4.5.4. *The Norm and Distance Formula*

Let A, $\tilde{A}$ be a two linear operators bounded in a Banach space $\mathcal{B}$, such that for any vector s in a certain linear subspace S, $\tilde{A}s = As$. Then

$$\|\tilde{A}f - Af\| \le \|\tilde{A} - A\|\,\mathrm{dist}(f, \mathcal{S}) \quad \forall f \in \mathcal{B}.$$

Proof. Set $R = \tilde{A} - A$. For any positive ϵ, there exists a vector $s_\epsilon \in \mathcal{S}$ such that

$$\|f - s_\epsilon\| = \mathrm{dist}(f, s_\epsilon) < \inf_{s\in\mathcal{S})} \mathrm{dist}(f, s) + \epsilon = \mathrm{dist}(f, \mathcal{S}) + \epsilon.$$

Then $\|Rf\| = \|Rf - Rs_\epsilon\| = \|R(f - s_\epsilon)\| \le \|R\|\|f - s_\epsilon\| < (\mathrm{dist}(f, \mathcal{S}) + \epsilon)\|R\|$. The theorem follows, since this holds for every $\epsilon > 0$. □

The following is a common particular case of the theorem. *If A, A_k are* **linear functionals** *such that $A_k p = Ap \ \forall p \in \mathcal{P}_k$, then*

$$|A_k f - Af| \le (\|A_k\| + \|A\|)\mathrm{dist}(f, \mathcal{P}_k). \tag{4.5.7}$$

Another important particular case of the theorem concerns **projections** P from a function space to a finite dimensional subspace, $\mathcal{S}_k$, e.g., *interpolation and series truncation operators*, $A = I$, $\tilde{A} = P$, see the beginning of Sec. 4.5.2. Then

$$\|Pf - f\| \le \|(P - I)f\| \le (\|P\| + 1)\mathrm{dist}(f, \mathcal{S}_k). \tag{4.5.8}$$

The Norm and Distance Formula requires bounds for $\|\tilde{A}\|$, $\|A\|$ and $\mathrm{dist}(f, \mathcal{S})$. We have seen examples above, how to obtain bounds for operator norms. Now we shall exemplify how to obtain bounds for the distance of f from some relevant subspace $\mathcal{S}$, in particular spaces of polynomials or trigonometric polynomials restricted to some real interval $[a, b]$. For the efficient estimation of $\mathrm{dist}(f, \mathcal{S})$ it may be important, e.g., to take into account that f is analytic in a larger domain than $[a, b]$.

Theorem 4.5.5. *Estimation of* $\operatorname{dist}_\infty(f, \mathcal{P}_k)$ *in terms of* $\|f^{(k)}\|_\infty$.

Let $f \in C^k[a,b] \subset \mathbf{R}$, $\|g\| = \max_{t\in[a,b]} |g(t)|$, $\forall g \in C[a,b]$. *Then*

$$\operatorname{dist}_{\infty,[a,b]}(f, \mathcal{P}_k) \le \frac{2\|f^{(k)}\|_{\infty,[a,b]}}{k!}\Big(\frac{b-a}{4}\Big)^k.$$

Proof. Let $p(t)$ be the polynomial which interpolates $f(t)$ at the points $t_j \in [a,b]$, $j = 1:k$. By the remainder term in interpolation, (4.2.6),

$$|f(t) - p(t)| \le \max_{\xi\in[a,b]} \frac{|f^{(k)}(\xi)|}{k!} \prod_{j=1}^{k} |t - t_j|.$$

Set $t = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)x$, and choose $t_j = \frac{1}{2}(b+a) + \frac{1}{2}(b-a)x_j$ where x_j are the zeros of the Chebyshev polynomial $T_k(x)$, i.e. p is the Chebyshev interpolation polynomial for f on the interval $[a,b]$. Set $M = \|f^{(k)}\|/k!$. Then

$$|f(t) - p(t)| \le M \prod_{j=1}^{k} \frac{(b-a)|x - x_j|}{2}, \quad x \in [-1,1].$$

Since the leading coefficient of $T_k(x)$ is 2^{k-1}, and $|T_k(x)| \le 1$, we have, for $t \in [a,b]$,

$$|f(t) - p(t)| \le M\left(\frac{b-a}{2}\right)^k \frac{1}{2^{k-1}}|T_k(x)| \le M\left(\frac{b-a}{2}\right)^k \frac{1}{2^{k-1}},$$

The bound stated for $\operatorname{dist}_\infty(f, \mathcal{P}_k)$ is thus satisfied. □

Example 4.5.4.

By the above theorem, the function e^t can be approximated on the interval $[0,1]$ by a polynomial in $\mathcal{P}_6$ with the error bound $2e\cdot 4^{-6}/6! \approx 2\cdot 10^{-6}$. According to the proof this accuracy is attained by Chebyshev interpolation on $[0,1]$.

If one instead uses the Maclaurin series, truncated to $\mathcal{P}_6$, then the remainder is $e^\theta/(6!) \ge 1.3\cdot 10^{-3}$. Similarly, with the truncated Taylor series about $t = \frac{1}{2}$ the remainder is $e^\theta/(2^6 6!) \ge 2\cdot 10^{-5}$, still significantly less accurate than the Chebyshev interpolation. Economization of power series (see Problem 3.2.4), yields approximately the same accuracy as Chebyshev interpolation.

If we do these things on an interval of length h (instead of the interval $[0,1]$) all the bounds are to multiplied by h^6.

Example 4.5.5. *The use of analyticity in estimates for* $\operatorname{dist}_\infty(f, \mathcal{P}_k)$ *etc.*

Denote by $\mathcal{E}_R$ an ellipse in $\mathbf{C}$ with foci at -1 and 1; R is equal to the sum of the semi-axes. Theorem 3.2.3 gives the following truncation error bound for the Chebyshev expansion for a function $f \in \mathbf{Hol}(\mathcal{E}_R)$ and real-valued on $[-1,1]$.

$$\Big|f(x) - \sum_{j=0}^{k-1} c_j T_j(x)\Big| \le \frac{2\|f\|_{\mathcal{E}_R} R^{-k}}{1 - R^{-1}}, \quad x \in [-1,1].$$

This implies that, on the same assumptions concerning f,

$$\operatorname{dist}_{\infty,[-1,1]}(f,\mathcal{P}_k) \le \frac{2\|f\|_{\mathcal{E}_R} R^{-k}}{1-R^{-1}}.$$

Suppose that $f \in \mathbf{Hol}(\mathcal{D})$, where $\mathcal{D} \supseteq \dfrac{b+a}{2} + \dfrac{b-a}{2}\mathcal{E}_R$. Then transforming from $[-1,1]$ to a general interval $[a,b] \subset \mathbf{R}$, we have

$$\operatorname{dist}_{\infty,[a,b]}(f,\mathcal{P}_k) \le 2\|f\|_{\mathcal{D}} \left(\frac{b-a}{2R}\right)^k \frac{2R}{2R-(b-a)}.$$

Example 4.5.6.

As a first simple example we shall derive an error bound for one step with the trapezoidal rule. Set

$$Af = \int_0^h f(x)\,dx, \qquad A_2 f = \frac{h}{2}\big(f(0)+f(h)\big).$$

We know that $A_2 p = Ap$ if $p \in \mathcal{P}_2$. By Theorem 4.5.5, $\mathrm{d}ist_{\,\infty}(f,\mathcal{P}_2) \le \|f''\|_\infty h^2/16$.

Furthermore, $\|A\|_\infty = \int_0^h dx = h$, $\|A_2\|_\infty = h$, hence by (4.5.7). the requested error bound becomes

$$\|A_2 f - Af\|_\infty \le 2h \cdot \|f''\|_\infty h^2/16 = \|f''\|_\infty h^3/8.$$

This general method does not always give the best possible bounds but, typically, it gives no gross overestimate. For the trapezoidal rule we know by Peano's method (Example 3.4.7) that $\|f''\|h^3/12$ is the best possible estimate, so we now obtained a 50% overestimate of the error.

The norm and distance formula can also be written in the form

$$\operatorname{dist}(f,\mathcal{S}) \ge |Af - \tilde{A}f|/\|A-\tilde{A}\|.$$

This can be used for finding a simple lower bound for $\mathrm{d}ist\,(f,\mathcal{P}_k)$ in terms of an easily computed functional that vanishes on $\mathcal{P}_k$.

Example 4.5.7.

Let $\tilde{A}=0$. The functional $Af = f(1)-2f(0)+f(-1)$ vanishes for $f \in \mathcal{P}_2$. If the maximum norm is used on [-1,1], then $\|A\| = 1+2+1 = 4$. Thus

$$\operatorname{dist}(f,\mathcal{P}_2)_{\infty,[-1,1]} \ge \frac{|Af|}{\|A\|} = \frac{1}{4}|f(1)-2f(0)+f(-1)|.$$

It follows, e.g., that the curve $y = e^x$ cannot be approximated by a straight line in $[-1,1]$ with an error less than $(e-2+e^{-1})/4 \approx 0.271$. (This can also be seen without the use of the Norm and Distance Formula.)

It is harder to derive the following generalization without the Norm and Distance Formula. By Example 4.5.2, $\|\Delta^k\| = 2^k$, $\Delta^k p = 0$ if $p \in \mathcal{P}_k$, hence

$$\operatorname{dist}(f, \mathcal{P}_k)_{\infty,[x_0,x_k]} \geq 2^{-k}|\Delta^k f(x_0)|. \tag{4.5.9}$$

There is another inequality that is usually sharper but less convenient to use. (It follows from the discrete orthogonality property of the Chebyshev polynomials, see Sec. 4.6.)

$$\operatorname{dist}(f, \mathcal{P}_k)_{\infty,[x_0,x_k]} \geq \frac{1}{k}\left|\sum_{j=0}^{k}(-1)^j a_j f\left(\cos\frac{j\pi}{k}\right)\right|, \tag{4.5.10}$$

where $a_j = \frac{1}{2}$, $j = 0, k$ and $a_j = 1$ otherwise. Inequalities of this type can reveal when one had better using *piecewise polynomial approximation* of a function on an interval instead of using a single polynomial over the whole interval. See also Sec. 4.4.

One of the fundamental theorems in approximations theory is Weierstrass'[34] approximation theorem.

Theorem 4.5.6. Weierstrass' Approximation Theorem

For every continuous function f defined on a closed, bounded interval $[a, b]$ it holds that

$$\lim_{n\to\infty} \operatorname{dist}(f, \mathcal{P}_n)_{\infty,[a,b]} = 0.$$

Proof. For an elegant proof due to S. Bernstein using Bernstein polynomials; see Davis [19, Sec. 6.2]. □

The smoother f is, the quicker $\operatorname{dist}(f, P_n)$ decreases, and the narrower the interval is, the less $\operatorname{dist}(f, P_n)$ becomes. In many cases $\operatorname{dist}(f, P_n)$ decreases so slowly toward zero (as n grows) that it is impractical to attempt to approximate f with only one polynomial in the entire interval $[a, b]$.

In infinite-dimensional spaces, certain operators may not be defined everywhere, but only in a set that is everywhere dense in the space. For example, in the space $C[a, b]$ of continuous functions on a bounded interval (with the maximum norm), the operator $A = d/dx$ is not defined everywhere, since there are continuous functions, which are not differentiable. By Weierstrass' Approximation Theorem any continuous function can be approximated uniformly to arbitrary accuracy by a polynomial. In other words: the set of polynomials is everywhere dense in C, and hence the set of differentiable functions is so too. Moreover, even if Au exists, A^2u may not exist. That A^{-1} may not exist, is no novel feature of infinite-dimensional spaces. In $C[a, b]$ the norm of $A = d/dx$ is infinite. This operator is said to be unbounded.

[34] Kurt Theodor Wilhelm Weierstrass (1815–1897) German mathematician, whose lectures at Berlin University attracted students from all over the world. He set high standards of rigor in his work and is known as the father of modern analysis.

4.5.3 Inner Product Spaces and Orthogonal Systems

An abstract foundation for least squares approximation is furnished by the theory of **inner product spaces**, which we now introduce.

Definition 4.5.7.

A normed linear space $\mathcal{S}$ will be called an inner product space, if for each two elements f, g in $\mathcal{S}$ there is a scalar designated by (f, g) with the following properties

1.	$(f+g,h) = (f,h) + (g,h)$	*(linearity)*
2.	$(f,g) = \overline{(g,f)}$	*(symmetry)*
3.	$(\alpha f, g) = \alpha(f,g), \quad \alpha$ *scalar*	*(homogeneity)*
4.	$(f,f) \geq 0, \quad (f,f) = 0,$ *iff* $f = 0$	*(positivity)*

The **inner product** (f, g) *is scalar, i.e. real in a real space and complex in a complex space. The* **norm** *is defined as*

$$\|f\| = (f,f)^{1/2}.$$

We shall show below that the triangle inequality is satisfied. (The other axioms for a norm are obvious.) The standard inner products introduced in § 1.6.2, are particular cases, if we set $(x, y) = y^T x$ in $\mathbf{R}^n$, and $(x, y) = y^H x$ in $\mathbf{C}^n$. A **complete** inner-product space is called a **Hilbert space** and is often denoted $\mathcal{H}$ in this book.

One can make computations using the more general definition of (f, g) given above in the same way that one does with scalar products in linear algebra. Note, however, the *conjugations necessary in a complex space*, e.g.,

$$(f, \alpha g) = \bar{\alpha}(f, g), \tag{4.5.11}$$

because, by the axioms, $(f, \alpha g) = \overline{(\alpha g, f)} = \overline{\alpha(g, f)} = \bar{\alpha}\overline{(g, f)} = \bar{\alpha}(f, g)$. By the axioms it follows by induction that

$$\Big(\phi_k, \sum_{j=1}^{n} c_j \phi_j\Big) = \sum_{j=1}^{n} (\phi_k, c_j \phi_j) = \sum_{j=1}^{n} \bar{c}_j (\phi_k, \phi_j). \tag{4.5.12}$$

Theorem 4.5.8.

The Cauchy–Schwarz inequality in a complex space

$$|(f,g)| \leq \|f\| \, \|g\|.$$

Proof. Let f, g be two arbitrary elements in an inner-product space. Then[35], for every *real* number λ,

$$0 \leq (f + \lambda(f,g)g, f + \lambda(f,g)g) = (f,f) + 2\lambda|(f,g)|^2 + \lambda^2 |(f,g)|^2 (g,g).$$

[35] We found this proof in [41, n°83]. The application of the same idea in a *real* space can be made simpler, see Problem X.

This polynomial in λ with real coefficients cannot have two distinct zeros, hence the discriminant cannot be positive, i.e.

$$|(f,g)|^4 - (f,f)|(f,g)|^2(g,g) \le 0.$$

So, even if $(f,g) = 0$), $|(f,g)|^2 \le (f,f)(g,g)$. □

By the definition and the Cauchy–Schwarz inequality,

$$\begin{aligned}\|f+g\|^2 &= (f+g, f+g) = (f,f) + (f,g) + (g,f) + (g,g)\\ &\le \|f\|^2 + \|f\|\,\|g\| + \|g\|\,\|f\| + \|g\|^2 = (\|f\| + \|g\|)^2.\end{aligned}$$

This shows that the **triangle inequality** is satisfied by the norm defined above.

Example 4.5.8.
The set of all complex infinite sequences $\{x_i\}$ for which $\sum_{i=1}^{\infty} |x_i|^2 < \infty$ and equipped with the inner product

$$(x,y) = \sum_{i=1}^{\infty} x_i \overline{y}_i,$$

constitutes a Hilbert space.

Definition 4.5.9.
Two functions f and g are said to be **orthogonal** *if $(f,g) = 0$. A finite or infinite sequence of functions $\phi_0, \phi_1, \ldots, \phi_n$ constitutes an* **orthogonal system**, *if*

$$(\phi_i, \phi_j) = 0, \quad i \ne j, \text{ and } \|\phi_i\| \ne 0, \quad \forall i. \tag{4.5.13}$$

If, in addition, $\|\phi_i\| = 1 \forall i$, then the sequence is called an **orthonormal system**.

Theorem 4.5.10 (Pythagoras' theorem).
Let $\{\phi_1, \phi_2, \ldots, \phi_n\}$ be an orthogonal system in an inner-product space. Then

$$\Big\| \sum_{j=1}^{n} c_j \phi_j \Big\|^2 = \sum_{j=1}^{n} |c_j|^2 \|\phi_j\|^2.$$

The elements of an orthogonal system are linearly independent.

Proof. We start as in the proof of the triangle inequality:

$$\|f+g\|^2 = (f,f) + (f,g) + (g,f) + (g,g) = (f,f) + (g,g) = \|f\|^2 + \|g\|^2.$$

Using this result and induction the first stement follwos. The second statement then follows because $\sum c_j \phi_j = 0 \;\Rightarrow\; |c_j| = 0, \forall j$. □

Theorem 4.5.11.

A linear operator P is called a **projection** *(or projector) if $P = P^2$. Let $\mathcal{V}$ be the range of the operator P. Then P is a projection if and only $Pv = v$ if for each $v \in \mathcal{V}$.*

Proof. If P is a projection, then $v = Px$ for some $x \in \mathcal{B}$, hence $Pv = P^2x = Px = v$. Conversely, if Q is a linear operator, such that $Qx \in \mathcal{V}$, $\forall x \in \mathcal{B}$, and $v = Qv$, $\forall v \in V$, then Q is a projection, in fact $Q = P$. □

Note that $I - P$ is also a projection, because

$$(I - P)(I - P) = I - 2P + P^2 = I - P.$$

Every vector $x \in \mathcal{B}$ can be written *uniquely* in the form $x = u + w$, where $u = Px$, $w = (I - P)x$, so that $u \in range\,(P)$, $w \in range\,(I - P)$.

Important examples of projections in function spaces are interpolation operators, e.g., the mapping of $C[a, b]$ into $\mathcal{P}_k$ by Newton or Lagrange interpolation, because each polynomial is mapped to itself. The two types of interpolation are the same projection, although they use different bases in $\mathcal{P}_k$. Another example is the mapping of a linear space of functions, *analytic* on the unit circle, into $\mathcal{P}_k$ so that each function is mapped to its *Maclaurin expansion truncated to* $\mathcal{P}_k$. There are analogous projections where, e.g., periodic functions and trigonometric polynomials are involved, or functions in C^3 and cubic splines.

In an inner product space, the **adjoint operator** A^* of a linear operator A is defined by the requirement that

$$(A^*u, v) = (u, Av), \ \forall\ u, v. \tag{4.5.14}$$

An operator A is called **self-adjoint** if $A = A^*$. In $\mathbf{R}^n$, we define $(u, v) = uv^T$, i.e. the standard scalar product. Then $(A^*u)^Tv = u^TAv$, i.e. $u^T((A^*)^Tv = u^TAv$ hence $(A^*)^T = A$. It follows that symmetric matrices are self-adjoint. Similarly, in $\mathbf{C}^n$, we define $(u, v) = uv^H$. It follows that $A^* = A^H$ and that Hermitean matrices are self-adjoint. An operator B is **positive definite** if $(u, Bu) > 0\ \forall\ u \neq 0$.

Example 4.5.9.

An important example of an orthogonal system is the sequence of trigonometric functions $\phi_j(x) = \cos jx$, $j = 0 : m$. These form an orthogonal system, with the either of the two inner products

$$(f, g) = \int_0^\pi f(x)g(x)\,dx \quad \text{(continuous case, } m = \infty\text{)},$$

$$(f, g) = \sum_{i=0}^{m} f(x_\mu)g(x_\mu), \quad x_\mu = \frac{2\mu + 1}{m + 1}\frac{\pi}{2} \quad \text{(discrete case)}.$$

Moreover, it holds that

$$\|\phi_j\|^2 = \frac{1}{2}\pi, \quad j > 0, \quad \|\phi_0\|^2 = \pi, \quad \text{(continuous case)},$$

$$\|\phi_j\|^2 = \frac{1}{2}(m+1), \quad 1 \le j \le m, \quad \|\phi_0\|^2 = m+1. \quad \text{(discrete case)}.$$

These results are closely related to the orthogonality of the Chebyshev polynomials; see Theorem 4.5.17.

Trigonometric interpolation and Fourier analysis will be treated in Sec. 4.6.

There are many other examples of orthogonal systems. Orthogonal systems of polynomials play an important role in approximation and numerical integration. Orthogonal systems also occur in a natural way in connection with eigenvalue problems for differential equations, which are quite common in mathematical physics.

4.5.4 Solution of the Approximation Problem

Orthogonal systems give rise to extraordinary formal simplifications in many situations. We now consider the least squares approximation problem of minimizing the norm of the error function $\|f^* - f\|$ over all functions $f^* = \sum_{j=0}^{n} c_j \phi_j$.

Theorem 4.5.12.

If $\phi_0, \phi_1, \ldots, \phi_n$ are linearly independent, then the least squares approximation problem has a unique solution,

$$f^* = \sum_{j=0}^{n} c_j^* \phi_j, \tag{4.5.15}$$

which is characterized by the orthogonality property that $f^* - f$ is orthogonal to all ϕ_j, $j = 0 : n$. The coefficients c_j^*, which are called **orthogonal coefficients** *(or Fourier coefficients), satisfy the linear system of equations*

$$\sum_{j=0}^{n} (\phi_j, \phi_k) c_j^* = (f, \phi_k). \tag{4.5.16}$$

called normal equations. In the important special case when $\phi_0, \phi_1, \ldots, \phi_n$ form an orthogonal system, the coefficients are computed more simply by

$$c_j^* = (f, \phi_j)/(\phi_j, \phi_j), \quad j = 0 : n. \tag{4.5.17}$$

Proof. Let $(c_0, c_1, \ldots, c_n)$ be a sequence of coefficients with $c_j \neq c_j^*$ for at least one j. Then

$$\sum_{j=0}^{n} c_j \phi_j - f = \sum_{j=0}^{n} (c_j - c_j^*)\phi_j + (f^* - f).$$

If $f^* - f$ is orthogonal to all the ϕ_j, then it is also orthogonal to the linear combination $\sum_{j=0}^{n} (c_j - c_j^*)\phi_j$, and according to the Pythagorean Theorem

$$\Big\| \sum_{j=0}^{n} c_j \phi_j - f \Big\|^2 = \Big\| \sum_{j=0}^{n} (c_j - c_j^*)\phi_j \Big\|^2 + \|(f^* - f)\|^2 > \|(f^* - f)\|^2.$$

Thus if $f^* - f$ is orthogonal to all ϕ_k, then f^* is a solution to the approximation problem. It remains to show that the orthogonality conditions

$$\Big(\sum_{j=0}^{n} c_j^*\phi_j - f, \phi_k\Big) = 0, \quad k = 0:n,$$

can be fulfilled. The above conditions are equivalent to the normal equations in (4.5.16). If $\{\phi_j\}_{j=0}^n$ constitutes an orthogonal system, then the system can be solved immediately, since in each equation all the terms with $j \neq k$ are zero. The formula in (4.5.17) then follows immediately.

Suppose now that we know only that $\{\phi_j\}_{j=0}^n$ are linearly independent. The solution to the normal equations exists and is unique, unless the homogeneous system,

$$\sum_{j=0}^{n}(\phi_j, \phi_k)c_j^* = 0, \quad k = 0:n$$

has a solution $c_0, c_1, \ldots, c_n$, with at least one $c_i \neq 0$. But this would imply

$$\Big\|\sum_{j=0}^{n} c_j\phi_j\Big\|^2 = \Big(\sum_{j=0}^{n} c_j\phi_j, \sum_{k=0}^{n} c_k\phi_k\Big) = \sum_{k=0}^{n}\sum_{j=0}^{n}(\phi_j, \phi_k)c_jc_k = \sum_{k=0}^{n} 0 \cdot c_k = 0,$$

which contradicts that the ϕ_j were linearly independent. □

In the case where $\{\phi_j\}_{j=0}^n$ form an orthogonal system, *the Fourier coefficients c_j^* are independent of n* (see formula (4.5.17)). This has the important advantage that one can increase the total number of parameters without recalculating any previous ones. Orthogonal systems are advantageous not only because they greatly simplify calculations; using them, one can often avoid numerical difficulties with round-off error which may occur when one solves the normal equations for a nonorthogonal set of basis functions.

With every continuous function f one can associate an infinite series,

$$f \sim \sum_{j=0}^{\infty} c_j^*\phi_j, \quad c_j^* = \frac{(f, \phi_j)}{(\phi_j, \phi_j)}.$$

Such a series is called an **orthogonal expansion**. For certain orthogonal systems this series converges with very mild restrictions on the function f.

Theorem 4.5.13.

If f^* is defined by formulas (4.5.15) *and* (4.5.17)*, then*

$$\|f^* - f\|^2 = \|f\|^2 - \|f^*\|^2 = \|f\|^2 - \sum_{j=0}^{n}(c_j^*)^2\|\phi_j\|^2.$$

Proof. Since $f^* - f$ is, according to Theorem 4.5.12, orthogonal to all ϕ_j, $0 \leq j \leq n$, then $f^* - f$ is orthogonal to f^*. The theorem then follows directly from the Pythagorean Theorem and Theorem 4.5.10. □

We obtain as corollary **Bessel's inequality**:

$$\sum_{j=0}^{n}(c_j^*)^2\|\phi_j\|^2 \le \|f\|^2. \tag{4.5.18}$$

The series $\sum_{j=0}^{\infty}(c_j^*)^2\|\phi_j\|^2$ is convergent. If $\|f^* - f\| \to 0$ as $n \to \infty$, then the sum of the latter series is equal to $\|f\|^2$, which is **Parseval's identity**.

Theorem 4.5.14.

If $\{\phi_j\}_{j=0}^m$ are linearly independent on the grid $\{x_i\}_{j=0}^m$, then the interpolation problem of determining the coefficients $\{c_i\}_{j=0}^m$ such that

$$\sum_{j=0}^{m} c_j\phi_j(x_i) = f(x_i), \quad i = 0:m, \tag{4.5.19}$$

has exactly one solution. Interpolation is a special case ($n = m$) of the method of least squares. If $\{\phi_j\}_{j=0}^m$ is an orthogonal system, then the coefficients c_j are equal to the orthogonal coefficients in (4.5.17).

Proof. The system of equations (4.5.19) has a unique solution, since its column vectors are the vectors $\phi_j(G)$, $j = 0:n$, which are linearly independent. For the solution of the interpolation problem it holds that $\|c_j\phi_j - f\| = 0$; that is, the error function has the least possible semi-norm. The remainder of the theorem follows from Theorem 4.5.12. □

The following collection of important and equivalent properties is named the **Fundamental theorem of orthonormal expansions**, by Davis [19, Theorem 8.9.1], whom we follow closely at this point.

Theorem 4.5.15.

Let $\phi_1, \phi_2, \cdots$, be a sequence of orthonormal elements in in a complete inner product space $\mathcal{H}$. The following seven statements are equivalent: [36]

(A) *The ϕ_j is a complete orthonormal system in $\mathcal{H}$.*

(B) *The orthonormal expansion of any element $y \in \mathcal{H}$ converges in norm to y; i.e.*

$$\lim_{n\to\infty}\Big\|y - \sum_{j=1}^{n}(y,\phi_j)\phi_j\Big\|. \tag{4.5.20}$$

(C) *Parseval's identity holds for every $y \in \mathcal{H}$, i.e.*

$$\|y\|^2 = \sum_{j=1}^{\infty}|(y,\phi_j)|^2. \tag{4.5.21}$$

[36] We assume that $\mathcal{H}$ is not finite-dimensional, in order to simplify the formulations. Only minor changes are needed in order to cover the finite-dimensional case.

(C') *The extended Parseval's identity holds for every $x, y \in \mathcal{H}$, i.e.*

$$(x, y) = \sum_{j=1}^{\infty} (x, \phi_j)(\phi_j, y). \tag{4.5.22}$$

(D) *There is no strictly larger orthonormal system containing $\phi_1, \phi_2, \cdots$*

(E) *If $y \in \mathcal{H}$ and $(y, \phi_j) = 0$, $j = 1, 2, \ldots$, then $y = 0$.*

(F) *An element of $\mathcal{H}$ is determined uniquely by its Fourier coefficients, i.e. if $(w, \phi_j) = (y, \phi_j)$, $j = 1, 2, \cdots$, then $w = y$.*

Proof. For the rest of the proof, See Davis [19, pp. 192ff]. □

Theorem 4.5.16.

The converse of statement (F) *holds, i.e. let $\mathcal{H}$ be a complete inner product space, and let a_j be constants such that $\sum_{j=1}^{\infty} |a_j|^2 < \infty$. Then there exists an $y \in \mathcal{H}$, such that $y = \sum_{j=1}^{\infty} a_j \phi_j$ and $(y, \phi_j) = a_j \ \forall\ j$.*

Proof. Omitted. □

4.5.5 Orthogonal Polynomials and Least Squares Approximation

By a family of **orthogonal polynomials** we mean a triangle family of polynomials, which (in the continuous case) is an orthogonal system with respect to a given inner product Expansions of functions in terms of orthogonal polynomials are very useful. They are easy to manipulate, have good convergence properties and usually give a well conditioned representation. The theory of orthogonal polynomials is also of fundamental importance for many problems, which at first sight seem to have little connection with approximation (e.g., numerical integration, continued fractions, and the algebraic eigenvalue problem).

Perhaps the most important example of a family of orthogonal polynomials is the Chebyshev polynomials $T_n(x) = \cos(n \arccos(x))$ introduced in Sec. 3.2.3. These are are orthogonal on $[-1, 1]$ with respect to the weight function $(1 - x^2)^{-1/2}$ and also with respect to a discrete inner product. Their properties can be derived by rather simple methods.

Theorem 4.5.17.

The Chebyshev polynomials have the following two orthogonality properties. Set

$$(f, g) = \int_{-1}^{1} f(x) g(x) (1 - x^2)^{-1/2}\, dx \tag{4.5.23}$$

(the continuous case). Then $(T_0, T_0) = \pi$, *and*

$$(T_j, T_k) = \begin{cases} 0 & \text{if } j \neq k, \\ \pi/2 & \text{if } j = k \neq 0. \end{cases} \tag{4.5.24}$$

Let x_k *be the zeros of* $T_{m+1}(x)$ *and set*

$$(f, g) = \sum_{k=0}^{m} f(x_k)g(x_k), \quad x_k = \cos\Big(\frac{2k+1}{m+1}\frac{\pi}{2}\Big) \tag{4.5.25}$$

(the discrete case). Then $(T_0, T_0) = m + 1$, *and*

$$(T_j, T_k) = \begin{cases} 0 & \text{if } j \neq k, \\ (m+1)/2 & \text{if } j = k \neq 0. \end{cases} \tag{4.5.26}$$

Proof. In the continuous case, let $j \neq k$, $j \geq 0$, $k \geq 0$. From $x = cos\phi$ it follows that $dx = \sin\phi d\phi = (1 - x^2)^{1/2} d\phi$. Hence

$$\begin{aligned}(T_j, T_k) &= \int_0^\pi \cos jx \cos kx\, dx = \int_0^\pi \frac{1}{2}\big(\cos(j-k)x + \cos(j+k)x\big)\, dx \\ &= \frac{1}{2}\Big(\frac{\sin(j-k)\pi}{j-k} + \frac{\sin(j+k)\pi}{j+k}\Big) = 0,\end{aligned}$$

whereby orthogonality is proved.

In the discrete case, set $h = \partial\mu/(m+1)$, $x_\mu = h/2 + \mu h$,

$$(T_j, T_k) = \sum_{\mu=0}^{m} \cos jx_\mu \cos kx_\mu = \frac{1}{2}\sum_{\mu=0}^{m}\big(\cos(j-k)x_\mu + \cos(j+k)x_\mu\big).$$

Using notation from complex numbers ($i = \sqrt{-1}$) we have

$$(T_j, T_k) = \frac{1}{2}\mathrm{Re}\Big(\sum_{\mu=0}^{m} e^{i(j-k)h(1/2+\mu)} + \sum_{\mu=0}^{m} e^{i(j+k)h(1/2+\mu)}\Big). \tag{4.5.27}$$

The sums in (4.5.27) are geometric series with ratios $e^{i(j-k)h}$ and $e^{i(j+k)h}$, respectively. If $j \neq k$, $0 \leq j \leq m$, $0 \leq k \leq m$, then the ratios are never equal to 1, since

$$0 < |\ (j \pm k)h| \leq \frac{2m}{m+1}\pi < \pi.$$

The first sum in (4.5.27) is, then, using the formula for the sum of a geometric series

$$e^{i(j-k)(h/2)}\frac{e^{i(j-k)(m+1)h} - 1}{e^{i(j-k)h} - 1} = \frac{e^{i(j-k)\pi} - 1}{e^{i(j-k)(h/2)} - e^{-i(j-k)(h/2)} - 1} = \frac{(-1)^{j-k} - 1}{2i\sin(j-k)h/2}.$$

The real part of the last expression is clearly zero. An analogous computation shows that the real part of the other sum in (4.5.27) is also zero. Thus the orthogonality property holds in the discrete case also. It is left to the reader to show that the expressions when $j = k$ given in the theorem are correct. □

Example 4.5.10. *Chebyshev interpolation*

Let $p(x)$ denote the Chebyshev interpolation polynomial of degree m. For many reasons it is practical to write this interpolation polynomial in the form

$$p(x) = \sum_{i=0}^{m} c_i T_i(x). \tag{4.5.28}$$

Then using the discrete orthogonality property (4.5.26)

$$c_i = \frac{(f, T_i)}{\|T_i\|^2} = \frac{1}{\|T_i\|^2} \sum_{k=0}^{m} f(x_k) T_i(x_k), \tag{4.5.29}$$

where

$$\|T_0\|^2 = m + 1, \quad \|T_i\|^2 = \tfrac{1}{2}(m+1), \quad i > 0.$$

The recursion formula (3.2.20) can be used for calculating the orthogonal coefficients according to (4.5.29). For computing $p(x)$ with (4.5.28) Clenshaw's algorithm (Theorem 3.2.5) can be used.

Occasionally on is interested in the partial sums of (4.5.28). For example, if the values of $f(x)$ are afflicted with statistically independent errors with standard deviation σ, then the series can be broken off when for the first time

$$\left\| f - \sum_{i=0}^{p} c_i T_i(x) \right\| < \sigma m^{1/2}.$$

We assume in the following that in the continuous case the weight function $w(x) \geq 0$ is such that the **moments**

$$\mu_k = (x^k, 1) = \int_a^b x^k w(x)\, dx. \tag{4.5.30}$$

are defined for all $k \geq 0$, and $\mu_0 > 0$. Note that the inner product (4.5.38) has the property that

$$(xf, g) = (f, xg). \tag{4.5.31}$$

Given a linearly independent sequence of vectors in an inner-product space an orthogonal system can be derived by a process analogous to Gram–Schmidt orthogonalization. The proof below is constructive and leads to a unique construction of the sequence of orthogonal polynomials ϕ_k, $n \geq 1$, with leading coefficients equal to 1.

Theorem 4.5.18.

In an inner product space with the inner product (4.5.38), there is a triangle family of orthogonal polynomials $\phi_k(x)$, $k = 1, 2, 3, \ldots$, such that $\phi_{k+1}(x)$ has exact degree k, and is orthogonal to all polynomials of degree less than k. The family is uniquely determined apart from the fact that the leading coefficients can be given arbitrary positive values.

The monic orthogonal polynomials satisfy the three-term recurrence formula,

$$\phi_{k+1}(x) = (x - \beta_k)\phi_k(x) - \gamma_{k-1}^2\phi_{k-1}(x), \quad k \geq 1, \tag{4.5.32}$$

with initial values $\phi_0(x) = 0$, $\phi_1(x) = 1$. *The recurrence coefficients are given by*

$$\beta_k = \frac{(x\phi_k, \phi_k)}{\|\phi_k\|^2}, \qquad \gamma_{k-1} = \frac{\|\phi_k\|}{\|\phi_{k-1}\|}. \tag{4.5.33}$$

If the weight distribution is symmetric about $x = \beta$, *then* $\beta_k = \beta$ *for all* k.

Proof. By induction: Suppose that the $\phi_j \neq 0$ have been constructed for $1 \leq j \leq k$. We now seek a polynomial ϕ_{k+1} of degree k with leading coefficient equal to 1, which is orthogonal to all polynomials in $\mathcal{P}_k$. Since $\{\phi_j\}_{j=1}^k$ is a triangle family, every polynomial of degree $k-1$ can be expressed as a linear combination of these polynomials. Therefore we can write

$$\phi_{k+1} = x\phi_k - \sum_{i=1}^{k} c_{k,i}\phi_i. \tag{4.5.34}$$

Clearly ϕ_{k+1} has leading coefficient one. The orthogonality condition is fulfilled if and only if

$$(x\phi_k, \phi_j) - \sum_{i=1}^{k} c_{k,i}(\phi_i, \phi_j) = 0, \quad j = 1 : k.$$

But $(\phi_i, \phi_j) = 0$ for $i \neq j$, and thus $c_{k,j}\|\phi_j\|^2 = (x\phi_k, \phi_j)$. From the definition of inner product (4.5.38), it follows that

$$(x\phi_k, \phi_j) = (\phi_k, x\phi_j).$$

But $x\phi_j$ is a polynomial of degree j. Thus if $j < k$, then it is orthogonal to ϕ_k. So $c_{kj} = 0$ for $j < k - 1$. From (4.5.34) it then follows that

$$\phi_{k+1} = x\phi_k - c_{k,k}\phi_k - c_{k,k-1}\phi_{k-1}, \tag{4.5.35}$$

which has the same form as the original assertion of the theorem if, we set

$$\beta_k = c_{k,k} = \frac{(x\phi_k, \phi_k)}{\|\phi_k\|^2}, \qquad \gamma_{k-1}^2 = c_{k,k-1} = \frac{(\phi_k, x\phi_{k-1})}{\|\phi_{k-1}\|^2}. \quad k \geq 1.$$

This shows the first part of (4.5.33).

The expression for γ_{k-1} can be written in another way. If we scalar-multiply (4.5.34) by ϕ_{k+1} we get

$$(\phi_{k+1}, \phi_{k+1}) = (\phi_{k+1}, x\phi_k) - \sum_{i=1}^{k} c_{k,i}(\phi_{k+1}, \phi_i) = (\phi_{k+1}, x\phi_k).$$

Thus $(\phi_{k+1}, x\phi_k) = \|\phi_{k+1}\|^2$, or if we decrease all indices by 1, $(\phi_k, x\phi_{k-1}) = \|\phi_k\|^2$. Substituting this in the expression for γ_{k-1} gives the second part of (4.5.33) ∎

Sometimes it is advantageous to consider corresponding orthonormal polynomials $\hat{\phi}_k(x)$, which satisfy $\|\hat{\phi}_k\| = 1$. Setting $\hat{\phi}_1 = c$, we find

$$\|\hat{\phi}_1\| = \int_a^b c^2 w(x)\,dx = c^2\mu_0 = 1,$$

and thus $\hat{\phi}_1 = 1/\sqrt{\mu_0}$. If we scale the monic orthogonal polynomials according to

$$\phi_k = (\gamma_1 \cdots \gamma_{k-1})\hat{\phi}_k, \quad k > 1, \tag{4.5.36}$$

then we find using (4.5.33) that

$$\frac{\|\hat{\phi}_k\|}{\|\hat{\phi}_{k-1}\|} = \frac{\gamma_1 \cdots \gamma_{k-2}}{\gamma_1 \cdots \gamma_{k-1}} \frac{\|\phi_k\|}{\|\phi_{k-1}\|} = 1.$$

Substituting (4.5.36) in (4.5.32) we obtain a recurrence relation for the orthonormal polynomials

$$\gamma_k\hat{\phi}_{k+1}(x) = (x - \beta_k)\hat{\phi}_k(x) - \gamma_{k-1}\hat{\phi}_{k-1}(x), \quad k \geq 1, \tag{4.5.37}$$

Let $\hat{p}_n$ denote the polynomial of degree n for which

$$\|f - \hat{p}_n\|_\infty = E_n(f) = \min_{p\in\mathcal{P}_{n+1}} \|f - p\|_\infty.$$

Set $p_n = \sum_{j=0}^n c_j\phi_j$, where c_j is the jth Fourier coefficient of f and $\{\phi_j\}$ are the orthogonal polynomials with respect to the inner product

$$(f, g)_w = \int_a^b f(x)g(x)w(x)\,dx, \quad w(x) \geq 0. \tag{4.5.38}$$

If we use the weighted Euclidian norm, $\hat{p}_n$ is of course not a better approximation than p_n. In fact

$$\begin{aligned}\|f - p\|_w^2 &= \int_a^b |f(x) - p_n(x)|^2 w(x)\,dx \\ &\leq \int_a^b |f(x) - \hat{p}_n(x)|^2 w(x)\,dx \leq E_n(f)^2 \int_a^b w(x)\,dx.\end{aligned} \tag{4.5.39}$$

This can be interpreted as saying that a kind of weighted mean of $|f(x) - p_n(x)|$ is less than or equal to $E_n(f)$, which is about as good result as one could demand. The error curve has an oscillatory behavior. In small subintervals $|f(x) - p_n(x)|$ can be significantly greater than $E_n(f)$. This is usually near the ends of the intervals or in subintervals where $w(x)$ is relatively small. Note that from (4.5.39) and Weierstrass approximation theorem it follows that

$$\lim_{n\to\infty} \|f - p\|_{2,w}^2 = 0$$

for every continuous function f. From (4.5.39) one gets after some calculations

$$\sum_{j=n+1}^{\infty} \|\phi_j\|^2 = \|f-p\|_{2,w}^2 \leq E_n(f)^2 \int_a^b w(x)\,dx,$$

which gives one an idea of how quickly the terms in the orthogonal expansion decrease.

In Sec. 4.1.3 we considered the discrete least squares approximation problem

$$f(x) \approx p(x) = \sum_{j=1}^{n} c_j p_j(x) \in \mathcal{P}_n,$$

where $p_1(x), p_2(x), \ldots, p_n(x)$ is a basis for $\mathcal{P}_n$. Then $S(c) = \|p-f\|^2$, the least squares problem to minimize $S(c)$ is equivalent to minimizing the norm of the error function $p-f$. Thus, if we could determine the basis functions so that they constitute an orthogonal system, i.e.

$$(p_i, p_k) = \begin{cases} 0, & i \neq k, \\ \|p_i\|^2 \neq 0, & i = k, \end{cases}$$

then by Theorem 4.5.12 the coefficients of the least squares approximations are

$$c_j = (f, p_j)/(p_j, p_j), \quad j = 1:n. \tag{4.5.40}$$

Example 4.5.11.

For the case $n = 1$, $p_1(x) = 1$, the normal equations reduce to the single equation $(p_0, p_0)c_0 = (f, p_0)$. Hence using the discrete inner product we get

$$c_1 = \frac{1}{\omega} \sum_{i=0}^{m} w_i f(x_i), \quad \omega = \sum_{i=0}^{m} w_i.$$

Here c_1 is said to be a **weighted mean** of the values of the function.

Let $\{x_i\}_{i=1}^m \in (a,b)$ be distinct points and $\{w_i\}_{i=1}^m$ a set of weights and define the weighted discrete inner product of two real-valued functions f and g on the grid $\{x_i\}_{j=1}^m$ by

$$(f,g) = \sum_{i=1}^{m} w_i f(x_i) g(x_i). \tag{4.5.41}$$

From Theorem 4.5.18 it follows that there is a unique associated triangle family of orthogonal polynomials $p_0, p_1, \ldots, p_{m-1}$, with leading coefficients equal to one, that satisfy the three-term recurrence

$$p_{-1}(x) = 0, \quad p_0(x) = 1, \tag{4.5.42}$$

$$p_{k+1}(x) = (x-\alpha_k)p_k(x) - \beta_k p_{k-1}(x), \quad k = 0, 1, 2, \ldots, \tag{4.5.43}$$

where

$$\alpha_k = \frac{(xp_k, p_k)}{\|p_k\|^2}, \qquad \beta_k = \frac{\|p_k\|^2}{\|p_{k-1}\|^2} \quad (k > 0). \tag{4.5.44}$$

If the weight distribution is symmetric about $x = \alpha$, then $\alpha_k = \alpha$ for all k.

For a discrete weight distribution on a grid with m points, the family ends with $p_m(x)$; $p_{m+1}(x)$ becomes zero at each grid point. In the continuous case, the family has infinitely many members.

Using (4.5.33 the coefficients α_k and β_k and the value of the polynomials p_k at the grid points x_i can be recursively computed, in the order

$$\alpha_0, p_1(x_i), \alpha_1, \beta_1, p_2(x_i), \ldots.$$

(Note that β_0 is not needed since $p_{-1} = 0$.) This procedure is called the **Stieltjes procedure**[37]

For $k = m$ the constructed polynomial p_m must be equal to

$$(x - x_0)(x - x_1) \cdots (x - x_m),$$

because this polynomial is zero at all the grid points, and thus orthogonal to all functions. From this it follows that $\|p_{m+1}\| = 0$; thus the computation of α_k cannot be carried out for $k = m + 1$ and the construction stops at $k = m$. This is natural, since there cannot be more than $m + 1$ orthogonal (or even linearly independent) functions on a grid with $m + 1$ points.

There are many computational advantages of using the recurrence relation (4.5.43) for discrete least squares data fitting. In a least squares approximation of the form

$$p(x) = \sum_{k=0}^{n} c_k p_k,$$

the coefficients

$$c_k = (p, p_k)/\|p_k\|^2,$$

are independent of n. In the computation of the coefficients c_k one can make use of the recursion formula (4.5.43). approximations of increasing degree can be recursively generated as follows. Suppose that p_i, $i = 0 : k - 1$, and the least squares approximation p_k of degree k have been computed. In the next step the coefficients β_k, γ_k are computed from (4.5.44) and then p_{k+1} by (4.5.43). The next approximation of f can now be obtained by

$$p_{k+1} = p_k + c_{k+1}p_{k+1}, \qquad c_{k+1} = (f, p_{k+1})/\|p_{k+1}\|^2. \tag{4.5.45}$$

Since p_{k+1} is orthogonal to p_k, an alternative expression for the new coefficient is

$$c_{k+1} = (r_k, p_{k+1})/\|p_{k+1}\|^2, \qquad r_k = f - p_k. \tag{4.5.46}$$

[37] Thomas Jan Stieltjes (1856–1894), was born in the Netherlands. After working with astronomical calculations at the Observatory in Leiden he got a university position in Toulouse, France. His work on continued fractions and the moment problem and invented a new concept of integral.

Assuming unit weights and that the grid is symmetric the coefficients α_k, c_k and the orthogonal functions p_k at the grid points can be using the Stieltjes procedure in about $4mn$ flops. If there are differing weights, then about mn additional operations are needed; similarly, mn additional operations are required if the grid is not symmetric. If the orthogonal coefficients are determined simultaneously for several functions on the same grid, then only about mn additional operations per function are required. (In the above, we assume $m \gg 1$, $n \gg 1$.) Hence the procedure is much more economical than the general methods based on normal equations which require $O(mn^2)$ flops.

Mathematically the two formulas (4.5.45) and (4.5.46) for c_{k+1} are equivalent. In finite precision, as higher degree polynomials p_{k+1} are computed, they will gradually lose orthogonality to previously computed p_j, $j \leq k$. In practice there is an advantage in using (4.5.46) since cancellation then will mostly take place in computing the residual $r_k = f - p_k$, and then the inner product (r_k, p_{k+1}) is computed accurately. Theoretically the error $\|p_k - f\|$ must be a non-increasing function of k. I However, when using the first formula one sometimes finds that *the residual norm increases when the degree of the approximation is increased*! With the modified formula (4.5.46) this is very unlikely to happen; see Problem 8.

Note that for $n = m$ we obtain the (unique) interpolation polynomial for the given points. Often the error decreases rapidly with k and then p_k provides a good representation of f already for small values of k. With some nets e.g., equidistant nets, one should choose n less than $2\sqrt{m}$, since otherwise *the approximation polynomial will have large oscillatory behavior between the grid points.*

When the coefficients c_j in the orthogonal expansion are known, then the easiest way to compute the numerical values of $p(x)$ is to use **Clenshaw's algorithm**; see Theorem 3.2.4.

For equidistant data, the **Gram polynomials** $\{P_{n,m}\}_{n=0}^{m}$ are of interest.[38] These polynomials are orthogonal with respect to the inner product

$$(f, g) = \frac{1}{m} \sum_{i=1}^{m} f(x_i) g(x_i), \quad x_i = -1 + (2i - 1)/m.$$

The weight distribution is symmetric around the origin $\alpha_k = 0$. and for the *monic* Gram polynomials the recursion formula is (see [2])

$$\begin{aligned} &P_{-1,m}(x) = 0, \qquad P_{0,m} = 1, \\ &P_{n+1,m}(x) = xP_{n,m}(x) - \beta_{n,m} P_{n-1,m}(x), \qquad n = 0 : m - 1, \end{aligned}$$

where $(n < m)$

$$\beta_{n,m} = \frac{n^2}{4n^2 - 1} \left(1 - \frac{n^2}{m^2}\right).$$

When $n \ll m^{1/2}$, these polynomials are well behaved. However, when $n \geq m^{1/2}$, the Gram polynomials have very large oscillations between the grid points,

[38] Jørgen Pedersen Gram (1850–1916), Danish mathematician, worked on probability and numerical analysis. Gram is now best remembered for the Gram–Schmidt orthogonalization process.

and a large maximum norm in $[-1, 1]$. This fact is related to the recommendation that when fitting a polynomial to *equidistant data*, one should never choose n larger than about $2m^{1/2}$.

4.5.6 Statistical Aspects of the Method of Least Squares

One of the motivations for the method of least squares is that it effectively reduces the influence of random errors in measurements. Let $f \in \mathbf{R}^m$ be a vector of observations that is related to a parameter vector $c \in \mathbf{R}^n$ by the linear relation

$$f = Ac + \epsilon, \quad A \in \mathbf{R}^{m\times n}, \tag{4.5.47}$$

where A is known matrix of full column rank and $\epsilon \in \mathbf{R}^m$ is a vector of random errors. We assume here that ϵ_i, $i = 1 : m$ has zero mean and covariance equal to σ^2, and that ϵ_i and ϵ_i are **uncorrelated** if $i \neq j$, that is

$$E(\epsilon) = 0, \qquad \text{var}(\epsilon) = \sigma^2 I,$$

(Recall the definitions of mean value and correlation in Sec. 1.5.1.) The parameter c is then a random vector, which we want to estimate in terms of the known quantities A and f.

Let $y^T c$ be a linear functional of the parameter c in (4.5.47). We say that $\theta = \theta(A, f)$ is an unbiased linear estimator of $y^T c$ if $E(\theta) = y^T c$. It is a **best linear unbiased estimator** (BLUE) if θ) has the smallest variance among all such estimators.

The **Gauss–Markov theorem**[39] places the method of least squares on a sound theoretical basis.

Theorem 4.5.19.

Consider a linear model (4.5.47), where ϵ is an uncorrelated random vector with zero mean and variance equal to σI. Then the best linear unbiased estimator of any linear functional $y^T c$ is $y^T \hat{c}$, where

$$\bar{c} = (A^T A)^{-1} A^T f$$

is the least squares estimator obtained by minimizing the sum of squares $\|f - Ac\|_2^2$.

The covariance matrix of the least squares estimate $\hat{c}$ equals

$$\text{var}(\hat{c}) = \sigma^2 (A^T A)^{-1}. \tag{4.5.48}$$

Furthermore, the quadratic form

$$s^2 = \|f - A\hat{c}\|_2^2/(m - n). \tag{4.5.49}$$

is an unbiased estimate of σ^2, i.e. $E(s^2) = \sigma^2$.

[39] This theorem is originally due to C. F. Gauss 1821. His contribution was somewhat neglected until rediscovered by the Russian mathematician A. A Markov in 1912.

Proof. See Zelen [58, pp. 560–561]. □

Suppose that the values of a function have been measured in the points $x_1, x_2, \ldots, x_m$. Let $f(x_p)$ be the measured value, and let $\bar{f}(x_p)$ be the "true" (unknown) function value, which is assumed to be the same as the *expected value* of the measured value. Thus *no systematic errors* are assumed to be present. Suppose further that *the errors in measurement at the various points are statistically independent*. Then we have a linear model $f(x_p) = \bar{f}(x_p) + \epsilon$, where

$$E(\epsilon) = 0, \qquad \text{var}(\epsilon) = \text{d}iag\,(\sigma_1^2, \ldots, \sigma_n^2), \tag{4.5.50}$$

where E denotes expected value and var variance. The problem is to use the measured data to estimate the coefficients in the series

$$f(x) = \sum_{j=1}^{n} c_j \phi(x), \quad n \le m.$$

where $\phi_1, \phi_2, \ldots, \phi_n$ are known functions. According to Theorem 4.5.19 the estimates c_j^*, which one gets by minimizing the sum

$$\sum_{p=1}^{m} w_p \Big(f(x_p) - \sum_{j=1}^{n} c_j \phi_j(x_p) \Big)^2, \quad w_p = \sigma_p^{-2},$$

have a smaller variance than the values one gets by any other linear unbiased estimation method. This minimum property holds not only for the estimates of the coefficients c_j, but also for every linear functional of the coefficients, e.g., the estimate

$$f_n^*(\alpha) = \sum_{j=1}^{n} c_j^* \phi(\alpha)$$

of the value $f(\alpha)$ at an arbitrary point α.

Suppose now that $\sigma_p = \sigma$ for all p and that the functions $\{\phi_j\}_{j=1}^n$ form an *orthonormal system* with respect to the discrete inner product

$$(f, g) = \sum_{p=1}^{m} f(x_p) g(x_p).$$

Then the least squares estimates are $c_j^* = (f, \phi_j)$, $j = 1 : n$. By Theorem 4.5.19 the estimates c_j^* and c_k^* are *uncorrelated if* $j \ne k$ and

$$E\{(c_j^* - \bar{c}_j)(c_k^* - \bar{c}_k)\} = \begin{cases} 0, & \text{if } j \ne k; \\ \sigma^2 & \text{if } j = k, \end{cases}$$

From this it follows that

$$\text{var}\{f_n^*(\alpha)\} = \text{var}\Big\{ \sum_{j=1}^{n} c_j^* \phi_j(\alpha) \Big\} = \sum_{j=1}^{n} \text{var}\{c_j^*\} |\phi_j(\alpha)|^2 = \sigma^2 \sum_{j=1}^{n} |\phi_j(\alpha)|^2.$$

As an average, *taken over the grid of measurement points*, the variance of the smoothed function values is

$$\frac{1}{m}\sum_{j=1}^{n}\operatorname{var}\{f_n^*(x_i)\} = \frac{\sigma^2}{m}\sum_{j=1}^{n}\sum_{i=1}^{m}|\phi_j(x_i)|^2 = \sigma^2\frac{n}{m}.$$

Between the grid points, however, the variance can in many cases be significantly larger. For example, when fitting a polynomial to measurements in *equidistant points*, the Gram polynomial $P_{n,m}$ can be much larger between the grid points when $n > 2m^{1/2}$. Set

$$\sigma_I^2 = \sigma^2\sum_{j=1}^{n}\tfrac{1}{2}\int_{-1}^{1}|\phi(x)|^2\,dx.$$

Thus σ_I^2 is an average variance for $f_n^*(x)$ *taken over the entire interval* $[-1,1]$. The following values for the ratio ρ between σ_I^2 and $\sigma^2(n+1)/(m+1)$ when $m = 42$ were obtained by H. Björk:

$n+1$	5	10	15	20	25	30	35
ρ	1.0	1.1	1.7	26	$7\cdot10^3$	$1.7\cdot10^7$	$8\cdot10^{11}$

These results are related to the recommendation that one should choose $n < 2m^{1/2}$ when fitting a polynomial to equidistant data. This recommendation seems to contradict the Gauss–Markov theorem, but in fact it just means that one gives up the requirement that there be no systematic errors. Still it is remarkable that this can lead to such a drastic reduction of the variance of the estimates.

If the measurement points are the Chebyshev abscissae, then no difficulties arise in fitting polynomials to data. The Chebyshev polynomials have a magnitude between grid points not much larger than their magnitude at the grid points. In this case the choice of n when m is given, is a question of compromising between taking into account the truncation errors (which decreases as n increases) and the random errors (which grow when n increases). If f is a sufficiently smooth function then in the Chebyshev case $|c_j|$ decreases quickly with j. In contrast the part of c_j which comes from errors in measurements varies randomly with a magnitude of about $\sigma(2/(m+1)^{1/2}$, using (4.5.25)) and $\|T_j\|^2 = (m+1)/2$. The expansion should be broken off when the coefficients begin to "behave randomly". An expansion in terms of Chebyshev polynomials can hence be used for *filtering away the "noise" from the signal, even when σ is initially unknown.*

Example 4.5.12.

Fifty-one equidistant values of a certain analytic function were rounded to four decimals. In Figure 4.5.?, a semilog diagram is given which shows haw $|c_i|$ varies in an expansion in terms of the Chebyshev polynomials for this data. For $i > 20$ (approximately) the contribution due to noise dominates the contribution due to signal. Thus it is sufficient to break off the series at $n = 20$.

Figure 4.5.1. *Magnitude of coefficients c_i in a Chebyshev expansion of an analytic function contaminated with roundoff noise.*

Review Questions

1. State the axioms that any norm must satisfy. Define the maximum norm and the Euclidean norm for a continuous function f on a closed interval.
2. Define $\text{dist}(f, P_n)$, and state Weierstrass' approximation theorem.
3. Prove the Pythagorean theorem in an inner product space.
4. Define and give examples of orthogonal systems of functions.
5. Formulate and prove Bessel's inequality and Parseval's identity, and interpret them geometrically.
6. (a) Give some reasons for using orthogonal polynomials in polynomial approximation with the method of least squares.

   (b) Give some argument against the assertion that orthogonal polynomials are difficult to work with.
7. The Gram polynomials are examples of orthogonal polynomials. With respect to what inner product are they orthogonal?

Problems and Computer Exercises

1. Compute $\|f\|_\infty$ and $\|f\|_2$ for the function $f(x) = (1+x)^{-1}$ on the interval $[0, 1]$.
2. Determine straight lines which approximate the curve $y = e^x$ such that
   (a) the discrete Euclidean norm of the error function on the grid $(-1, -0.5, 0, 0.5, 1)$ is as small as possible;

(b) the Euclidean norm of the error function on the interval $[-1, 1]$ is as small as possible.

(c) the line is tangent to $y = e^x$ at the point $(0, 1)$, i.e. the Taylor approximation at the midpoint of the interval.

3. Determine, for $f(x) = \pi^2 - x^2$, the "cosine polynomial" $f^* = \sum_{j=0}^{n} c_j \cos jx$, which makes $\|f^* - f\|_2$ on the interval $[0, \pi]$ as small as possible.

4. (a) Show that on any interval containing the points $-1, -1/3, 1/3, 1$,

$$E_2(f) \geq \frac{1}{8}\Big|f(1) - 3f(1/3) + 3f(-1/3) - f(-1)\Big|.$$

(b) Compute the above bound and the actual value of $E_2(f)$ for $f(x) = x^3$.

5. (a) Let a scalar product be defined by $(f, g) = \int_a^b f(x)g(x)\,dx$. Calculate the matrix of normal equations, when $\phi_j(x) = x^j$, $j = 0 : n$, when $a = 0$, $b = 1$.

(b) Do the same when $a = -1$, $b = 1$. Show how in this case the normal equations can be easily decomposed into two systems, with approximately $(n + 1)/2$ equations in each.

6. Verify the formulas for $\|\phi_j\|^2$ given in Example 4.5.9.

7. (a) Show that $\|f - g\|$ $ge\|f\| - \|g\|$ for all norms. (use the axioms mentioned in Sec. 4.5.1.)

(b) Show that if $\{c_j\}_0^n$ is a set of real numbers and if $\{f_j\}_0^n$ is a set of vectors, then $\|\sum c_j f_j\| \leq \sum |c_j| \|f_j\|$.

8. Let $G \in \mathbf{R}^{n\times n}$ be a symmetric positive definite matrix. Show that an inner product is defined by the formula $(u, v) = u^T G v$. Show that $A^* = G^{-1}A^T G$.

9. In a space of complex-valued twice differentiable functions of t, which vanish at t=0 and t=1, let the inner product be:

$$(u, v) = \int_0^1 u(t)\bar{v}(t)\,dt.$$

What is the adjoint of the operator $A = d/dt$? Is it true that the operator iA is self-adjoint, and that $-A^2$ is self-adjoint and positive definite?

10. a) Show that, in a real inner-product space,

$$4(u, v) = \|u + v\|^2 - \|u - v\|^2.$$

In a complex space this gives only the real part of the inner product. Show that one has to add $\|u - iv\|^2 - \|u + iv\|^2$.

(b) This can be used to reduce many questions concerning inner-products to questions concerning norms. For example, in a general inner product space a **unitary operator** is defined by the requirement that $\|Au\| = \|u\|$ $\forall u$. Show that $(Au, Av) = (u, v)$ $\forall u, v$.

Note, however, that the relation $(u, Au) = (Au, u)$ $\forall u$, which, in a real space, holds for every operator A, does *not* imply that $(u, Av) = (Au, v)$ $\forall u, v$. The latter holds only if A is self-adjoint.

11. Show that $(AB)^* = B^*A^*$. Also show that if C is self-adjoint and positive definite, then A^*CA is so too. (A is not assumed to be self-adjoint.)

12. Show that

$$(A^{-1})^* = ((A^*))^{-1}, \qquad (A^p)^* = ((A^*))^p,$$

for all integers p, provided that the operators mentioned exist. Is it true that C^p is self-adjoint and positive definite, if C is so?

13. Show the following **minimum property** of orthogonal polynomials: Among all $n$the degree polynomials p_n with leading coefficient 1, the smallest value of

$$\|p_n\|^2 = \int_a^b p_n^2(x)w(x)dx, \qquad w(x) \geq 0$$

is obtained for $p_n = \phi_n/A_n$, where ϕ_n is the orthogonal polynomial with leading coefficient A_n associated with the weight distribution $w(x)$.

Hint: Determine the best approximation to x^n in the above norm or consider the expansion $p_n = \phi_n/A_n + \sum_{j=0}^{n-1} c_j\phi_j$.

14. Verify the formulas for $\|T_j\|^2$ given in Theorem 4.5.17.

14. (a) Write a MATLAB function `c = stieltjes(x,y,w,n)` that computes the least squares polynomial fit to data (x_i, f_i) and weights w_i, $i = 1 : n$, using the Stieltjes procedure. Compute the orthogonal polynomials p_j, from the recursion (4.5.43) and the coefficients c_j, $j = 1 : n$, from (4.5.45) or (4.5.45).

(b) (L. F. Shampine) Apply the function in (a) to the case when $f_i = x_i^7$, $w_i = 1/(f_i)^2$, and $n = 20$. Compute and print the error $\|p - f\|$, for $n = 1 : 10$ using the expression (4.5.45) for c_{k+1}. Note that the fits for $k > 7$ should be exact!

(c) Repeat the calculations, now using the modified formula (4.5.45). Compare the error for $n = 1 : 10$ with the results in (a).

4.6 Trigonometric Interpolation and Fourier Transforms

Many natural phenomena, e.g., acoustical and optical, are of a periodic character. For instance, it is known that a musical sound is composed of regular oscillations, partly a fundamental tone with a certain frequency f, and partly overtones with frequencies $2f$, $3f$, $4f$,.... The ratio of the strength of the fundamental tone to that of the overtones is decisive for our impression of the sound. Sounds, which are free from overtones occur, for instance, in electronic music, where they are called pure sine tones.

In an electronic oscillator, a current is generated whose strength at time t varies according to the formula $r\sin(\omega t + v)$, where r is called the amplitude of the oscillation; ω is called the angular frequency, and is equal to 2π times the frequency; v is a constant which defines the state at the time $t = 0$. In a loudspeaker, variations of current are converted into variations in air pressure which, under ideal conditions, are described by the same function. In practice, however, there is always a certain

distortion, overtones occur. The variations in air pressure which reach the ear can, from this viewpoint, be described as a sum of the form

$$\sum_{k=0}^{\infty} r_k \sin(k\omega t + v_k). \tag{4.6.1}$$

The separation of a periodic phenomenon into a fundamental tone and overtones permeates not only acoustics, but also many other areas. It is related to an important, purely mathematical theorem, first given by Fourier[40] According to this theorem, every function $f(t)$ with period $2\pi/\omega$ can, under certain very general conditions, be expanded in a series of the form (4.6.1). (A function has period p if $f(t+p) = f(t)$, for all t.) A more precise formulation will be given later in Theorem 4.6.2.

An expansion of the form of (4.6.1) can be expressed in many equivalent ways. If we set $a_k = r_k \sin v_k$, $b_k = r_k \cos v_k$, then using the addition theorem for the sine function we can write

$$f(t) = \sum_{k=0}^{\infty} (a_k \cos k\omega t + b_k \sin k\omega t), \tag{4.6.2}$$

where a_k, b_k are real constants. Another form, which is often the most convenient, can be found with the help of Euler's formulas,

$$\cos x = \frac{1}{2}(e^{ix} + e^{-ix}), \qquad \sin x = \frac{1}{2i}(e^{ix} - e^{-ix}), \quad (i = \sqrt{-1}).$$

Then one gets

$$f(t) = \sum_{k=-\infty}^{\infty} c_k e^{ik\omega t}, \tag{4.6.3}$$

where

$$c_0 = a_0, \qquad c_k = \frac{1}{2}(a_k - ib_k), \qquad c_{-k} = \frac{1}{2}(a_k + ib_k), \quad k = 1, 2, 3, \ldots. \tag{4.6.4}$$

In the rest of this chapter we shall use the term **Fourier series** to denote an expansion of the form of (4.6.3) or (4.6.4). We shall call the partial sums of the form of these series **trigonometric polynomials**. Sometimes the term **spectral analysis** is used to describe the above methods.

Fourier series are valuable aids in the study of phenomena which are periodic in time (vibrations, sound, light, alternating currents, etc.) or in space (waves, crystal structure, etc.). One very important area of applications is in digital signal and image processing, which is used in interpreting radar and sonar signals. Another is statistical time series, which are used in communications theory, control theory, and the study of turbulence. For the numerical analyst, Fourier analysis is partly

[40] Jean Baptist Joseph Fourier (1768–1830), French mathematician and engineer. In 1807 Fourier presented before the French Academy his famous theorem.

a very common computational task and partly an important aid in the analysis of properties of numerical methods.

Basic formulas and theorems are derived in Sec. 4.6.1, which relies to a great extent on the theory in Sec. 4.5. Modifications of pure Fourier methods are used as a means of analyzing nonperiodic phenomena; see, e.g., Sec. 4.6.2 (periodic continuation of functions) and Sec. 4.6.3 (Fourier transforms). The approximation of Fourier transforms using sampled data and discrete Fourier transforms are treated in Sec. 4.6.4. FFT (Fast Fourier Transforms) algorithms have had an enormous impact and have caused a complete change of attitude toward what can be done using discrete Fourier methods. Sec. 4.7 treats the computational aspects of the basic FFT algorithms.

4.6.1 Basic Formulas and Theorems

We shall study *functions with period* 2π. If a function of t has period L, then the substitution $x = 2\pi t/L$ transforms the function to a function of x with period 2π. We assume that the function can have complex values, since the complex exponential function is convenient for manipulations.

The inner product of two complex-valued functions f and g of period 2π is defined in the following way (the bar over g indicates complex conjugation)

$$(f, g) = \int_{-\pi}^{\pi} f(x)\bar{g}(x)dx, \quad \text{(continuous case)}. \tag{4.6.5}$$

(It makes no difference what interval one uses, as long as it has length 2π—the value of the inner product is unchanged.) Often the function f is known only at equidistant arguments $x_\alpha = 2\pi\alpha/N$, $\alpha = 0 : N-1$. In this case we define

$$(f, g) = \sum_{\alpha=0}^{N-1} f(x_\alpha)\bar{g}(x_\alpha), \quad x_\alpha = \frac{2\pi\alpha}{N} \quad \text{(discrete case)}. \tag{4.6.6}$$

As usual the norm of the function f is defined by $\|f\| = (f, f)^{1/2}$. One can make computations with these inner products in the same way as with the inner products defined in Sec. 4.3.1, with certain obvious modifications. Notice especially that $(g, f) = \overline{(f, g)}$. In the continuous case,

Theorem 4.6.1.

The following **orthogonality relations** *hold for the functions*

$$\phi_j(x) = e^{ijx}, \quad j = 0, \pm 1, \pm 2, \ldots.$$

Continuous case:

$$(\phi_j, \phi_k) = \begin{cases} 2\pi, & \text{if } j = k, \\ 0, & \text{if } j \neq k. \end{cases}$$

Discrete case:

$$(\phi_j, \phi_k) = \begin{cases} N, & \text{if } (j-k)/N \text{ is an integer}, \\ 0, & \text{otherwise}. \end{cases}$$

Proof. In the continuous case, if $j \neq k$, it holds that

$$(\phi_j, \phi_k) = \int_{-\pi}^{\pi} e^{ijx} e^{-ikx} dx = \Big|_{-\pi}^{\pi} \frac{e^{i(j-k)x}}{i(j-k)} = \frac{(-1)^{j-k} - (-1)^{j-k}}{i(j-k)} = 0.$$

whereby orthogonality is proved. For $j = k$

$$(\phi_k, \phi_k) = \int_{-\pi}^{\pi} e^{ikx} e^{-ikx} dx = \int_{-\pi}^{\pi} 1 \; dx = 2\pi.$$

In the discrete case, set $h = 2\pi/N$, $x_\alpha = h\alpha$,

$$(\phi_j, \phi_k) = \sum_{\alpha=0}^{N-1} e^{ijx_\alpha} e^{-ikx_\alpha} = \sum_{\alpha=0}^{N-1} e^{i(j-k)h\alpha}.$$

This is a geometric series with ratio $q = e^{i(j-k)h}$. If $(j-k)/N$ is an integer, then $q = 1$ and the sum is N. Otherwise $q \neq 1$, but $q^N = e^{i(j-k)2\pi} = 1$. From the summation formula of a geometric series

$$(\phi_j, \phi_k) = (q^N - 1)/(q - 1) = 0.$$

□

If one knows that the function $f(x)$ has an expansion of the form

$$f = \sum_{j=a}^{b} c_j \phi_j,$$

where $a = -\infty$, $b = \infty$ in the continuous case and $a = 0$, $b = N - 1$ in the discrete case, then it follows formally that

$$(f, \phi_k) = \sum_{j=a}^{b} c_j (\phi_j, \phi_k) = c_k (\phi_k, \phi_k), \quad a \leq k \leq b,$$

since $(\phi_j, \phi_k) = 0$ for $j \neq k$. Thus, changing k to j, we have

$$c_j = \frac{(f, \phi_j)}{(\phi_j, \phi_j)} = \begin{cases} \dfrac{1}{2\pi} \displaystyle\int_{-\pi}^{\pi} f(x) e^{-ijx} dx, & \text{in the continuous case;} \\ \dfrac{1}{N} \displaystyle\sum_{\alpha=0}^{N-1} f(x_\alpha) e^{-ijx_\alpha}, & \text{in the discrete case).} \end{cases} \tag{4.6.7}$$

These coefficients are called **Fourier coefficients**, see the more general case in Theorem 4.3.??. The purely formal treatment given above is, in the discrete case, justified by Theorem 4.3.??. In the continuous case, more advanced methods are required, but we shall not go into this further.

Now consider the *continuous* case. From a generalization of Theorem 4.3.??, we know that

$$\Big\| f - \sum_{j=-n}^{n} k_j \phi_j \Big\|, \quad n < \infty,$$

becomes as small as possible if we choose $k_j = c_j$, $-n \le j \le n$.

In accordance with (4.6.4), set $a_j = c_j + c_{-j}$, $b_j = i(c_j - c_{-j})$. Then with $a_0 = 2c_0$,

$$a_j = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos jx\,dx, \qquad b_j = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin jx\,dx, \quad j \ge 0, \tag{4.6.8}$$

$$\begin{aligned}\sum_{j=-N}^{N} c_j e^{ijx} &= c_0 + \sum_{j=1}^{N}\big(c_j(\cos jx + i\sin jx) + c_{-j}(\cos jx - i\sin jx)\big)\\ &= \frac{1}{2}a_0 + \sum_{j=1}^{N}(a_j\cos jx + b_j\sin jx).\end{aligned}$$

(Notice that the factors preceding the integral are different in the expressions for c_j and for a_j, b_j, respectively.)

Theorem 4.6.2. *Fourier Analysis, Continuous Case.*

Every piecewise continuous function f with period 2π can be associated with a Fourier series in the following two ways:

$$\frac{1}{2}a_0 + \sum_{j=1}^{\infty}(a_j\cos jx + b_j\sin jx),$$

$$\sum_{j=-\infty}^{\infty} c_j e^{ijx}.$$

The coefficients a_j, b_j, and c_j can be computed using (4.6.8) *in the first case and* (4.6.7) *in the second case. If f and its first derivative are everywhere continuous, then the series is everywhere convergent to $f(x)$. If f and f' have a finite number of jump discontinuities in each period, the series gives the mean of the limiting values on the right and on the left of the relevant point. The partial sums of the above expansions give the best possible approximations to $f(x)$ by trigonometric polynomials, in the least squares sense.*

The proof of the convergence results is outside the scope of this book (see, however, the beginning of Sec. 4.7; see also Courant and Hilbert [16].) The rest of the assertions follow from previously made calculations in Theorem 4.6.1 and the comments following; see also the proof of Theorem 4.3.??.

The more regular a function is, the faster its Fourier series converges.

Theorem 4.6.3.
If f and its derivatives up to and including order k are periodic and everywhere continuous, and if $f^{(k+1)}$ is piecewise continuous, then

$$|c_j| \le j^{-(k+1)} \|f^{(k+1)}\|_\infty. \tag{4.6.9}$$

This useful result is relatively easy to prove using (4.6.7) and integrating by parts $k+1$ times.

Theorem 4.6.4.
If f is an even function, i.e. if

$$f(x) = f(-x) \quad \forall x,$$

then $b_j = 0$ for all j; thus the Fourier series becomes a cosine series.
If f is an odd function, i.e. if

$$f(x) = -f(-x) \quad \forall x,$$

then $a_j = 0$ for all j; thus the Fourier series becomes a sine series.
The proof is left as an exercise to the reader (use the formulas in (4.6.10)).

Example 4.6.1. *Fourier Expansion of a Rectangular Wave*
Make a periodic continuation outside the interval $(-\pi, \pi)$ of the function

$$f(x) = \begin{cases} -1, & -\pi < x < 0, \\ 1, & 0 < x < \pi, \end{cases},$$

see Figure 4.6.3.

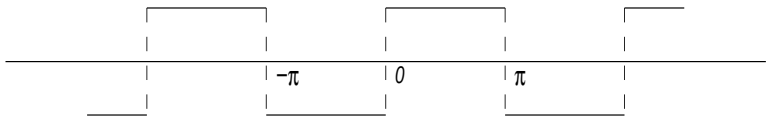

Figure 4.6.1. *Rectangular wave*

The function f is odd, so $a_j = 0$ for all j, and

$$b_j = \frac{2}{\pi} \int_0^\pi \sin jx \, dx = \frac{2}{j\pi}(1 - \cos j\pi).$$

Hence $b_j = 0$ if j is even, and $b_j = 4/(j\pi)$ if j is odd, and

$$f(x) = \frac{4}{\pi}\Big(\sin x + \frac{\sin 3x}{3} + \frac{\sin 5x}{5} + \cdots\Big).$$

Notice that the coefficients c_j decay as j^{-1} in agreement with Theorem 4.6.3. The sum of the series is zero at the points where f has a jump discontinuity; this agrees with the fact that the sum should equal the average of the limiting values to the left and to the right of the discontinuity. The Euler transformation can be used for accelerating the convergence of such series, except in the immediate vicinity of singular points, see Problem 3 of Sec. 4.6.

Theorem 4.5.13 and its corollary, Parseval's identity

$$2\pi \sum_{j=-\infty}^{\infty} |c_j|^2 = \|f\|^2 = \int_{-\pi}^{\pi} |f(x)|^2 \, dx, \tag{4.6.10}$$

are of great importance in many applications of Fourier analysis. The integral in (4.6.10) can be interpreted as the "energy" of the function $f(x)$.

Although the data to be treated in Fourier analysis are often continuous in the time or space domain, for computational purposes this data must usually be represented in terms of a finite discrete sequence. For example, a function $f(t)$ of time, is recorded at evenly spaced intervals Δ in time $f_i = f(i\Delta)$, $i = 0, 1, 2, \ldots$. Such data can be analyzed by discrete Fourier analysis.

Theorem 4.6.5. *Fourier Analysis, Discrete Case*

Every function, defined on the equidistant grid $\{x_0, x_1, \ldots, x_{N-1}\}$, *where* $x_\alpha = 2\pi\alpha/N$, *can be interpolated by the trigonometric polynomial*

$$f(x) = \begin{cases} \displaystyle\sum_{j=-k}^{k+\theta} c_j e^{ijx}, \\ \displaystyle\tfrac{1}{2}a_0 + \sum_{j=1}^{k}(a_j \cos jx + b_j \sin jx) + \frac{1}{2}\theta a_{k+1}\cos(k+1)x, \end{cases} \tag{4.6.11}$$

Here

$$\theta = \begin{cases} 1, & \text{if } N \text{ even}, \\ 0, & \text{if } N \text{ odd}, \end{cases}, \qquad k = \begin{cases} N/2 - 1, & \text{if } N \text{ even}, \\ (N-1)/2, & \text{if } N \text{ odd}, \end{cases} \tag{4.6.12}$$

and

$$c_j = \frac{1}{N}\sum_{\alpha=0}^{N-1} f(x_\alpha) e^{-ijx_\alpha}, \tag{4.6.13}$$

$$a_j = \frac{2}{N}\sum_{\alpha=0}^{N-1} f(x_\alpha)\cos jx_\alpha, \qquad b_j = \frac{2}{N}\sum_{\alpha=0}^{N-1} f(x_\alpha)\sin jx_\alpha. \tag{4.6.14}$$

If the sums in (4.6.11) *are terminated when* $|j| < k + \theta$, *then one obtains the trigonometric polynomial which is the best least squares approximation, among all trigonometric polynomials with the same number of terms, to* f *on the grid.*

Proof. The expression for c_j was formally derived previously, see (4.6.7), and the derivation is justified by Theorem 4.3.??. By Eqs. (4.6.13)–(4.6.14)

$$a_j = c_j + c_{-j}, \qquad b_j = i(c_j - c_{-j}), \qquad c_{k+1} = \frac{1}{2}a_{k+1}.$$

The two expressions for $f(x)$ are equivalent, because

$$\begin{aligned}\sum_{j=-k}^{k+\theta} c_j e^{ijx} &= c_0 + \sum_{j=1}^{k}\big(c_j(\cos jx + i\sin jx) + c_{-j}(\cos jx - i\sin jx)\big) \\ &\quad + \theta c_{k+1}\cos(k+1)x \\ &= c_0 + \sum_{j=1}^{k}(a_j\cos jx + b_j\sin jx) + \frac{1}{2}\theta a_{k+1}\cos(k+1)x.\end{aligned}$$

□

The function $f(x)$ coincides *on the grid* with the function

$$f^*(x) = \sum_{j=0}^{N-1} c_j e^{ijx}, \tag{4.6.15}$$

because $e^{-i(N-j)x_\alpha} = e^{ijx_\alpha}$, $c_{-j} = c_{N-j}$. The functions f and f^* are, however, not identical between the grid points.

Notice that the calculations required to compute the coefficients c_j according to (4.6.13), **Fourier analysis**, are of essentially the same type as the calculations needed to compute $f^*(x)$ at the grid points

$$x_\alpha = 2\pi\alpha/N, \quad \alpha = 0 : N-1,$$

when the expansion in (4.6.15) is known, so-called **Fourier synthesis**. Both calculations can be performed very efficiently using FFT algorithms; see Sec. 4.7.

Functions of several variables are treated analogously. Quite simply, one takes one variable at a time. As an example, consider the discrete case, with two variables. Set

$$x_\alpha = 2\pi\alpha/N, \qquad y_\beta = 2\pi\beta/N,$$

and assume that $f(x_\alpha, y_\beta)$ is known for $\alpha = 0 : N-1$, $\beta = 0 : N-1$. Set

$$c_j(y_\beta) = \frac{1}{N}\sum_{\alpha=0}^{N-1} f(x_\alpha, y_\beta)e^{-ijx_\alpha},$$

$$c_{j,k} = \frac{1}{N}\sum_{\beta=0}^{N-1} c_j(y_\beta)e^{-iky_\beta}.$$

From Theorem 4.6.5, then (with obvious changes in notations),

$$c_j(y_\beta) = \sum_{k=0}^{N-1} c_{j,k} e^{iky_\beta},$$

$$f(x_\alpha, y_\beta) = \sum_{j=0}^{N-1} c_j(y_\beta) e^{ijx_\alpha} = \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} c_{j,k} e^{(ijx_\alpha + iky_\beta)}.$$

The above expansion is of considerable importance, e.g., in crystallography.

4.6.2 Periodic Continuation of a Function

Sometimes Fourier series are used for functions which are defined only on the interval $(-\pi, \pi)$. The series then defines a *periodic continuation of the function* outside the interval (see Figure 4.6.2). Thus the definition is extended so that $f(x) = f(x+2\pi)$ for all x.

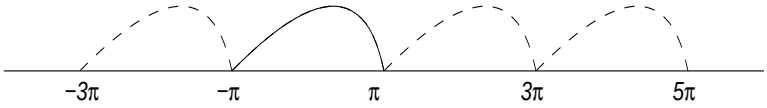

Figure 4.6.2. *Periodic continuation of a function outside* $[-\pi, \pi]$.

With this method, discontinuities in the values of the function or its derivatives can occur at $x = \pi$. This singularity can give rise to a slow rate of convergence, even for a function with good regularity properties in the open interval $(-\pi, \pi)$; cf. Theorem 4.6.3.

There are other ways to make a continuation of a function outside its interval of definition. If the function is defined in $[0, \pi]$, and if $f(0) = f(\pi) = 0$, then one can continue f to the interval $[-\pi, 0]$ by making the definition $f(x) = f(-x)$; thereafter the function is periodically continued outside $[-\pi, \pi]$ by $f(x) = f(x+2\pi)$, see Figure 4.6.3. Since the resulting function is an *odd* function, by Theorems 4.6.4

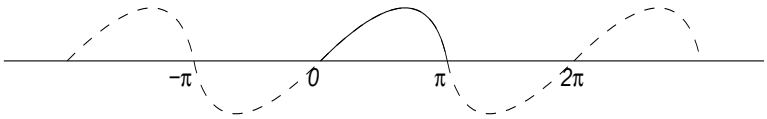

Figure 4.6.3. *Periodic continuation of* f *outside* $[0, \pi]$ *as an odd function.*

and 4.6.5, its Fourier expansion becomes a sine series:

Continuous case:

$$\sum_{j=1}^{\infty} b_j \sin jx, \qquad b_j = \frac{2}{\pi} \int_0^\pi f(x) \sin jx \, dx. \tag{4.6.16}$$

Discrete case: $(x_\alpha = \pi\alpha/N)$

$$\sum_{j=1}^{N-1} b_j \sin jx, \qquad b_j = \frac{2}{N}\sum_{\alpha=1}^{N-1} f(x_\alpha)\sin jx_\alpha. \tag{4.6.17}$$

If $f(0) \neq 0$ or $f(\pi) \neq 0$, one can still use such an expansion, but it will converge slowly.

If the function is defined in $[0, \pi]$, and if $f'(0) = f'(\pi) = 0$, then one can make a continuation of f into an *even* function on $[-\pi, \pi]$. Outside $[-\pi, \pi]$ the function is continued periodically. Its Fourier series then becomes a pure cosine series:

Continuous case:

$$\frac{1}{2}a_0 + \sum_{j=1}^{\infty} a_j \cos jx, \qquad a_j = \frac{2}{\pi}\int_0^\pi f(x)\cos jx\, dx, \tag{4.6.18}$$

Discrete case: $(x_\alpha = \pi\alpha/N$

$$\frac{1}{2}a_0 + \sum_{j=1}^{N-1} a_j \cos jx, \qquad a_j = \frac{2}{N}\sum_{\alpha=0}^{N-1} f(x_\alpha)\cos jx_\alpha. \tag{4.6.19}$$

4.6.3 The Fourier Integral Theorem

In Sec. 4.6.2 we showed how Fourier methods can be used on a nonperiodic function defined on a finite interval. Suppose now that the function $f(x)$ is defined on the entire real axis, and that it satisfies the regularity properties which we required in Theorem 4.6.2. Set

$$\varphi(\xi) = f(x), \quad \xi = 2\pi x/L \in [-\pi, \pi],$$

and continue $\varphi(\xi)$ outside $[-\pi, \pi]$ so that it has period 2π. By Theorem 4.6.2, if

$$c_j = \frac{1}{2\pi}\int_{-\pi}^{\pi} \varphi(\xi)e^{-ij\xi}\, d\xi = \frac{1}{L}\int_{-L/2}^{L/2} f(x)e^{-2\pi ixj/L}\, dx, \tag{4.6.20}$$

then $\varphi(\xi) = \sum_{j=-\infty}^{\infty} c_j e^{ij\xi}$, $\xi \in (-\pi, \pi)$, and hence

$$f(x) = \sum_{j=-\infty}^{\infty} c_j e^{2\pi ixj/L}, \quad x \in (-L/2, L/2).$$

If we set

$$g_L(\omega) = \int_{-L/2}^{L/2} f(x)e^{-2\pi ix\omega}\, dx, \quad \omega = j/L, \tag{4.6.21}$$

then by (4.6.20) we have $c_j = (1/L)g_L(\omega)$, and hence

$$f(x) = \frac{1}{L}\sum_{j=-\infty}^{\infty} g_L(\omega)e^{2\pi ix\omega}, \quad x \in (-L/2, L/2). \tag{4.6.22}$$

Now by passing to the limit $L \to \infty$, one avoids making an artificial periodic continuation outside a finite interval. The sum in (4.6.22) is a "sum of rectangles" similar to the sum which appears in the definition of a definite integral. However, here the argument varies from $-\infty$ to $+\infty$, and the function $g_L(t)$ depends on L. By a somewhat dubious passage to the limit, then, the pair of formulas of (4.6.21) and (4.6.22) becomes the pair

$$g(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ix\omega}\,dx \iff f(x) = \int_{-\infty}^{\infty} g(\omega)e^{2\pi ix\omega}\,d\omega. \tag{4.6.23}$$

One can, in fact, after a rather complicated analysis, show that the above result is correct; see, e.g., Courant–Hilbert [16]. The proof requires, besides the previously mentioned "local" regularity conditions on f, the "global" assumption that

$$\int_{-\infty}^{\infty} |f(x)|\,dx$$

is convergent. The beautiful, almost symmetric relation of (4.6.23) is called the **Fourier integral theorem**. This theorem, and other versions of it, with varying assumptions under which they are valid, is one of the most important aids in both pure and applied mathematics. The function g is called the **Fourier transform**[41] of f.

Clearly the Fourier transform is a *linear operator*. Two other elementary properties that can easily be verified are:

$$f(ax) \iff \frac{1}{|a|}g(\omega/a), \tag{4.6.24}$$

$$\frac{1}{|b|}f(x/b) \iff g(b\omega). \tag{4.6.25}$$

If the function $f(x)$ has even or odd symmetry and is real or pure imaginary this leads to relations between $g(\omega)$ and $g(-\omega)$ that can be used to increase computational efficiency. Some of these properties are summarized in the table below.

Example 4.6.2.

The function $f(x) = e^{-|x|}$ has Fourier transform

$$\begin{aligned} g(\omega) &= \int_{-\infty}^{\infty} e^{-|x|}e^{-2\pi ix\omega}\,dx = \int_0^{\infty} \left(e^{-(1+2\pi i\omega)x} + e^{-(1-2\pi i\omega)x}\right) dx \\ &= \frac{1}{1+2\pi i\omega} + \frac{1}{1-2\pi i\omega} = \frac{2}{1+4\pi^2\omega^2}. \end{aligned}$$

Here $f(x)$ is real and an even function. In agreement with the table above the Fourier transform is also real and even.

[41] The terminology in the literature varies somewhat as to the placement of the factor 2π—e.g., it can be taken out of the exponent by a simple change of variable.

Function	Fourier transform
$f(x)$ real	$g(-\omega) = \overline{g}(\omega)$
$f(x)$ imaginary	$g(-\omega) = -\overline{g}(\omega)$
$f(x)$ even	$g(-\omega) = g(\omega)$
$f(x)$ odd	$g(-\omega) = -g(\omega)$
$f(x)$ real even	$g(\omega)$ real even
$f(x)$ imaginary odd	$g(\omega)$ real odd

Table 4.6.1. *Useful symmetry properties of Fourier transforms.*

From (4.6.23) it follows that

$$e^{-|x|} = \int_{-\infty}^{\infty} \frac{2}{1+4\pi^2\omega^2} e^{2\pi i x\omega}\, d\omega = \frac{2}{\pi}\int_0^{\infty} \frac{1}{1+x^2}\cos \pi x\, dx, \quad (2\pi\omega = x).$$

It is not so easy to prove this formula directly.

Many applications of the Fourier transform involve the use of convolutions.

Definition 4.6.6.
The **convolution** *of f_1 and f_2 is the function*

$$h(\xi) = conv\,(f_1, f_2) = \int_{-\infty}^{\infty} f_1(x) f_2(\xi - x)\, dx. \tag{4.6.26}$$

It is not difficult to verify that $conv\,(f_1, f_2) = conv\,(f_2, f_1)$. The following theorem states that the convolution of f_1 and f_2 can be computed as the inverse Fourier transform of the product $g_1(\omega)g_2(\omega)$. This fact is of great importance in the application of Fourier analysis, e.g., to differential equations and probability theory.

Theorem 4.6.7.
Let f_1 and f_2 have Fourier transforms g_1 and g_2, respectively. Then the Fourier transform g of the convolution of f_1 and f_2, is the product $g(\omega) = g_1(\omega)g_2(\omega)$.

Proof. By definition the Fourier transform of the convolution is

$$\begin{aligned} g(\omega) &= \int_{-\infty}^{\infty} e^{-2\pi i\xi\omega} \left(\int_{-\infty}^{\infty} f_1(x) f_2(\xi - x)\, dx \right) d\xi \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} e^{-2\pi i(x+\xi-x)\omega} f_1(x) f_2(\xi - x)\, dx\, d\xi \\ &= \int_{-\infty}^{\infty} e^{-2\pi i x\omega} f_1(x)\, dx \int_{-\infty}^{\infty} e^{-2\pi i(\xi-x)\omega} f_2(\xi - x)\, d\xi \\ &= \int_{-\infty}^{\infty} e^{-2\pi i x\omega} f_1(\xi)\, dx \int_{-\infty}^{\infty} e^{-2\pi i x\omega} f_2(x)\, dx = g_1(\omega)g_2(\omega) \end{aligned}$$

The legitimacy of changing the order of integration is here taken for granted. □

In many physical applications, the following relation, analogous to Parseval's identity (corollary to Theorem 4.5.13), is of great importance. If g is the Fourier transform of f, then

$$\int_{-\infty}^{\infty} |g(\omega)|\, d\omega = \int_{-\infty}^{\infty} |f(\xi)|\, d\xi. \tag{4.6.27}$$

In signal processing this can be interpreted to mean that the total power in a signal is the same whether computed in the time domain or the frequency domain.

4.6.4 Sampled Data and Aliasing

Consider a function $f(x)$ which is zero outside the interval $[0, L]$. The Fourier transform of $f(x)$ is then given by

$$g(\omega) = \int_0^L f(x) e^{-2\pi i \omega x}\, dx. \tag{4.6.28}$$

We want to approximate $g(\omega)$ using values of $f(x)$ sampled at intervals Δx,

$$f_j = f(j\Delta x), \quad 0 < j < N-1, \quad L = N\Delta x.$$

The integral (4.6.28) can be approximated by

$$g(\omega) \approx \frac{L}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi i \omega j \Delta x}\, dx. \tag{4.6.29}$$

Since only N values of f_j are used as input and we want the computed values to be linearly independent, we cannot approximate $g(\omega)$ at more than N points. The wave of lowest frequency associated with the interval $[0, L]$ is $\omega = 1/L = 1/(N\Delta x)$, since then $[0, L]$ corresponds to one full period of the wave. We therefore choose in the frequency space points $\omega_k = k\Delta\omega$, $i = 0 : N$, such that the following **reciprocity relations** hold:

$$LW = N, \qquad \Delta x \Delta\omega = 1/N \tag{4.6.30}$$

With this choice it holds that

$$W = N\Delta\omega = 1/\Delta x, \qquad L = N\Delta x = 1/\Delta\omega. \tag{4.6.31}$$

Noting that $(j\Delta x)(k\Delta\omega) = jk/N$ we get from the trapezoidal approximation

$$g(\omega_k) \approx \frac{L}{N} \sum_{j=0}^{N-1} f_j e^{-2\pi i k j/N}\, dx = L c_k, \quad k = 0 : N-1,$$

where c_k is the coefficient of the discrete Fourier transform.

The frequency $\omega_c = 1/(2\Delta x) = \Delta\omega/2$ is the so called **Nyquist critical frequency**. Sampling the wave $\sin(2\pi\omega_c x)$ with sampling interval Δx will sample

exactly *two points per cycle*. It is a remarkable fact that if a function $f(x)$, defined on $[-\infty, \infty]$, is *band-width limited* to frequencies smaller or equal to ω_c, then $f(x)$ is completely determined by its sample values $j\Delta x$, $\infty \leq j \leq \infty$; see the Sampling Theorem Sec. 4.8.3.

If the function is not band-width limited the spectral density outside the critical frequency is moved into that range. This is called **aliasing**. The relationship between the Fourier transform $g(\omega)$ and the discrete Fourier transform of a finite sampled representation can be characterized as follows. Assuming that the reciprocity relations (4.6.30) are satisfied, the discrete Fourier transform of $f_j = \tilde{f}(j\Delta x)$, $0 \leq j < N$, will approximate the periodic aliased function

$$\tilde{g}_k = \tilde{g}(k\Delta\omega), \quad 0 \leq j < N. \tag{4.6.32}$$

where

$$\tilde{g}(\omega) = g(\omega) + \sum_{k=1}^{\infty} \left(g(\omega + kW) + g(\omega - kW) \right), \quad \omega \in [0, W] \tag{4.6.33}$$

Since by (4.6.31) $W = 1/\Delta x$, we can *increase* the frequency range $[0, W]$ covered by *decreasing* Δx.

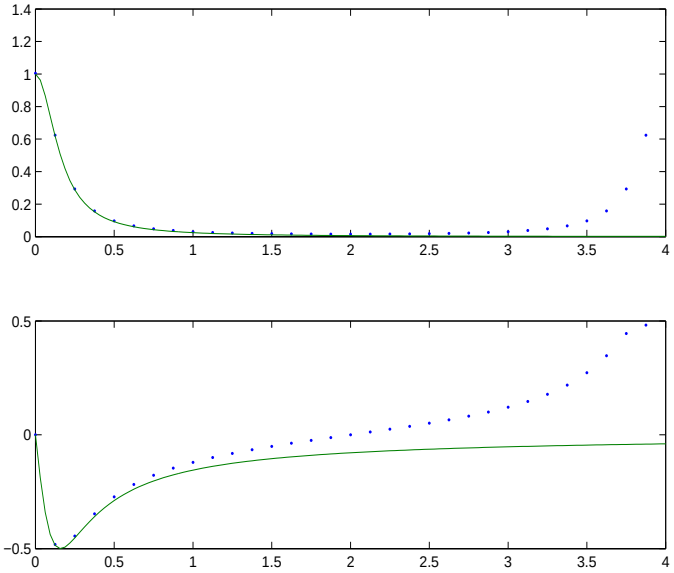

Figure 4.6.4. *The real (top) and imaginary (bottom) parts of the Fourier transform of e^{-x} and the corresponding DFT with $N = 32$, $T = 8$.*

Example 4.6.3.

The function $f(x) = e^{-x}$, $x > 0$, $f(x) = 0$, $x < 0$, has Fourier transform

$$g(\omega) = 1/(1 + 2\pi i\omega) = \frac{1 - i2\pi\omega}{1 + 4\pi^2\omega^2}$$

(cf. Example 4.6.2. Set $f(0) = 1/2$, the average of $f(-0)$ and $f(+0)$, which is the value given by the inverse Fourier transform at a discontinuity.

Set $N = 32$, $T = 8$, and sample the f in the interval $[0, T]$, at equidistant points $j\Delta x$, $j = 0 : N - 1$. Note that T is so large that the aliased function (4.6.32) is nearly equal to f. This sampling rate corresponds to $\Delta x = 8/32 = 1/4$ and $W = 4$.

The effect of aliasing in the frequency domain is evident. The error is significant for frequencies larger than the critical frequency $W/2$. To increase the accuracy W can be increased by decreasing the sampling interval Δx.

Review Questions

1. Derive the orthogonality properties and coefficient formulas which are fundamental to Fourier analysis, for both the continuous and the discrete case.
2. Under what conditions does the Fourier series of the function f converge to f in the continuous case?
3. How does one compute a Fourier expansion of a function of two variables?
4. Explain what a periodic continuation of a function is. What disadvantage for Fourier analysis is incurred if the periodic continuation has a discontinuity in its function value or derivatives at certain points?
5. Formulate the Fourier integral theorem.
6. (a) Explain what a periodic continuation of a function is.
   (b) What disadvantage (for Fourier analysis) is incurred if the periodic continuation has a discontinuity—e.g., in its derivative at certain points?

Problems and Computer Exercises

1. Give a simple characterization of the functions which have a sine expansion containing odd terms only.
2. Let f be an even function, with period 2π, such that

$$f(x) = \pi - x, \quad 0 \le x \le \pi.$$

   (a) Plot the function $y = f(x)$ for $-3\pi \le x \le 3\pi$. Expand f in a Fourier series.
   (b) Use this series to show that $1 + 3^{-2}] + 5^{-2}] + 7^{-2}] + \cdots = \pi^2/8$.
   (c) Compute the sum $1 + 2^{-2}] + 3^{-2}] + 4^{-2}] + 5^{-2}] + \cdots$.

(d) Compute, using (4.6.10), the sum $1 + 3^{-4}] + 5^{-4}] + 7^{-4}] + \cdots$.

(e) Differentiate the Fourier series term by term, and compare with the result in Example 4.6.1.

3. Show that the function $G_1(t) = t - 1/2$, $0 < t < 1$, has the expansion

$$G_1(t) = -\sum_{n=1}^{\infty} \frac{\sin 2n\pi t}{n\pi}.$$

Derive by term-wise integration, the expansion for the functions $G_p(t)$, and show the statement (made in Sec. 10.3.1) that $c_p - G_p(t)$ has the same sign as c_p. Show also that $\sum_{n=1}^{\infty} n^{-p} = \frac{1}{2}|c_p|(2\pi)^p$, p even.

4. (a) Prove that

$$\sum_{k=1}^{N-1} \sin \frac{\pi k}{N} = \cot \frac{\pi}{2N}.$$

Hint: $\sin x$ is the imaginary part of e^{ix}.

(b) Determine a sine polynomial $\sum_{j=1}^{n-1} b_j \sin jx$, which takes on the value 1 at the points $x_\alpha = \pi\alpha/n$, $\alpha = 1 : n - 1$.

Hint: Use (4.6.16) or recall that the sine polynomial is an odd function.

(c) Compare the limiting value for b_j as $n \to \infty$ with the result in Example 9.6.2.

5. (a) Prove the inequality in (4.6.9)!

(b) Show, under the assumptions on f which hold in (4.6.9), that, for $k \geq 1$, f can be approximated by a trigonometric polynomial such that

$$\Big\| f - \sum_{j=-n}^{n} c_j e^{ijx} \Big\|_\infty < \frac{2}{kn^k} \|f^{(k+1)}\|_\infty.$$

In the following problems, we do not require any investigation of whether it is permissible to change the order of summations, integrations, differentiations, etc.; it is sufficient to treat the problems in a purely formal way.

6. The partial differential equation $\dfrac{\partial u}{\partial t} = \dfrac{\partial^2 u}{\partial x^2}$ is called the heat equation. Show that the function

$$u(x,t) = \frac{4}{\pi} \sum_{k=0}^{\infty} \frac{\sin(2k+1)x}{2k+1}\, e^{-(2k+1)^2 t},$$

satisfies the differential equation for $t > 0$, $0 < x < \pi$, with boundary conditions $u(0,t) = u(\pi,t) = 0$ for $t > 0$, and initial condition $u(x,0) = 1$ for $0 < x < \pi$ (see Example 4.6.1).

7. Show that if $g(t)$ is the Fourier transform of $f(x)$, then

(a) $e^{2\pi\alpha t}]g(t)$ is the Fourier transform of $f(x + \alpha)$.

(b) $(2\pi it)^k g(t)$ is the Fourier transform of $f^{(k)}](x)$, assuming that $f(x)$ and its derivatives up to the kth order tend to zero, as $x \to \infty$.

8. The **correlation** of $f_1(x)$ and $f_2(x)$ is defined by

$$c(\xi) = \int_{-\infty}^{\infty} \phi_1(x+\xi)\phi_2(x)\,dx. \tag{4.6.34}$$

Show that if $f_1(x)$ and $f_2(x)$ have Fourier transforms $g_1(t)$ and $g_2(t)$, respectively, then the Fourier transform of $c(\xi)$ is $h(t) = g_1(t)g_2(-t)$.

Hint: Compare Theorem 4.6.7.

9. Derive Parseval's identities (4.6.27) and (4.6.10)

4.7 The Fast Fourier Transform

4.7.1 The Fast Fourier Algorithm

Consider the problem to compute the discrete Fourier coefficients $\{c_j\}_{j=0}^{N-1}$

$$f(x) = \sum_{j=0}^{N-1} c_j e^{ijx},$$

for a function, whose values f_α are known at the points $x_\alpha = 2\pi\alpha/N$, $\alpha = 0 : N-1$. According to Theorem 4.6.5

$$c_j = \frac{1}{N}\sum_{\alpha=0}^{N-1} f_\alpha e^{-ijx_\alpha}, \quad j = 0 : N-1.$$

Setting $\omega_N = e^{-2\pi i/N}$ (i.e. ω is an Nth root of unity, $\omega^N = 1$), we can rewrite the problem as follows: compute

$$c_j = \frac{1}{N}\sum_{\alpha=0}^{N-1} \omega_N^{j\alpha} f_\alpha, \quad j = 0 : N-1. \tag{4.7.1}$$

It seems from (4.7.1) that to compute the discrete Fourier coefficients would require N^2 complex multiplications and additions. As we shall see, with the **Fast Fourier Transform** (**FFT**) one needs only about $N \log_2 N$ complex multiplications and additions if $N = 2^k$. For example, when $N = 2^{20} = 1\,048\,576$ the FFT algorithm is theoretically a factor of 84 000 faster than the "conventional" $O(N^2)$ algorithm. On a 266 MHz Pentium laptop, a real FFT of this size takes about 1.2 seconds using MATLAB 6, whereas 28 hours would be required by the conventional algorithm! The FFT not only uses fewer operations to evaluate the DFT, it also is more accurate. Whereas using the conventional method the roundoff error is proportional to N, for the FFT algorithm it is proportional to $\log_2 N$.

In many areas of application (digital signal and image processing, time-series analysis, to name a few) the FFT has caused a complete change of attitude toward what can be done using discrete Fourier methods. Without the FFT many modern devices like cell phones, digital cameras, CAT scans and DVDs would not be

possible. Some future applications considered in astronomy are expected to require FFTs of several gigapoints

Similar ideas were used already by Gauss and several other mathematicians, e.g., Danielson and Lanczos [18].[42].

In the following we will use the common convention *not* to scale the sum in (4.7.1) by $1/N$.

Definition 4.7.1.

The **Discrete Fourier Transform (DFT)** *of the vector* $f \in \mathbf{C}^N$ *is*

$$y = F_N f. \tag{4.7.2}$$

where $F_n \in \mathbf{R}^{N\times N}$ *is the DFT matrix with elements*

$$(F_N)_{j\alpha} = \omega_N^{j\alpha}, \quad j, \alpha = 0 : N-1, \tag{4.7.3}$$

where $\omega_N = e^{-2\pi i/N}$.

From the definition it follows that the Fourier matrix F_N is a complex Vandermonde matrix. Since $\omega_N^{j\alpha} = \omega_N^{\alpha j}$, F_N is symmetric. By Theorem 4.6.5

$$\frac{1}{N} F_N^* F_N = I,$$

where F_N^* is the complex conjugate transpose of F_N. Hence the inverse transform can be written

$$f = \frac{1}{N} F_N^* y.$$

Example 4.7.1.

For $n = 2^2 = 4$, the DFT matrix is

$$F_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega_4 & \omega_4^2 & \omega_4^3 \\ 1 & \omega_4^2 & \omega_4^4 & \omega_4^6 \\ 1 & \omega_4^3 & \omega_4^6 & \omega_4^9 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}. \tag{4.7.4}$$

where $\omega_4 = e^{-2\pi i/4}$, and $\omega_4^4 = 1$.

We now describe the central idea of the FFT algorithm. Let $N = 2^k$ and set

$$\alpha = \begin{cases} 2\alpha_1, & \text{if } \alpha \text{ is even,} \\ 2\alpha_1 + 1 & \text{if } \alpha \text{ is odd,} \end{cases}, \quad 0 \le \alpha_1 \le m-1.$$

[42]The modern usage of FFT started in 1965 with the publication of the [15] by James W. Cooley of IBM Research and John W. Tukey, Princeton University. Tukey came up with the basic algorithm at a meeting of President Kennedy's Science Advisory Committee. One problem discussed at this meeting was that the ratification of a US–Sovjet Union nuclear test ban depended on a fast method to detect nuclear tests by analyzing seismological time-series data.

where $m = N/2 = 2^{k-1}$. Split the DFT sum in an even and an odd part

$$y_j = \sum_{\alpha_1=0}^{m-1} (\omega_N^2)^{j\alpha_1} f_{2\alpha_1} + \omega_N^j \sum_{\alpha_1=0}^{m-1} (\omega_N^2)^{j\alpha_1} f_{2\alpha_1+1}, \quad j = 0 : N-1,$$

Let β be the quotient and j_1 the remainder when j is divided by m, i.e. $j = \beta m + j_1$. Then, since $\omega_N^N = 1$,

$$(\omega_N^2)^{j\alpha_1} = (\omega_N^2)^{\beta m \alpha_1} (\omega_N^2)^{j_1\alpha_1} = (\omega_N^N)^{\beta\alpha_1} (\omega_N^2)^{j_1\alpha_1} = \omega_m^{j_1\alpha_1}.$$

Thus if, for $j_1 = 0 : m-1$, we set

$$\phi_{j_1} = \sum_{\alpha_1=0}^{m-1} f_{2\alpha_1} \omega_m^{j_1\alpha_1}, \qquad \psi_{j_1} = \sum_{\alpha_1=0}^{m-1} f_{2\alpha_1+1} \omega_m^{j_1\alpha_1}. \tag{4.7.5}$$

then, $y_j = \phi_{j_1} + \omega_N^j \psi_{j_1}$, The two sums on the right are elements of the DFTs of length $N/2$ applied to the parts of f with odd and even subscripts. *The entire DFT of length N is obtained by combining these two DFTs!* Since $\omega_N^m = -1$ we have

$$y_{j_1} = \phi_{j_1} + \omega_N^{j_1} \psi_{j_1}, \tag{4.7.6}$$

$$y_{j_1+N/2} = \phi_{j_1} - \omega_N^{j_1} \psi_{j_1}, \quad j_1 = 0 : N/2 - 1. \tag{4.7.7}$$

These expressions, noted already by Danielson and Lanczos [18], are often called **butterfly relations** because of the data flow pattern. Note that these can be performed in place, i.e. no extra vector storage is needed.

The computation of ϕ_{j_1} and ψ_{j_1} means that one does *two* Fourier transforms with $m = N/2$ terms instead of one with N terms. If $N/2$ is even the same idea can be applied to these two Fourier transforms. One then gets *four* Fourier transforms, each of with has $N/4$ terms; If $N = 2^k$ this reduction can be continued recursively until we get N DFTs with 1 term. But $F_1 = I$, the identity.

The number of complex operations required to compute $\{y_j\}$ from the butterfly relations when $\{\phi_{j_1}\}$ and $\{\psi_{j_1}\}$ have been computed is 2^k, assuming that the powers of ω are precomputed and stored. Thus, if we denote by p_k the total number of operations needed to compute the DFT when $N = 2^k$, we have

$$p_k \le 2p_{k-1} + 2^k, \quad k \ge 1.$$

Since $p_0 = 0$, it follows by induction that $p_k \le k \cdot 2^k = N \cdot \log_2 N$. *Hence, when N is a power of two, the fast Fourier transform solves the problem with at most $N \cdot \log_2 N$ operations.* The FFT is an example of the general technique of divide-and-conquer algorithms (see Sec. 1.3.2). For a recursive implementation of the FFT algorithm, see Problem 11.

Example 4.7.2.

Let $N = 2^4 = 16$. Then the 16-point DFT (0:1:15) can be split into two 8-points DFTs (0:2:14) and (1:2:15), which each can be split in two 4-point DFTs.

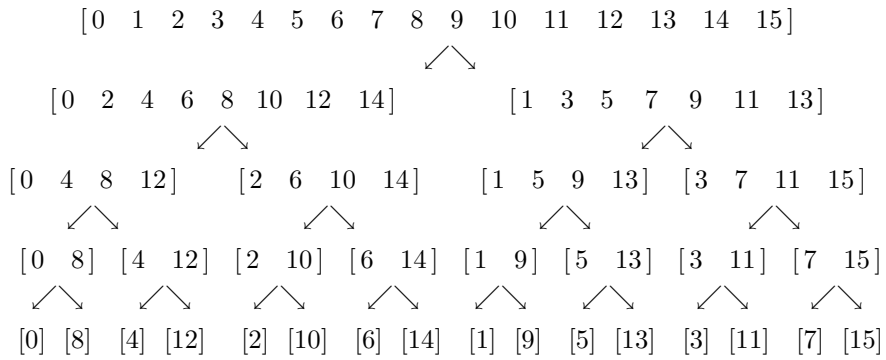

Figure 4.7.1. *The structure of an* $2^4 = 16$*-point FFT.*

Repeating these splittings we finally get 16 one-point DFTs, which are the identity $F_1 = 1$; see Figure 4.6.1.

In most implementations the explicit recursion is avoided. Instead the FFT algorithm is implemented in two stages:

- a reordering stage in which the data vector f is permuted;
- a second stage in which first $N/2$ FFT transforms of length 2 are computed on adjacent elements, next $N/4$ transforms of length 4, etc, until the final result is obtained by merging two FFTs of length $N/2$.

We now consider each stage in turn.

Each step of the recursion involves an even-odd permutations. In the first step the points with last digit equal to 0 are ordered first and those with last digit equal to 1 last. In the next step the two resulting subsequences of length $N/2$ are reordered according to the second binary digit, etc. It is not difficult to see that the combined effect of the reorderings in stage 1 is a **bit-reversal permutation** of the data points. For $i = 0 : N - 1$, let the index i have the binary expansion

$$i = b_0 + b_1 \cdot 2 + \cdots + b_{t-1} \cdot 2^{t-1}$$

and set

$$r(i) = b_{t-1} + \cdots + b_1 \cdot 2^{t-2} + b_0 \cdot 2^{t-1}.$$

That is, $r(i)$ is the index obtained by reversing the order of the binary digits. If $i < r(i)$ then exchange f_i and $f_{r(i)}$. This reordering is illustrated for $N = 16$ in Figure 4.7.1.

We denote the permutation matrix corresponding to bit-reversal ordering by P_N. Note that if an index is reversed twice we end up with the original index. This means that $P_N^{-1} = P_N^T = P_N$, that is P_N is symmetric. The permutation can be carried out "in place" by a sequence of pairwise interchanges or transpositions of the data points. For example, for $N = 16$ the pairs (1,8), (2,4), (3,12), (5,10), (7,14)

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

$\Longrightarrow$

Decimal	Binary
0	0000
8	1000
4	0100
12	1100
2	0010
10	1010
6	0110
14	1110
1	0001
9	1001
5	0101
13	1101
3	0011
11	1011
7	0111
15	1111

Figure 4.7.2. *Bit-reversal ordering. The original order left and the bit-reversal order right.*

and (11,13) are interchanged. The bit-reversal permutation can take a substantial fraction of the total time to do the FFT. Which implementation is best depends strongly on the computer architecture.

We now consider the second stage of the FFT. The key observation to develop a matrix-oriented description of this stage is to note that the Fourier matrices F_N after an odd-even permutation of the columns can be expressed as a 2×2 block matrix, where each block is either $F_{N/2}$ or a diagonal scaling of $F_{N/2}$.

Theorem 4.7.2. Van Loan [56, Theorem 1.2.1]

Let Π_N^T be the permutation matrix, which applied to a vector groups the even-indexed components first and the odd-indexed last.[43] *If $N = 2m$ then*

$$F_N \Pi_N = \begin{pmatrix} F_m & \Omega_m F_m \\ F_m & -\Omega_m F_m \end{pmatrix} = \begin{pmatrix} I_m & \Omega_m \\ I_m & -\Omega_m \end{pmatrix} \begin{pmatrix} F_m & 0 \\ 0 & F_m \end{pmatrix},$$

$$\Omega_m = \mathrm{d}iag\,(1, \omega_N, \ldots, \omega_N^{m-1}), \quad \omega_N = e^{-2\pi i/N}. \tag{4.7.8}$$

Proof. The proof essentially follows from the derivation of the butterfly relations (4.7.6)–(4.7.7). □

[43] Note that $\Pi_N^T = \Pi_N^{-1}$ is the so called **perfect shuffle permutation**. In this the permuted vector $\Pi_N^T f$ is obtained by splitting f in half and then "shuffling" the top and bottom halves.

Example 4.7.3.

We illustrate Theorem 4.7.2 for $N = 2^2 = 4$. The DFT matrix F_4 is given in Example 4.7.3. After a permutation of the columns F_4 can be written as a 2×2 block-matrix

$$F_4\Pi_4^T = \left(\begin{array}{cc|cc} 1 & 1 & 1 & 1 \\ 1 & -1 & -i & i \\ \hline 1 & 1 & -1 & -1 \\ 1 & -1 & i & -i \end{array}\right) = \begin{pmatrix} F_2 & \Omega_2 F_2 \\ F_2 & -\Omega_2 F_2 \end{pmatrix},$$

where

$$F_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \qquad \Omega_2 = \mathrm{d}iag\,(1, -i).$$

When $N = 2^k$ the FFT algorithm can be interpreted as a sparse factorization of the DFT matrix

$$F_N = A_k \cdots A_2 P_N, \tag{4.7.9}$$

where P_N is the bit-reversal permutation matrix and

$$A_q = \mathrm{d}iag\,(\underbrace{B_L, \ldots, B_L}_{r}), \quad L = 2^q, \quad r = n/L. \tag{4.7.10}$$

Here $B_k \in \mathbf{C}^{L\times L}$ is the radix-2 butterfly matrix defined by

$$B_L = \begin{pmatrix} I_{L/2} & \Omega_{L/2} \\ I_{L/2} & -\Omega_{L/2} \end{pmatrix}, \tag{4.7.11}$$

$$\Omega_{L/2} = \mathrm{d}iag\,(1, \omega_L, \ldots, \omega_L^{L/2-1}), \quad \omega_L = e^{-2\pi i/L}. \tag{4.7.12}$$

The FFT algorithm described above is usually referred to as the Cooley–Tukey FFT algorithm. Using the fact that both P_N and the DFT matrix F_n is symmetric, we obtain by transposing (4.7.9) the factorization

$$F_N = F_N^T = P_N A_1^T A_2^T \cdots A_k^T. \tag{4.7.13}$$

This gives rise to a "dual" FFT algorithm, referred to as the Gentleman–Sande algorithm [27]. In this the bit-reversal permutation comes *after* the other computations. In many important applications such as convolution and the solution of discrete Poisson equation, this permits the design of in-place FFT solutions that avoid bit-reversal altogether.

In the operation count for the FFT above we assumed that the weights ω_L^j, $j = 1 : L-1$, $\omega_L = e^{-2\pi i/L}$ are precomputed. To do this one could use at

$$\omega_L^j = \cos(j\theta) - i\sin(j\theta), \quad \theta = 2\pi/L.$$

for $L = 2^q$, $q = 2 : k$. This is accurate, but expensive, since it involves $L-1$ trigonometric functions calls. An alternative is to compute $\omega = \cos(\theta) - i\sin(\theta)$ and use repeated multiplication,

$$\omega^j = \omega\omega^{j-1}, \quad j = 2 : L-1.$$

This replaces one sine/cosine call with a single complex multiplication, but has the drawback that accumulation of roundoff errors will give an error in ω_L^j of order ju.

4.7.2 FFTs and Discrete Convolutions

Many applications of the FFT involve the use of a discrete version of the convolution,

Definition 4.7.3.
Given two sequences f_i and g_i, $i = 0 : N-1$. Then the convolution of f and g is the sequence defined by

$$h_k = conv\,(f,g) = \sum_{i=0}^{N-1} f_i g_{k-i}, \quad i = 0 : N-1, \tag{4.7.14}$$

where the sequences are extended to have period N, by setting $f_i = f_{i+jN}$, $g_i = g_{i+jN}$, for all integers i, j.

The discrete convolution can be used to approximate the convolution defined for continuous functions in Definition 4.6.6 in a similar way as the Fourier transform was approximated Using sampled values in Sec. 4.6.4.

We can write the sum in (4.7.14) as a matrix-vector multiplication $h = Gf$, or writing out components

$$\begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ \vdots \\ h_{N-1} \end{pmatrix} = \begin{pmatrix} g_0 & g_{N-1} & g_{N-2} & \cdots & g_1 \\ g_1 & g_0 & g_{N-1} & \cdots & g_2 \\ g_2 & g_1 & g_0 & \cdots & g_3 \\ \vdots & \vdots & \vdots & \cdots & \\ g_{N-1} & g_{N-2} & g_{N-3} & \cdots & g_0 \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ f_2 \\ \vdots \\ f_{N-1} \end{pmatrix}.$$

Note that each column in G is a cyclic down-shifted version of the previous column. Such a matrix is called a **circulant matrix**. We have

$$G = [\, g \quad R_N g \quad R_N^2 g \quad \cdots \quad R_N^{N-1} g \,],$$

and R_N is a circulant permutation matrix. For example,

$$R_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

(see Problem 11).

Theorem 4.7.4.
Let f_i and g_i, $i = 0 : N-1$ be two sequences with DFTs equal to $F_N f$ and $F_N g$. Then the DFT of the **convolution** *of f and g, is $F_N f .* F_N g$, where $.*$ denotes elementwise product.*

Proof. The proof depends on the fact that the circulant matrix G is diagonalized by the DFT matrix F_N, i.e.

$$G = F_N^{-1} \mathrm{d}iag\,(F_N g) F_N;$$

see Problem 15. (Here d$iag\,(x)$, where x is a vector, denotes a diagonal matrix with diagonal elements equal to x.) It follows that

$$h = Gf = F_N^{-1}\mathrm{d}iag\,(F_N g)F_N f = F_N^{-1}((F_N g).*(F_N f)). \tag{4.7.15}$$

It follows from (4.7.15) that $conv\,(f,g) = conv\,(g,f)$. □

This shows that using the FFT algorithm the discrete convolution can be computed in order $N\log_2 N$ operations as follows. First the two FFTs of f and g are computed and multiplied (pointwise) together. Then the inverse DFT of this product is computed. This is one of the most useful properties of the FFT. Computing convolutions is of great importance in signal processing.

Using the Gentleman–Sande algorithm $F_N = P_N A^T$ for the forward DFT and the Cooley–Tukey algorithm for the inverse DFT, $F^{-1} = (1/N)\bar{F}_N = (1/N)\bar{A}P_N/N$, we get from (4.7.15)

$$f = \frac{1}{N}\bar{A}P_N((P_N A^T h).*(P_N A^T g)) = \frac{1}{N}\bar{A}((A^T h).*(A^T g)). \tag{4.7.16}$$

This shows that h can be computed without the bit-reversal permutation P_N, which typically can save 10–30 percent of the overall computation time.

4.7.3 Real Data and Fast Trigonometric Transforms

Frequently the FFT of a real data vector is required. The complex FFT algorithm can still be used, but is inefficient both in terms of storage and operations. Better alternatives can be found by using symmetries in the DFT, which correspond to the symmetries noted in the Fourier transform in Table 4.6.1.

We first show that the conjugate transpose of the DFT matrix F_N can be obtained by reversing the order of the last $N-1$ rows.

Lemma 4.7.5. Van Loan [56, Theorem 4.3.1]

Let T_N be the $N\times N$ permutation matrix which reverses the last $N-1$ elements. Then $\overline{F}_N = T_N F_N = F_N T_N$.

Proof. To verify that $\overline{F}_N = T_N F_N$, observe that

$$[T_N F_N]_{j\alpha} = \omega_N^{(N-j)\alpha} = \omega_N^{-j\alpha} = \overline{\omega}_N^{j\alpha} = [\overline{F}_N]_{j\alpha}, \quad 1 \le j \le N-1.$$

Since F_N and T_N are both symmetric, we also have $\overline{F}_N = (T_N F_N)^T = F_N T_N$. □

We say that a vector $x \in \mathbf{C}^N$ is **conjugate even** if $\bar{x} = T_N y$, and **conjugate odd** if $\bar{x} = T_N y$. Suppose now that f is real and $u = F_N f$. Then it follows that

$$\overline{u} = \overline{F}_N f = T_N F_N f = T_N u,$$

i.e. u is conjugate even. If a vector u of even length $N = 2m$ is conjugate even, this implies that

$$u_j = \bar{u}_{N-j}, \quad j = 1:m.$$

In particular u_j is real for $j = 0, m$.

For purely imaginary data g and $v = F_N g$, we have

$$\overline{v} = \overline{F}_N \overline{g} = -\overline{F}_N g = -T_N F_N g = -T_N v,$$

i.e. v is conjugate odd. Some other useful symmetry properties are given in the table below, where E_N denotes the $N \times N$ permutation matrix which reverses all elements in an N-vector.

Data f	Definition	DFT $F_N f$
real		conjugate even
imaginary		conjugate odd
real even	$f = T_n f$	real
real odd	$f = -T_n f$	imaginary
conjugate even	$\overline{f} = T_n f$	real
conjugate odd	$\overline{f} = -T_n f$	imaginary

Table 4.7.1. *Useful symmetry properties of DFTs.*

We now outline how symmetries can be used to compute the DFTs $u = F_N f$ and $v = F_N g$ of *two* real functions f and g simultaneously. First form the complex function $f + ig$ and compute its DFT

$$w = F_N(f + ig) = u + iv$$

by any complex FFT algorithm. Multiplying by T_N we have

$$T_N w = T_N F_N(f + ig) = T_N(u + iv) = \bar{u} + i\bar{v},$$

where we have used that u and v are conjugate even. Adding and subtracting these two equations we obtain

$$\begin{aligned} w + T_N w &= (u + \bar{u}) + i(v + \bar{v}), \\ w - T_N w &= (u - \bar{u}) + i(v - \bar{v}). \end{aligned}$$

We can now retrieve the two DFTs from

$$u = F_N f = \frac{1}{2}\big[\mathrm{Re}(w + T_N w) + i\,\mathrm{Im}(w - T_N w)\big], \tag{4.7.17}$$

$$v = F_N g = \frac{1}{2}\big[\mathrm{Im}(w + T_N w) - i\,\mathrm{Re}(w - T_N w)\big]. \tag{4.7.18}$$

Note that because of the conjugate even property of u and v there is no need to save the entire transforms.

The above scheme is convenient when, as for convolutions, two real transforms are involved. It can also be used to efficiently compute the DFT of a single real

function of length $N = 2^k$. First express this DFT as a combination of the two real FFTs of length $N/2$ corresponding to even and odd numbered data points (see as in (4.7.5)). Then apply the procedure above to simultaneously compute these two real FFTs.

Two more real transforms, the **discrete sine transform (DST)** and **discrete cosine transform (DCT)**, are of interest. These are defined as follows:

- Given real f_j, $j = 1 : m-1$ compute

$$y_k = \sum_{j=1}^{m-1} \sin(kj\pi/m)\, f_j \quad \text{(DST)}. \tag{4.7.19}$$

- Given real f_j, $j = 0 : m$ compute

$$y_k = \frac{1}{2}(f_0 + (-1)^k f_m) + \sum_{j=1}^{m-1} \sin(kj\pi/m)\, f_j \quad \text{(DCT)}. \tag{4.7.20}$$

These can be computed by applying the FFT algorithm (for real data) to an auxiliary vector formed by extending the given data f either into an odd or even sequence.

For the DST the f_j, $j = 1 : m-1$ is extended to an *odd* sequence of length $N = 2m$ by setting

$$f_0 = f_m = 0, \quad f_{2m-j} \equiv -f_j, \quad j = 1 : m-1.$$

For example, the data $\{f_1, f_2, f_3\}$, $(m = 2^2)$ is extended to

$$\tilde{f} = \{f_0, f_1, f_2, f_3, f_0, -f_3, -f_2, -f_1\}.$$

The extended vector satisfies $\tilde{f} = -T_N \tilde{f}$, and thus by Table 4.6.2 the DFT of $\tilde{f}$ will imaginary.

For the DCT the data f_j, $j = 0 : m$ is extended to an *even* sequence of length $N = 2m$ by setting

$$f_0 = f_m = 0, \quad f_{2m-j} \equiv f_j, \quad j = 1 : m-1.$$

For example, the data $\{f_0, f_1, f_2, f_3, f_4\}$, $(m = 2^2)$ is extended to

$$\tilde{f} = \{f_0, f_1, f_2, f_3, f_4, f_3, f_2, f_1\}.$$

so that $\tilde{f} = T_N \tilde{f}$. By Table 4.6.2 the DFT of $\tilde{f}$ will then be real.

Theorem 4.7.6. Van Loan [56, Sec. . 4.4]

Let f_j, $j = 1 : m-1$ form a real data vector f and extend it to a vector $\tilde{f}$ with $\tilde{f}_0 = \tilde{f}_m = 0$, so that $\tilde{f} = -T_N \tilde{f}$. Then $y(1 : m-1)$ is the DST of f, where

$$y = \frac{i}{2} F_{2m} \tilde{f}.$$

Let f_j, $j = 0 : m$ form a real data vector f and extend it to an vector $\tilde{f}$ so that $\tilde{f} = T_N \tilde{f}$. Then $y(0 : m)$ is the DST of f, where

$$y = \frac{1}{2} F_{2m} \tilde{f}.$$

There is an inefficiency factor of two in the above procedure. This can be eliminated by using a different auxiliary vector. For details we refer to [39, p. 420–421] and [56, Sec. 4.4.].

4.7.4 The General Case FFT

It can be argued ([39, p. 409]) that one should always choose $N = 2^k$ when using the FFT. If necessary the data can be padded with zeros to achieve this. To introduce an odd factor s, let $N = sr$, where r is a power of two. Then one can combine the power of two algorithm for the r-point subseries with a special algorithm for the s-point subseries. If s is a small number then one could generate the DFT matrix F_s and use matrix-vector multiplication; see Problem 10. However, the general case when if N is not a power of two, is at least of theoretical interest.

Suppose that $N = r_1 r_2 \cdots r_p$. We will describe an FFT algorithm which requires $N(r_1 + r_2 + \cdots + r_p)$ operations. Set

$$N_\nu = \prod_{i=\nu+1}^{p} r_i, \quad \nu = 0 : p-1, \quad N_p = 1.$$

Thus

$$N = r_1 r_2 \cdots r_\nu N_\nu, \quad N_0 = N.$$

The algorithm is based on two representations of integers, which are generalizations of the position principle (see Sec. 2.2.1).

I. Every integer j, $0 \le j \le N-1$ has a unique representation of the form

$$j = \alpha_1 N_1 + \alpha_2 N_2 + \cdots + \alpha_{p-1} N_{p-1} + \alpha_p, \quad 0 \le \alpha_i \le r_i - 1. \tag{4.7.21}$$

II. For every integer β, $0 \le \beta \le N-1$, β/N has a unique representation of the form

$$\frac{\beta}{N} = \frac{k_1}{N_0} + \frac{k_2}{N_1} + \cdots + \frac{k_p}{N_{p-1}}, \quad 0 \le k_i \le r_i - 1. \tag{4.7.22}$$

Set

$$j_\nu = \sum_{i=\nu+1}^{p} \alpha_i N_i, \qquad \frac{\alpha_\nu}{N_\nu} = \sum_{i=\nu}^{p-1} \frac{k_{i+1}}{N_i}, \quad (j_\nu < N_\nu). \tag{4.7.23}$$

As an exercise, the reader can verify that the coefficients in the above representations can be recursively determined from the following algorithms: [44]

$$j_0 = j, \qquad j_{i-1}/N_i = \alpha_i + j_i/N_i, \quad i = 1 : p;$$

[44] These algorithms can, in the special case that $r_i = B$ for all i, be used for converting integers or fractions to the number system whose base is B; see Algorithm 2.2.1.

$$\beta_0 = \beta, \qquad \beta_{i-1}/r_i = \beta_i + k_i/r_i, \quad i = 1:p.$$

From (4.7.21)–(4.7.23), it follows that, since N_i is divisible by N_ν for $i \le \nu$,

$$\frac{j\beta}{N} = \text{integer} + \sum_{\nu=0}^{p-1} \frac{k_{\nu+1}}{N_\nu}\Big(\sum_{i=\nu+1}^{p} \alpha_i N_i\Big) = \sum_{\nu=0}^{p-1} \frac{k_{\nu+1} j_\nu}{N_\nu} + \text{integer}.$$

From this, it follows that

$$\omega^{j\beta} = e^{2\pi i j\beta/N} = \prod_{\nu=0}^{p-1} e^{k_{\nu+1} j_\nu 2\pi i/N_\nu} = \prod_{\nu=0}^{p-1} \omega_\nu^{j_\nu k_{\nu+1}}, \tag{4.7.24}$$

where $\omega_\nu = e^{2\pi i/N_\nu}$, $\omega_0 = \omega$.

We now give an illustration of how the factorization in (4.7.24) can be utilized in fast Fourier transform for the case $p = 3$. Set, in accordance with (4.7.22),

$$f_\beta = c^{(0)}(k_1, k_2, k_3).$$

We have then

$$c_j = \sum_{\beta=0}^{N-1} f_\beta \omega^{j\beta} = \sum_{k_1=0}^{r_1-1} \sum_{k_2=0}^{r_2-1} \sum_{k_3=0}^{r_3-1} c^{(0)}(k_1, k_2, k_3) \omega_2^{j_2 k_3} \omega_1^{j_1 k_2} \omega^{j k_1}.$$

One can thus compute successively (see (4.7.23))

$$c^{(1)}(k_1, k_2, \alpha_3) = \sum_{k_3=0}^{r_3-1} c^{(0)}(k_1, k_2, k_3) \omega_2^{j_2 k_3} \quad (j_2 \text{ depends only on } \alpha_3),$$

$$c^{(2)}(k_1, \alpha_2, \alpha_3) = \sum_{k_2=0}^{r_2-1} c^{(1)}(k_1, k_2, \alpha_3) \omega_1^{j_1 k_2} \quad (j_1 \text{ depends only on } \alpha_2, \alpha_3),$$

$$c_j = c^{(3)}(\alpha_1, \alpha_2, \alpha_3) = \sum_{k_1=0}^{r_1-1} c^{(2)}(k_1, \alpha_2, \alpha_3) \omega^{j k_1} \quad (j \text{ depends on } \alpha_1, \alpha_2, \alpha_3).$$

The quantities $c^{(i)}$ are computed for all $r_1 r_2 r_3 = N$ combinations of the values of the arguments. Thus the total number of operations for the entire Fourier analysis becomes at most $N(r_3 + r_2 + r_1)$. The generalization to arbitrary p is obvious.

Review Questions

1. Suppose we want to compute the DFT for $N = 2^{10}$. Roughly how much faster is the FFT algorithm compared to the straightforward $O(N)$ algorithm?
2. Show that the matrix $U = \frac{1}{\sqrt{N}} F_N$ is unitary, i.e. $U^* U = I$, where $U^* = (\overline{U})^T$.

3. Show that the DFT matrix F_4 can be written as a 2×2 block matrix where each block is related to F_2. Give a generalization of this for F_N, $N = 2^m$, that holds for arbitrary m..

4. Work out on your own the bit-reversal permutation of the vector $[0 : N-1]$ for the case $N = 2^4 = 16$. How many exchanges need to be performed?

Problems and Computer Exercises

1. The following MATLAB script uses an algorithm due to Cooley et al. to permute the vector $x(1 : 2^m)$, in bit-reversal order:

```
n = 2^m;
nv2 = n/2; nm1 = n - 1;
j = 1;
for i = 1:nm1
   if i < j
      t = x(j); x(j) = x(i); x(i) = t;
   end
   k = nv2;
   while k < j
      j = j - k; k = k/2;
   end
   j = j + k;
end
```

Plot the time taking by this algorithm on your computer for $m = 5 : 10$. Does the execution time depend linearly on $N = 2^m$?

2. The following MATLAB program (C. Moler and S. Eddins [37]) demonstrates how the FFT idea can be implemented in a simple but efficient recursive MATLAB program. The program uses the fast recursion as long as n is a power of two. When it reaches an odd length it sets up the Fourier matrix and uses matrix vector multiplication.

```
function y = fftx(x);
% FFT computes the Fast Fourier Transform of x(1:n)
x = x(:);
n = length(x);
omega = exp(-2*pi*i/n);
if rem(n,2) == 0
% Recursive divide and conquer
   k = (0:n/2-1)
   w = omega.^k;
   u = fftx(x(1:2:n-1));
   v = w.*fftx(x(2:2:n));
   y = [u+v; u-v];
```

```
else
% Generate the Fourier matrix
   j = 0:n-1;
   k = j';
   F = omega.^(k*j);
   y = F*x;
end
```

Apply this program to compute DFT of the function treated in Example 4.6.1 sampled at the points $2\pi\alpha/N$, $\alpha = 0 : N-1$. Choose, for instance, $N = 32, 64, 128$.

3. Write an efficient Matlab program for computing the DFT of a real data vector of length $N = 2^m$. As outlined in Sec. 4.7.3, first split the data in odd and even data points. Compute the corresponding DFTs using one call of the function fftx in Problem 10 with complex data of length $N/2$.

4. Verify the last four symmetry properties of DFTs in Table 4.6.2.

5. Let C be the square matrix

$$C_n = \begin{pmatrix} 0_{n-1}^T & 1 \\ I_{n-1} & 0_{n-1} \end{pmatrix} = \begin{pmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & & & & 0 \\ & 1 & & & \vdots \\ & & \ddots & & 0 \\ & & & 1 & 0 \end{pmatrix} \in \mathbf{R}^{n\times n}.$$

Show that the effect of C_nA and AC_n, respectively, is a circular shift downwards of the rows and a shift to the left of the columns in $A \in \mathbf{R}^{n\times n}$. What about C^T?

6. A circulant matrix $A \in \mathbf{R}^{n\times n}$ generated by $(a_1, a_2, \ldots, a_{n-1}, a_n)$ has the form

$$A = \begin{pmatrix} a_0 & a_{n-1} & \cdots & a_2 & a_1 \\ a_1 & a_0 & \cdots & a_3 & a_2 \\ \vdots & \vdots & & \vdots & \vdots \\ a_{n-2} & a_{n-3} & \cdots & a_0 & a_{n-1} \\ a_{n-1} & a_{n-2} & \cdots & a_1 & a_0 \end{pmatrix}.$$

(a) Show that $A = a_0I + a_1C + \ldots + a_{n-1}C^{n-1}$, where C is the circulant matrix in Problem 6.

(b) Show that the eigenvalues and eigenvectors of A are given by

$$\lambda_j = a_0 + a_1\omega_j + \ldots + a_{n-1}\omega_j^{n-1}, \quad x_j = \frac{1}{\sqrt{n}}(1, \omega_j, \ldots, \omega_j^{n-1})^T,$$

where $\omega_j = e^{2\pi j/n}$, $j = 1 : n$ are the n roots of unity $\omega^n = 1$.

(c) Show that the result in (b) implies that $C = F\Lambda F^H$, where

$$\Lambda = \mathrm{d}iag\,(\lambda_1, \ldots, \lambda_n), \qquad F = (x_1, \ldots, x_n)$$

is the matrix of the discrete Fourier transform, and the eigenvalues are given by the Fourier transform of its first column

$$F(a_0, a_{n-1}, \ldots, a_2, a_1)^T = (\lambda_1, \ldots, \lambda_n)^T.$$

4.8 Complex Analysis in Interpolation

In this section we make a more detailed theoretical and experimental study of interpolation of an analytic function $f(z)$ on a real interval. including an analysis of the Runge phenomenon (see Sec. 4.1.4). We then study interpolation at an infinite equidistant point set from the point of view of Complex Analysis. This interpolation problem, which was studied by Whittaker and others at the beginning of the century, became revived at the middle of the century under the name of the **Shannon sampling theorem**, with important applications to Communication Theory.

We shall encounter multi-valued functions: the logarithm and the square root. For each of these we choose that branch, which is positive for large positive values of the argument. They will appear in such contexts that we can then keep them non-ambiguous by forbidding z to pass the interval $[-1, 1]$. (We can, however, allow z to approach that interval.)

4.8.1 Interpolation of Analytic Functions

We first consider the general problem of polynomial interpolation of an analytic function, at an arbitrary sequence of points in $\mathbf{C}$. Multiple points are allowed. Set[45]

$$\Phi(z) = (z - u_1)(z - u_2) \cdots (z - u_n), \quad z, u_j \in \mathbf{C}.$$

Let $\mathcal{D}$ be a simply connected open domain in $\mathbf{C}$ that contains the point u and the nodes $u_1, u_2, \ldots, u_n$. We consider the interpolation problem to find the polynomial $p^* \in \mathcal{P}_n$ that is determined by the conditions $p^*(u_j) = f(u_j)$, $j = 1 : n$, or the appropriate Hermite interpolation problem in the case of multiple nodes. We know that p^* depends linearly on f, that is there exists a linear mapping L_n from some appropriate function space so that $p^* = L_n f$.

Assume that f is an analytic function in the closure of $\mathcal{D}$, perhaps except for a finite number of poles p. A pole must not be a node. Recall the elementary identity

$$\frac{1}{z-u} = \sum_{j=1}^{n} \frac{\Phi_{j-1}(u)}{\Phi_j(z)} + \frac{\Phi_n(u)}{\Phi_n(z)(z-u)}, \tag{4.8.1}$$

which is valid also for multiple nodes. Introduce the linear operator K_n,

$$(K_n f)(u) = \frac{1}{2\pi i} \int_{\partial\mathcal{D}} \frac{\Phi_n(u) f(z)}{\Phi_n(z)(z-u)}, \tag{4.8.2}$$

[45] We use the notation u, u_i instead of x, x_i here, since x is traditionally associated with the real part of a complex variable z.

multiply the above identity by $f(z)/(2\pi i)$, and integrate along the boundary of $\mathcal{D}$:

$$\frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{f(z)}{z-u} = \sum_{j=1}^{n} \Phi_{j-1}(u)\frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{f(z)}{\Phi_j(z)}\,dz + (K_n f)(u). \tag{4.8.3}$$

The following theorem is valid, when the interpolation points x_j are in the complex plane, although we shall here mainly apply it to the case, when they are located in the interval $[-1, 1]$.

Theorem 4.8.1.

Assume that $f(z)$ is analytic in a domain $\mathcal{D}$ that contains the points $x_1, x_2, \dots x_n$, as well as the point $u \in \mathbf{C}$. Let $L_n f$ be the solution of the interpolation problem $(L_n f)(x_j) = f(x_j)$, $j = 1 : n$. Then the interpolation error can be expressed as a complex integral, $f(u) - (L_n f)(u) = I_n(u)$, where

$$I_n(u) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{\Phi(u)f(z)}{\Phi(z)(z-u)}\,dz.$$

Proof. By the residue theorem,

$$I_n(u) = \sum_{j=1}^{n} \frac{\Phi(u)f(x_j)}{(u-x_j)\Phi'(x_j)} + f(u),$$

where the sum, with reversed sign, is Lagrange's form of the interpolation polynomial. □

(Note the relation between the Lagrange interpolation formula and the expansion of $f(z)/\Phi(z)$ into partial fractions, when, e.g., $f(z)$ is a polynomial.)

We now proceed to *Chebyshev interpolation*, i.e. interpolation at the the zeros of the Chebyshev polynomials. We shall show that it is almost as efficient as the truncation of a Chebyshev expansion. In this case, $\Phi(z) = 2^{1-n}T_n(z)$. Let $\mathcal{D} = E_R$, $x \in [-1, 1]$, $z \in \partial E_R$, where E_R is the ellipse

$$E_R = \{z : |z-1| + |z+1| \le R + R^{-1}\},$$

introduced in Sec. 3.5.1 (see (3.2.24). Consider the integral in Theorem 4.8.1 and assume that $|f(z)| \le M$ for $z \in \partial E_R$. It can be shown (Problem 2) that $|T_n(x)| \le 1$ and

$$|T_n(z)| \ge \tfrac{1}{2}(R^n - R^{-n}), \quad |z-x| \ge a-1, \quad \int_{\partial E_R} |dz| \le 2\pi a,$$

where a is the major semi-axis of E_R, i.e. $a = \frac{1}{2}(R + R^{-1})$. Then, by a straightforward calculation,

$$|f(x) - (L_n f)(x)| \le \frac{2MR^{-n}a}{(1-R^{-2n})(a-1)}. \tag{4.8.4}$$

This is somewhat less sharp than the result obtained by the Chebyshev expansion, in particular when $R \approx 1$. The details are left for Problem 2.

Note that $f(z)$ is allowed to have a singularity arbitrarily close to the interval $[-1, 1]$, and the convergence of Chebyshev interpolation will still be exponential. Of course, the exponential rate will be very poor, when $R \approx 1$.

4.8.2 Analysis of a Generalized Runge Phenomenon

It is well known that the Taylor series of an analytic function converges at an exponential rate inside its circle of convergence, while it diverges at an exponential rate outside. We shall see that a similar result holds for certain interpolation processes. In general, the domains of convergence are not disks but bounded by level curves of a logarithmic potential, related to the asymptotic distribution of the interpolation points.

For the sake of simplicity, we now confine the discussion to the case, when the points of interpolation are located in the standard interval $[-1, 1]$, but we are still interested in the evaluation of the polynomials in the complex domain. Part of the discussion can, however, be generalized to a case, when the interpolation points are on an arc in the complex plane.

Let $q : [a, b] \mapsto [-1, 1]$ be an increasing and continuously differentiable function. Set $t_{n,j} = a + (b - a)j/n$, $j = 0 : n$, and let the interpolation points be $x_{n,j}$, $j = 1 : n$, where $q(t_{n,j-1}) < x_{n,j} \leq q(t_{n,j})$, i.e. one interpolation point in each of n subintervals of $[-1, 1]$. In the definition of Φ, we now write $x_{n,j}$, Φ_n instead of x_j, Φ. Note that, as $n \to \infty$,

$$\frac{1}{n} \ln \Phi_n(z) = \frac{1}{n} \sum_{j=1}^{n} \ln (z - x_{n,j}) \;\to\; \psi(z) := \frac{1}{b-a} \int_a^b \ln (z - q(t))\, dt, \quad z \notin [-1, 1]. \tag{4.8.5}$$

Put $x = q(t)$, and introduce a density function $w(x), x \in]-1, 1[$ that is the derivative of the inverse function of q, i.e. $w(x) = 1/\big(q'(t(x))(b - a)\big) > 0$. Then

$$\psi(z) = \int_{-1}^{1} \ln(z - x) w(x)\, dx, \quad w(x) > 0, \quad \int_{-1}^{1} w(x)\, dx = 1, \tag{4.8.6}$$

and $\psi(z)$ is analytic in the whole plane outside the interval $[-1, 1]$. Its real part $P(z)$ is the **logarithmic potential** of a weight distribution,

$$P(z) = \Re\psi(z) = \int_{-1}^{1} \ln |z - x| w(x)\, dx, \quad w(x) > 0. \tag{4.8.7}$$

The function $\frac{1}{n} \ln |\Phi_n(z)|$ is itself the logarithmic potential of a discrete distribution of equal weights $\frac{1}{n}$, at the interpolation points $x_{j,n}$. This function is less pleasant to deal with than $P(z)$, since it becomes $-\infty$ at the interpolation points while, according to classical results of potential theory, $P(z)$ *is continuous everywhere, also on the interval* $[-1, 1]$. If we set $z = x + iy$, $\partial P(z)/\partial x$ is also continuous for $z \in]-1, 1[$, while $\partial P(z)/\partial y$ has a jump there. We write it thus,

$$\psi'(x - 0i) - \psi'(x + 0i) = 2\pi i w(x). \tag{4.8.8}$$

Figure 4.8.1. *Figure to be made.*

By (4.8.6), $\psi(z) = \ln z + O(z^{-1})$, $|z| \to \infty$, or even $O(z^{-2})$, if the weight distribution is symmetric around the origin.

We make the definition

$$\mathcal{D}(v) = \{z \in \mathbf{C} : P(z) < P(v)\}.$$

and set $P^* = \max_{x\in[-1,1]} P(x)$. It can be shown that $\mathcal{D}(v)$ is a simply connected domain if $P(v) > P^*$. The level curve $\partial\mathcal{D}(v) = \{z : P(z) = P(v)\}$ then encloses $[-1, 1]$. A level curve $\{z : P(z) = a\}$ is strictly inside the level curve $\{z : P(z) = a'\}$ if $a \le P^* < a'$. (The proof of these statements essentially utilizes the minimum principle for harmonic functions and the fact that $P(z)$ is a regular harmonic function outside $[-1, 1]$ that grows to ∞ with $|z|$.)

We now consider two examples.

Example 4.8.1. *Equidistant interpolation*

In this case we may take $q(t) = t$, $t \in [-1, 1]$, hence $w(x) = 1/2$. For the **equidistant case** we have if $z \notin [-1, 1]$,

$$P(z) = \tfrac{1}{2}\Re \int_{-1}^{1} \ln(z - x)\, dx = \tfrac{1}{2}\Re\big((1 - z)\ln(z - 1) + (1 + z)\ln(z + 1)\big) - 1.$$

The upper half of the level curves may look something like Figure 4.8.1.:

On the imaginary axis,

$$P(iy) = \tfrac{1}{2}\ln(1 + y^2) + y(\tfrac{1}{2}\pi - \arctan y) - 1.$$

When $z \to x \in [-1, 1]$, from any direction, $P(z)$ tends to

$$P(x) = \tfrac{1}{2}\big((1 - x)\ln(1 - x) + (1 + x)\ln(1 + x)\big) - 1. \tag{4.8.9}$$

$P'(x)$ is continuous in the interior, but becomes infinite at $x = \pm 1$. The imaginary part of $\psi(z)$ has, however, different limits, when the interval is approached from above and below: $\Im(\psi(x \pm 0i)) = \pm\pi(1 - x)$.

The level curve of $P(z)$ that passes through the points ± 1, intersects the imaginary axis at the points $\pm iy$, determined by the equation $P(iy) = P(1) = \ln 2 - 1$, with the root $y = 0.5255$. Theorem 4.8.2 (below) will tell us that $L_n f(x) \to f(x)$, $\forall x \in]-1, 1[$, if $f(z)$ is analytic inside and on this contour.

In the classical example of Runge, $f(z) = 1/(1 + 25z^2)$ has poles inside this contour at $z = \pm 0.2i$. Proposition 4.8.3 will tell us that the level curve of $P(z)$ that passes through these poles will separate between the points, where the interpolation process converges and diverges. Its intersections with the real axis is determined by the equation $P(x) = P(0.2i) = -1.41142$. The roots are $x = \pm 0.72668$.

Example 4.8.2. *Chebyshev interpolation*

In this example we have

$$q(t) = \cos(\pi(1-t)), \quad t \in [0,1], \qquad w(x) = \frac{1}{\pi}(1-x^2)^{-\frac{1}{2}}.$$

Moreover (see Sec. 3.5.1) substitute s for w,

$$\Phi_n(z) = 2^{1-n} T_n(z) = 2^{-n}(s^n + s^{-n}),$$

where $z = \frac{1}{2}(s + s^{-1})$, $s = z + \sqrt{z^2 - 1}$. Note that $|s| \geq 1$, according to our convention about the choice of branch for the square root. Hence,

$$P(z) = \lim \frac{1}{n} \ln |\Phi_n(z)| - \ln 2 = \ln \frac{|s|}{2} = \ln |z + \sqrt{z^2 - 1}| - \ln 2.$$

Therefore, the family of confocal ellipses ∂E_R are, in this example, the level curves of $P(z)$. In fact, by (1.3') and the formula for $P(z)$, the interior of E_R equals $\mathcal{D}(\ln R - \ln 2)$. The family includes, as a limit case ($R = 1$), the interval $[-1, 1]$, in which $P(z) = -\ln 2$.

Our problem is related to a more conventional application of potential theory, namely the problem of finding the electrical charge distribution of a long insulated charged metallic plate in the strip

$$\{(x, y) \in \mathbf{R}^2 : \ -1 < x < 1, -L < y < L\}, \quad L \gg 1.$$

Such a plate will be equipotential. The charge density at the point (x, y) is then proportional to

$$w(x) = \frac{1}{\pi}(1-x^2)^{-1/2};$$

a fascinating relationship between electricity and approximation.

Note that if $z \notin [-1, 1]$, we can, by the definition of $P(z)$ as a Riemann sum (see (4.1)) find a sequence $\{\epsilon_n\}$ that decreases monotonically to zero, such that

$$\frac{1}{n} |\ln \Phi_n(z) - \psi(z)| < \epsilon_n, \quad z \notin [-1, 1]. \tag{4.8.10}$$

It is conceivable that the same sequence can be used for all z on a curve that does not touch the interval $[-1,1]$. (The proof is omitted.)

We can only claim a *one-sided inequality*, if we allow that $u \in [-1,1]$.

$$\frac{1}{n}(\Re \ln \Phi_n(u) - \psi(u)) < \epsilon_n, \quad u \in \mathbf{C}. \tag{4.8.11}$$

(Recall that $\Re \ln \Phi_n(u) = -\infty$ at the interpolation points.) We can use the same sequence for z and u. We can also say that $|\Phi_n(u)|$ *behaves like* $\exp\big((P(u) \pm \delta)n\big)$ *outside the immediate vicinity of the interpolation points.*

Theorem 4.8.2.

Assume that $[-1,1]$ *is strictly inside a simply connected domain* $\mathcal{D} \supseteq \mathcal{D}(v)$. *If* $f(\zeta)$ *is analytic in the closure of* $\mathcal{D}$, *then the interpolation error* $(L_n f)(u) - f(u)$ *converges like an exponential to* 0 *for any* $u \in \mathcal{D}(v)$.

Proof. By Theorem 4.8.1, $f(u) - (L_n f)(u) = I_n(u)$, where

$$I_n(u) = \frac{1}{2\pi i} \int_{\partial \mathcal{D}} \frac{\Phi_n(u) f(z)}{\Phi_n(z)(z-u)}\, dz. \tag{4.8.12}$$

Note that $P(z) \geq P(v)$, because $\mathcal{D} \supseteq \mathcal{D}(v)$. Then, by (4.8.10) and (4.8.11),

$$|\Phi_n(u)/\Phi_n(z)| < \exp n\big(P(u) - P(v) + 2\epsilon_n\big)$$

Let $|f(z)| \leq M$. For any $u \in \mathcal{D}(v)$, we can choose $\delta > 0$, such that $P(u) < P(v) - 3\delta$, $|z-u| > \delta$. Next, choose n large enough so that $\epsilon_n < \delta$. Then

$$|f(u) - (L_n f)(u)| < \frac{1}{2\pi} M \exp n(-3\delta + 2\delta) \int_{\partial \mathcal{D}(v)} \frac{|dz|}{\delta} \leq \frac{K \exp(-n\delta)}{\delta}.$$

where $K = K(v)$ does not depend on n, δ and u, hence the convergence is exponential. □

Remark 4.8.1. *If* u *is away from the boundary* $\partial\mathcal{D}(v)$, *more realistic estimates of the interpolation error and the speed of convergence is for* $n \gg 1$ *given by*

$$K_1(v)|\Phi_n(u)|e^{(-P(v)+\delta)n} \leq K_1(v) e^{(P(u)-P(v)+\delta)n},$$

where $K_1(v)$ *is another constant. The latter estimate is realistic outside the immediate vicinity of the interpolation points.*

It seems, as if one should choose $|v|$ as large as possible, in order to increase $P(v)$. A bound for $P(v)$ is usually set by the singularities of $f(z)$. If $f(z)$ is an entire function, the growth of the maximum modulus of $|f(z)|$, $|z| \in \mathcal{D}(v)$, hidden in $K_1(v)$, sets a bound for $P(v)$ that usually increases with n.

We shall now derive a complement and a kind of converse to Theorem 4.8.2, for functions $f(z)$ that have simple poles in $\mathcal{D}(v)$.

Proposition 4.8.3.

Assume that $[-1,1]$ *is strictly inside a domain* $\mathcal{D} \supset \mathcal{D}(v)$*, and that* $f(\zeta)$ *is analytic in the closure of* $\mathcal{D}$*, except for a finite number of simple poles* p *in the interior, all with the same value of* $P(p)$*.*

Outside the interval $[-1,1]$*, the curve* $\partial\mathcal{D}(p)$ *then separates the points, where the sequence* $\{(L_n f)(u)\}$ *converges, from the points, where it diverges. The behavior of* $|(L_n f)(u) - f(u)|$*, when* $u \in \mathcal{D}(v)$*,* $n \gg 1$ *is roughly described by the formula,*

$$|(f - L_n f)(u)| \approx K|\Phi_n(u)|e^{(-P(p)\pm\delta)n} / \max_p(1/|p-u|). \tag{4.8.13}$$

This can be further simplified, if u *is not in immediate vicinity of the interpolation points, see* (4.8.14)*.*

Proof. (Sketch:) At the application of the residue theorem to the integral $I_n(u)$, see (4.8.12), we must this time also consider the poles of $f(z)$. We obtain

$$\frac{I_n(u)}{\Phi_n(u)} = \frac{(f - L_n f)(u)}{\Phi_n(u)} + \sum_p \frac{\mathrm{res_f}(p)}{\Phi_n(p)(p-u)},$$

where $\mathrm{res_f}(p)$ is the residue of f at the pole $p \in \mathcal{D}(v)$. Roughly speaking, for $n \gg 1$,

$$I_n(u) = O\big(e^{-P(v)+\delta)n}\big), \qquad \sum_p = O\big(e^{(-P(p)\pm\delta)n} / \max_p(1/|p-u|), \quad P(v) > P(p).$$

It follows that $|I_n| \ll \sum$, unless there is a cancellation of terms in $\sum$. For fixed n, $\sum = 0$ is equivalent to an algebraic equation of degree less than the number of poles, and the roots will depend on n. We conclude that the case of cancellation can be ignored, and hence we obtain (4.8.13), and the following simplified version, valid if u is not in the immediate vicinity of the interpolation points. $|\Phi_n(u)|$ behaves like $\exp(P(u) \pm \delta)n$.

$$|(f - L_n f)(u)| \approx K e^{(P(u)-P(p)\pm\delta)n} / \max_p(1/|p-u|). \tag{4.8.14}$$

The separation statement follows from this. □

There are several interpolation processes with interpolation points in $[-1,1]$ that converge for all $u \in [-1,1]$, when the condition of analyticity is replaced by a more modest smoothness assumption, e.g., $f \in C^p$. This is the case, when the sequence of interpolation points are the zeros of the orthogonal polynomials which belong to a density function that is continuous and strictly positive in $]-1,1[$. We shall prove the following result.

Proposition 4.8.4.

Consider an interpolation process where the interpolation points has a (perhaps unknown) asymptotic density function $w(x)$*,* $x \in [-1,1]$*. Assume that*

$$(L_n f - f)(x) \to 0, \quad \forall x \in [-1,1], \quad \forall f \in C^k[-1,1],$$

as $n \to \infty$, for some $k \geq 1$. Then the logarithmic potential $P(x)$ must be constant in $[-1, 1]$, and the density function must be the same as for Chebyshev interpolation, i.e. $w(x) = \frac{1}{\pi}(1 - x^2)^{-1/2}$.

Proof. Let $f(z)$ be analytic in some neighborhood of $[-1, 1]$, e.g. any function with a pole at a point p (arbitrarily) close to this interval. A fortiori, for such a function our interpolation process must converge at all points u in some neighborhood of the interval $[-1, 1]$.

Suppose that $P(x)$ is not constant, and let x_1, x_2 be points, such that $P(x_1) < P(x_2)$. We can then choose the pole p so that $P(x_1) + \delta < P(p) < P(x_2) - \delta$. By Proposition 4.8.3, the process would then diverge at some points u arbitrarily close to x_2. This contradiction shows that $P(x)$ must be constant in $[-1, 1]$, $P(x) = a$, (say).

This gives a Dirichlet problem for the harmonic function $P(z)$, $z \notin [-1, 1]$, which has a unique solution, and one can verify that the harmonic function $P(z) = a + \Re \ln(z + \sqrt{z^2 - 1})$ satisfies the boundary condition. We must also determine a. This is done by means of the behaviour as $z \to \infty$. We find that

$$\begin{aligned} P(z) &= a + \Re \ln(z + z(1 - z^{-2})^{1/2}) \\ &= a + \Re \ln(2z - O(z^{-1}) = a + \Re \ln z + \ln 2 - O(z^{-2}). \end{aligned}$$

This is to be matched with the result of the discussion of the general logarithmic potential in the beginning of Sec. 4.8.2. In our case, where we have a symmetric distribution, and $\int_1^1 w(x)\,dx = 1$, we obtain $P(z) = \Re\psi(z) = \Re \ln z + O(z^{-2})$. The matching yields $a = -\ln 2$.

Finally, by (4.8.6), we obtain after some calculation, $w(x) = (1-x^2)^{-1/2}$. The details are left for Problem 3. □

Compare the above discussion with the derivations and results concerning the asymptotic distribution of the zeros of orthogonal polynomials, given in the standard monograph G. Szegö [54].

4.8.3 The Sampling Theorem

The ideas of this paper can be applied to other interpolation problems than polynomial interpolation. We shall apply them to a derivation of the celebrated sampling theorem which is an interpolation formula that expresses a function that is **band-limited** to the frequency interval $[-W,\ W]$, i.e. a function that has a Fourier representation of the following form (see also Strang [53, p. 325].

$$f(z) = \frac{1}{2\pi} \int_{-W}^{W} \hat{f}(k) e^{ikz}\,dk, \quad |\hat{f}(k)| \leq M, \tag{4.8.15}$$

in terms of its values at all integer points. The **Shannon Sampling Theorem** reads,

$$f(z) = \sum_{j=-\infty}^{\infty} f\left(\frac{j\pi}{W}\right) \frac{\sin(Wz - j\pi)}{(Wz - j\pi)}. \tag{4.8.16}$$

This is, like Lagrange's interpolation formula, a so-called *cardinal interpolation formula*. As Wz/π tends to an integer m, all terms except one on the right hand side become zero; for $j = m$ the term becomes $f(m\pi/W)$.

We shall sketch a derivation of this for $W = \pi$. We first note that (4.8.15) shows that $f(z)$ is analytic for all z. Then we consider the same Cauchy integral as many times before,

$$I_n(u) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}_n} \frac{\Phi(u)f(z)}{\Phi(z)(z-u)}\,dz, \quad u \in \mathcal{D}_n.$$

Here $\Phi(z) = \sin \pi z$, which vanishes at all integer points, and $\mathcal{D}_n$ is the *open* rectangle with vertices at $\pm(n + 1/2) \pm bi$. By the residue theorem, we obtain after a short calculation,

$$I_n(u) = f(u) + \sum_{j=-n}^{n} \frac{\Phi(u)f(j)}{\Phi'(j)(j-u)} = f(u) - \sum_{j=-n}^{n} \frac{f(j)\sin\pi(j-u)}{\pi(j-u)}.$$

Set $z = x + iy$. Note that

$$|f(z)| \le \frac{1}{2\pi}\int_{-\pi}^{\pi} Me^{-ky}dk \le \frac{M(e^{|\pi y|} - e^{-|\pi y|})}{|2\pi y|}, \quad |\Phi(z)| \ge e^{|\pi y|}.$$

These inequalities, applied for $y = b$, allow us to let $b \to \infty$; ($2b$ is the height of the symmetric rectangular contour). Then it can be shown that $I_n(u) \to 0$ as $n \to \infty$, which establishes the sampling theorem for $W = \pi$. The general result is then obtained by "regula de tri", but it is sometimes hard to get it right Strang [53] gives an entirely different derivation, based on Fourier analysis.

Problems and Computer Exercises

1. We use the notations and assumptions of Theorem 4.8.1. (a) Using the representation of the interpolation operator as an integral operator, show that

$$(L_n f)(x) = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} K(x,z)\frac{f(z)}{\Phi(z)}\,dz, \quad K(x,z) = \frac{\Phi(x) - \Phi(z)}{(x-z)},$$

also if $x \notin \mathcal{D}$. Note that $K(x, z)$ is a polynomial, symmetric in the two variables x, z.

(b) *A formula for the divided difference.* Show that

$$[x_1, x_2, \ldots, x_n]f = \frac{1}{2\pi i}\int_{\partial\mathcal{D}} \frac{f(z)}{\Phi(z)}\,dz.$$

Hint: Look at the leading term of the polynomial $(L_n f)(x)$.

2. Check the omitted details of the derivations in Sec. 4.8.3.

3. Check the validity of (4.8.6) on the Chebyshev and the equidistant cases. Also show that $\int_{-1}^{1} w(x)\,dx = 1$, and check the statements about the behaviour of $P(z)$ for $|z| \gg 1$.

4. (a) Work out the details of the proof of the Sampling Theorem.

(b) The formulation of the Sampling Theorem with a general W in Strang [53] does not agree with ours in (4.8.16). Who is right?

5. (a) Write a program for solving equations of the form $\psi(z) = c$, where c runs through a rectangular grid in a complex plane, not necessarily equidistant. You may assume that ψ is defined in such a way, that that its derivative is rather easily computed. When applicable, compute also the intersections of two families of level curves, i.e. with constant $\Re c$ and constant $\Im c$, with the real axis.

(b) Apply your program(s) to the plotting of these level curves in the two cases of the text, (related to, respectively, equidistant and Chebyshev interpolation). Due to the symmetry it is sufficient to draw the curves in a quarter-plane. Think of the "aspect" of the plotting so that the conformality of the mappings becomes visible.

If the scanning of the grid (of c) leads z to cross the forbidden interval $[-1, 1]$, or to some other exceptional situation, the program should return a nice message, and continue the scanning without interrupt, with more fruitful values of c, so that nothing is lost. By the way, find out, how the system you work with, handles the logarithm and square root in the complex domain. It may not be entirely according to our conventions, but it almost certainly produces some value that your own program can modify appropriately.

(c) If $\Re c \gg 1$, the level curve for the real part is, close to a circle (why?). Use equidistant values of $\Im c \in [0, \frac{1}{2}\pi]$. The values of $\Re c$ are to be chosen so that the drawings become intellectually interesting and/or visually pleasing. You are then likely to find that the density of the level curves for the imaginary part, when they approach the interval $[0, 1]$ is different for the Chebyshev case and the equidistant case. Explain theoretically how this is related to the density function w in the text.

(d) The level curves of the imaginary part intersect the interval $[0, 1]$, at different angles in the Chebyshev and the equidistant cases. Give a theoretical analysis of this.

6. (a) (After Meray (1884) and Cheney [11, p. 65]. Let $L_n f$ be the polynomial of degree $< n$, which interpolates to the function $f(z) = 1/z$ at the n'th roots of unity. Show that $(L_n f)(z) = z^{n-1}$, and that

$$\lim_{n\to\infty} \max_{|u|=1} |(L_n f - f)(u)| > 0.$$

Hint: Solve this directly, without the use of the previous theory.

(b) Modify the theory of Sec. 4.8.1 to the case in (a) with equidistant interpolation points on the unit circle, and make an application to $f(z) = 1/(z-a)$, $a > 0$, $a \neq 1$. Here, $\Phi_n(z) = z^n - 1$. What is $\psi(z)$, $P(z)$? The density function? (The integral for $\psi(z)$ is a little tricky, but you may find it in a

table. There are, however, simpler alternatives to the integral, see the end of Sec. 4.8.1. Check your result by thinking like Faraday.) Find out for which values of a, u, ($|u| \neq 1$, $|u| \neq a$), $(L_n f - f)(u) \to 0$, and estimate the speed of convergence (divergence).

(c) What can be said about the cases excluded above, i.e. $|u| = 1$, $|u| = a$? Also look at the case, when $|a| = 1$, ($a \neq 1$).

(d) Is the equidistant interpolation on the unit circle identical to the Cauchy FFT method (with $a = 0$, $R = 1$) for the approximate computation of the coefficients in a power series? See, in particular (3.1.10).

7. (a) We saw in 6 (b) that the equidistant interpolation on the unit circle gives no good polynomial approximation when the pole is inside the unit circle. The coefficients computed by the Cauchy FFT are however useful with a different interpretation, namely as coefficients in an interpolation polynomial $p(z^{-1})$ for $f(z) = 1/(z-a)$, or as approximate coefficients in a Laurent series $1/(z-a) = \sum_{j=1}^{\infty} c_j z^{-j}$, that converges for $|z > a|$. Note that the Cauchy integral, (3.1.8), is valid also for the coefficients of a Laurent expansion. Now consider

$$f(z) = \frac{-5}{(3-z)(1-2z)} = \frac{1}{3-z} - \frac{2}{1-2z}.$$

This has three Laurent expansions, i.e. an ordinary Taylor series for $|z| < \frac{1}{2}$, an expansion into negative powers for $|z| > 3$, and a mixed expansion for the annulus $\frac{1}{2} < |z| < 3$. It is conceivable that the the first two expansions can be found by FFT, with different interpretations of the results, but what about the annulus case? It is easily seen from the above partial fraction form of $f(z)$ what the Laurent expansion should be. When the FFT is applied to $f(z)$, it does therefore, in principle, find a coefficient by adding a coefficient of a *negative* power of z from the first term of the partial fraction decomposition, to the coefficient of a *positive* power of z from the second term. *Can this really work?*

Explain, why things go so well with a careful treatment, in spite that we almost tried to convince you above that it would not work. Also try to formulate what "careful treatment" means in this case.

Hint: Generalize to the case of a Laurent expansion the relation between the FFT output and the series coefficients given (for a Taylor series) in (3.1.11). Also read in Strang [53, Chapter 4] or somewhere else about "aliasing".

Notes and Further Reading

The problem of choosing a good orderings of points in Newton and Lagrange interpolations is discussed in [57]. Newton interpolation using the Leja ordering of points has been analyzed by Reichel [40]. The barycentric form of Lagrange's interpolation formula was advocated in lecture notes by Rutishauser [43] already in the 1960's. Berrut and Trefethen [3] argue convincingly that this should be the standard method of polynomial interpolation, and in historical notes discuss why

it is not better known. The scheme for computing the inverse of a Vandermonde matrix is due to Higham [31, Sec. 22.1].

The $O(n^2)$ algorithm for solving primal Vandermonde systems decribed in Sec. 4.3.4 is due to Björck and Pereyra [4]. It has been generalized to yield fast algorithm for Vandermonde-like matrices defined by $V = (v_{ij}) = ((p_i(x_j))$, where p_i is a polynomial of degree n that satisfies a thre term recurrence relation; see Higham [31, Sec. 22.2]. Also so called Cauchy linear systems can be solved with a Björck–Pereyra-type algorithm; see Boros, Kailath and Olshevsky [8].

The computational advantage of the Stieltjes approach for discrete least squares fitting was pointed out by Forsythe [24, 1956]. Shampine [48, 1975] established the advantage in using the alternative formula involving the residual r_k.

Working for the French car companies Renault and Citroën, Bézier and de Casteljau, independently in 1962 developed the Bézier curve as a tool in Computer Aided Design (CAD) for fitting curves and surfaces. A more geometric view of spline functions is taken in Farin [23]. Several packages are available for computing with splines, e.g., the spline toolbox in MATLAB and FITPACK Dierckx [20]–[21].

The FFT algorithm algorithm has been discovered independently by several people. Indeed the idea was published in a paper by Gauss and the doubling algorithm is contained in a textbook by Runge and König [42]. The modern usage of FFT started in 1965 with the publication of the papers [15, 14] by James W. Cooley of IBM Research and John W. Tukey, Princeton University. The re-discovery of the FFT algorithm is surveyed by James W. Cooley in [13]. Applications are surveyed in [10] and [9]. The matrix-oriented framework for the FFT used in this book is developed in [56]). A roundoff error analysis is given in [1].) Algorithms for the bit-reversal permutation are reviewed in [34].

Ideas related to those in Sec. 4.8.2 were applied in the thesis of Lothar Reichel at KTH. He studied the Helmholtz equation in 2D, with a regionally constant complex coefficient, with potential applications (excuse our pun!), e.g., to the microwave heating of cheeseburgers. Since one can find nice bases of particular solutions of the Helmholtz equation in different regions, i.e. bread, meat and cheese, one may try **boundary collocation**, to express the appropriate continuity conditions at the interfaces between meat and cheese etc. in a finite number of points.

Bibliography

[1] Mario Arioli, Hans Z. Munthe-Kaas, and L. Valdettaro. Componentwise error analysis for FFTs with applications to fast Helmholz solvers. *Numer. Algorithms*, 12:65–88, 1996.

[2] R. W. Barnard, Germund Dahlquist, K. Pearce, Lothar Reichel, and K. C. Richards. Gram polynomials and the Kummer function,. *J. Approx. Theory*, 94:128–143, 1998.

[3] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric Lagrange interpolation. *SIAM Review*, 46:3:501–517, 2004.

[4] Åke Björck and Victor Pereyra. Solution of Vandermonde system of equations. *Math. of Comp.*, 24:893–903, 1970.

[5] P. Bloomfield. *Fourier Analysis and Time Series*. John Wiley, New York, 1976.

[6] Carl de Boor. On calculating with B-splines. *J. Approx. Theory*, 6:50–62, 1972.

[7] Carl de Boor. *A Practical Guide to Splines*. Springer-Verlag, Berlin, revised edition, 1991.

[8] Tibor Boros, Thomas Kailath, and Vadim Olshevsky. A fast parallel Björck–Pereyra-type algorithm for solving Cauchy linear systems. *Linear Algebra Appl.*, 302–303:265–293, 1999.

[9] W. L. Briggs and Van Emden Henson. *The DFT. An Owners Manual for the Discrete Fourier Transform*. SIAM, Philadelphia, PA, 1995.

[10] E. O. Brigham. *The Fast Fourier Transform and Its Application*. Prentice-Hall, Englewood Cliffs, NJ, 1988.

[11] E. W. Cheney. *Introduction to Approximation Theory*. McGraw-Hill, New York, NY, 1966.

[12] E. W. Cheney and W. Light. *A Course in Approximation Theory*. Brooks/Cole, Pacific Grove, CA, 2000.

[13] James W. Cooley. The re-discovery of the fast Fourier transform algorithm. *Mikrochimica Acta,*, 3:33–45, 1987.

[14] James W. Cooley, Peter A. W Lewis, and Peter D. Welsh. The fast Fourier transform and its application. *IEEE Trans. Education,*, E–12:27–34, 1969.

[15] James W. Cooley and John W. Tukey. An algorithm for machine calculation of complex Fourier series. *Math. Comp.,*, 19:297–301, 1965.

[16] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume I. Interscience, New York, 1953.

[17] Maurice G. Cox. The numerical evaluation of B-splines. *J. Inst. Math. Appl.*, 10:134–149, 1972.

[18] G. Danielson and Cornelius Lanczos. Some improvements in practical Fourier analysis and their applications to x-ray scattering from liquids. *J. Franklin Inst.*, 233:365–380, 435–452, 1942.

[19] Philip J. Davis. *Interpolation and Approximation*. Dover, New York, NY, 1975.

[20] P. Dierckx. FITPACK user guide part i: Curve fitting routines. TW Report 89, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, 1983.

[21] P. Dierckx. FITPACK user guide part i: Surface fitting routines. TW Report 122, Department of Computer Science, Katholieke Universiteit, Leuven, Belgium, 1983.

[22] P. Dierckx. *Curve and Surface Fitting with Splines*. Clarendon Press, New York, 1993.

[23] Gerald Farin. *Curves and Surfaces for Computer Aided Geometric Design: A Practical Guide*. Academic Press, New York, 1988.

[24] George E. Forsythe. Generation and use of orthogonal polynomials for data-fitting with a digital computer,. *J. Soc. Indust. Appl. Math.*, 5:74–88, 1957.

[25] Walter Gander. Change of basis in polynomial interpolation. *Numer. Linear Algebra Appl.*, page submitted, 2004.

[26] Walter Gautschi. *Numerical Analysis, an Introduction*. Birkhäuser, Boston, MA, 1997.

[27] W. M. Gentleman and G. Sande. Fast Fourier transforms—for fun and profit. In *Proceedings AFIPS 1966 Fall Joint Computer Conference*, pages 503–578. Spartan Books, Washington, D.C., 1966.

[28] Ernst Hairer and Gerhard Wanner. *Analysis by Its History*. Springer Verlag, Berlin, third corrected printing edition, 2000.

[29] Peter Henrici. *Essentials of Numerical Analysis*. John Wiley, New York, 1982.

[30] Nicholas J. Higham. Error asnalysis of the Björck–Pereyra algorithm for solving Vandermonde systems. *Numer. Math.*, 50:613–632, 1987.

[31] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[32] Nicholas J. Higham. The numerical stability of barycentric Lagrange interpolation. *IMA J. Numer. Anal.*, 24:547–556, 2004.

[33] F. B. Hildebrand. *Introduction to Numerical Analysis*. McGraw-Hill, New York, 1974.

[34] Alan H. Karp. Bit reversal on uniprocessors. *SIAM Review*, 38:1:1–26, 1996.

[35] Fred T. Krogh. A variable step variable order multistep method for the numerical solution of ordinary differential equations. In A. J. Morell, editor, *Proceedings of the IFIP Congress 1968*, pages 194–199. North-Holland, Amsterdam, 1969.

[36] J. G. Mason and D. C. Handscomb. *Chebyshev Polynomials*. Chapman & Hall/CRC, London, 2003.

[37] Cleve Moler and Steve Eddins. Fast finite Fourier transforms. *MATLAB News and Notes*, pages 14–15, Winter, 2001.

[38] M. J. D. Powell. *Approximation Theory and Methods*. Cambridge University Press, Cambridge, UK, 1981.

[39] William H. Press, Brian P. Flannery, Saul A. Teukolsky, and William T. Vetterling. *Numerical Recipes in Fortran; The Art of Scientific Computing*. Cambridge University Press, Cambridge, GB, second edition, 1992.

[40] Lothar Reichel. Newton interpolation at Leja points. *BIT*, 30:332–346, 1990.

[41] Friedrich Riesz and Béla Sz.-Nagy. *Vorlesungen Über Funktionalanalysis*. VEB Deutscher Verlag der Wissenschaften, Berlin, 1956.

[42] Carle Runge and H. König. *Vorlesungen über Numerisches Rechnen. Band XI*. Verlag Julius Springer, Berlin, 1924.

[43] Heinz Rutishauser. *Vorlesungen über numerische Mathematik, Vol. I*. Birkhäuser, Basel–Stuttgart, 1976. English translation *Lectures on Numerical Mathematics*, by W. Gautschi, Birkhäuser, Boston, 1990.

[44] I. J. Schoenberg. Contributions to the problem of approximation of equidistant data by analytic functions. *Quart. Appl. Math.*, 4:45–99 and 112–141, 1946.

[45] I. J. Schoenberg and A. Whitney. On Pólya frequency functions III: The positivity of translation determinants with an application to the interpolation problem by spline curves. *Trans. Amer. Math. Soc.*, 74:246–259, 1953.

[46] H. R. Schwarz. *Numerische Methematik*. Teubner, Stuttgart, fourth edition, 1997. English translation of 2nd ed.: Numerical Analysis: A Comprehensive Introduction, John Wiley, New York.

[47] Hubert Schwetlick and Torsten Schütze. Least squares approximation by splines with free knots. *BIT*, 35:361–384, 1995.

[48] Lawrence F. Shampine. Discrete least squares polynomial fits. *Comm. ACM*, 18:179–180, 1975.

[49] R. C. Singleton. On computing the fast Fourier transform. *Comm. ACM*, 10:647–654, 1967.

[50] R. C. Singleton. Algorithm 338: Algol procedure for the fast Fourier transform with arbitrary factors. *Comm. ACM*, 11:773–779, 1968.

[51] J. F. Steffensen. *Interpolation.* Chelsea, New York, second edition, 1950.

[52] Joseph Stoer and Roland Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, third edition, 2002.

[53] Gilbert Strang. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, MA, 1986.

[54] Gabor Szegö. *Orthogonal Polynomials*, volume 23 of *Colloq. Publ.* Amer. Math. Soc., Providence, RI, fourth edition, 1975.

[55] T. N. Thiele. *Interpolationsrechnung.* B. G. Teubner, Leipzig, 1909.

[56] Charles F. Van Loan. *Computational Framework for the Fast Fourier Transform.* SIAM, Philadelphia, 1992.

[57] Wilhelm Werner. Polynomial interpolation: Lagrange versus Newton. *Math. Comp.*, 43:205–217, 1984.

[58] M. Zelen. Linear estimation and related topics. In John Todd, editor, *Survey of Numerical Analysis*, pages 558–584. McGraw-Hill, New York, 1962.

Index